EINDHOVEN UNIVERSITY OF TECHNOLOGY

Eindhoven University of Technology

Report of the Graduation Project

Project phase

Implementation of a controller for an autonomously flying quad-copter using Optitrack.

Master:	Automotive Technology
Department:	Mechenical Engineering
Research group:	Dynamics and Control

Student:	B.J.F. te Boekhorst
ID number:	0950789
Thesis supervisor:	E. Lefeber
Date:	May 11, 2023
Report number:	DC 2023.011

Contents

Nomenclature

The next list describes several symbols that are used within the body of the document

ν	Linear velocity with respect to body-fixed frame
ω	Angular velocity with respect to body-fixed frame
ϕ	Roll angle of a body
ψ	Yaw angle of a body
ρ	Position vector
τ	Torque generated by the rotors
θ	Pitch angle of a body
e_i	Unit vector i of a frame of reference
f	Total force generated by the motors
g	gravitational acceleration constant
I_n	Identity matrix of size n times n
J	Inertia of a body
m	Mass of a body
R	Rotation matrix of a body
S	Function providing skew symmetric matrix of its vector argument

1 Background and Problem definition

1.1 Background

Drones, also called UAVs (Unmanned Aerial Vehicle) are being used for decades now. They are seen in many different areas and professions. Starting in the military in the early stages of development, UAVs were mainly used to perform attacks without needing a pilot. Nowadays drones have a more widespread use case, though they are still used in the military for missions where an operator might be at risk or for reconnaissance. Furthermore, drones are used in the movie industry, farming and even recreational use for civilians [1].

Within the category of UAVs, there are a lot of different types of drones. There exist airplane-like drones, quadrotor drones and even drones with more than four rotors. An advantage of a multi-rotor drone is that vertical take off and landings are possible. Another advantage is that hovering in a specific location in 3D space can be done, which is something that is not possible for airplane-like drones. Within the multi-rotor types of drone, the quadrotor is the most common, mostly because of production costs. Since a quadrotor has four propellers, not all 6 degrees of freedom (DOF) can be controlled directly. For example, first a rotation around one of the horizontal axes has to be made before a movement in the direction of the other horizontal axis can be achieved.

Currently drones are usually controlled with a remote controller. One of the main challenges in this field is to have the drone operate autonomously. At Eindhoven University of Technology, research is done by a dedicated part of the Dynamics and Control research group on a relatively cheap drone to make it fly autonomously. So far, experiments have been done with the Parrot AR.Drone 2.0 and the Parrot Mambo fly drones [2], [3]. Since the Parrot AR.Drone 2.0 is no longer in production, further research is done on the Parrot Mambo Fly. A detailed comparison between the two drones can be found in [3].

The Parrot Mambo fly uses an optical flow sensor in combination with a gyroscope and an accelerometer to determine its position. This works fine in most cases, however when flying autonomously it can be the cause of problems. The way the optical flow sensor works can give the wrong location output to the controller of the drone, resulting in a deviation from the desired path. One way to solve this sensor issue is to use a different method of localizing the drone.

One method is to use markers on the floor for the drone to recognise and determine its location based on the markers [4]. According to [4], this method is cheaper to implement than using a motion capture system, is more robust to the environment of the experiments but needs intensive image processing. Fortunately a motion capture system is available in the robotics lab at the Technical University of Eindhoven. This motion capture system is accurate up to 1mm, which is beneficial for the performance of controllers implemented on the UAV. Another solution is to use a radar system to localize the drone [5]. It is shown in [5] that radar can be used to localize a drone in 3D space with an accuracy of circa 9cm. With the Parrot Mambo drone being only 18cm in size itself, this is a substantial error and thus motion capture is a better alternative to locate the drone.

The drone can also be localized using a camera system [6]. It is shown in [6] that a camera system to localize the drone works better than the internal sensors of the drone itself, a Parrot AR Drone 2.0. This is the same drone used by Brekelmans in his comparison to the Parrot Mambo fly [3]. The camera system used in [6] is similar to the Optitrack system present at Eindhoven University of Technology [7]. Most commonly the markers designed and made by Optitrack are used to locate specific objects. These markers can be attached to the Mambo drone and the camera system is then able to localize the markers.

The Parrot Mambo fly drone is also used in [9] and [10]. The Mambo fly drone is used in [9] to implement the controller designed in [11]. It is concluded in [9] that the optical flow estimator drifts continuously for horizontal velocity and most of the drift is caused by the accelerometer. The controller is implemented successfully in simulations but not in the final implementation, due to the drift mentioned before. A better implementation result could be realized by using a better state estimation.
The Mambo fly drone is used in [10] with a focus on collision avoidance of multiple UAVs. One of the observations in [10] is that, since the algorithm considers the drones as point masses and no drone exists as point mass, collisions in practice could still occur. This could be avoided by using a very precise localization method.

One of the main advantages of using the Parrot Mambo is that a Simulink add-on package can be used. This package focuses specifically on Parrot drones and uses the aerospace toolbox as a basis. The Parrot package is a very useful starting point to communicate and control the drone. It is used successfully by [2], [3], [9] and [10], confirming that it can potentially be very useful to complete the challenge set in this thesis.

1.2 Problem definition and research question

This thesis focuses mainly on the implementation of a controller on the Parrot Mambo fly drone, using an Optitrack camera system to localize the drone. Future research as well as applications described in section 1.1 will benefit from very accurate localization of the drone. The controller that is implemented is the one described in [11]. This goal can be achieved by completing multiple smaller goals, consisting of

- Describing the behaviour of the Parrot Mambo fly drone.
- Determine a reference trajectory for the drone to follow in detail.
- Implement the controller in simulations.
- Achieving a connection between the Parrot Mambo fly drone and the computer used for this research.
- Achieve a connection with the Optitrack camera system to localize the drone.
- Implement the location output data within the Matlab environment to use as input for the controller.
- Achieve flight using the Simulink support package.

The outline of this thesis is as follows. Firstly in Chapter 2 a model is presented to describe the dynamics of a drone and the reference trajectory. Three different trajectories are chosen from which the rest of the reference dynamics are derived. Chapter 3 then describes the controller that is considered in this thesis. This controller is then tested in simulations in chapter 4, together with the reference trajectories determined in chapter 2. Chapter 5 explains more about the context of the environment the tests are done and gives a visualization of the camera system setup. Chapter 6 explains the main functionalities of the different system blocks within the Simulink support package for parrot mini drones. Chapter 7 focuses on the Parrot mini drone used in this research, specifically how a connection is made to the drone. Chapter 8 describes how the output of the camera system can be imported to Matlab in Windows and how a connection can be made in Ubuntu. Chapter 9 then concludes the thesis and gives recommendations for future research. The structure of this thesis is such that the individual chapters can be used independently of one another and still be relevant. Such that if a different drone is used, the chapters about the reference, controller and camera system still apply.

2 Dynamics and reference trajectory

The first step towards achieving the goals discussed before is to describe the behaviour of the quad-copter drone. The general dynamic representation of a drone is used as a base for this model. This chapter focuses on the dynamics of a drone, what flat output signals are used and how the reference dynamics are derived from these flat output signals. The reference trajectory is described in enough detail that it can be copied for simulations without the need for further calculations. After the reference trajectory is determined, a controller can be designed and then tested in simulations to see if the reference is followed.

2.1 Drone dynamics

The dynamics of the drone can be described as

$$\begin{aligned} \dot{\rho} &= R\nu, \\ \dot{\nu} &= -S(\omega)\nu + gR^T e_3 - \frac{f}{m}e_3, \\ \dot{R} &= RS(\omega), \\ J\dot{\omega} &= S(J\omega)\omega + \tau, \end{aligned} \tag{2.1}$$

where ρ is the position vector, ν is the linear velocity, R is the rotation matrix, ω is the angular velocity, g is the gravitational constant, f is the total thrust magnitude, m is the mass of the drone, S is a function providing a screw symmetric matrix, J is the inertia matrix and τ is the total moment vector in the body fixed frame. In this case, f and τ are considered as inputs.

These equations are the base of the reference trajectory, making the reference feasible to be followed by the drone. The dynamics have six degrees of freedom, of which four can be chosen independently as a function of time, these four are the flat output signals. The remaining two degrees of freedom are a function of the flat output signals. The detailed relation between these signals is described in a later chapter. To specify a reference trajectory, it is sufficient to only choose the flat output signals as

$$\begin{aligned} \begin{bmatrix} \gamma_1(t) \\ \gamma_2(t) \\ \gamma_3(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \rho(t) \\ \gamma_4(t) = \psi(t). \end{aligned} \tag{2.2}$$

The reference trajectory has to adhere to the same dynamics, to make the path feasible to follow. Giving the following expression for these signals

$$\begin{aligned} \begin{bmatrix} \gamma_{r1}(t) \\ \gamma_{r2}(t) \\ \gamma_{r3}(t) \end{bmatrix} = \begin{bmatrix} x_r(t) \\ y_r(t) \\ z_r(t) \end{bmatrix} = \rho_r(t) \\ \gamma_{r4}(t) = \psi_r(t). \end{aligned} \tag{2.3}$$

Once the components of $\gamma_r(t)$ have been chosen, the rest of the reference dynamics follow from these flat output signals.

2.2 Chosen reference

To determine the full reference dynamics, the reference trajectory needs to be chosen. These trajectories and how they are derived is discussed in this section. The reference trajectories become increasingly challenging. Starting at simply hovering, advancing to a back and forth movement and ending with a manoeuvre done by Lefeber et al. [11]. For the reference trajectory, the different components of $\gamma_r(t)$ have to be chosen. Namely $x_r(t)$, $y_r(t)$, $z_r(t)$ and $\psi_r(t)$. The fourth time derivative of the chosen values of $\gamma_r(t)$ are needed, except for $\psi(t)_r$, where only the first and second derivative are required. The rest of the reference dynamics can be derived from these four components of $\gamma_r(t)$. In this section, the third and fourth derivative are indicated in the form of $x_r^{(i)}$. Where $x_r^{(i)}$is the i-th order time derivative of x_r.

The simplest reference chosen is a hovering reference. The corresponding values chosen for $\gamma_r(t)$ are

$$x_r(t) = 0, \quad y_r(t) = 0, \quad z_r(t) = -1, \quad \psi_r(t) = 0. \tag{2.4}$$

Here the z coordinate is negative, because of the chosen orientation of the coordinate system (NED). For this reference all the derivatives do not need to be calculated, since they are all equal to zero.

The more challenging reference, the back and forth movement, the values for $\gamma(t)$ are chosen as

$$x_r(t) = a_x \sin(b_x t), \quad y_r(t) = a_y \sin(b_y t), \quad z_r(t) = a_z \sin(b_z t) - 1, \quad \psi_r(t) = a_\psi \sin(b_\psi t), \tag{2.5}$$

where a and b can be chosen to vary the amplitude and frequency of the reference sine wave. The derivatives for $x_r(t)$, $y_r(t)$, $z_r(t)$ and $\psi_r(t)$ are given below.

$$\begin{aligned}
&\dot{x}_r(t) = a_x b_x \cos(b_x t), \quad \ddot{x}_r(t) = -a_x b_x^2 \sin(b_x t), \quad x_r^{(3)}(t) = -a_x b_x^3 \cos(b_x t), \quad x_r^{(4)}(t) = a_x b_x^4 \sin(b_x t),\\
&\dot{y}_r(t) = a_y b_y \cos(b_y t), \quad \ddot{y}_r(t) = -a_y b_y^2 \sin(b_y t), \quad y_r^{(3)}(t) = -a_y b_y^3 \cos(b_y t), \quad y_r^{(4)}(t) = a_y b_y^4 \sin(b_y t),\\
&\dot{z}_r(t) = a_z b_z \cos(b_z t), \quad \ddot{z}_r(t) = -a_z b_z^2 \sin(b_z t), \quad z_r^{(3)}(t) = -a_z b_z^3 \cos(b_z t), \quad z_r^{(4)}(t) = a_z b_z^4 \sin(b_z t),\\
&\dot{\psi}_r(t) = a_\psi b_\psi \cos(b_\psi t), \quad \ddot{\psi}_r(t) = -a_\psi b_\psi^2 \sin(b_\psi t),
\end{aligned} \tag{2.6}$$

where a and b signify amplitude and frequency respectively. Both a and b are assumed to be constants. The values for these variables in this research are

$$\begin{aligned}
a_x = 0.7, \quad a_y = 0.5, \quad a_z = 0, \quad a_\psi = 0,\\
b_x = 0.5, \quad b_y = 1.0, \quad b_z = 0, \quad b_\psi = 0.
\end{aligned} \tag{2.7}$$

Because this reference is harder to visualize than the hover reference, it is shown in Figure 2.1.

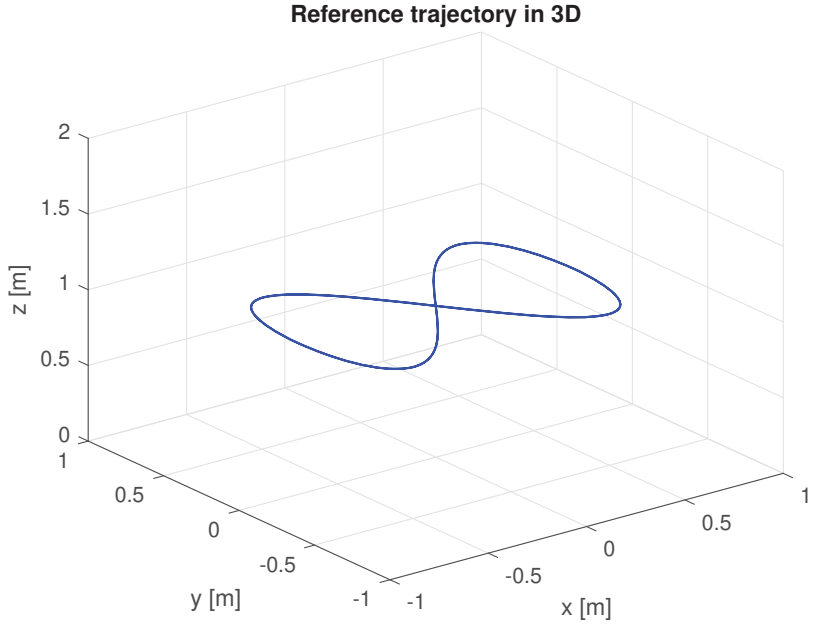

Figure 2.1: Reference trajectory in 3D for the back and forth movement.

The final and most challenging reference trajectory is based on the research done by Lefeber et al [11]. The different components that correspond to this trajectory are

$$x_r(t) = (6 + 2\cos(\omega_\nu t))\cos(\omega_u t), \quad y_r(t) = (6 + 2\cos(\omega_\nu t))\sin(\omega_u t), \quad z_r(t) = -2\sin(\omega_\nu t), \quad \psi_r(t) = \omega_u t + \pi, \tag{2.8}$$

where ω_ν and ω_u are related to the amplitude and frequency. In this case they are chosen to be $\omega_\nu = 1.2\pi$ (rad/s) and $\omega_u = 0.2\pi$ (rad/s) respectively. The corresponding derivatives are described below.

$$\dot{x}_r(t) = -2\omega_\nu \sin(\omega_\nu t)\cos(\omega_u t) - \omega_u (6 + 2\cos(\omega_\nu t))\sin(\omega_u t) \tag{2.9a}$$

$$\ddot{x}_r(t) = -2\omega_\nu^2 \cos(\omega_\nu t)\cos(\omega_u t) - \omega_u^2 (6 + 2\cos(\omega_\nu t))\cos(\omega_u t) + 4\omega_\nu\omega_u \sin(\omega_\nu t)\sin(\omega_u t) \tag{2.9b}$$

$$\begin{aligned} x_r^{(3)}(t) =& 2\omega_\nu^3 \sin(\omega_\nu t)\cos(\omega_u t) + \omega_u^3 (6 + 2\cos(\omega_\nu t))\sin(\omega_u t) + 6\omega_\nu^2\omega_u \cos(\omega_\nu t)\sin(\omega_u t) \\ &+ 6\omega_u^2\omega_\nu \sin(\omega_\nu t)\cos(\omega_u t) \end{aligned} \tag{2.9c}$$

$$\begin{aligned} x_r^{(4)}(t) =& 2\omega_\nu^4 \cos(\omega_\nu t)\cos(\omega_u t) + \omega_u^4 (6 + 2\cos(\omega_\nu t))\cos(\omega_u t) - 8\omega_\nu^3\omega_u \sin(\omega_\nu t)\sin(\omega_u t) \\ &- 8\omega_u^3\omega_\nu \sin(\omega_\nu t)\sin(\omega_u t) + 12\omega_\nu^2\omega_u^2 \cos(\omega_\nu t)\cos(\omega_u t) \end{aligned} \tag{2.9d}$$

$$\dot{y}_r(t) = -2\omega_\nu \sin(\omega_\nu t)\sin(\omega_u t) + \omega_u (6 + 2\cos(\omega_\nu t))\cos(\omega_u t) \tag{2.10a}$$

$$\ddot{y}_r(t) = -2\omega_\nu^2 \cos(\omega_\nu t)\sin(\omega_u t) - \omega_u^2 (6 + 2\cos(\omega_\nu t))\sin(\omega_u t) - 4\omega_\nu\omega_u \sin(\omega_\nu t)\cos(\omega_u t) \tag{2.10b}$$

$$\begin{aligned} y_r^{(3)}(t) =& 2\omega_\nu^3 \sin(\omega_\nu t)\sin(\omega_u t) - \omega_u^3 (6 + 2\cos(\omega_\nu t))\cos(\omega_u t) - 6\omega_\nu^2\omega_u \cos(\omega_\nu t)\cos(\omega_u t) \\ &+ 6\omega_u^2\omega_\nu \sin(\omega_\nu t)\sin(\omega_u t) \end{aligned} \tag{2.10c}$$

$$\begin{aligned} y_r^{(4)}(t) =& 2\omega_\nu^4 \cos(\omega_\nu t)\sin(\omega_u t) + \omega_u^4 (6 + 2\cos(\omega_\nu t))\sin(\omega_u t) + 8\omega_\nu^3\omega_u \sin(\omega_\nu t)\cos(\omega_u t) \\ &+ 8\omega_u^3\omega_\nu \sin(\omega_\nu t)\cos(\omega_u t) + 12\omega_\nu^2\omega_u^2 \cos(\omega_\nu t)\sin(\omega_u t) \end{aligned} \tag{2.10d}$$

$$\dot{z}_r(t) = 2\omega_\nu \cos(\omega_\nu t), \quad \ddot{z}_r(t) = -2\omega_\nu^2 \sin(\omega_\nu t), \quad z_r^{(3)}(t) = -2\omega_\nu^3 \cos(\omega_\nu t), \quad z_r^{(4)}(t) = 2\omega_\nu^4 \sin(\omega_\nu t) \tag{2.11}$$

$$\dot{\psi}_r(t) = \omega_u, \quad \ddot{\psi}_r(t) = 0. \tag{2.12}$$

To visualize this reference trajectory it is shown in Figure 2.2.

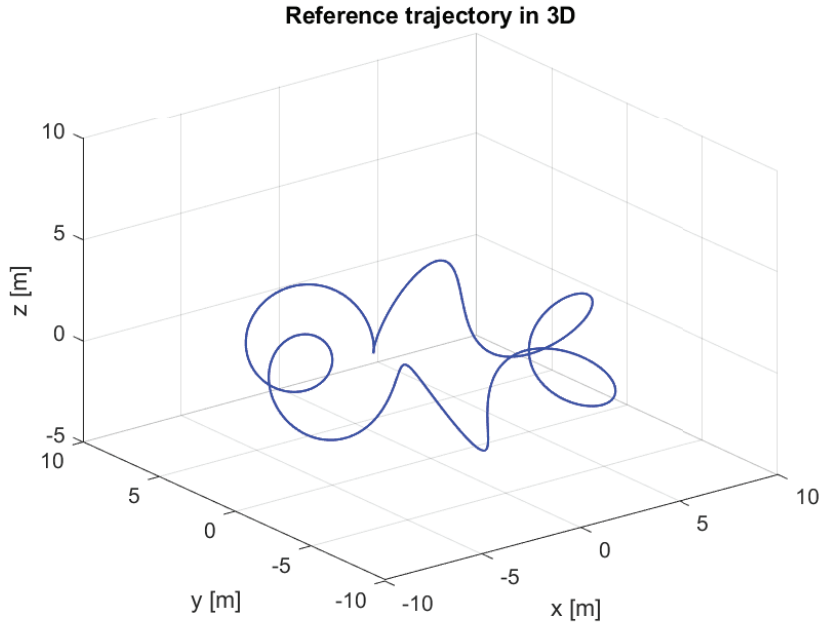

Figure 2.2: Reference trajectory in 3D for the reference from [11].

The components of γ_r have now been defined and chosen. All of the reference dynamics follow from the reference trajectory. Thus the next step is to derive the full reference dynamics.

2.3 Reference dynamics

This section shows how the full reference dynamics are derived from the dynamics of the drone and follow from the flat output signals $\gamma_r(t)$. Firstly the derivations and how the different expressions follow from one another are explained. After which the derivation is shown. The notation within this derivation is kept without the expressions in their complete length, for the sake of simplicity. A detailed derivation is included in appendix A.

The reference dynamics are similar to the actual dynamics and can be described as

$$\begin{aligned} \dot{\rho}_r &= R_r \nu_r, \\ \dot{\nu}_r &= -S(\omega_r)\nu_r + g R_r^T e_3 - \frac{f_r}{m} e_3, \\ \dot{R}_r &= R_r S(\omega_r), \\ J\dot{\omega}_r &= S(J\omega_r)\omega_r + \tau_r. \end{aligned} \tag{2.13}$$

The value of ρ_r is already known, since it was defined in section 2.2. The reference force of the rotors f_r is obtained from the second time derivative of the position vector ρ_r. The second derivative of the position vector is also used to determine the third column of the rotation matrix R_r. The rotation matrix R_r can be obtained by multiplying the rotations for roll pitch and yaw as $R_r = R_Z(\psi_r)R_Y(\theta_r)R_X(\phi_r)$.
This combined with the third column derived before, gives the expressions for ϕ_r and θ_r expressed in terms of ψ_r. With both ρ_r and R_r known, the reference velocity can be derived from equation (2.13).
The next step is to derive an expression for ω_r, which can be obtained from $S(\omega_r)$. To do this, the derivative of the rotation matrix R_r is required.
This derivative is calculated using the expression for the rotation matrix derived before.
To get the expression for τ_r, the derivative of ω_r needs to be calculated, which requires the second derivative of R_r. With all these expressions derived, an expression for τ_r is made.
This would then complete the whole reference trajectory and a controller can be designed to follow this reference.

The thrust of the reference f_r can be obtained from the second time derivative of ρ_r as follows,

$$\ddot{\rho}_r = \dot{R}_r \nu_r + R_r \dot{\nu}_r = g e_3 - (f_r/m)\, R_r e_3. \tag{2.14}$$

Therefore,

$$f_r = m \left\| g e_3 - \ddot{\rho}_r \right\| = m\sqrt{\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2}. \tag{2.15}$$

The first and second derivative of the thrust can then be written as

$$\begin{aligned} \dot{f}_r &= -m \frac{(g e_3 - \ddot{\rho}_r)^T \rho_r^{(3)}}{\| g e_3 - \ddot{\rho}_r \|} \\ \ddot{f}_r &= m \frac{\left(\rho_r^{(3)}\right)^T \rho_r^{(3)}}{\| g e_3 - \ddot{\rho}_r \|} - m \frac{(g e_3 - \ddot{\rho}_r)^T \rho_r^{(4)}}{\| g e_3 - \ddot{\rho}_r \|} - \frac{\dot{f}_r^2}{\| g e_3 - \ddot{\rho}_r \|}, \end{aligned} \tag{2.16}$$

where $\rho_r^{(i)}$ is the i-th order time derivative of ρ_r.

The last column of R_r can be derived from (2.15) using the second derivative of ρ_r as follows

$$R_r e_3 = \frac{m}{f_r}(g e_3 - \ddot{\rho}_r) = \frac{1}{\sqrt{\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2}} \begin{bmatrix} -\ddot{x}_r \\ -\ddot{y}_r \\ g - \ddot{z}_r \end{bmatrix} := \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} R_{r(1,3)} \\ R_{r(2,3)} \\ R_{r(3,3)} \end{bmatrix}, \tag{2.17}$$

provided that $\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2 \neq 0$.

The rotation matrix is determined as,

$$R_r = R_Z(\psi_r)R_Y(\theta_r)R_X(\phi_r) = \begin{bmatrix} \cos\psi_r & -\sin\psi_r & 0 \\ \sin\psi_r & \cos\psi_r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_r & 0 & \sin\theta_r \\ 0 & 1 & 0 \\ -\sin\theta_r & 0 & \cos\theta_r \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_r & -\sin\phi_r \\ 0 & \sin\phi_r & \cos\phi_r \end{bmatrix}, \tag{2.18}$$

which results in

$$R_r = \begin{bmatrix} \cos\psi_r\cos\theta_r & \cos\psi_r\sin\phi_r\sin\theta_r - \cos\phi_r\sin\psi_r & \sin\phi_r\sin\psi_r + \cos\phi_r\cos\psi_r\sin\theta_r \\ \cos\theta_r\sin\psi_r & \cos\phi_r\cos\psi_r + \sin\phi_r\sin\psi_r\sin\theta_r & \cos\phi_r\sin\psi_r\sin\theta_r - \cos\psi_r\sin\phi_r \\ -\sin\theta_r & \cos\theta_r\sin\phi_r & \cos\phi_r\cos\theta_r \end{bmatrix}. \tag{2.19}$$

Using (2.17) and (2.19), the values of ϕ_r and θ_r can be expressed in terms of ψ_r:

$$\begin{aligned} \sin\phi_r &= r_1\sin\psi_r - r_2\cos\psi_r, \\ \cos\phi_r &= \sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}, \\ \cos\theta_r &= \frac{r_3}{\sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}}, \\ \sin\theta_r &= \frac{r_1\cos\psi_r + r_2\sin\psi_r}{\sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}}. \end{aligned} \tag{2.20}$$

These new expressions can be substituted into the rotation matrix, but for the sake of simplicity, this substitution is not done in detail here. The detailed derivation can be found in Appendix A.

Next, the position vector and the rotation matrix are used to construct the reference velocity as

$$\nu_r = R_r^T\dot{\rho}_r. \tag{2.21}$$

The derivative of the rotation matrix is calculated and is written as

$$\dot{R}_r = \begin{bmatrix} \dot{R}_{r(1,1)} & \dot{R}_{r(1,2)} & \dot{R}_{r(1,3)} \\ \dot{R}_{r(2,1)} & \dot{R}_{r(2,2)} & \dot{R}_{r(2,3)} \\ \dot{R}_{r(3,1)} & \dot{R}_{r(3,2)} & \dot{R}_{r(3,3)} \end{bmatrix}, \tag{2.22}$$

for which the full expression is included in Appendix A.
The angular velocity is determined using

$$S(\omega_r) = R_r^T\dot{R}_r, \tag{2.23}$$

which can be rewritten to

$$S(\omega_r) = \begin{bmatrix} 0 & -\omega_{r3} & \omega_{r2} \\ \omega_{r3} & 0 & -\omega_{r1} \\ -\omega_{r2} & \omega_{r1} & 0 \end{bmatrix}, \tag{2.24}$$

where

$$S_{32} = R_{r31}^T\dot{R}_{r12} + R_{r32}^T\dot{R}_{r22} + R_{r33}^T\dot{R}_{r32} = \omega_{r1} \tag{2.25a}$$

$$S_{13} = R_{r11}^\top\dot{R}_{r13} + R_{r12}^\top\dot{R}_{r23} + R_{r13}^\top\dot{R}_{r33} = \omega_{r2} \tag{2.25b}$$

$$S_{21} = R_{r21}^\top\dot{R}_{r11} + R_{r22}^\top\dot{R}_{r21} + R_{r23}^\top\dot{R}_{r31} = \omega_{r3} \tag{2.25c}$$

Thus ω_r becomes

$$\omega_r = \begin{bmatrix} \omega_{r1} & \omega_{r2} & \omega_{r3} \end{bmatrix}^\top. \tag{2.26}$$

The second derivative of the rotation matrix R_r is written in a similar way as the first derivative, again to keep the expression easy to follow. This is done as

$$\ddot{R}_r = \begin{bmatrix} \ddot{R}_{r(1,1)} & \ddot{R}_{r(1,2)} & \ddot{R}_{r(1,3)} \\ \ddot{R}_{r(2,1)} & \ddot{R}_{r(2,2)} & \ddot{R}_{r(2,3)} \\ \ddot{R}_{r(3,1)} & \ddot{R}_{r(3,2)} & \ddot{R}_{r(3,3)} \end{bmatrix}. \tag{2.27}$$

To then derive the derivative of the angular velocity ω_r, (2.25) is used to get

$$\dot{\omega}_r = \begin{bmatrix} \dot{\omega}_{r1} & \dot{\omega}_{r2} & \dot{\omega}_{r3} \end{bmatrix}^\top, \tag{2.28}$$

with

$$\dot{\omega}_{r1} = \dot{R}_{r(1,3)}\dot{R}_{r(1,2)} + R_{r(1,3)}\ddot{R}_{r(1,2)} + \dot{R}_{r(2,3)}\dot{R}_{r(2,2)} + R_{r(2,3)}\ddot{R}_{r(2,2)} + \dot{R}_{r(3,3)}\dot{R}_{r(3,2)} + R_{r(3,3)}\ddot{R}_{r(3,2)} \tag{2.29a}$$

$$\dot{\omega}_{r2} = \dot{R}_{r(1,1)}\dot{R}_{r(1,3)} + R_{r(1,1)}\ddot{R}_{r(1,3)} + \dot{R}_{r(2,1)}\dot{R}_{r(2,3)} + R_{r(2,1)}\ddot{R}_{r(2,3)} + \dot{R}_{r(3,1)}\dot{R}_{r(3,3)} + R_{r(3,1)}\ddot{R}_{r(3,3)} \tag{2.29b}$$

$$\dot{\omega}_{r3} = \dot{R}_{r(1,2)}\dot{R}_{r(1,1)} + R_{r(1,2)}\ddot{R}_{r(1,1)} + \dot{R}_{r(2,2)}\dot{R}_{r(2,1)} + R_{r(2,2)}\ddot{R}_{r(2,1)} + \dot{R}_{r(3,2)}\dot{R}_{r(3,1)} + R_{r(3,2)}\ddot{R}_{r(3,1)} \tag{2.29c}$$

The torque can then be determined as

$$\tau_r = J\dot{\omega}_r - S\left(J\omega_r\right)\omega_r. \tag{2.30}$$

This completes the derivation of the reference trajectory.

2.4 Concluding remarks

This chapter has described the general dynamics of a quad-copter drone. These general dynamics have been used to derive a reference trajectory. The majority of the reference trajectory follows from the initial expression for the flat outputs $\gamma_r(t)$ and its derivatives. The way the reference trajectory follows from the flat output signals has been described in detail. The descriptions given in this chapter can be copied to run a simulation and need no further calculations. Three different representations of these flat output signals were chosen and described, namely a hover manoeuvre, a back and forth movement in x and y, and a reference used in research done by Lefeber et al [11]. The reference trajectory derived in this chapter follows the same dynamics as the real quad-copter, ensuring that the reference is feasible for the drone to follow. With the reference known, a controller can be designed and tested. The next chapter goes into more detail about the specifics of the controller used in this thesis.

3 Controller

The reference trajectory is determined in the previous chapter. For the drone to follow this reference a controller is required. This chapter dives into what controller is used in this thesis. The controller that is used, is the one designed by Lefeber et al. [11]. Since the description of the controller in [11] is not detailed enough to implement in simulations immediately, this chapter describes the controller in more detail. This way it can be copied to implemented in simulations. The position and attitude controller are presented in sections 3.1 and 3.2 respectively. These two are then combined to make the final controller. Once this is finished, the controller can be tested. The testing is done in detail in chapter 4.

3.1 Position control

In this section a position tracking controller is considered. Under the assumption that body-fixed linear accelerations can be used as (virtual) input. The tracking error in the body-fixed frame of the reference is defined as

$$\begin{bmatrix} \rho_e \\ \nu_e \end{bmatrix} = \begin{bmatrix} R_r^T (\rho_r - \rho) \\ \nu_r - R_r^T R \nu \end{bmatrix}. \tag{3.1}$$

With this definition, the error dynamics are defined as

$$\begin{aligned} \dot{\rho}_e &= -S(\omega_r)\,\rho_e + \nu_e \\ \dot{\nu}_e &= -S(\omega_r)\,\nu_e + \frac{f}{m} R_r^T R e_3 - \frac{f_r}{m} e_3. \end{aligned} \tag{3.2}$$

To stabilize these error dynamics, a virtual input u is defined as

$$u = \frac{f}{m} R_r^T R e_3 - \frac{f_r}{m} e_3, \tag{3.3}$$

which can be controlled by using the thrust magnitude f and the attitude. Rearranging this virtual input gives the expression for the thrust magnitude

$$\begin{aligned} mu + f_r e_3 &= f R_r^T R e_3 \\ f &= \|mu + f_r e_3\|. \end{aligned} \tag{3.4}$$

This gives the dynamics

$$\begin{aligned} \dot{\rho}_e &= -S(\omega_r)\,\rho_e + \nu_e \\ \dot{\nu}_e &= -S(\omega_r)\,\nu_e + u, \end{aligned} \tag{3.5}$$

in closed loop with the dynamic output feedback

$$\begin{aligned} u &= -\sigma\left(k_\rho \hat{\rho}_e + k_\nu \hat{\nu}_e\right) \\ \dot{\hat{\rho}}_e &= -S(\omega_r)\,\hat{\rho}_e + \nu_e + L_1 z \\ \dot{\hat{\nu}}_e &= -S(\omega_r)\,\hat{\nu}_e + u + L_2 z \\ \dot{z} &= -S(\omega_r)\,z - (L_1 + L_3)\,z + (L_1 + L_3)\,\tilde{\rho}_e \end{aligned} \tag{3.6}$$

with $k_\rho > 0$, $k_\nu > 0$ (both feedback gains), $L_1 > 0$, $L_2 > 0$, $L_3 > 2L_2/L_1$ (all three are filter parameters). This system is UGAS and ULES as proven in [11]. To avoid reaching the maximum motor ratio, the saturation function $\sigma(x)$ is defined as

$$\sigma(x) = \gamma \tanh\left(\|x\|_2/\gamma\right) \|x\|_2^{-1} x. \tag{3.7}$$

In this saturation function, when x approaches zero, the Taylor series expansion is used.
This concludes the position controller, the next section goes into detail about the attitude controller.

3.2 Attitude control

In this section the attitude controller is considered. The following dynamics and reference dynamics are used

$$\begin{aligned} \dot{R} &= RS(\omega) & \dot{R}_r &= R_r S(\omega_r) \\ J\dot{\omega} &= S(J\omega)\omega + \tau & J\dot{\omega}_r &= S(J\omega_r)\omega_r + \tau_r. \end{aligned} \tag{3.8}$$

The errors are defined as $R_e = R_r R^\top$, $\tilde{R} = \hat{R}R^\top$, $\omega_e = \omega_r - \omega$, $\tilde{\omega} = \hat{\omega} - \omega$ and $\hat{\omega}_e = \omega_r - \hat{\omega}$.

The input then becomes

$$\tau = \tau_r + S\left(J\hat{\omega}_e\right)\omega_r + K_\omega \hat{\omega}_e + \sum_{i=1}^{n} k_i S\left(R_r^\top v_i\right)\hat{R}^\top v_i \tag{3.9}$$

$$\begin{aligned} \dot{\hat{R}} &= \hat{R}S(\omega + \delta_R) \\ J\dot{\hat{\omega}} &= S(J\omega)\omega + \tau + \delta_\omega \end{aligned} \tag{3.10}$$

where the innovation terms δ_R and δ_ω are given by

$$\begin{aligned} \delta_R &= -c_R \sum_{i=1}^{n} k_i S\left(\hat{R}^\top v_i\right)\left(R_r^\top v_i + R^\top v_i\right) \\ \delta_\omega &= -c_\omega JS\left(\omega_r\right)\omega_e - c_\omega K_\omega \omega_e - C_\omega \tilde{\omega} \end{aligned} \tag{3.11}$$

This input with the requirements $K_\omega = K_\omega^\top > 0$, $C_\omega = C_\omega^\top > 0$, $c_R > 0$, $c_\omega > 0$ and $k_i > 0$ such that $M = \sum_{i=1}^{n} k_i v_i v_i^\top$ has distinct eigenvalues results in the equilibrium point $(R_e, \tilde{R}, \omega_e, \tilde{\omega}) = (I, I, 0, 0)$ being UaGAS and ULES.

This concludes the attitude controller. In the next section both the position and attitude controller are combined into the final controller.

3.3 Combined control

In this section the position and attitude controllers from section 3.1 and 3.2 are combined into a final controller by introducing desired reference dynamics for the attitude.
To find the desired thrust direction, it is required that $fR_r^\top Re_3$ converges to $mu + f_r e_3$. To achieve this, the desired thrust direction is defined as

$$f_d = \begin{bmatrix} f_{d1} \\ f_{d2} \\ f_{d3} \end{bmatrix} = \frac{f_r e_3 + mu}{\|f_r e_3 + mu\|} = \frac{f_r e_3 + mu}{f}. \tag{3.12}$$

The matrix

$$R_d = \begin{bmatrix} 1 - \frac{f_{d1}^2}{1+f_{d3}} & -\frac{f_{d1}f_{d2}}{1+f_{d3}} & f_{d1} \\ -\frac{f_{d1}f_{d2}}{1+f_{d3}} & 1 - \frac{f_{d2}^2}{1+f_{d3}} & f_{d2} \\ -f_{d1} & -f_{d2} & f_{d3} \end{bmatrix} \in \mathrm{SO}(3), \tag{3.13}$$

and its derivative

$$\dot{R}_d = R_d S(\omega_d), \tag{3.14}$$

then denote the desired rotation matrix and also gives the desired angular rate

$$\omega_d = \begin{bmatrix} -\dot{f}_{d2} + \frac{f_{d2}\dot{f}_{d3}}{1+f_{d3}} \\ \dot{f}_{d1} - \frac{f_{d1}\dot{f}_{d3}}{1+f_{d3}} \\ \frac{f_{d2}\dot{f}_{d1} - f_{d1}\dot{f}_{d2}}{1+f_{d3}} \end{bmatrix}, \tag{3.15}$$

and its derivative

$$\dot{\omega}_d = \begin{bmatrix} -\ddot{f}_{d2} + \frac{\left(\dot{f}_{d2}\dot{f}_{d3}+f_{d2}\ddot{f}_{d3}\right)(1+f_{d3})-f_{d2}\dot{f}_{d3}^2}{(1+f_{d3})^2} \\ \ddot{f}_{d1} - \frac{\left(\dot{f}_{d1}\dot{f}_{d3}+f_{d1}f_{d3}\right)-f_{d1}\dot{f}_{d3}^2}{(1+f_{d3})^2} \\ \frac{\left(f_{d2}\ddot{f}_{d1}-f_{d1}\ddot{f}_{d2}\right)(1+f_{d3})-\left(f_{d1}f_{d2}-f_{d1}\dot{f}_{d2}\right)\dot{f}_{d3}}{(1+f_{d3})^2} \end{bmatrix} \tag{3.16}$$

The expressions for $\dot{f}_d$ and $\ddot{f}_d$ are

$$\dot{f}_d = \frac{m\dot{u} - \dot{f}f_d}{f}, \tag{3.17}$$

$$\ddot{f}_d = \frac{1}{f}\left(-\ddot{f}f_d - 2\dot{f}\dot{f}_d + m\ddot{u}\right), \tag{3.18}$$

using

$$\dot{f} = \frac{(f_r e_3 + mu)^\top}{f} m\dot{u} \tag{3.19}$$

and

$$\ddot{f} = \frac{1}{f}(-\dot{f}^2 + m^2\dot{u}^\top\dot{u} + (f_r e_3 + mu)^\top m\ddot{u}) \tag{3.20}$$

The expressions for $\dot{f}_d$ and $\ddot{f}_d$ require the first and second derivative of the virtual input signal u defined in (3.6). These derivatives are shown in (3.22) and (3.23) respectively. The expressions for a, b, c, d, e and the derivatives are shown in (3.24). When the expression for x approaches zero, the Taylor series expansion is used.

$$u = -\sigma(x) = \gamma \tanh\tanh(\frac{|x\|_2}{\gamma})\|x\|_2^{-1}x. \tag{3.21}$$

$$\dot{u} = -\frac{\mathrm{d}\sigma(x)}{\mathrm{d}t} = -(\gamma x^\top\dot{x}(\mathrm{sech}^2(\frac{||x||_2}{\gamma})\frac{1}{||x||_2^2} - \tanh(\frac{||x||_2}{\gamma})\frac{1}{||x||_2^3})x + \gamma\tanh(\frac{||x||_2}{\gamma})\frac{1}{||x||_2}\dot{x}) \tag{3.22}$$

$$\ddot{u} = -((\dot{a}(bc-de) + a((\dot{b}c + b\dot{c}) - (\dot{d}e + d\dot{e})))x + \gamma\tanh(\frac{||x||_2}{\gamma})\frac{1}{||x||_2}\ddot{x}) \tag{3.23}$$

$$a = \gamma x^\top\dot{x} \tag{3.24a}$$

$$\dot{a} = \gamma(\dot{x}^\top\dot{x} + x^\top\ddot{x}) \tag{3.24b}$$

$$b = \mathrm{sech}^2(\frac{||x||_2}{\gamma}) \tag{3.24c}$$

$$\dot{b} = -2\tanh(\frac{||x\|_2}{\gamma})\,\mathrm{sech}^2(\frac{||x||_2}{\gamma})\frac{x^\top\dot{x}}{||x||_2} \tag{3.24d}$$

$$c = \frac{1}{||x||_2^2} \tag{3.24e}$$

$$\dot{c} = -\frac{2\dot{x}}{||x||_2^3} \tag{3.24f}$$

$$d = \tanh(\frac{||x\|_2}{\gamma}) \tag{3.24g}$$

$$\dot{d} = \mathrm{sech}^2(\frac{||x||_2}{\gamma})\frac{x^\top\dot{x}}{||x||_2} \tag{3.24h}$$

$$e = -\frac{3x^\top\dot{x}}{||x||_2^5} \tag{3.24i}$$

$$\dot{e} = \frac{1}{||x||_2^3} \tag{3.24j}$$

Within these equations the expression for $x = k_\rho \hat{\rho}_e + k_\nu \hat{\nu}_e$ is used. The first and second derivative of this expression are

$$\dot{x} = -S\left(\omega_r\right) x + k_\rho \hat{\nu}_e + k_\nu u + \left(k_\rho L_1 + k_\nu L_2\right) z, \tag{3.25}$$

and

$$\ddot{x} = -S\left(\dot{\omega}_r\right) x - S\left(\omega_r\right) \dot{x} + k_\rho \dot{\hat{\nu}}_e + k_\nu \frac{\mathrm{d}\sigma(x)}{\mathrm{d}t} + \left(k_\rho L_1 + k_\nu L_2\right) \dot{z}. \tag{3.26}$$

With these desired dynamics known, the reference dynamics can be updated. Making the new reference variables

$$R_{rd} = R_r R_d, \tag{3.27a}$$

$$\omega_{rd} = R_d^\top \omega_r + \omega_d. \tag{3.27b}$$

Using these new variables, the reference dynamics then become

$$\dot{R}_{rd} = R_{rd} S(\omega_{rd}) \tag{3.28a}$$

$$J\dot{\omega}_{rd} = S(J\omega_{rd})\omega_{rd} + \tau_{rd}, \tag{3.28b}$$

where τ_{rd} can be derived using equations (3.27b) and (3.28b) to be

$$\tau_{rd} = J(\dot{R}_d^\top \omega_r + R_d^\top \dot{\omega}_r + \dot{\omega}_d) - S(J\omega_{rd})\omega_{rd}. \tag{3.29}$$

This then changes the closed loop input τ to

$$\tau = \tau_{rd} + S\left(J\hat{\omega}_e\right)\omega_{rd} + K_\omega \hat{\omega}_e + \sum_{i=1}^{n} k_i S\left(R_{rd}^\top v_i\right) \hat{R}^\top v_i \tag{3.30}$$

The errors from the attitude controller of the previous section are changed to include these new reference dynamics. The errors are changed to $R_e = R_r R_d R^\top$ and $\omega_e = R_d^\top \omega_r + \omega_d - \omega$.
This completes the controller, which can now be tested in simulations to see if it can follow the references defined in chapter 2.

3.4 Concluding remarks

In this chapter the controller used in this research was described. This controller is the one designed by Lefeber et al [11]. The controller was described in more detail in this chapter than in [11], this way it is possible to copy and implement the controller from this chapter in simulations. The position and attitude control were described, after which both were combined into the final controller. With both the reference trajectory known and a controller designed, the controller can now be implemented in simulations.

4 Controller implementation in simulations

Previous chapters have discussed the reference trajectories and the controller that is used. This chapter focuses on testing the controller in simulations with the different references. These simulations give an indication of the behaviour of the drone and the controller when different reference trajectories are given.

Three different reference trajectories are used to test the controller. These tests are described in the subsections of this chapter. The simulations are done in Matlab using the solver ode15s. This solver is used instead of ode45 because the simulation ran significantly slower when using ode45. When looking for alternatives, ode15s resulted in a faster simulation while still giving the same numerical result. The dynamics that are simulated have been described in detail in previous chapters, but are repeated below in short.

$$\begin{aligned} \dot{\rho} &= R\nu, \\ \dot{\nu} &= -S(\omega)\nu + gR^T e_3 - \frac{f}{m}e_3, \\ \dot{R} &= RS(\omega), \\ J\dot{\omega} &= S(J\omega)\omega + \tau. \end{aligned} \tag{4.1}$$

The state variables used in this simulation are $\rho, \nu, R, \omega, \hat{\rho}_e, \hat{\nu}_e, z, \hat{R}$ and $\hat{\omega}$. The controller used in the simulations is the one described in section 3.3, where the closed loop input τ was determined to be

$$\tau = \tau_{rd} + S\left(J\hat{\omega}_e\right)\omega_{rd} + K_\omega \hat{\omega}_e + \sum_{i=1}^{n} k_i S\left(R_{rd}^\top v_i\right)\hat{R}^\top v_i. \tag{4.2}$$

The initial conditions of the state variables and the controller gains and parameters are depicted in Table 4.1 and Table 4.2 respectively. The mass and inertia tensor are obtained from the system identification done by Brekelmans [3], the rest of the gains and parameters, with the exception of K_ω and γ which have been slightly tuned, are obtained from Lefeber et al [11], The values shown in these two tables are used in the simulations for all three different reference trajectories.

State variable	Initial value
ρ	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$
ν	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$
ω	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$
$\hat{\rho}_e$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$
$\hat{\nu}_e$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$
z	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$
$\hat{\omega}$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\top$
R	$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$
$\hat{R}$	$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$

Table 4.1: Initial conditions used in simulations

Parameter	Value	Description
m	0.068	Mass of the drone (kg)
g	9.81	Gravitational acceleration (m/s^2)
γ	4	Saturation bound ($\|\|u(t)\|\|_2 \leq \gamma$)
(k_ρ, k_ν)	(2,2)	Translational control gains
(L_1, L_2, L_3)	(4,4,4)	Translational filter gains
(k_1, k_2, k_3, K_ω)	(10,20,30,0.5I)	Attitude control gains
(c_R, c_ω, C_ω)	(1,10,15I)	Attitude filter gains
v_1	$\begin{bmatrix}0 & 0 & -1\end{bmatrix}^\top$	Direction (gravity)
v_2	$\begin{bmatrix}0.98 & 0.17 & 0\end{bmatrix}^\top$	Direction (Magnetic field)
v_3	$v_1 \times v_2$	Virtual measurement direction
J	$\frac{1}{1000}\begin{bmatrix}0.069 & 0 & 0\\ 0 & 0.0775 & 0\\ 0 & 0 & 0.150\end{bmatrix}$	Inertia tensor (kg/m^2)

Table 4.2: Parameters used in simulations

4.1 Reference 1

This section focuses on the first reference described in section 2.2. Note that only part of the signals is shown in (4.3). For the simulation, the first and second derivative of $\psi_r(t)$ are required, as well as the first up to the fourth derivatives of $x_r(t)$, $y_r(t)$ and $z_r(t)$. In this specific case all the derivatives are zero, because of the simple hover reference. This reference trajectory is simulated with the initial conditioned mentioned before.

$$x_r(t) = 0, \quad y_r(t) = 0, \quad z_r(t) = -1, \quad \psi_r(t) = 0. \tag{4.3}$$

The position (x, y, z), velocity $(\dot{x}, \dot{y}, \dot{z})$ and attitude rate (roll, pitch, yaw) are shown in Figure 4.1.

As can be seen from this figure, all three variables end up following the reference given. The x, y and z position and velocity follow the reference after a short period. What stands out from this plot is the behaviour of the attitude rate. Since this is a simple hover reference, it was expected that no rotation would be present. The peak in the attitude rate is in the range of e^{-3}, making it relative enough that it can not be assumed to be noise.
To find out what causes this behaviour in the attitude rate, two different versions of the simulations are made. One where only the translational dynamics are considered (with u as a virtual input), and one where only the attitude dynamics are considered. These two simulations are discussed in further detail in subsection 4.1.1 and 4.1.2 respectively.

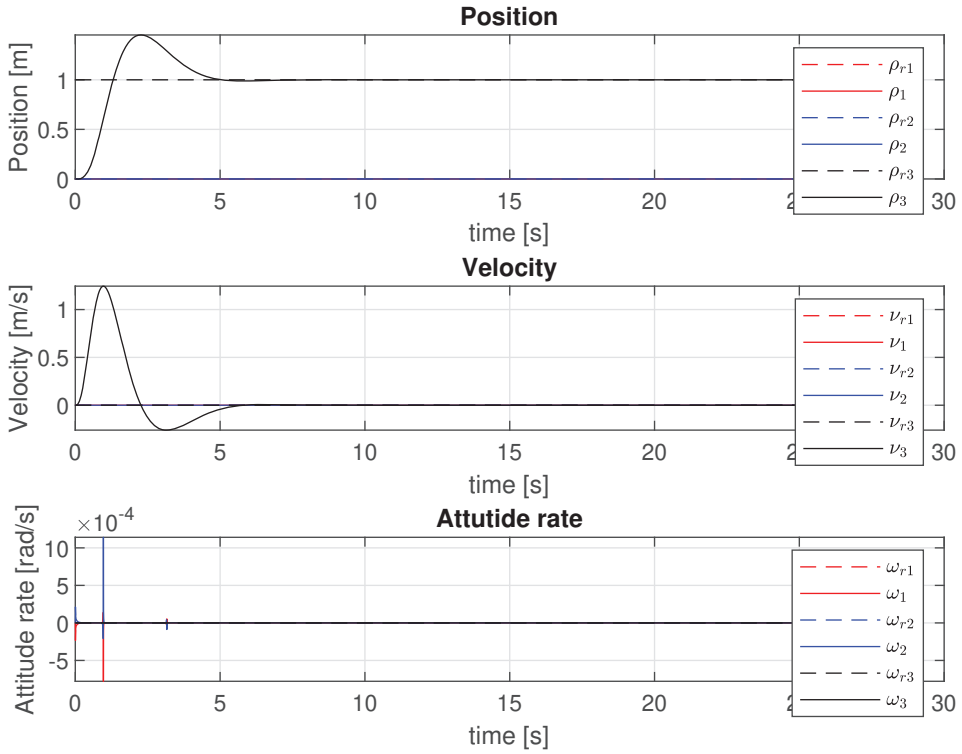

Figure 4.1: Plot of ρ, ν and ω against time for a hover reference

4.1.1 Translational dynamics

In this section a separate simulation is made only considering the translational dynamics

$$\begin{aligned} \dot{\rho} &= R\nu, \\ \dot{\nu} &= -S(\omega)\nu + gR^T e_3 - \frac{f}{m}e_3, \end{aligned} \tag{4.4}$$

and uses the controller described in section 3.1, using u as a virtual input. The issue with these expressions, is that R and ω are not considered in this case. Thus an alternative way of expressing these dynamics is required, where these two variables are not present. This is done by taking the second derivative of the position ρ, which can be described as

$$\ddot{\rho} = \dot{R}\nu + R\dot{\nu} = RS(\omega)\nu + R(-S(\omega)\nu + gR^T e_3 - \frac{f}{m}e_3), \tag{4.5}$$

which reduces to

$$\ddot{\rho} = ge_3 - R(\frac{f}{m})e_3. \tag{4.6}$$

In this expression for $\ddot{\rho}$ only R remains. It is possible to use the expression for u used in (3.3) to help rewriting this. The expression for u can be rewritten to

$$R(\frac{f}{m})e_3 = R_r(u + \frac{f_r}{m}e_3), \tag{4.7}$$

which can then be substituted into (4.6) to obtain the expression for the second derivative of the position

$$\ddot{\rho} = ge_3 - R_r(u + \frac{f_r}{m}e_3). \tag{4.8}$$

This expression does not depend on R or ω anymore, and can thus be simulated. The resulting behaviour of the position and velocity is shown in Figure 4.2. This plot shows that the controller is capable of following the hover reference. The behaviour in this simulation is very similar to the one shown before.

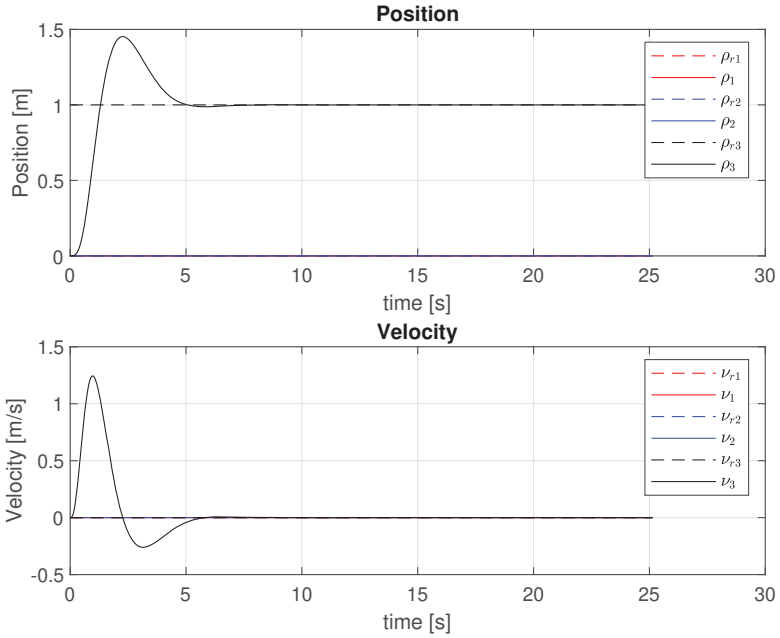

Figure 4.2: Position and velocity over time

4.1.2 Attitude dynamics

This section only considers the attitude dynamics

$$\begin{aligned} \dot{R} &= RS(\omega), \\ J\dot{\omega} &= S(J\omega)\omega + \tau, \end{aligned} \tag{4.9}$$

and uses the controller described in section 3.2. Thus the dynamics for $\hat{R}$ and $\hat{\omega}$ are also simulated. The behaviour of ω is shown in Figure 4.3. As expected there is no rotation with this specific reference, since it is just a hover reference. This indicates that, at least judging only by this simple reference, that both parts of the controller work separately, and the unexpected behaviour of Figure 4.1 is caused by something in the combining of these two controller parts. A more detailed look into this expectation is taken in section 4.4.

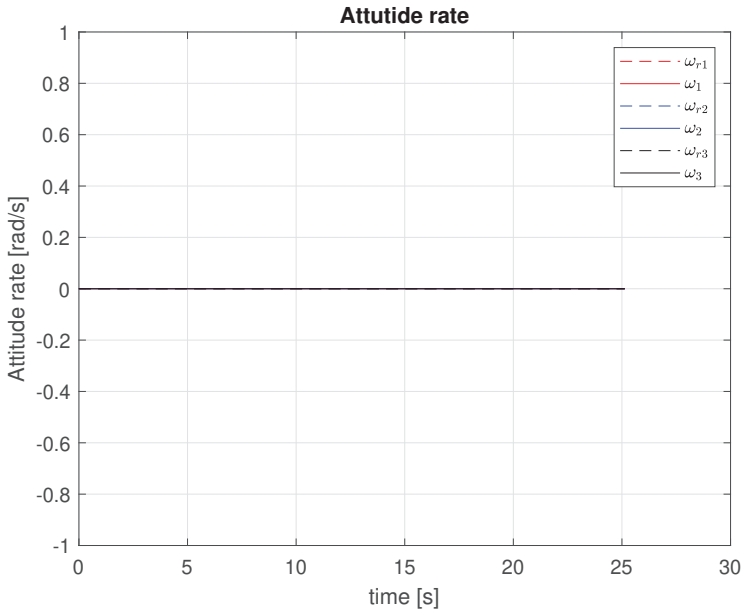

Figure 4.3: Angular velocity over time

4.2 Reference 2

This section focuses on the second reference described in section 2.2.
The signals for $x_r(t)$, $y_r(t)$, $z_r(t)$ and $\psi_r(t)$ for this reference are

$$x_r(t) = a_x \sin(b_x t), \quad y_r(t) = a_y \sin(b_y t), \quad z_r(t) = a_z \sin(b_z t) - 1, \quad \psi_r(t) = a_\psi \sin(b_\psi t), \tag{4.10}$$

with $a_x = 0.7$, $b_x = 0.5$, $a_y = 0.5$, $b_y = 1.0$, $a_z = 0$ and $a_\psi = 0$.
The derivatives of the signals have not been repeated here, these derivatives, together with a visualization of the reference, can be found in section 2.2, equation 2.6.

The same initial conditions as before are used in this simulation. The resulting behaviour is shown in Figure 4.4. This plot shows that all variables follow the reference after circa 4-5 seconds. This confirms that the controller is able to control the drone, even with a more challenging reference trajectory than hovering.

What stands out from this figure is the behaviour of the attitude rate. Both the first and second component of ω seem to move away from the reference at first. Interestingly, the peaks are at the same point in time as they were with the hover reference, showing that it is worth investigating the source. Despite this behaviour, the controller does manage to follow the reference trajectory.

Looking back at the hover reference, the attitude rate shows similar behaviour, where it seems to have a peak before converging to it. Since this simulation shows similar behaviour as with the previous reference, the same checks are done with this reference trajectory, separating the translational and attitude dynamics.
Since the details on how these separate simulations are done has been discussed before, it is not repeated here.

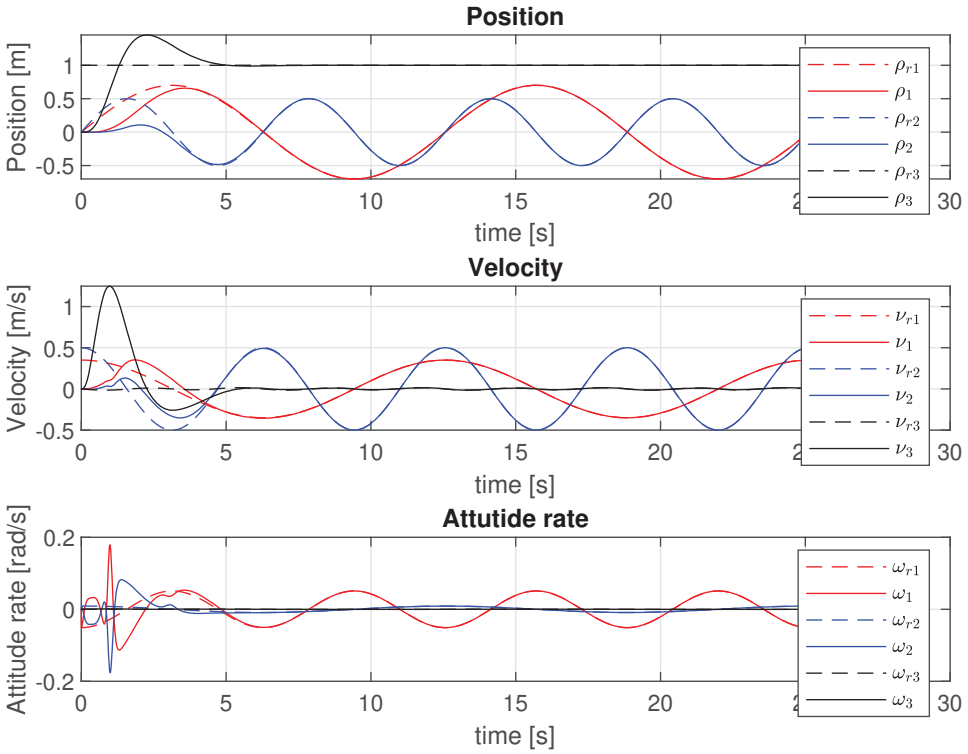

Figure 4.4: Plot of ρ, ν and ω against time for a wave movement in x and y reference

4.2.1 Translational dynamics

The details on how this simulation is constructed has been described in section 4.1.1 and is not repeated here. Simulating only the translational dynamics for the second reference trajectory gives the behaviour shown in Figure 4.5. This shows that the behaviour in the x and y direction is very similar to what is shown in Figure 4.4. Both the position and velocity converge to the reference within 4-5 seconds.

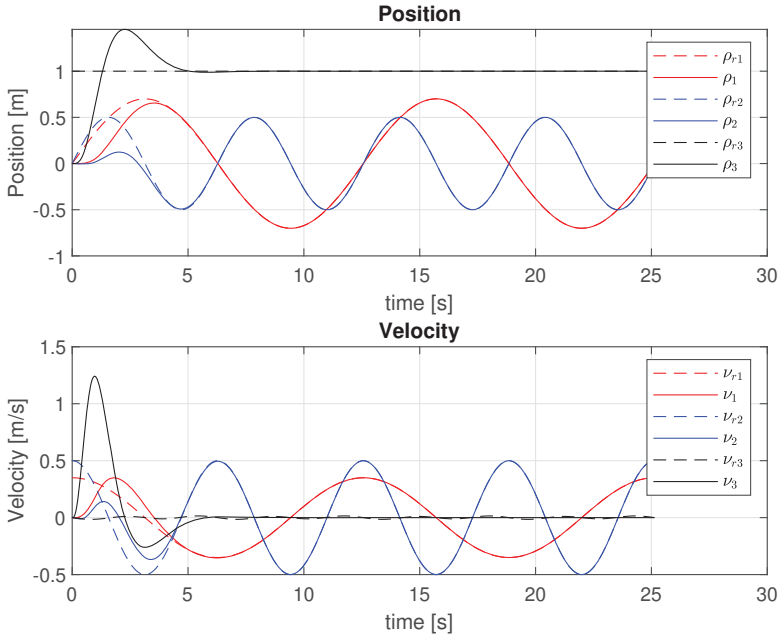

Figure 4.5: Position and velocity over time

4.2.2 Attitude dynamics

The details on how this simulation is constructed has been described in section 4.1.2 and is not repeated here. Simulating the attitude dynamics for the second reference trajectory gives the behaviour shown in Figure 4.6. Which shows that the controller is not able to fully match the reference, though the difference is relatively small (error of circa $2.5e^{-3}$). Interestingly, the x and y component are going in the same direction as the reference from the start. This is very different from the results shown before in Figure 4.4. This indicates that both parts of the controller work separately, even with a more challenging reference than hovering. These results further supports the expectation that the unexpected behaviour from Figure 4.1 is caused by something in the combining of the controller parts.

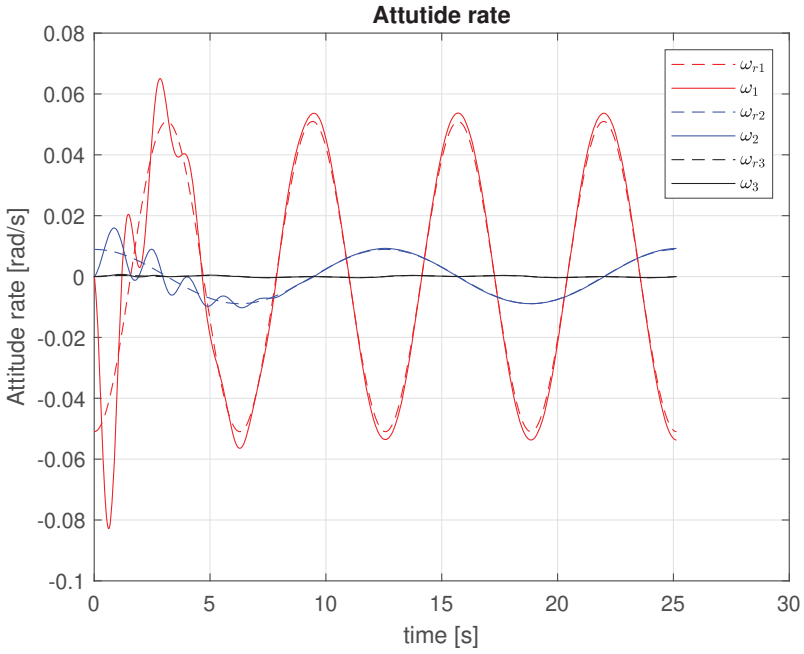

Figure 4.6: Angular velocity over time

4.3 Reference 3

This section focuses on the third reference described in section 2.2. Namely

$$x_r(t) = (6 + 2\cos(\omega_\nu t))\cos(\omega_u t), \quad y_r(t) = (6 + 2\cos(\omega_\nu t))\sin(\omega_u t), \quad z_r(t) = 2\sin(\omega_\nu t), \quad \psi_r(t) = \omega_u t + \pi, \tag{4.11}$$

where $\omega_\nu = 1.2\pi$ (rad/s) and $\omega_u = 0.2\pi$ (rad/s). The derivatives required for the simulation are left out in this section, they have been described in detail in section 2.2, a visualization of the reference trajectory is also found in that section.

The same initial conditions as before are used in this simulation. The behaviour resulting from this simulation is shown in Figure 4.7. Since the attitude rate is difficult to see because of the initial peak, a zoomed in version is shown in Figure 4.8. This plot shows that the controller manages to follow the complicated reference trajectory. However, comparing this result to that of the previous two references, it does not seem to reduce the error completely to zero. This error is shown in Figure 4.9.

What stands out here is the peak of the attitude rate. After this peak it seems to approach the reference relatively quickly. Unlike the previous references, the peak in attitude rate is at a different moment in time. The velocity and position both do not have a similar peak, but take considerably longer to converge to the reference than with the simpler reference trajectories. This is presumably because the reference is complicated.

The peak in the attitude rate could be caused by the same thing that caused the unexpected behaviour in the previous simulations. If there is a deviation during a hover reference, it might not influence the overall behaviour as much, but with a complicated reference like the one in this section, relatively small deviations could result in a much bigger impact on the behaviour.
To see if the controller parts work separately, this reference is also split up into translational and attitude dynamics only to investigate the behaviour.

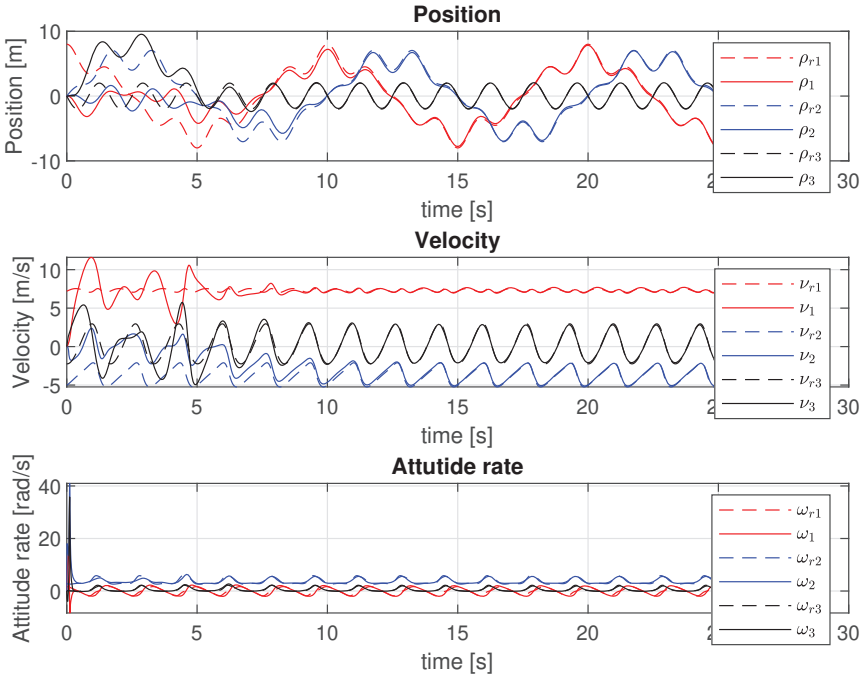

Figure 4.7: Plot of ρ, ν and ω against time for the reference from Lefeber et al. [11]

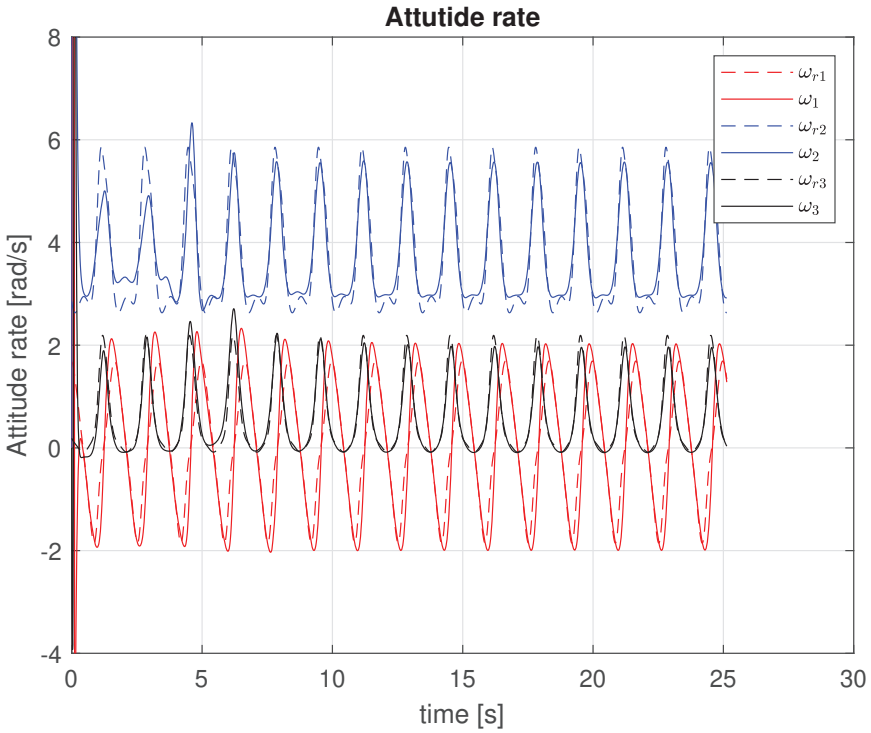

Figure 4.8: Zoomed in version of the attitude rate.

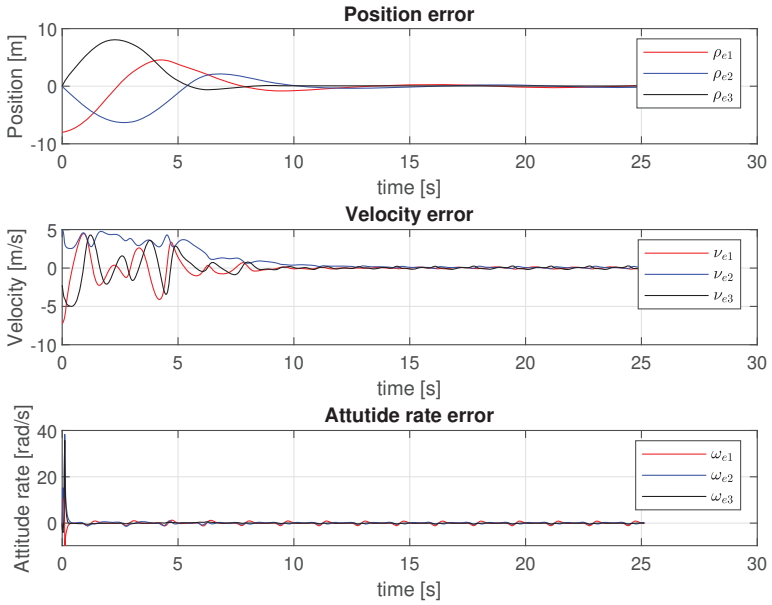

Figure 4.9: Error plot of the position, velocity and attitude rate.

4.3.1 Translational dynamics

Simulating only the translational dynamics for the second reference trajectory gives the behaviour shown in Figure 4.10. This shows that the controller is able to track the position reference. Interestingly, the z component seems to deviate from the reference significantly at the start, but converges back to the reference after circa 10 seconds.
The velocity, however, is not following the reference at all. Since Figure 4.7 shows that the combined controller does manage to follow the reference eventually, it might be that the simulation used for the translational dynamics only is too simplified for this reference. Another option is that the position or velocity estimators might not behave as expected. However, Figure 4.11 shows that the estimator errors go to zero and these thus work as intended. It could also be that the combination of the two controller parts is required for complicated reference trajectories, such as the one from this section.

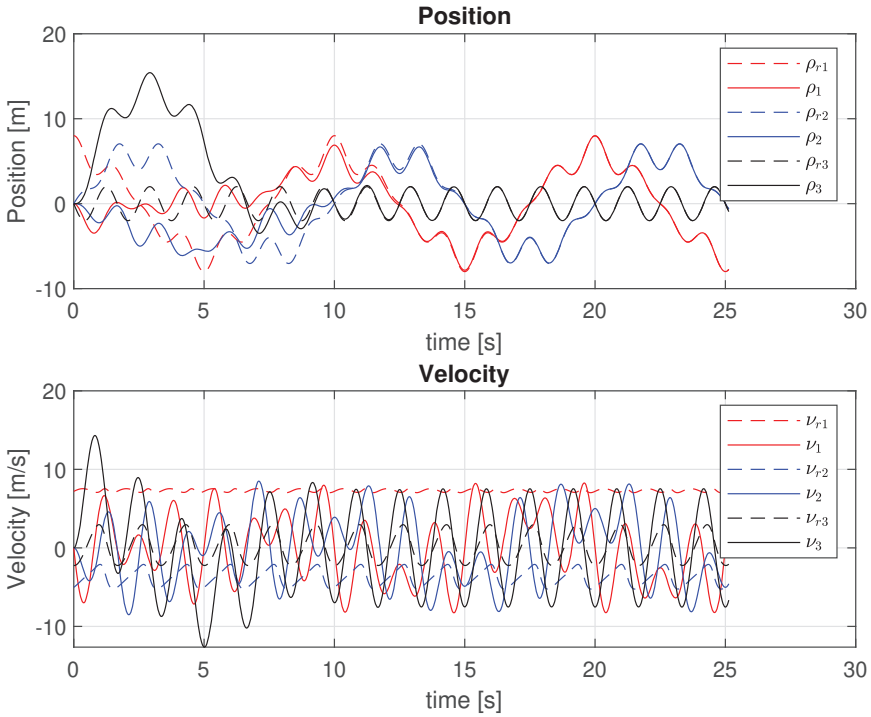

Figure 4.10: Position and velocity over time

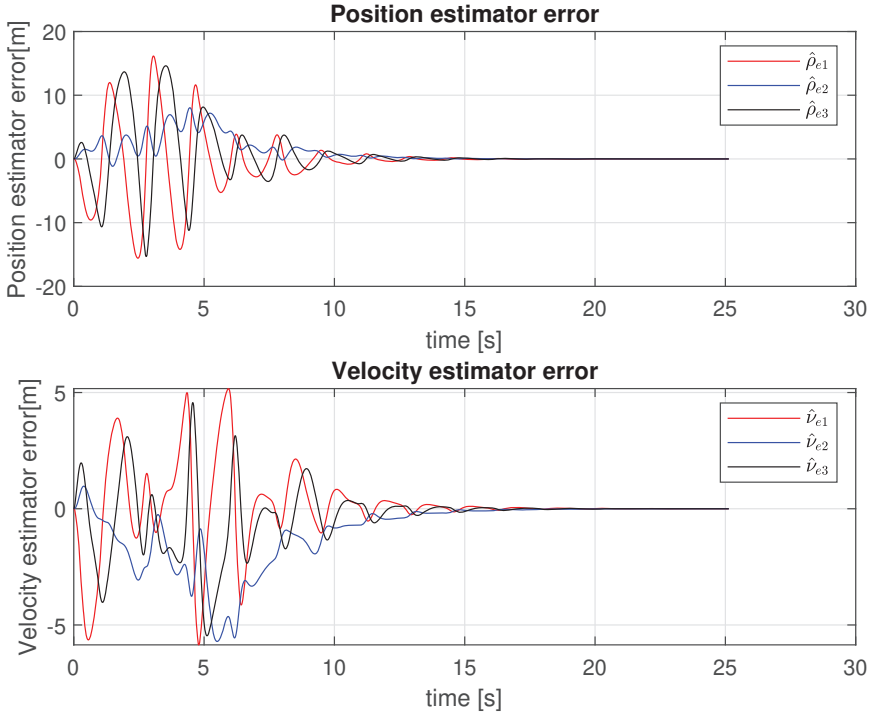

Figure 4.11: Position and velocity estimator error over time

4.3.2 Attitude dynamics

Simulating the attitude dynamics for the third reference trajectory separately, gives the behaviour depicted in Figure 4.12. This shows that the attitude controller part is not able to follow the reference. A similar pattern is achieved, but it does not seem to converge to the reference. It could be that the combination of the two controller parts is required for complicated reference trajectories, such as the one from this section.

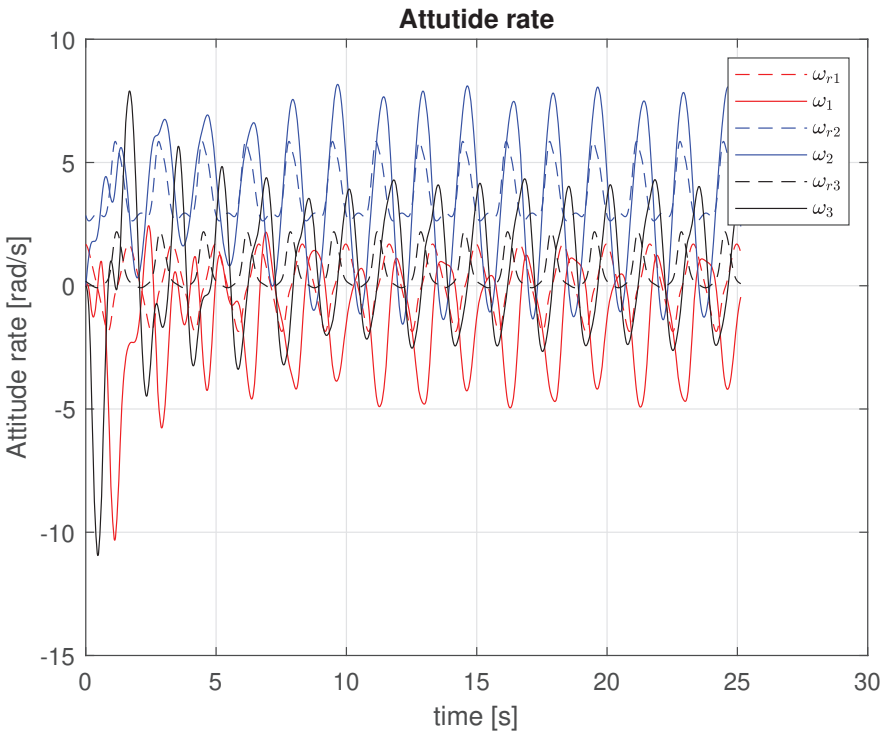

Figure 4.12: Angular velocity over time

4.4 Combined controller check

This section looks deeper into the cause of the unexpected behaviour in the controller tested in this chapter. As was clear from the previous sections, especially with the first two reference trajectories, the unexpected behaviour starts after the translational and attitude controller are combined. In other words, after the expressions for f_d, R_d and ω_d have been introduced. A possible explanation for the behaviour is that these expressions are not defined correctly. To test this, it is assumed that f_d and R_d are derived correctly, and ω_d is constructed from them.
To differentiate the result from this derivation from the previously used ω_d, the new variable is defined as ω_{dis} This derivation is done in a similar way as how ω_r was derived in the reference dynamics. Using the expression

$$S\left(\omega_{dis}\right) = R_d^T \dot{R}_d, \tag{4.12}$$

ω_{dis} then becomes $\begin{bmatrix}\omega_{dis1} & \omega_{dis2} & \omega_{dis3}\end{bmatrix}^\top$ with

$$\omega_{dis1} = R_{d31}^T \dot{R}_{d12} + R_{d32}^T \dot{R}_{d22} + R_{d33}^T \dot{R}_{d32}, \tag{4.13a}$$

$$\omega_{dis2} = R_{d11}^\top \dot{R}_{d13} + R_{d12}^\top \dot{R}_{d23} + R_{d13}^\top \dot{R}_{d33}, \tag{4.13b}$$

$$\omega_{dis3} = R_{d21}^\top \dot{R}_{d11} + R_{d22}^\top \dot{R}_{d21} + R_{d23}^\top \dot{R}_{d31}. \tag{4.13c}$$

To be able to use this newly derived ω_{dis} for simulations, the derivative is also required. This derivative is used to construct the expression of τ_{rd} as shown in section 3.3. As done before, for ease of understanding, the full written out derivation for ω_{dis} and expression of its derivative is included in Appendix C.

The expression for ω_{dis} is included in the simulation and the hover reference simulation is done again. The resulting behaviour is shown in Figure 4.13. This figure indicates that the newly derived ω_{dis} had a slight impact on the behaviour of the drone. The peak in the attitude rate has been removed in comparison to Figure 4.1. This shows that the expression for ω_{dis} gives behaviour closer to what was expected than the expression for ω_d. To check if the behaviour is improved in the other reference trajectories as well, the second reference is shown in Figure 4.14 with the expression for ω_{dis} implemented. This shows that, similar to the hover reference, the peak in the attitude rate is removed, however there is still an oscillation present in the attitude rate. This shows that the expression for ω_{dis} did not completely solve the issue. This could be because of the assumption taken before, that f_d and R_d are derived correctly. It would be good for further research to check the derivations of f_d, R_d and ω_d in more detail.

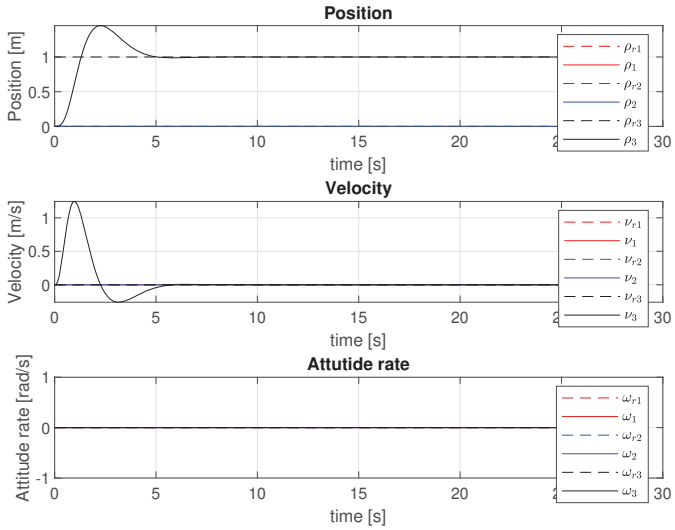

Figure 4.13: Plot of ρ, ν and ω against time for a hover reference

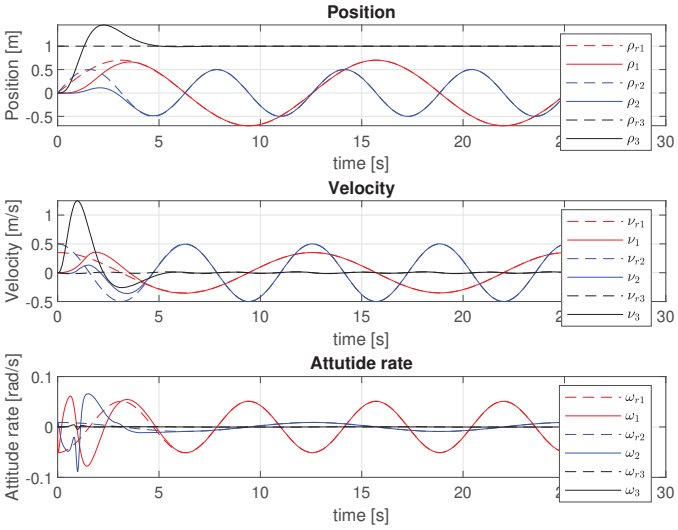

Figure 4.14: Plot of ρ, ν and ω against time for a back and forth movement reference

4.5 Concluding remarks

This chapter has tested the controller described in previous chapters. Three different references were given to be followed. In all three cases, the controller managed to converge to the reference. During the tests some unexpected behaviour was discovered and separate simulations were done to investigate the source of the unexpected behaviour. These tests were done by separating the translational and attitude dynamics. Because of these tests, the conclusion was drawn that the unexpected behaviour is caused by the combining of the controller parts, i.e. by introducing the variables for f_d, R_d and ω_d. The value for ω_d was derived again with the assumption that f_d and R_d were correct. This resulted in the variable ω_{dis}, which was implemented in the simulations and showed different behaviour than ω_d, but did not solve the issue completely. Indicating that the assumption was not correct and the expressions for f_d, R_d and ω_d have to be checked again.

This chapter showed the implementation of the controller. The controller can converge the real position to the reference trajectory with simple but also complicated references. The next step is to implement the controller on the hardware itself.

5 Lab setup

In the previous chapters, the main focus was theoretical. The reference trajectory and the controller have been discussed. These two were then tested in simulations. This chapter serves as a stepping stone to move from software to hardware and give a visualization of the context of the hardware. The layout of the lab is shown and described, after which the Parrot Mambo fly drone is shown and information about the specific drone is given.

5.1 Lab layout

This section shows the layout of the lab. The Optitrack camera system is present in this lab. Figure 5.1 shows in red where the cameras are located in the space. This picture also shows the on/off switch of the system in yellow. Once the system is turned on, it is advised to wait for circa 30 minutes for the cameras to warm up, this improves the accuracy of the localization. The computer shown in yellow in Figure 5.2 is used to run the Motive software. This software is used as an interface to interact with the Optitrack cameras. After the warm up period, the calibration can be done using Motive.

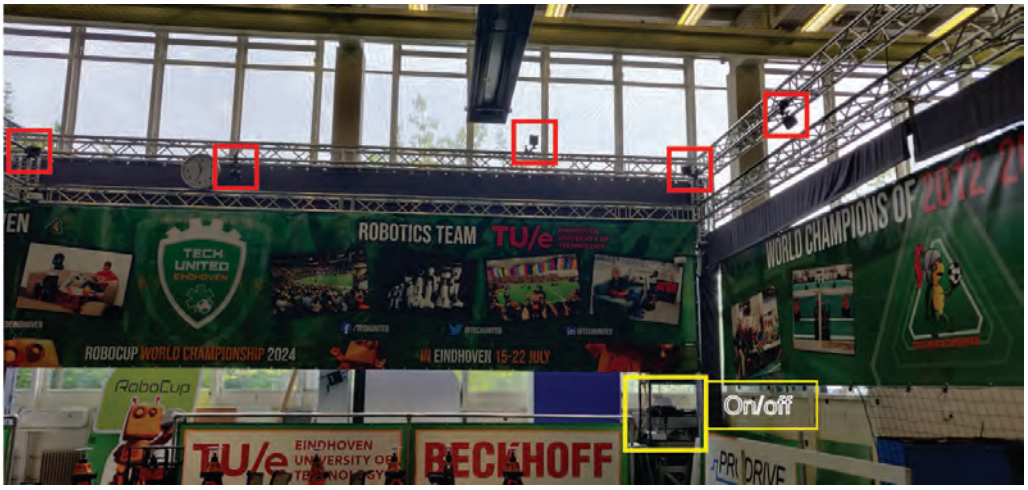

Figure 5.1: Optitrack cameras in the lab

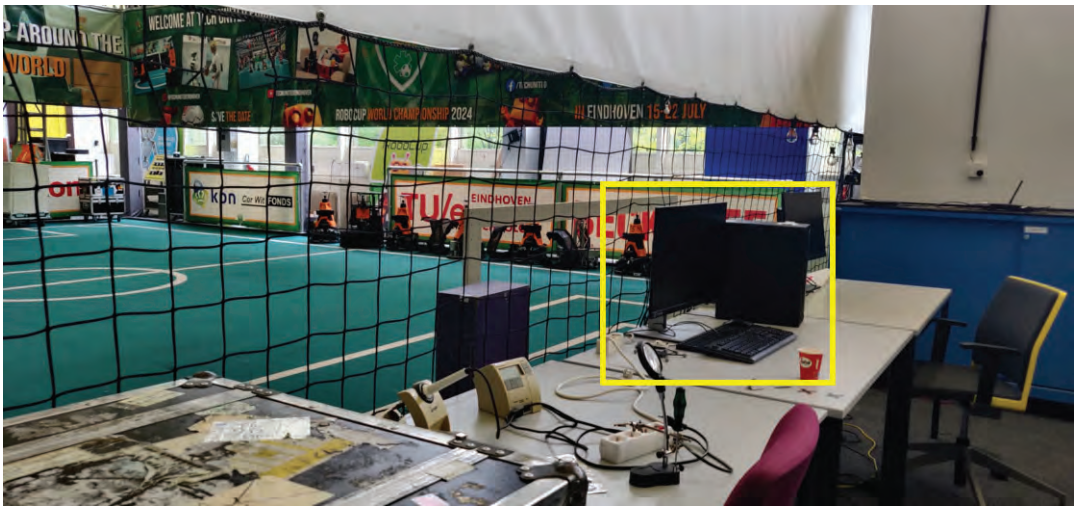

Figure 5.2: PC on which Motive runs in the lab

5.2 Drone information

The drone that is used in this thesis is the Parrot Mambo fly mini drone, shown in Figure 5.3. Within Matlab there exists a toolbox that can be used with this drone. This toolbox uses a simulink template to upload flight code to the drone. This flight code is where all the control logic happens. This simulink template is discussed in more detail in chapter 6. The mass and inertia tensor are determined by .

The values for these are $m = 0.068$ kg and $J = \begin{bmatrix} 0.069 & 0 & 0 \\ 0 & 0.0775 & 0 \\ 0 & 0 & 0.150 \end{bmatrix}$.

Figure 5.3: Parrot Mambo fly drone

5.3 Concluding remarks

In this chapter the lab environment and the Parrot Mambo fly drone have been discussed. This gives a visualization of the context of the hardware, and thus provides a stepping stone to move from software to hardware.

6 Simulink

This chapter focuses on the simulink model template provided in the Simulink Support Package for Parrot mini drones. This template can be used to load controller logic on the hardware. The simulink model is described and it is explained what the different system blocks are responsible for within the template. After the workings of the model has been explained, a description is given how a custom controller could be implemented and a theoretical example case is shown using the controller described in the previous chapters of this thesis.

6.1 Description of the Simulink model

This section gives a detailed description of the simulink model provided. To open the project, the matlab command "asbQuadcopterStart" is used. This opens the simulink file on the page shown in Figure 6.1.

The **Airframe** block is used to simulate the behaviour of the quadcopter. Two versions of this airframe can be selected, Linear and Non-linear. Linear can be used for PID controller tuning and Non-linear can be used to simulate the model in a more realistic way. In a way, this block can be seen as the "model", which is similar to the model described in this thesis. This block, together with the Environment block and the Sensors block, simulate the behaviour of the quadcopter in real life and what outputs the sensors give to the controller.
The **Environment** block is used to simulate the environmental factors that influence the quadcopter. It defines the variables gravity, air density, speed of sound, atmospheric pressure and the air temperature. Two modes can be selected within this block. One sets the environmental factors as a constant, the other sets the factors as variables that change with position.
The **Sensors** block simulates the sensors on the drone. The outputs it gives are in a similar form as the real sensors on the hardware would give to the controller. Two different modes can be selected, one where the signals are noisy, and one where they are not.
The **Visualization** block is used to show the results of the simulation. This can be set to four different modes: Scopes, Workspace, Flightgear, or Simulink 3D. This block is not necessary for the simulation, but gives a way to monitor different parameters during the simulation.
The **Command** block is used to construct the reference signal. This can be edited within the block to define a reference trajectory as desired. It is important to keep the same structure for the output values of this block, since it is used in this structure by the other blocks in the model.
The **FCS (Flight Control System)** block (highlighted in red) is responsible for the control logic in the system. If the simulink template is used to run the hardware, only the FCS block is loaded on the drone as flight code. The rest of this section dives deeper in the specifics of this block.

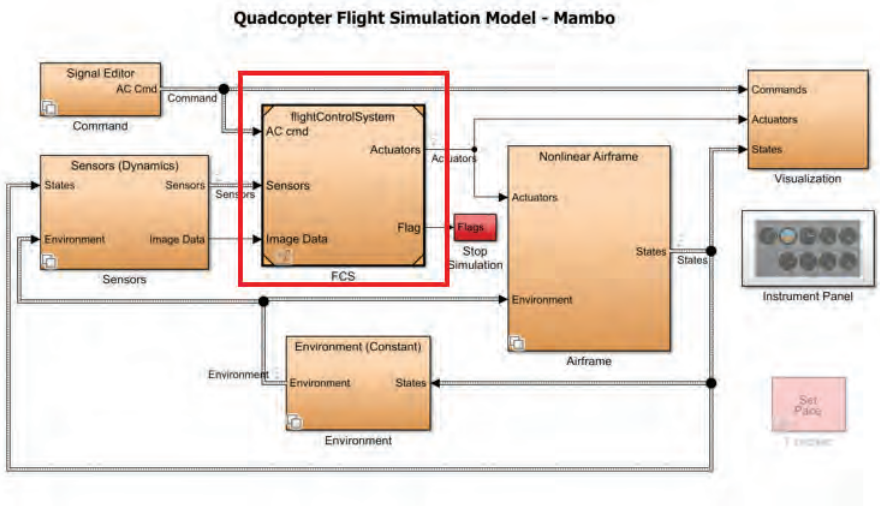

Figure 6.1: Main Model

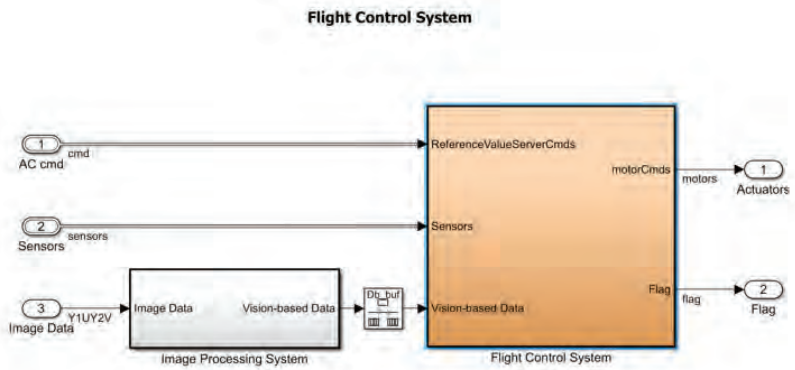

Figure 6.2: Flight Control system block

Opening the FCS block gives the screen shown in Figure 6.2. Input 3 gives the information needed for the optical flow sensor of the drone. Since this optical flow sensor is not used in this case, the input can be removed.

Opening the Flight Control System block then gives the overview shown in Figure 6.3. In this overview, there are again a number of subsystems present.
The **Landing logic** block is responsible for the drone landing smoothly after a test has been completed.
The **Sensor data group** block is for signal processing purposes to make sure the data from the sensors can be used by the other blocks in this system.
The **Estimator** block is responsible for the state estimation during the simulation.
The **Logging** block makes sure the data from the tests is saved for post processing if applicable.
The **Crash predictor** block makes sure to terminate the system when a possible crash is expected.
The **Controller** block (highlighted in red) is where all the control logic is defined. This thesis only focuses on this controller block, since that is the most relevant.

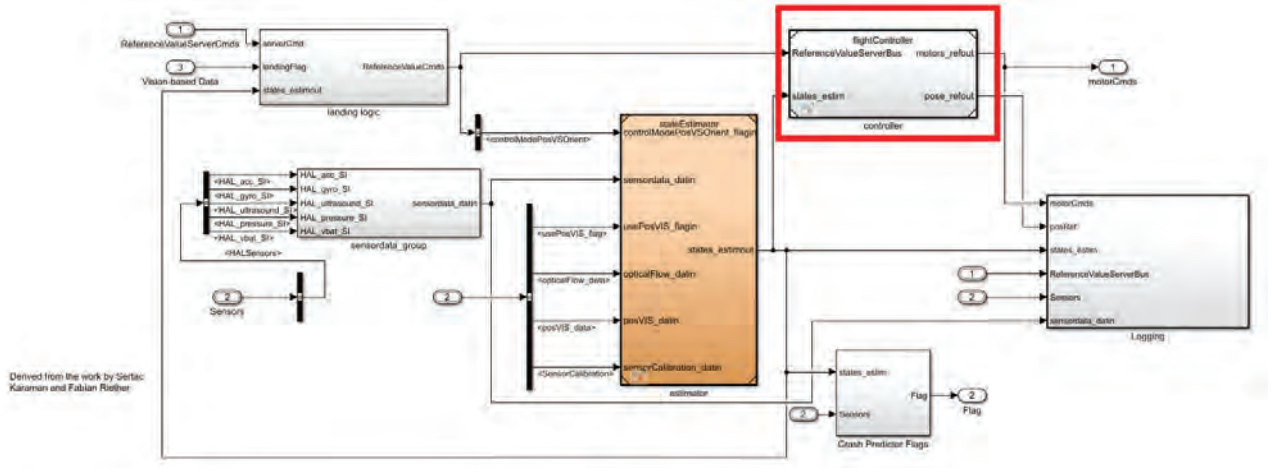

Figure 6.3: Flight control system

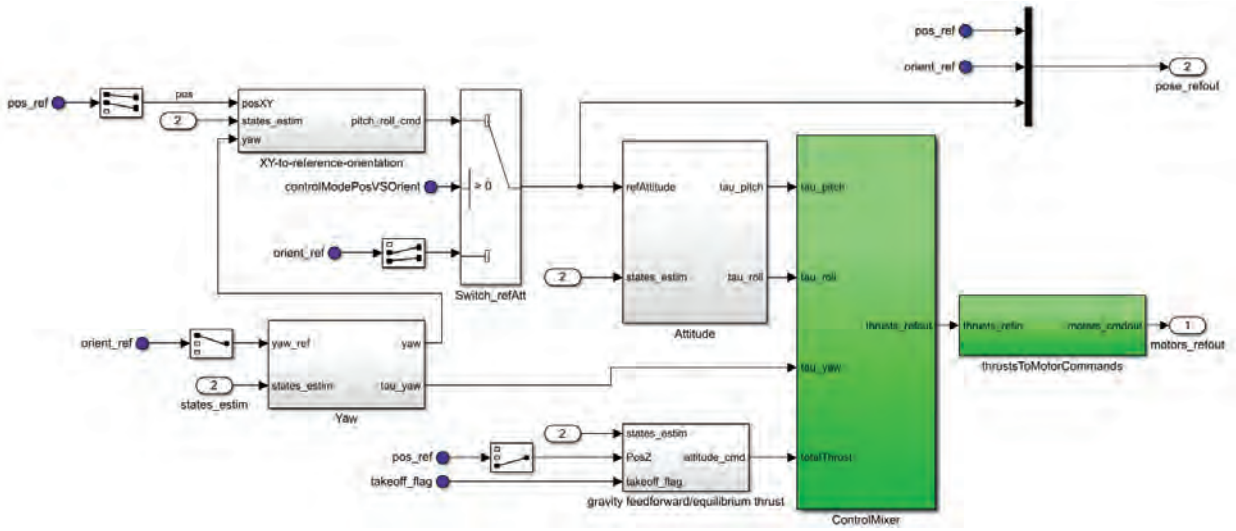

Figure 6.4: Controller block

When opening the controller block, the system shown in Figure 6.4 is opened. Within this block the controller actions are determined. The model splits the dynamics in the drone so that multiple P(I)D controllers can work separately to control the whole system. This is the block where changes can be made to implement a specially designed controller. It is important to note that the blocks in green are supposed to stay in place, as well as the inputs the whole controller system block has.

6.2 Implementing a custom controller

This section focuses on how a custom controller could theoretically be implemented in the simulink model. The most important thing is that the FCS block from the main model overview (Figure 6.1) is the only block that is set on the drone. This means that all the important logic necessary in the drone programming should be within this block. Most importantly the control logic and the reference trajectory.
To implement the reference trajectory inside the FCS block, the Landing logic block from Figure 6.3 can be altered. Here it is important to keep the structure of the outputs that is already present, since the other blocks in the model rely on this structure.
The majority of changes are made within the controller block shown in Figure 6.4. Once again it is important to keep the existing input and output structure of the main block intact and only modify the blocks within. In the template model, the control logic is modelled by using simulink operations. In theory this could all be changed to a Matlab function. This way the controller described in a previous chapter can be implemented in simulink as well.
This function would then use the inputs R_r, ω_r, $\dot{\omega}_r$, f_r, $\hat{R}$, $\hat{\omega}$, $\hat{\rho}_e$, $\hat{\nu}_e$, z, ρ and ω. In this case, ρ and ω are obtained from the sensor measurements and the rest of the inputs are generated by a separate block. The outputs of the function would be τ and f, which are then guided to the green blocks in Figure 6.4.
Future research could go deeper into what the best way is to implement a custom controller in simulink. Whether using the Matlab function is indeed easier or if using simulink blocks is a better approach.

6.3 Concluding remarks

This chapter has described the simulink template model from the support package for Parrot mini drones. All the different system blocks have been explained and the location of the control logic within the template is shown. An example is given of how a custom controller could theoretically be implemented. With control logic implemented in this simulink template model, the FCS block can be loaded on the hardware to test the drone in the real world.

7 Connecting with the drone

With the reference dynamics and the controller tested in simulations, the next step is applying it to the hardware. The first step in doing so is connecting to the drone, this chapter goes into detail on how this connection should be achieved in Windows.

7.1 Connection in Windows

To connect with the drone the Matlab documentation can be followed [8]. In practice these steps are similar but not exactly the same. This is due to the different layout of different operating system versions. When the drone is connected for the first time, a special setup has to be done in Matlab. To do this the "Simulink support package for parrot mini drones" add-on has a setup program.

This setup program starts with connecting the drone via a micro USB cable. The wired connection is required for the computer to identify the drone initially. If the drone is connected correctly the program can detect the drone and the next step can be taken. This step updates the firmware on the drone. To do this the drone has to be disconnected, it starts updating automatically. After this step the drone should be ready to connect via Bluetooth. The next screen in the setup program references the menu "My Bluetooth Devices" within the windows explorer, however on this specific computer the references menu is empty and no buttons are able to be pressed. To add the device, the menu "Devices and Printers" has to be used. When "add a device" is pressed, the drone can be selected to be connected to. One of the two drones used gave an error message in this step, the other drone gave two different options to connect to, where one worked and the other option did not work. There was no clear reason why there was a difference between the two drones. The option that works is also the "joystick" referred to in the setup program. After the correct version of the drone is added to the devices and printers menu, the "connect using" button can be used to connect to the drone via Bluetooth. The setup program then gives a test screen where the connection can be tested, this is the final step in the initial setup of the drone. The specific screens and error messages are included in Appendix B.

At this moment, it is unclear why one drone works and the other does not. Multiple different drones were tried, but only one of them was able to make a connection with Bluetooth. Using Linux did not improve the connection, and the same problems were encountered. Another attempt was made by using a Bluetooth "dongel" to try to have a better signal from the computer. However, since one of the drones worked without this extension, it should not be necessary for the connection to work. A possible explanation is that one of the drones may have been altered by previous students to make the connection work, since the drone that connects correctly has been used before. Willem de Jonge was contacted about this issue, and he recognized it but did not remember how he solved it. It is also possible that the working drone is an older hardware model and thus has different hardware causing problems. Additionally, connecting to the drone using different computers was attempted, but they could not connect in the right way either.
There were also some issues with the drone that did connect correctly. When trying to run templates supplied by the support package, the data of the flights could not be logged. Since no data can be logged, further tests on the hardware would not give usable results.

7.2 Concluding remarks

It is unclear why the two drones give different results when trying to connect to them. A possible reason is that a previous student changed something in the code of the drone which made the connection possible. It would be useful for future research to dive deeper into why there is a difference between the drones at all. Another issue arose when testing with the drone that did connect, where the data from test flights could not be logged correctly. This makes running tests not useful since the results can not be analyzed.

8 Connection to Optitrack

To be able to use the Optitrack camera system as the localization method for the Parrot mini drone, a connection has to be established with the camera system. This can be done in different ways, depending on the application. For application within Matlab, a direct connection can be made using either the ROS toolbox from Matlab or using the NatNet SDK. For applications outside of Matlab, different steps are required. This chapter goes into detail how the Optitrack camera output data can be imported to Matlab.

This chapter is divided in three main sections, firstly a short overview is given of how a ROS system works in general. The next section focuses on a direct connection to Matlab in Windows, whereas the last section uses Ubuntu to connect to the Optitrack system. In the second and third section, the required software is shown and an explanation given on how to install it correctly. The section focusing on the direct connection to Matlab shows the modifications that are made to provided files to extract location data from the camera system and import it into Matlab. The section focusing on Ubuntu shows how to read live location data after the installation guide.

8.1 Workings ROS system

ROS can be used to have different components in a system communicate with each other. For example letting a drone communicate with a computer. Both the drone and the computer would be called "Nodes" within the ROS system. Different nodes can communicate with each other using a "Topic". A "Publisher" node can send messages to a topic, whereas a "Subscriber" node can receive these messages if it is subscribed to the specific topic. This concept is shown in Figure 8.1.

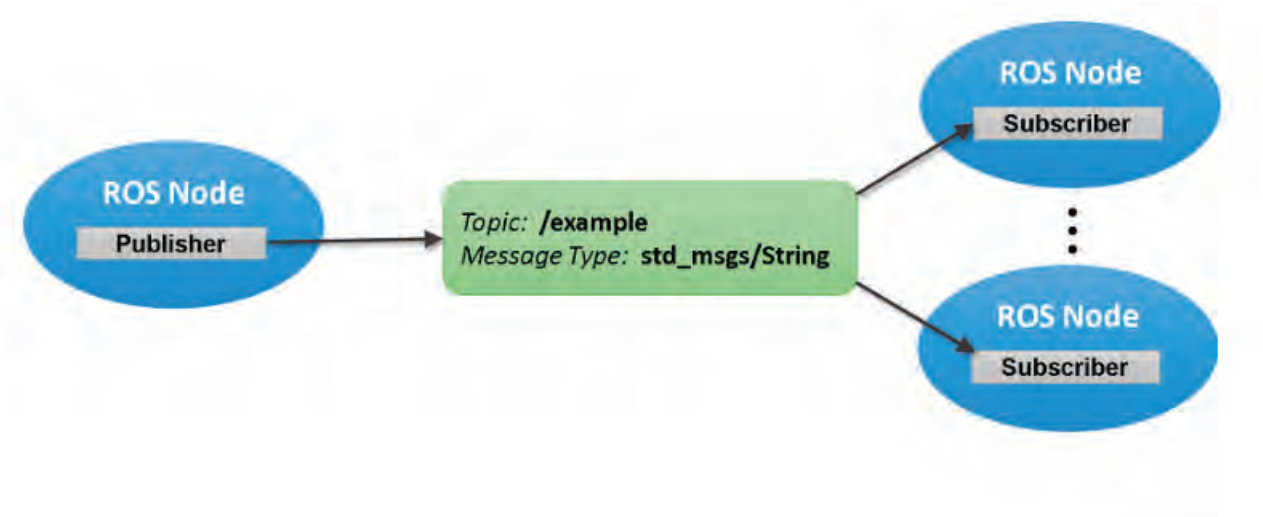

Figure 8.1: Illustration of the ROS communication concept from [14]

8.2 Optitrack to Matlab in Windows

This section focuses on connecting to the Optitrack camera system and importing location data from there into Matlab. Two ways are explored, using the ROS toolbox and using the NatNet SDK. Within these sections a guide is given on how to install the required software.

8.2.1 ROS toolbox

The ROS Matlab toolbox is required to be installed within Matlab. To make sure the toolbox can be used in the specific Matlab version, the right Python software needs to be installed. For the application in this thesis Matlab R2021a is used. A detailed overview of which version of Python is required for each version of Matlab can be found in the Matlab documentation under the section "ROS Toolbox System Requirements". For Matlab

version R2021a, the desired Python version is 2.7. To check the current version of Python on a system, use the command "*pyenv*" in the Matlab Command Window. This gives the result shown in Figure 8.2 if the python version is installed correctly.

```
>> pyenv

ans =

  PythonEnvironment with properties:

         Version: "2.7"
      Executable: "C:\Python27\pythonw.exe"
         Library: "C:\Python27\python27.dll"
            Home: "C:\Python27"
          Status: NotLoaded
   ExecutionMode: InProcess
```

Figure 8.2: Current version of Python on the system

If the command "*pyenv*" does not give this response, but instead shows either a different version or no version at all, the version can be modified using the command "$pyenv(Version = "2.7")$". The specific version used in this thesis is 2.7.18.
The specified version of Python has to be included in the path as well. This is done by one of two ways, either during the installation process of Python (Some versions give the option to add it to the path), or setting the path manually. To set the path manually, open the run dialog box (Windows key + r) and run the command *sysdm.cpl*. This opens the System properties menu. Navigate to "Advanced" and then "Environment Variables". This menu now shows the current user variables. A "Path" variable needs to be present here, showing the path to the Python version. If this variable is not present, it needs to be created. To do this, press "New...". For the variable Name, enter "Path". For the variable Value, enter the system path to the Python version installed on the system, and the path to the folder "scripts", separated by a ";", for example `C:\Python27; C:\Python27\Scripts`.
After these steps Matlab should be able to set the version of Python as the current one and the ROS toolbox can be used. To check this, the command "*rosinit*" can be used in the Matlab command window. This launches the ROS core, if the system does this without errors, the software is installed correctly. Use "*rosshutdown*" to shut down the ROS core.

During testing the approach of the ROS toolbox in Matlab was unsuccessful. The different nodes that are present in the system could not be found when using the toolbox. Because of this, another option to get the location data from the camera system is required. This other option is described in the next section.

8.2.2 NatNet SDK

To import the location data from the Optitrack camera system in Matlab, two different ways are possible. Using the ROS toolbox provided by Matlab, or using the NatNet SDK. This section shows how a connection is made using the latter. The ROS toolbox is not utilized in this thesis. The NatNet software can be downloaded from the Optitrack website [7].
Within the downloaded folder, different Matlab files are found within the `\NatNetSDK\Samples\Matlab` folder. The file "NatNetPollingSample.m" is used to get data from the Optitrack system. This function has to be modified to work with the specific camera setup used in this research. Within the code the following has to be changed:

natnetclient.HostIP = '*IP address of camera system*'
natnetclient.ClientIP = '*IP address of laptop running Matlab*'

The code prints the location of the rigid bodies that the cameras can see. However, this data cannot be used in calculations if the function is run this way, thus more modifications are necessary. The part of the function that is responsible for connecting to the system can be taken outside of the function. Taking this part out of the

function, a connection can be made before running the function. This is useful because the function does not try to connect every time it is run, speeding up the process of receiving the location data. An example layout of the function is given in Figure 8.3.

```
function Output = NatNetPollingSample(natnetclient,model)
data = natnetclient.getFrame;
Output.x = data.RigidBodies( 1 ).x ;
Output.y = data.RigidBodies( 1 ).y ;
Output.z = data.RigidBodies( 1 ).z ;

end
```

Figure 8.3: Matlab function to get data from Optitrack

In this example the variables "natnetclient" and "model" come from the code that connects to the camera system. I.e. $natnetclient = natnet$ and $model = natnetclient.getModelDescription$.
With these modifications to the code, the function can be called whenever the location data is required. This can then also be implemented in the controller logic simulation.

8.3 Ubuntu

The previous section has shown how to connect to the system for applications within Matlab. This section focuses on how a connection to the Optitrack camera system can be achieved for applications in Ubuntu.

Firstly some general information about the software used is given. Afterwards a guide is given on how to install the necessary software. This guide is in the form of terminal commands, all the responses given by the system have been left out for ease of reading. When everything is installed, a short guide is then shown to view the location streamed by the Optitrack system.

8.3.1 Software

The research in this report is done with the following versions of the software used.

- Ubuntu 20.04
- ROS Noetic Ninjemys, Released May, 2020
- Information on the network/streaming settings of the camera system can be found on the corresponding gitlab page made by Ö. Arslan [15].

8.3.2 Installation process

To make sure the right versions of all the packages and software is installed, follow the command list shown below.

```
Tests if installed correct:

bruno@S153072-Ubuntu:~$ roscore
bruno@S153072-Ubuntu:~$ gedit .bashrc
```

When the installation is not done correctly the following response is shown:

```
bruno@S153072-Ubuntu:~$ roscore

Command 'roscore' not found, but can be installed with:

sudo apt install python3-roslaunch

bruno@S153072-Ubuntu:~$ gedit .bashrc

(gedit:5721): Gtk-WARNING **: 15:20:10.908: GTK+ module /usr/lib/x86_64-linux-gnu/gtk-2.0/modules/libgail.so cannot be
      loaded.
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.
Gtk-Message: 15:20:10.908: Not loading module "atk-bridge": The functionality is provided by GTK natively. Please try
      to not load it.

```

```
(gedit:5721): Gtk-WARNING **: 15:20:11.043: GTK+ module /usr/lib/x86_64-linux-gnu/gtk-2.0/modules/libcanberra-gtk-module.so cannot be loaded.
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.
Gtk-Message: 15:20:11.043: Failed to load module "canberra-gtk-module"

(gedit:5721): Gtk-WARNING **: 15:20:11.044: GTK+ module /usr/lib/x86_64-linux-gnu/gtk-2.0/modules/libcanberra-gtk-module.so cannot be loaded.
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.
Gtk-Message: 15:20:11.044: Failed to load module "canberra-gtk-module"
```

If the installation was done correctly the given response is:

```
\textbf{bruno@S153072-Ubuntu:~$ roscore
... logging to /home/bruno/.ros/log/f1b64bd0-4bc4-11ed-8286-6fffbe0f5fc5/roslaunch-S153072-Ubuntu-23235.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://S153072-Ubuntu:44101/
ros_comm version 1.15.14


SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.15.14

NODES

auto-starting new master
process[master]: started with pid [23245]
ROS_MASTER_URI=http://S153072-Ubuntu:11311/

setting /run_id to f1b64bd0-4bc4-11ed-8286-6fffbe0f5fc5
process[rosout-1]: started with pid [23255]
started core service [/rosout]
^C[rosout-1] killing on exit
[master] killing on exit
shutting down processing monitor...
... shutting down processing monitor complete
done
```

To continue the correct installation process follow:

```

Installation:
(From http://wiki.ros.org/noetic/Installation/Ubuntu ):

bruno@S153072-Ubuntu:~$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
bruno@S153072-Ubuntu:~$ sudo apt install curl
bruno@S153072-Ubuntu:~$ curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
bruno@S153072-Ubuntu:~$ sudo apt update
bruno@S153072-Ubuntu:~$ sudo apt install ros-noetic-desktop-full
bruno@S153072-Ubuntu:~$ source /opt/ros/noetic/setup.bash
bruno@S153072-Ubuntu:~$ echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
bruno@S153072-Ubuntu:~$ sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
bruno@S153072-Ubuntu:~$ sudo apt install python3-rosdep
bruno@S153072-Ubuntu:~$ sudo rosdep init
bruno@S153072-Ubuntu:~$ rosdep update


(Another test, now installed correctly):
bruno@S153072-Ubuntu:~$ roscore
bruno@S153072-Ubuntu:~$ sudo apt install ros-noetic-vrpn-client-ros
bruno@S153072-Ubuntu:~$ mkdir catkin_make
bruno@S153072-Ubuntu:~$ cd catkin_make/
bruno@S153072-Ubuntu:~/catkin_make$ mkdir src
bruno@S153072-Ubuntu:~/catkin_make$ cd src/
bruno@S153072-Ubuntu:~/catkin_make$ sudo apt install ros-noetic-vrpn-client-ros
bruno@S153072-Ubuntu:~/catkin_make$ catkin_make
bruno@S153072-Ubuntu:~/catkin_make$ source devel/setup.bash

(Get launch file from https://github.com/ros-drivers/vrpn_client_ros/blob/kinetic-devel/launch/sample.launch)

(Connect with the Motion Capture Wifi)

bruno@S153072-Ubuntu:~/catkin_make$ roslaunch mocap mocap.launch
```

This list of commands is shortened for ease of understanding and so that people can easily copy it in their own terminal commands.

8.3.3 Live location tracking

Opening a new command terminal and using the commands below should then give the live location of the object, in this example, the name of the object is "turtlebot3".

```

bruno@S153072-Ubuntu:~$ rostopic list

/rosout
/rosout_agg
/tf
/vrpn_client_node/turtlebot3/pose

bruno@S153072-Ubuntu:~$ rostopic echo /vrpn_client_node/turtlebot3/pose

header:
  seq: 0
  stamp:
```

```
14        secs: 1665755653
15        nsecs: 287479433
16      frame_id: "world"
17    pose:
18      position:
19        x: 0.46207594871520996
20        y: 0.045421015471220016
21        z: 0.3110859990119934
22      orientation:
23        x: -0.0013987818965688348
24        y: 0.0022063779179006815
25        z: -0.8867406249046326
26        w: -0.4622599482536316
27  ---
28  header:
29    seq: 1
```

8.4 Concluding remarks

This chapter has shown how to use the NatNet SDK to get location data from the Optitrack camera system in Matlab. It is shown how to set up the connection and what needs to be modified in the provided Matlab files to make the connection possible. An example is given of how the location data can be obtained in a variable, which can then be used in the Matlab workspace.

This chapter has also shown how to receive the live location of a rigid body using the Optitrack camera system in Ubuntu. The used software is described and a short explanation is given of how a ROS system works conceptually. A detailed guide is then given on how the correct software is installed and how the live location can be received.

The next step would be to implement this location data into the control logic. Replacing the localization method of the Parrot mini drone, from its standard method of using internal sensors, to the Optitrack camera system providing the location data.

9 Conclusions and recommendations

This chapter concludes the thesis and gives recommendations for future research. Firstly a summary is given of what has been done in all the chapters, after which conclusions are drawn. Based on these conclusions some recommendations are given for future research.

9.1 Summary and conclusions

This thesis focuses mainly on the implementation of a controller on the Parrot Mambo fly drone, using Optitrack to localize the drone. In the first chapter, background information is given and the problem is defined. Multiple goals are set to be achieved in the thesis. These goals were to describe the behaviour of a quad copter drone, determine a reference trajectory, implement a controller in simulations, connect to both the drone hardware and the camera system and achieving flight with the drone.

Chapter 2 showed the behaviour of a drone and dove deep into constructing a reference trajectory based on the dynamics of a drone. This chapter has shown that most of the reference trajectory follows from four flat output signals, these signals need to be chosen. Three different flat output signals were discussed in detail and a reference trajectory was derived. The reference trajectory dynamics were shown in enough detail that they can be copied for simulations without the need for further calculations. The reference trajectories determined in this chapter are later used for simulations.

Chapter 3 focused on the controller that was used in this thesis, the one designed by Lefeber et at. [11]. The position and attitude controller parts were described in detail, after which they were combined to give the final expression for the controller. Similar to the reference trajectory, the expressions given in this chapter are in enough detail, that they can be used in simulations without the need for further calculations. This controller is used later for simulations.

The reference trajectories and the controller are both used in simulations in chapter 4. This chapter shows the initial conditions used and the gains that were chosen for the simulations. The controller is then used to follow the three different reference trajectories. During these simulations interesting behaviour was discovered. The controller was able to follow the hover reference, where a peak in the attitude rate was discovered. Since this movement was not expected, a more detailed look was taken to see what caused this movement. This was done by splitting the controller into the translational part and the attitude part, which are then discussed in detail. This detailed analysis showed that both parts work separately.
The second reference, the back and forth movement, showed similar behaviour, where the controller was able to follow the reference but the attitude rate seemed to move in the opposite direction of the reference at the start of the simulation, with a peak present at the same moment in time as with the hover reference. The same analysis was done as with the hover reference, splitting up the controller. This showed the attitude controller not being able to reduce the error to zero, but following the reference relatively close.
The third reference trajectory, the one used in [11], showed that the controller is able to follow a complicated trajectory. However it is not able to reduce the error to zero with the parameters used. The attitude rate showed a big peak at the start of the simulation, which is suspected to be a consequence of the same thing that caused the peak in the hover reference and the opposite movement in the second reference.
A possible cause for these results was investigated by re-calculating the value of ω_d, it was assumed that f_d and R_d were correct and ω_{dis} was derived from them. This resulted in a different expression than ω_d. ω_{dis} was then implemented in the simulations and the hover reference was used again to see the difference when using ω_{dis} instead of ω_d. The peak in the attitude rate for the hover reference was removed, as well as the peak with the back and forth reference. The movement in the opposite direction of the reference in this second reference was not removed. It was concluded that the assumption of f_d and R_d being correct was wrong. Indicating that these need to be derived again to make sure the expressions are correct. This is taken as a recommendation for future research.

Chapter 5 showed an overview of the camera system setup and gave a description of the context surrounding the hardware used in this thesis.

Chapter 6 described the simulink support package template in detail. This template can be used to load flight code on the hardware drone to implement controllers. This chapter described all the different components of the template and showed which parts are the most relevant. After this a theoretical approach is given on how a custom controller could be implemented within this template.

Chapter 7 focused on connecting to the Parrot Mambo fly drone. A detailed description was given on how the connection should be achieved. Multiple problems arose in this section, one of which was that a connection to the drone was not possible. Different versions of the same drone were tried and only one was able to connect. It was not discovered why the different versions of the drone gave different results when connecting to them. A possible explanation is that a previous student changed something in the firmware to make it possible to connect, but this was not confirmed. The drone that was able to connect had a different issue, namely that flight data was not able to be logged. Because of this, tests on the hardware would not give useful results. This is determined as another recommendation to investigate further in future research.

Chapter 8 focused on connecting to the Optitrack camera system. Three different ways of connecting were explored, the first two in Windows and the third one in Ubuntu. Firstly the ROS toolbox in Matlab was tried, a detailed description was given of how to install the right software, but a connection using this method could not be achieved. Because of this a second option was required, using the NatNet SDK. A guide was given to install the necessary software and how to modify files to make the connection. This method was used successfully and a connection was made. The location data was successfully imported into Matlab and could be used in calculations. The third method of connecting was done in Ubuntu. A detailed guide consisting of terminal commands was given on how to install the correct software and how to read the live location data from the Optitrack system. This method was also completed successfully.

Looking back at the goals set at the start of this thesis, most of them were completed. The reference trajectory was determined from flat output signals, the controller was described in detail. Both the reference and controller were implemented in simulations successfully. A connection to the camera system was achieved in more than one way, but the drone hardware caused issues. Because the flight data could not be logged and multiple drones could not establish a connection. This made it so that tests would not yield usable results and thus the controller was not implemented on the hardware.

The structure of the report is built in such a way that the individual chapters can be used separate from each other. This would enable future research to use parts of this thesis without the need for the same specific hardware. For example a different drone could be used, where the reference and controller description is still relevant, just as the connection guide for the camera system.

9.2 Recommendations

This section gives examples that could be interesting to investigate in future research.
As touched on before, there were considerable issues with the hardware used in this thesis. Therefore it is recommended to either investigate the source of the issues described in this thesis, or use a different drone as the base to implement the controller on. If a different drone is used, the remaining chapters of this thesis can still be used, as they are written in general form. The implementation in Matlab needs to be modified slightly in this case, namely the mass of the drone and the inertia matrix.

Another point of interest are the expressions for f_d, R_d and ω_d. As described in chapter 4, these expressions might cause unwanted behaviour in the attitude rate. It is therefore recommended to check these expressions and make sure they are correct.

Since this thesis did not fully implement the controller in simulink and on the hardware, it is recommended to further investigate how this implementation is done. The theoretical implementation given in chapter 6 can be used as a base for this, although it is not tested. With this it could be interesting to see if there is a way of getting the live location data directly in simulink, so the step of implementing it in Matlab first can be skipped. A thing to note here is that it is recommended to investigate how the flight code communicates with the camera system after implementation. For example, is the drone receiving the data directly from the camera system or is the laptop that is connected to the drone forwarding this information.

References

[1] J. Alkobi, *The Evolution of Drones: From Military to Hobby & Commercial.* Article written on January 15, 2019. Online: https://percepto.co/the-evolution-of-drones-from-military-to-hobby-commercial/

[2] S.J.A.M. van den, Eijnden, *Cascade Based Tracking Control of Quadrotors*, DC 2017.012, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, MSc Thesis, 2017.

[3] G.H, Brekelmans, *Extended Quadrotor Dynamics: from Simulations to Experiments*, DC2019.090, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, MSc Thesis, 2019.

[4] M. SungTae, C. DongHyun, H. Sanghyuck, R. DongYoung, and S. Eun-Sup, *Development of Multiple AR.Drone Control System for Indoor Aerial Choreography**, Aerospace Convergence Technology Team, Korea Aerospace Research Institute, Daejeon, Korea, January 24th, 2014.

[5] P. Zhaoy, C. Xiaoxuan Lux, B. Wangy, N. Trigoniy, and A. Markhamy, *3-D Motion Capture of an Unmodified Drone with Single-chip Millimeter Wave Radar*, Department of Computer Science, University of Oxford, United Kingdom, xSchool of Informatics, University of Edinburgh, United Kingdom, 13 Nov 2020.

[6] S. Al Habsi, M. Shehada, M. Abdoon, A. Mashood, H. Noura, *Integration of a Vicon Camera System for Indoor Flight of a Parrot AR Drone*, Department of Electrical Engineering, Research paper, UAE University Al Ain, Abu Dhabi, UAE, 07 January 2016.

[7] Optitrack, Online: https://www.optitrack.com

[8] Matlab documentation for the simulink package for parrot mini drones, Online: https://nl.mathworks.com/help/supportpkg/parrot/setup-and-configuration.html

[9] X. Zeng, *Implementing Tracking Error Control for Quadrotor UAV*, DC2021.044, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, MSc Thesis, 2021.

[10] W. de Jonge, *Social behavior in a network of UAVs with collision avoidance*, DC2020.093, Eindhoven University of Technology, Dynamics and Control Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, MSc Thesis, 2020.

[11] E. Lefeber, M. Greiff, A. Robertsson, *Filtered Output Feedback Tracking Control of a Quadrotor UAV*, In: Proceedings of the 21st IFAC World Congress, Berlin, Germany, 2020.

[12] E. Lefeber, M.F.A. van de Westerlo, H. Nijmeijer, *Almost global decentralised formation tracking for multiple distinct UAVs*, Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands. IFAC PapersOnLine 52-16 (2019) 186–191

[13] Khalil, Hassan K, *Textbook: Nonlinear systems; 3rd ed.*, Prentice-Hall, Upper Saddle River, NJ, 2002.

[14] Matlab documentation for ROS, Online: https://nl.mathworks.com/help/ros/ug/exchange-data-with-ros-publishers-and-subscribers.html

[15] Ö. Arslan, gitlab page, Online: https://gitlab.tue.nl/20195150/dsd\char‘_mocap\char‘_optitrack/-/wikis/home

A Detailed derivation reference trajectory

This chapter compliments chapter 2.3 by explaining the derivations in more detail.

The expressions for ϕ_r and θ_r in terms of ψ_r are used to fill in R_r. This gives the expression:

$$R_{r(:,1)} = \begin{bmatrix} \frac{r_3 \cos\psi}{\sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}} \\ \frac{r_3 \sin\psi}{\sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}} \\ \frac{-(r_1\cos\psi + r_2\sin\psi)}{\sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}} \end{bmatrix}, \tag{A.1a}$$

$$R_{r(:,2)} = \begin{bmatrix} -\sin\psi(\sqrt{1+(r_1\sin\psi_r - r_2\cos\psi_r)}) - \frac{(\cos\psi(r_1\cos\psi + r_2\sin\psi))(r_2\cos\psi_r - r_1\sin\psi)}{\sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}} \\ \cos\psi(\sqrt{1+(r_1\sin\psi_r - r_2\cos\psi_r)}) - \frac{(\sin\psi(r_1\cos\psi + r_2\sin\psi))(r_2\cos\psi_r - r_1\sin\psi)}{\sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}} \\ \frac{r_3(r_1\sin\psi_r - r_2\cos\psi_r)}{\sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}} \end{bmatrix}, \tag{A.1b}$$

$$R_{r(:,3)} = \begin{bmatrix} \frac{\cos\psi(r_1\cos\psi + r_2\sin\psi)\sqrt{1+(r_1\sin\psi_r - r_2\cos\psi_r)}}{\sin\psi(r_1\sin\psi_r - r_2\cos\psi_r) + \sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}} \\ -\cos\psi(r_1\sin\psi_r - r_2\cos\psi_r) + \frac{\sin\psi(r_1\cos\psi + r_2\sin\psi)\sqrt{1+(r_1\sin\psi_r - r_2\cos\psi_r)}}{\sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}} \\ \frac{r_3\sqrt{1+(r_1\sin\psi_r - r_2\cos\psi_r)}}{\sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}} \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}. \tag{A.1c}$$

To determine the derivative of the rotation matrix, the derivative of the expressions for ϕ_r and θ_r are required. These derivatives are as follows

$$\begin{aligned} \frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3} &= \sin\psi_r(\dot{r_1} + r_2\dot{\psi_r}) + \cos\psi_r(r_1\dot{\psi_r} - \dot{r_2}) \\ \frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3} &= \frac{(\sin\phi_r)(\frac{d\sin\phi_r}{d\psi dr_1 dr_2 dr_3})}{(1-(\sin\phi_r)^2)^{\frac{3}{2}}} \\ \frac{d\cos\theta_r}{d\psi_r dr_1 dr_2 dr_3} &= \frac{\dot{r_3}\cos\phi_r - r_3(\frac{d\cos\phi_r}{d\psi dr_1 dr_2 dr_3})}{(\cos\phi_r)^2} \\ \frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3} &= \frac{\cos\phi_r(\sin\psi_r(\dot{r_2} - r_1\dot{\psi_r}) + \cos\psi_r(\dot{r_1} + r_2\dot{\psi_r})) - (r_1\cos\psi_r + r_2\sin\psi_r)(\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3})}{(\cos\phi_r)^2} \end{aligned} \tag{A.2}$$

The derivative of the rotation matrix is then

$$\frac{dR_{r(:,1)}}{d\psi_r dr_1 dr_2 dr_3} = \begin{bmatrix} -\frac{\sin\psi_r \cdot r_3\dot{\psi_r}}{\sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}} + \cos\psi_r(\frac{d\cos\theta_r}{d\psi_r dr_1 dr_2 dr_3}) \\ (\frac{d\cos\theta_r}{d\psi_r dr_1 dr_2 dr_3})\sin\psi + (\frac{r_3}{\sqrt{1-(r_1\sin\psi_r - r_2\cos\psi_r)^2}})\cos\psi\dot{\psi} \\ -(\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3}) \end{bmatrix}, \tag{A.3a}$$

$$\frac{dR_{r(:,2)}}{d\psi_r dr_1 dr_2 dr_3} = \begin{bmatrix} \dot{R}_{r(1,2)} \\ \dot{R}_{r(2,2)} \\ \dot{R}_{r(3,2)} \end{bmatrix}, \tag{A.3b}$$

$$\frac{dR_{r(:,3)}}{d\psi_r dr_1 dr_2 dr_3} = \begin{bmatrix} \dot{r_1} \\ \dot{r_2} \\ \dot{r_3} \end{bmatrix}. \tag{A.3c}$$

where

$$\dot{R}_{r(1,2)} = -\sin\psi_r \sin\phi_r \sin\theta_r \dot{\psi}_r + \cos\psi_r (\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \sin\theta_r + \cos\psi_r \sin\phi_r (\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3}) - ((\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \sin\psi_r + \cos\phi_r \cos\psi_r \dot{\psi}_r), \tag{A.4a}$$

$$\dot{R}_{r(2,2)} = (\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \cos\psi - \cos\phi \sin\psi \dot{\psi} + (\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \sin\psi \sin\theta + \sin\phi \cos\psi \sin\theta \dot{\psi} + \sin\phi \sin\psi (\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3}), \tag{A.4b}$$

$$\dot{R}_{r(3,2)} = (\frac{d\cos\theta_r}{d\psi_r dr_1 dr_2 dr_3}) \sin\phi + \cos\theta (\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3}), \tag{A.4c}$$

and

$$\dot{r}_1 = -x_r^{(3)} (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{1}{2}} + \frac{1}{2} \ddot{x}_r (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{3}{2}} (2\ddot{x}_r x_r^{(3)} + 2\ddot{y}_r y_r^{(3)} + (2\ddot{z}_r - 2) z_r^{(3)}), \tag{A.5a}$$

$$\dot{r}_2 = -y_r^{(3)} (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{1}{2}} + \frac{1}{2} \ddot{y}_r (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{3}{2}} (2\ddot{x}_r x_r^{(3)} + 2\ddot{y}_r y_r^{(3)} + (2\ddot{z}_r - 2) z_r^{(3)}), \tag{A.5b}$$

$$\dot{r}_3 = -z_r^{(3)} (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{1}{2}} + (\ddot{z}_r - g)(\ddot{x}_r x_r^{(3)} + \ddot{y}_r y_r^{(3)} + (\ddot{z}_r - g) z_r^{(3)}) (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{3}{2}}. \tag{A.5c}$$

This concludes the calculation of the derivative of the rotation matrix. The next step is to calculate the second derivative of the rotation matrix. The first step is do take the derivatives of r_1, r_2 and r_3. This is done below by splitting up the partial derivatives and adding them up afterwards for ease of reading. Here $x^{(i)^2}$ is the i-th derivative to the power 2.

$$\frac{d\dot{r}_1}{dx_r} = -x_r^{(4)} (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{1}{2}} + (3\ddot{x}_r x_r^{(3)^2} + \ddot{x}_r^2 x_r^{(4)}) (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{3}{2}} - 3\ddot{x}_r^3 x_r^{(3)^2} (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{5}{2}} \tag{A.6a}$$

$$\frac{d\dot{r}_1}{dy_r} = (\ddot{x}_r (\ddot{y}_r y_r^{(4)} + y_r^{(3)^2}) + x_r^{(3)} \ddot{y}_r y_r^{(3)}) (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{3}{2}} - 3\ddot{x}_r \ddot{y}_r^2 y_r^{(3)^2} (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{5}{2}} \tag{A.6b}$$

$$\frac{d\dot{r}_1}{dz_r} = (x_r^{(3)} z_r^{(3)} (\ddot{z}_r - 1) + \ddot{x}_r (\ddot{z}_r z_r^{(4)} + z_r^{(3)^2} - z_r^{(4)})) (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{3}{2}} - 3\ddot{x}_r (\ddot{z}_r^2 z_r^{(3)^2} - 2\ddot{z}_r z_r^{(3)^2} + z_r^{(3)^2}) (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{5}{2}} \tag{A.6c}$$

$$\ddot{r}_1 = \frac{d\dot{r}_1}{dx_r} + \frac{d\dot{r}_1}{dy_r} + \frac{d\dot{r}_1}{dz_r} \tag{A.6d}$$

$$\frac{d\dot{r}_2}{dx_r} = (\ddot{y}_r (\ddot{x}_r x_r^{(4)} + x_r^{(3)^2}) + y_r^{(3)} \ddot{x}_r x_r^{(3)}) (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{3}{2}} - 3\ddot{y}_r \ddot{x}_r^2 x_r^{(3)^2} (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{5}{2}} \tag{A.7a}$$

$$\frac{d\dot{r}_2}{dy_r} = -y_r^{(4)} (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{1}{2}} + (3\ddot{y}_r y_r^{(3)^2} + \ddot{y}_r^2 y_r^{(4)}) (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{3}{2}} - 3\ddot{y}_r^3 y_r^{(3)^2} (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{5}{2}} \tag{A.7b}$$

$$\frac{d\dot{r}_2}{dz_r} = (y_r^{(3)} z_r^{(3)} (\ddot{z}_r - 1) + \ddot{y}_r (\ddot{z}_r z_r^{(4)} + z_r^{(3)^2} - z_r^{(4)})) (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{3}{2}} - 3\ddot{y}_r (\ddot{z}_r^2 z_r^{(3)^2} - 2\ddot{z}_r z_r^{(3)^2} + z_r^{(3)^2}) (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{5}{2}} \tag{A.7c}$$

$$\ddot{r}_2 = \frac{d\dot{r}_2}{dx_r} + \frac{d\dot{r}_2}{dy_r} + \frac{d\dot{r}_2}{dz_r} \tag{A.7d}$$

$$\frac{d\dot{r}_3}{dx_r} = (\ddot{x}_r x_r^{(3)} z_r^{(3)} + (\ddot{z}_r - g)(\ddot{x}_r x_r^{(4)} + x_r^{(3)^2})) (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{3}{2}} - 3\ddot{x}_r x_r^{(3)} (\ddot{z}_r - g) (\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{5}{2}} \tag{A.8a}$$

$$\frac{d\dot{r}_3}{dy_r} = (\dddot{y}_r y_r^{(3)} z_r^{(3)} + (\ddot{z}_r - g)(\dddot{y}_r y_r^{(4)} + {y_r^{(3)}}^2))(\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{3}{2}} - 3\dddot{y}_r y_r^{(3)}(\ddot{z}_r - g)(\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{5}{2}} \quad \text{(A.8b)}$$

$$\begin{aligned}\frac{d\dot{r}_3}{dz_r} = {} & -z_r^{(4)}(\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{1}{2}} \\ & + ({z_r^{(3)}}^2(\ddot{z}_r - g) + z_r^{(3)}(\ddot{x}_r x_r^{(3)} + \ddot{y}_r y_r^{(3)} + z_r^{(3)}(\ddot{z}_r - g)) + ({z_r^{(3)}}^2 + \ddot{z}_r z_r^{(4)} - g z^{(4)})(\ddot{z}_r - g))(\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{5}{2}} \\ & - 3z_r^{(3)}(\ddot{z}_r - g)^2(\ddot{x}_r x_r^{(3)} + \ddot{y}_r y_r^{(3)} + z_r^{(3)}(\ddot{z}_r - g))(\ddot{x}_r^2 + \ddot{y}_r^2 + (g - \ddot{z}_r)^2)^{-\frac{5}{2}}\end{aligned} \quad \text{(A.8c)}$$

$$\ddot{r}_3 = \frac{d\dot{r}_3}{dx_r} + \frac{d\dot{r}_3}{dy_r} + \frac{d\dot{r}_3}{dz_r} \quad \text{(A.8d)}$$

The next step to complete the second derivative of the rotation matrix, is to find the second derivatives of the expressions for $\sin\phi_r$, $\cos\phi_r$, $\cos\theta_r$ and $\sin\theta_r$:

$$\begin{aligned}\frac{d(\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3} = {} & (\cos\psi_r(\dot{r}_1 + r_2\dot{\psi}_r) - \sin\psi(r_1\dot{\psi}_r - \dot{r}_2))\dot{\psi}_r \\ & + (r_1\cos\psi_r + r_2\sin\psi_r)\ddot{\psi}_r + \ddot{r}_1\sin\psi_r + \dot{r}_1\cos\psi_r\dot{\psi}_r + \dot{r}_2\sin\psi_r\dot{\psi}_r - \ddot{r}_2\cos\psi_r\end{aligned} \quad \text{(A.9)}$$

$$\begin{aligned}\frac{d(\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3} = {} & ((\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3})^2 + \sin\phi_r(\frac{d(\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3}))(1 - (\sin\phi_r)^2)^{-\frac{3}{2}} \\ & - 3(\sin\phi_r)^2(\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3})^2(1 - (\sin\phi_r)^2)^{-\frac{5}{2}}\end{aligned} \quad \text{(A.10)}$$

$$\frac{d(\frac{d\cos\theta_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3} = \frac{(\cos\phi_r)^2(\ddot{r}_3\cos\phi_r - r_3(\frac{d(\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3})) - 2(\dot{r}_3\cos\phi_r - r_3(\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}))\cos\phi(\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3})}{(\cos\phi_r)^4} \quad \text{(A.11)}$$

from equation (A.2) it is known that

$$\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3} = \frac{a + b}{(\cos\phi_r)^2} \quad \text{(A.12)}$$

where

$$a = \cos\phi_r(\sin\psi_r(\dot{r}_2 - r_1\dot{\psi}_r) + \cos\psi_r(\dot{r}_1 + r_2\dot{\psi}_r)) \quad \text{(A.13)}$$

and

$$b = -(r_1\cos\psi_r + r_2\sin\psi_r)(\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \quad \text{(A.14)}$$

$$\begin{aligned}\frac{da}{d\psi} = {} & (\frac{d\cos\phi_r}{d\psi_r})(\sin\psi_r(\dot{r}_2 - r_1\dot{\psi}_r) + \cos\psi_r(\dot{r}_1 + r_2\dot{\psi}_r)) \\ & + \cos\phi_r(\cos\psi_r(\dot{r}_2 - r_1\dot{\psi}_r)\dot{\psi}_r - r_1\sin\psi_r\ddot{\psi}_r + r_2\cos\psi_r\ddot{\psi}_r - \sin\psi_r(\dot{r}_1 + r_2\dot{\psi}_r)\dot{\psi}_r)\end{aligned} \quad \text{(A.15)}$$

$$\frac{db}{d\psi} = -(-r_1\sin\psi_r\dot{\psi}_r + r_2\cos\psi_r\dot{\psi}_r)(\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}) - (r_1\cos\psi_r + r_2\sin\psi_r)(\frac{d\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}}{d\psi_r}) \quad \text{(A.16)}$$

$$\frac{da}{dr_1} = (\frac{d\cos\phi_r}{dr_1})(\sin\psi_r(\dot{r}_2 - r_1\dot{\psi}_r) + \cos\psi_r(\dot{r}_1 + r_2\dot{\psi}_r)) + \cos\phi_r(-\dot{r}_1\sin\psi_r\dot{\psi}_r + \ddot{r}_1\cos\psi_r) \quad \text{(A.17)}$$

$$\frac{db}{dr_1} = -\dot{r}_1\cos\psi_r(\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}) - (r_1\cos\psi_r + r_2\sin\psi_r)(\frac{d\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}}{dr_1}) \quad \text{(A.18)}$$

$$\frac{da}{dr_2} = (\frac{d\cos\phi_r}{dr_2})(\sin\psi_r(\dot{r}_2 - r_1\dot{\psi}_r) + \cos\psi_r(\dot{r}_1 + r_2\dot{\psi}_r)) + \cos\phi_r(\ddot{r}_2\sin\psi_r + \dot{r}_2\dot{\psi}_r\cos\psi_r) \quad \text{(A.19)}$$

$$\frac{db}{dr_2} = -\dot{r}_2\sin\psi_r(\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}) - (r_1\cos\psi_r + r_2\sin\psi_r)(\frac{d\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}}{dr_2}) \quad \text{(A.20)}$$

$$\frac{da}{dr_3} = 0 \tag{A.21}$$

$$\frac{db}{dr_3} = 0 \tag{A.22}$$

$$\frac{da}{d\psi_r dr_1 dr_2 dr_3} = \frac{da}{d\psi} + \frac{da}{dr_1} + \frac{da}{dr_2} + \frac{da}{dr_3} \tag{A.23}$$

$$\frac{db}{d\psi_r dr_1 dr_2 dr_3} = \frac{db}{d\psi_r} + \frac{db}{dr_1} + \frac{db}{dr_2} + \frac{db}{dr_3} \tag{A.24}$$

$$\frac{d(\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3} = \frac{(\cos\phi_r)^2(\frac{da}{d\psi_r dr_1 dr_2 dr_3} + \frac{db}{d\psi_r dr_1 dr_2 dr_3}) - 2(a+b)\cos\phi(\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3})}{(\cos\phi_r)^4} \tag{A.25}$$

With these two derivations done, all the parts to construct the second derivative of the rotation matrix are obtained. Below the expression of every entry of the the second derivative of the rotation matrix is shown.

$$\ddot{R}_{r(1,1)} = -2\sin\psi_r(\frac{d\cos\theta_r}{d\psi_r dr_1 dr_2 dr_3})\dot{\psi}_r + \cos\psi_r(\frac{d(\frac{d\cos\theta_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3}) - \cos\psi_r\cos\theta\dot{\psi}_r^2 - \sin\psi_r\cos\theta\ddot{\psi}_r \tag{A.26}$$

$$\ddot{R}_{r(2,1)} = -\sin\psi_r\cos\theta\dot{\psi}_r^2 + 2\cos\psi_r(\frac{d\cos\theta_r}{d\psi_r dr_1 dr_2 dr_3})\dot{\psi}_r + \cos\psi_r\cos\theta\ddot{\psi}_r + \sin\psi_r(\frac{d(\frac{d\cos\theta_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3}) \tag{A.27}$$

$$\ddot{R}_{r(3,1)} = -(\frac{d(\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3}) \tag{A.28}$$

$$\ddot{R}_{r(1,3)} = \ddot{r}_1 \tag{A.29}$$

$$\ddot{R}_{r(2,3)} = \ddot{r}_2 \tag{A.30}$$

$$\ddot{R}_{r(3,3)} = \ddot{r}_3 \tag{A.31}$$

from equation (A.4a)

$$\dot{R}_{r(1,2)} = a + b + c - (d + e) \tag{A.32}$$

with

$$\begin{aligned} a &= -\sin\psi_r\sin\phi_r\sin\theta_r\dot{\psi}_r \\ b &= \cos\psi_r(\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3})\sin\theta_r \\ c &= \cos\psi_r\sin\phi_r(\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3}) \\ d &= (\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3})\sin\psi_r \\ e &= \cos\phi_r\cos\psi_r\dot{\psi}_r \end{aligned} \tag{A.33}$$

then the derivative becomes

$$\ddot{R}_{r(1,2)} = \dot{a} + \dot{b} + \dot{c} - (\dot{d} + \dot{e}) \tag{A.34}$$

with

$$
\begin{aligned}
\dot{a} &= -\cos\psi_r \sin\phi_r \sin\theta_r \dot{\psi}_r^2 - \sin\psi_r (\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \sin\theta_r \dot{\psi}_r - \sin\psi_r \sin\phi_r (\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3}) \dot{\psi}_r - \sin\psi_r \sin\phi_r \sin\theta_r \ddot{\psi}_r \\
\dot{b} &= -\sin\psi_r (\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \sin\theta_r \dot{\psi}_r + \cos\psi_r (\frac{d(\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3}) \sin\theta_r + \cos\psi_r (\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3}) (\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3}) \\
\dot{c} &= -\sin\psi_r \sin\phi_r (\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3}) \dot{\psi}_r + \cos\psi_r (\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3}) (\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3}) + \cos\psi_r \sin\phi_r (\frac{d(\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3}) \\
\dot{d} &= -\cos\psi_r (\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \dot{\psi}_r - \sin\psi_r (\frac{d(\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3}) \\
\dot{e} &= \sin\psi_r \cos\phi_r \dot{\psi}_r^2 - \cos\psi_r (\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \dot{\psi}_r - \cos\psi_r \cos\phi_r \ddot{\psi}_r
\end{aligned}
\tag{A.35}
$$

from equation (A.4b)

$$
\dot{R}_{r(2,2)} = a + b + c + d + e \tag{A.36}
$$

with

$$
\begin{aligned}
a &= (\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \cos\psi_r \\
b &= -\cos\phi_r \sin\psi_r \dot{\psi}_r \\
c &= (\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \sin\psi_r \sin\theta_r \\
d &= \sin\phi_r \cos\psi_r \sin\theta_r \dot{\psi}_r \\
e &= \sin\phi_r \sin\psi_r (\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3})
\end{aligned}
\tag{A.37}
$$

then the derivative becomes

$$
\ddot{R}_{r(2,2)} = \dot{a} + \dot{b} + \dot{c} + \dot{d} + \dot{e} \tag{A.38}
$$

with

$$
\begin{aligned}
\dot{a} &= (\frac{d(\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3}) \cos\psi_r - (\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \sin\psi_r \dot{\psi}_r \\
\dot{b} &= -(\frac{d\cos\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \sin\psi_r \dot{\psi}_r - \cos\phi_r \cos\psi_r \dot{\psi}_r^2 - \cos\phi_r \sin\psi_r \ddot{\psi}_r \\
\dot{c} &= (\frac{d(\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3}) \sin\psi_r \sin\theta_r + (\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \cos\psi_r \sin\theta_r \dot{\psi}_r + (\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \sin\psi_r (\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3}) \\
\dot{d} &= (\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \cos\psi_r \sin\theta_r \dot{\psi}_r - \sin\phi_r \sin\psi_r \sin\theta_r \dot{\psi}_r^2 + \sin\phi_r \cos\psi_r (\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3}) \dot{\psi}_r + \sin\phi_r \cos\psi_r \sin\theta_r \ddot{\psi}_r \\
\dot{e} &= (\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3}) \sin\psi_r (\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3}) + \sin\phi_r \cos\psi_r (\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3}) \dot{\psi}_r + \sin\phi_r \sin\psi_r (\frac{d(\frac{d\sin\theta_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3})
\end{aligned}
\tag{A.39}
$$

$$
\ddot{R}_{r(3,2)} = \sin\phi_r (\frac{d(\frac{d\cos\theta_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3}) + \cos\theta (\frac{d(\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3})}{d\psi_r dr_1 dr_2 dr_3}) + 2(\frac{d\sin\phi_r}{d\psi_r dr_1 dr_2 dr_3}) (\frac{d\cos\theta_r}{d\psi_r dr_1 dr_2 dr_3}) \tag{A.40}
$$

Which concludes the full derivation of the second derivative of the rotation matrix in detail.

B Connection to the drone Figures

This setup starts with the screen in Figure B.1. A connection with a wire is necessary, this allows the computer to identify the drone, which is shown in Figure B.2. This step then writes the Matlab firmware to the drone. For the drone to update this new firmware the drone has to be disconnected, as shown in Figure B.3. After this update the drone should be ready to connect to the computer using Bluetooth, this is described in Figure B.4. This screen references the menu "My Bluetooth Devices" within the windows explorer, however in this case the references menu is empty and no buttons are able to be pressed, as shown in Figure B.5. To add the device, the menu "Devices and Printers" has to be used as shown in Figure B.6. When selecting "Add a device", the menu shown in Figure B.7 pops up. Here the drone can be selected and the connection made, however, for some reason this gives an error, as shown in Figure B.8 and B.9. These errors seem to be two outcomes of the connection process, although there is no difference in the steps taken before. There is also no option to enter a password, making the error of Figure B.9 somewhat confusing.

When using a different drone, the same steps can be taken as before, using Figures B.1 - B.6. In this case the menu for adding a device looks different, as shown in Figure B.10. Here there are two different devices with the same name as the drone. One of these two is the "Joystick" referenced in Figure B.4, the other one does not work. Figure B.11 shows that the device is added, without error. In this case the wrong icon of the two was added, thus the other one has to be added too, as shown in Figure B.12. Figure B.13 shows that now both devices with the same name are added, where only one of them has the added option in the menu "Connect using", this is the correct one. This "Connect using" option gives the result that is referenced in the setup in Figure B.14. When this step is completed the drone is connected and the connection can be tested as in Figure B.15.

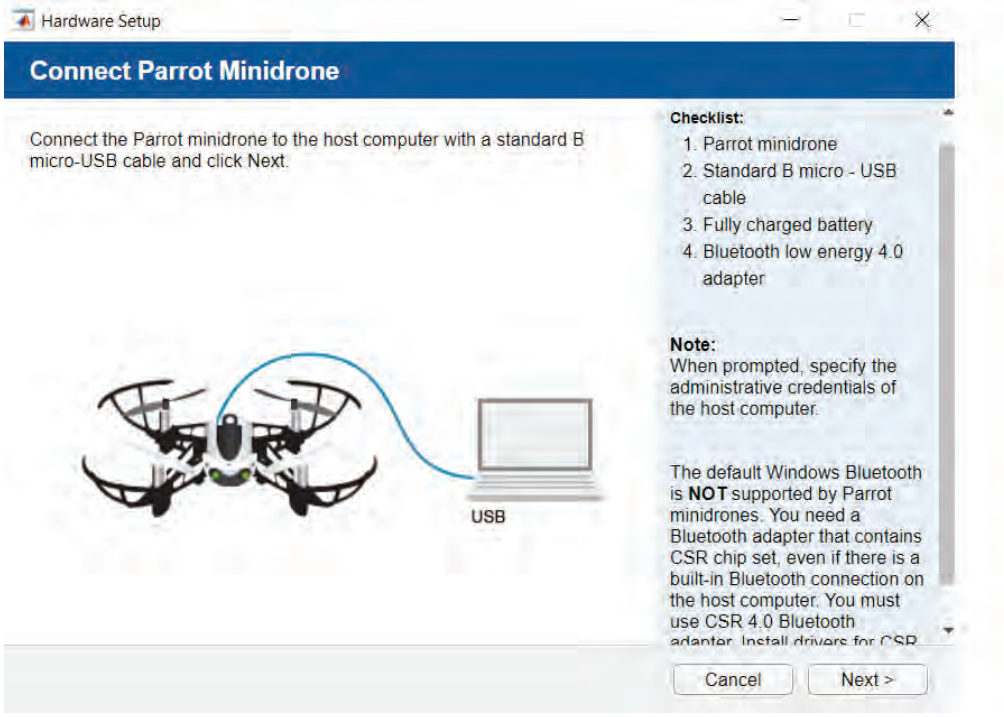

Figure B.1: Setup start screen

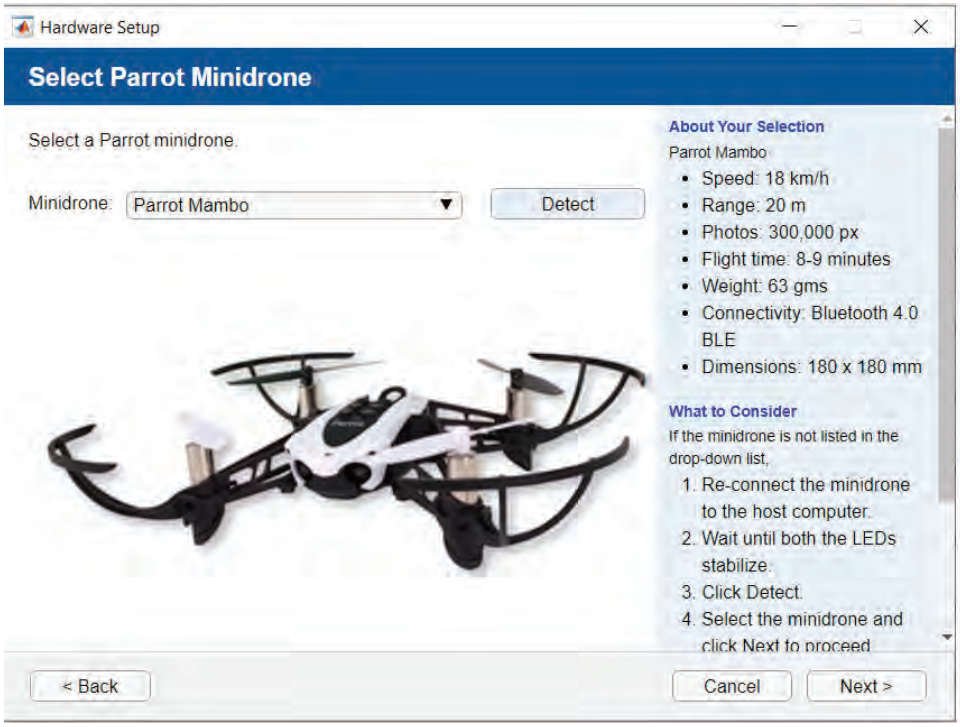

Figure B.2: Detecting drone

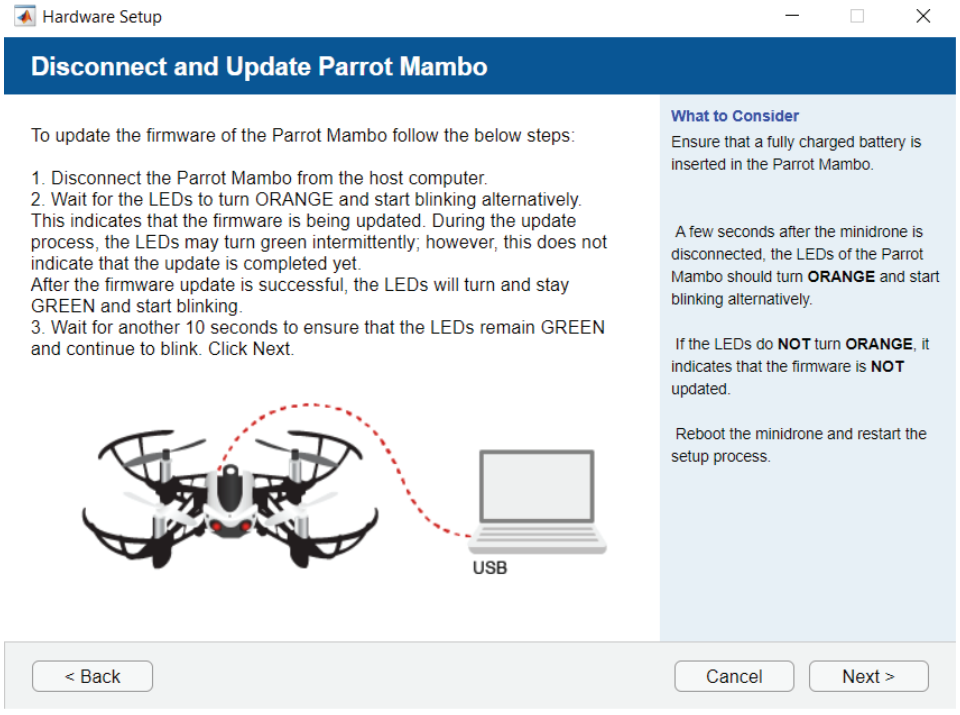

Figure B.3: Firmware update

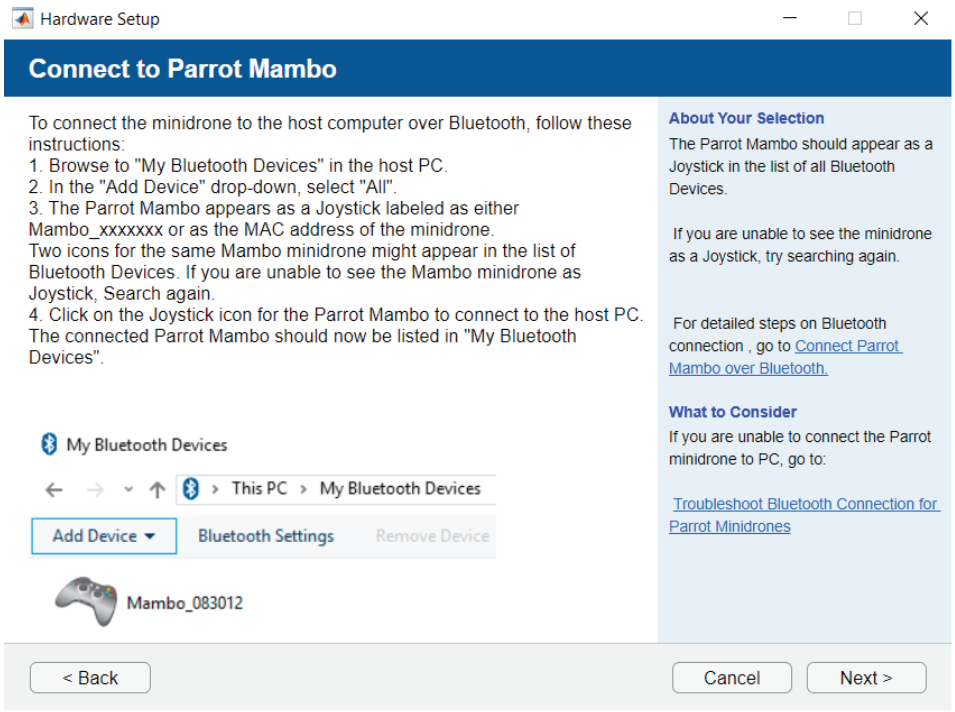

Figure B.4: Bluetooth device

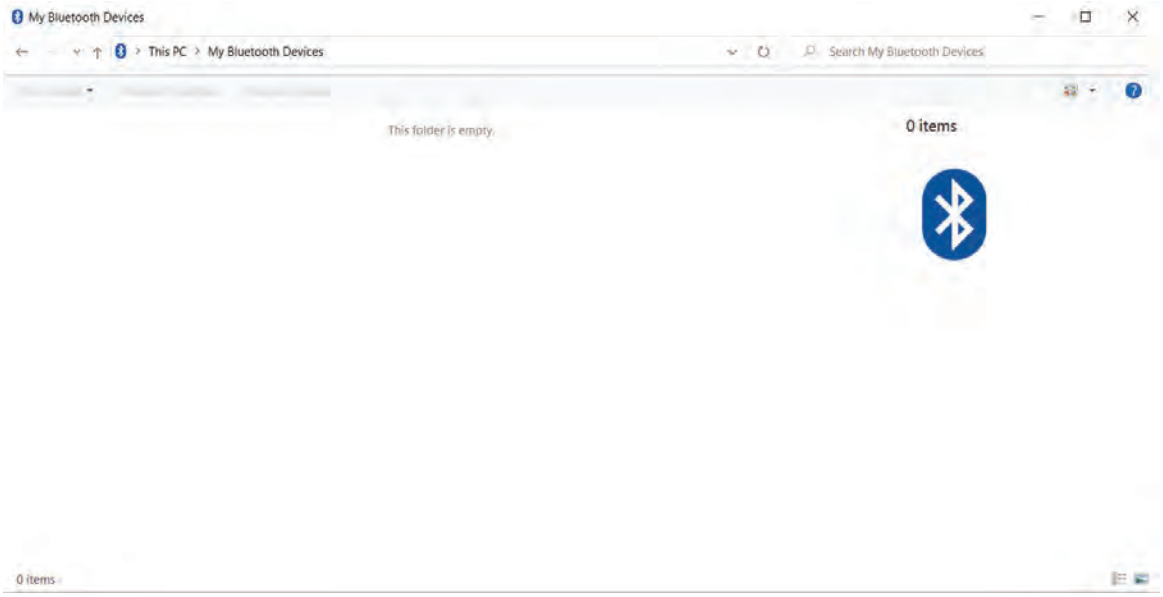

Figure B.5: My bluetooth devices

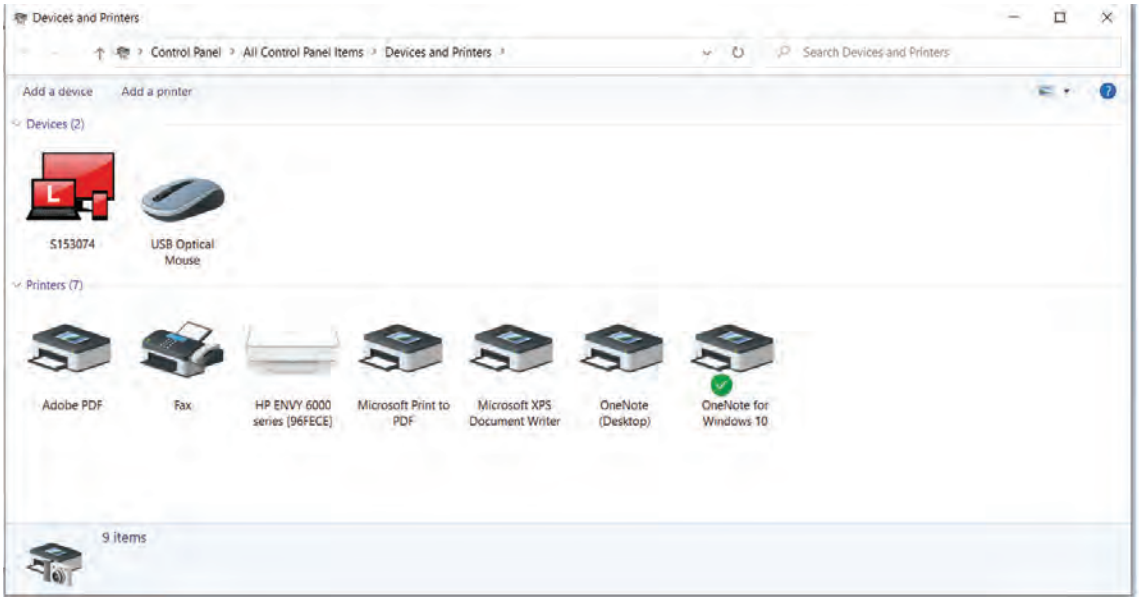

Figure B.6: Devices and printers

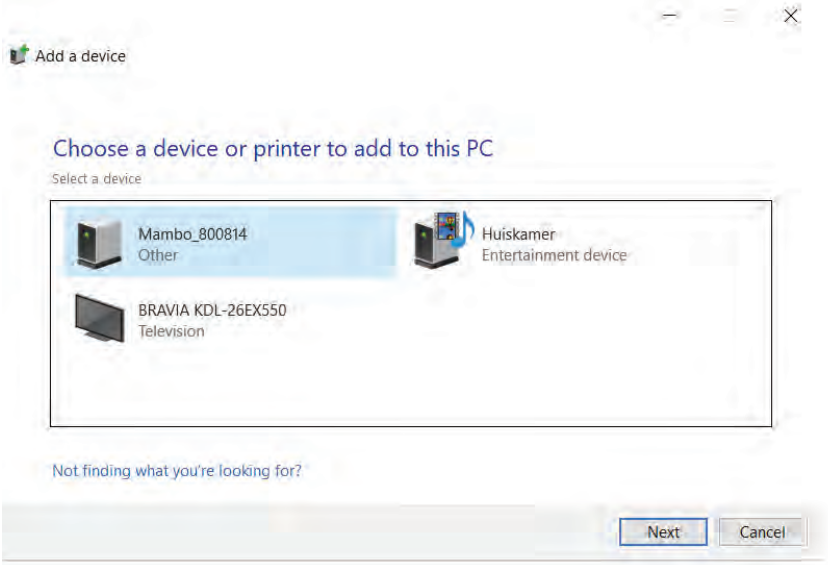

Figure B.7: Drone 1 added

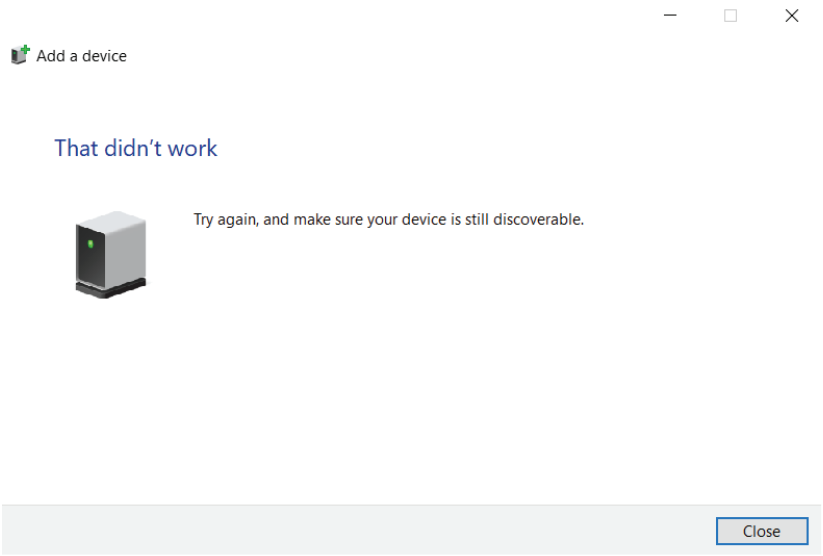

Figure B.8: Error message 1

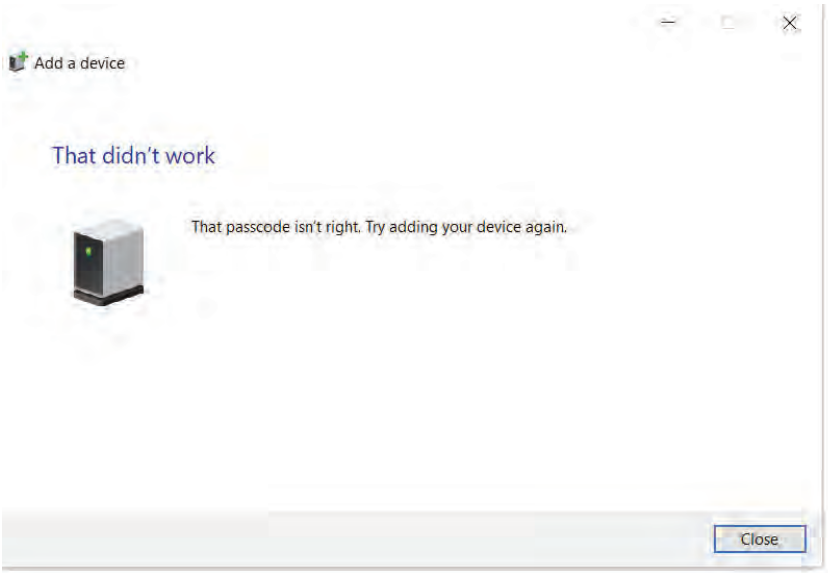

Figure B.9: Error message 2

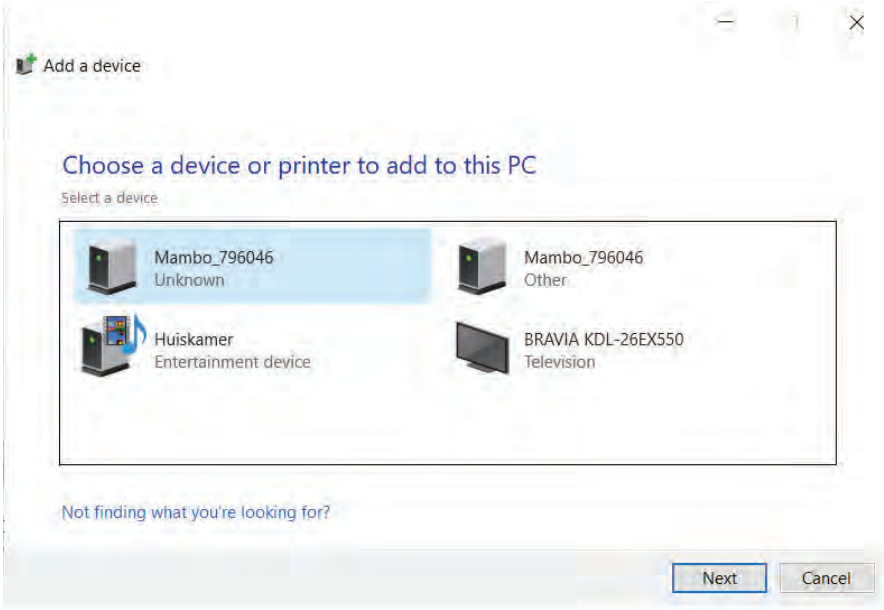

Figure B.10: Drone 2 added

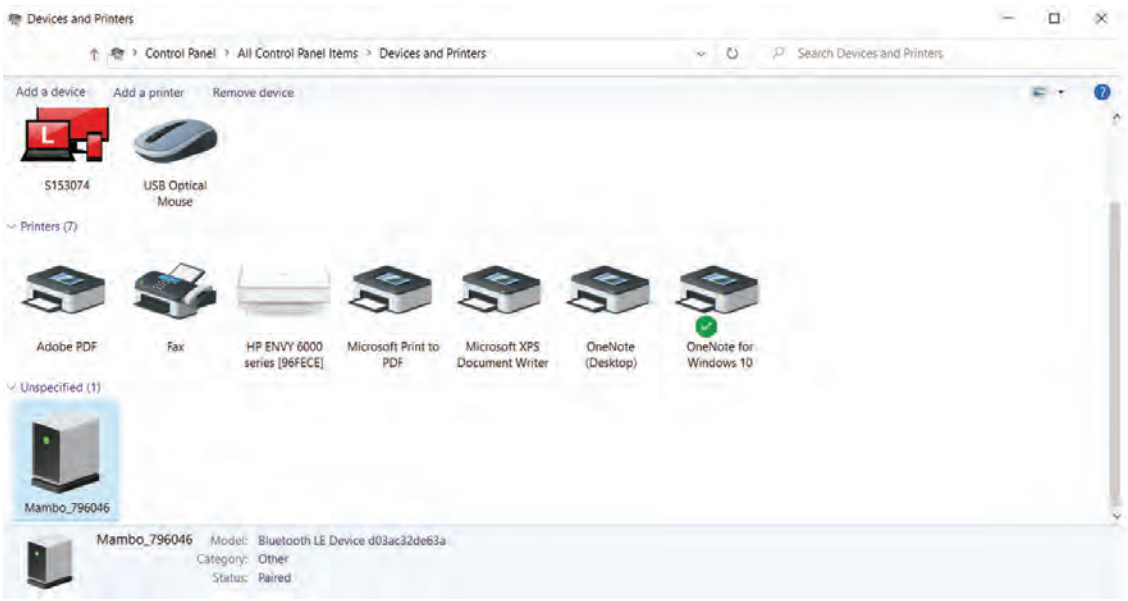

Figure B.11: Wrong option of drone added

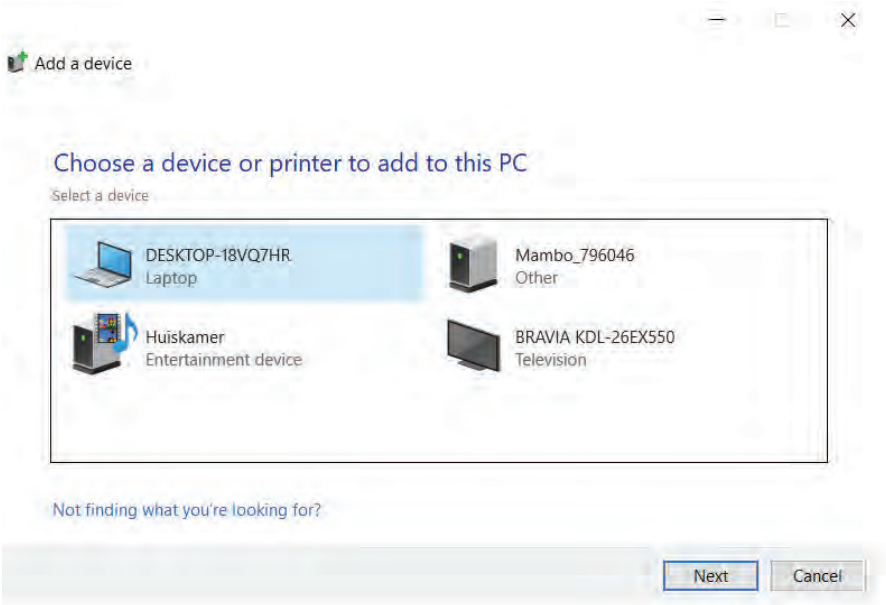

Figure B.12: Finding correct version

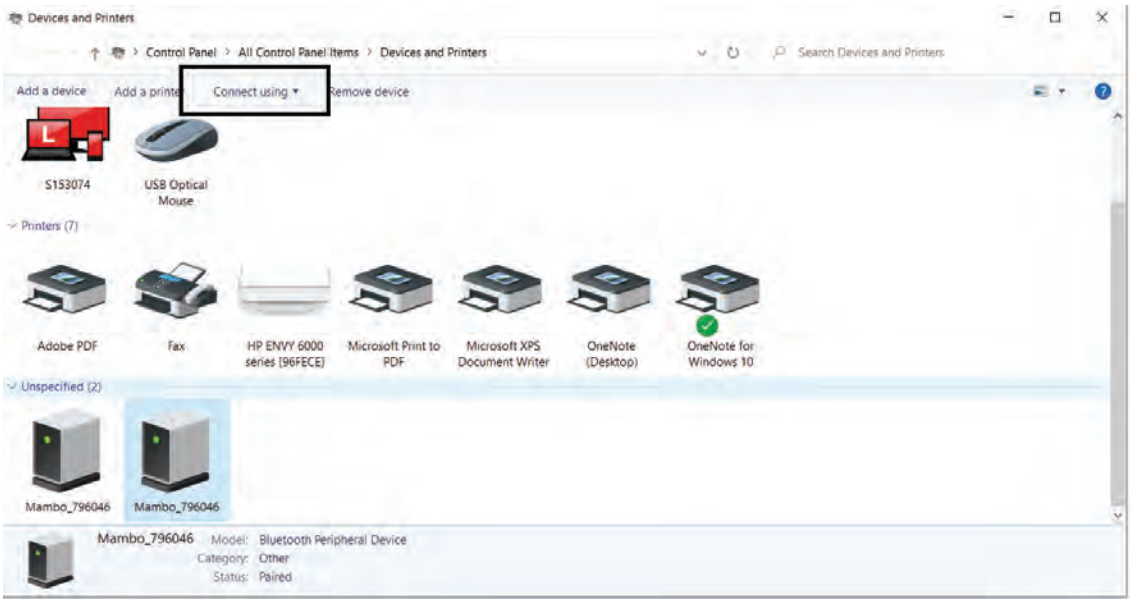

Figure B.13: Correct version added

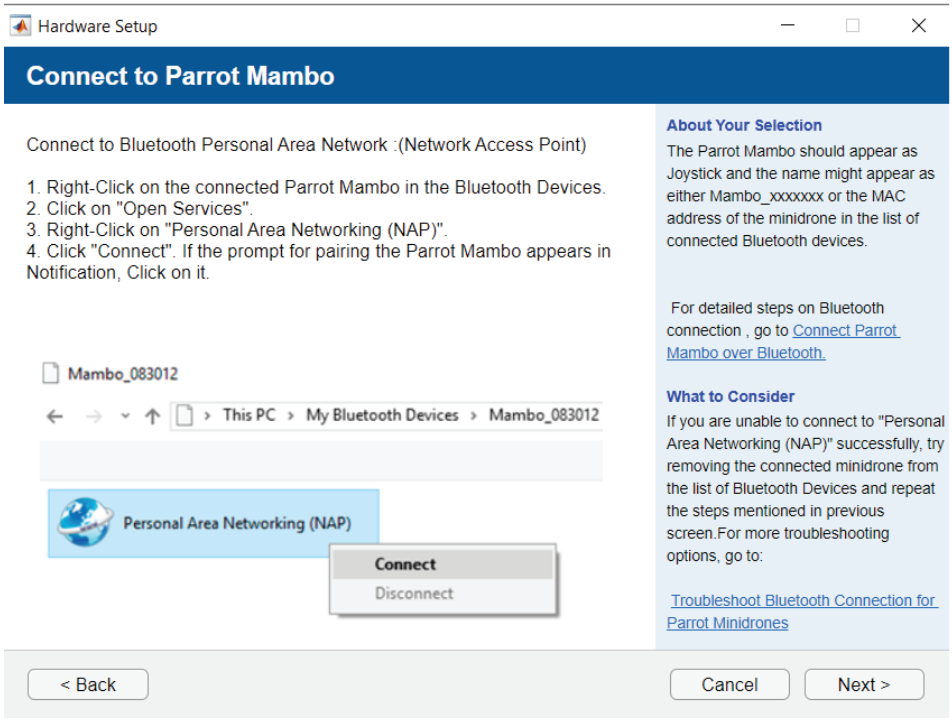

Figure B.14: Connecting via NAP step

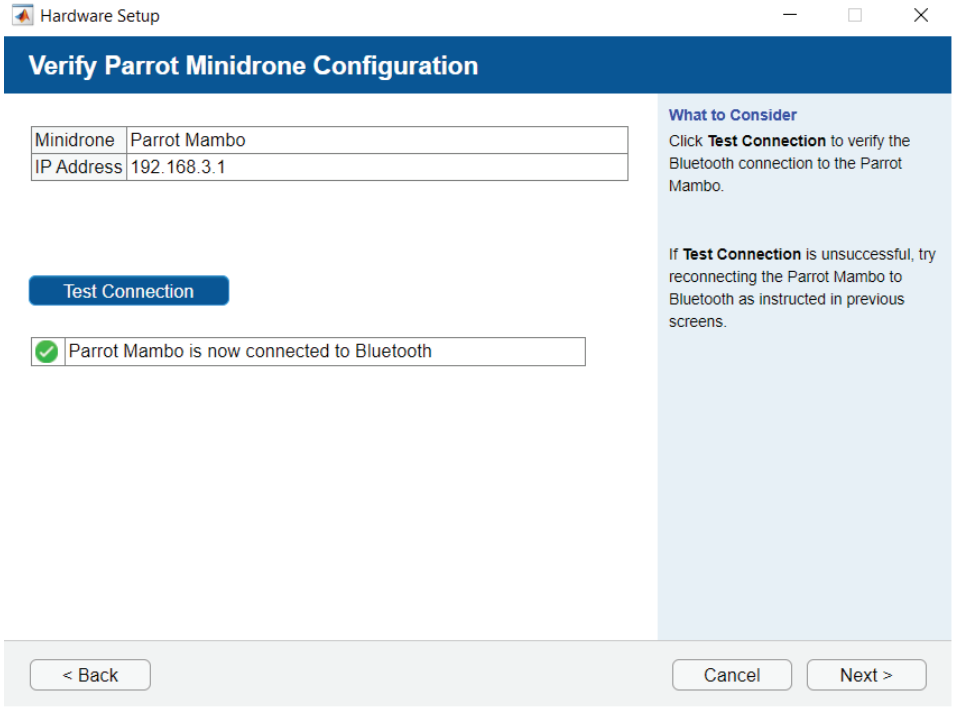

Figure B.15: Testing connection

C Detailed derivation of desired angular velocity

This section gives a detailed description of the newly derived versions of ω_{dis} and its derivative. The expression for ω_{dis} is given by

$$\omega_{dis1} = R_{d31}^T \dot{R}_{d12} + R_{d32}^T \dot{R}_{d22} + R_{d33}^T \dot{R}_{d32}, \tag{C.1a}$$
$$\omega_{dis2} = R_{d11}^\top \dot{R}_{d13} + R_{d12}^\top \dot{R}_{d23} + R_{d13}^\top \dot{R}_{d33}, \tag{C.1b}$$
$$\omega_{dis3} = R_{d21}^\top \dot{R}_{d11} + R_{d22}^\top \dot{R}_{d21} + R_{d23}^\top \dot{R}_{d31}. \tag{C.1c}$$

Writing this in full gives,

$$\omega_{dis1} = \frac{a+b+c}{d}, \tag{C.2}$$

with

$$a = -\dot{f}_{d1}(f_{d1}(f_{d2} + 2f_{d3} + f_{d2}f_{d3} + 2f_{d3}^2)), \tag{C.3a}$$
$$b = -\dot{f}_{d2}(f_{d1}^2 + 2f_{d2}^2 + f_{d1}^2 f_{d3} + 2f_{d2}^2 f_{d3}), \tag{C.3b}$$
$$c = \dot{f}_{d3}(f_{d1}^2(f_{d2} + f_{d3}) + f_{d2}^3), \tag{C.3c}$$
$$d = (1 + f_{d3})^2. \tag{C.3d}$$

The derivative then becomes

$$\dot{\omega}_{dis1} = \frac{d(\dot{a}+\dot{b}+\dot{c}) - (a+b+c)\dot{d}}{d^2}, \tag{C.4}$$

with

$$\begin{aligned}
\dot{a} &= -\ddot{f}_{d1}(f_{d1}(f_{d2} + 2f_{d3} + f_{d2}f_{d3} + 2f_{d3}^2)) - \dot{f}_{d1}(\dot{f}_{d1}(f_{d2} + 2f_{d3} + f_{d2}f_{d3} + 2f_{d3}^2)) \\
&\quad -\dot{f}_{d1}(f_{d1}(\dot{f}_{d2} + 2\dot{f}_{d3} + \dot{f}_{d2}f_{d3} + f_{d2}\dot{f}_{d3} + 4f_{d3}\dot{f}_{d3})) \\
\dot{b} &= -\ddot{f}_{d2}(f_{d1}^2 + 2f_{d2}^2 + f_{d1}^2 f_{d3} + 2f_{d2}^2 f_{d3}) \\
&\quad -\dot{f}_{d2}(2f_{d1}\dot{f}_{d1} + 4f_{d2}\dot{f}_{d2} + 2f_{d1}\dot{f}_{d1}f_{d3} + f_{d1}^2\dot{f}_{d3} + 4f_{d2}\dot{f}_{d2}f_{d3} + 2f_{d2}^2\dot{f}_{d3}) \\
\dot{c} &= \ddot{f}_{d3}(f_{d1}^2(f_{d2} + f_{d3}) + f_{d2}^3) + \dot{f}_{d3}(2f_{d1}\dot{f}_{d1}(f_{d2} + f_{d3}) + f_{d1}^2(\dot{f}_{d2} + \dot{f}_{d3}) + 3f_{d2}^2\dot{f}_{d2}) \\
\dot{d} &= 2\dot{f}_{d3}(1 + f_{d3})
\end{aligned} \tag{C.5}$$

$$\omega_{dis2} = a + b, \tag{C.6}$$

with

$$a = \dot{f}_{d1} - f_{d1}\dot{f}_{d3}, \tag{C.7a}$$
$$b = -f_{d1}\left(\frac{f_{d1}\dot{f}_{d1} + f_{d2}^2}{1 + f_{d3}}\right), \tag{C.7b}$$

The derivative then becomes

$$\dot{\omega}_{dis2} = \dot{a} + \dot{b}, \tag{C.8}$$

with

$$\begin{aligned}
\dot{a} &= \ddot{f}_{d1} - (\dot{f}_{d1}\dot{f}_{d3} + f_{d1}\ddot{f}_{d3}) \\
\dot{b} &= -\dot{f}_{d1}\left(\frac{f_{d1}\dot{f}_{d1} + f_{d2}^2}{1 + f_{d3}}\right) - f_{d1}\left(\frac{(1 + f_{d3})(\dot{f}_{d2}^2 + f_{d1}\ddot{f}_{d1} + 2f_{d2}\dot{f}_{d2}) - (f_{d1}\dot{f}_{d1} + f_{d2}^2)\dot{f}_{d3}}{(1 + f_{d3})^2}\right)
\end{aligned} \tag{C.9}$$

$$\omega_{dis3} = a + b + c, \tag{C.10}$$

with

$$a = (\frac{f_{d1}f_{d2}}{1+f_{d3}})(\frac{(1+f_{d3})2f_{d1}\dot{f}_{d1} - f_{d1}^2\dot{f}_{d3}}{(1+f_{d3})^2}), \tag{C.11a}$$

$$b = -(1 - \frac{f_{d2}^2}{1+f_{d3}})(\frac{(1+f_{d3})(f_{d1}\dot{f}_{d2} + \dot{f}_{d1}f_{d2}) - f_{d1}f_{d2}\dot{f}_{d3}}{(1+f_{d3})^2}), \tag{C.11b}$$

$$c = f_{d1}f_{d2} \tag{C.11c}$$

The derivative then becomes

$$\dot{\omega}_{dis3} = \dot{a} + \dot{b} + \dot{c}, \tag{C.12}$$

with

$$\begin{aligned}
\dot{a} &= (\frac{(1+f_{d3})(f_{d1}\dot{f}_{d2} + \dot{f}_{d1}f_{d2}) - f_{d1}f_{d2}\dot{f}_{d3}}{(1+f_{d3})^2})(\frac{(1+f_{d3})2f_{d1}\dot{f}_{d1} - f_{d1}^2\dot{f}_{d3}}{(1+f_{d3})^2}) + \\
&(\frac{f_{d1}f_{d2}}{1+f_{d3}})(\frac{(1+f_{d3})^2((1+f_{d3})(2\dot{f}_{d1}^2 + 2f_{d1}\ddot{f}_{d1}) - f_{d1}^2\ddot{f}_{d3}) - (2\dot{f}_{d3}(1+f_{d3})((1+f_{d3})2f_{d1}\dot{f}_{d1} - f_{d1}^2\dot{f}_{d3}))}{(1+f_{d3})^4}) \\
x &= (1+f_{d3})(f_{d1}\dot{f}_{d2} + \dot{f}_{d1}f_{d2}) - f_{d1}f_{d2}\dot{f}_{d3} \\
\dot{x} &= \dot{f}_{d3}(f_{d1}\dot{f}_{d2} + \dot{f}_{d1}f_{d2}) + (1+f_{d3})(\ddot{f}_{d1}f_{d2} + 2\dot{f}_{d1}\dot{f}_{d2} + f_{d1}\ddot{f}_{d2}) - (\dot{f}_{d1}f_{d2}\dot{f}_{d3} + f_{d1}\dot{f}_{d2}\dot{f}_{d3} + f_{d1}f_{d2}\ddot{f}_{d3}) \\
\dot{b} &= (\frac{(1+f_{d3})2f_{d1}\dot{f}_{d1} - f_{d1}^2\dot{f}_{d3}}{(1+f_{d3})^2})(\frac{(1+f_{d3})(f_{d1}\dot{f}_{d2} + \dot{f}_{d1}f_{d2}) - f_{d1}f_{d2}\dot{f}_{d3}}{(1+f_{d3})^2}) \\
&- (1 - \frac{f_{d2}^2}{1+f_{d3}})(\frac{(1+f_{d3})^2\dot{x} - 2x(1+f_{d3})\dot{f}_{d3}}{(1+f_{d3})^4}) \\
\dot{c} &= f_{d1}\dot{f}_{d2} + \dot{f}_{d1}f_{d2}
\end{aligned} \tag{C.13}$$