Technische Universiteit
Eindhoven
University of Technology

Department of Mechanical Engineering
Control System Technology
Den Dolech 2, 5612 AZ Eindhoven
P.O. Box 513, 5600 MB Eindhoven
The Netherlands
www.tue.nl

Authors
B.H. Marsman 0842277

Supervisor
dr. Guerreiro Tome Antunes, D.J.

Date
February 19, 2018

Autonomous flying with a quadcopter in close-quarter environment

Internship at CSIR

Technische Universiteit
Eindhoven
University of Technology

Table of contents

Title
Autonomous flying with a quadcopter in close-quarter environment

Technische Universiteit
Eindhoven
University of Technology

Table of contents

Title
Autonomous flying with a quadcopter in close-quarter environment

1 Introduction

- Vergeet niet verschillend opties te geven met bijvoorbeeld Mapping/localization.
- Error analysis bij thrust profiles.
- analytical results bij Inertias.

Mining technology has been improving for decades and mineshafts are dug up to three kilometers in depth. Consequently, these depths go along with grave dangers as shaft collaption, high temperatures and even exposure to toxic gasses. Especially the first one lead to mineworkers, that is why the mineshafts are up to regular inspection. At this moment, an inspector has to go inside the mines to big depths, to inspect walls for tears and cracks. Simultaniously, could the mine be under great pressure and could be collapsing any time. A way must be found to deduct this danger to a human being.

Unmanned Aerial Vehicle (UAV) usage application may be possible for use as a substitution for mining inspections. The application of multirotor copters is widely increasing. Drones have been used for numerous reason already, i.e. UAV Aerial Imaging, surveying, mapping, and making video footages from almost impossible angles. Also UAV inspection is an uprising application as well. Drones can substitude human kind with thereby preventing humans in encountering lifethreatning danger.

To accomplish the usage of drones inside mines, the drone has to be able to perform a mission. Fly a drone into a specific area, taking a picture and fly out again. It basically has to be deployed autonomously in a constrained environment (tight spaces, no GPS) performing a mapping and reconnaissance mission and bring back the data. the following issues need to be resolved for that to function: Firstly, sensor integration on the platform, such that obstacle avoidance can be done, secondly, a localization mechanism has to be found, and lastly, autonomous control of the platform.

This report is divided in different chapters .....

2 Mathematical model

The axis orientation that is used throughout the report is defined in the first section. The equations of motion of the vehicle can be expressed after defining the axis orientation.

2.1 Axis and Force orientation

Goede figuur toevoegen met juiste assen, onze drone is niet symmetrisch!

The axis orientations is chosen as a body fixed frame as depicted in Figure 2.1. The rotation around the X_b axis is called Pitch, the rotation around the Y_b axis is called Roll and the rotation around the Z_b axis is called Yaw. Because the drone used in this report has a slender design, will make roll control harder than pitch control.

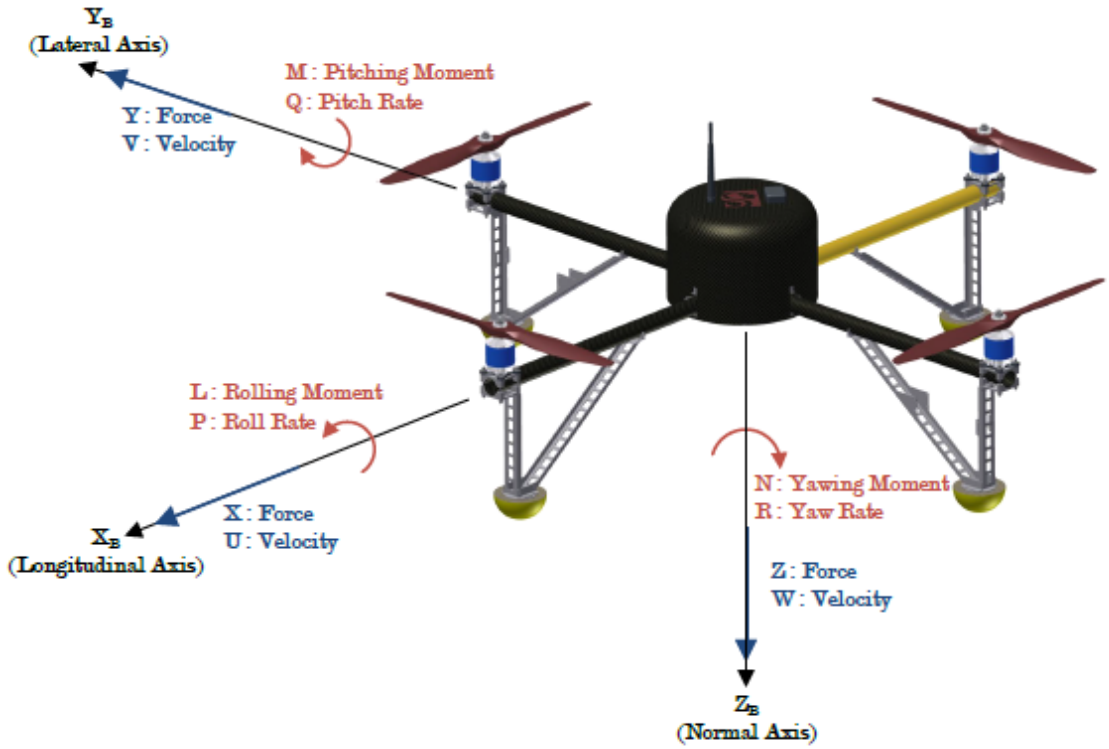

Figure 2.1: Mode shapes of a pinned-sliding beam system

Additionally, velocities,the forces and moments are also visible in Figure 2.1. The total velocity $\bar{V}$

is defined in Equation 2.1.

$$\bar{V} = \sqrt{U^2 + V^2 + W^2} \tag{2.1}$$

2.2 Kinematics

The six degrees of freedom equations of motion for a drone are defined in Equations 2.2.

$$\begin{aligned} X &= m\left(\dot{U} + WQ - VR\right) \\ Y &= m\left(\dot{V} + UR - WP\right) \\ Z &= m\left(\dot{W} + VP - UQ\right) \\ L &= I_{xx}\dot{P} + QR\left(I_{zz} - I_{yy}\right) \\ M &= I_{yy}\dot{Q} + PR\left(I_{xx} - I_{zz}\right) \\ N &= I_{zz}\dot{R} + PQ\left(I_{yy} - I_{xx}\right) \end{aligned} \tag{2.2}$$

In which I_{xx}, I_{yy} and I_{zz} are the moments of inertia around the body's axis, and m the mass of the vehicle.

Thereby it is assumed that:

- The aircraft has a constant mass.
- The aircraft is a rigid body.
- I_{xy} and I_{yz} are zero, because of aircraft symmetry.
- I_{xz} is negligibly small.

2.2.1 State-Space representation

A state-Space respresentation is an easy way to describe the dynamics. A linear model needs to be created if the state matrices are desired to simulate the system. The Forces and moments are seen as inputs into the plant, gives as $[F_x F_y F_z \tau_\phi \tau_\theta \tau_\psi]^T$. Linearization around point x^* is derived if the following trigionomic assumptions are made: $sin(\alpha) \approx \alpha$,$cos(\alpha) \approx 1$ and $tan(\alpha) \approx \alpha$. Thereby is assumed that the states are small. The resulting system is visible in Equation **??**

$$
\begin{aligned}
\dot{\phi} &= p + q\phi\theta + r\theta \\
\dot{\theta} &= q - r\psi \\
\dot{\psi} &= q\phi + r \\
\dot{p} &= \frac{(I_y - I_z)}{I_x} rq + \frac{\tau_\phi}{Ix} \\
\dot{q} &= \frac{(I_z - I_x)}{I_y} pr + \frac{\tau_\theta}{I_y} \\
\dot{r} &= \frac{(I_x - I_y)}{I_z} pq + \frac{\tau_\psi}{I_z} \\
\dot{u} &= -qw + rv - g\theta \\
\dot{v} &= pw - ru + g\phi \\
\dot{w} &= -pv + qu + g - \frac{Fz}{m} \\
\dot{x} &= u + v(\theta\psi - \psi) + w(\phi\psi + \theta) \\
\dot{y} &= u\psi + v(\phi\theta\psi + 1) + w(\psi\theta - \phi) \\
\dot{z} &= -u\theta + v\phi + w
\end{aligned}
\tag{2.3}
$$

2.3 Forces and moments

The forces and moments working on the vehicle can be diveded in three categories: Gravity, actuators and aerodynamics.

2.3.1 Gravity

The gravitational force is a force that pulls an object towards the center of earth. That is why it is defined as follows in earth fixed frame in Equation 2.4.

$$
\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} mg \tag{2.4}
$$

Since the system is defined in a body fixed frame axis, the gravitational force varies when the vehicle experiences pitch or roll rotation. This is why the forces are defined as Equation **??** with use of the euler angles transformation.

$$
\begin{bmatrix} X_g \\ Y_g \\ Z_g \end{bmatrix} = \begin{bmatrix} -sin(\theta) \\ cos(\theta)sin(\phi) \\ cos(\theta)cos(\phi) \end{bmatrix} mg \tag{2.5}
$$

2.3.2 Actuator

Figuur toevoegen met motorvolgorde!

The vehicle is able to hover by use of generating a thrust by the propellers attached to brushless motors. The total thrust is defined in Equation 2.6.

$$
Z = -\left(T_1 + T_2 + T_3 + T_4\right) \tag{2.6}
$$

This thrust is generated in upwards Z_b direction. Moreover, the thrust also introduces several moments inside the systems, because of the propellors generating thrust outside the center of gravity. The moments on the vehicle are defined as in Equation 2.7 to 2.9.

$$L = d\left(T_3 + T_2 - T_4 - T_1\right)) \tag{2.7}$$

$$M = d\left(T_2 + T_4 - T_1 - T_3\right)) \tag{2.8}$$

$$N = l_c \tag{2.9}$$

2.3.3 Aerodynamics

Due to airflow there will also occur some by-effects, for instance drag. The force that is generated by drag is defined in Equation 2.10.

$$F_D = \frac{1}{2}\rho C_D A \bar{V}^2 \tag{2.10}$$

In which, F_D is the drag force, ρ density of the medium, C_D the drag constant and A the front surface.
The forces are again transformed in the body fixed frame, leading to Equation 2.11.

$$\begin{bmatrix} X_D \\ Y_D \\ Z_D \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\rho C_D A_X U^2 \\ \frac{1}{2}\rho C_D A_Y V^2 \\ \frac{1}{2}\rho C_D A_Z W^2 \end{bmatrix} \tag{2.11}$$

2.4 Parameter identification

Certain parameters needs to be known to complete the system dynamics. Some of these parameters have to be defined by experiments i.e. Center of Gravity (CoG), mass moment of inertia and thrust profile per motor-propeller set.

2.4.1 Thrust profiles

Seperate thrust profiles have been created to take the differences in thrust into account in the simulation. No component gives the same results, because of manufacturing uncertainties. Additionally are there some steps to generate thrust, from the output send to the motor. It goes through the ESC, which transforms the 5 Voltage input into a 22 Volts blocksignal output into the motor. Consequently will the motor transform this into a rotational speed and as a result will the aerodynamics generate a thrust via the propellers. Since these components differ from one another, the thrust outputs will vary as well. For instance ESC's from the same producer can differ $\pm 2\ \%$ from eachother and there is some manufacturing variation in the motors and propellers as well.
Thereby are some measurements done to take this effects into account. All inaccuracies can be taken into account by creating a thrust-curve dependent on the pwm motor output. A test setup has been created, which is depicted in Appendix Figure .... , with a loadcell to measure thrust. During the experiments, there has been taken into account that the turbulences are not or nearly affecting results.

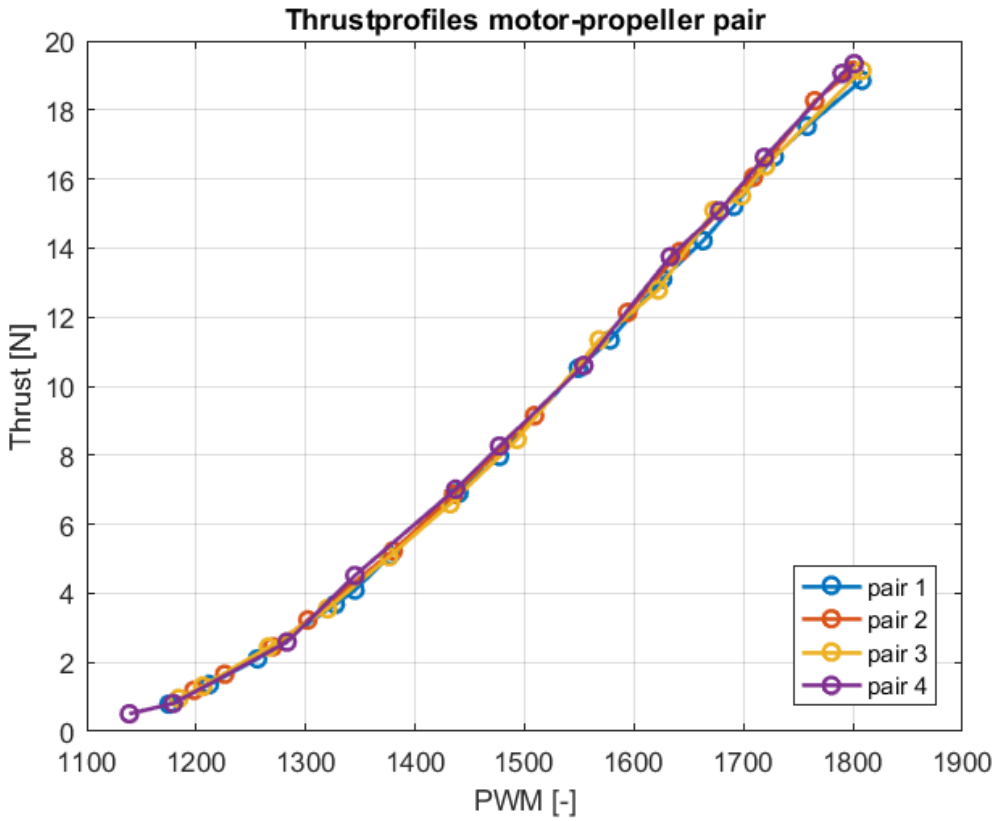

Figure 2.2: Experimental approach of defining thrust-profiles.

The results of the thrust-profiles are depicted in Figure 2.2. As can be seen in Figure 2.2, the difference is almost neglictable in the area which the motors operating most of the time (1400-1600 pwm). At higher and lower pwm-values, there is some difference in thrust.
Additionally, the thrustprofiles show that the PWM shows a linear progress, especially around hovering (1400-1550 PWM). This means that there does not have to take into account much nonlinear effects here.

2.4.2 Center of Gravity

The center of gravity has been defined by tethering it on the ceiling. The location in xy-plane has been found by getting tether it on different places on the drone and receiving a horizontal oriented drone. Furthermore this is also done to obtain the CoG in the z-direction. Alternetly the drone had to be tethered in a way such that the orientation of the drone would be vertical. Hence the z-direction would be found.

2.4.3 Mass moment of inertia

The mass moment of Inertia is derived with an experimental approach called a 'bifilar pendulum'. A bifilar pendulum consists of tethering the body with two parallel wires. Therefore it is allowed to rotate freely about a given axis. A small moment is applied to the body, which starts to oscillate around this axis. The moment of inertia can be determined with Equation 2.12[1].

$$I_{ii} = \frac{mgr^2}{4\pi^2 L f^2} \tag{2.12}$$

wherein I_{ii} is the moment of inertia around xx-, yy- and zz-axis, m the mass of the body, g is gravitational acceleration, r is the distance from CoG to wire-body connection, L length of the wire and f the frequency of oscillating equal to $f = 2\pi\omega$. This means that the only variable will be the frequency, which can be measured by the oscillation time $T = \frac{1}{f}$. To get more accurate result, the oscillation is mearured n times, and thereby be averaged. The results of the experiments are depicted in Table 2.1.

Table 2.1: Moments of inertia

	I $[kg \cdot m^2]$
I_{xx}	0.025028
I_{yy}	0.16196
I_{zz}	0.17079

Hereby has been taken into account that the off-diagonal terms are zero. This is because the I_{xy} and I_{yx} are zero because of symmetry. The same accounts for I_{xz} and I_{zx}. Thereby is I_{yz} and Izy is neglitably small.

add pictures of the test setup in appendix and derive the equation

`https://conservancy.umn.edu/bitstream/handle/11299/182514/Habeck_UROP%20Final%20Report.pdf?sequence=3&isAllowed=y`

3 Ardupilot

Flow diagram Angus showen.

The software on which the drone runs on is called Ardupilot. Ardupilot is an open source available software, which is developed by a team of diverse professional engineers and computer scientists [2] `http://ardupilot.org/about`. The software is capable of controlling multirotors and other vehicles, which is easy adjustable to desire. Figure 3.1 shows a highlevel view of the ardupilot architecture.

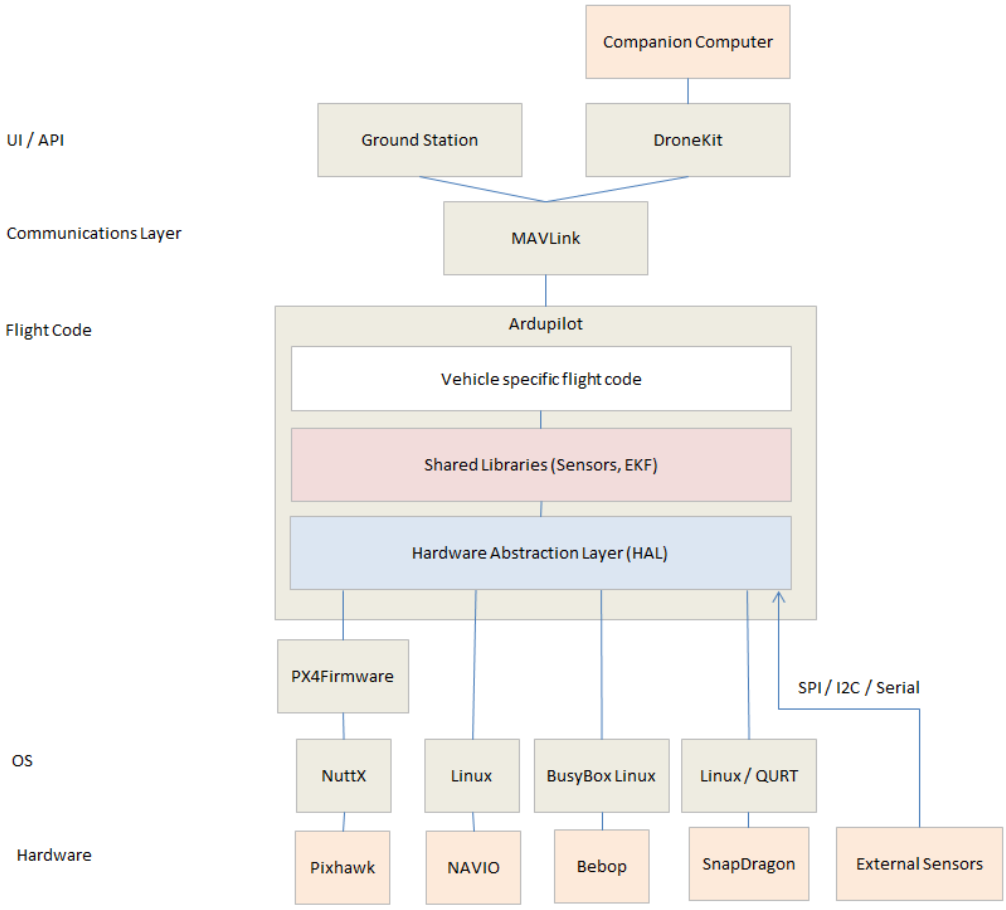

Figure 3.1: Highlevel view of the ardupilot architecture.

Zooming a bit closer explains the code better. As can be seen in Figure 3.2, the base of the code is divided in several subfunctions which are explained in this chapter.

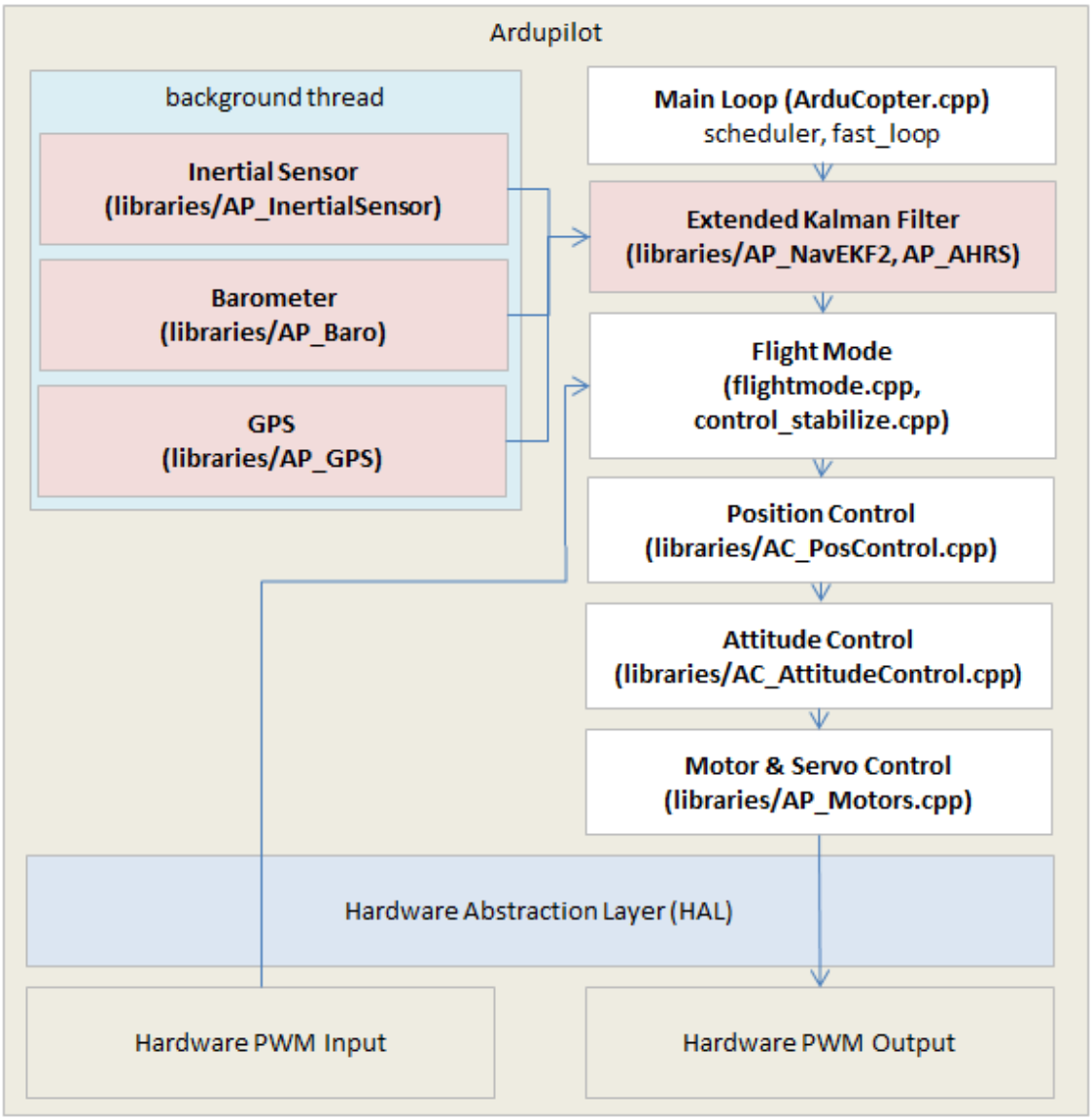

Figure 3.2: Zoomed highlevel view of the ardupilot architecture.

3.1 Main loop

The software runs with a main loop working at a 400 Hz frequency.

3.2 Flight mode

Flight mode checks the vehicle's flight mode and calls the matching function. A flight mode converts the given input by the user into a lean angle, climb rate, etc. that is suitable for that flight mode. A few flightmodes, which are used throughout the project, are explained in detail.

- Stabilize: This modes allows manual flying, with automatic control of the roll and pitch axis.

The automated roll and pitch control ensures the drone from not tilting, however it can experience drift by wind. The pilot can counter this by giving a lean angle input to the copter. With respect to yaw, the pilot can give a yaw-rate input to apply yaw rotations. Furthermore does the pilot has to give enough thrust input so that the drone maintains altitude. Give less thrust to descent and give more to climb.

- Loiter: automatically attempts to maintain the current location, heading and altitude. When no lean angle input is given by the pilot, the vehicle will try to hold position. Furthermore can the pilot give lean angles input to move in horizontal plane and can the altitude be altered by increasing or decreasing throttle input. Thereby it is needed to have good GPS connection, low magnetic interference on the compass and low vibrations to achieve good loiter performance.

- AltHold: The copter maintains altitude while allowing roll, pitch and yaw to be controlled normally. This means that the throttle will be automatically controlled to maintain altitude. Operating in lean angles will be the same as in Stabilize mode.

- Auto: This mode is a combination of different modes, since it uses altitude control from AltHold and the position control from Loiter. Auto mode is able to fly a pre-programmed mission, which will take the drone from given location to location. After taken the desired path, a Return To Launch can be used to land at Home, which is the initiated location where the vehicle was armed.

3.3 Position Control

3.4 Attitude control

The attitude control defines the controller to convert the Roll/Pitch/Yaw errors (the difference between the target angle and actual angle) into the desired rotation rate followed by a PID controller to vonvert the rotate rate error into a high level motor command. A flow scheme has is visible in Figure 3.3.

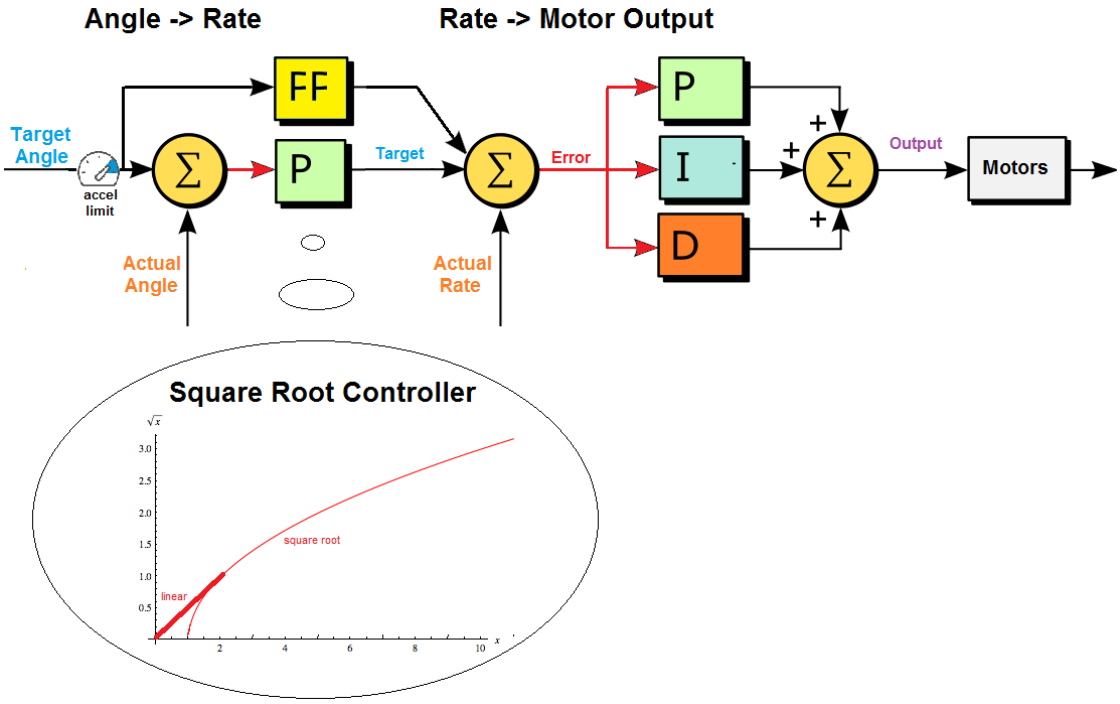

Figure 3.3: Flow diagram showing the attitude control per axis.

$AC_{A}ttitudeControl$ library

3.5 Motor and Servo Control

3.6 Other

Furthermore are there some other functions which can be added to Ardupilot as well, i.e. other sensor drivers.
Frame orientation.
Motor mixing.

4 Test Flights

There has been some reinforcement to the drone because of the weak frame.

Write the steps of how what steps were done and for what reasons (for instance the arms were vibration a lot, because it was torsion weak. Therefor was there added a lightweight waterjetted reinforcement to add rigidity in torsion.

magnetometer issues.

DESCRIBE THE TEST FLIGHTS WE HAD SO FAR

5 Model validation

Making changes in the way of flying is quite dangerous with a quadcopter. These changes can namely cause unstability resulting in a crash. Hence, modeling the behaviour of a quadcopter can be of great addition. The Ardupilot code is transformed into a simulink model and simplified, so that it is more userfriendly. Because of the complexity of the Ardupilot code, it is very difficult to make changes. Thereby is taken into account that the dynamics are still according the rules of nature. Controllers, observers and routeplanning can now be tested in a more controllable environment.

5.1 Build up

Starting with the Equations of motion, explained in Chapter 2. The plant is created by a six D.O.F. block that basically gives all the required outputs, calculated from the incoming moments, forces and current states. Parts of the To simulate reality, the simulink model is created as real as possible, including measurement and process noise and other affects as well. Changes in the system can be made by validating the directions and there is even a possibility to add the motormixer if required. However, this is excluded for the rest of this report to simplify the system. Including this motormixer will be stable with retuning the control parameters.

Furthermore is the position controller also implemented. This is another flying mode, different from the Stabilize mode explained in Chapter 3, namely Auto. This implementation requires more than the inner Rate loop, since has not direct influence into the plant, but outputs desired angular rates. Consequently, it does not use the previous attitude controller anymore, but has this included in the horizontal error angle controller. This challenge can be solved in various ways, but the Moller's [2 Moller] method. The control scheme is also visualised in Figure 5.1.

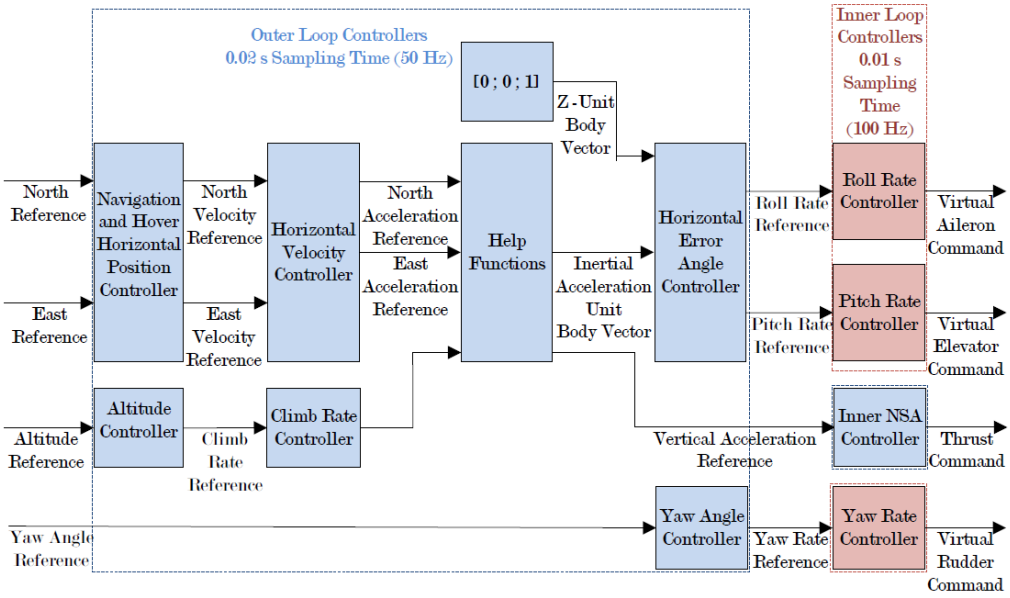

Figure 5.1: Moment vs wall gap (square marker: moments about y-axis; circle marker: moments about x-axis).

Note that yaw control is completely seperate from pitch and roll. Yaw control is seperate, because of roll and pitch are prioritised for stable hovering. The block helping functions consists of... HIER NOG AANVULLEN. ook bijvoegen met acceleratie vector blabla.

5.2 Comparison

Since there is no obtained flightdata for now, the model still has to be verified. However, the created model constists of a lot of details, besides the equations of motion. The model in personalized in the matter of the following list: Inertia, geometry, Thrust-profiles, vibrations and mass. There is assumed that the drone repesents reality eventhough the real validation cannot be done. This is already been proved by research in the past. Hier nog bronnen bijgeven dat drones goed the modeleren zijn

5.3 Near-wall effect

There are two different effects that occur when flying close to a wall caused by airflow. Because of a not uniformly distributed thrust over the propeller there are extra moments working upon the propeller [1 near wall paper]. Additionally, the airflow not being able to have sufficient instream causing the propeller to lose some thrust. The thrust losses are obtained by experiments, which are discussed Chapter 5.3.1. For the external moments caused by the not uniformly distributions of air flow are calculated by scaling the moments with the following non-dimensionalised moment.

$$M^* = \frac{M}{WGmg} \tag{5.1}$$

With M^* as non-dimensionalised moment, M moment, WG wallgap distance. The results from the numerical simulation from [1 Near wall effect] are shown in Figure 5.2.

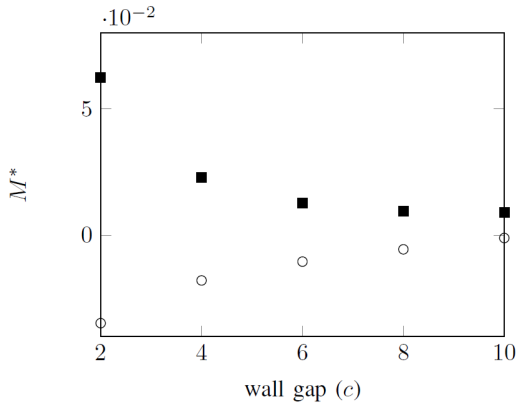

Figure 5.2: Moment vs wall gap (square marker: moments about y-axis; circle marker: moments about x-axis).

The result that is shown in Figure 5.2 can be used with scaling M^* to M and are implemented.
Extra information about the not uniformly distribution over the propeller. $https://www.mathworks.com/videos/prog$
$drones-with-simulink-1513024653640.html$

5.3.1 Implementation

As mentioned before, the implementation of the disturbance is seen as a thrustloss of one of the motors and an extra moment upon the quadcopter. Because of this significantly big sudden effect on the drone, it is not controllable by a simple PID-controller. Consequently, other techniques should be used to counter sudden effect.

Three options:

- Very agressive controller.
- Feed forward.
- Disturbance observer could be designed.
  `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6842357` see the discussion to see approach how to make a disturbance observer.

Disturbance Observer Based Control (DOBC) is hereby tried to account for the near-wall effects. The basic idea of DOBC is estimating the disturbance (or effects caused by it) from measured variables and control that observed disturbance to compensate for the influence of the disturbance. Normally the DOBC is used independently of the tracking control. DOBC is benificial that it uses no extra sensors to detect the disturbance, but tries to observe it. The disturbance observer should moreover not interfere with the tracking control. The system could get unstable if the influence is too big.

Hier plaatjes van de disturbance in het system implementation en uitleg

6 Controller design

Controller design of Attitude controller is based upon the simulink model, described in Chapter 5. Hovering the quadcopter is done with an inner-(rotational rate) and outer-loop (attitude). Regarding position control uses these two loops as well and adds two extra loops; position and velocity feedback. All loops are explained in detail below.

6.1 Angle control

The angle control which is done with feedback of angular position and angular velocity. The inner loop is tuned first while using multiple feedback loops, which is the rotational rate loop. p,q and r (roll-, pitch-, yaw-rate) are tuned seperately and checked if functional in the combined system. The tuning for roll is shown in Figure **??**.

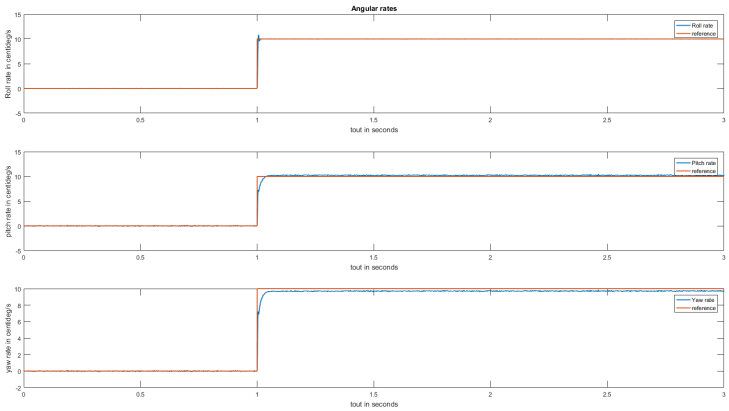

Figure 6.1: Step response in rotational rate showing that the tracking controller works proper.

nieuw plotje maken (netter) Firstly P-action is added to increase the speed of the system, D-action to prevent overshoot and an extra I-action to decrease the overall error. The tuned controller shows that the rotational rate follows the reference nicely. Thereafter is the attitude controlled. Since inner loop is already controlled, only P-action is needed in attitude to obtain good tracking. Increasing the gain is however shows that the attitude introduces an overshoot. This can be solved by adding a D-action in the attitude, or by increasing the gain in the rotational rate controller, which is also derivative action for the attitude. After retuning, the result is visible in Figure 6.2.

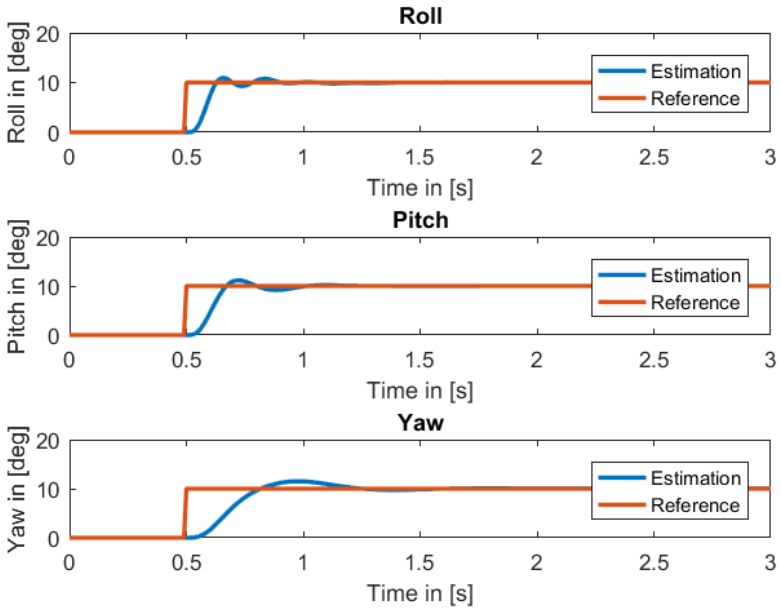

Figure 6.2: Step response in the three angles showing that the tracking controllers works proper.

Misschien nog een figuur bij stoppen die de rate laat zien bij een step response in attitude.

Note that in case of position control, the controller of the attitude has to be tuned at another point in the control scheme, but will be tuned with the same method.

6.2 Position Control

Hier plaatjes van horizontal velocity (PID?) en horizontal position(P?) plaatje van desired angle en hoe die hem trackt Plaatje van altitude control.

6.3 Compensating Near-wall effect

Attitude dynamics of rotational rate is needed to be described to create a disturbance observer. These can be discribed in the same way as in [3 hier profs paper] described in the set equations of 6.1

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= F(t) + bu \\ y &= x_1 \end{aligned} \tag{6.1}$$

With x_1 and x_2 as rotational-rate and -acceleration, $F(t)$ as external disturbance, b and u are statematrix and input. Since $F(t)$ is not known, because of fluctions and dependencies of wall gap. Consequently, $F(t)$ is considered state x_3 and let $\dot{x}_3 = G(t)$. The equations of 6.1 become 6.2

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 + bu \\ \dot{x}_3 &= G(t)y \qquad = x_1 \end{aligned} \tag{6.2}$$

It is possible to create an extended state observer now to estimate the states x_1 and x_2 and disturbance x_3, since the dynamics are now in canonical form and is therefore observable. Examples of a linear observer is shown in Equation 6.3

$$\begin{aligned} \dot{z}_1 &= z_2 - L_1 e + bu \\ \dot{z}_2 &= -L_2 e \end{aligned} \tag{6.3}$$

With

$$e = y - z_1 \tag{6.4}$$

With z as the observed states and disturbance and L as the observer gains. The choice of L is created by the designer, which is explained in further detail lateron. Discretizing the system gives

$$\begin{aligned} z_1(k+1) &= z_1(k) + T_s\left(z_3(k) + bu(k)\right) - \beta_{01} e(k) \\ z_2(k+1) &= z_2(k) - \beta_{02} e(k) \end{aligned} \tag{6.5}$$

with

$$e(k) = y(k) - z_1(k) \tag{6.6}$$

These final equations are used in the model which is visible in Figure 6.3.

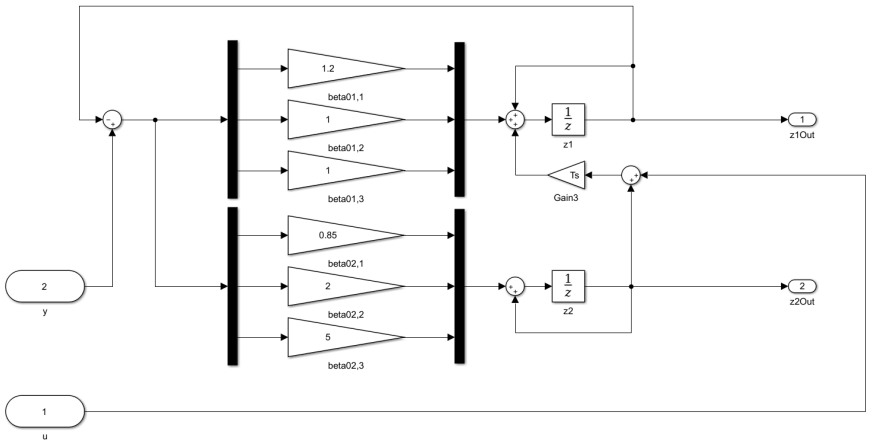

Figure 6.3: Control scheme of DOBC with y the output, u the input, z_{1out} estimated rotational rate and z_{2out} the estimated disturbance.

7 Route-planning

As discussed before, flying inside mines produces a lot of complexity. Thereby is the routeplanning one of the most important things, since it can make it easier for the UAV to fly indoors. When there is no neccesacy to fly close to a wall, do not do it. It remains possible to occur instability in combination of wind gusts and near-wall disturbances. Consequently, a route-planning needs to be created to move smoothly and smart from point A to B. Several things have to be taken into account in this routplanning:

- Go around corners slowly. Use yaw control to maintain good distance from both sides of the wall. Figure 7.1 shows a possible structure of a minehshaft. The drone will in our minds keep up with distances to wall on horizontal 360 degrees plane (Lidar). Hence the route-planning can make sure that the drone flies in the middle of the mine. The same things accounts for the ground and ceiling (can be done with sensors facing up and down), since there are also undesired effects coming up there.
- Do not fly close to walls, except if not possible otherwise. Localization in horizontal plane is in this case also done by the Lidar 360 range sensor. When encountering close to a wall, the DOBC will compensate for the disturbance. The route-planning can make it easier for the drone again to move away from the wall by stepping back a little back.
- Obstacle avoidance is required to prevent crashing into hanging stalactites. In the Ardupilot code is a feature present which can detect obstacles already. This option can be a good option when implemented in the route-planning.

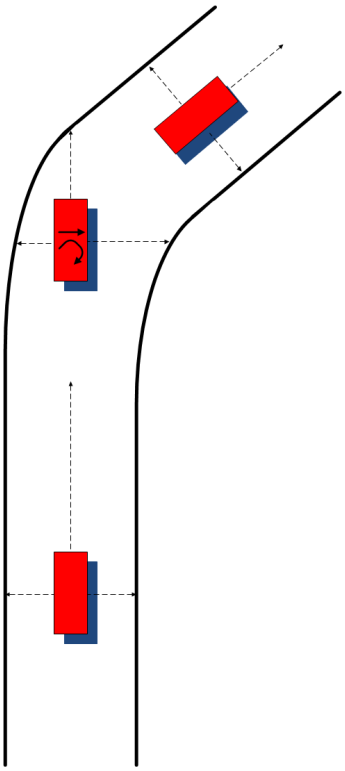

Figure 7.1: Route-planning in a possible shaped mineshaft

The reason for why there is still thought of a DOBC to counter the near-wall effect is that it is possible to get close to a wall by sensor unreliablity. Other reasons can be oddly shaped mineshaft which interrupts the airflow, or no other possibility of flying close to the wall. The latter can be caused by

A proper route-planning can also use a localization method. A realizable way of doing is, is the 3D mapping with two cammeras on front. The quadcopter can now detect the obstacle avoidances and walls all at once. Maybe extend this a little bit on how to implement this and what techniques are used. Add link of example on youtube or so.

8 Conclusions

Unfortunately it was not able to verify the model in Chapter 5 with experimental data, because of sensor interference and a non-robust quadcopter that got demolished by a crash. Several things are added in the simulation which should make it realiable, i.e. thrust profiles, noises, geometry, etc. Previous research however shows that simulating a drone represents reality quite nicely. The simulation has provided information that it is possible to compensate near-wall effects with disturbance observer based control. At least, this compensation works in simulation and has to be verified in reality, which requires some extra tuning, which is explained in further detail in Chapter 9. This is quite an interesting outcome, since flying close to walls has been an issue for quite some time. The use of DOBC could be a good solution to the problem, since there is currently no existing solution to fly close to a wall in an uncontrolled environment.

Additionally, the theoretical route-planning has been created to fly smoothly through mine corridors. Hereby has been taken into account that there can be use of GPS underground and thereby uses another mapping/localisation technique. This theoretical routeplanning has to be fused with the mapping/localisation technique that will be used for the final version, as discussed in Chapter 7. Further research should provide the likelyhood of succes with route-planning, DOBC and obstacle avoidance combined.

Lastly, there has been great understanding in the quadcopter system and there are several things to take into account if flying the drone. Consequently, the report provides some Do's and Dont's to simplify the drone usage. The Kalman-filter has to be checked that the states are estimated correctly. Thereby has to be checked if there is for instance, no magnetic interference of bad GPS data.

9 Recommendations and Discussion

Add a part which says that we should make the disturbance also dependent upon the rotorspeed.

Several things can be added to make the simulation more realistic and thus comparable to experimental data. This gives a benefit over tuning in the simulation instead of risking the drone to crash. Firstly, extending the model with non-linearities, which are neglected around hovering. There are neglected non-linearities in the attitude dynamics, Kalman filter and even in the near-wall disturbance. These non-linearities are however still present and give a standard error. Secondly, making controllers more robust to modeling errors by creating controllers with Robust Control. A non-linear Robust controller should even be a possibility.
Furthermore needs to be noted that an extra observer requires some computation time, that can overflow the OBC. Especially at the same looprate as the attitude control of 100Hz. To get the DOBC working in reality, an addition of an external On Board Computer (OBC) will solve this issue. Some additional tuning of the DOBC to the real system or creating a robust controller in the simulation already.
Besides the simulation, some recommendations for the quadcopter including some additions can be done as well. First of all, and the most important one, is to buy or build a robust quadcopter that can absorb multiple crashes without being demolished. For an inexperienced dronetechnician, it will occur highly probable that the system turns unstable when testing new controllers. Hence, if the drone breaks and looses some parts or flexibility, the system also changes. This happens due to changes in mass or inertia or extra vibrations in the system (loose bolts or screws) and thereby will the controllers need to be retuned. A solution to this could be tethered flying, however this technique should be approached with caution. Tethering too loose and the drone will still be able to crash, and tethering too tight will affect your system. Thereby is the tethering that is used in Chapter 4, only applicable for hovering purposes. Testing near wall-effects and position control will need other kind of tethering, if those exist.
Additional sensors need to be added to the quadcopter to realise the route-planning. It is recommended that the drone uses a 3D mapping method that uses SLAM Explain in more detail!. These method uses two gimbal cameras to map the surroundings.
The experimental data regarding the near-wall thrustloss has not been reliable as discussed before. Since the sensitivity of the experimental setup was not high enough, were the differences between the results hard or impossible to read. The results are now taken with a significantly high safetyfactor, and gives decent results. Controllers can be tuned more precise when the simulation is provided with a better representation of this disturbance. This means that the experiments need to be performed with a higher precision loadcell and a solid test setup.
Moreover the near-wall effect, it has come to the attention that it is highly dependent upon the rotorspeed, and the assumption of the propeller rotation on the speed of hovering is probably not viable. This is because the controller will boost the propeller that is close to the wall, amplifying the effects and is not taken into account so far. Consequently, these effects has to be added to the simulation to properly tune the observer gains and see if the drone still compensates these effects.
As discussed before, is there a conflict with the code generation that drives the platform. It is currenly the Px4v2 that runs on Ardupilot, however it is not very user friendly to make addaptions.

Thereby is it recommended base the code on Simulink itself, which can be done with Hardware In the Loop (HIL) Explain this a little bit. Or do some research, since I actually do not understand .. :)