BEng Mechatronics

University of Southern Denmark Sønderborg

Semester Project 4 - Ladybug

Ruta Miglava, Johan Paul, Choyon Mainul Hasan, Thor Uerkvitz, Artis Fils, Tobias Blaabjerg Karlsen

Supervised By

Lars Duggen

Davi Goncalves Accioli

June 2023

SDU

Contents

1 Background

Possessing the ability of flight and minimising effort and casualties has always been desirable for the utility flight can provide. The first unmanned aircrafts can be dated back to 1849, where Austria seemingly had utilised unmanned air balloons with stuffed explosives to attack Venice. [42] Ever since an unmanned aircraft vehicle (UAV), is one that is flown by technological means or as a pre-programmed flight without pilot control, as defined by the ECAA Transport Agency [19], nowadays called drones, have risen in popularity.

Because of this, UAVs come in a wide range of sizes and weights. UAVs often include multirotor, radio-controlled miniature helicopters, and aeroplanes [13]. As a result, there are several methods to categorize drones. The performance parameters of UAVs, such as weight, wingspan, wing load, flight range, maximum flying altitude, speed, and production cost, are typically used to categorize them [24]. According to how the lift is produced, drones may also be divided into fixed-wing and rotating-wing types. According to the drone code category, the European Aviation Safety Agency (EASA) categorizes unmanned aircraft by weight. The EASA regulations for open categories, or drones without an EASA class designation, are summarized succinctly and simply in Figure 1 [11].

Self-built drones weighing up to 250 g, as described in Figure 1, may be used without registration if the drone is a toy or the drone is not equipped with a camera, the remaining drones must be registered, and the pilot must pass examinations [11]. In this paper, self-built rotary drones with four wings or propellers are the objective, making weight-based classification suitable.

UAS	Operation		Drone operator/pilot		
Max weight	**Subcategory**	**Operational restrictions**	**Drone operator registration**	**Remote pilot competence**	**Remote pilot minimum age**
< 250 g	A1 (can also fly in subcategory A3)	— No flight expected over uninvolved people (if it happens, overflight should be minimised) — No flight over assemblies of people	No, unless camera / sensor on board **and** the drone is not a toy	— No training required	No minimum age
< 500 g			Yes	— Read carefully the user manual — Complete the training and pass the exam defined by your national competent authority or have a 'Proof of completion for online training' for A1/A3 'open' subcategory	16*

Figure 1: Classification and restrictions for non-EASA class drones [11]

When it comes to the state-of-the-art project, PULP-DroNet is a deep learning-powered visual navigation engine that enables autonomous navigation of a pocket-size quadrotor in a previously unseen environment. Thanks to PULP-DroNet the nano-drone can explore the environment, avoiding collisions also with dynamic obstacles, in complete autonomy – no human operator, no ad-hoc external signals, and no remote laptop! This means that

all the complex computations are done directly aboard the vehicle and very fast. The visual navigation engine is composed of both a software and a hardware part. [34]

When it comes to the future, the simulated pollination of agricultural plants by means of nano copter can provide collecting and delivering pollen in the mode of automatic control. A design of nano copter for pollination can be made on the basis of innovative modification of existing model by its reprogramming with regard to its flight controller that is to be fully adapted to computer interface. The robotic system is offered specially for artificial pollination in conditions of greenhouses and minor agricultural enterprises. [10]

2 Problem statement

The utility of smaller drones are immense, where it can be used in surveillance, toys and potentially to also be part of a swarm of drones. Although, there are smaller drones existing in the current market, we would like to challenge ourselves to build one ourselves, where certain goals ranging from functionality to budget are listed below.

2.1 Primary goals

- Net maximum weight of the drone is 250 grams. Weight under 250 grams ensures it falls under A1 category in EU regulations. 1
- Flight time of 20 seconds.
- Stress of the structural system should not exceed rupture point. System does not experience fracture.

2.2 Secondary goals

- Flight time of minimum one minute.
- Can land with acceleration less than 9.8 m/s2.
- Stress of the drone system should not exceed the yield point. System does not experience plastic deformation.
- Drone is remote controllable.
- Drone can fly in formation with another identical drone.
- Total production cost of the drone is under 500 DKK (Not including remote controller).
- Drone can play audio.

2.3 Constraints

- Budget for entirety of project is 2000 DKK.
- Time available to finish the project is 4 months.
- Drone should have a minimum hover time of 5 seconds.
- Drone should be fully functional and able to take off again after landing.
- No use of flight controller software or unmanned vehicle Autopilot software Suite, capable of controlling autonomous vehicles.

3 Test Specifications

3.1 Primary goals:

- To test this, the drone will be weighed with a scale of a precision on 0,1 grams.
- In order to test the flight time, a stopwatch will be started from the moment the drone leaves the ground and is stopped as soon as it lands.
- This goal will be the tested trough FEM, ensuring that the chosen material for the drones body, will not rupture.

3.2 Secondary goals

- This will be tested with the same method as primary goals tests point 2.
- This will be tested with a mobile phone, recording the drones landing, using the drones position compared to the timestamp of the video.
- This will be tested with the same goals as primary goals test point 3.
- This will be tested by the possibility of sending wireless signals to the drone, with the drone reacting to those send signals.
- This will be tested purely by ear, listening to the drones output.
- This will be tested by mobile phone video, looking at the drones positions at given timestamps.
- This will be tested trough summing the price for each single part, ensuring that it doesn't exceed 500 DKK.

3.3 Constraints

- This will be done with the same method as the secondary goals test, though ensuring the project cost is over 2000 DKK.
- To evaluate the time constraint point of the project, the goal fullfilments will be evaluated in the end of the project period. In the case that all primary goals are fulfilled, the constraint is succeeded.
- This will be tested with a stopwatch, ensuring that the hover time is atleast 5 seconds.
- This will be tested with making the drone take off right after a landing, making sure that the drone is fully operational at the second take-off.
- This will fulfilled by not employing any of the aforementioned in the drone.

3.4 Morphology table

In this section the morphology table is provided where the specific functions of the product and their potential solutions are provided. Three different possibilities for a combination of solutions were chosen in total and they are marked with the following respective colours.

Functions	Solutions			
Number of Propellers	4 	6 	8 	
Motor type	DC 	Brushless DC 	Stepper 	
Drone MCU	Seeed Studio XIAO nRF52840 Sense	Arduino Nano 33	ESP32	
Remote controller MCU	Raspberry Pi Pico	ESP32	Seeed Studio XIAO nRF52840	
Remote controller communication	BLE	NRF24L01	MQTT	CoAP
Remote controller physical	Joystick	Buttons	Switch	Touchscreen
Battery	Li-Po	Li-ion	Ni-Cd	LiFePO
IMU	6-axis	9-axis		
Logging	SD card	Wireless data transfer	Flash memory	

Figure 2: Morphology table for quadrotor design

3.5 Evaluation matrix

Below is the evaluation matrix, where all the combinations of solutions are graded according to each criterion with a certain importance. The criterions stem from the project goals and constraints. For every combination each grade given to a criterion is multiplied with the importance and then added with the other grades in their respective criterions multiplied with their importance.

Criteria	Importance	Solution 1	Solution 2	Solution 3
Manufacturing	2	3	2	1
		Absence of ESCs and a smaller number of propellers and motors make the drone easier to configure and manufacture.	Getting the ESCs for the BLDC motors and 9-axis IMU to function contains a longer time commitment.	Configuring ESCs and 6 motors to work together requires higher time commitment.
Weight & compactness	3	3	2	2
		Li-Po batteries contribute more compactness and less weight.	Li-ion batteries weigh more than Li-Po batteries.	Li-ion batteries weigh more than Li-Po batteries.
Cost	1	3	2	1
		Brushed DC motors and 6-axis IMU contribute to cheap costs.	BLDC motors are generally more expensive than brushed DC motors.	Higher number of motors equal higher costs.
Performance (thrust and reaction time)	3	2	3	2
		6-axis IMU cannot accurately measure yaw of the drone.	9-axis IMU gives more accurate readings, and the BLDC motors provide more thrust and torque.	6-axis IMU cannot accurately measure yaw of the drone.
Total		**24**	21	15

Figure 3: Quadrotor evaluation matrix

3.6 Risk Assesment

To ensure the safety under operation and development of the drone, a risk assessment is created for the drone project, based on the Fine & Kinney method [22].

#	Risk	Description	Actions to minimize risk
1	**Safety risks in a operational perspective**		
1.a	**Collision** [R=1.5]	Collisions by the drone flying into either items or persons	Ensuring that the drone is only operated in enclosed spaces or under restraints, so that the drone is unable to fly into unwanted areas.
1.b	**Wear/breakage of components** [R = 0.3]	E.g. by a propeller being operated in a broken condition	Inspecting the propellers and other vital components before every flight or test session.
1.c	**Battery charging malfunctions, fire** [R = 1.75]	Charging the battery with incorrect charger or under undesirable conditions	Ensure that the batteries are only charged with the correct charger.
2	**Safety risks concerning the development phase**		
2.a	**Collision** [R=1.5]	Collisions caused to malfunctions in sensor readings or adjustments of control software giving unwanted results	Ensuring that the drone is only operated in enclosed spaces or under restraints, so that the drone is unable to fly into unwanted areas.
2.b	**Wear/breakage of components** [R = 1.5]	Incorrect mounting of components causing breakage or failure	Checking components after mounting, ensuring correct mounting
2.c	**Battery charging malfunctions** [R = 1.75]	Charging the battery with incorrect charger or under undesirable conditions	Ensure that the batteries are only charged with the correct charger.
2.d	**Shorting** [R = 6]	Having unwanted connections in the produced PCB or a wire connection soldered onto an incorrect pad.	Checking all connections compared to the schematic and checking for unwanted shorts with a multimeter in continuity mode
2.e	**Cuts** [R = 0.1]	Getting cut by propellers	Never operating the drone unrestrained and close to people.

Figure 4: Risk assesment table

Marked in all the risks are the risk scores calculated by the Fine & Kinney method of Risk, Risk score = Probability * Exposure * Consequence. These risk scores are then compared in the risk clusters of the method, with five defined classes. After calculating the proposed risk ratings of the found risks, it's apparent that all risks fall under the lowest factor of the Fine & Kinney method of scores the: 'Risk; Perhabs acceptable category'.

3.7 GitHub Interaction

Each semester project has a main focus-point for educational purposes. The main focus-point for SPRO4 is control engineering which covers implementations such as IMU combined with programming and PID for the drone. This means a lot of the project development will be software related.

Working together in a large group for a single deterministic purpose can be cumbersome, so in order to better track progress, a GitHub repository has been made to share files and collaborate more effectively. GitHub requires little to no extra effort to implement and use, and will only require a small change in work culture to use correctly, potentially saving a lot of time.

The GitHub repository is an easy way to share and keep track of the software development progress, and gives many advantages when working with embedded systems.

3.7.1 Advantages of Using GitHub

- GitHub works by having one main code or branch which is shared between everyone. Users can then create their own branches and work on something individually, before it is sent back to the main. This means past revisions of work can be saved and reviewed by everyone, and changes can easily be reverted if a fundamental mistake occurs- or is spotted during development. Mistakes in the software are also easier to identify and can be solved without having to change the main program at all. This also means that changes can be made without interfering with other peoples workflow, giving a clear advantage when working/collaborating on similar problems in the software. [35] With GitHub, open source software becomes easily available and secure, while also being compatible with other GitHub repositories. This means GitHub can also be used to keep dependencies up to date for already existing projects, and provide live updates to multiple libraries and dependencies automatically and safely. [33]

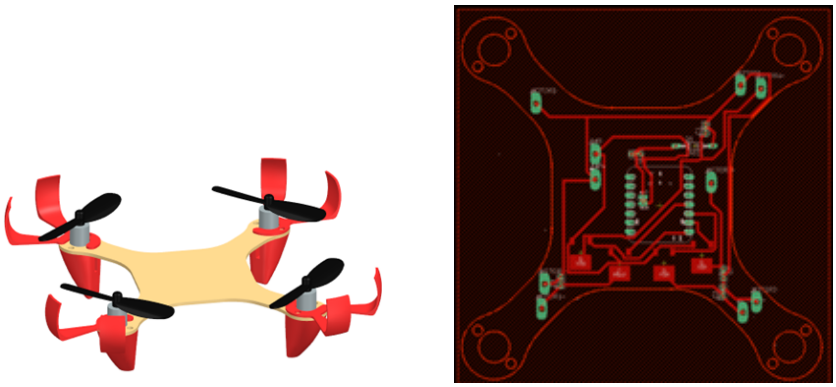
3.7.2 Usefulness Beyond the Project

GitHub is very standard among many engineering companies, and is one of the most widespread UML service. Its used by many companies in the industry, so its very useful to familiarize with GitHub before entering jobs, especially in software development. And with GitHub's wide usage and accessibility it has quickly become the worlds largest software collaboration tool in the world with over 100 million developers. [21] GitHub is also owned by Microsoft, and works with Microsoft programs, which is very standard for many work environments, providing high compatibility.

All in all, GitHub seems to be a necessary skill to have in the future as an engineer.

4 Design and manufacturing of quadrotor

The main body of the drone is created as a PCB, in order for the body to host both the structural element and the electrical connections of the drone. Since the project formulation's main goals dictates the drones size and mechanical properties, the drones size was chosen to be $100mm^2$, so that the PCB could accommodate all the needed components of the drone. The PCB shape itself was designed in Siemens NX as a simple outline with holes for the motor mounts. This outline was exported in a DXF format into Autodesk Eagle were the actual PCB was designed from the needed electrical circuit. The electrical schematic was created in Eagle's schematic designer and then automatically converted to a PCB board with the auto-router feature. The trace width of the PCB is specifically chosen to be 30 mills so it can supply the needed 1.72 amps [7] for the motors at maximum rpm. The microcontroller of the drone is mounted by pin headers, the motors by the motor mounts and the remaining components by soldering onto the top layer. The position of the microcontroller is offset from the center, in order to ensure that the In-built 6-axis IMU is centered to the exact middle of the PCB.



(a) CAD View (b) PCB View

Figure 5: Drone model

In order to control the speed of the motors, one of the potential options would be to use an H-bridge to control the DC motors. However that would be excessive, as the motors does not need to run in reverse as contain many components As a less power consuning simple and lighter option, a single N-channel mosfet was used for each motor.

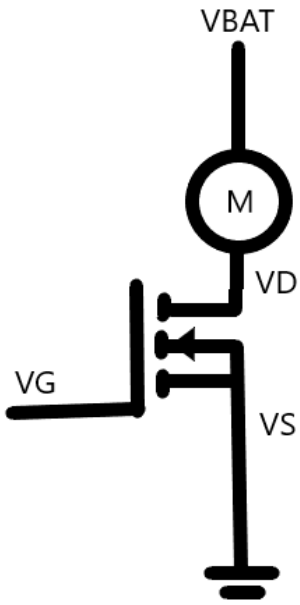
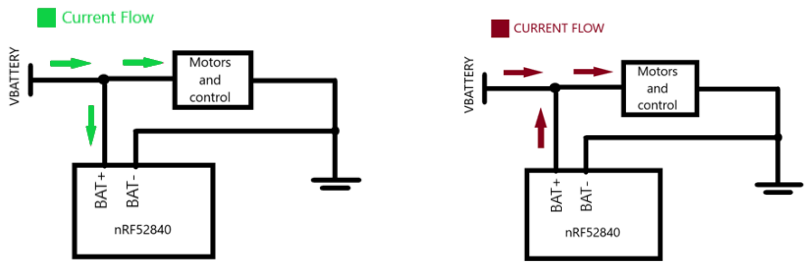


Figure 6: Motor control with MOSFET.

As illustrated in figure 6, the drain of the MOSFET is connected to the motor, which is supplied by the battery, and the source of the MOSFET is grounded. Meanwhile, the VG pins are connected to one of the PWM pins of the nRF52840 MCU, where the opening of the gate is proportional to the PWM. Thus, when VG (simulated by PWM) is smaller than the V threshold of the MOSFET, the motors are static, and if VG surpasses V threshold, then the motors start spinning with higher RPM as PWM is increased. The MOSFETs used for this situation are FDD8896 [4], as it has a low threshold voltage of 2.5V (MCU pins can supply up to 3.3V), and can handle up 94A in continuous drain current.

As there is high currents being drawn from the motors, one of the things needed to be configured was protection against overcurrent being drawn from the microcontroller. When the motors are switched on or off, a surge current arises which can result in additional overcurrent being pulled from the microcontroller if the battery is not alone capable of supplying the current.



(a) Continuous Current (b) Potential Surge current

Figure 7: Current flows

In order to prevent reverse current from the MCU, during a surge, one of the potential options to utilise would be a diode placed in the following configuration.

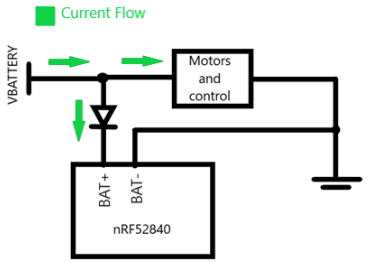


Figure 8: Diode added to prevent reverse current flow from MCU.

Additionally decoupling capacitors were added in parallel to the positive and negative motor pins and the MCU BAT+ and BAT- pins.

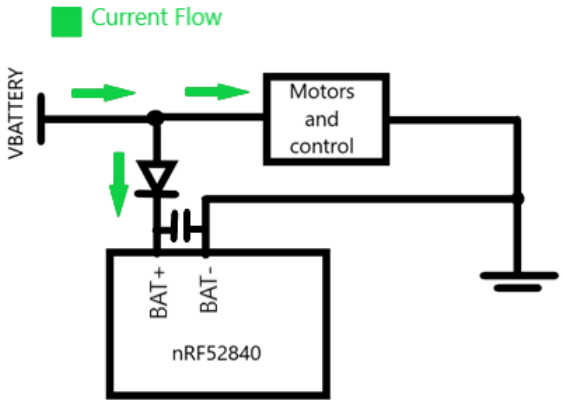


Figure 9: Example of decoupling capacitor for stable voltage to MCU.

The diode utilised is 1N5818 [8], as it has a low forward voltage of under 0.5V, resulting in low power losses. Moreover, the decoupling capacitors are cermaic capacitors as they generally smaller in size, and they are rated at 100 nF, which follows the general guidelines of decoupling capacitor values. [2]

With the combination of the CAD-outline and the electrical schematic, PCB Gerber files could be output for production. The only excess parts needed for the drone would be motor mounts and prop guards (prop guards solely for testing, not for final use). The Motor mounts act as both landing legs and motor mounts. They would be designed in order for the motors to press fit into the central hole, mounted onto the PCB by screws into the motor mount going through the PCB. The designed mounting parts was printed in PLA on a Prusa MK3S+ 3D printer. The Parts were printed with a single perimeter and 3% infill in order for each leg to weigh 1.1 grams while still having the nescessary structure to have a stable press-fit.

(a) Motor Mount (b) Mounted Motor (c) Prop Guard

Figure 10: 3D-Printed drone parts

The project formulation states that the "Stress of the drone's structural system should not exceed rupture point. System does not experience fracture". This was tested through Ansys Mechanical Static Structural Analysis. Using the CAD model of the drone, the following boundary conditions were applied: Fixed constraint on the bottom area (excluding the drones arms), Z direction forces (upwards) was applied on each of the motor mount screw holes according to the found motor thrust, a single force vector in the Z direction (downwards) was applied to the top face (excluding the drones arms), to resemble the gravity force caused by the weight of the drone. Besides the boundary condition, the setup contained a mesh refinement of 3 steps, a material assignment of FR4 Fiber Glass Epoxy Board and a solution setting of equivalent stress. Below is a picture of the stress simulation of the initial design, which showed clear stress concentrations in the corners where the motor arms runs into the body section. The stress concentration had a magnitude of 1.6839 MPa which is satisfactory regarding the requirement of fracture, since the tensile strength of the given material is 320 MPa [6].

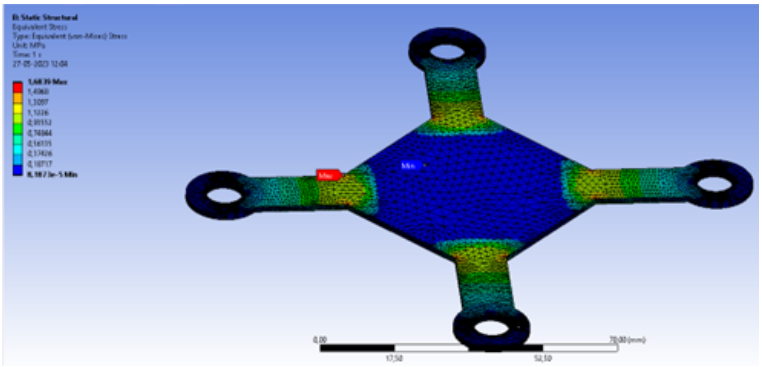


Figure 11: Initial Ansys Simulation

Although it was clear that the drone body would be strong enough to operate under max power, the stress concentrations were undesirable. Therefore 20 mm radii was added to the stress all corner points of the drone body. Repeating the simulation with the same boundary conditions, it was clear that the added radii removed the stress concentrations and instead distributed the stress over a broader area of the motor arms. Furthermore, it

was found that the maximum stress was found to be 0.87154 MPa, which is approximately half the maximum strength of the initial model that had the stress concentrations.

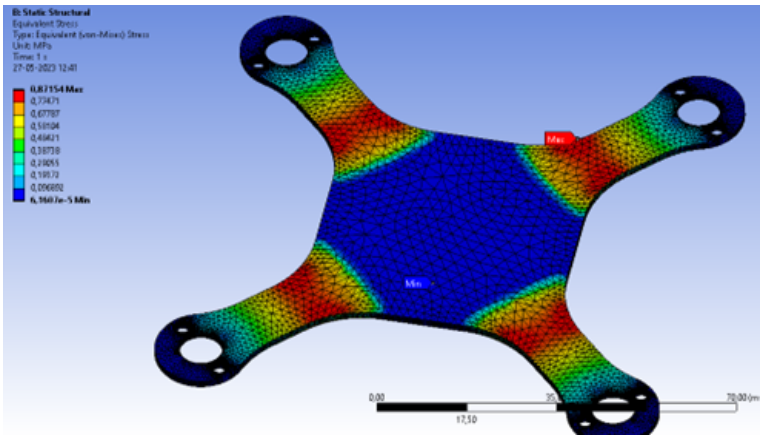


Figure 12: Final Ansys Simulation

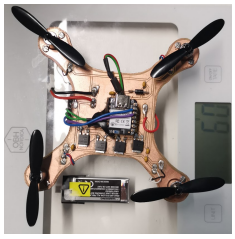
To power the drone a single cell Turnigy Li-Po battery is used, at 3.7 - 4.2 V and 300 mAh. The battery is chosen specifically to be able to supply the needed current of the drone. With a C-rating of 35C, the battery is able to discharge continously at 10.5 amps which is sufficent for the electrical circuit. The motors used are 8520 brushed Coreless DC-motors, which properties are characterized in the control chapter under thrust testing using the stock 2-bladed propellers that the motors were shipped with.

(a) Turnigy Battery (b) 8520 DC-motor

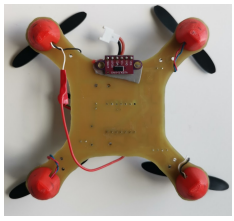
Figure 13: Motor and Battery used on the drone

After assembly of the drone including all the parts, it became apparent that the initial weight estimation of the drone was too low. After weighing the final physical setup it weighed 70 grams, XX grams over the intital estimation. This of course meant that the drones motors had less excess thrust than initially expected, meaning less control authority. In an attempt to mitigate this, the option of 3-bladed propellers were explored since they in theory would be able to provide extra thrust. The 3-bladed propellers were printed with SLA in the Tough 2000 resin on a Form 3 machine. Testing the new propellers in the same thrust setup used to characterize the motors using the 2-bladed propellers, it

showed that the 3-bladed propellers produced 12 grams total extra (combining 4 motors), but also pulled a total of 4 amps extra. Since the battery used does not have the sufficient continous amp supply this option was not further explored, instead lightweighting options was used to reduce the total weight to 60 grams. In order to have an alitude feedback a GY-VL53L0XV2 ToF sensor was emplyed on the underside of the drone, to measure the drone-to-ground distance. In development a budget was created to ensure that the possibilty of achieving the project goals was always possible, the budget for the project can be found in appendix C.



(a) Motor Mount



(b) Mounted Motor

(c) Prop Guard

Figure 14: 3D-Printed drone parts

5 Control prototyping

5.1 Plant Model Equations

This section of the report will explain how the plant model equations are obtained. Followed by how these equations are implemented in MATLAB/Simulink.

The plant model is a series of equations with multiple variables, which uses inputs, states, and parameters to get a certain output. For this project, the plant model is designed to take a height input the drone should hover at, paired with the inputs from the sensors, to achieve the desired hover height and stability.

5.1.1 State Equations

The drone possesses 6 degrees of freedom, enabling it to navigate in various directions and perform rotations. It can move along the X, Y, and Z axes, while also being capable of pitch (rotation around the Y axis), roll (rotation around the X axis), and yaw (rotation around the Z axis). We can identify the precise angle and position of the drone at any time by defining the axis of rotation and its accompanying state variables. [16]

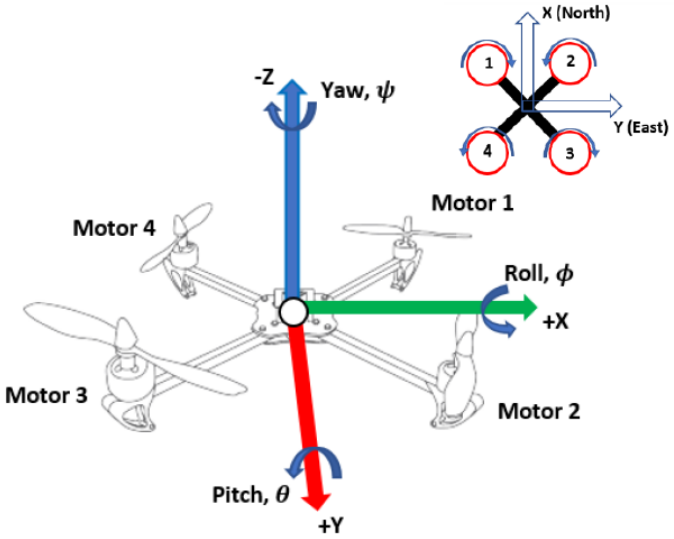


Figure 15: Quadcopter 'x' configuration

The drone's states include critical physical properties like as acceleration, velocity, and position, which are all detected by sensors. A gyroscope is used to record the angular changes of the drone. An accelerometer is a device that measures acceleration along the x, y, and z axes. Since the gyroscope drifts and the accelerator has measurement spikes when subjected to high acceleration, a complementary filter will be needed, which will be discussed in the "IMU" subsection. A dedicated infrared sensor is also used to measure distances in the z-direction, allowing for exact location information. A state table can be found in Appendix E.

Before plant model equations can be generated, parameters for the drone need to be calculated. The parameters are what explains the drone mathematically and is an integral part of modelling the drone. The more accurate these values are the more realistic the plant model will be. The parameters needed can be seen in the table below: [16]

Symbol	Description	Unit
I_{xx}, I_{yy}, I_{zz}	Inertia moments	$Kg\ m^2$
J_r	Rotor inertia	$Kg\ m^2$
l	Rotor axis to copter center distance	m
b	Thrust coefficient	N/s^2
d	Drag coefficient	Nm/s^2

Figure 16: State table

5.1.2 Inertia Moments

A moment of inertia is a constant value that explains the force/torque required to rotate an object in the direction of the moment. A desired angular velocity around a rotation

axis can be found. The moments of inertia are found using estimation. If it is assumed that the drone is symmetrical in all axis, the moments of inertia can be found using parallel axis theorem to approximate the moments for each individual part of the drone in the x, y, z directions. For the calculations it's assumed that the centre of mass for the drone is perfectly in the centre of the drone's body. The equations used for each individual part of the drone can be found in. [16]

For the approximation smaller electrical components were not considered, such as wires and mosfets, since their collective total weight is less than a gram, they are deemed negligible for the accuracy estimation.

Motor MoI	$I_{motor,x} = 2\left(\frac{1}{12}m(3r^2 + h^2)\right) + 2\left(\frac{1}{12}m(3r^2 + h^2) + md_{a2a}^2\right)$ $I_{motor,y} = 2\left(\frac{1}{12}m(3r^2 + h^2)\right) + 2\left(\frac{1}{12}m(3r^2 + h^2) + md_{a2a}^2\right)$ $I_{motor,z} = 4(\frac{1}{2}mr^2 + md_{a2a}^2)$
Arms MoI	$I_{Arm,x} = 2\left(\frac{1}{12}m(w^2 + h^2)\right) + 2\left(\frac{1}{12}m(h^2 + d^2) + md_{a2a}^2\right)$ $I_{Arm,y} = 2\left(\frac{1}{12}m(w^2 + h^2)\right) + 2\left(\frac{1}{12}m(h^2 + d^2) + md_{a2a}^2\right)$ $I_{Arm,z} = 4(\frac{1}{12}m(w^2 + d^2) + md_{a2a}^2)$
Battery MoI	$I_{Battery,x} = \frac{1}{12}m(w^2 + h^2)$ $I_{Battery,y} = \frac{1}{12}m(h^2 + d^2)$ $I_{Battery,z} = \frac{1}{12}m(w^2 + d^2)$
Flight Controller MoI	$I_{FC,x} = \frac{1}{12}m(w^2 + h^2)$ $I_{FC,y} = \frac{1}{12}m(h^2 + d^2)$ $I_{FC,z} = \frac{1}{12}m(w^2 + d^2)$
Rotor moment of inertia	$J_r = \frac{1}{2}mR^2$

Figure 17: Inertia moments equations

A mathematical document can be found in the appendix as "SPRO4math". Which shows the detailed process of calculation.

$$I_{rotor} := 729$$

$$I_x := 48932.24391$$

$$I_y := 49929.46266$$

$$I_z := 99299.32284$$

Figure 18: Moments of inertia in the x, y, z axis, in $g * mm^2$ as calculated from equations, used in the final version of the quadcopter

5.1.3 Characterization of Motor

Some drone parameters can not be determined due to a lack of data concerning the motors used in the project. The motors are of a commercial product, but with no available data sheet. This means that motor constants and thrust coefficients needs to be determined through testing. Other motors within the same weight class were available, but those other alternatives did not provide the thrust desired compared to their weight, since the weight requirement set for the project is around 40 grams. For the same reason the motors currently in use are good, since they have a good thrust to weight ratio, with the caveat of not having a data sheet
The next part of the report is a small summary of the tests done to determine the data needed for the motors. The full reports concerning testing and acquiring these values can be found in the appendix.

- Thrust testing can be found in appendix "Motor Thrust Testing"
- Motor constants testing can be found in appendix "Motor Testing"

5.1.4 Thrust testing

For thrust testing the motor is placed in a frame on a scale to measure the change in weight when the motor is operating. A sketch of the setup can be found in appendix A2,2. The motor used had an operating voltage window of 0-4,2 volts, therefore the motor was tested between 1-4,2 volts incremented with 0,1 volts, and the weight change noted for each increment. Characterizing the motor yielded the results seen in the figure below. As a general rule of thumb, provided by project supervisors, the motor's need to produce double the thrust of the total weight of the drone to operate ideally. At maximum voltage level of the battery, a single motor produces 22.2 grams of thrust. In conclusion,

at maximum voltage level the drone can ideally weigh 44.4 grams in total. The battery will slowly discharge meaning that it's impossible to always have max output, so it should be a little less than 44.4 grams in practice.

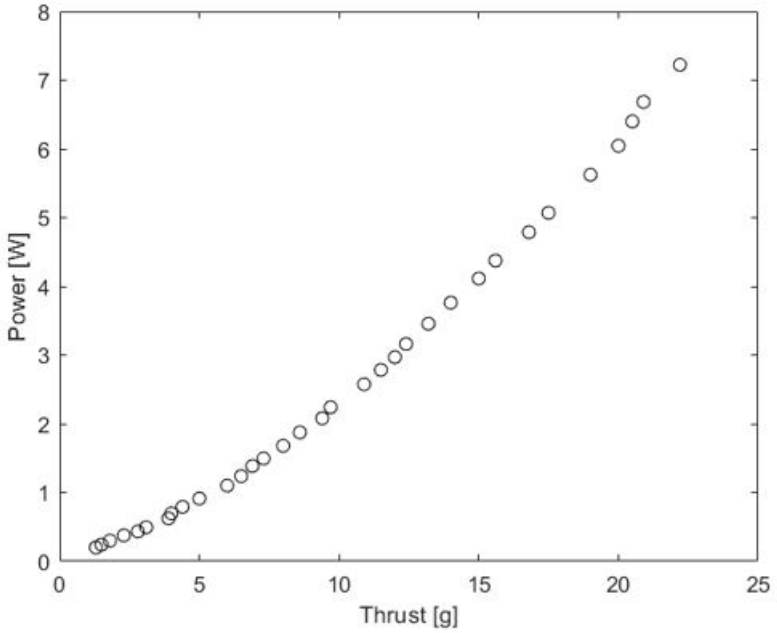


Figure 19: Motor Thrust Graphed

5.1.5 Motor Constants, and Thrust Coefficient

Motor constants relevant for this project include, "Motor velocity constant" (K_v) and "Motor torque constant" (K_T) With a simple setup the motor's RPM can be tested proportionally to PWM. The motor was tested in 20 increments, using a photo sensor to measure RPM. Using the data from the test, we get the following result. [9]

$$K_{\mathrm{v}} = \frac{\omega_{\text{no-load}}}{V_{\text{peak}}} \qquad K_{\mathrm{T}} = \frac{\tau}{I_{\mathrm{a}}} = \frac{60}{2\pi K_{\mathrm{v(RPM)}}}$$

Figure 20: Equations used for calculating thrust- and velocity constants

$$K_v := \frac{\omega}{V} = 8943.333333 \frac{RPM}{V}$$

$$K_\tau := \frac{60}{2 \cdot \pi \cdot K_v} = 0.001067755861 \frac{N \cdot m}{A}$$

Figure 21: Motor Constants

Combining the two test data sets it is possible to determine the thrust coefficient of the motor. Using the graph from the thrust test the thrust can be calculated by taking the voltage drop over the motor. This thrust is then divided with the angular velocity to get the thrust coefficient ($TC = 3.59E - 05$).

The constants/coefficients are calculated around 3.0V inputs, which is the voltage needed to achieve the theoretical hover thrust. This is sensible since most of the time the drone will be operating around these values.

5.1.6 Environmental Forces

The drone is very small, weighing only approximately 55grams as per requirement and with little surface area. Because of this, aerodynamic effects have been neglected for the modelling. Environmental forces mainly include but are not limited to wind resistance and liftoff turbulence.

5.1.7 Generating Plant Model Equations

Now with the all the states and parameter defined, plant model equations can be generated. First, we will look at how the motors should operate together to move the quadcopter in the desired direction. A quadcopter can move in x and y directions using pitch and roll. When the quadcopter rotates around one of its axis, all the rotor blades will generate thrust at an angle, moving the drone in a direction. It is important to control the individual motors so the drone won't tilt too much, causing it to flip in the air. On the picture below the different cases of directional movement for the quadcopter can be seen.

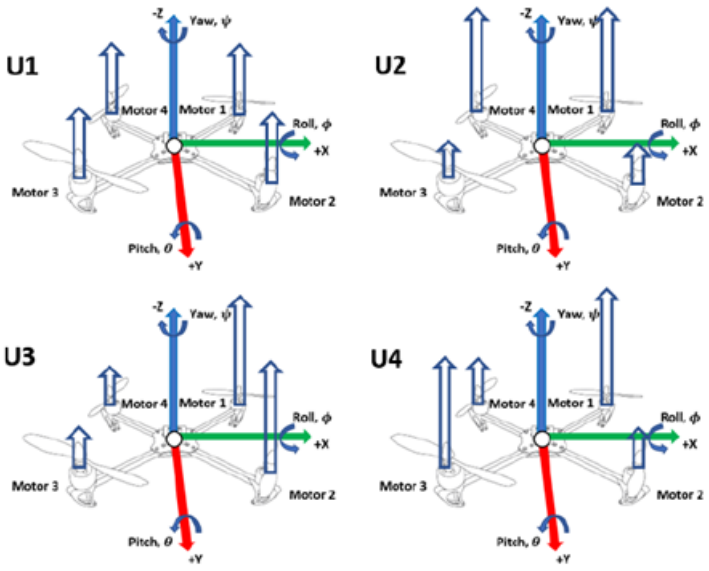


Figure 22: Visual representation of thrust equations [16]

Here U1 is the general thrust behaviour, providing thrust equally and symmetrically to move only the Z axis. U2 creates a roll, thrusting the drone in the Y direction. U3 creates a pitch, thrusting the drone in the X direction. U4 increases the thrust for motors turning the same direction and decreases the thrust of the motors rotating the opposite direction, allowing the drone to rotate in the Z axis e.g., yaw due to non-symmetrical rotational forces created by the moments of the motors. [16]

Each motors thrust is then defined as angular rotational speed to acquire the following equations.

Input	Thrust Equation	Description	
$U1_x$	$b(\Omega_1^2+\Omega_2^2+\Omega_3^2+\Omega_4^2)$	General Thrust	(2.8)
$U2_x$	$b\sin\left(\frac{pi}{4}\right)(\Omega_1^2-\Omega_2^2-\Omega_3^2+\Omega_4^2)$	Roll Thrust	(2.9)
$U3_x$	$b\sin\left(\frac{pi}{4}\right)(\Omega_1^2+\Omega_2^2-\Omega_3^2-\Omega_4^2)$	Pitch Thrust	(2.10)
$U4_x$	$d(\Omega_1^2-\Omega_2^2+\Omega_3^2-\Omega_4^2)$	Yaw Thrust	(2.11)

Figure 23: Thrust equations [16]

The plant equations for this project were gotten from source [16]. The source has a more in depth description of how the plant equations were found. A summary of all the plant equations can be found in appendix E.

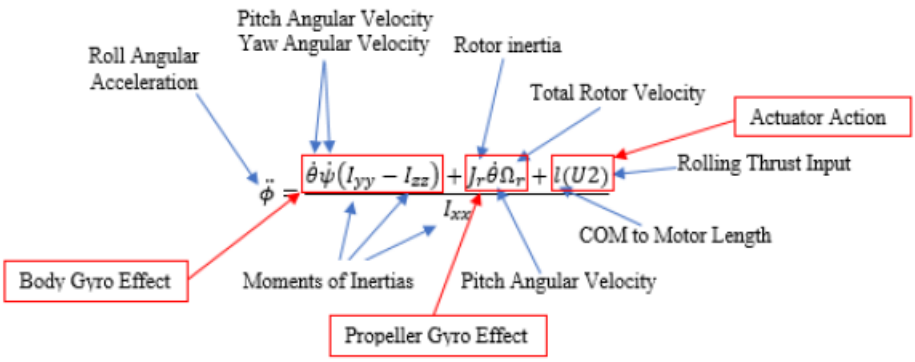


Figure 2.14-Rolling acceleration equation explained

Figure 24: Example of plant equation for phi. [16]

Body gyroscopic effects from changes in the quadcopter orientation. Propeller gyroscopic effects from propeller rotation and angle/orientation Actuation action forces produced by the rotors/propellers.

5.1.8 SIMULINK/Matlab Implementation

To make the Simulink model the thrust equations must be transferred into the program using the angular speed as inputs and the motor parameters. Below we see the thrust

equations in Simulink with the inputs on the left and outputs on the right. (The outputs are U1, U2, U3, and U4, as well as Omega) In SIMULINK the math is done using "blocks", the addition and subtraction is performed first, and then the next blocks take the product. Essentially the math is converted using blocks in SIMULINK.

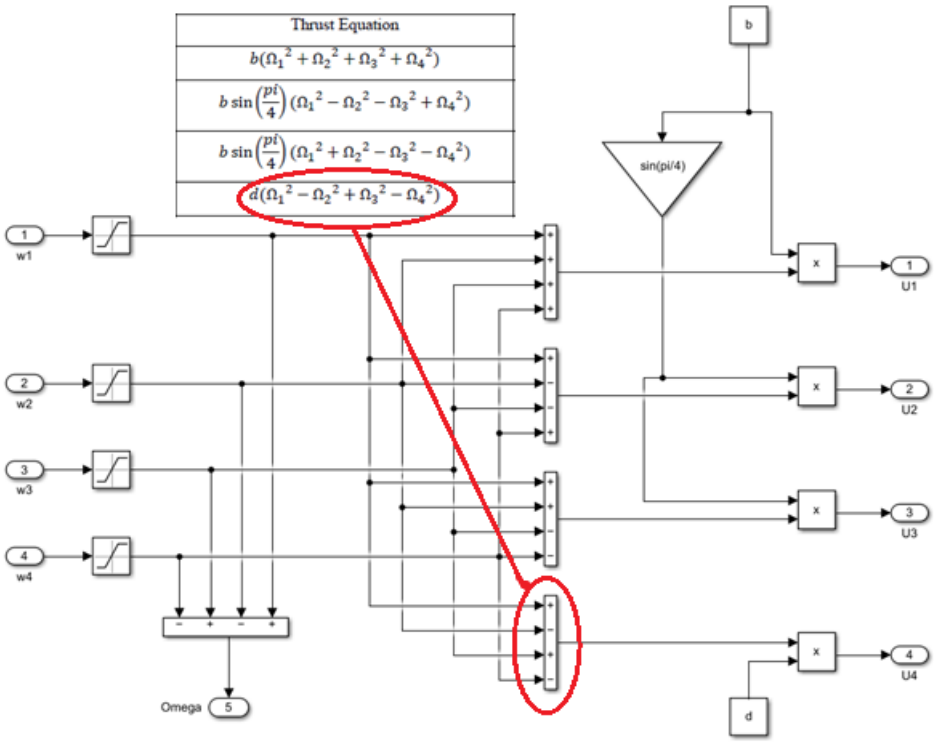


Figure 25: Thrust equations in SIMULINK

The rest of plant equations are then also transferred to SIMULINK using the same process, using blocks to represent the math. The next part of the plant takes the thrust equations as an input to calculate theta, phi, and psi, for position, velocity, and acceleration.

Below is an example for the equation of phi double dot. We use the thrust equations and feedback together with the drone parameters to get the result. Since all the equations need feedback from the other plant equations, an initial value must be set. It makes sense to make the initial value 0 since we haven't moved yet and there has been no change in any direction or angle. Throughout the flight the drone will then fill out the theta, phi, and psi values using the censors to determine the change in position. For example, if the drone is slanted a bit in the direction of motor 4 and 1, the drone will have a change in the roll aka phi value. The censor will detect the magnitude of this change, and U2 will increase to give more thrust to those motors to stabilize. Thrust is increased proportionally to smoothing the stabilization.

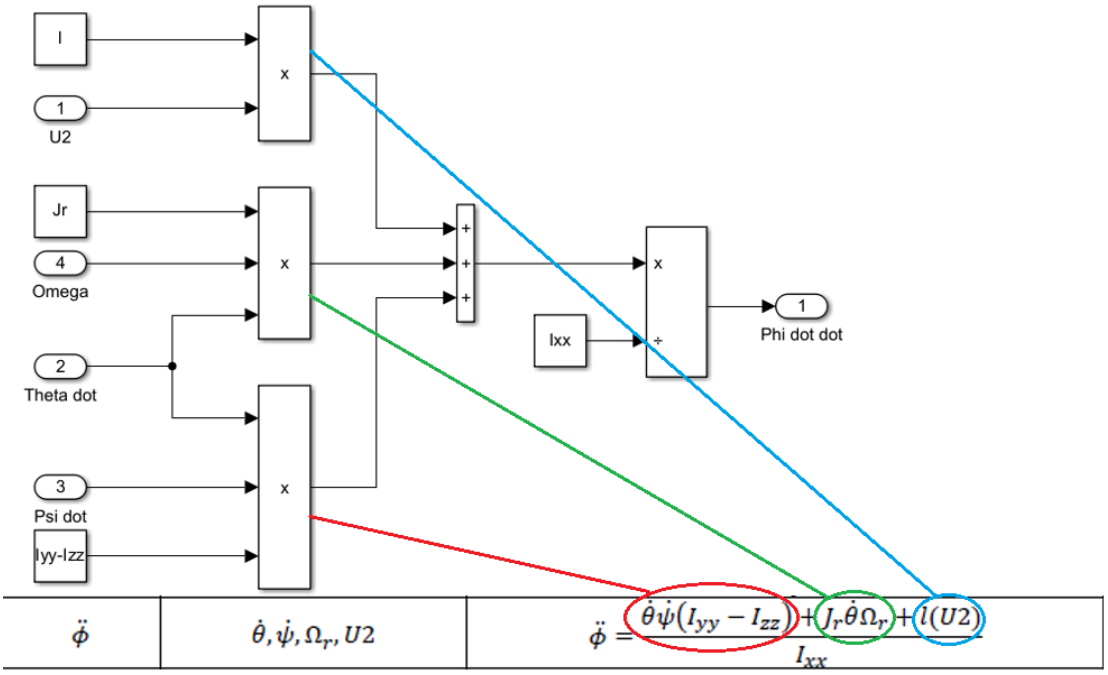


Figure 26: Plant equation for phi double dot

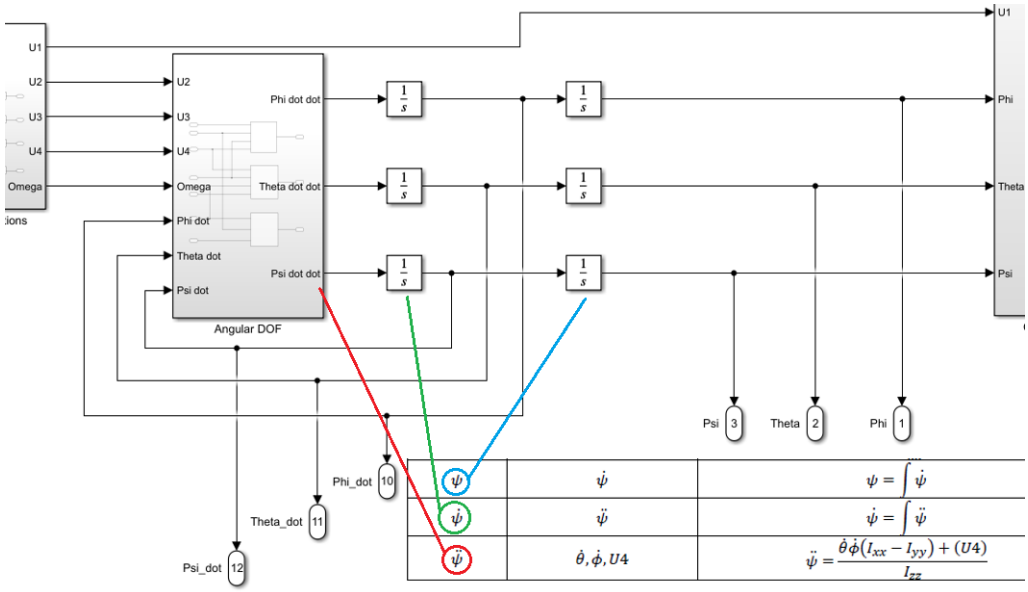


Figure 27: Angular equations overview example

To finish out the plant, the X, Y, and Z values for acceleration, velocity, and position are calculated using the values from theta, phi, and psi, these equations rely on feedback from each other in the same way that angles do, and use the same principles when it comes to determining the initial value.

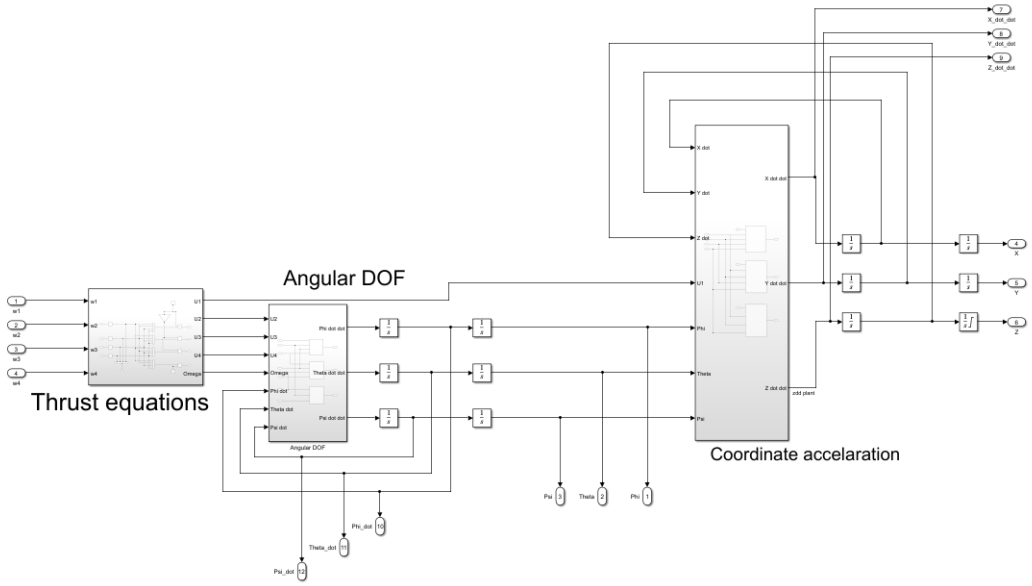


Figure 28: Plant overview in SIMULINK

5.2 Controller

In order to fly the drone in a controlled manner, a PID controller is implemented. The controller takes a desired state (for example hovering at 1.5 meters) and subtracts the output of the system, to create an "error signal". This error signal tells the drone how displaced it is based on the magnitude of the error. The job of the controller is to correct this error through PID to drive it as close to 0 as possible, achieving a stable state/flight.

A more in depth explanation of how the PID works will be presented later in the report.

In the figure above we see the controller model. The highlighted part is the change in Z. The PID outputs a new angular speed needed, which is divided by the change in theta and phi, since if the drone is tilted at an angle there will be a change in upwards thrust. This new value is then divided with the motor thrust coefficients to determine the magnitude of the change in Z compared to the thrust the motors provide. In short, the controller takes these changes Z depend on, and mathematically determines the change needed in angular speed for each motor, in order to go towards the desired height. [16]

This process is very much the same for the change in phi, theta, and yaw. The depending changes affect the outcome of the angular speed, which is put into the thrust equations to determine change for each specific motor, until the drone is stable at the set values.

It's important to note that a saturation block is put right before the output of the system to set an upper limit on the output to 3500, which is the maximum angular speed possible for the motors. This ensures that the model is realistic with the real life motors.

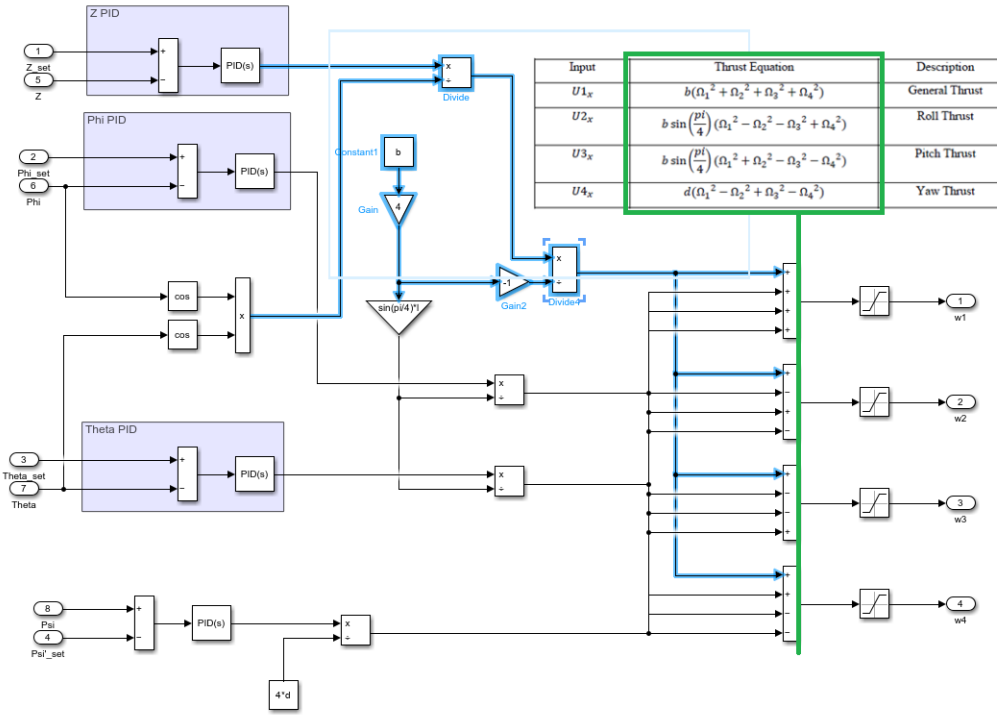


Figure 29: Overview of controller, with Z highlighted

5.3 UAV Toolbox

To achieve a visual way for checking if the quadrotor plant has been constructed correctly, the UAV Toolbox was used. (See Figure 30) For studying PID control, the MATLAB UAV 3D simulation provides a realistic and interactive environment. The simulation was used to examine the behavior of the UAV system rather than relying exclusively on theoretical concepts and mathematical equations. Experiments with different PID controller parameters were conducted, to see how they affect the quadrotor's reaction and acquire hands-on experience tweaking the controller. This immersive learning experience aid in the integration of theory and practice. Control performance in complex circumstances can be analyzed using the MATLAB UAV 3D simulation. When numerous PIDs are involved, it can become difficult to intuitively understand how one effects another; thus, a visual representation aids in grasping and understanding the concepts.

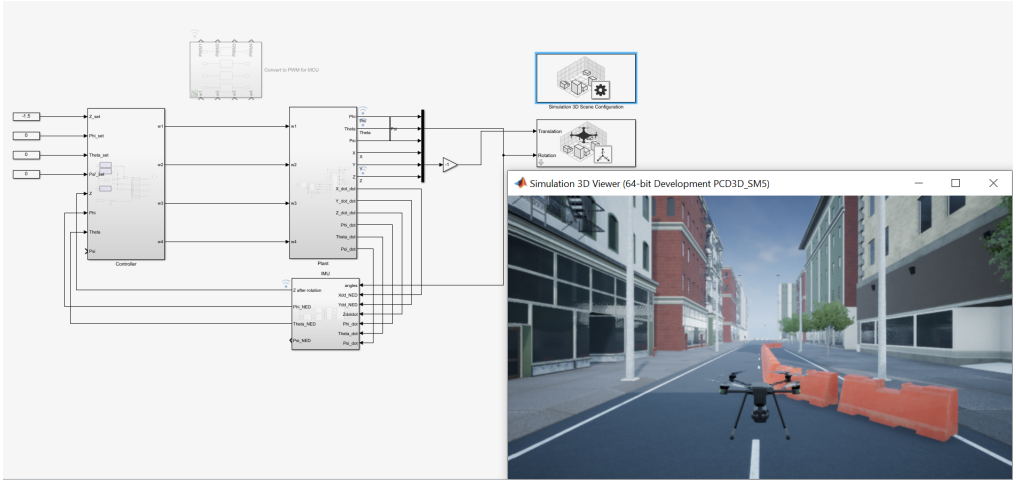


Figure 30: Ilustration of the workflow, with the utilization of the UAV Toolbox

5.4 The theory of PID

5.4.1 Principles of PID Control

PID (Proportional-Integral-Derivative) control is a popular feedback control algorithm that is utilized in a variety of control systems. PID control is made up of three parts: proportional, integral, and derivative. The proportional term controls the process based on the current difference between the desired setpoint and the measured process variable. To decrease steady-state mistakes and improve system stability, the integral term accumulates past errors over time. The derivative term computes the error's rate of change and aids in dampening quick changes, improving response time and lowering overshoot. [28]

5.4.2 PID Control Loop

The PID control loop is made up of four basic components: the process or plant under control, in this case roll and pitch, the process or plant under control, and the process or plant under control. A sensor or measurement device. The PID controller and the actuator. The IMU measures the process variable (angle) and compares it to the desired setpoint. The PID controller calculates the control signal based on the mistake, and the actuator alters the system accordingly. In this closed-loop feedback system, the control signal is continuously changed to keep the process variable close to the setpoint. [28]

5.4.3 Benefits of PID Control

PID is a versatile and frequently used control algorithm that provides excellent control in a variety of applications. Other solutions exist, such as model predictive control, but they are less widespread and thus have a smaller community. [20]

Stability PID control helps maintain system stability by continuously adjusting the control signal based on the error feedback. The proportional, integral, and derivative terms work together to provide a balance between stability and responsiveness. [28]

Robustness PID control is robust and effective in handling disturbances, noise, and external factors that may affect the system's performance. The integral term, in particular, helps eliminate steady-state errors caused by external disturbances. [28]

Adaptability PID control can be tuned and adjusted to optimize performance based on specific system requirements. The controller gains can be modified to enhance stability, reduce overshoot, or improve response time. [28]

Simplicity PID control is relatively easy to implement and understand. It offers a straightforward approach to control systems without requiring complex mathematical models or extensive computational resources. [28]

5.4.4 Ziegler-Nichols method

The performance of a PID controller depends on appropriate tuning to match the characteristics of the controlled system. Tuning involves adjusting the proportional, integral, and derivative gains to achieve the desired response. To tune the quadrotor's PID Ziegler-Nichols method was used, as it is compromise between complexity and time. [30]

The value of the proportional-only gain that allows the control loop to oscillate forever at steady state is used to calculate the ultimate gain, K_u . This means that the gains from the I and D controllers are set to zero in order to determine the influence of P. It evaluates the robustness of the K_c value in order to optimize it for the controller. The final period is another key number related with this proportional-only control tuning approach (P_u). The ultimate period is the amount of time it takes to complete one entire oscillation while the system is in steady state. These two parameters, K_u and P_u, are employed to determine the controller's loop-tuning constants (P, PI, or PID). [14]

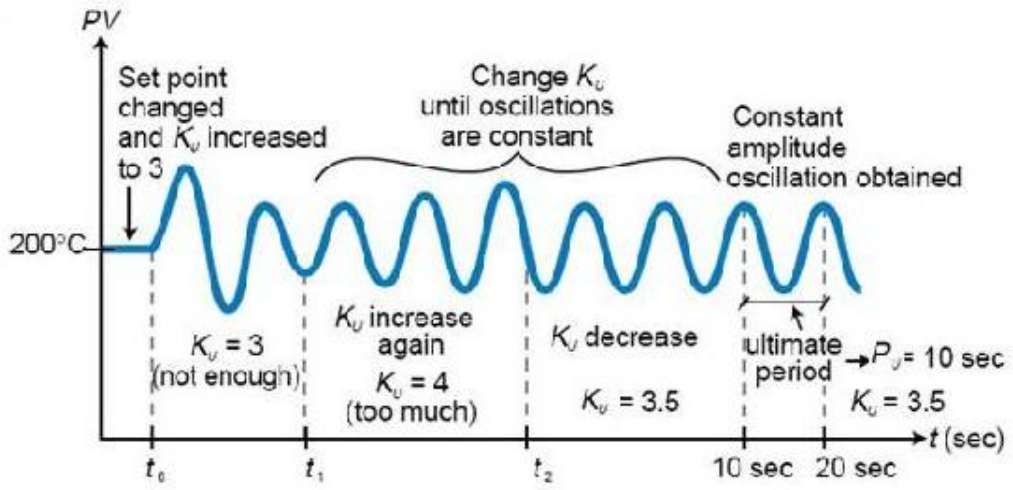


Figure 31: Example of a system tuned using the Ziegler-Nichols method [14]

Ziegler-Nichols method steps are as follows:

1. Take out the integral and derivative actions. Set the integral time (T_i) to 999 or the highest number possible, and the derivative controller (T_d) to zero.
2. Change the set point to cause a slight perturbation in the loop. Increase or decrease the gain of the proportional until the oscillations have a constant amplitude.
3. Mark down the gain value (K_u) and period of oscillation (P_u).
4. Input these values into the Ziegler-Nichols equations to find the controller settings. (See Table 1)

Control Type	K_p	T_i	T_d	K_i	K_d
P	$0.5K_u$	–	–	–	–
PI	$0.45K_u$	$0.83T_u$	–	$0.54K_u/T_u$	–
PD	$0.8K_u$	–	$0.125T_u$	–	$0.1K_uT_u$
classic PID	$0.6K_u$	$0.5T_u$	$0.125T_u$	$1.2K_u/T_u$	$0.075K_uT_u$
some overshoot	$0.33K_u$	$0.5T_u$	$0.33T_u$	$0.66K_u/T_u$	$0.11K_uT_u$
no overshoot	$0.2K_u$	$0.5T_u$	$0.33T_u$	$0.4K_u/T_u$	$0.066K_uT_u$

Table 1: Formulas for determening the PID constants via Ziegler-Nichols

5.5 Linearization of the plant

Multiple sources indicate, that linearization needs to be considered. “PID is a linear-type controller and hence is only efficient for a limited operating range when used to control non-linear processes.” [12] “PID controllers perform well only on linear systems or systems that are linearized.” [23] “PID control can only be applied for linear plant dynamics 99 percent of applications” [25]

As the quadrotor, is a non-linear system, due to its non-linear terms in the plant equations, to use PID, linearization would be needed. It is planned to have three PID controllers: Z, ϕ/roll, and θ/pitch. The equations below indicate that the only non-linear equation is Z.

$$\ddot{\phi} = \frac{\dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) - J_r\dot{\theta}\omega_r + l(U_2)}{I_{xx}}$$

$$\ddot{\theta} = \frac{\dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) - J_r\dot{\phi}\omega_r + l(U_3)}{I_{yy}}$$

$$\ddot{Z} = \frac{mg - (cos\theta cos\phi)U_1 - A_z\dot{Z}}{m}$$

The non-linear terms are due to ϕ and θ. There is a simple way of linearizing the Z-axis plant equation, as the intended goal is only hovering. For hovering, ϕ, and θ would be zero, which would be ensured due to PID control to respective axis, they would become constants, thus the non-linear terms

$$cos(\phi) * cos(\theta) \longrightarrow cos(0) * cos(0) = 1$$

$$\ddot{Z} = \frac{mg - (cos\theta cos\phi)U_1 - A_z\dot{Z}}{m} \longrightarrow \ddot{Z} = \frac{mg - U_1 - A_z\dot{Z}}{m}$$

Thus the plant Z-axis equation is linearized. Only downside, is to accept this assumtion, it should be ensured, that ϕ and θ PID controllers should be calibrated first, to ensure, that the values remain zero, when calibrating Z-axis PID.

5.6 The simulation of PID

5.6.1 ϕ and θ hyper-tuning

As mentioned before, the calibration in ϕ and θ axis should be done first. As they are linear terms, the Simulink PID tune app can be used. Before using the tool, one must understand the usable sliders. By adjusting the Response Time slider, how aggressively the controller should respond to changes in the setpoint can be specified. If the Response Time is increased, the controller will be more aggressive in responding to any difference between the current output and the desired setpoint, it may also cause more overshoots or oscillations. Setting the Response Time to a lower value will cause the controller to respond more slowly to setpoint changes, resulting in smoother but potentially slower response times. Transient Behavior Slider: The Transient Behavior slider in the PID tune app regulates the contribution of the derivative term. The derivative term aids in dampening oscillations and lowering overshoot, as discussed in the "Theory of PID" chapter. The PID tuner app was used to hyper-tune and obtain results that could then be used in the physical setup and fine-tuned, thus the two sliders were set in the middle

for later adjustment. The results were optimal. (See figure 32 and 33)

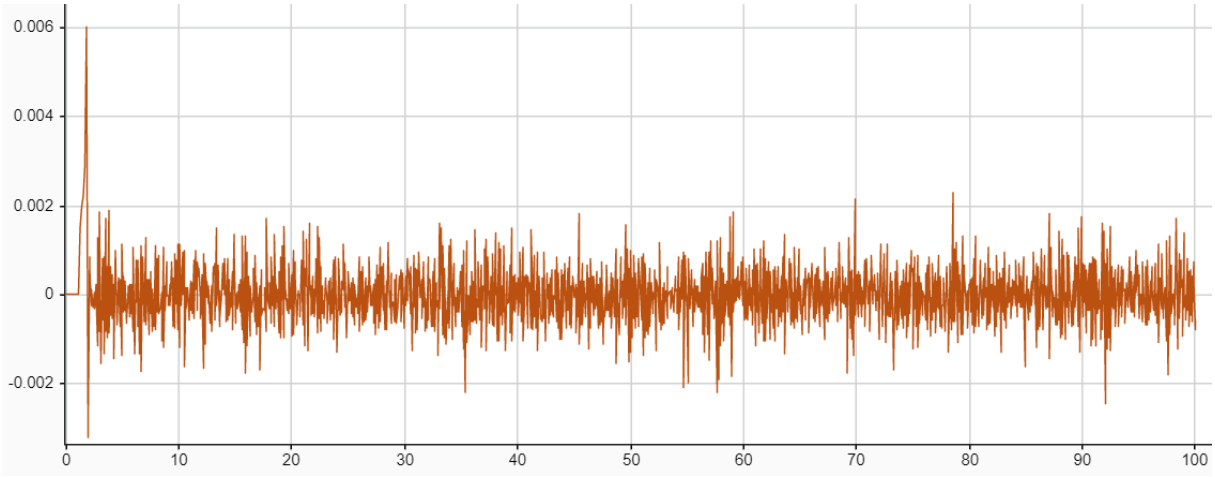


Figure 32: Response of ϕ, with hyper-tuned PID

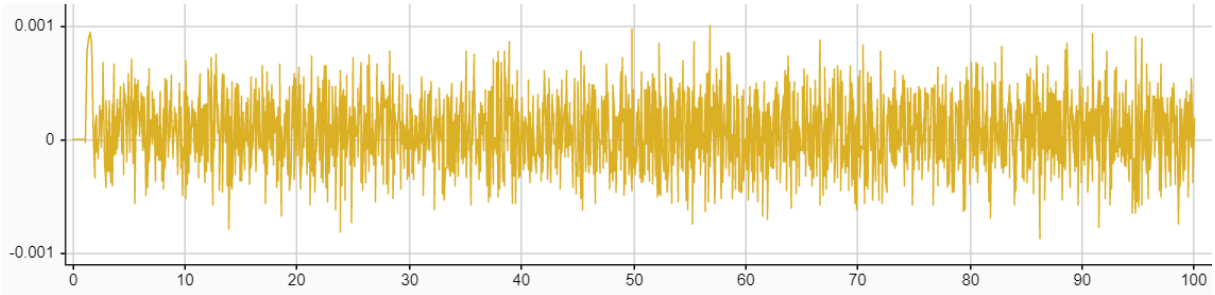


Figure 33: Response of θ, with hyper-tuned PID

5.6.2 Z hyper-tuning

After ϕ and θ have been hyper-tuned, it can be safely assumed, that they are 0, thus the previously discussed linearization can be utilized. The Ziegler-Nichols approach, as discussed in the previous chapter, was employed for calibration in the Z PID controller simulation. To begin, the Simulink PID blocks were adjusted to only have the gain value and then adjusted until an oscillating response was obtained, around a specified setpoint. (See figure 34) Simulink data inspector was used to viewing the output because it also included useful features, such as measuring with two cursors, which came in handy while measuring the period of the oscillation period.

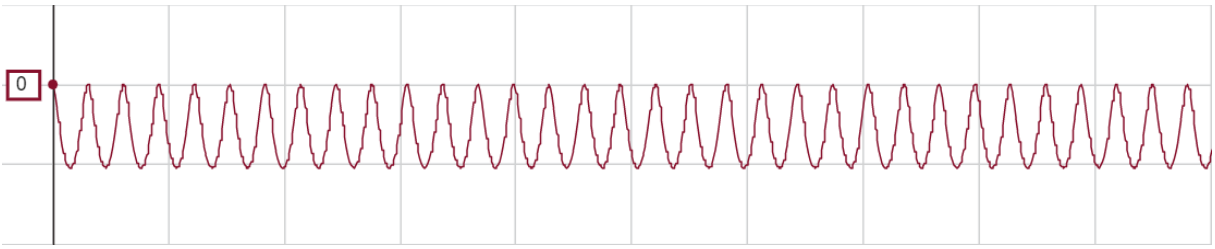


Figure 34: An oscillating response with only gain in Simulink

The measured K_u and T_u, were plugged into the Ziegler-Nichols formula table (See table 1) and then fine-tuned to achieve less of an overshoot and faster response time. The final values for an adjusted PID system where as follows, where red indicates the calculated values, and green - the fine-tuning. (See figure 35)

Control Type	K_p	T_i	T_d	K_i	K_d
P	$0.5K_u$	–	–	–	–
PI	$0.45K_u$	$0.83T_u$	–	$0.54K_u/T_u$	–
PD	$0.8K_u$	–	$0.125T_u$	–	$0.1K_uT_u$
classic PID	$0.6K_u$	$0.5T_u$	$0.125T_u$	$1.2K_u/T_u$	$0.075K_uT_u$
some overshoot	$0.33K_u$	$0.5T_u$	$0.33T_u$	$0.66K_u/T_u$	$0.11K_uT_u$
no overshoot	$0.2K_u$	$0.5T_u$	$0.33T_u$	$0.4K_u/T_u$	$0.066K_uT_u$

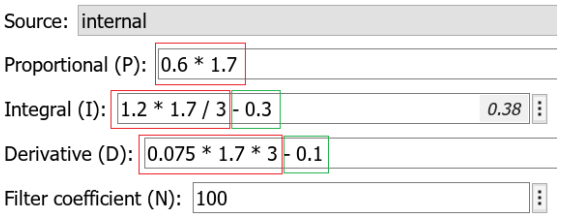


Figure 35: Tuned PID controller block values

After tuning the PID, the output response was ideal. (See figure 36)

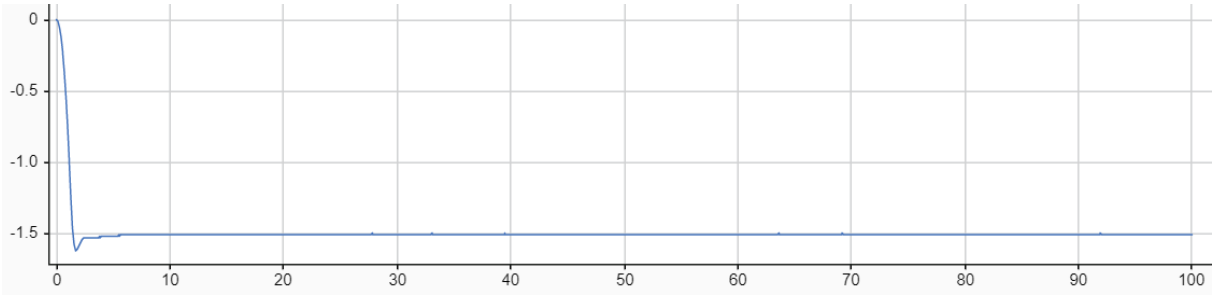


Figure 36

Unfortunately, this is the plant's response when there is no noise, thus when noise is introduced, the response drifts. (See figure 37) This was caused by the integral term summarizing all noise-induced errors. This was a clear indication that more than one sensor was required for accurate Z-axis calibration. Thus, instead of using the IMU Z acceleration as input, an IR sensor was added to the real system and used as input.

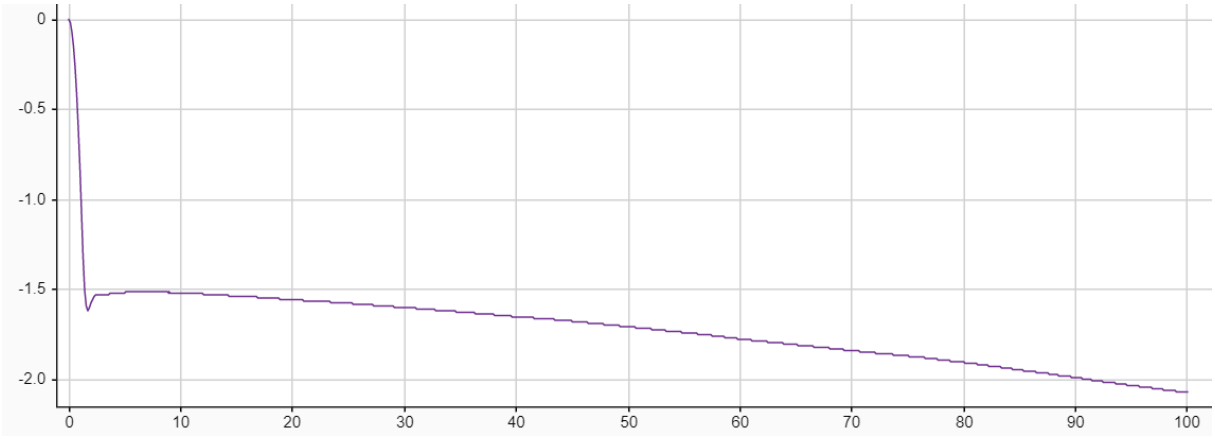


Figure 37: Z-axis response, tuned using the Ziegler-Nichols method

5.7 The Implementation of PID

5.7.1 One-axis setup

Making PID for all axis is possible, but not ideal as determining the source of the problem is more difficult. Therefore, it was decided to tune one axis at a time, excluding the Z-axis which would be tuned after fully calibrating the rest, as it is non-linear. The initial 'P' calculation was made from the simulation formula for maximum thrust. The value was calculated to be 0.0045, but as seen in figure 39, the system is stable, but for the Z-N method, systems need to be marginally stable as shown in figure 40. In figure 39, the PWM effect of PID on motors is also shown to make sure that QuickPID works as predicted and without delay. To test the stability of system, it was displaced by a finger in order to mimic large turbulence.

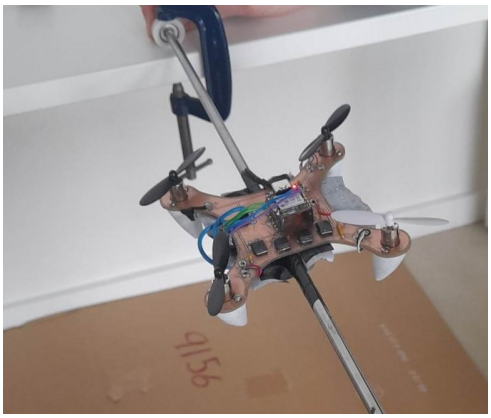


Figure 38: Picture of axis rig

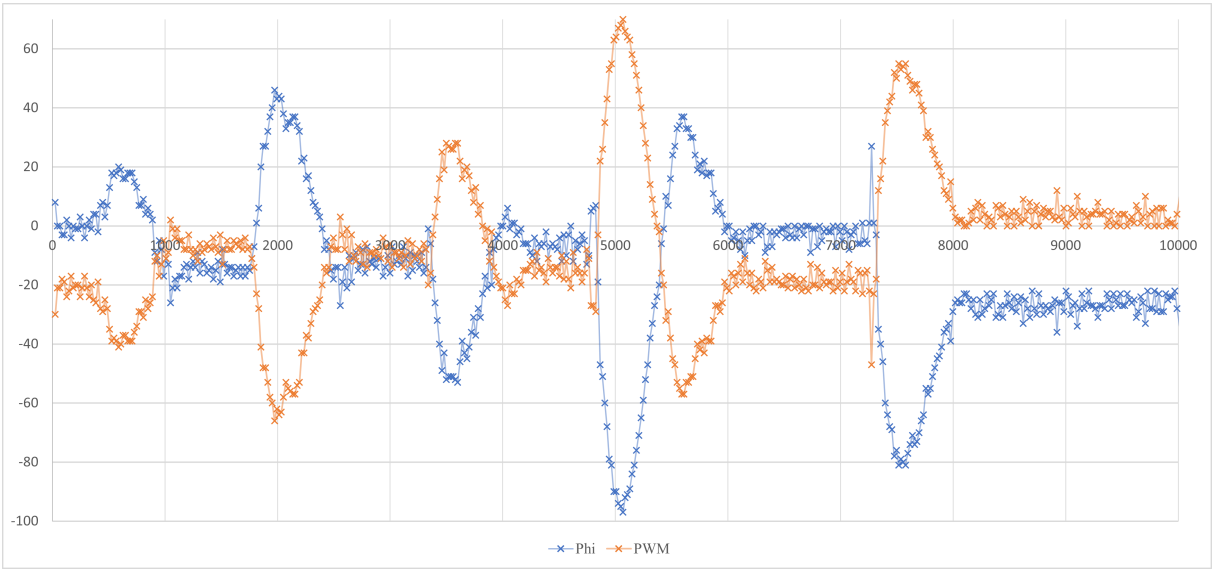


Figure 39: Plot of marginally stable system on Y-axis with purely proportional control

After determining the marginally stable value of P to be 0.0065, the Z-N method was used and applied to the system resulting in figure 41. The system is stable, but has a large settling time and long period. This was improved by enlarging the 'P' constant by 0.002 as shown in figure 42.

For tuning the θ, the same testing rig was utilised and the values of the calibrated ϕ were used as starting point. Those were calibrated and the result is shown in figure 43.

From the testing, it was evident that overshoot was common, but it was chosen as a trade-

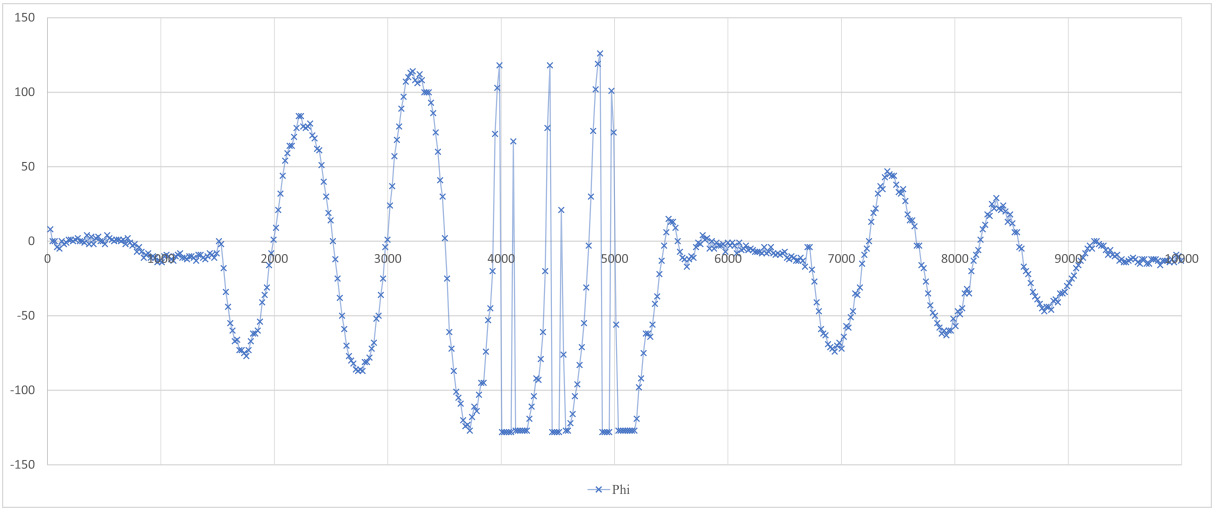


Figure 40: Plot of stable system on Y-axis with purely proportional control

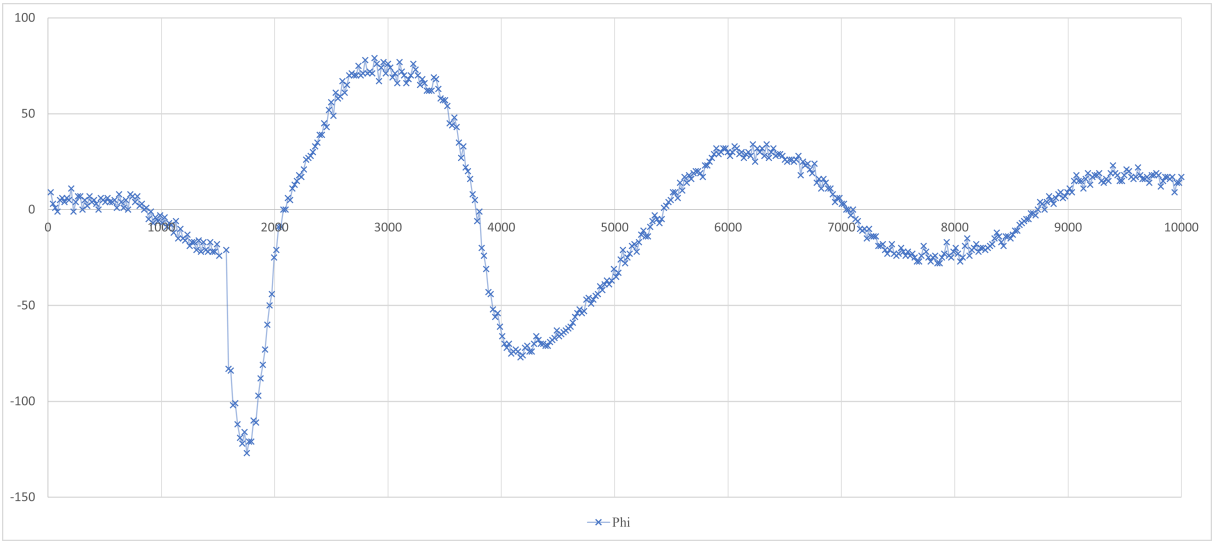


Figure 41: Plot of the drone on Y-axis with values from Z-N method

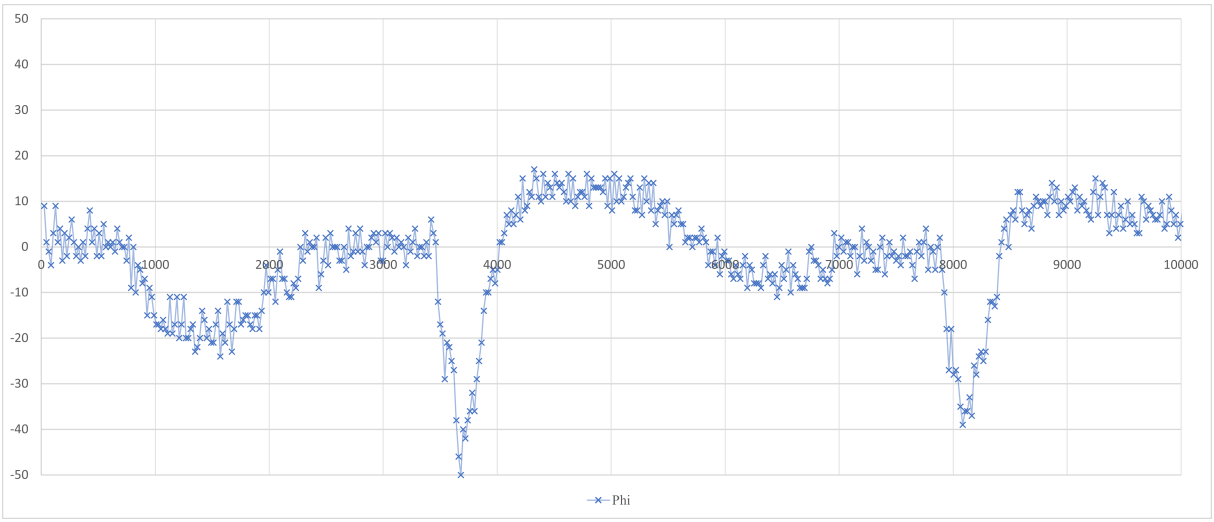


Figure 42: Plot of the drone on Y-axis with calibrated values from Z-N method

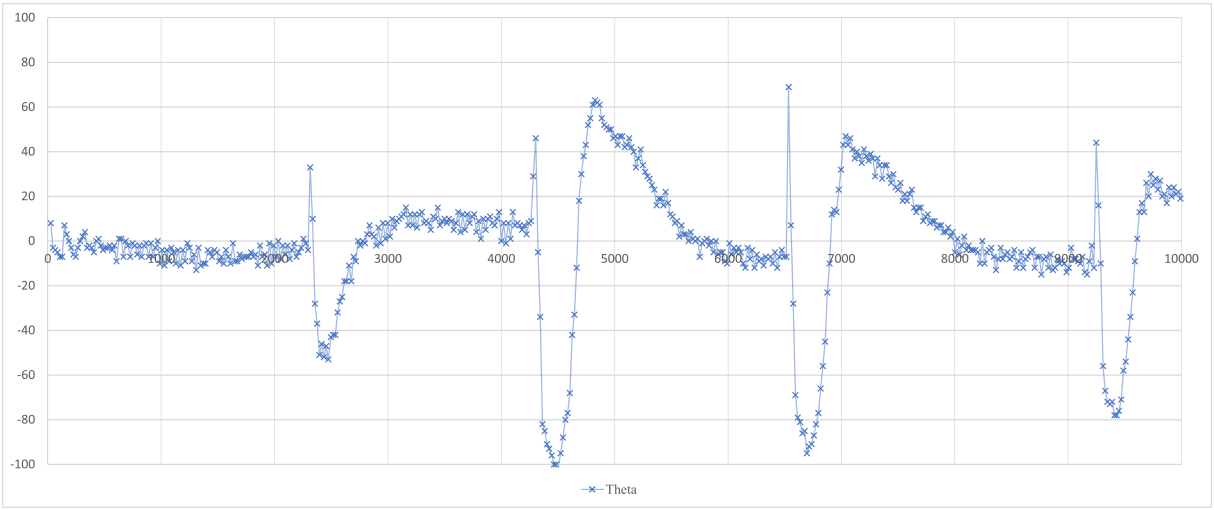


Figure 43: Plot of the drone on X-axis with calibrated values from Z-N method

off for the aggressive transient behaviour. Additionally, the interference from the finger results in a larger deviation in angle than it would in actual free flight circumstances.

5.7.2 Diagonal setup

Testing in the string rig after tuning values in one-axis resulted in uncontrolled oscillations of the drone with no regards to the PID values. Due to that, the diagonal rig was set up and the code was changed accordingly. The same procedure of axial testing was repeated for the diagonal rig and is described with figures 44 and 45.

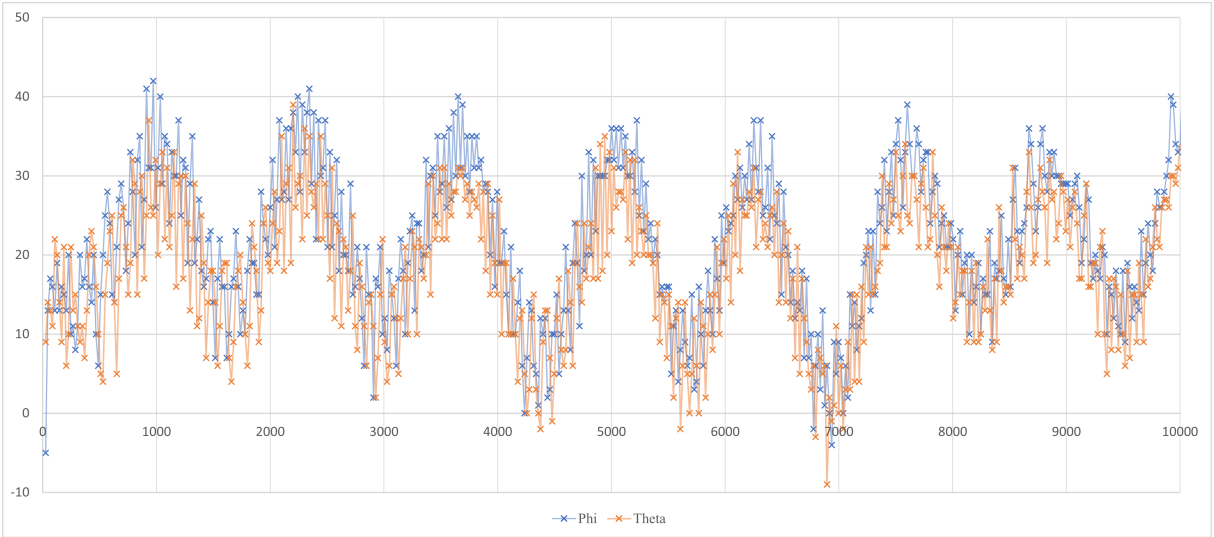


Figure 44: Plot of purely proportional control in diagonal setup for Z-N method

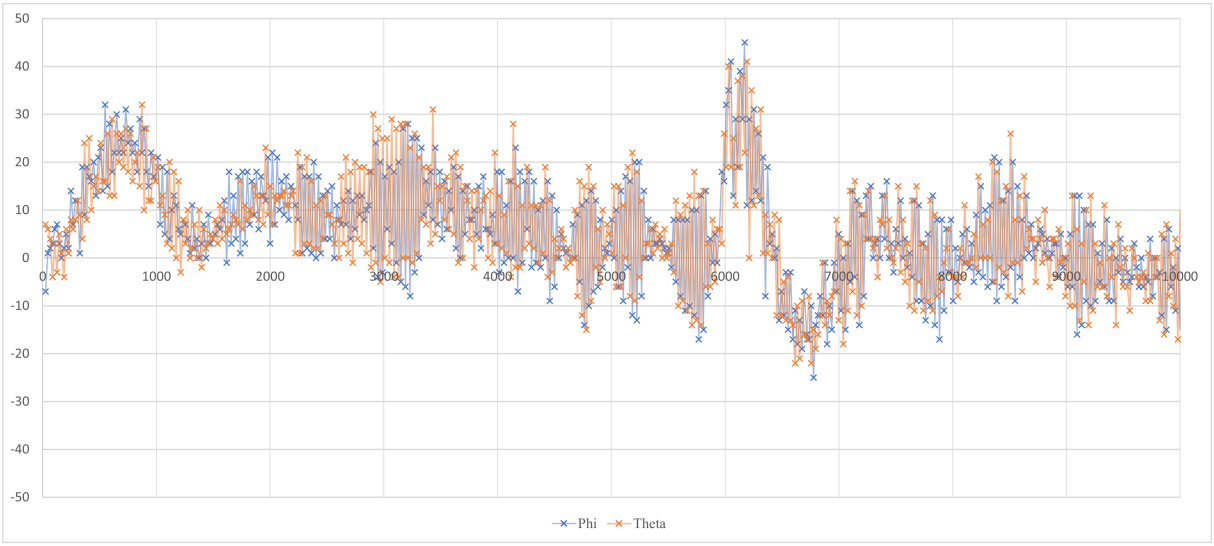


Figure 45: Plot of angles after using Z-N method from 'P' in figure 44

5.7.3 String setup

The final testing rig was chosen to be the string rig (figure 46). A halo was attached to the top of the drone, so that the string attached to the drone would be put through, thus resulting in the strings not colliding with the propellers in different angles.

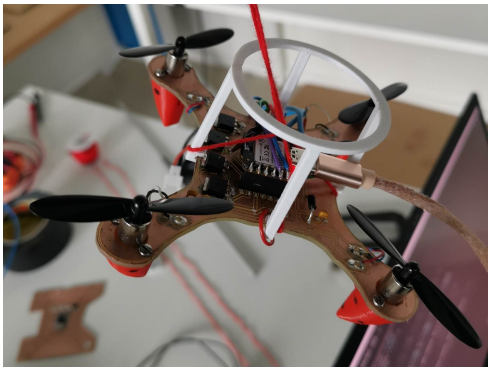


Figure 46: String rig

This provides the closest possible resemblance to flight that could achieved. It may adversely effect the flight characteristics, as the string pulls the drone to the center, which may result in tilt while flying away from the anchor point, but the other option was a gimbal, which introduced large friction.

The values from previous test runs were used at the start, but were found to be insufficient to compensate. Post testing, the ideal values were found to be 10 times the P and 5 times the D compared to the ones used in the diagonal rig. With these values ϕ was stable, however θ was oscillating between stability and instability, as it was initially stable, then became unstable, then became stable again. This was accounted for with higher PID constants for theta.

5.7.4 Conclusion of PID testing

While independent flight was not achieved due to insufficient thrust, the PID system worked in a limited time frame. The instability in θ during string testing was nearly unaffected even with the 5x multiplier. This points to the possibility of the insufficient thrust affecting the authority of the PID control. The other sources of instability could

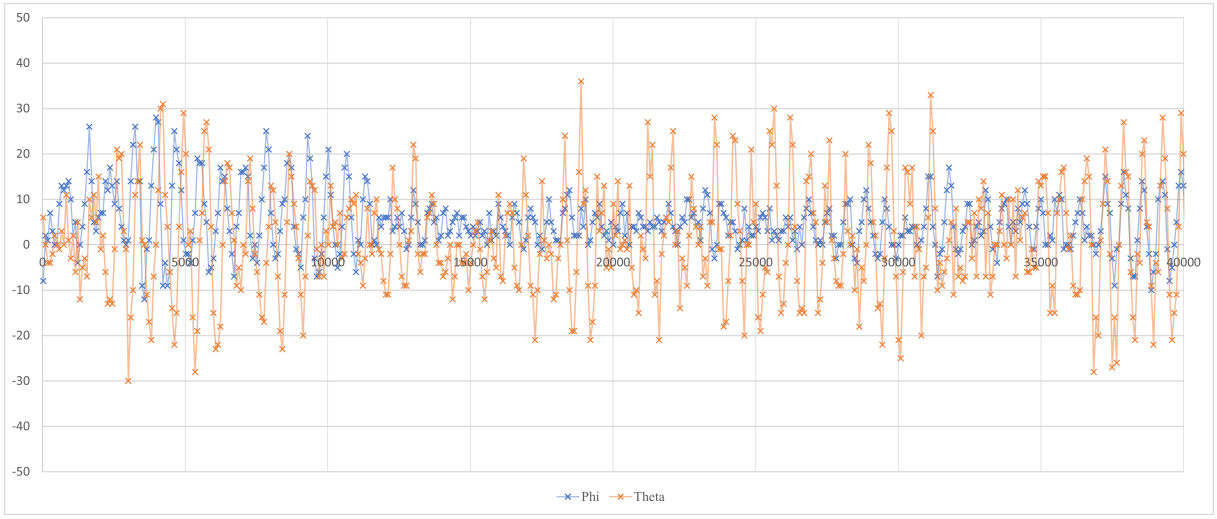


Figure 47: Plot of angles in string setup with calibrated PID values

potentially be the differences between motors in different temperatures and RPMs or IMU deviation and drift.

6 Software prototyping

6.1 Embedded development environment

Software shoud take care of the motor control, IMU output readings and remote control, this could create complications, as essentially they would interrupt each other. A way of multitasking should be introduced. "An RTOS (Real-Time Operating System) is a software component that lets you rapidly switch between different running sections of your code. Think of it as having several loop() functions in an Arduino sketch where they all run at the same time." [27]

After research, the list was narrowed to two top condenders – FreeRTOS and Zephyr. Both solutions are open source, widely used and support the Microcontroller board we have chosen. [43] According to 2018 IoT Developer Survey [17], FreeRTOS is one of the most popular OS used and while Zephyr only received a 2.8 % rating in 2018, it is often described as one of the fastest growing RTOS and in 2022 has become the largest open-source RTOS project by the number of commits and developers. (See Figure 48)

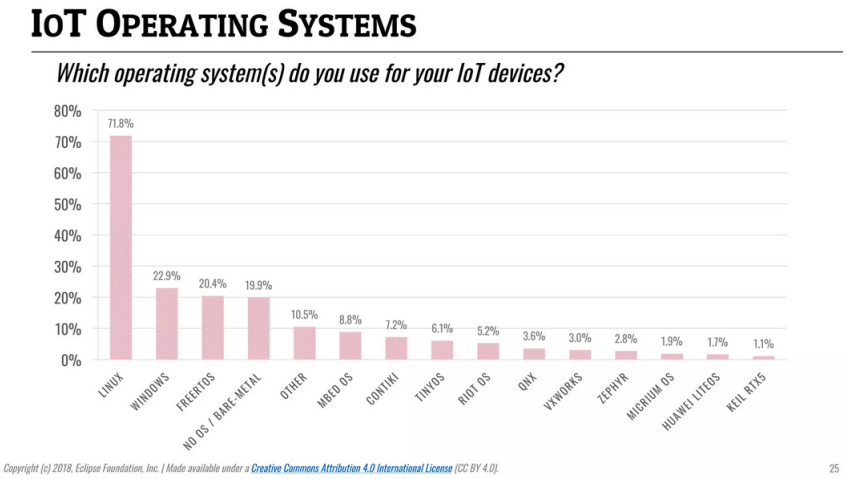


Figure 48: IoT Developer Survey 2018 results

To decide between the RTOS choice, a pros and cons table was created and evaluated. [31] (See Table 2)

RTOS	Advantages	Disadvantages
FreeRTOS	• Suitable for begginers • Constantly improved • Fast code execution • Low memory consuption	• Not event-driven (scheduler will only be called once in a certain period of time) • Less flexible
ZephyrRTOS	• Constantly improved • Designed to ensure energy efficiency • Highly configurable • Event-driven • Kernel can create additional system threads • Possible to exclude multithreading • Additional debugging features • Supported by Nordic Semiconductor	• More difficult to set-up • Potentially complicated to use with no experience

Table 2: Comparison between ZephyrRTOS and FreeRTOS

Overall FreeRTOS is better established and would be easier to use, in case of no experience, whereas Zephyr offers more flexibility and is a fast growing solution, well suited for our application and would be a worthwile investment for building our skillset. [26] After setting up a embedded environment for software development with ZephyrRTOS. (See appendix D), it was conducted, that the use of ZephyrRTOS is possible. After further consultation with the supervisors, it was conducted, that there is a lack of skill in RTOS, more specifically in threading. With the guidence of Davi, it was concluded, that acquiring such skills would be outside the scope of the project, as the main focus is control engineering. In conclusion, the choice was made to use Arduino IDE with the Arduino framework, with the chance of switching to Mbed OS, if a need arises for it.

6.2 QuickPID

For implementing the PID controller on the MCU, the QuickPID library was chosen. QuickPID is an updated implementation of the Arduino PID library with additional features for PID control. [18] It was chosen due to the following reasons:

- Advanced Anti-Windup Mode: When compared to the normal PID implementation, the library features a more robust anti-windup mode. Integral windup can occur when the controller's integral term accumulates error during periods of saturation or when the actuator cannot keep up with the needed control effort, and anti-windup approaches help prevent this. [18] The QuickPID library can lessen integral windup, reduce overshoot, and improve stability during transient situations by adding an advanced anti-windup mode.
- Timer-Based Control: QuickPID is a timer-based control option that lets you use external timers or Interrupt Service Routines (ISRs) for precise timing control. [18] This functionality is especially beneficial in systems with strict timing requirements or when employing external devices or sensors that are synced with the control loop. In this situation, if issues with the IMU or the controller develop, the timer-based technique can simply remedy them.
- Compatibility with Arduino IDE: updated implementation of the Arduino PID library. As the Arduino ecosystem was chosen over Zephyr RTOS using the QuickPID library provides a seamless integration.

When utilizing the library, the PID compute sample time, default = 100000 µs, was changed to 2500 µs to match the IMU 400 Hz measuring frequency. Matching the PID sample frequency with the IMU measuring frequency ensures synchronization, consistent data availability, and accurate control in quadrotor systems. It enables the PID controller to operate based on up-to-date sensor measurements, respond effectively to dynamic changes, simplify system identification and tuning, and optimize computational efficiency. Additionally, the output limits of the PID (0-255 by default), were changed to (-255) – (255), as a negative error can be achieved in the quadrotor system.

6.3 NanoBLEFlashPrefs

Data logging has evolved as a significant approach for overcoming the issues of the PID tuning process. [32] During quadrotor operation, data logging entails recording numerous flying characteristics, sensor measurements, control inputs, and system responses. Engineers and researchers can obtain useful insights into the quadrotor's behavior and make informed judgments to fine-tune PID controller parameters by collecting this comprehensive set of data. [41] Data logging also allows for the comparison and evaluation of various

PID tuning strategies or algorithms. Thru analysis, the influence of each adjustment on the system's behavior by methodically changing PID controller parameters can be observed in the quadrotor's response. This iterative method makes it easier to identify ideal PID parameter combinations that produce the required flight characteristics. The Seeduino XIAO nRF52840 lacks an inbuilt EEPROM (Electrically Erasable Programmable Read-Only Memory). The nRF52840 SoC has 1 MB of inbuilt Flash memory, and the XIAO board has 2MB of onboard flash memory for storing data. The choice was based on the following benefits of adopting flash memory:

- Faster Read and Write Operations: Flash memory generally offers faster read and write speeds compared to EEPROM. [15] This will be advantageous in the iteration process when there is a need to access data quickly or update it frequently.
- Endurance and Lifetime: Flash memory typically has a higher endurance level than EEPROM. It can withstand a larger number of read and write cycles before it starts to degrade. [15] This endurance is particularly important when you need to update or modify the contents of the memory frequently.
- Cost and Space Efficiency: Since the nRF52840 chip already integrates Flash memory, using it eliminates the need for an additional EEPROM chip. This reduces the component count, simplifies the PCB layout, and potentially lowers the overall cost and space requirements of your design.

NanoBLEFlashPrefs is a Arduino library, which is a substitute for missing EEPROM storage on Arduino Nano 33 BLE and 33 BLE Sense (not for Nano 33 IoT or other Nano boards). [3] As the Seeduino XIAO nRF52840 Sense uses the same SoC chip as the Arduino Nano 33 BLE Sense, it can be easily used for the project. [1] [39] Unfortunately, the library requires Mbed OS; nonetheless, due to its design principles and development architecture, it is frequently regarded as easier to use than other Real-Time Operating Systems (RTOS). Furthermore, it is tightly integrated with the Arduino IDE, [40], which is utilized as the IDE of choice for writing code to the quadrotor. Because it supports filesystems and storage APIs, Mbed OS is necessary. Mbed OS features lightweight filesystems like FAT and LittleFS that make it easier to read, write, and manage data on flash memory. [5] These filesystems abstract away the complexity of direct flash memory access, allowing for more ordered data storage and retrieval. Furthermore, because Mbed is a simple RTOS, it can be easily adopted for Bluetooth logging, which could be necessary in the future. As it does not have memory constraints, but would be limited in processing power, allowing an additional thread to be used.

6.4 Wireless communication

When prototyping, quadrotors are able to easily disconnect their wired connection. Therefore if mid-air communication is required, it is necessary to have wireless communication.

Commonly used communication for drones are primarily various forms of 2.4 GHz, as it provides the best middle ground between reach and speed. The 2.4 GHz has wide range of technologies supporting it, for example Bluetooth, Wi-Fi and RF communication protocols. If a high performance drone was the goal, RF communication would be chosen, but for the purpose of this project Bluetooth or rather BLE is optimal, as the microchip has it inbuilt. BLE stands for Bluetooth low energy and is a separate technology of Bluetooth and requires different parts. BLE has lower power requirements, as it is primarly one way communication, where the packets are not being send constantly, but only when needed or requested.

The implementation of BLE for nRF52840 in Arduino IDE usually relies on either Bluefruit LE or Adafruit BLE library, depending on whether pure C or Mbed is used. For a device to use BLE, it needs to be either peripheral, which advertises itself or central, which connects to the advertised peripheral. Peripheral has services, for example control service, which can have characteristics like state with 0 and 1 for turning on and off. This characteristic can have different options. Write, where the central writes to this characteristic, read if central reads from it and notify which lets the central know, that it changed. Also a combination of these can define each characteristic.

In this case, the drone was chosen as a peripheral with characteristics mentioned above. To it, a controller with a joystick and two buttons connects as a central and writes to the characteristics. This was not used in the project, as the project did not reach stages requiring this. Instead, there is kill-switch. This was made for first free-flight testing. It contains only one characteristic, which switches between 0 and 1 no matter what it receives which starts the drone on 1 and stops on 0. It can be controlled from a mobile phone for the sake of reliability. The option of sending gyro-data from a phone to the drone to mimic a controller was also explored, but required packet management, as the data packets send were too long.

Last option which was explored was logging data through BLE. The challenging part is that BLE is aimed mainly at mobile device and proprietary applications. Therefore there are no available applications for Windows which allow reading data from BLE devices and mobile phone applications do not offer the option of saving larger amounts of data. As a consequence of mentioned, data-logging through BLE was abandoned.

6.5 Operation flow

Firstly, all variables used for PID, IMU, and flash memory are declared. If the drone is attached to a serial monitor, it will then enter debug mode. If the user enters the character “i” on the serial monitor while in debug mode, the quadrotor will display all of the flash memory data, including previously recorded data and the memory’s state. If the user enters the character "d", the memory is cleaned, as are the residual pointers that lead to the deleted memory (garbage collection). If the quadrotor was not previously attached to a serial monitor before entering debug mode, it would enter flying mode. To counteract the current spike, the motors initially ramp up. The quadrotor will thereafter enter the control loop as long as there was no BLE signal recieved. Additionally, if the value of "time" is less than the value of "set time." The IMU and IR measurements are recorded in the control loop before being sent to the complementary filter. The current Z, ϕ/roll, and θ/pitch are all sent to their individual PID controllers, where the error is determined using the previously defined PID values. The errors are transformed into a specific motion that should be accomplished using the motion equations (See Chapter 5.1) As the output is still an angular velocity, it gets converted to PWM and saturated to be between 0 and 255 (See Figure 5.2). The motors are then updated with their respective PWM value. A certain flash memory logging period is declared during initialization, and after updating the motors, a check is made to see if it is time to record the values, such as current z, ϕ/roll, and θ/pitch, for data analysis. (See chapter 6.3) If the "set time" is exceeded, the recorded values are saved in flash memory, the motors are completely switched off, and the code terminates execution.

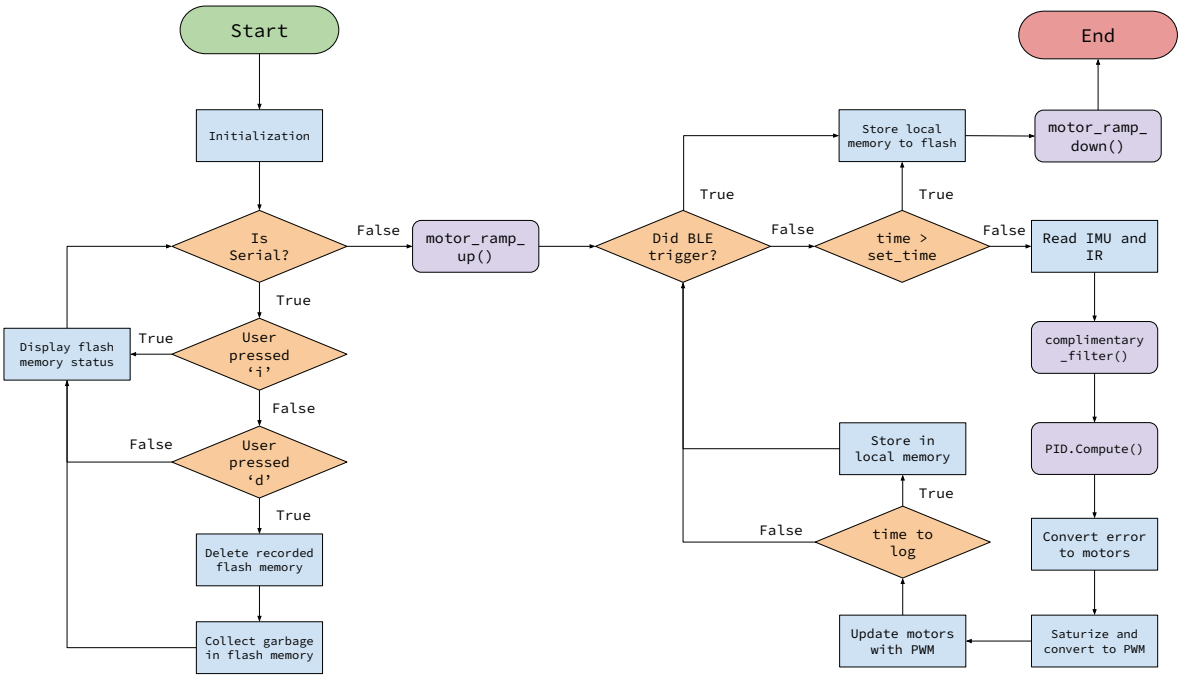


Figure 49: Flowchart describing the operation flow

References

[1] Arduino® nano 33 ble sense, product reference manual, sku: Abx00031. https://docs.arduino.cc/static/c78bcba85187bcd88d447f50050b4e89/ABX00031-datasheet.pdf.

[2] Autodesk what are decoupling capacitors? https://www.autodesk.com/products/fusion-360/blog/what-are-decoupling-capacitors/.

[3] Dirk-/nanobleflashprefs: Substitute for the missing eeprom storage on arduino nano 33 ble and ble sense. https://github.com/Dirk-/NanoBLEFlashPrefs/tree/main.

[4] Fairchild Semiconductor fdd8896 datasheet. https://pdf1.alldatasheet.com/datasheet-pdf/view/85382/FAIRCHILD/FDD8896.html.

[5] Filesystem - api references and tutorials | mbed os 6 documentation. https://os.mbed.com/docs/mbed-os/v6.16/apis/filesystem.html.

[6] Matweb fr4 material properties. https://www.matweb.com/search/DataSheet.aspx?MatGUID=3c5f252f235844fb9ffec6d1856ba0e3&ckck=1.

[7] Millenium Circuits Limited pcb trace width. https://www.mclpcb.com/blog/pcb-trace-width-vs-current-table/.

[8] Motorola 1n5818 datasheet. https://pdf1.alldatasheet.com/datasheet-pdf/view/2817/MOTOROLA/1N5818.html.

[9] Wikipedia motor constants. https://en.wikipedia.org/wiki/Motor_constants.

[10] Renat N. Abutalipov, Yuriy V. Bolgov, and Hamisha M. Senov. Flowering plants pollination robotic system for greenhouses by means of nano copter (drone aircraft). *2016 IEEE Conference on Quality Management, Transport and Information Security, Information Technologies, IT and MQ and IS 2016*, pages 7–9, 11 2016.

[11] European Union Aviation Safety Agency. Open category - civil drones | easa, 2023. https://www.easa.europa.eu/en/domains/civil-drones/drones-regulatory-framework-background/open-category-civil-drones.

[12] Tareq Aziz Hasan AL-Qutami and Rosdiazli Ibrahim. (pdf) development of an advanced controller for flow process applications using fuzzy logic. 1 2016.

[13] Cavoukian Ann. Privacy and drones: Unmanned aerial vehicles, 2012. http://www.ipc.on.ca/English/Resources/Discussion-Papers/.

[14] James Bennett, Ajay Bhasin, Jamila Grant, and Wen Chung Lim. Pid tuning via classical methods - engineering libretexts. https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Chemical_Process_

Dynamics_and_Controls_(Woolf)/09%3A_Proportional-Integral-Derivative_(PID)_Control/9.03%3A_PID_Tuning_via_Classical_Methods.

[15] Sandeep Bhandari. Difference between eeprom and flash. https://askanydifference.com/difference-between-eeprom-and-flash-with-table/?utm_content=cmp-true.

[16] Vadim A. Budnyaev, Ivan F. Filippov, Valeriy V. Vertegel, and Sergey Yu Dudnikov. Simulink-based quadcopter control system model. *2020 1st International Conference Problems of Informatics, Electronics, and Radio Engineering, PIERE 2020*, pages 246–250, 12 2020.

[17] Benjamin Cabé. Iot developer survey 2018, 2018. https://www.slideshare.net/kartben/iot-developer-survey-2018.

[18] Dlloydev. Dlloydev/quickpid: A fast pid controller with multiple options. various integral anti-windup, proportional, derivative and timer control modes. https://github.com/Dlloydev/QuickPID.

[19] Trafikstyrelsen Droner. Flying drones in denmark, 2022. https://www.droneregler.dk/english.

[20] Hafed Efheij, Abdulgani Albagul, and Nabela Ammar Albraiki. Comparison of model predictive control and pid controller in real time process control system. *19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering, STA 2019*, pages 64–69, 5 2019.

[21] MAIA ESTRERA. Who uses github?, 2 2022. https://careerkarma.com/blog/who-uses-github/.

[22] Muhammet Gul, Suleyman Mete, Faruk Serin, and Erkan Celik. Fine–kinney occupational risk assessment method and its extensions by fuzzy sets: A state-of-the-art review. *Studies in Fuzziness and Soft Computing*, 398:1–11, 2021. https://link.springer.com/chapter/10.1007/978-3-030-52148-6_1.

[23] Johannes Günther, Elias Reichensdörfer, Patrick M. Pilarski, and Klaus Diepold. Interpretable pid parameter tuning for control engineering using general dynamic neural networks: An extensive comparison. *PLoS ONE*, 15, 12 2020.

[24] M. Hassanalian and A. Abdelkefi. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*, 91:99–131, 5 2017.

[25] M.Sc. Graham Hunter. Pid control fundamentals - hunter consulting llc, 6 2022.

[26] Connected Industry. Freertos vs threadx vs zephyr: The fight for true open source rtos, 2023. https://

www.iiot-world.com/industrial-iot/connected-industry/freertos-vs-threadx-vs-zephyr-the-fight-for-true-open-source-rtos/.

[27] Joe Jaworski. A real-time operating system for the arduino - nuts and volts magazine. *Copyright © 2023 T and L Publications*, 2019. https://www.nutsvolts.com/magazine/article/a-real-time-operating-system-for-the-arduino.

[28] Michael A Johnson and Mohammad H Moradi. Pid control: New identification and design methods. *© Springer-Verlag London Limited 2005.*

[29] PlatformIO Labs. A professional collaborative platform for embedded development · platformio. https://platformio.org/.

[30] P. M. Meshram and Rohit G. Kanojiya. Tuning of pid controller using ziegler-nichols method for speed control of dc motor. pages 117–122, 2012. https://ieeexplore.ieee.org/abstract/document/6216102.

[31] micro ROS. Comparison between rtoses. https://micro.ros.org/docs/concepts/rtos/comparison/.

[32] Dennis Nash. How to access data for optimal controller tuning | rockwell automation. https://www.rockwellautomation.com/en-us/company/news/the-journal/access-live-historical-data-for-optimal-PID-controller-tuning.html.

[33] Chris Reddington Clay Nelson. How github accelerates development for embedded systems | the github blog. https://github.blog/2023-03-09-how-github-accelerates-development-for-embedded-systems/.

[34] Vlad Niculescu, Lorenzo Lamberti, Francesco Conti, Luca Benini, and Daniele Palossi. Improving autonomous nano-drones performance via automated end-to-end optimization and deployment of dnns. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 11:548–562, 12 2021.

[35] Ekaterina Novoseltseva. Top 7 benefits you get by using github | apiumhub. https://apiumhub.com/tech-blog-barcelona/using-github/.

[36] Zephyr Project. Introduction — zephyr project documentation, 2022. https://docs.zephyrproject.org/latest/introduction/index.html.

[37] Zephyr Project. Zephyr project github repository, 2023. https://github.com/zephyrproject-rtos/zephyr/.

[38] Nordic Semiconductor. nrf connect for vs code - nrf connect for vs code, 2022. https://nrfconnect.github.io//vscode-nrf-connect/index.html.

[39] Seeed Studio. Getting started with seeed studio xiao nrf52840 (sense) | seeed studio wiki, 2023. https://wiki.seeedstudio.com/XIAO_BLE/.

[40] Karl Söderby. Installing mbed os nano boards | arduino documentation, 5 2023. https://docs.arduino.cc/software/ide-v1/tutorials/getting-started/cores/arduino-mbed_nano.

[41] Jasper J. van Beers and Coen C. de Visser. Peaking into the black-box: Prediction intervals give insight into data-driven quadrotor model reliability. 1 2023.

[42] Kashyap Vyas. A brief history of drones: The remote controlled unmanned aerial vehicles (uavs), 2020. https://interestingengineering.com/innovation/a-brief-history-of-drones-the-remote-controlled-unmanned-aerial-vehicles-uavs.

[43] Oleksandr Zozulia. Freertos vs. zephyr project for embedded iot projects - lemberg solutions, 2022. https://lembergsolutions.com/blog/freertos-vs-zephyr-project-embedded-iot-projects.

A Motor Thrust Testing

The 4th semester project in mechatronics is based on building a VTOL drone. For the drone to take flight, sufficient thrust provided by the propellers attached to the motors of the drone is required. Thus, the following is the report of thrust testing using different motors and propellers. Initial testing setup

A.1 Initial Assumptions

The initial idea to test the thrust provided by a certain motor was to create a x-shaped frame where, the motors were connected to their respective corners and a bolt was tightened in the centre as a means of weight. The frame with the motors were then placed on a scale where the motors were facing in the direction of the ceiling, thus meaning that propellers were pushing the air downwards and the setup upwards. The initial weight of the frame was then noted, where after powering up the motors, the weight of the setup would decrease due to the thrust.

A.2 Initial Results

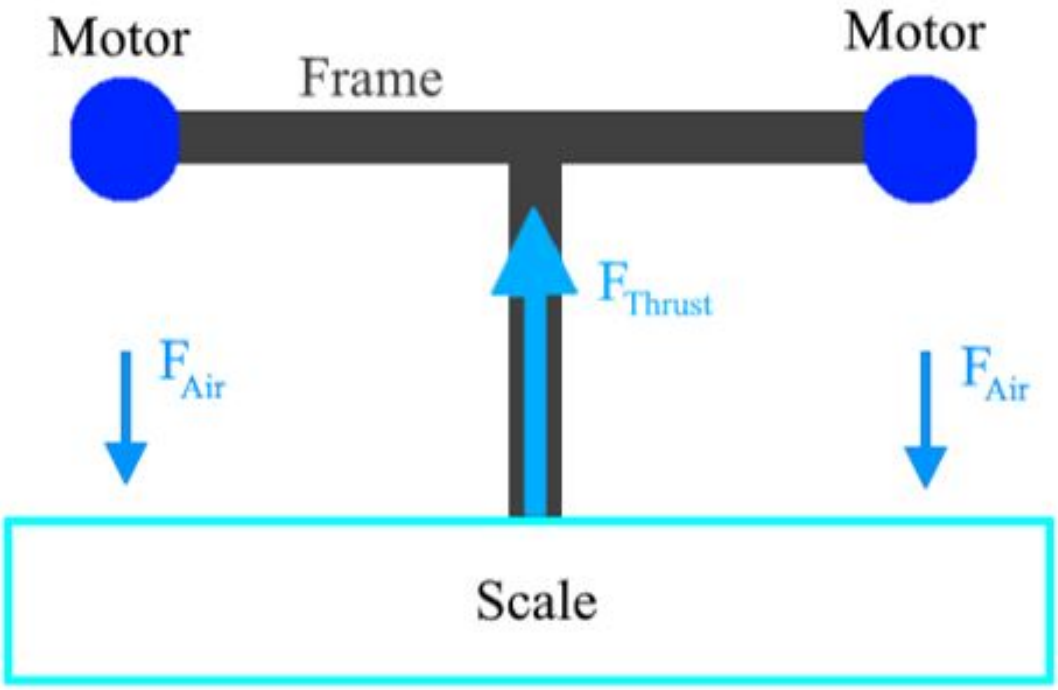
The setup of the bolt supporting the frame on the scale was unstable, and because of that the wiring of the motors had also interfered with the weight. Additionally, as the propellers were pushing the frame with a certain thrust force, the air that was being pushed down also had the same force. Therefore, as the frame was very close to the scale, the force by which the setup was being lifted by was also being pushed down by the air on the scale, thus making almost no changes on the scale readings.

A.2.1 Final Assumptions

To fix the problems of the initial test setup, a second version of the x-shaped frame was made where the corners were connected to 3d printed pillars, to make the system more stable. Additionally, instead of placing all the motors at each corner and measuring thrust, only one motor would be placed in the centre instead, as the other motors were identical and thus would produce close to identical thrust. Moreover, the motors would now be placed in a direction opposite to the ceiling as then the air would not be pushed on the scale, but the frame would be. Thus, the additional weight that occurs after the motors are powered on would be equal to the thrust provided by the motor and propeller.

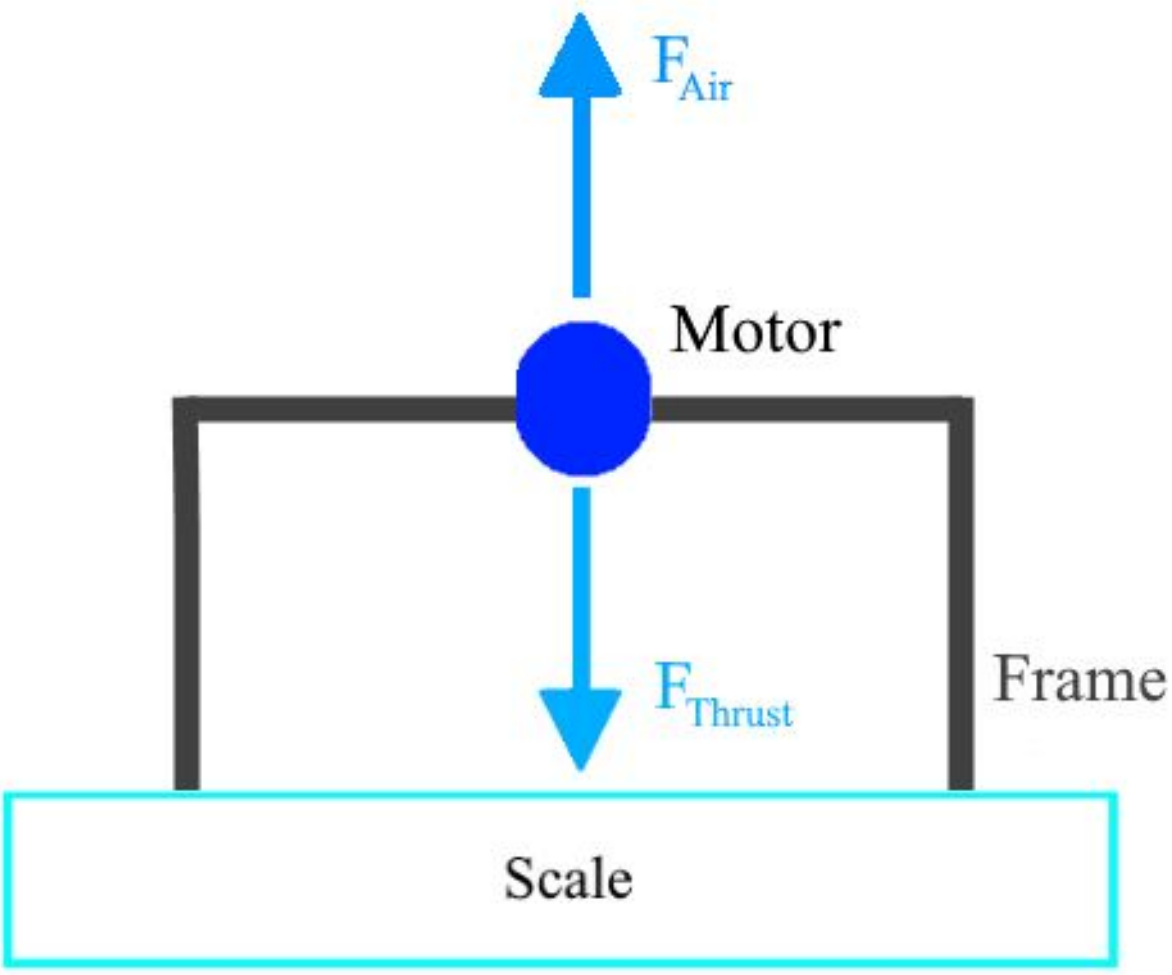
A.2.2 Final Results

The motor used had an operating voltage window of 0-4,2 volts, therefore the motor was tested between 1-4,2 volts incremented with 0,1 volts. Characterizing the motor yielded the following results:



$$4 \cdot F_{Thrust} \approx 4 \cdot F_{air}$$

Figure 50: Initial Thrust Setup



$$F_{Thrust} \approx F_{air}$$

Figure 51: Final Thrust Setup

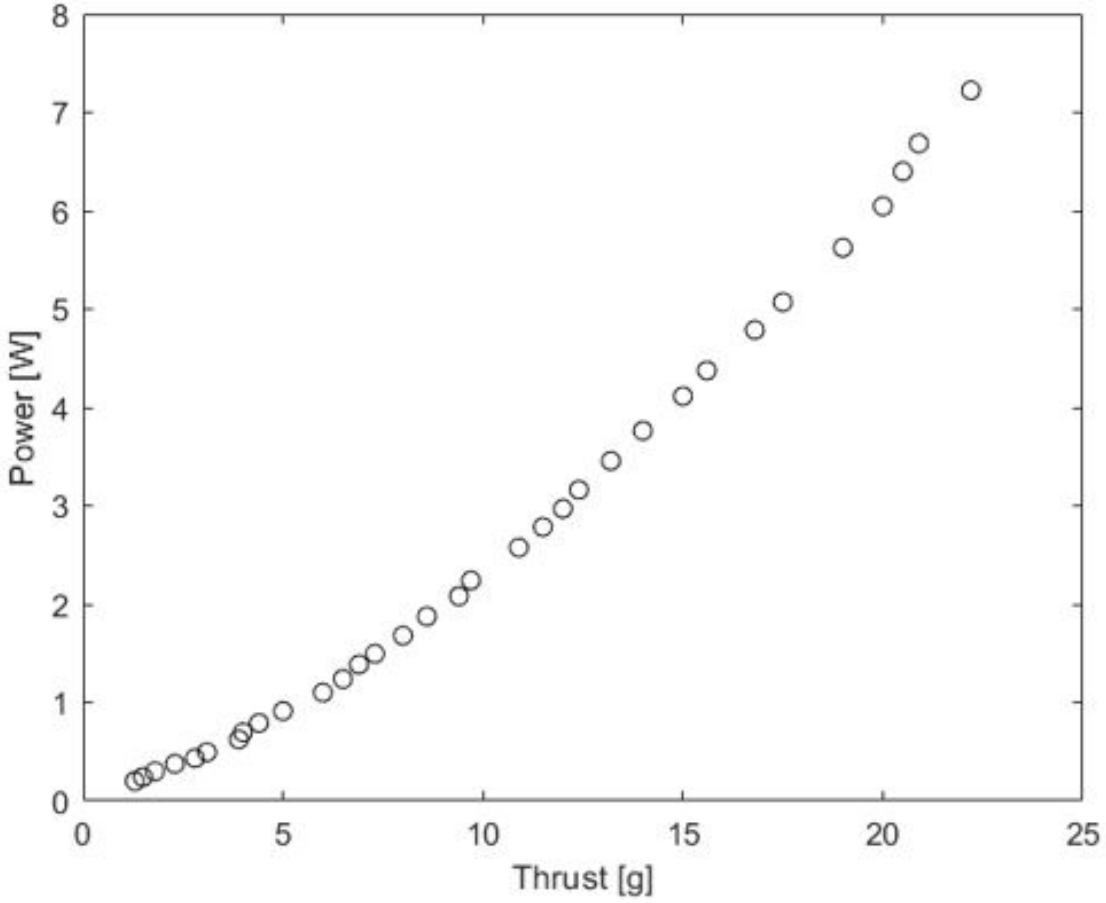


Figure 52: Motor Thrust Graphed

It became clear that the total thrust that could be produced by a single motor was 17.5 grams at 3.7 Volts (The minimum level of the battery). Subtracting the weight of the motor itself, the motor can produce a net thrust of 12.4 grams, and since the drone would be flown in a quad configuration, the total net thrust possible for the four motors were 49.6 grams at lowest voltage. This then set the limits of how heavy the drone could be, and since a general rule of thumb is that the motors need to produce double the needed thrust to lift the drone , it's clear that the remaining components of the drone can weigh 24,8 grams for it to handle optimally. At maximum voltage level of the battery, a motor produces 22.2 grams of thrust, so an excess of 17,1 grams subtracting the motors weight itself. So at maximum voltage level, the drones remaining components can weigh 34.2 grams.

B Motor Testing

For designing the controller, it is important to have accurate values and constants for the motors used in the drone if you want to accurately determine and calculate its position in the physical space. The motors used for the project did not have a data sheet available with motor constants/characteristics, so testing would have to be done to determine these values. motor constants relevant for this project include, "Motor velocity constant" (K_v) and "Motor torque constant" (K_T)

B.1 Test Setup

For the test setup the following items were used:

- A photo sensor (to measure RPM)
- Marked Propeller
- One coreless dc motor of unknown origin.
- A vise
- AFD-8896 Mosfet
- Seeed xiao ble nrf52840 sense
- Power supply

The photo sensor works by measuring a change in light reflection, so a white propeller was marked with black touché to create an RPM measuring tool. The motor with the propeller is mounted to a table with a vise to reduce vibration. To test using PWM, the motor is turned on through the Seeed, so the rpm can be measured and changed. All the values are recorded 3 times for each PWM to increase consistency in the measurements. The motor can handle inputs at both 3.7V and 4.2V, so the test was done with both voltages, since the battery output will be somewhere in between depending on the discharge. The voltage the motor draws is measured through a multimeter.

The Seeed is used to control the PWM but cannot alone supply enough current to power the motor, therefore the power is provided directly through the power supply and controlled through a mosfet. At lower values of PWM however, the gate on the mosfet will close so the motor will not turn on.

B.2 Theory

Motor velocity- or motor speed constant is measured in RPM per volt and can be calculated by taking the motors unloaded rotational speed divided by the peak voltage. It was

necessary to load the motor slightly, to measure using a photosensor. The small load is therefore deemed negligible.

The motor speed constant can then be used to calculate the motor torque constant, which is measured in newton meter per ampere. This constant will be important for the model to determine the change in thrust needed to achieve the desired outcome. [9]

$$K_{\mathrm{v}} = \frac{\omega_{\text{no-load}}}{V_{\text{peak}}} \qquad K_{\mathrm{T}} = \frac{\tau}{I_{\mathrm{a}}} = \frac{60}{2\pi K_{\mathrm{v(RPM)}}}$$

Figure 53: Motor Constant Equations

The test was also done to determine the thrust coefficient of the motor. Using the thrust testing from before, the thrust can be determined based on the voltage drop. Then, by converting the RPM to angular velocity, it is possible to derive the thrust coefficient by dividing the thrust with the angular velocity. For both the motor constants and values used to calculate the constants are from tests with 3.0V inputs, which is the voltage needed to achieve the theoretical hover thrust. This is sensible since most of the time the drone will be operating around these values.

B.3 Data

Some data could not be gathered from the setup since the motor was outputting too high an RPM to be measured with the equipment. And the lower PWM missing values is because of the mosfet closing.

PWM (%)	PWM (8bit)	RPM							Angular velocity (rad/s)	Measured voltage (V) (3.7V)	Measured voltage (V) (4.2V)	Thrust (N)	Thrust coefficient (N/s)
		Battery voltage - 3.7 V			Battery voltage - 4.2 V			Average					
		1	2	3	1	2	3						
100%	255	30500	30600	30600	n/a	n/a	n/a	n/a	#VALUE!	n/a	3.95	#######	#VALUE!
95%	242	30300	30400	30500	33400	33000	33400	31833	3333.58	3.5	3.92	0.16	4.77E-05
90%	230	29000	29200	29200	32100	32000	32000	30583	3202.68	3.3	3.71	0.14	4.42E-05
85%	217	25300	25200	25400	27400	27500	27500	26383	2762.86	2.75	3.07	0.10	3.59E-05
80%	204	22400	22600	22500	24400	24500	24500	23483	2459.17	2.4	2.63	0.08	3.09E-05
75%	191	21700	21700	21700	24000	23900	23900	22817	2389.36	2.3	2.56	0.07	2.93E-05
70%	179	21400	21500	21500	23800	23800	23800	22633	2370.16	2.26	2.55	0.07	2.85E-05
65%	166	21400	21400	21400	23700	23800	23800	22583	2364.92	2.25	2.54	0.07	2.83E-05
60%	153	21300	21300	21300	23600	23700	23700	22483	2354.45	2.22	2.53	0.07	2.77E-05
55%	140	20700	20800	20900	23000	23100	23100	21933	2296.85	2.17	2.45	0.06	2.72E-05
50%	128	20000	20000	20000	21700	21800	21900	20900	2188.64	2.05	2.28	0.06	2.55E-05
45%	115	18000	18000	18100	19300	19400	19300	18683	1956.51	1.83	1.98	0.04	2.28E-05
40%	102	15400	15400	15400	16100	16100	16100	15750	1649.34	1.52	1.59	0.03	1.88E-05
35%	89	13000	13100	13100	14000	14000	14000	13533	1417.21	1.25	1.38	0.02	1.50E-05
30%	76	11400	11500	11500	12800	12800	12800	12133	1270.60	1.06	1.24	0.02	1.21E-05
25%	64	10300	10300	10300	12000	11900	11900	11117	1164.13	0.96	1.14	0.01	1.09E-05
20%	51	9000	9100	9100	10700	10700	10800	9900	1036.73	0.84	1.01	0.01	9.39E-06
15%	38	7300	7200	7300	8500	8600	8500	7900	827.29	0.66	0.78	0.01	7.34E-06
10%	25	2000	2100	2000	2700	2700	2400	2317	242.60	0.19	0.24	0.00	2.19E-06
5%	13	n/a	n/a	n/a	n/a	n/a	n/a	n/a	#VALUE!	n/a	n/a	#######	#VALUE!
0%	0	n/a	n/a	n/a	n/a	n/a	n/a	n/a	#VALUE!	n/a	n/a	#######	#VALUE!

Figure 54: Testing Data with Thrust Coefficient highlighted in green

B.4 Conclusion of Theory

$$K_v := \frac{\omega}{V} = 8943.333333 \frac{RPM}{V}$$

$$K_\tau := \frac{60}{2 \cdot \pi \cdot K_v} = 0.001067755861 \frac{N \cdot m}{A}$$

Figure 55: Motor Constants

C Budget

In order to keep track of the used money in the project, a budget was created for the project. The budget is sepperated in testing phase and final product cost, in order to track the budget against the goals set in the project. In cases there was ordered spares or replacements hence the quantity difference in the final product cost and testing phase.

Testing Phase:

ITEM	QTY	UNIT COST	TOTAL COST	SOURCE
Seeed Studio XIAO Sense	2	109,20 kr.	218,40 kr.	XIAO Sense
4* 8520 Coreless DC-Motor	2	152,00 kr.	304,00 kr.	8520 DC-motor
1N5818 Diode	1	2,12 kr.	2,12 kr.	1N5818 Diode
FDD8896 MOSFET	4	7,25 kr.	29,00 kr.	FDD8896 MOSFET
Hubsan propeller set	1	36,00 kr.	36,00 kr.	Hubsan propeller
Turnigy 1S Battery	1	39,20 kr.	39,20 kr.	TURNIGY 1S Battery
Ceramic Capacitor 100 nF	5	0,72 kr.	3,61 kr.	Capacitor 100 nF
GY-VL53L0XV2 ToF Module	1	79,00 kr.	79,00 kr.	ToF Sensor
PLA [in grams]	5	0,13 kr.	0,64 kr.	PLA, Red
TOTAL:			632,33 kr.	

Final Product Cost:

ITEM	QTY	UNIT COST	TOTAL COST	SOURCE
Seeed Studio XIAO Sense	1	109,20 kr.	109,20 kr.	XIAO Sense
4* 8520 Coreless DC-Motor	1	152,00 kr.	152,00 kr.	8520 DC-motor
1N5818 Diode	1	2,12 kr.	2,12 kr.	1N5818 Diode
FDD8896 MOSFET	4	7,25 kr.	29,00 kr.	FDD8896 MOSFET
Hubsan propeller set	1	36,00 kr.	36,00 kr.	Hubsan propeller
Turnigy 1S Battery	1	39,20 kr.	39,20 kr.	TURNIGY 1S Battery
Ceramic Capacitor 100 nF	5	0,72 kr.	3,61 kr.	Capacitor 100 nF
GY-VL53L0XV2 ToF Module	1	79,00 kr.	79,00 kr.	ToF Sensor
PLA [in grams]	5	0,13 kr.	0,64 kr.	PLA, Red
TOTAL:			371,13 kr.	

ITEMS MARKED WITH ORANGE, WERE NOT PURCHASED (PERSONAL ITEMS)
ALL PRICES EXCLUDING VAT

Figure 56: Budget for the semester project

D Setting up the embedded environment for software development

PlatformIO is an open-source extension of Visual studio code, which supports hundreds of boards with different frameworks. [29] Seeed XIAO nRF52840 is not natively suported. Seeed made library for Arduino IDE, which supports Arduino and Mbed framework, that was implemented through Github to PlatformIO and allows non-problematic implementation. The XIAO has library in the Zephyr Github repository, but the PlatformIO does not use most recent version. The version of Zephyr itself is altered in PlatformIO and therefore does not work even after cross referencing with other boards already implemented in PlatformIO, such as Seeeduino (as a reference for the formfactor) or nRF52840 DK (as a reference for the chip). The PlatformIO does not have inbuild serial USB communication, which results in inability to reprogram the board through the VS Code and need to go through bootloader first. This poses inconvinience, which can be avoided by use of a Arduino IDE, therefore if route of MBed or Arduino framework is chosen, Arduino IDE should be chosen. If the route of Zephyr is chosen, SDK by nordic semi is the optimal route, because event though it also does not have serial USB, it uses barebone Zephyr, as the Nordic Semiconductor SDK provides useful tools for development, such as build, flashing and debugging actions. [38]

There is a caviat, profiles from GitHub of the current version of Zephyr via `Zephyr/boards/arm/xiao_ble` at main branch [37] can be used and imported the needed files into the SDK version of Zephyr we have. In our case the path of the files would be in `~\ncs\v2.3.0-rc1\zephyr\boards\arm` . After doing so, following the steps in the DevAcademy, nRF Connect SDK Fundamentals, Lesson 1 can be followed for setup. Then a blinky application can be created, and built via nRF Connect. After a build of the application is created, a files for flashing will be created in the "build" subdirectory of the application. To flash the Seeed XIAO nRF52840 Sense chip, entering a bootloader is needed, as it ships with the Adafruit nRF52 Bootloader, which supports UF2 flashing. [36] Further more, a zephyr.ut2 file can be found in the "build" subdirectory of the application, after building the application, as we have done. To access the bootloader, connect the Seeed XIAO to a PC. Now to enter and flash an application, the reset button on the Seeed board should be clicked two times in quick succession, this will propmt the memory of the Seeed board on your PC. Now find the previously mentioned zephyr.ut2 and drag it in to the memory of the Seeed board, and this will flash the board with the new application.

A issue accured when trying to test the "Hello World" application, when connecting to serial monitor no output is given. After further investigation, it was noticed that the Seeed documentation, mentiones no debugging interface, hence no USB serial exists. To solve the issue, an application called "console", from `zephyr/samples/subsys/usb/console`

was cloned, in order to test a virtual USB serial connection, with the use of CDC ACM UART. [37] After building and flashing, results were achieved. (See Figure 57)

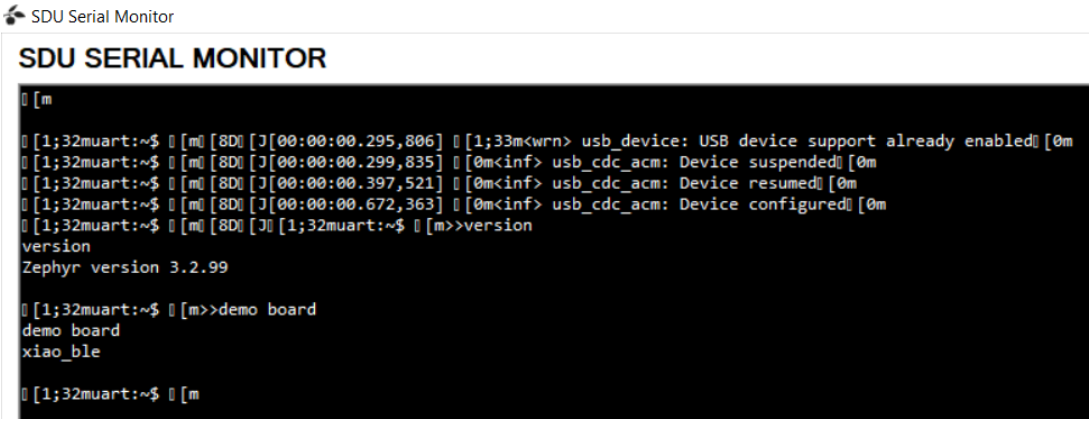


Figure 57: Serial output of the XIAO MCU

Another issue that arose was debugging. The Seeed XIAO nRF52840 Sense does not have any sort of built in debugging tools. [39] One can use a J-link debugging tool, but a caviet was found, which was GDB stub, which Zephyr supports, this would save budget. [36]

E Plant Appendix

Symbol	Description	Unit	Observability
θ	Pitch Euler Angle	rad	Stereo Vision
$\dot{\theta}$	Pitch Euler Angular Velocity	rad/s	Gyroscope
$\ddot{\theta}$	Pitch Euler Angular Accel.	rad/s^2	-
ϕ	Roll Euler Angle	rad	Stereo Vision
$\dot{\phi}$	Roll Euler Angular Velocity	rad/s	Gyroscope
$\ddot{\phi}$	Roll Euler Angular Accel.	rad/s^2	-
ψ	Yaw Euler Angle	rad	Stereo Vision
$\dot{\psi}$	Yaw Euler Angular Velocity	rad/s	Gyroscope
$\ddot{\psi}$	Yaw Euler Angular Accel.	rad/s^2	-
X	Position in X	m	Stereo Vision
$\dot{X}$	Velocity in X	m/s	-
$\ddot{X}$	Accel. in X	m/s^2	Accelerometer
Y	Position in Y	m	Stereo Vision
$\dot{Y}$	Velocity in Y	m/s	-
$\ddot{Y}$	Accel. in Y	m/s^2	Accelerometer
Z	Position in Z	m	Stereo Vision
$\dot{Z}$	Velocity in Z	m/s	-
$\ddot{Z}$	Accel. in Z	m/s^2	Accelerometer
Ω_1	Motor 1 Angular Velocity	rad/s	Voltage/Current/Optical
Ω_2	Motor 2 Angular Velocity	rad/s	Voltage/Current/Optical
Ω_3	Motor 3 Angular Velocity	rad/s	Voltage/Current/Optical
Ω_4	Motor 4 Angular Velocity	rad/s	Voltage/Current/Optical
θ-Theta (/they-tuh/), ϕ-Phi (/fi/), ψ-Psi (/sai/), Ω-Omega (/oh-mega/)			

Figure 58: State table [16]

State/Input	Dependencies	Equation
θ	$\dot{\theta}$	$\theta = \int \dot{\theta}$
$\dot{\theta}$	$\ddot{\theta}$	$\dot{\theta} = \int \ddot{\theta}$
$\ddot{\theta}$	$\dot{\phi}, \dot{\psi}, \Omega_r, U3$	$\ddot{\theta} = \frac{\dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) - J_r\dot{\phi}\Omega_r + l(U3)}{I_{yy}}$
ϕ	$\dot{\phi}$	$\phi = \int \dot{\phi}$
$\dot{\phi}$	$\ddot{\phi}$	$\dot{\phi} = \int \ddot{\phi}$
$\ddot{\phi}$	$\dot{\theta}, \dot{\psi}, \Omega_r, U2$	$\ddot{\phi} = \frac{\dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) + J_r\dot{\theta}\Omega_r + l(U2)}{I_{xx}}$
ψ	$\dot{\psi}$	$\psi = \int \dot{\psi}$
$\dot{\psi}$	$\ddot{\psi}$	$\dot{\psi} = \int \ddot{\psi}$
$\ddot{\psi}$	$\dot{\theta}, \dot{\phi}, U4$	$\ddot{\psi} = \frac{\dot{\theta}\dot{\phi}(I_{xx} - I_{yy}) + (U4)}{I_{zz}}$
X	$\dot{X}$	$x = \int \dot{x}$
$\dot{X}$	$\ddot{X}$	$\dot{x} = \int \ddot{x}$
$\ddot{X}$	$\psi, \phi, \theta, U1$	$\ddot{X} = \frac{(\sin\psi \sin\phi - \cos\psi \sin\theta \cos\phi)U1 - A_x\dot{X}}{m}$
Y	$\dot{Y}$	$Y = \int \dot{Y}$
$\dot{Y}$	$\ddot{Y}$	$\dot{Y} = \int \ddot{Y}$
$\ddot{Y}$	$\psi, \phi, \theta, U1$	$\ddot{Y} = \frac{(\cos\psi \sin\phi + \sin\psi \sin\theta \cos\phi)U1 - A_y\dot{Y}}{m}$
Z	$\dot{Z}$	$Z = \int \dot{Z}$
$\dot{Z}$	$\ddot{Z}$	$\dot{Z} = \int \ddot{Z}$
$\ddot{Z}$	$\psi, \phi, U1$	$\ddot{Z} = \frac{mg - (\cos\theta \cos\phi)U1 - A_z\dot{Z}}{m}$
Ω_r	$\Omega_1, \Omega_2, \Omega_3, \Omega_4$	$\Omega_r = \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4$
$U1_x$	$\Omega_1, \Omega_2, \Omega_3, \Omega_4$	$U1_x = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)$
$U2_x$	Ω_2, Ω_4	$U2_x = b \sin\left(\frac{pi}{4}\right)(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2)$
$U3_x$	Ω_1, Ω_3	$U3_x = b \sin\left(\frac{pi}{4}\right)(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2)$
$U4_x$	$\Omega_1, \Omega_2, \Omega_3, \Omega_4$	$U4_x = d(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2)$

Figure 59: Summary of plant equations [16]