

UNIVERSITY OF SOUTHERN DENMARK

FACULTY OF ENGINEERING

BEng Mechatronics

Mechatronics Semester Project 4

Ladybug

Supervisor:

Lars Duggen

Davi Goncalves

Accioli

Group: 7

Tobias Blaabjerg

Karlsen

Artis Fils

Thor Uerkvitz

Choyon Mainul Hasan

Johan Paul

Ruta Miglava

Year 2023

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. [23] 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 [10], 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 [6]. 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 [11]. 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 [5].

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 [5]. 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		
	Subcategory	Operational restrictions	Drone operator registration	Remote pilot competence	Remote pilot minimum age
< 250 g	A1 (can also fly in subcategory A3)	<ul style="list-style-type: none"> — 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	<ul style="list-style-type: none"> — 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 [5]

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. [17]

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. [4]

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/s^2 .
- 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 through 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.

4 Design and manufacturing of quadrotor

5 Control prototyping

6 Control

6.1 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.

6.1.1 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. [7]

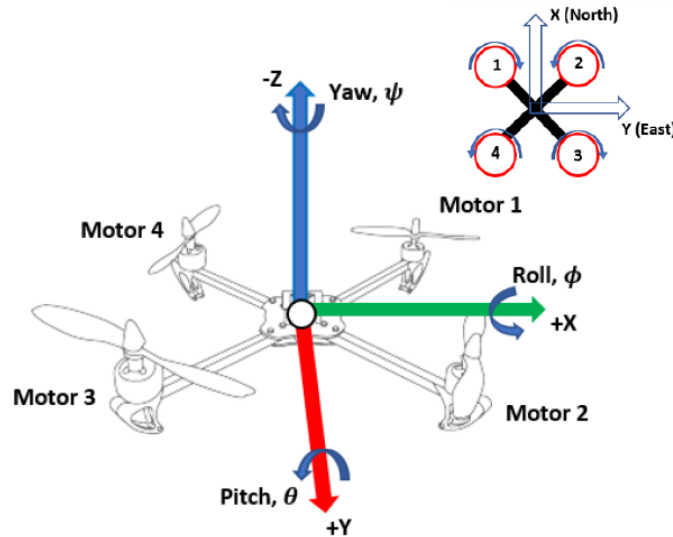


Figure 2: Quadcopter 'x' configuration

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.

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 3: State table

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: [7]

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 4: State table

6.1.2 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. [7]

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	$I_{motor,x} = 2 \left(\frac{1}{12} m(3r^2 + h^2) \right) + 2 \left(\frac{1}{12} m(3r^2 + h^2) + md_{a2a}^2 \right)$ $I_{motor,y} = 2 \left(\frac{1}{12} m(3r^2 + h^2) \right) + 2 \left(\frac{1}{12} m(3r^2 + h^2) + md_{a2a}^2 \right)$ $I_{motor,z} = 4 \left(\frac{1}{2} mr^2 + md_{a2a}^2 \right)$
Arms MoI	$I_{Arm,x} = 2 \left(\frac{1}{12} m(w^2 + h^2) \right) + 2 \left(\frac{1}{12} m(h^2 + d^2) + md_{a2a}^2 \right)$ $I_{Arm,y} = 2 \left(\frac{1}{12} m(w^2 + h^2) \right) + 2 \left(\frac{1}{12} m(h^2 + d^2) + md_{a2a}^2 \right)$ $I_{Arm,z} = 4 \left(\frac{1}{12} m(w^2 + d^2) + md_{a2a}^2 \right)$
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 5: Inertia moments equations

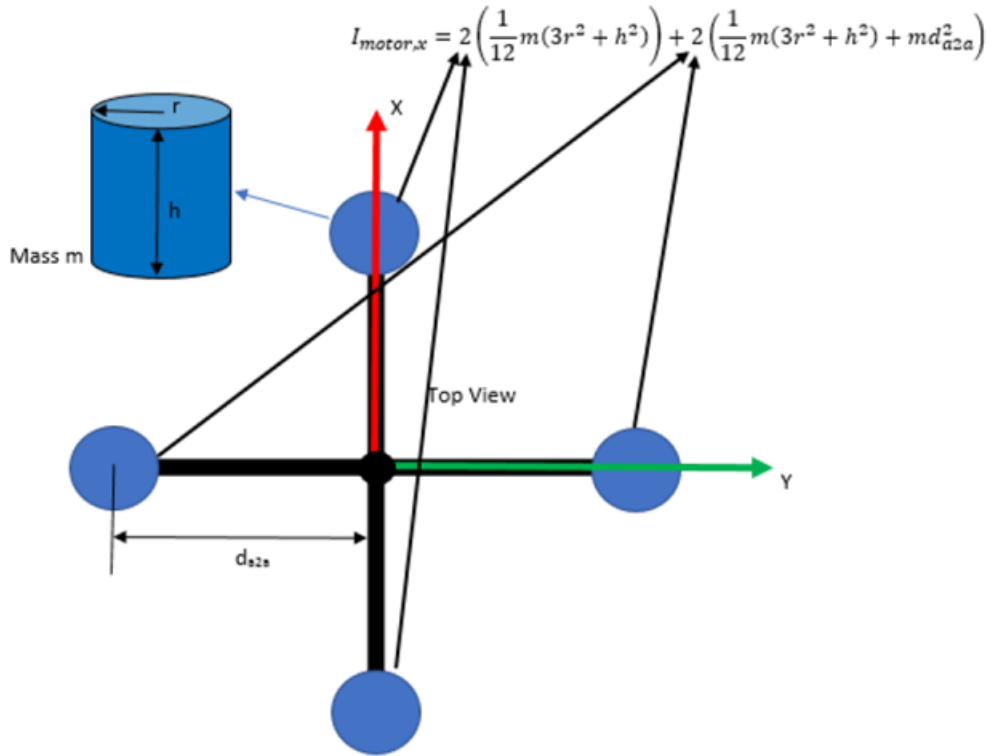


Figure 6: Top view of quadcopter

A mathematical document can be found in the appendix as “SPRO4math”. Which shows the detailed process of calculation. The mathematical equations are set up in maple, allowing quick changes to values in case a part is changed or removed. This feature proved useful since the weight estimate of the drone has changed rapidly over the course of the project period, as well as new components being added, which could then easily be integrated in the calculations.

$$I_{rotor} := 729$$

$$I_x := 48932.24391$$

$$I_y := 49929.46266$$

$$I_z := 99299.32284$$

Figure 7: 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

6.1.3 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 Report"
- Motor constants testing can be found in appendix "Motor Testing"

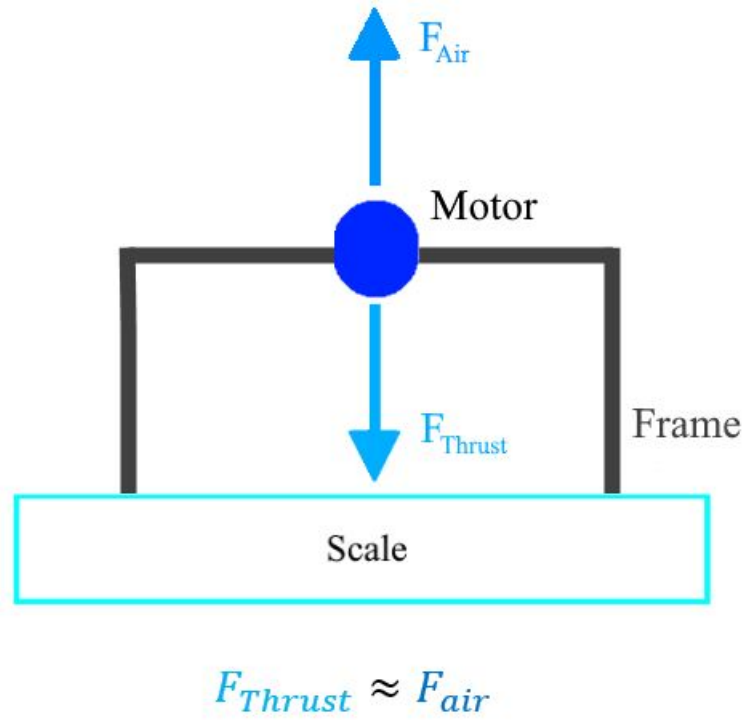


Figure 8: Thrust testing setup

6.1.4 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. 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.

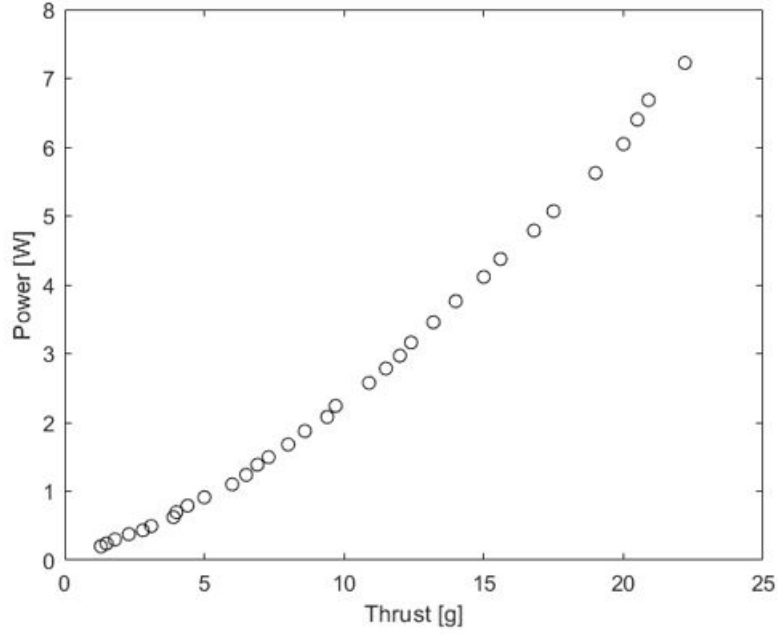


Figure 9: Thrust test data graphed

6.1.5 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. [3]

$$K_v = \frac{\omega_{\text{no-load}}}{V_{\text{peak}}} \quad K_T = \frac{\tau}{I_a} = \frac{60}{2\pi K_{v(\text{RPM})}}$$

Figure 10: 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 11: 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.

6.1.6 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.

6.2 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.

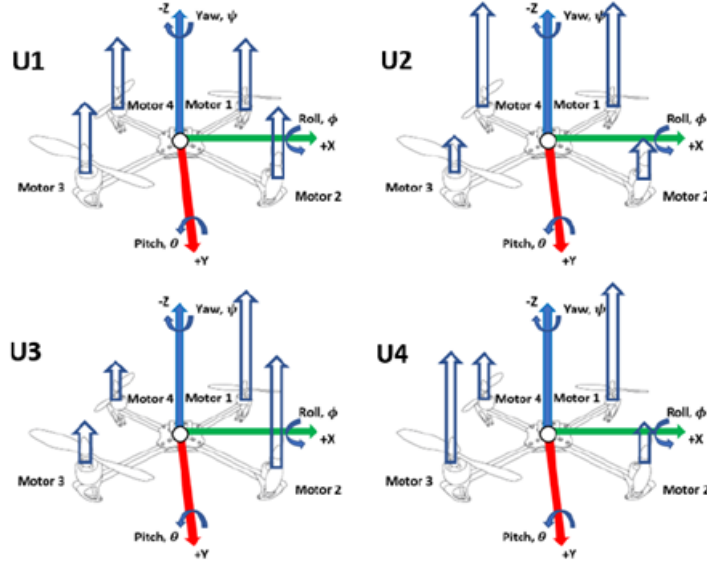


Figure 12: Visual representation of thrust equations [7]

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. [7]

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)
$U2_x$	$b \sin\left(\frac{p^l}{4}\right) (\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2)$	Roll Thrust	(2.9)
$U3_x$	$b \sin\left(\frac{p^l}{4}\right) (\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2)$	Pitch Thrust	(2.10)
$U4_x$	$d(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2)$	Yaw Thrust	(2.11)

Figure 13: Thrust equations [7]

The plant equations for this project were gotten from one of the sources. For a more in depth description of how the plant equations are found see [7]

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	$U2_x = b \sin\left(\frac{\pi i}{4}\right)(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2)$
$U3_x$	Ω_1, Ω_3	$U3_x = b \sin\left(\frac{\pi i}{4}\right)(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2)$
$U4_x$	$\Omega_1, \Omega_2, \Omega_3, \Omega_4$	$U4_x = d(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2)$

Figure 14: Summary of plant equations [7]

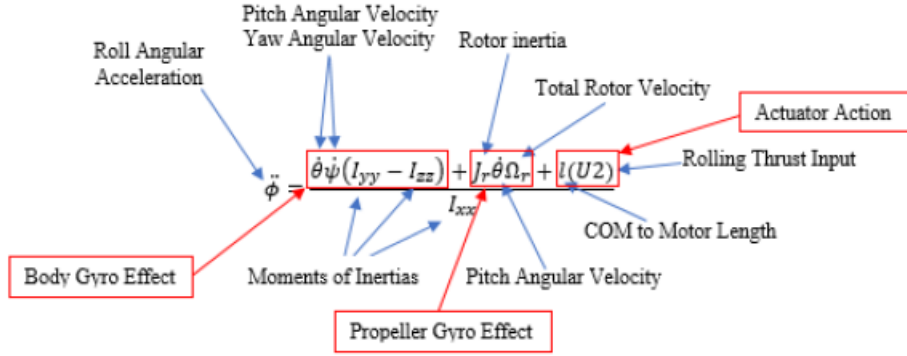


Figure 2.14-Rolling acceleration equation explained

Figure 15: Example of plant equation for phi. [7]

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.

6.3 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.

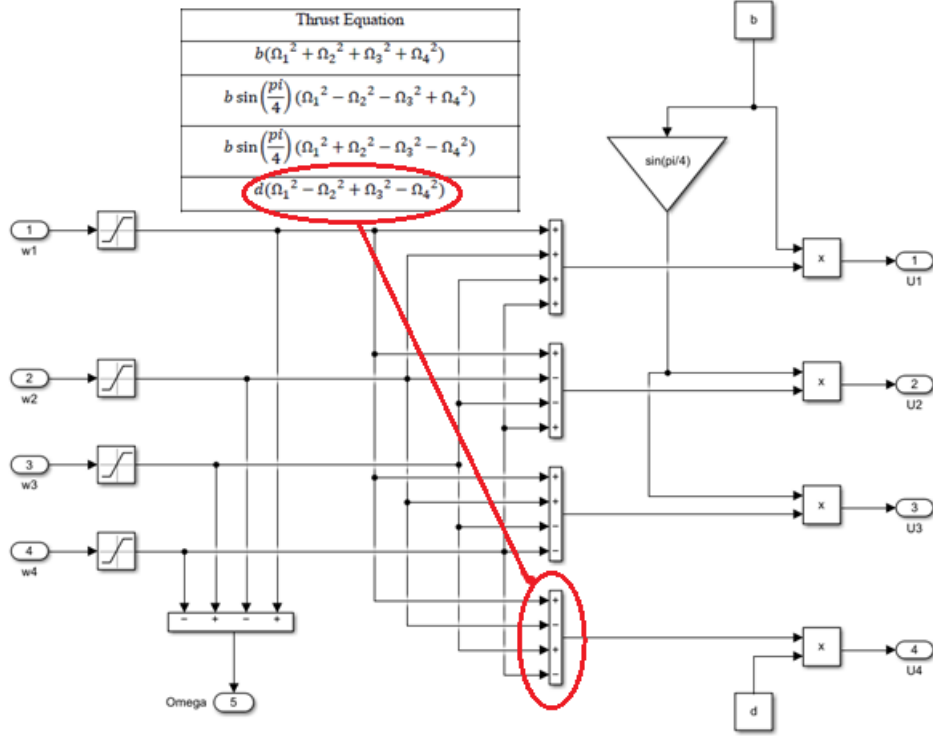


Figure 16: Thrust equations in SIMULINK

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 sensors 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 sensor 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.

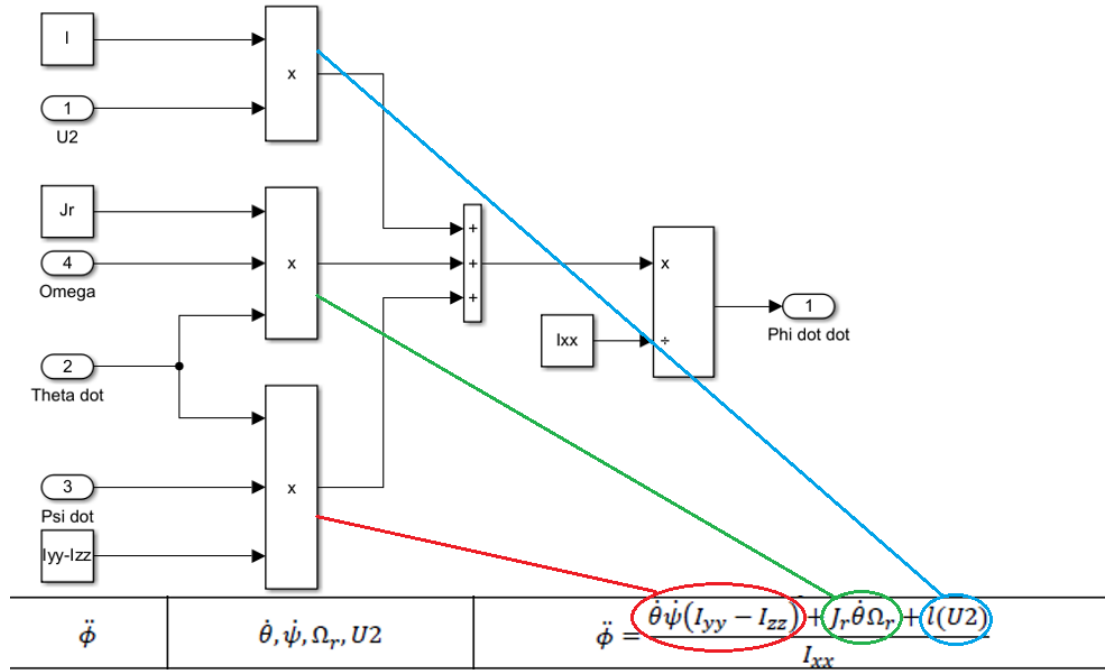


Figure 17: Plant equation for phi double dot

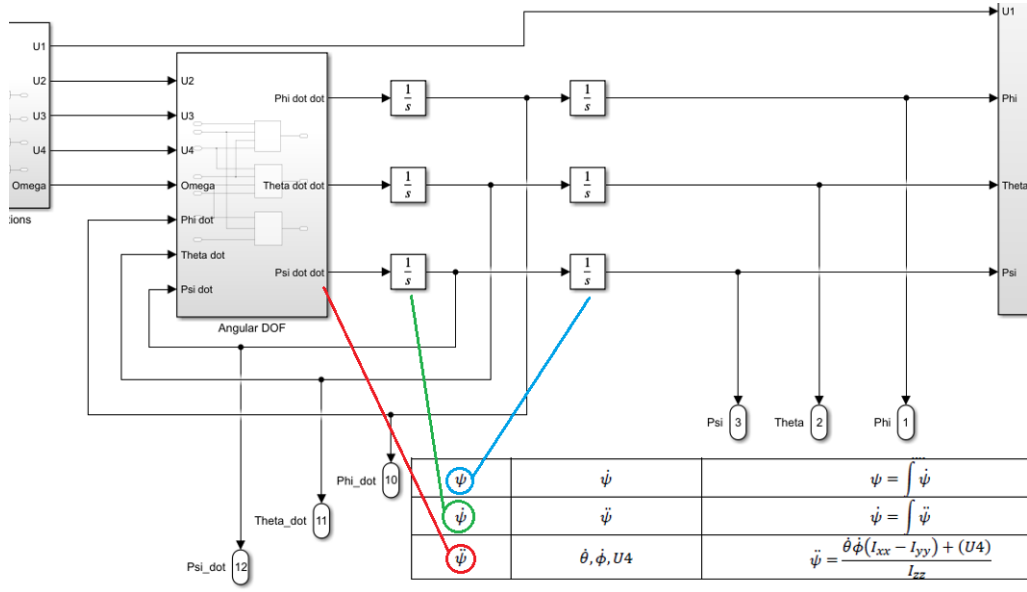


Figure 18: Angular equations overview example

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.

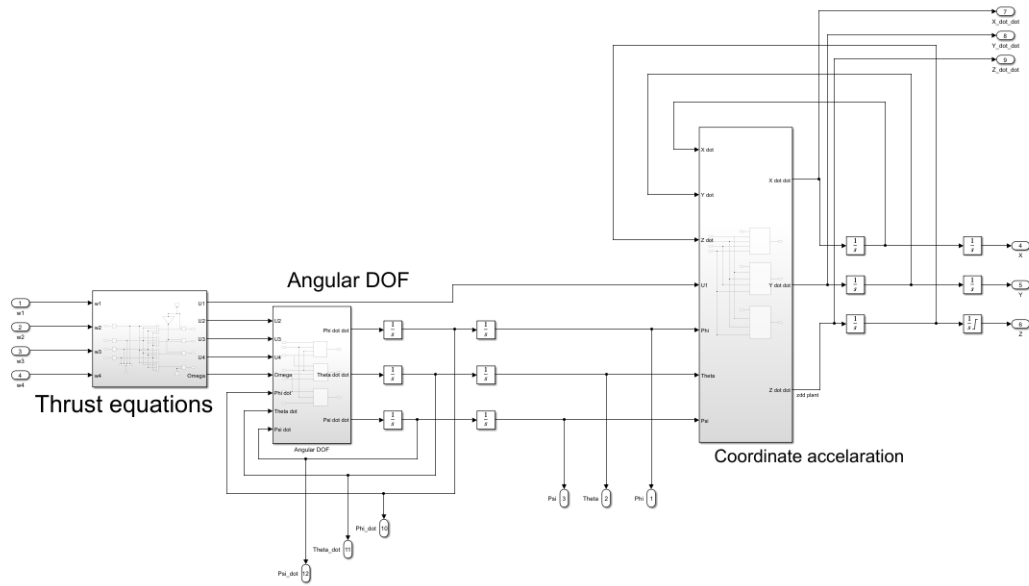


Figure 19: Plant overview in SIMULINK

7 Software prototyping

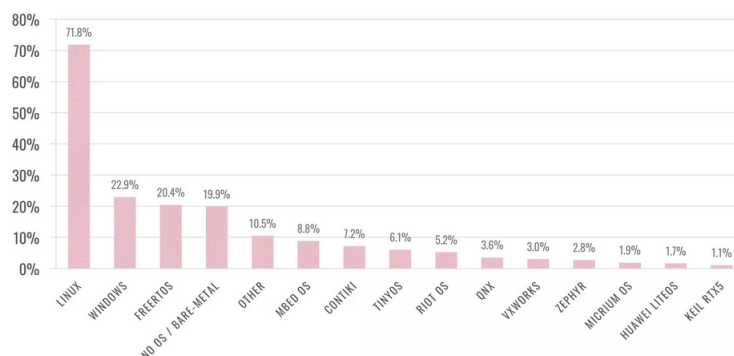
7.1 Embedded development environment

Software should 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." [13]

After research, the list was narrowed to two top contenders – FreeRTOS and Zephyr. Both solutions are open source, widely used and support the Microcontroller board we have chosen. [24] According to 2018 IoT Developer Survey [8], 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 20)

IoT OPERATING SYSTEMS

Which operating system(s) do you use for your IoT devices?



Copyright (c) 2018, Eclipse Foundation, Inc. | Made available under a [Creative Commons Attribution 4.0 International License](#) (CC BY 4.0).

25

Figure 20: IoT Developer Survey 2018 results

To decide between the RTOS choice, a pros and cons table was created and evaluated. [15] (See Table 1)

RTOS	Advantages	Disadvantages
FreeRTOS	<ul style="list-style-type: none"> • Suitable for begginers • Open source – online community support available • Libraries for Seeed XIAO Sense available • Constantly improved • Fast code execution • Low memory consupction 	<ul style="list-style-type: none"> • Not event-driven (scheduler will only be called once in a certain period of time) • Less flexible
ZephyrRTOS	<ul style="list-style-type: none"> • Open source – online community support available • Libraries for Seeed XIAO Sense available • Constantly improved • Designed to ensure energy efficiency • Highly configurable • Event-driven • Kernel can create additional system threads • Possible to exclude multi-threading • Additional debugging features • Supported by Nordic Semiconductor 	<ul style="list-style-type: none"> • More difficult to set-up • Potentially complicated to use with no experience

Table 1: Comparison between ZephyrRTOS and FreeRTOS

It was evaluated, that overall FreeRTOS seems to be a better established more simple solution to be used in case of no experience, whereas Zephyr offers more flexibility and is a fast growing popular solution well suited for our application and would be a worthwhile investment to build our skillset for future projects. [12]

PlatformIO is an open-source extension of Visual studio code, which supports hundreds of boards with different frameworks. [14] 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 crossreferencing 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 inbuilt serial USB communication, which results in inability to reprogram the board through the VS Code and need to go through bootloader first. This poses inconvenience, 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 even though it also does not have serial USB, it uses barebone Zephyr.

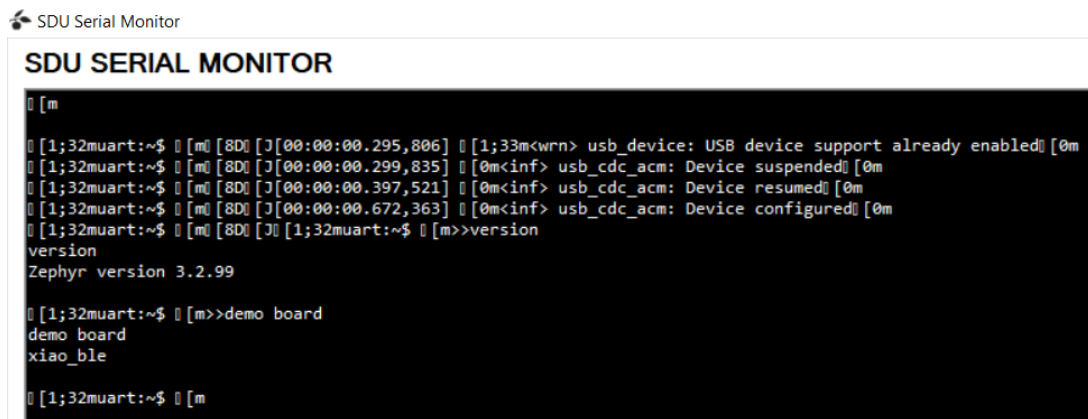
Upon further research, the Seeeduino XIAO nRF52840 Sense uses the nRF52840 microcontroller, so looking to Nordic Semiconductor nRF Connect SDK for development, which uses a RTOS called ZephyrRTOS. Additionally, the SDK provides useful tools for development, such as build, flashing and debugging actions. [20]

Seeed XIAO nRF52840 is not natively supported on nRF Connect SDK, but as ZephyrRTOS supports it. [18]

There is a caveat around this, we can fetch profiles from GitHub of the current version of Zephyr via `Zephyr/boards/arm/xiao_ble` at main branch [19] and import 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 file 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. [18] Furthermore, 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 prompt 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.

An issue occurred 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, mentions 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. [19] After building and flashing, results were achieved. (See Figure 21)



```
SDU SERIAL MONITOR
[m]
[1;32muart:~$ [m] [8D] [00:00:00.295,806] [1;33m<wrn> usb_device: USB device support already enabled [0m
[1;32muart:~$ [m] [8D] [00:00:00.299,835] [0m<inf> usb_cdc_acm: Device suspended [0m
[1;32muart:~$ [m] [8D] [00:00:00.397,521] [0m<inf> usb_cdc_acm: Device resumed [0m
[1;32muart:~$ [m] [8D] [00:00:00.672,363] [0m<inf> usb_cdc_acm: Device configured [0m
[1;32muart:~$ [m] [8D] [0] [1;32muart:~$ [m]>>version
version
Zephyr version 3.2.99

[1;32muart:~$ [m]>>demo board
demo board
xiao_ble

[1;32muart:~$ [m]
```

Figure 21: 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. [21] One can use a J-link debugging tool, but a caveat was found, which was GDB stub, which Zephyr supports, this would save budget. [18]

After achieving a successful development cycle, it was conducted, that the use of ZephyrRTOS is possible with our current setup. After further consultation with the supervisors, it was conducted, that the writers of the project have no skills in RTOS, more specifically in threading, and with the guidance of Davi, it was concluded, that acquiring such skills would be outside the scope of the project, as the main focus is control engineering.

7.2 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. [9] 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. [9] 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. [9] 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 μ s, was changed to 2500 μ s 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.

7.3 NanoBLEFlashPrefs

Data logging has evolved as a significant approach for overcoming the issues of the PID tuning process. [16] 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. [22] 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. 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. 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). [2] 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] [21]

References

- [1] Arduino® nano 33 ble sense, product reference manual, sku: Abx00031.
- [2] Dirk-/nanobleflashprefs: Substitute for the missing eeprom storage on arduino nano 33 ble and ble sense.
- [3] Wikipedia motor constants. https://en.wikipedia.org/wiki/Motor_constants.
- [4] 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.
- [5] 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>.
- [6] Cavoukian Ann. Privacy and drones: Unmanned aerial vehicles, 2012. <http://www.ipc.on.ca/English/Resources/Discussion-Papers/>.
- [7] 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.
- [8] Benjamin Cabé. Iot developer survey 2018, 2018. <https://www.slideshare.net/kartben/iot-developer-survey-2018>.
- [9] Dlloydev. Dlloydev/quickpid: A fast pid controller with multiple options. various integral anti-windup, proportional, derivative and timer control modes.
- [10] Trafikstyrelsen Droner. Flying drones in denmark, 2022. <https://www.droneregler.dk/english>.

- [11] M. Hassanalian and A. Abdelkefi. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*, 91:99–131, 5 2017.
- [12] Connected Industry. Freertos vs threadx vs zephyr: The fight for true open source rtos, 2023. <https://www.iiot-world.com/industrial-iiot/connected-industry/freertos-vs-threadx-vs-zephyr-the-fight-for-true-open-source-rtos/>.
- [13] 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>.
- [14] PlatformIO Labs. A professional collaborative platform for embedded development · platformio. <https://platformio.org/>.
- [15] micro ROS. Comparison between rtoses. <https://micro.ras.org/docs/concepts/rtos/comparison/>.
- [16] Dennis Nash. How to access data for optimal controller tuning | rockwell automation.
- [17] 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.
- [18] Zephyr Project. Introduction — zephyr project documentation, 2022. <https://docs.zephyrproject.org/latest/introduction/index.html>.
- [19] Zephyr Project. Zephyr project github repository, 2023. <https://github.com/zephyrproject-rtos/zephyr/>.
- [20] Nordic Semiconductor. nrf connect for vs code - nrf connect for vs code, 2022. <https://nrfconnect.github.io//vscode-nrf-connect/index.html>.

- [21] Seeed Studio. Getting started with seeed studio xiao nrf52840 (sense) | seeed studio wiki, 2023. https://wiki.seeedstudio.com/XIAO_BLE/.
- [22] 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.
- [23] 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>.
- [24] Oleksandr Zozulia. Freertos vs. zephyr project for embedded iot projects - leMBERg solutions, 2022. <https://leMBERgsolutions.com/blog/freertos-vs-zephyr-project-embedded-iot-projects>.

Motor thrust testing:

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

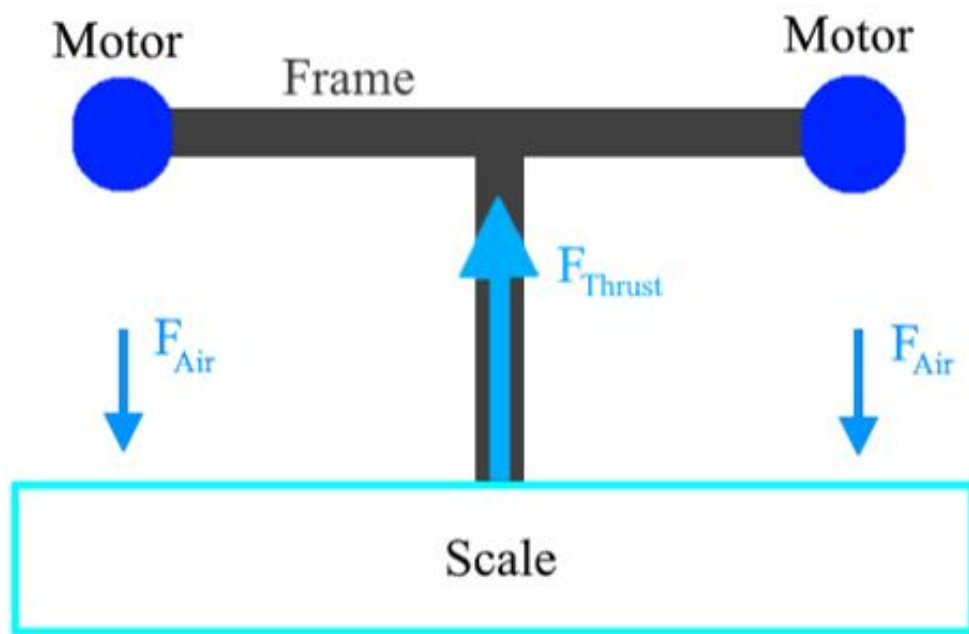
The initial idea to test the thrust provided by a certain motor was to create a x-shaped frame where, the motors were connected to their respective corners and a bolt was tightened in the centre as a means of weight. The frame with the motors were then placed on a scale where the motors were facing in 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



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

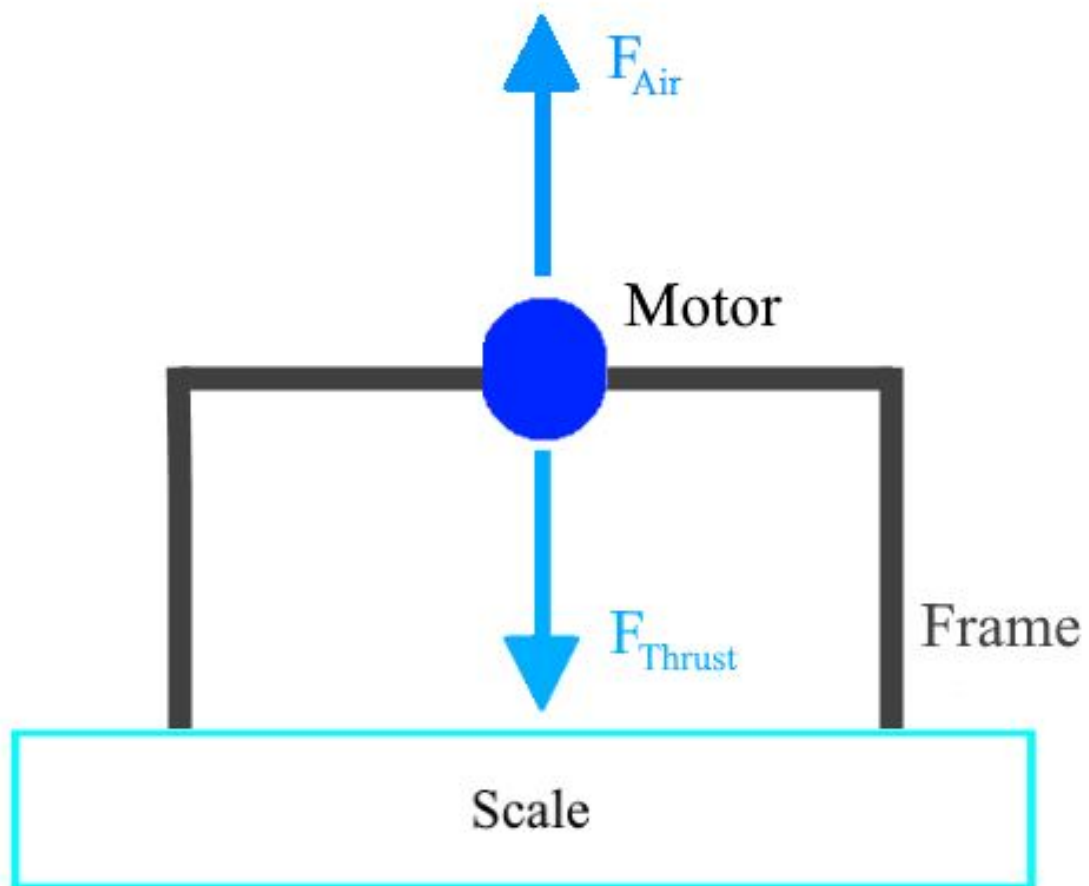
Figure 22: Initial Thrust Setup

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:

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.



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

Figure 23: Final Thrust Setup

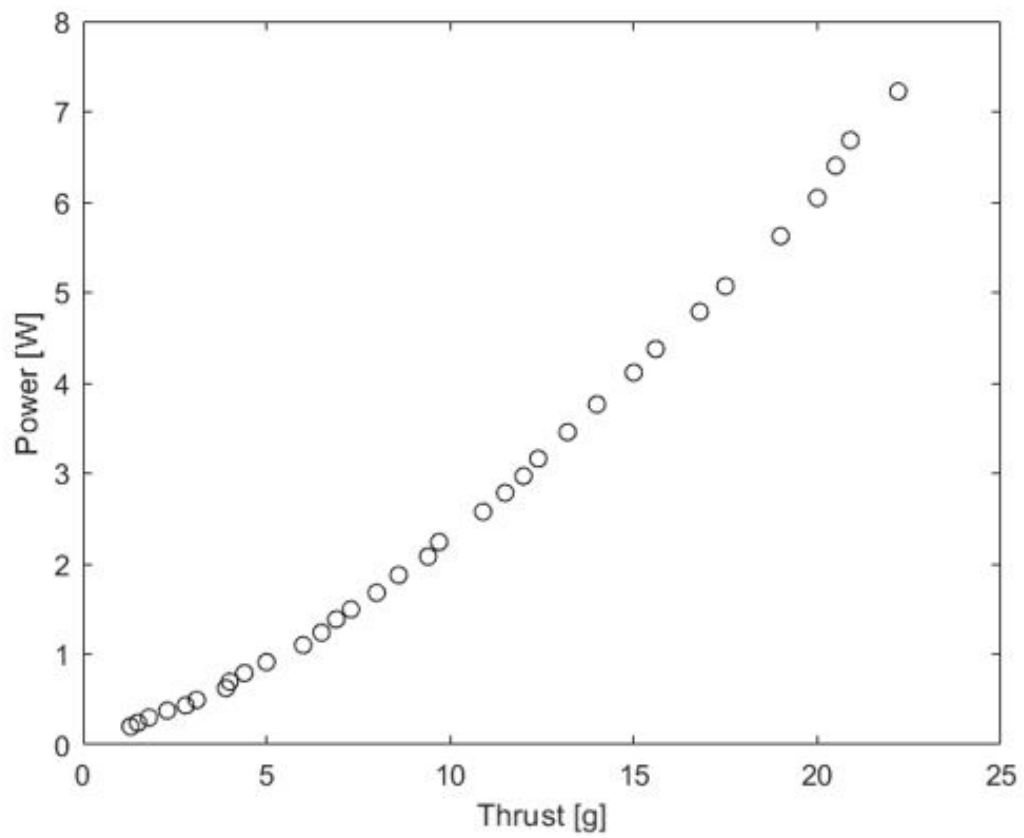


Figure 24: Final Thrust Setup