Lakehead University

Attitude Control of a Quadrotor UAV: Experimental Implementation

by

Biwen Tang

Under the Supervision of Dr. Abdelhamid Tayebi

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Control Engineering

Lakehead University, Thunder Bay, Ontario, Canada

September 2022

Abstract

Unmanned aerial vehicles (UAVs) are becoming ubiquitous due to their potential benefits in many civilian and military applications. UAVs are designed to accomplish complex tasks safely and effectively, such as search and rescue missions, package delivery, building inspections, film making...etc. The attitude controller is one of the most important and fundamental components that allows the UAV to balance itself in the air.

In this thesis, a PX4-based quadrotor experimental platform has been built and used to test the attitude control algorithms. MATLAB/Simulink R2021a with the UAV Toolbox Support Package for PX4 Autopilots is used to program the control algorithms. A nonlinear attitude estimator and three PID-type attitude controllers have been successfully implemented on the quadrotor.

A discussion about the performance of the obtained experimental results for the three different attitude control schemes is provided.

Acknowledgments

I would like to express my deepest appreciation to my supervisor, Dr. Abdelhamid Tayebi during my graduate studies at Lakehead University. He has proven to be an excellent supervisor and a role model in life who is always happy to share his considerable experience and knowledge. I sincerely thank him for his patience, support, and guidance, and for instilling in me a genuine passion for control engineering.

I would also like to thank our doctoral students Mouaad Boughellaba, Ishak Cheniouni, and Zeke Sedor for their friendship, insights, and kind help whenever it is needed. A special thanks to our postdoctoral fellow Miaomiao Wang for his support, knowledge, and efforts in assisting me during my studies. Additional thanks to Mr. Daniel Vasiliu for his assistance in fixing an issue I was facing with my motors after a crash, and my friend Keiran Smith for his assistance with the full-stick flight tests.

Finally, I am deeply grateful to my parents and my grandfather for their unconditional support, appreciation, encouragement, and keen interest in my academic achievements throughout my study and life.

Biwen Tang

Contents

List of Figures

[1]https://blog.csdn.net/SimuArch

[2]https://www.bosch-sensortec.com/media/boschsensortec/downloads/product_flyer/bst-bmi055-fl000.pdf

[3]https://docs.px4.io/main/en/concept/architecture.html

List of Tables

List of Abbreviations

UAV	–	Unmanned Aerial Vehicle.
SVD	–	Singular Value Decomposition.
QUEST	–	Quaternion Estimation.
KF	–	Kalman Filter.
EKF	–	Extended Kalman Filter.
MEKF	–	Multiplicative Extended Kalman Filter.
AEKF	–	Additive Extended Kalman Filter.
GPS	–	Global Positioning System.
PID	–	Proportional Integral Derivative.
PD	–	Proportional Derivative.
LQR	–	Linear-Quadratic Regulator.
$\mathsf{SE}(\mathbf{3})$	–	Special Euclidean group of dimension 3.
$\mathsf{SO}(\mathbf{3})$	–	Special Orthogonal group of dimension 3.
FFT	–	Fast Fourier Transform.
DCM	–	Direction Cosine Matrix.
APM	–	ArduPilot Mega.
ADC	–	Analog-to-Digital Converter.
SPI	–	Serial Peripheral Interface.
$\mathbf{I^2C}$	–	Inter-Integrated Circuit.
PWM	–	Pulse Width Modulation.
PPM	–	Pulse Position Modulation.
ESC	–	Electronic Speed Controller.
DC	–	Direct Current.
AC	–	Alternating Current.
EMF	–	Electro-Motive Force.
IGRF	–	International Geomagnetic Reference Field.
WMM	–	World Magnetic Model.
IMU	–	Inertial Measurement Unit.
DLPF	–	Digital Low Pass Filter.
LIPO	–	Lithium Polymer.

Chapter 1

Introduction

Unmanned aerial vehicles (UAVs), commonly known as drones, are flying robotic vehicles without human operators on board. Quadrotors were originally developed for military missions, then their uses were expanded to many non-military applications, due to their powerful capabilities, such as vertical take-off and landing (VTOL), hover capability, high maneuverability, and agility. In response to this disruptive innovation in the drone industry and the associated growth in civilian applications, research and development in UAV technology have become increasingly important. Embedded systems are commonly used to control UAVs, where C code is written, then compiled and loaded into the drone firmware to operate. By using the available open-source hardware and software platform, researchers and developers are enabled to implement their new control strategy and verify the performance and effectiveness. A survey of the publicly available open-source drone platforms that can be used for research and development can be found in this paper [1].

However, writing C code manually is complex and time-consuming, and it also requires relatively high software programming skills from developers. As a result, a significant number of developers have only modified the control structure and tweaked the control parameters based on the original open-source framework. Moreover, the use of C code makes it more difficult to visualize the control architecture and more challenging to understand how changes impact the whole system.

For rapid prototyping, it is interesting to use Matlab/Simulink environment for real-time implementation and control algorithms. In fact, a Simulink UAV toolbox for PX4 autopilots is available. With this method, flight controllers can be developed in Simulink by using block diagrams. From the Simulink models, C code is automatically generated, compiled, and uploaded onto the autopilot hardware. This mitigates the cost and challenges of quadrotor development as it enables users to devote more time to their control algorithm rather than to the details of programming. However, there is no free open-source autopilot system available, which uses a block-diagram-based design that requires little or no reliance on third-party software.

In this thesis, a MATLAB/Simulink-based software platform is built to develop and implement the attitude controllers for our quadrotor UAV. Three different PID-type attitude controllers are implemented and compared.

1.1 Overview of Quadrotor UAV Control Approaches

Developing an autonomous quadrotor aerial robot involves two main tasks: attitude estimation and attitude control. In order to control a quadrotor UAV, it is necessary to determine its orientation which is then used in the attitude control law. An overview of attitude estimation algorithms and attitude controllers is briefly discussed in the remainder of this section.

1.1.1 Attitude Estimation

Attitude cannot be measured directly, and it must be calculated (or estimated) from a set of measurements. In this case, the development of reliable attitude estimation algorithms that can run on low-cost computational hardware and low-cost sensing systems is the key to successful quadrotor UAV development. A number of attitude estimation/filtering techniques have been discussed in the literature [2,3].

Angular velocity measurements provided by gyroscopes can be used to integrate rigid body attitude kinematics equations theoretically. However, the gyroscope's measurements may be biased and noise may diverge beyond a few hundred milliseconds. Alternatively, inertial measurements can be used to construct the attitude algebraically [4]. Multiple attempts have been proposed to solve the attitude reconstruction problem from inertial vector measurements as an optimization problem, known as Wahba's problem, such as QUEST algorithm [5], Singular Value Decomposition (SVD) [6] and their extensions. However, significant errors may be generated from imperfect measurements provided by accelerometers and magnetometers, and/or imprecise knowledge of the considered inertial vectors.

The Kalman Filter (EK) theory and its advantages in real-time applications led to the development of many attitude filtering methods for aircraft systems. In Kalman filtering, two steps are performed: the first involves predicting the current state variables based on the previous estimated variables; the second involves updating the estimated state variables based on the weighted predicted states and noisy sensor measurements. In [7–9], extensions of the Kalman filter to nonlinear systems, known as the Extended Kalman Filter (EKF), have been proposed since attitude dynamics are nonlinear and the Kalman filter is originally designed for linear systems. Nevertheless, poor performances or even divergence arising from the real-time linearization in the EKF have led to the development of Multiplicative EKF (MEKF) [10] and Additive EKF (AEKF) [11]. Various versions of the Kalman filter have proven their effectiveness in real-time applications. In spite of this, it is difficult to prove closed-loop stability due to their computational complexity.

In recent decades, a class of new and powerful techniques relying on nonlinear observers have brought new hopes for the attitude estimation problem [12–22]. Compared to Kalman-like filtering techniques, nonlinear observers have some significant advantages, such as rigorous stability proofs and strong mathematical arguments. Moreover, some of the nonlinear attitude observers have been successfully implemented on low-cost microprocessors using the rotation matrix representation [14, 17, 18], and unit quaternion representation [16, 21, 22]. It is commonly assumed that the accelerometer measures the gravity vector in the body frame when designing the attitude observers by using inertial vector measurements under near-hover operations (*i.e.,* low translational accelerations) (see, for example, [14, 17, 19]). The attitude estimators are not guaranteed to provide reliable estimated attitudes under relatively high translational acceleration, because the accelerometer measurements are not equal to gravity any longer. To overcome this problem, an alternative attitude observer, known as the velocity-aided observer, relying on some additional measurements (such as the linear velocity obtained from a GPS) has been proposed in the literature (*e.g.,* [21]). Attitude estimation with gyro bias has been discussed in the literature [14, 17, 19, 22]. In practical applications, the magnetometer reading can also be affected by the magnetic fields generated by motors and other electrical equipment in the environment. To solve this problem, a vector decoupling strategy has been proposed in [19] and [23] to locally remove the disturbances of the magnetometers from the estimation dynamics of the roll and pitch. In [22], the authors extended the results to global vector decoupling and fully tested the estimation algorithm in real-time applications. This modification shows that the global vector decoupling estimation algorithm improved the overall quality of the nonlinear attitude estimators. However, only almost-global stability and local exponential stability have been achieved. The issue of global stability for attitude nonlinear estimation algorithms has remained unsolved in the literature until the work in [24] which relies on hybrid feedback techniques.

1.1.2 Attitude Control

Developing efficient attitude stabilization and/or attitude tracking schemes is one of the most critical aspects of autonomous quadrotor UAV control. A number of control techniques have been proposed in the literature, for instance [25–27] and their references.

One of the widely used methods known as the Proportional-Integral-Derivative (PID) controller has been implemented in [28] and [29]. However, the PID controller, as well as other linear controllers such as LQR [30], fail to work while dealing with aggressive maneuvers or large attitude errors, since the stability of these methods is only guaranteed in a restricted domain. Another approach has been proposed by introducing a decomposition of the quadrotor system into an outer-loop for attitude control and an inner-loop for body rate control [31]. The main advantage of this approach is that the process of the controller design is simplified. Similarly, it is not easy to show the stability of the overall system.

A variety of nonlinear control methods have been proposed in the literature [32–36], including feedback linearization, backstepping, and sliding mode control. The study, however, was limited to ideal dynamics without considering aerodynamic effects or sensor measurements, and only simulation results were presented. In [37–39], some practical nonlinear controllers with experimental results have been proposed. For example, a nonlinear PID-like controller designed directly on $\mathsf{SE}(3)$ for quadrotor UAV was tested [38]. In [37], the authors proposed a nonlinear control scheme based on Euler angles using a backstepping-like linearization method. However, in order to cancel the nonlinear terms in the time derivative of the Lyapunov function candidate, the control law requires information on model parameters like the moment of inertia. In [39], a new quaternion-based PD-like feedback control scheme was introduced for quadrotor stabilization. The PD feedback structure involves vector quaternion and angular velocity without any system model parameters, which is known as a model-independent controller. The proposed PD-like controller guarantees almost global asymptotic stability.

The attitude control problem, with full state feedback (*i.e.,* attitude and angular velocity) is well understood in the literature. Another performance and implementation-cost optimization issue that arises is the design of an attitude controller without angular velocity. Several attempts have been made by introducing an observer-like passive system to reconstruct the angular velocity in [40–43]. Although the angular velocity is not directly involved in these control algorithms, it is still required to construct the attitude in the attitude estimation algorithms. In [44], a true velocity-free attitude controller was first proposed based on inertial measurements and the unwinding phenomenon was avoided. The authors in [45–47] improved the result by introducing a modified observer-like passive system based on inertial measurements, which reduces the set of unstable equilibria.

1.2 Thesis Contribution

The thesis contributions are as follows:

- Obtained the moments of inertia parameters by using the bifilar pendulum principle with the attitude estimation algorithm proposed in [22] to collect the oscillation data around the roll, pitch, and yaw axes, respectively. The Fast Fourier Transform (FFT) has been applied to find the natural frequencies.
- Built a PX4-based quadrotor experimental platform (hardware and software) equipped with a Pixhawk 4 flight controller. Used MATLAB/Simulink and the UAV Toolbox Support Package for PX4 Autopilots to implement the attitude control schemes.
- Implemented a PD-like unit quaternion-based nonlinear attitude controller (proposed in [39]) in Chapter 6. The orientation of the quadrotor was estimated by a nonlinear attitude estimation algorithm proposed in [22]. Indoor flying results of the quadrotor UAV have been presented.
- Improved the flight performance of the PD-like controller, by introducing and implementing a PID-like controller and a nested PID-like controller.

1.3 Thesis Outline

This thesis is divided into 7 chapters. Chapter 1 provides an introduction to quadrotor UAVs, attitude estimation, and attitude control.

Chapter 2 provides a review of attitude representation. Chapter 3 presents the mathematical model of the quadrotor used in this thesis. An attitude estimator is introduced and implemented in Chapter 4, with the sensor calibration techniques used. Chapter 5 gives a detailed description of the quadrotor hardware and software setup, parameter identification, and related calibration and settings. Three PID-type unit quaternion-based attitude controllers have been implemented and compared in Chapter 6, with a non-linear attitude estimation algorithm.

Finally, Chapter 7 summarizes the work in this thesis and provides some suggestions for future possible research work.

Chapter 2

Attitude Representation

An attitude representation is a set of coordinates that describes the orientation of a given reference frame with respect to a second reference frame which more generally is a reference frame attached to the Earth, the Sun, or other stars. As shown in Fig. 2.1, let $\mathcal{I}$ denote the inertial (fixed) frame which is commonly attached to the Earth with the unit vector $\hat{z}_{\mathcal{I}}$ orthogonal to the ground and the unit vector $\hat{x}_{\mathcal{I}}$ pointing to the North direction, and $\mathcal{B}$ denote the body-attached frame which is attached to the quadrotor with the unit vector $\hat{z}_{\mathcal{B}}$ orthogonal to the platform and the unit vector $\hat{x}_{\mathcal{B}}$ pointing to the forward direction.

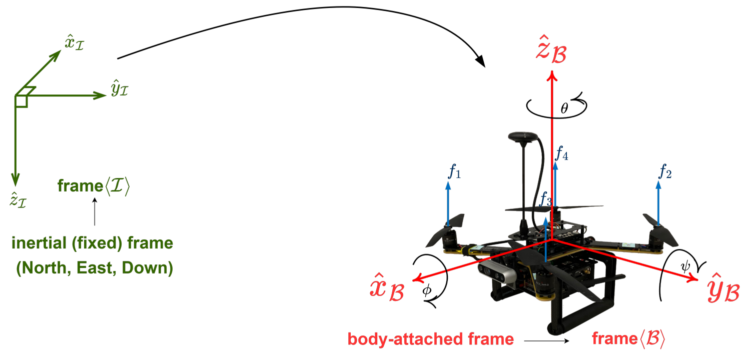


Figure 2.1: Configuration of the inertial frame $\mathcal{I}$ and the body frame $\mathcal{B}$

There are numerous attitude representations and three methods will be covered in this section to describe the rotational motion of a quadrotor in three-dimensional space: (1) Euler angles representation, (2) rotation matrix representation, and (3) unit quaternion representation. The attitude will be described as the orientation of the body-attached frame ($\mathcal{B}$) with respect to a fixed inertial frame ($\mathcal{I}$) (see [48] for more details).

2.1 Euler Angles Representation

The Euler angles, introduced by Leonard Euler, describe the attitude of a reference frame relative to an inertial frame by three successive rotations about the body fixed axes. The orientation of a rigid body in a three dimensional Euclidean space is given by

$$(\phi,\ \theta,\ \psi)\ (rad) \tag{2.1}$$

where, the three parameters (ϕ, θ, ψ) are known as roll, pitch, and yaw, respectively.

Euler angles are easy to visualize because of their clear physical definitions. However, there exists an inherent problem in the Euler angles representation known as the singularity problem. The Euler angles encounter singularity for $\theta = \pm k\pi$, where k is any integer $(k = 1, 2, \cdots)$. This geometric singularity further induces the singularity in the corresponding Euler angles kinematic differential equations [49].

2.2 Rotation Matrix Representation

The rotation matrix, also known as the Direction Cosine Matrix (DCM), is the most popular representation of the attitude of a rigid body. Consider the orientation of frame $\mathcal{B}$ with respect to the inertial frame $\mathcal{I}$. It can be described by three vectors:

$$^{\mathcal{I}}\hat{X}_{\mathcal{B}} = \begin{bmatrix} \hat{x}_{\mathcal{B}} \cdot \hat{x}_{\mathcal{I}} \\ \hat{y}_{\mathcal{B}} \cdot \hat{x}_{\mathcal{I}} \\ \hat{z}_{\mathcal{B}} \cdot \hat{x}_{\mathcal{I}} \end{bmatrix} \quad {}^{\mathcal{I}}\hat{Y}_{\mathcal{B}} = \begin{bmatrix} \hat{x}_{\mathcal{B}} \cdot \hat{y}_{\mathcal{I}} \\ \hat{y}_{\mathcal{B}} \cdot \hat{y}_{\mathcal{I}} \\ \hat{z}_{\mathcal{B}} \cdot \hat{y}_{\mathcal{I}} \end{bmatrix} \quad {}^{\mathcal{I}}\hat{Z}_{\mathcal{B}} = \begin{bmatrix} \hat{x}_{\mathcal{B}} \cdot \hat{z}_{\mathcal{I}} \\ \hat{y}_{\mathcal{B}} \cdot \hat{z}_{\mathcal{I}} \\ \hat{z}_{\mathcal{B}} \cdot \hat{z}_{\mathcal{I}} \end{bmatrix}$$

where, $\hat{x}_{\mathcal{I}}, \hat{y}_{\mathcal{I}}$ and $\hat{z}_{\mathcal{I}}$ are the coordinate vectors of frame $\mathcal{I}$, $\hat{x}_{\mathcal{B}}, \hat{y}_{\mathcal{B}}$ and $\hat{z}_{\mathcal{B}}$ are the coordinate vectors of frame $\mathcal{B}$, and ${}^{\mathcal{I}}\hat{X}_{\mathcal{B}}, {}^{\mathcal{I}}\hat{Y}_{\mathcal{B}}$ and ${}^{\mathcal{I}}\hat{Z}_{\mathcal{B}}$ are unit vectors corresponding to the orthogonal projection of the coordinate vectors of frame $\mathcal{B}$ onto the coordinate vectors of frame $\mathcal{I}$. Note that $u \cdot v = \|u\|\|v\| \cos\vartheta$ represents the dot product and ϑ is the angle between the vectors u and v. The rotation matrix of frame $\mathcal{B}$ with respect to frame $\mathcal{I}$ is defined as

$$^{\mathcal{I}}_{\mathcal{B}}R = \begin{bmatrix} {}^{\mathcal{I}}\hat{X}_{\mathcal{B}}^{\top} & {}^{\mathcal{I}}\hat{Y}_{\mathcal{B}}^{\top} & {}^{\mathcal{I}}\hat{Z}_{\mathcal{B}}^{\top} \end{bmatrix} = \begin{bmatrix} \hat{x}_{\mathcal{B}} \cdot \hat{x}_{\mathcal{I}} & \hat{y}_{\mathcal{B}} \cdot \hat{x}_{\mathcal{I}} & \hat{z}_{\mathcal{B}} \cdot \hat{x}_{\mathcal{I}} \\ \hat{x}_{\mathcal{B}} \cdot \hat{y}_{\mathcal{I}} & \hat{y}_{\mathcal{B}} \cdot \hat{y}_{\mathcal{I}} & \hat{z}_{\mathcal{B}} \cdot \hat{y}_{\mathcal{I}} \\ \hat{x}_{\mathcal{B}} \cdot \hat{z}_{\mathcal{I}} & \hat{y}_{\mathcal{B}} \cdot \hat{z}_{\mathcal{I}} & \hat{z}_{\mathcal{B}} \cdot \hat{z}_{\mathcal{I}} \end{bmatrix}$$

The rotation matrix belongs to the *special orthogonal group* of dimension three $\mathsf{SO}(3)$. Let $R \in \mathsf{SO}(3)$ be a rotation matrix, then one has

$$R^{\top}R = RR^{\top} = I_3 \quad \det(R) = 1$$

where, I_3 is the 3×3 identity matrix. We restrict our analysis to $\det(R) = 1$, since rotation matrices for which $\det(R) = -1$ are not rigid-body transformations [50]. Assume that R denotes a rotation from the inertial frame $\mathcal{I}$ to the body-attached frame $\mathcal{B}$. Let $v_{\mathcal{I}} \in \mathbb{R}^3$ be the coordinates of a vector in frame $\mathcal{I}$, then the coordinates of this vector expressed in frame $\mathcal{B}$ are given by

$$v_{\mathcal{B}} = R^{\top} v_{\mathcal{I}}$$

where $v_{\mathcal{B}} \in \mathbb{R}^3$. The rotation matrix allows for easy computation of multiple rotations through simple matrix multiplication.

2.3 Unit Quaternion Representation

Another representation of the attitude of a rigid body is the unit quaternion, which is defined as

$$Q = \begin{bmatrix} \eta \\ q \end{bmatrix}, \qquad (\eta^2 + q^\top q = 1) \tag{2.2}$$

where, $\eta \in \mathbb{R}$ and $q \in \mathbb{R}^3$ are the scalar part and the vector part of the quaternion, respectively. The unit quaternion can also be written as

$$Q = \begin{bmatrix} \cos(\varphi/2) \\ \sin(\varphi/2)\hat{k} \end{bmatrix} \tag{2.3}$$

which represents a rotation by an angle φ about an arbitrary unit vector $\hat{k}$. The unit quaternion belongs to the set of unit quaternions given by

$$\mathbb{Q} = \{Q \in \mathbb{R}^4 \mid \quad |Q| = 1\} \tag{2.4}$$

Define two unit quaternions Q_1 and Q_2 as:

$$Q_1 = \begin{bmatrix} \eta_1 \\ q_{v1} \end{bmatrix} \quad Q_2 = \begin{bmatrix} \eta_2 \\ q_{v2} \end{bmatrix}$$

Then, the quaternion multiplication is given by

$$Q_1 \odot Q_2 = \begin{bmatrix} \eta_1\eta_2 - q_{v1}^\top q_{v2} \\ \eta_1 q_{v2} + \eta_2 q_{v1} + q_{v1} \times q_{v2} \end{bmatrix} \tag{2.5}$$

where, $\odot$ denotes the quaternion product and $\times$ denotes the vector cross product. The unit quaternion multiplication is non-commutative since the cross product is non-commutative. The inverse of a unit quaternion Q is given by

$$Q^{-1} = \begin{bmatrix} \eta \\ -q \end{bmatrix} \tag{2.6}$$

and one has

$$Q \odot Q^{-1} = Q^{-1} \odot Q = Q_I = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.7}$$

where Q_I is the identity quaternion, which can be viewed as a rotation by a zero angle about an arbitrary unit vector. The unit quaternion multiplication can also be used to transform a vector from one reference frame to another. Let $v_{\mathcal{I}} \in \mathbb{R}^3$ be a vector in the inertial frame $\mathcal{I}$ and $v_{\mathcal{B}} \in \mathbb{R}^3$ be a vector projection of $v_{\mathcal{I}}$ in the body-attached frame $\mathcal{B}$. Then

$$\begin{bmatrix} 0 \\ v_{\mathcal{B}} \end{bmatrix} = Q^{-1} \odot \begin{bmatrix} 0 \\ v_{\mathcal{I}} \end{bmatrix} \odot Q \tag{2.8}$$

where Q represents the attitude of the body frame. The quaternion representation has some advantages over the other attitude representations. Its reduced number of parameters makes it more suitable for real-time implementations. The main drawback of the quaternion representation is its non-uniqueness. In particular, this is a twofold covering map, where there exist a pair of antipodal unit quaternions $\pm Q \in \mathbb{Q}$ for each attitude $R \in \mathsf{SO}(3)$. Without careful designing, quaternion-based controllers may cause undesirable phenomena such as unwinding [51].

2.4 Comparison of Attitude Representations

From the above discussion, we can conclude that each attitude representation has its own pros and cons. The comparison of the above attitude representations is given by Table 2.1 (see [26] for more details).

	Euler Angles	**Unit Quaternion**	**Rotation Matrix**
Pros	Easy to visualize	No singularities, simplicity	No singularities
Cons	Singularity problem	Non-unique, no intuitive physical meanings	Computationally expensive
Global	×	✓	✓
Unique	×	×	✓
Para	3 parameters	4 × 1 unit vector (4)	3 × 3 matrix (9)

Table 2.1: Comparison of attitude representations

In practical applications, different attitude representations can be combined for specific requirements. Some useful equalities used in the remainder of the thesis are presented as follows:

- *Unit Quaternion and Euler Angles*
  The unit-quaternion can be obtained from the Euler angles as follows:

$$Q = \begin{bmatrix} \eta \\ q \end{bmatrix} = \begin{bmatrix} c(\phi/2)c(\theta/2)c(\psi/2) + s(\phi/2)s(\theta/2)s(\psi/2) \\ s(\phi/2)c(\theta/2)c(\psi/2) - c(\phi/2)s(\theta/2)s(\psi/2) \\ c(\phi/2)s(\theta/2)c(\psi/2) + s(\phi/2)c(\theta/2)s(\psi/2) \\ c(\phi/2)c(\theta/2)s(\psi/2) - s(\phi/2)s(\theta/2)c(\psi/2) \end{bmatrix} \tag{2.9}$$

  with c and s denoting the cosine and sine operators. The quaternion can also be converted to Euler angles as follows:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}(2(\eta q_1 + q_2 q_3), 1 - 2(q_1^2 + q_2^2)) \\ \arcsin(2(\eta q_2 - q_3 q_1)) \\ \text{atan2}(2(\eta q_3 + q_1 q_2), 1 - 2(q_2^2 + q_3^2)) \end{bmatrix} \tag{2.10}$$

  where, $\text{atan2}(\cdot) : \mathbb{R} \times \mathbb{R} \to (-\pi, \pi]$ denotes the four-quadrant inverse tangent. It is easy to check that $Q = Q_I$ for $\phi = 0, \theta = 0$ and $\psi = 0$.

- *Rotation Matrix and Euler Angles*
  The function that maps a vector of Euler angles to its rotation matrix from $(x - y - z)$

is given by

$$R = R_x(\phi)R_y(\theta)R_z(\psi) = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{2.11}$$

where $R_x(\phi)$, $R_y(\theta)$ and $R_z(\psi)$ denote rotations around the x, y and z axes by angles ϕ, θ and ψ, respectively.

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \; R_y(\theta) = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \; R_z(\psi) = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where, it is easy to check that $R = R_x(0)R_y(0)R_z(0) = I_3$

- *Unit Quaternion and Rotation Matrix*
  The rotation matrix is related to the unit-quaternion through the Rodriguez formula $R = \mathcal{R}(Q)$ [48]. The mapping $\mathcal{R} : \mathbb{Q} \to \mathsf{SO}(3)$ is given by

$$\mathcal{R}(Q) = I_3 + 2\eta S(q) + 2S(q)^2 = (\eta^2 - q^\top q)I_3 + 2qq^\top + 2\eta S(q) \tag{2.12}$$

where, $S : \mathbb{R}^3 \to \mathfrak{so}(3)$, and

$$S(x) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

with $x = [x_1, x_2, x_3]^\top$, and $\mathfrak{so}(3) = \{S \in \mathbb{R}^{3\times 3} \mid S^\top = -S\}$ denotes the set of 3×3 skew symmetric matrices. Given a rotation matrix R and two vectors $x, y \in \mathbb{R}^3$, we have the following useful properties: $S(x)y = -S(y)x = x \times y$, $S(x)x = 0, S(x)S(y) = yx^\top - (x^\top y)I_3$, $S(S(x)y) = S(x)S(y) - S(y)S(x) = yx^\top - xy^\top$ and $S(Rx) = RS(x)R^\top$. The rotation matrix (2.12) can be further expanded in a matrix form as

$$\mathcal{R}(Q) = \begin{bmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1q_2 - \eta q_3) & 2(\eta q_2 + q_1q_3) \\ 2(q_1q_2 + \eta q_3) & 1 - 2(q_1^2 + q_3^2) & 2(q_2q_3 - \eta q_1) \\ 2(q_1q_3 - \eta q_2) & 2(\eta q_1 + q_2q_3) & 1 - 2(q_1^2 + q_2^2) \end{bmatrix}$$

where, it is easy to check that $\mathcal{R}(Q_I) = I_3$.

Note that the mapping from $\mathsf{SO}(3)$ to $\mathbb{Q}$ is not a one-to-one mapping. The two unit quaternions Q and $-Q$ represent the same rotation matrix R.

Chapter 3

Mathematical Model

3.1 Quadrotor Motion Description

The quadrotor aircraft under consideration consists of a rigid cross frame equipped with four motors as shown in Fig.3.1. The up (down) motion is achieved by increasing (decreasing) the total thrust while maintaining an equal individual thrust. The roll, pitch, and yaw motions are achieved through a differential control strategy of the thrust generated by each motor. In order to avoid the yaw drift due to the reactive torques, the quadrotor aircraft is configured such that the set of motors (motor 1 and 2) rotates clockwise and the set of motors (motor 3 and 4) rotates counterclockwise. There is no change in the direction of rotation of the motors ($i.e.,$, $\omega_i \geq 0, \quad i \in \{1,2,3,4\}$). If a yaw motion is desired, one has to reduce the thrust of one set of motors and increase the thrust of the other set while maintaining the same total thrust to avoid an up–down motion. Hence, the yaw motion is then realized in the direction of the induced reactive torque. On the other hand, forward (backward) motion is achieved by pitching in the desired direction by increasing the rear (front) motors thrust and decreasing the front (rear) motors thrust to maintain the total thrust. Finally, a sideways motion is achieved by rolling in the desired direction by increasing the left (right) motors thrust and decreasing the right (left) motors thrust to maintain the total thrust.

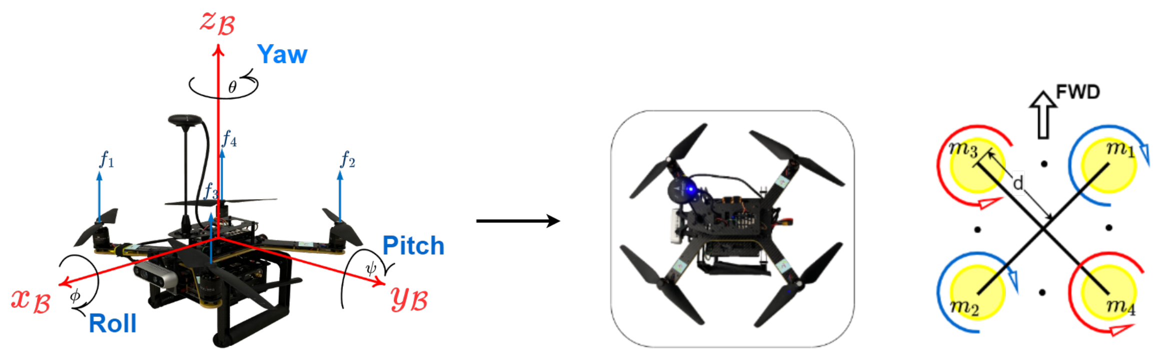


Figure 3.1: Quadrotor orientation and airframe diagram

The general motion of a quadrotor UAV in space is a combination of translational and rotational motions. The translational dynamics are given by

$$\begin{cases} \dot{p} &= v \\ \dot{v} &= ge_3 - \dfrac{1}{m}\mathcal{T}Re_3 \end{cases} \tag{3.1}$$

where, $e_3 := [0,0,1]^\top \in \mathbb{R}^3$, and p and v are respectively the position and the linear velocity of the origin of the airframe (frame $\mathcal{B}$) with respect to the frame $\mathcal{I}$. The mass of the quadrotor is denoted m, and $\mathcal{T}$ denotes the total thrust generated by the four motors, $R \in \mathsf{SO}(3)$ is the orientation of the airframe with respect to the frame $\mathcal{I}$. The rotational dynamics of the rigid body are given by

$$\begin{cases} \dot{Q} &= \dfrac{1}{2}\begin{bmatrix} -q^\top \\ S(q) + q_0 I \end{bmatrix}\omega \\ I_f\dot{\omega} &= -\,\omega \times I_f\omega + \tau \end{cases} \tag{3.2}$$

where, $Q := [\eta, q]^\top$ is the unit quaternion representation of the quadrotor orientation, ω is the rigid body angular velocity expressed in frame $\mathcal{B}$, $I_f \in \mathbb{R}^{3\times3}$ is the symmetric positive-definite constant inertia matrix around the center of mass expressed in frame $\mathcal{B}$, and $\tau \in \mathbb{R}^3$ is the airframe torque generated by the four motors.

The total thrust $\mathcal{T}$ generated by the four motors is given by

$$\mathcal{T} = \sum_{i=1}^{4} |f_i| = b\sum_{i=1}^{4} \varpi_i^2 \tag{3.3}$$

where, b is a positive coefficient and $\varpi_i \in \mathbb{R}$ is the angular velocity of motor i, and the reactive torque Q_i generated by the motor i due to rotor drag is given by

$$Q_i := -\kappa\varpi_i^2$$

where, κ is a positive coefficient. By following the configuration shown in Fig.3.1, we can easily get the airframe torques $\tau = [\tau_\phi, \tau_\theta, \tau_\psi]^\top$ and total thrust $\mathcal{T}$ as

$$\tau_\phi = \frac{db}{\sqrt{2}}(-\varpi_1^2 + \varpi_2^2 + \varpi_3^2 - \varpi_4^2) \tag{3.4}$$

$$\tau_\theta = \frac{db}{\sqrt{2}}(\varpi_1^2 - \varpi_2^2 + \varpi_3^2 - \varpi_4^2) \tag{3.5}$$

$$\tau_\psi = \kappa(\varpi_1^2 + \varpi_2^2 - \varpi_3^2 - \varpi_4^2) \tag{3.6}$$

$$\mathcal{T} = b(\varpi_1^2 + \varpi_2^2 + \varpi_3^2 + \varpi_4^2) \tag{3.7}$$

where, b is a positive coefficient and d is the distance between the rotor and the center mass of the quadrotor aircraft. The coefficients κ and b are dependent on the density of air, size, and shape of blades, as well as other factors (see [52] and [27] for more details). These two parameters will be identified in Section 3.2.

The complete dynamical model of a quadrotor described in [53] and [39] is given by

$$\dot{p} = v \tag{3.8}$$

$$\dot{v} = ge_3 - \frac{1}{m}\mathcal{T}Re_3 \tag{3.9}$$

$$\dot{R} = RS(\omega) \tag{3.10}$$

$$I_f\dot{\omega} = -\omega \times I_f\omega + \tau \tag{3.11}$$

$$I_r\dot{\varpi}_i = \tau_i - Q_i, \quad i \in \{1,2,3,4\} \tag{3.12}$$

where

$p \in \mathbb{R}^3$: position of the origin of the airframe (frame $\mathcal{B}$) with respect to the frame $\mathcal{I}$,

$v \in \mathbb{R}^3$: linear velocity of the quadrotor expressed in frame $\mathcal{I}$,

$m \in \mathbb{R}$: mass of the quadrotor,

$g \in \mathbb{R}$: acceleration due to gravity,

$e_3 := [0,0,1]^\top \in \mathbb{R}^3$: unit vector expressed in frame $\mathcal{I}$,

$\mathcal{T} \in \mathbb{R}$: total thrust generated by the four motors,

$R \in \mathsf{SO}(3)$: orientation of the airframe,

$\omega \in \mathbb{R}^3$: angular velocity of quadrotor expressed in the body-fixed frame $\mathcal{B}$,

$I_f \in \mathbb{R}^{3\times3}$: symmetric positive-definite constant inertial matrix around the center of mass expressed in frame $\mathcal{B}$,

$\tau \in \mathbb{R}^3$: airframe torque generated by the rotors,

$\varpi_i \in \mathbb{R}$: angular velocity of the motor i,

$\tau_i \in \mathbb{R}$: torque produced by the motor i,

$I_r \in \mathbb{R}$: the moment of inertia of the motor i.

The rotational dynamics can also be written in terms of the Euler angles representation [54] as

$$\begin{aligned}
\dot{\phi} &= \omega_1 + \omega_2 \sin\phi \tan\theta + \omega_3 \cos\phi \tan\theta \\
\dot{\theta} &= \omega_2 \cos\phi - \omega_3 \sin\phi \\
\dot{\psi} &= \omega_2 \sin\phi \sec\theta + \omega_3 \cos\phi \sec\theta \\
\dot{\omega}_1 &= \frac{(I_\theta - I_\psi)}{I_\phi}\omega_2\omega_3 + \frac{1}{I_\phi}\tau_\phi \\
\dot{\omega}_2 &= \frac{(I_\psi - I_\phi)}{I_\theta}\omega_1\omega_3 + \frac{1}{I_\theta}\tau_\theta \\
\dot{\omega}_3 &= \frac{(I_\phi - I_\theta)}{I_\psi}\omega_1\omega_2 + \frac{1}{I_\psi}\tau_\psi
\end{aligned} \tag{3.13}$$

where, the inertia matrix $I_f = diag(I_\phi, I_\theta, I_\psi)$, angular velocity $\omega = [\omega_1, \omega_2, \omega_3]^\top$, and control torque $\tau = [\tau_\phi, \tau_\theta, \tau_\psi]^\top$.

3.2 Model Identification

3.2.1 Mass

The mass of the quadrotor is measured by using a digital scale, the mass is found to be 1.3913 kg.

3.2.2 Propeller Aerodynamics

Based on the aerodynamics model, the steady-state thrust generated by the propeller while hovering (*i.e.,* a rotor that is not translating horizontally or vertically) in free air can be modeled using momentum theory (see [55] and Section 2.2.6 in [56] for more details) as

$$\mathcal{T} = C_T \rho n^2 D^4 = b\varpi^2 \tag{3.14}$$

where, C_T is the coefficient of thrust dependent on the propeller geometry and aerodynamics characteristics, ρ is the density of air, n is the speed revolutions per sec, D is the diameter of the propeller, ϖ is the angular velocity of the motor, and b denotes the thrust factor.

The propeller reactive torque is generated by the propeller motion as follows

$$Q = C_Q \rho n^2 D^5 = \kappa\varpi^2 \tag{3.15}$$

where, $C_Q = C_P/2\pi$ and C_P are the power coefficients of the propeller dependent on the propeller geometry and aerodynamics characteristics, and κ denotes the torque coefficient.

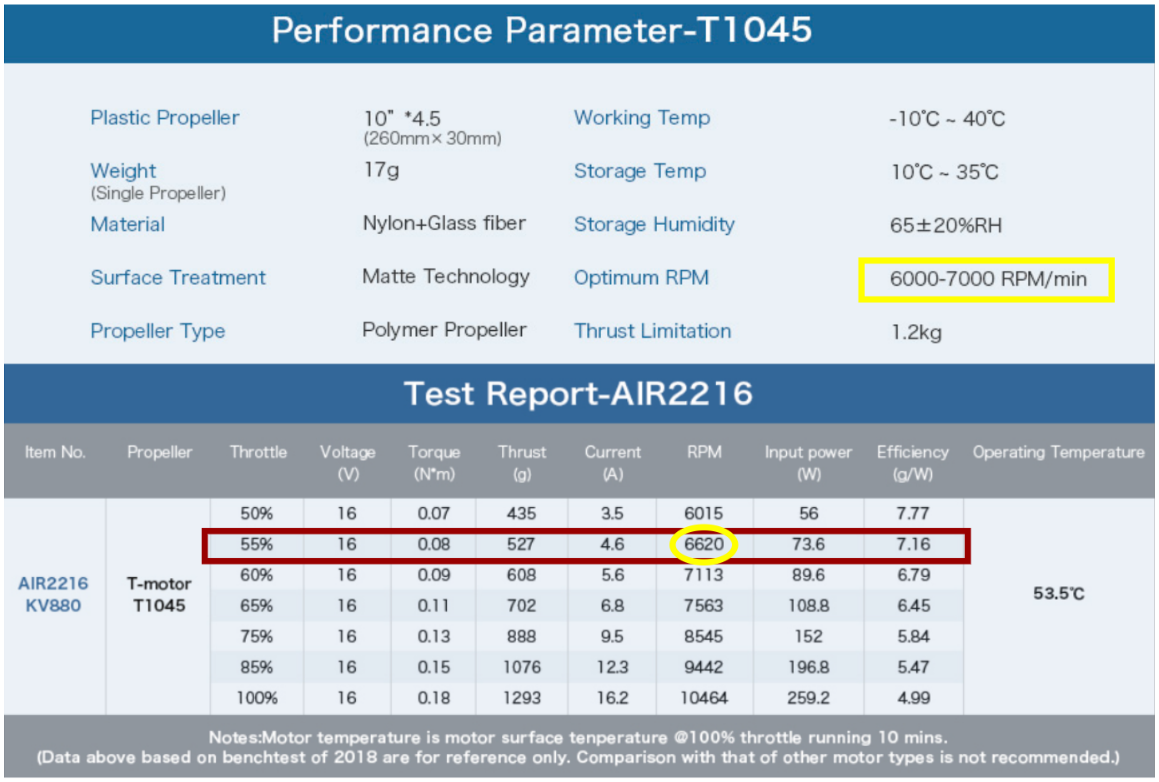
Name	Parameter	Value	Units
Steady-state Thrust	$\mathcal{T}$	0.527	kg
Reactive Torque	Q	0.08	$N \cdot m$
RPM (53.5℃)	$-$	6620	$revolution/min(rpm)$
Angular Velocity	ϖ	693.2448	rad/s

Table 3.1: Propeller thrust parameters

The propellers used in our experiments are the T-motor T1045 propellers with a 10-inch diameter and a pitch of 4.5-inch per revolution. The propeller test report is given in Fig.3.2 with the optimum RPM within a range between 6000 to 7000 *rpm*. Thus we take the performance of a 55% throttle and get the corresponding steady-state thrust $\mathcal{T}$ and reactive torque Q and write them down in Table 3.1. From (3.14) and (3.15), and the parameters provided in Table 3.1, one can obtain the thrust coefficient b and torque coefficient κ as

$$b = \frac{\mathcal{T}}{\varpi^2} = 1.0757e^{-5}\ N \cdot s^2/rad^2 \tag{3.16}$$

$$\kappa = \frac{Q}{\varpi^2} = 1.6646e^{-7}\ N \cdot m \cdot s^s/rad^2 \tag{3.17}$$



Performance Parameter-T1045

Plastic Propeller	10" *4.5 (260mm× 30mm)	Working Temp	-10℃ ~ 40℃
Weight (Single Propeller)	17g	Storage Temp	10℃ ~ 35℃
Material	Nylon+Glass fiber	Storage Humidity	65±20%RH
Surface Treatment	Matte Technology	Optimum RPM	6000-7000 RPM/min
Propeller Type	Polymer Propeller	Thrust Limitation	1.2kg

Test Report-AIR2216

Item No.	Propeller	Throttle	Voltage (V)	Torque (N*m)	Thrust (g)	Current (A)	RPM	Input power (W)	Efficiency (g/W)	Operating Temperature
AIR2216 KV880	T-motor T1045	50%	16	0.07	435	3.5	6015	56	7.77	53.5℃
		55%	16	0.08	527	4.6	6620	73.6	7.16	
		60%	16	0.09	608	5.6	7113	89.6	6.79	
		65%	16	0.11	702	6.8	7563	108.8	6.45	
		75%	16	0.13	888	9.5	8545	152	5.84	
		85%	16	0.15	1076	12.3	9442	196.8	5.47	
		100%	16	0.18	1293	16.2	10464	259.2	4.99	

Notes:Motor temperature is motor surface tenperature @100% throttle running 10 mins.
(Data above based on benchtest of 2018 are for reference only. Comparison with that of other motor types is not recommended.)

Figure 3.2: T-motor propeller test report

3.2.3 Moment of Inertia

The moments of inertia of an aircraft are important in understanding its aerodynamic properties and thus its translational and rotational motion during flight. In our case, they can be determined with the bifilar pendulum method [57] and [58]. As shown in Fig.3.3, a bifilar pendulum consists of suspending an airframe from two parallel wires, or filars, that allow it to rotate freely about a given axis to measure the moments of inertia for the axis of rotation parallel to the filars.

As shown in Fig.3.4, three small moments are applied to the airframe respectively along the $x_{\mathcal{B}}$, $y_{\mathcal{B}}$, and $z_{\mathcal{B}}$ axis to obtain the natural frequencies of the oscillations for roll pitch and yaw, denoted by $f_{n,\phi}$, $f_{n,\theta}$, and $f_{n,\psi}$, by using the Fast Fourier Transform (FFT) in MATLAB. The moment of inertia can be obtained as

$$I_i = \frac{\mathrm{d}^2 \cdot mg}{16\pi^2 \cdot l \cdot {f_{n,i}}^2} \tag{3.18}$$

where

d: distance between the strings (m),

m: mass of the test object (kg),

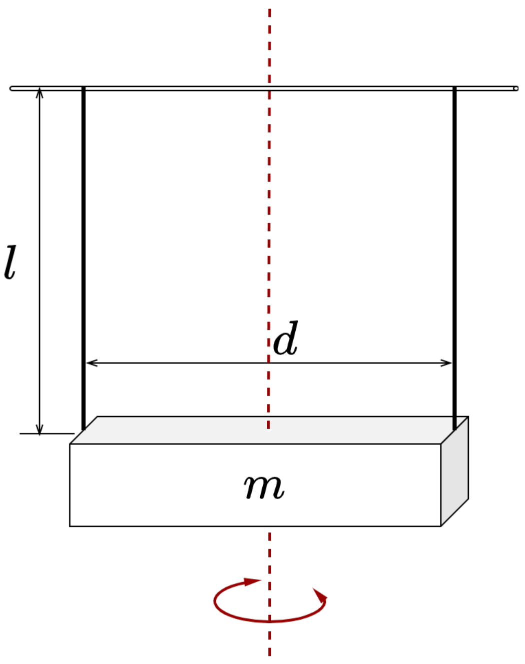


Figure 3.3: The bifilar pendulum principle model

l: length of the strings suspending the object (m),

$f_{n,i}$: the natural frequency of the oscillation (sec^{-1}).

Tests are applied respectively for the roll, pitch, and yaw three times each, and the natural frequencies have been shown in Table 3.2

	Test 1	Test 2	Test 3
Natural frequency of roll	1.06673	1.06732	1.06589
Natural frequency of pitch	1.00344	1.00102	1.00339
Natural frequency of yaw	1.28866	1.28874	1.28623

Table 3.2: Natural frequencies got from experiments

Taking the average values from the table, the natural frequencies can be obtained as follows:

$$f_{n,\phi} = 1.06665\ sec^{-1} \quad f_{n,\theta} = 1.00262\ sec^{-1} \quad f_{n,\psi} = 1.28788\ sec^{-1} \tag{3.19}$$

Substituting these natural frequencies in (3.18) with $m = 1.3913kg$, $l = 0.540m$, d $= 0.285m$ (for the pitch and roll) and d $= 0.430m$ (for the yaw), we have the following moment of inertia around the roll, pitch and yaw axes respectively:

$$I_\phi = 0.0114\ kg \cdot m^2 \quad I_\theta = 0.0129\ kg \cdot m^2 \quad I_\psi = 0.0178\ kg \cdot m^2 \tag{3.20}$$

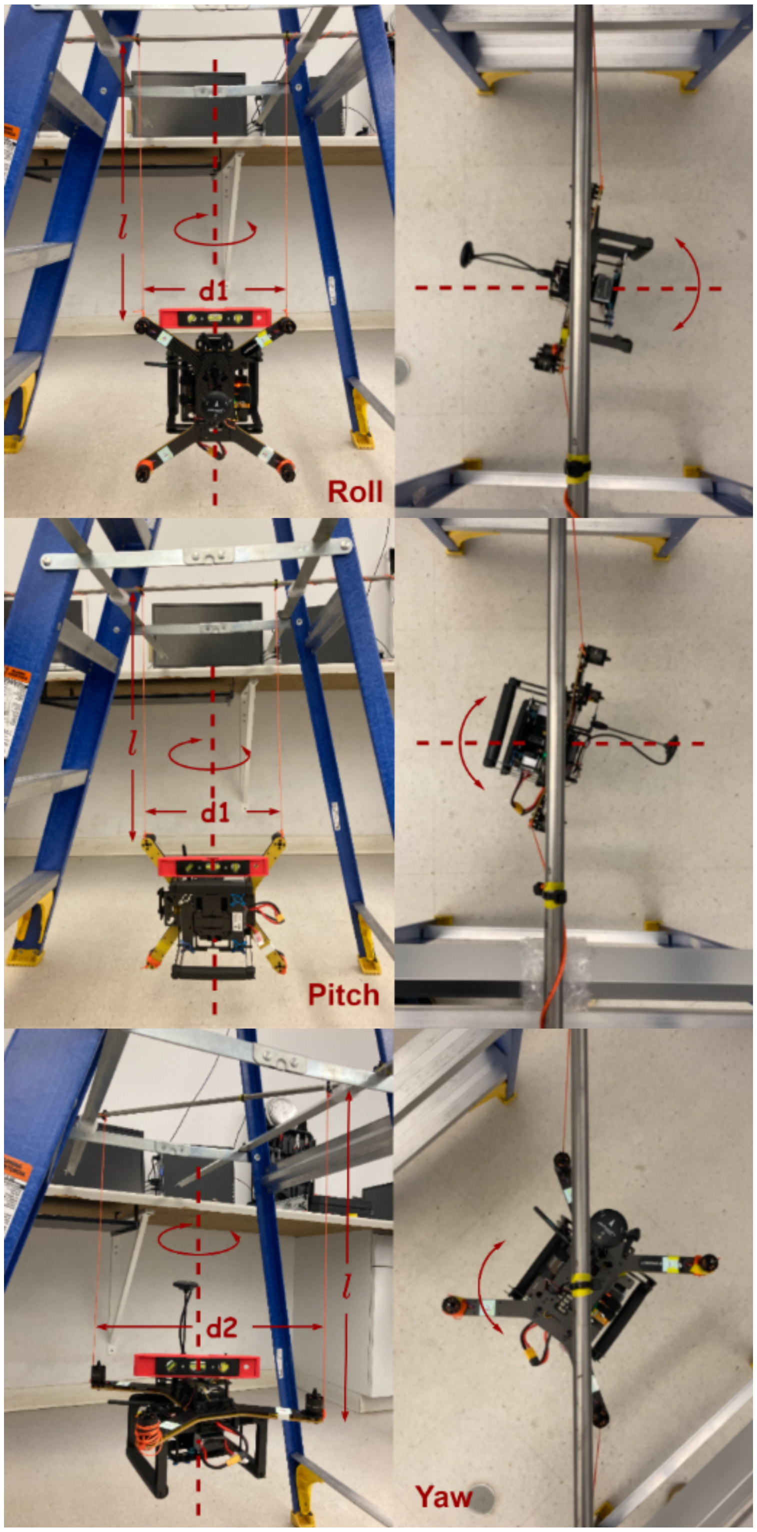


Figure 3.4: The experimental setup of bifilar pendulums for moments of inertia measurements

3.2.4 Other Parameters

From (3.4)-(3.7), the distances from the center mass of the quadrotor to the center of each motor are required to find the actual torque applied to the quadrotor. Assuming careful design, the center of gravity can be considered as the intersection point of the four arms. So we measured the distance between a pair of motors and divided it by two. The distance d is given by 0.200 m.

Finally, the model parameters involved in our quadrotor model are given in Table 3.3.

Parameter	Description	Value	Units
I_ϕ	Roll Inertia	0.0114	$kg \cdot m^2$
I_θ	Pitch Inertia	0.0129	$kg \cdot m^2$
I_ψ	Yaw Inertia	0.0178	$kg \cdot m^2$
b	Thrust coefficient	1.0757e-05	$N \cdot s^2/rad^2$
κ	Torque coefficient	1.6646e-07	$N \cdot m \cdot s^s/rad^2$
d	Distance	0.200	m
m	Mass	1.3913	kg
g	Gravity Constant	9.81	m/s^2

Table 3.3: Quadrotor system parameters

Chapter 4

Attitude Estimation

In the absence of direct attitude measurement, the development of a robust and reliable attitude estimator is key to a successful implementation of an efficient attitude control scheme. Theoretically, it is possible to estimate the attitude by integrating the rigid-body attitude kinematics using the angular velocity measurements or reconstructing algebraically the orientation using the inertial measurements (at least two noncollinear inertial vector measurements). However, in practice where the measurements are affected by noise, dynamic estimation algorithms relying on angular velocity and inertial vector measurements are used. These estimation algorithms usually rely on the inertial measurement unit (IMU), typically including a gyroscope, an accelerometer, and a magnetometer. The nonlinear attitude estimator proposed in [22] will be implemented to estimate the attitude of the quadrotor.

4.1 Attitude Estimation

4.1.1 Observer Design

Let us make the approximation that $a_{\mathcal{B}} \approx -gR^\top e_3$ for the case of hovering, and define the following vectors [22]:

$$\begin{aligned} u_{\mathcal{I}} &:= e_3, \qquad v_{\mathcal{I}} := \frac{\pi_{u_{\mathcal{I}}} m_{\mathcal{I}}}{\|\pi_{u_{\mathcal{I}}} m_{\mathcal{I}}\|} \\ u_{\mathcal{B}} &:= -\frac{a_{\mathcal{B}}}{g}, \quad v_{\mathcal{B}} := \frac{\pi_{u_{\mathcal{B}}} m_{\mathcal{B}}}{\|\pi_{u_{\mathcal{B}}} m_{\mathcal{B}}\|} \end{aligned} \tag{4.1}$$

where $\|x\|$ is the norm of vector x, and $\pi_x := \|x\|^2 I_3 - xx^\top, \forall x \in \mathbb{R}^3$ denotes the orthogonal projection on the plane orthogonal to x. It is obvious that $u_{\mathcal{B}} = R^\top u_{\mathcal{I}}$, and using the facts

$$\begin{aligned} R^\top \pi_{u_{\mathcal{I}}} m_{\mathcal{I}} &= R^\top (\|u_{\mathcal{I}}\|^2 I_3 - u_{\mathcal{I}} u_{\mathcal{I}}^\top) m_{\mathcal{I}} \\ &= (R^\top - R^\top u_{\mathcal{I}} u_{\mathcal{I}}^\top) R R^\top m_{\mathcal{I}} \\ &= (I_3 - u_{\mathcal{B}} u_{\mathcal{B}}^\top) m_{\mathcal{B}} \\ &= \pi_{u_{\mathcal{B}}} m_{\mathcal{B}} \end{aligned}$$

and

$$\|\pi_{u_\mathcal{B}} m_\mathcal{B}\|^2 = (\pi_{u_\mathcal{B}} m_\mathcal{B})^\top (\pi_{u_\mathcal{B}} m_\mathcal{B}) = (\pi_{u_\mathcal{I}} m_\mathcal{I})^\top R R^\top (\pi_{u_\mathcal{I}} m_\mathcal{I}) = \|\pi_{u_\mathcal{I}} m_\mathcal{I}\|^2$$

one has $v_\mathcal{B} = R^\top v_\mathcal{I}$. Define the estimates of the vectors $u_\mathcal{B}$ and $v_\mathcal{B}$ as

$$\hat{u}_\mathcal{B} := \hat{R}^\top u_\mathcal{I}, \quad \hat{v}_\mathcal{B} := \hat{R}^\top v_\mathcal{I} \tag{4.2}$$

where, $\hat{R}$ is the estimate of the actual attitude R, and the update laws are given by [22]:

$$\begin{aligned} \dot{\hat{R}} &= \hat{R}(\omega_\mathcal{B} - \hat{b} + \sigma_R)_\times && (4.3)\\ \dot{\hat{b}} &= -k_b \hat{b} + k_b \text{sat}_\Delta(\hat{b}) + \sigma_b && (4.4)\\ \sigma_R &:= k_1 u_\mathcal{B} \times \hat{u}_\mathcal{B} + k_2 \hat{u}_\mathcal{B} \hat{u}_\mathcal{B}^\top (v_\mathcal{B} \times \hat{v}_\mathcal{B}) && (4.5)\\ \sigma_b &:= -k_3 u_\mathcal{B} \times \hat{u}_\mathcal{B} - k_4 v_\mathcal{B} \times \hat{v}_\mathcal{B} && (4.6) \end{aligned}$$

where $\hat{b}$ is the estimate of the unknown bias b_g with $\|\hat{b}(0)\| < \Delta$ and Δ denoting positive constant, k_1, k_2, k_3, k_4 and k_b denote positive constants with $k_3 > k_4$, and the saturation function is defined by $\text{sat}_\Delta(x) := x \min(1, \Delta/\|x\|), \Delta > 0$. This estimator guarantees that, for almost all initial conditions, the trajectory of $(\hat{R}(t), \hat{b}(t))$ converges to the trajectory of $(R(t), b(t))$ asymptotically (see Theorem 1 in [22]).

4.1.2 Quaternion Representation

It is computationally expensive to compute the observer (4.3-4.6) mentioned above using rotation matrix representation since the rotation matrix has nine variables. However, as discussed earlier, the unit-quaternion representation presents some advantages with respect to the rotation matrix representation in terms of computational efficiency.

We can rewrite the observer given in (4.3-4.4) in terms of unit quaternion representation as follows:

$$\begin{aligned} \dot{\hat{Q}} &= \frac{1}{2} A(\hat{\Omega}) \hat{Q} && (4.7)\\ \dot{\hat{b}} &= -k_b \hat{b} + k_b \text{sat}_\Delta(\hat{b}) + \sigma_b && (4.8) \end{aligned}$$

where, $\hat{\Omega} = [\hat{\omega}_1, \hat{\omega}_2, \hat{\omega}_3]^\top := \omega_\mathcal{B} - \hat{b} + \sigma_R$, $\hat{Q} = [\hat{\eta} \quad \hat{q}]^\top$, and

$$A(\hat{\omega}) := \left[\begin{array}{cc} 0 & -\hat{\omega}^\top \\ \hat{\omega} & -\hat{\omega}_\times \end{array}\right] = \begin{bmatrix} 0 & -\hat{\omega}_1 & -\hat{\omega}_2 & -\hat{\omega}_3 \\ \hat{\omega}_1 & 0 & \hat{\omega}_3 & -\hat{\omega}_2 \\ \hat{\omega}_2 & -\hat{\omega}_3 & 0 & \hat{\omega}_1 \\ \hat{\omega}_3 & \hat{\omega}_2 & -\hat{\omega}_1 & 0 \end{bmatrix}$$

4.2 Sensors Calibration

The sensors used in our quadrotor, for example, the gyroscope, the accelerometer, and the magnetometer are prone to have biases in the measurements. Since the changes in these biases during one flight are negligible, this allows us to estimate them once at the beginning of a mission and then keep them constant. For the sensors' calibration, we assume that the noise in the measurements has zero mean. The details of the sensors' calibration are as follows:

- *Gyroscope Calibration:* The gyroscopes measure the angular velocity of the quadrotor relative to the inertial frame expressed in the body-fixed frame $\mathcal{B}$.

  $$\omega_{\mathcal{B}} = \omega + b_g + n_g \tag{4.9}$$

  where b_g is a constant (or slowly time-varying) gyro bias, n_g denotes the additive measurement noise, and ω is the actual angular velocity expressed in the body-fixed frame. We can average the gyroscope measurements over N samples at the beginning (before take-off *i.e.,* $\omega = 0$) to estimate the gyroscope measurement bias b_g, which is given by

  $$\hat{b}_g = \frac{1}{N}\sum_{k=1}^{N} \omega_{\mathcal{B}}(k)$$

- *Accelerometer Calibration:* The accelerometers measure the instantaneous linear acceleration of the quadrotor expressed in body-fixed frame $\mathcal{B}$.

  $$a_{\mathcal{B}} = R^{\top}(a - g e_3) + b_a + n_a \tag{4.10}$$

  where, R is the rotation matrix, b_a is a bias term, n_a denotes the additive measurement noise, and a is the derivative of the linear velocity expressed in the inertial frame. Similarly, the accelerometer measurements can also be averaged over N samples before take-off *i.e.,* $R = I_3$ given by

  $$\hat{b}_a = \frac{1}{N}\sum_{k=1}^{N} a_{\mathcal{B}}(k) + g$$

  When performing the sensor biases calibration, we sampled the IMU reading over a $6s$ period, which is enough to provide accurate estimates of the biases. The accelerometers and gyroscopes are highly susceptible to vibrations, therefore, they are mounted on foam tape or gel to reduce the noise in the sensor measurements. Moreover, a Digital Low Pass Filter (DLPF) has also been implemented to further clean the sensor signals.

- *Magnetometer Calibration:* The magnetometers provide measurements of the ambient magnetic field, which is defined by

  $$m_{\mathcal{B}} = D R^{\top} m_{\mathcal{I}} + b_m + n_m \tag{4.11}$$

  where, D is the distortion, $m_{\mathcal{I}}$ denotes the Earth's magnetic field vector (expressed in the inertial frame), b_m is a body-fixed frame expression for the local magnetic field, and n_m denotes the measurement noise. The noise n_m is usually low for magnetometer reading; however, the local magnetic field b_m can be significant, especially if the sensor is placed near the power wires and/or the motors.

The compensation approach proposed in [59] and [60] has been implemented in our quadrotor system. Define

$$D = \begin{bmatrix} \epsilon_x & 0 & 0 \\ \epsilon_y \sin\delta_x & \epsilon_y \cos\delta_x & 0 \\ \epsilon_z \sin\delta_y \cos\delta_z & \epsilon_z \sin\delta_z & \epsilon_z \cos\delta_y \cos\delta_z \end{bmatrix} \qquad b_m = \begin{bmatrix} \varrho_x \\ \varrho_y \\ \varrho_z \end{bmatrix}$$

with $(\epsilon_x, \epsilon_y, \epsilon_z)$ are the total scale errors, $(\delta_x, \delta_y, \delta_z)$ are the sensor misalignment angles, and $(\varrho_x, \varrho_y, \varrho_z)$ are the sensor offsets. In the absence of noise, one can solve the (4.11) for $R^\top m_\mathcal{I}$ as

$$R^\top m_\mathcal{I} = D^{-1}(m_\mathcal{B} - b_m) \tag{4.12}$$

Using the fact that $m_\mathcal{B} = [m_{\mathcal{B}_x}, m_{\mathcal{B}_y}, m_{\mathcal{B}_z}]^\top$, and taking the norm of (4.12) on both sides, one has

$$\begin{aligned} C_1 m_{\mathcal{B}_x}^2 + C_2 m_{\mathcal{B}_x} m_{\mathcal{B}_y} + C_3 m_{\mathcal{B}_x} m_{\mathcal{B}_z} + C_4 m_{\mathcal{B}_y}^2 + C_5 m_{\mathcal{B}_y} m_{\mathcal{B}_z} \\ + C_6 m_{\mathcal{B}_z}^2 + C_7 m_{\mathcal{B}_x} + C_8 m_{\mathcal{B}_y} + C_9 m_{\mathcal{B}_z} = C_{10} \end{aligned} \tag{4.13}$$

where, the coefficients C_k, $1 \le k \le 10$ are the functions of the 10 parameters $\epsilon_i, \delta_i, \varrho_i, i \in \{x, y, z\}$ and $\|m_\mathcal{I}\|$. The value of $\|m_\mathcal{I}\|$ can be found from the local magnetic field, and the best estimates of C_k in a least-square sense can be found by restructuring (4.13), putting it in matrix form with N samples

$$\mathbb{X}\mathbb{C} = \mathbb{W} \tag{4.14}$$

where

$$\mathbb{X} = \underbrace{\begin{bmatrix} m_{\mathcal{B}_{x1}}^2 & m_{\mathcal{B}_{x1}} m_{\mathcal{B}_{y1}} & \cdots & m_{\mathcal{B}_{z1}} \\ m_{\mathcal{B}_{x2}}^2 & m_{\mathcal{B}_{x2}} m_{\mathcal{B}_{y2}} & \cdots & m_{\mathcal{B}_{z2}} \\ \vdots & \vdots & \ddots & \vdots \\ m_{\mathcal{B}_{xN}}^2 & m_{\mathcal{B}_{xN}} m_{\mathcal{B}_{yN}} & \cdots & m_{\mathcal{B}_{zN}} \end{bmatrix}}_{N\times 9} \quad \mathbb{C} = \underbrace{\begin{bmatrix} C_1/C_{10} \\ C_2/C_{10} \\ \vdots \\ C_9/C_{10} \end{bmatrix}}_{9\times 1} \quad \mathbb{W} = \underbrace{\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}}_{N\times 1}$$

where the magnetometer data set $\{m_\mathcal{B}\}$ consisting of N data points (at least 1000 samples) are collected by rotating the quadrotor along all the axes and recorded them using the serial communication. Finally, $\mathbb{C}_{est}$, a least-squares best fit estimate for $\mathbb{C}$, can be generated as

$$\mathbb{C}_{est} = (\mathbb{X}^\top \mathbb{X})^{-1} \mathbb{X}^\top \mathbb{W} \tag{4.15}$$

Now the estimates of $\mathbb{C}$ have been found, solutions for $\epsilon_i, \delta_i, \varrho_i, i \in \{x, y, z\}$ can be obtained by solving nine nonlinear equations with nine unknowns. In this case, MATLAB is used to solve for the parameters from the measured data.

The nine parameters involved in our test are solved and given by

$$\epsilon = \begin{bmatrix} 0.7215 \\ 0.6998 \\ 0.8479 \end{bmatrix} \qquad \delta = \begin{bmatrix} -0.0438 \\ -0.0106 \\ 0.0065 \end{bmatrix} \qquad \varrho = \begin{bmatrix} -0.0033 \\ -0.0350 \\ 0.0393 \end{bmatrix} \tag{4.16}$$

Fig. 4.1 shows the calibration results. The calibrated 3-axis magnetometer measurements trace out a sphere centered at the origin since the magnitude of the magnetic vector is constant. The uncalibrated magnetometer measurements appear elliptical instead of spherical. With the estimated distortion D and local magnetic field measurements b_m, the magnetometer measurements can be corrected as follows:

$$\hat{m}_{\mathcal{B}} = D^{-1}(m_{\mathcal{B}} - b_m) \tag{4.17}$$

Note that the magnetometer measurements need to be calibrated again once the environment is changed.

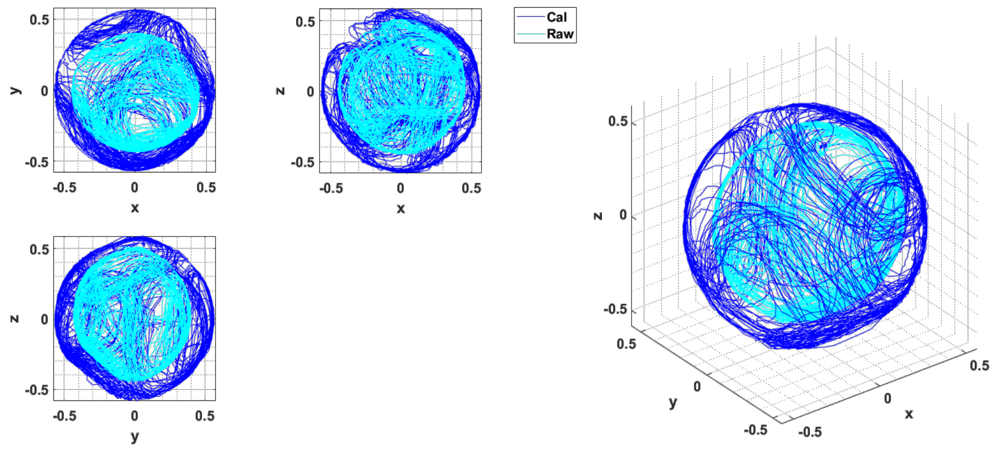


Figure 4.1: Magnetometer calibration results

4.3 Experimental Results

In order to explore the real-time performance of the attitude estimator (4.3 - 4.4) with unit quaternion representation (4.7), the experiment is performed on our quadrotor platform mentioned in Chapter 5. The implementation is carried out for the following scenario: an IMU is fixed to the center of a quadrotor UAV to measure the inertial vectors. The inertial vectors expressed in the inertial frame $\mathcal{I}$ are taken based on the local actual values as $a_{\mathcal{I}} = [0, 0, -9.81]^{\top}$ and $m_{\mathcal{I}} = [0.158177,\ -0.010346,\ 0.537342]^{\top}$.

The gains and parameters involved in the nonlinear observer are chosen as

$$k_1 = 0.8,\ k_2 = 0.3,\ k_3 = 0.025,\ k_4 = 0.01,\ k_b = 30,\ \Delta = 0.1 \tag{4.18}$$

The P-gain k_1 is chosen larger than k_2 due to the fact that the measurements of the gravity direction are more reliable than the measurements of the geomagnetic field. Small values of k_3 and k_4 are chosen in order to reduce the integral wind-up effects. A large value of k_b is chosen to obtain a fast desaturation rate of the bias estimation.

In this implementation, two discrete low-pass filters are applied respectively to reduce the noise for the accelerometer and magnetometer measurements with the cut-off frequencies at $40Hz$ and $10Hz$. The initial estimated Euler angles are chosen as $\hat{\phi}(0) = -30\,\text{deg},\ \hat{\theta}(0) =$

$-30\,\text{deg}$ and $\hat{\psi}(0) = -30\,\text{deg}$, and the actual angles are considered constants for all time, *i.e.*, $\phi(t) = 0\,\text{deg},\ \theta(t) = 0\,\text{deg},$ and $\psi(t) = 0\,\text{deg}$. The initial estimated gyro-bias is taken as $\hat{b}(0) = [-0.0015; 0.0007; -0.0002]^\top\ (\text{deg}/s)$.

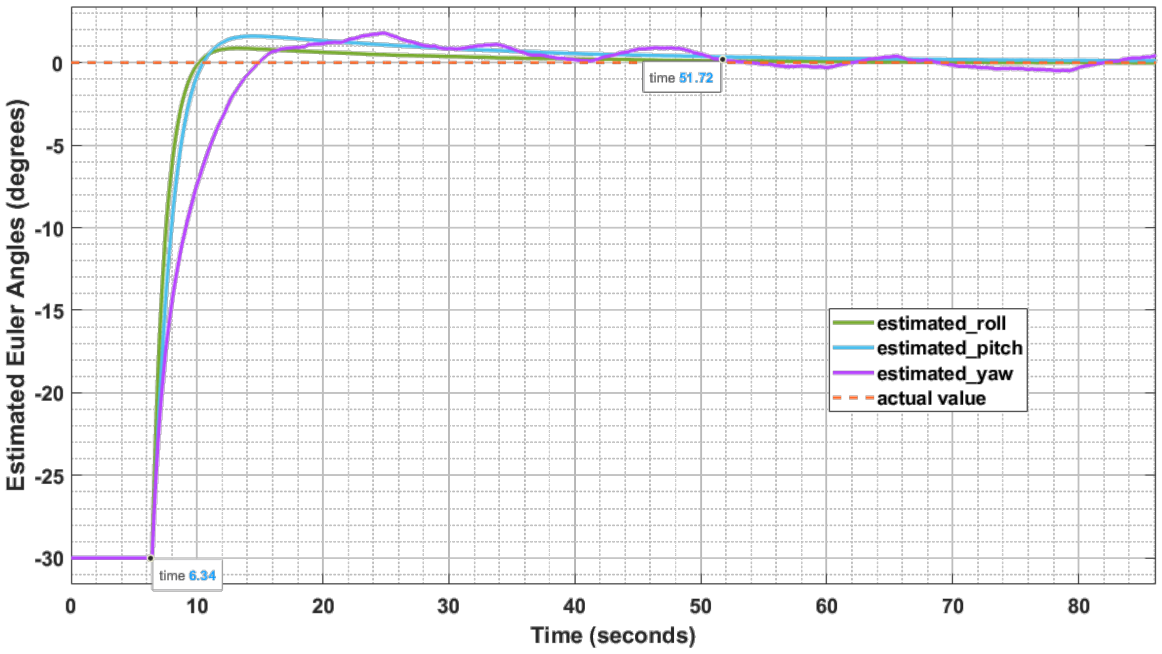


Figure 4.2: Real-time estimation for static attitude

The experimental results are reported in Fig. 4.2 which shows the performance of the nonlinear observer. The estimated attitude is converted into the roll, pitch, and yaw angles for visualization. One can see a very fast convergence of the estimated variables to the actual values, and the quasi-absence of overshoot of the estimated attitude despite the use of the large initial estimation errors. Regarding the estimation of the yaw angles in Fig. 4.2, the presence of visible estimation errors may be explained by the fact that the inertial magnetic field vector might be slightly perturbed by the electrical equipment in the lab. However, it can also be seen that the presence of magnetic disturbance does not degrade the estimation performance of the roll and pitch, as well as their convergence to the actual values.

Chapter 5

Experimental Platform

Autopilot is a well-known flight control system for drones and other unmanned vehicles. It consists of flight stack software running on vehicle controller (flight controller) hardware. PX4 [61], the autopilot flight stack, supports a professional open-source autopilot system, which powers a variety of vehicles and allows users to customize the algorithms to meet their own needs. Our experimental platform is a PX4-based quadrotor equipped with a Pixhawk 4 [62] flight controller and other electronic components. MATLAB/Simulink R2021a with the UAV Toolbox Support Package for PX4 Autopilots [63] is used to program the control algorithms. An overview of the platform architecture is shown in Fig. 5.1.

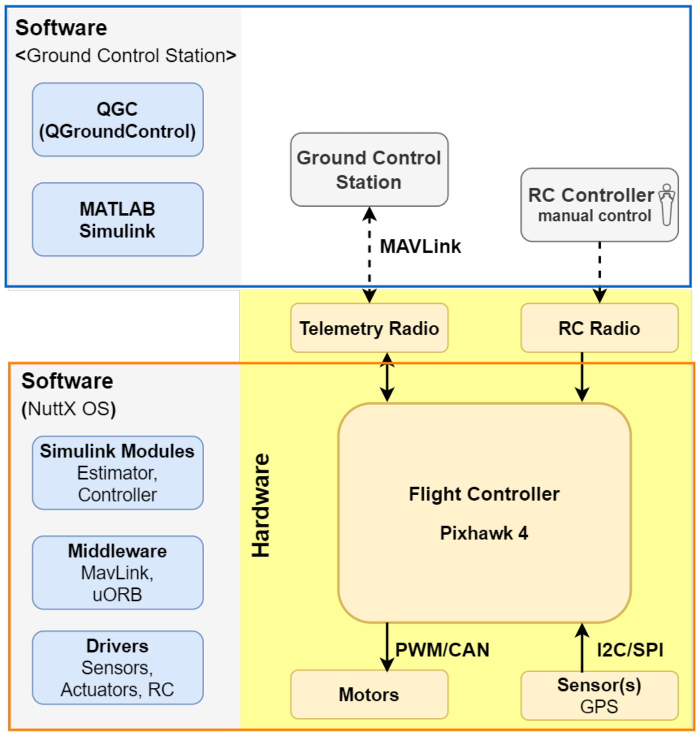


Figure 5.1: Platform architecture

The hardware consists of:

- Flight controller, running the PX4 flight stack, including microprocessors, internal IMUs, a compass, and a barometer.
- External GPS connected via I^2C/SPI.
- RC radio transmitter and receiver for manual control.
- Telemetry radios for communicating between the ground base station computer and the quadrotor UAV.
- ESC and motors connected to the PWM outputs.

The left-hand side of the diagram shows the software stack, horizontally aligned with the hardware parts.

- The ground station computer runs QGroundControl [64] and MATLAB/Simulink.
- The PX4 flight stack running on the flight controller includes Simulink modules (estimator and controllers), middleware, and drivers.

5.1 General Overview of the Hardware Platform

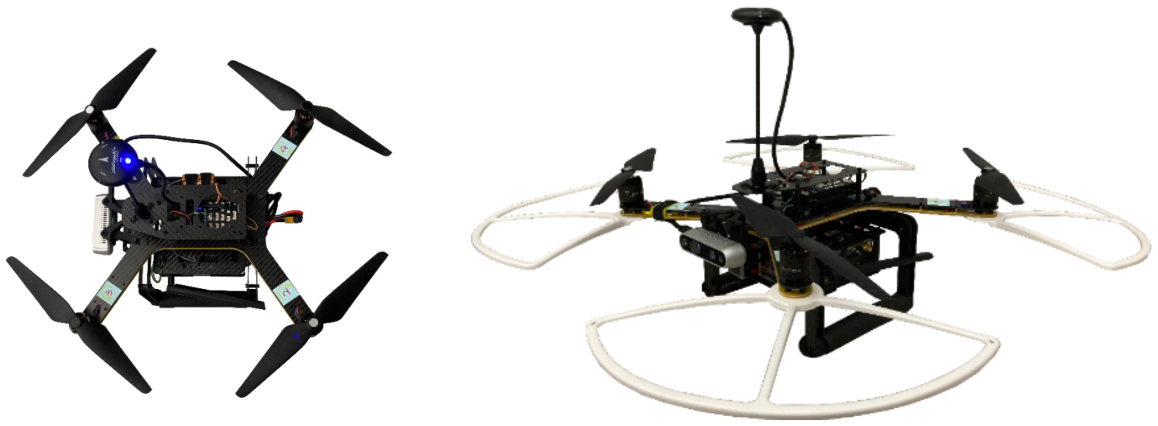


Figure 5.2: Hardware platform

As shown in Fig. 5.2, the quadrotor is a cross airframe type made with full lightweight and robust carbon fiber twill, which consists of 4 motors attached to the end of each arm, a GPS mast, a stereo-camera mount, a damping flight controller plate, top, and bottom carrier plates, and a square landing gear. The hardware includes a flight controller, a GPS, an onboard computer, a power management board (PM board), a 4-in-1 electronic speed controller (ESC), and other necessary components. All the parts and accessories are affordable, widely available, and easy to assemble and to be replaced. The remainder of this section will go through the individual components in detail. An overview of the experimental hardware is presented in Fig. 5.3.

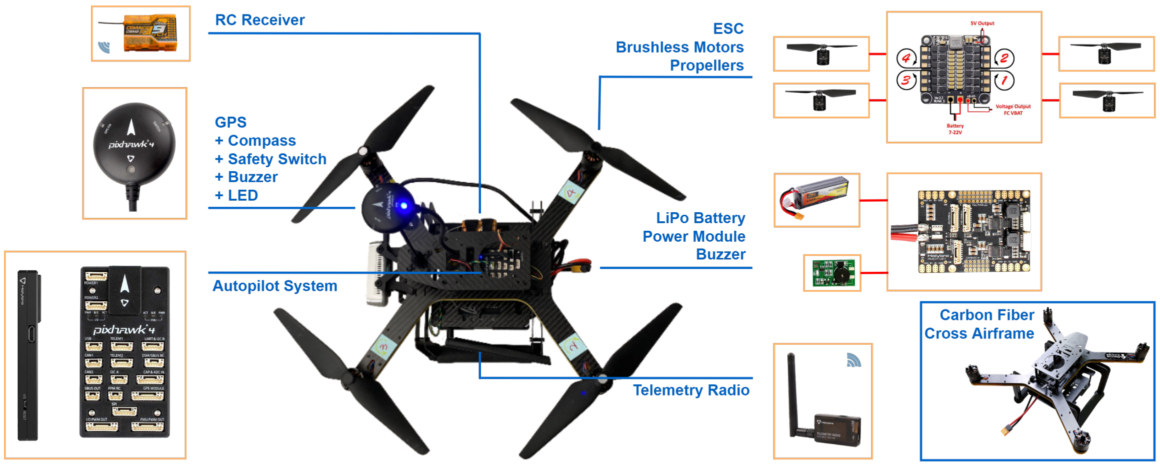


Figure 5.3: Overview of the quadrotor hardware

5.1.1 Flight Controller

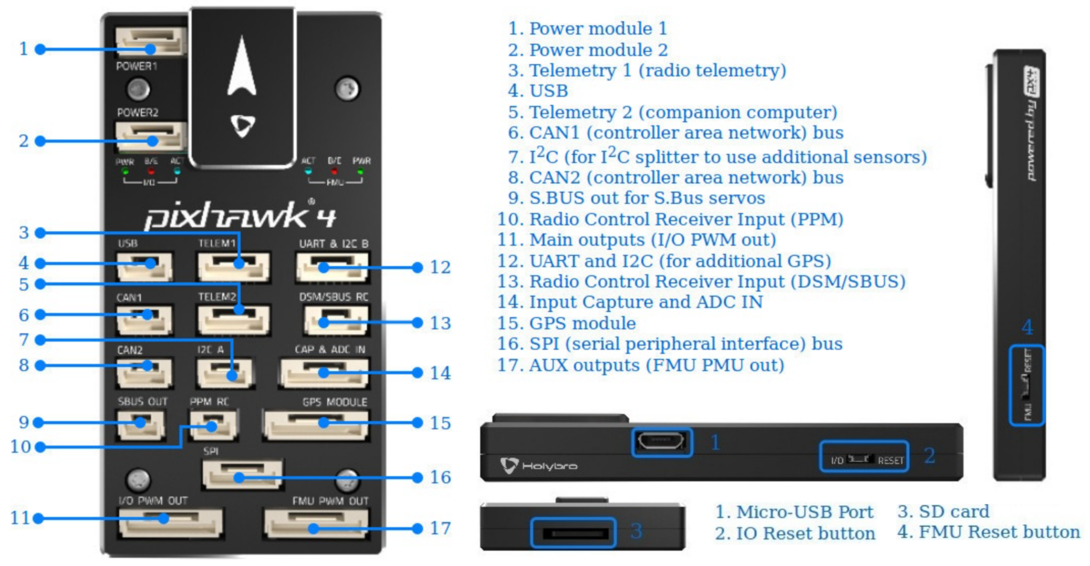

1. Power module 1
2. Power module 2
3. Telemetry 1 (radio telemetry)
4. USB
5. Telemetry 2 (companion computer)
6. CAN1 (controller area network) bus
7. I2C (for I2C splitter to use additional sensors)
8. CAN2 (controller area network) bus
9. S.BUS out for S.Bus servos
10. Radio Control Receiver Input (PPM)
11. Main outputs (I/O PWM out)
12. UART and I2C (for additional GPS)
13. Radio Control Receiver Input (DSM/SBUS)
14. Input Capture and ADC IN
15. GPS module
16. SPI (serial peripheral interface) bus
17. AUX outputs (FMU PMU out)
1. Micro-USB Port
2. IO Reset button
3. SD card
4. FMU Reset button
POWER1
POWER2
USB
TELEM1
UART & I2C B
CAN1
TELEM2
DSM/SBUS RC
CAN2
I2C A
CAP & ADC IN
SBUS OUT
PPM RC
GPS MODULE
SPI
I/O PWM OUT
FMU PWM OUT
pixhawk 4

Figure 5.4: Overview of the quadrotor hardware [1]

The Pixhawk 4® (shown in Fig. 5.4) is the flight controller used in our quadrotor UAV based on the Pixhawk FMUv5 Open Standard and the Pixhawk Autopilot Bus Standard. It is a low-cost system, features the latest advanced processor technology from STMicroelectronics®, sensor technology from Bosch® and InvenSense®, and a NuttX real-time operating system which makes the system ready-to-fly capable. Pixhawk 4's main FMU Processor has $2MB$ of Flash memory and $512KB$ of RAM running at up to $216MHz$, and its IO Processor is a 32-bit microprocessor running at up to $24MHz$. The FMUv5 open standard includes two high-performance, low-noise IMUs on board providing double redundancy. Further details are listed below:

[1] https://blog.csdn.net/SimuArch

- *Main FMU Processor: STM32F765* [2]

  32 Bit Arm® Cortex®-M7, 216MHz, 2MB memory, 512KB RAM

  The main FMU Processor STM32F765 is of very high-performance MCUs based on the 32-bit Arm® Cortex®-M7 core, with DPFPU, ART Accelerator™ and L1-cache: 16 Kbytes I/D cache, allowing 0-wait state execution from embedded Flash and external memories, running up to $216MHz$ while reaching low static power consumption. ST's $90nm$ process, ART Accelerator, and dynamic power scaling enable the power consumption in Run mode and executing from Flash memory to be at 7 $CoreMark/mW$ at $1.8V$. In Stop mode, power consumption is typically $100\mu A$. The graphics accelerator performs content creation twice as fast as the core alone. As well as efficient $2-D$ raw data copy, additional functions are supported by the Chrom-ART Accelerator such as image format conversion or image blending (image mixing with some transparency). As a result, the Chrom-ART Accelerator boosts graphics content creation and saves the processing bandwidth of the MCU core for the rest of the application. For more details, please refer to [65].

- *IO Processor: STM32F100* [3]

  32 Bit Arm® Cortex®-M3, 24MHz, 8KB SRAM

  The STM32F100 IO processor incorporates the high-performance ARM® Cortex®-M3 32-bit RISC core operating at a $24MHz$ frequency, high-speed embedded memories (flash memory up to $128Kbytes$ and SRAM up to $8Kbytes$), and an extensive range of enhanced peripherals and I/Os connected to two APB buses. It offers standard communication interfaces (up to two I^{2}Cs, two SPIs, one HDMI CEC, and up to three USARTs), one 12-bit ADC, two 12-bit DACs, up to six general-purpose 16-bit timers, and an advanced-control PWM timer. It operates in the –40 to 85℃ and –40 to 105℃ temperature ranges, from a 2.0 to $3.6V$ power supply. A comprehensive set of power-saving mode allows the design of low-power applications. These features make these microcontrollers suitable for a wide range of applications. Further details are provided in [66].

- *Accel/Gyro: ICM-20689*

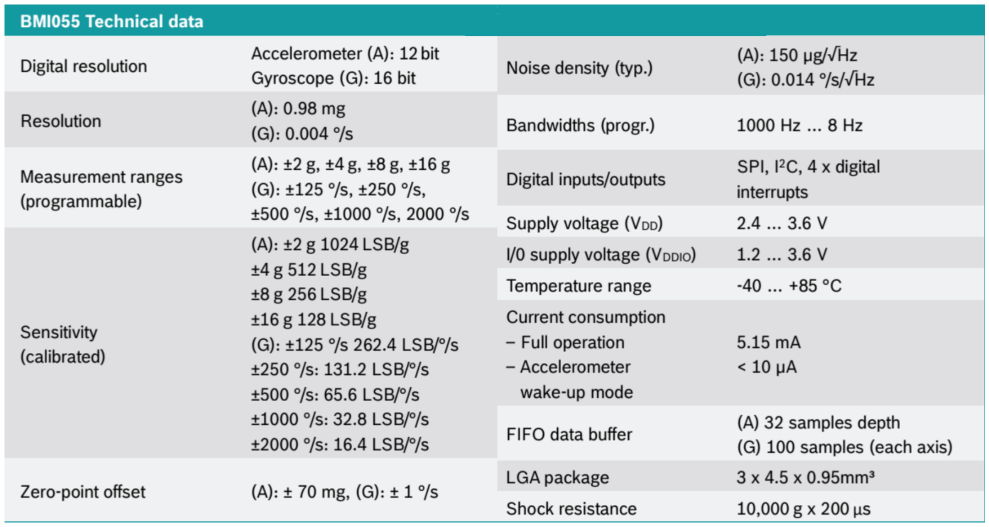
  The ICM-20689 is a device that combines a 3-axis gyroscope, a 3-axis accelerometer, and a Digital Motion Processor™ (DMP) in a small 4x4x0.9 mm (24-pin QFN) package. It also features a 4 Kbyte FIFO that can lower the traffic on the serial bus interface, and reduce power consumption by allowing the system processor to burst read sensor data and then go into a low-power mode. ICM-20689, with its 6-axis integration, on-chip DMP, and run-time calibration firmware, enables manufacturers to eliminate the costly and complex selection, qualification, and system-level integration of discrete devices, guaranteeing optimal motion performance.

  The gyroscope has a programmable full-scale range of ±250, ±500, ±1000, and ±2000 degrees/sec. The accelerometer has a userprogrammable accelerometer full-scale range of ±2g, ±4g, ±8g, and ±16g. Factory-calibrated initial sensitivity of both sensors reduces production-line calibration requirements. More details can be found in [67].

[2]https://www.st.com/en/microcontrollers-microprocessors/stm32f7x5.html
[3]https://www.st.com/en/microcontrollers-microprocessors/stm32f100rb.html

- *Accel/Gyro: BMI055*

  The BMI055 is an ultra-small, 6-axis inertial measurement unit (IMU) consisting of a digital, tri-axial 12-bit acceleration sensor and a triaxial 16-bit gyroscope. This sensor is unique in the class of high-stability IMUs measuring angular rates and accelerations in three perpendicular axes. This inertial sensor integrates a multitude of features that especially facilitate motion detection applications such as device orientation measurement, gaming, HMI, and menu browser control. See Fig. 5.5 and [68] for more details.



BMI055 Technical data			
Digital resolution	Accelerometer (A): 12 bit Gyroscope (G): 16 bit	Noise density (typ.)	(A): 150 µg/√Hz (G): 0.014 °/s/√Hz
Resolution	(A): 0.98 mg (G): 0.004 °/s	Bandwidths (progr.)	1000 Hz ... 8 Hz
Measurement ranges (programmable)	(A): ±2 g, ±4 g, ±8 g, ±16 g (G): ±125 °/s, ±250 °/s, ±500 °/s, ±1000 °/s, 2000 °/s	Digital inputs/outputs	SPI, I²C, 4 x digital interrupts
Sensitivity (calibrated)	(A): ±2 g 1024 LSB/g ±4 g 512 LSB/g ±8 g 256 LSB/g ±16 g 128 LSB/g (G): ±125 °/s 262.4 LSB/°/s ±250 °/s: 131.2 LSB/°/s ±500 °/s: 65.6 LSB/°/s ±1000 °/s: 32.8 LSB/°/s ±2000 °/s: 16.4 LSB/°/s	Supply voltage (V_{DD})	2.4 ... 3.6 V
		I/0 supply voltage (V_{DDIO})	1.2 ... 3.6 V
		Temperature range	-40 ... +85 °C
		Current consumption – Full operation – Accelerometer wake-up mode	 5.15 mA < 10 µA
		FIFO data buffer	(A) 32 samples depth (G) 100 samples (each axis)
Zero-point offset	(A): ± 70 mg, (G): ± 1 °/s	LGA package	3 x 4.5 x 0.95mm³
		Shock resistance	10,000 g x 200 µs

Figure 5.5: BMI055 technical data [4]

- *Mag: IST8310*

  iSentek IST8310 is a 3-axis digital magnetometer with $3.0 \times 3.0 \times 1.0$ mm^3, 16-pin LGA package. It is an integrated chip with 3-axis magnetic sensors, digital control logic, a built-in temperature compensation circuit, and a self-test function. IST8310 provides an I^{2}C digital output with fast mode up to $400kHz$. The high output data rate, ultra-low hysteresis, excellent temperature drift, and low noise performance feathers make it a perfect candidate for high-accuracy applications. See [69] for more details.

- *Holybro M8N GPS Module*

  GPS + Compass + Safety Switch + Buzzer + LED

  The M8N GPS, shown in Fig. 5.6, has a UBLOX M8N module, IST8310 compass, tri-colored LED indicator which indicates the current readiness for the flight status of our vehicle [61], and a safety switch which makes the connection more convenient and simple. There are 3 different connector options for different purposes. More details can be found in [70].

[4]https://www.bosch-sensortec.com/media/boschsensortec/downloads/product_flyer/bst-bmi055-fl000.pdf

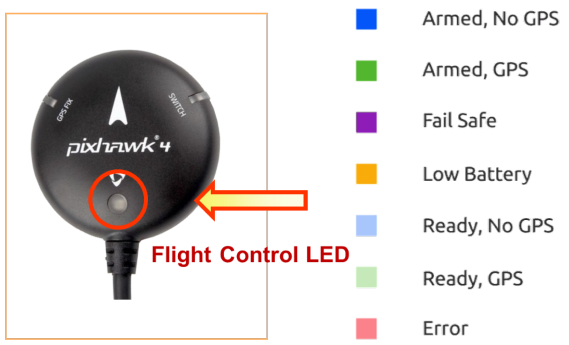


Figure 5.6: M8N GPS with the tri-colored LED indicator

5.1.2 Communication

RC Channels

The RC radio transmitter (Turnigy 9XR [5]) and the RC radio receiver (ORX R910 DSM2 receiver) set work in pairs, which are used for radio communication in our quadrotor platform as shown in Fig. 5.7. It is a 2.4 GHz system with 8 channels and 5 channels that have been used in our quadrotor system for sending the commands (PPM signals) of the roll, pitch, yaw, throttle, and mode switch.

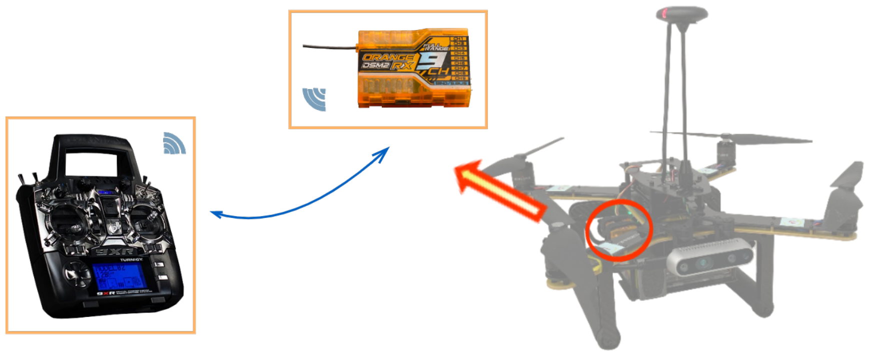


Figure 5.7: Turnigy 9XR transmitter (left) and ORX R910 DSM2 receiver (right)

The throttle stick does not return to the middle position when the finger is removed, while the roll and pitch control sticks do return to the center. The PPM signals from the transmitter vary while changing the position of the control sticks having a value between 900-2200. The maximum and minimum throttle values vary slightly and must be recorded in order to properly generate the desired thrust. The PPM values of the pitch and yaw at the center are set as zero angles since the pitch and roll sticks can return to the center automatically. In order to properly generate the desired roll and pitch angles, the PPM values at the center must be recorded. The yaw command is a little different from the roll/pitch. The position of the yaw stick provides the yaw rates.

[5] https://www.rcworld.co.za/downloads/T9XR.pdf

- *RC Channels Calibration*

  The RC transmitter/receiver, shown in Fig. 5.7, requires some simple calibration to ensure proper operation. The calibration of RC channels is a straightforward process – simply move each of the enabled sticks through their full range and record the maximum, minimum, and trim values for each RC channel. By moving the control sticks to their limits of travel, one has the channel parameters presented in Table 5.1.

Channel	Minimum	Trim	Maximum	Function
1	955	1500	2045	Pitch
2	968	1513	2062	Roll
3	971	971	2062	Throttle
4	968	1516	2062	Yaw

Table 5.1: RC channels parameters

The desired thrust of the quadrotor at the period k can be generated as

$$\mathcal{T}_d(k) = 4b\left(\frac{P_{\mathcal{T},r}(k) - P_{\mathcal{T},min}}{P_{\mathcal{T},max} - P_{\mathcal{T},min}}\varpi_{\max}\right)^2 \tag{5.1}$$

where, b is the thrust coefficient defined in Section 3.2.2, $P_{\mathcal{T},min}$ and $P_{\mathcal{T},max}$ are the minimum and maximum throttle values respectively, $P_{\mathcal{T},r}$ is the throttle PPM value received from the transmitter, and $\varpi_{\max}$ is the maximum angular velocity of the motor defined in Section 5.1.3.

The desired roll, pitch, and yaw angles at the period k are given by

$$\phi_d(k) = \frac{P_{\phi,r}(k) - P_{\phi,trim}}{S_f} \tag{5.2}$$

$$\theta_d(k) = \frac{P_{\theta,r}(k) - P_{\theta,trim}}{S_f} \tag{5.3}$$

$$\psi_d(k) = \psi_d(k-1) + \frac{P_{\psi,r}(k) - P_{\psi,trim}}{S_y} \tag{5.4}$$

where, $P_{\phi,r}$, $P_{\theta,r}$ and $P_{\psi,r}$ denote the PPM values of the roll, pitch, and yaw commands respectively, $P_{\phi,trim}$, $P_{\theta,trim}$ and $P_{\psi,trim}$ are the recorded trim PPM values of the roll, pitch, and yaw respectively. The parameter S_f denotes the stick factor for the roll and pitch converting the PPM values to Euler angles. The choice of this stick factor is 1/20 providing a maximum desired angle of about 25 degrees for the roll and pitch. The parameter S_y determines the sensitivity of the yaw rate, which is set as 1000.

Electronic speed controllers are responsible for spinning the motors at the speed requested by the autopilot. ESCs calibration is also important since the ESCs need to know the minimum and maximum PWM values that the flight controller will send. A complete process of ESC calibration using the ground station (QGC) can be found in [71]. After successfully calibrating, the mapping from the motor speed to the PWM value of motor i is given by

$$P_i = P_{\mathcal{T},min} + \frac{\varpi_i}{\varpi_{max}} * (P_{\mathcal{T},max} - P_{\mathcal{T},min}) \tag{5.5}$$

where, ϖ_i denotes the speed of motor i and ϖ_{max} denotes the maximum motor speed, and $P_{\mathcal{T},max}$ and $P_{\mathcal{T},min}$ are the maximum and minimum throttle PWM values.

Telemetry Radio

The wireless serial connectivity is provided to make communication between the ground base station computer and the quadrotor UAV. The Holybro SiK Telemetry Radio serial communication system operates at 915MHz With one module connected to Pixhawk to send data and the other module connected to the computer (see Fig. 5.8). Experimental data, for example, the sensor measurements, are sent to the ground station for analysis. The baud rate of the serial communication has been set as 57600bps in our quadrotor system.

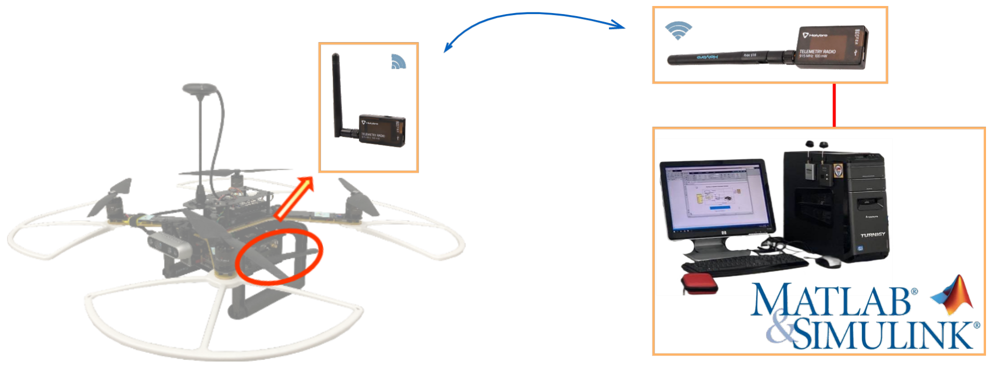


Figure 5.8: Radio serial 1 module (left) and radio USB module (right)

5.1.3 Power Module and Actuators

The entire platform is powered by a 4200mAh 4-cell lithium polymer battery. The battery voltage when fully charged is 16.8V, and 14.8V when discharged. The brushless motor and 4-in-1 electronic speed controller (ESC), as shown in Fig. 5.9, serve as actuators for the quadrotor.

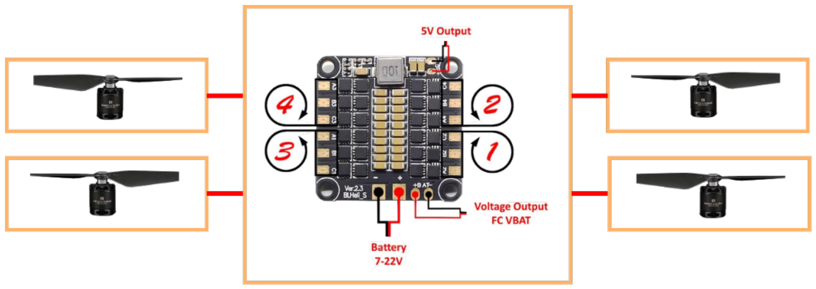
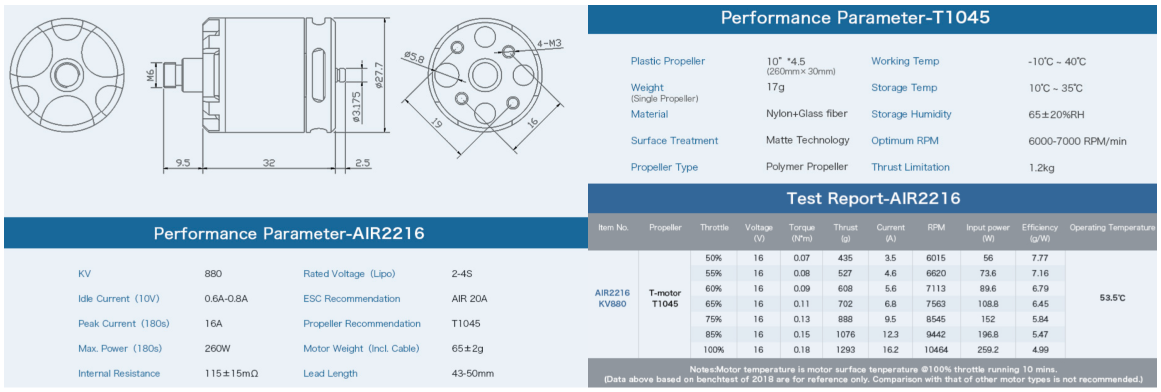


Figure 5.9: BLHeli 4-in-1 ESC and T-motor brushless motors and propellers

The 4-in-1 ESC is driven by the PWM signals of the desired motor speeds sent from the Pixhawk4. The ESC converts the DC voltage provided by the battery to a 3-phase AC current to drive the motor to work at the desired speed. Since the speed of the motor is controlled by the ESC, feedback from the motor is embedded. Earlier speed controllers employed Hall effect sensors but more recent ones measure the back-EMF generated in the un-driven coils. The motor (AIR2216) used in our quadrotor is an outrunner brushless motor. Its specifications provided by the manufacturer are given in Fig. 5.10.


M6
ø27.7
ø3.175
ø5.8
4-M3
19
16
9.5
32
2.5

Performance Parameter-T1045

Plastic Propeller	10" *4.5 (260mm× 30mm)	Working Temp	-10℃ ~ 40℃
Weight (Single Propeller)	17g	Storage Temp	10℃ ~ 35℃
Material	Nylon+Glass fiber	Storage Humidity	65±20%RH
Surface Treatment	Matte Technology	Optimum RPM	6000-7000 RPM/min
Propeller Type	Polymer Propeller	Thrust Limitation	1.2kg

Performance Parameter-AIR2216

KV	880	Rated Voltage (Lipo)	2-4S
Idle Current (10V)	0.6A-0.8A	ESC Recommendation	AIR 20A
Peak Current (180s)	16A	Propeller Recommendation	T1045
Max. Power (180s)	260W	Motor Weight (Incl. Cable)	65±2g
Internal Resistance	115±15mΩ	Lead Length	43-50mm

Test Report-AIR2216

Item No.	Propeller	Throttle	Voltage (V)	Torque (N*m)	Thrust (g)	Current (A)	RPM	Input power (W)	Efficiency (g/W)	Operating Temperature
AIR2216 KV880	T-motor T1045	50%	16	0.07	435	3.5	6015	56	7.77	53.5℃
		55%	16	0.08	527	4.6	6620	73.6	7.16	
		60%	16	0.09	608	5.6	7113	89.6	6.79	
		65%	16	0.11	702	6.8	7563	108.8	6.45	
		75%	16	0.13	888	9.5	8545	152	5.84	
		85%	16	0.15	1076	12.3	9442	196.8	5.47	
		100%	16	0.18	1293	16.2	10464	259.2	4.99	

Notes:Motor temperature is motor surface tenperature @100% throttle running 10 mins.
(Data above based on benchtest of 2018 are for reference only. Comparison with that of other motor types is not recommended.)

Figure 5.10: Motor and propellers performance report

5.2 Overview of the Software Configuration

PX4, supported by the Dronecode Project [72], is a core part of a broader drone platform that includes the ground control station, Pixhawk hardware, and MAVSDK for integration with companion computers (e.g. NVIDIA Jetson NANO), cameras, and other hardware using the MAVLink [73] protocol. PX4 consists of two main layers: the flight stack (shown in Fig. 5.11) is an estimation and flight control system, and the middleware is a general robotics layer that can support any type of autonomous robot, providing internal/external communications and hardware integration. PX4 middleware includes PX4 internal communication mechanisms (uORB [74]), and between PX4 and offboard systems like the ground control station (e.g. MAVLink). For more details, please refer to [75] written by the founder of PX4.

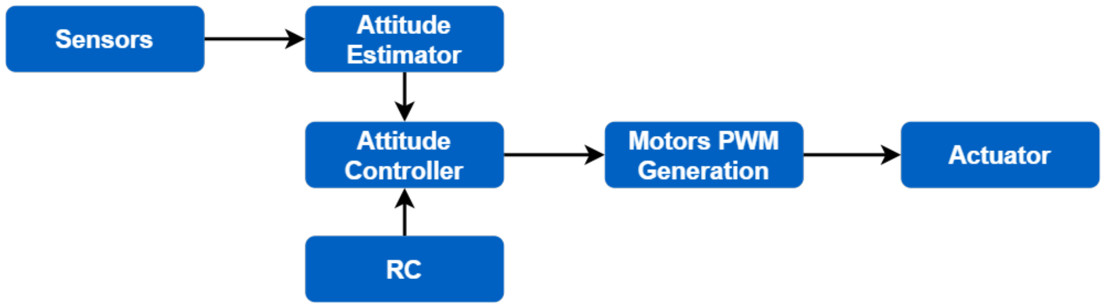
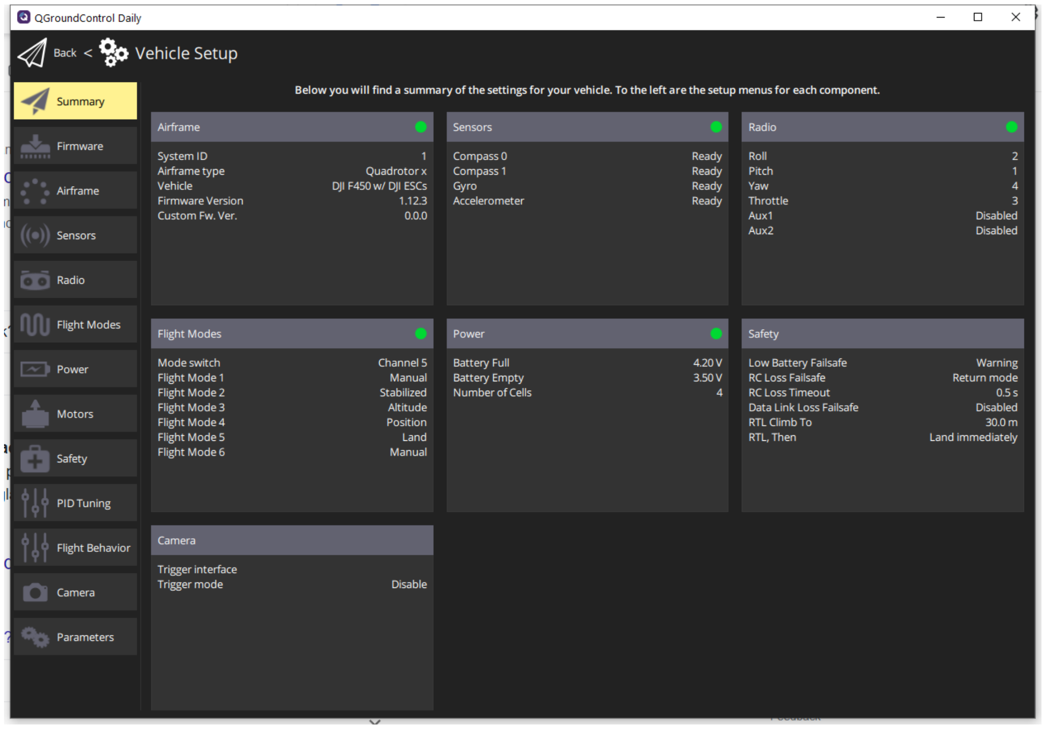


Figure 5.11: Block diagram of the attitude control scheme

The ground control station (GCS) is a very important part of the entire UAV system, which is a channel for ground operators to directly interact with the UAV. It integrates control, communication, real-time monitoring, and data processing capabilities, which makes it the command and control center for the entire UAV system.

5.2.1 QGroundControl

The Dronecode ground control station is called QGroundControl (QGC) which provides full flight control and vehicle setup for PX4-powered vehicles. It can be used to load (flash) the PX4 with the flight controller, set up the vehicle, change or upload different parameters, get real-time flight information, and create and execute fully autonomous missions. The vehicle setup page is shown in Fig. 5.12. Further details are provided in [64].



Below you will find a summary of the settings for your vehicle. To the left are the setup menus for each component.

Airframe	
System ID	1
Airframe type	Quadrotor x
Vehicle	DJI F450 w/ DJI ESCs
Firmware Version	1.12.3
Custom Fw. Ver.	0.0.0

Sensors	
Compass 0	Ready
Compass 1	Ready
Gyro	Ready
Accelerometer	Ready

Radio	
Roll	2
Pitch	1
Yaw	4
Throttle	3
Aux1	Disabled
Aux2	Disabled

Flight Modes	
Mode switch	Channel 5
Flight Mode 1	Manual
Flight Mode 2	Stabilized
Flight Mode 3	Altitude
Flight Mode 4	Position
Flight Mode 5	Land
Flight Mode 6	Manual

Power	
Battery Full	4.20 V
Battery Empty	3.50 V
Number of Cells	4

Safety	
Low Battery Failsafe	Warning
RC Loss Failsafe	Return mode
RC Loss Timeout	0.5 s
Data Link Loss Failsafe	Disabled
RTL Climb To	30.0 m
RTL, Then	Land immediately

Camera	
Trigger interface	
Trigger mode	Disable

Figure 5.12: QGC vehicle setup summary page

Our preliminary testing has been done using QGC by following the subsequent sequence of uploading firmware to the Pixhawk board, choosing the vehicle configuration (airframe), calibrating onboard sensors and power system (the battery and ESC), establishing a connection between the radio transmitter and receiver, and modifying the parameters (e.g. by modifying the IMU_ACCEL_CUTOFF to change the cutoff frequency of the internal accelerometer). QGC provides a fundamental understanding of the concepts used to develop the UAV flight control system, which gives us the benefits of getting familiar with the whole system within a short time, as well as verifying our hardware and all components are working properly and in good condition. This is an important step before moving on to the development and implementation of the controllers by using Simulink.

5.2.2 UAV Toolbox Support Package for PX4 Autopilots

MATLAB® and Simulink® can also be used as the ground control station. With the UAV Toolbox Support Package for PX4® Autopilots [6], users are enabled to design estimators, controllers, and navigators in Simulink, and deploy to PX4 Autopilot boards. The Embedded Coder® enables Simulink models to be automatically generated into C++ code, and by using the PX4 toolchain, algorithms are built and deployed tailored for Pixhawk flight controllers, all while incorporating onboard sensor data and other PX4-specific services.

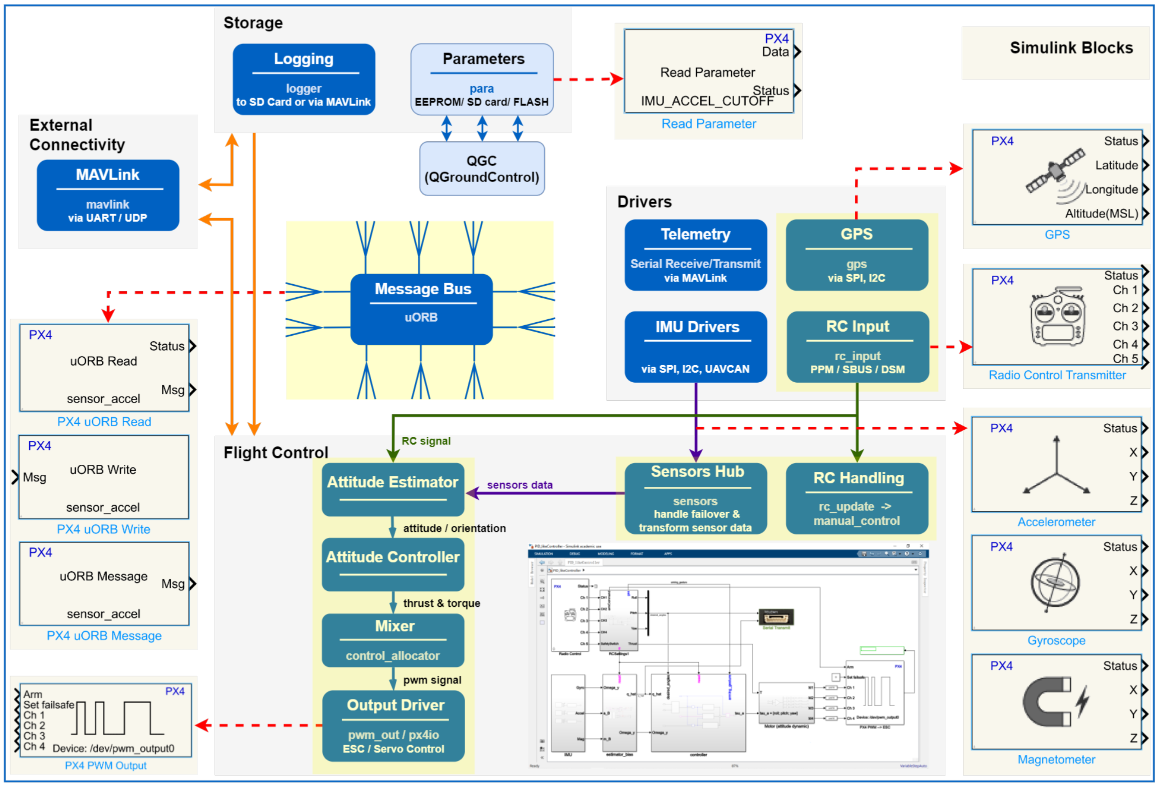


Figure 5.13: Supported Simulink blocks interface with the general PX4 architecture[7] modules

The support package provides interfaces for some of the components in the PX4 architecture by using Simulink blocks. These blocks can be used as input and output for the algorithms in the Simulink model. As shown in Fig. 5.13, the high-level software architecture of PX4 includes modules for storage, external connectivity, drivers, uORB publish-subscribe message bus, and flight control components. Modules in the general PX4 architecture can be replaced with user-defined Controller algorithms by using UAV Toolbox Support Package for PX4 Autopilots, presented in Table 5.2. For more details, please refer to [63].

[6] https://www.mathworks.com/help/supportpkg/px4
[7] https://docs.px4.io/main/en/concept/architecture.html

Component in PX4 Architecture	Simulink Block in the Support Package
Parameters	Read Parameter
uORB Message Bus	PX4 uORB Read PX4 uORB Write PX4 uORB Message
RC Input	Radio Control Transmitter
Sensors Hub	Accelerometer Gyroscope Magnetometer
GPS	GPS
Output Driver	PX4 PWM Output
External Serial Communication	Serial Receive Serial Transmit

Table 5.2: Supported Simulink blocks interface with the PX4 modules

MAVLink

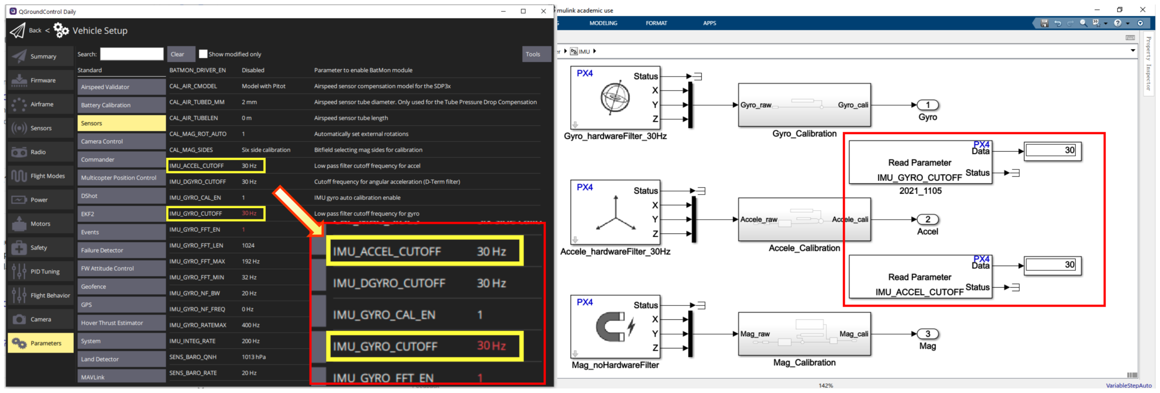
MAVLink, the Micro Aerial Vehicle link, is a very lightweight messaging protocol that has been designed for the drone ecosystem. PX4 uses MAVLink to communicate with ground control stations, and as the integration mechanism for connecting to drone components outside of the flight controller: companion computers, MAVLink-enabled cameras, etc. The protocol defines a number of standard messages and microservices for exchanging data [76]. MAVLink developer guide is provided in [73], and the MAVLink messaging tutorial can be found in [76].

- Enable MAVLink Protocol to Tune Sensor Parameters in QGC

  PX4 behaviour can be configured or tuned by parameters, such as the low-pass filter cutoff frequency for the accelerometer and gyroscope. The QGC Parameters screen allows users to find and modify any of the parameters associated with the vehicle [77]. QGC communicates over MAVLink to the PX4 Autopilot, by enabling MAVLink protocol, parameters can be read from Simulink and be modified and updated from QGC. We tuned the low-pass filter cutoff frequency for the accelerometer and gyroscope, and achieved the best sensor performance at $30Hz$, then verified the values in Simulink, which is shown in Fig. 5.14. In QGC Parameters list, the modified values are red-colored while the default ones are white-colored.

- External Serial Communication via MAVLink

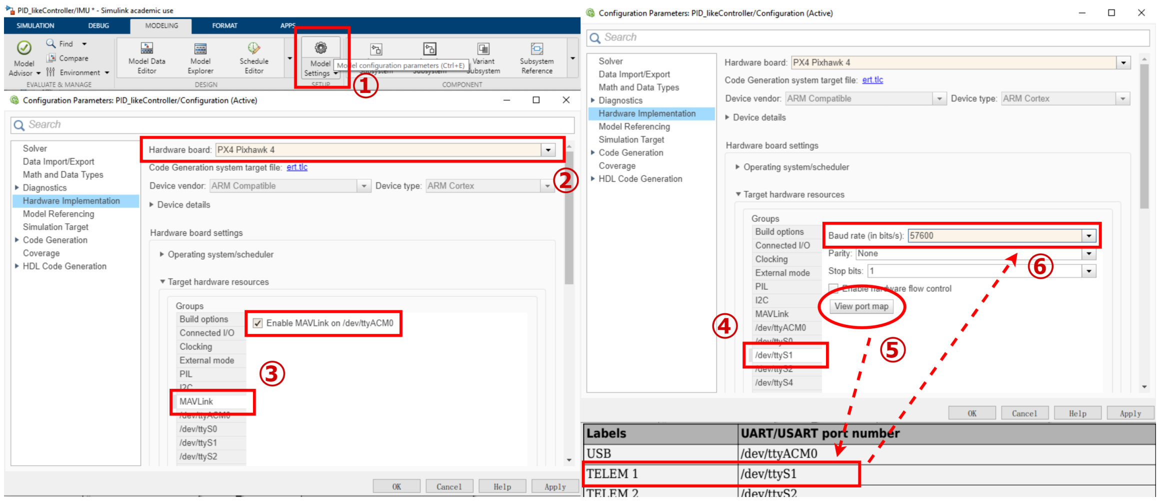
  Telemetry Radios via MAVLink connection can provide wireless serial communication between Simulink and the quadrotor UAV. This makes it possible to inspect telemetry in real-time and log the detailed sensor data while a drone is in flight, then analyze the performance and make improvements for the next flight. Two Simulink programs, a serial transmit program, and a serial receive program, are working in pairs to realize this wireless communication. To enable the settings required for using the related



BATMON_DRIVER_EN	Disabled	Parameter to enable BatMon module
CAL_AIR_CMODEL	Model with Pitot	Airspeed sensor compensation model for the SDP3x
CAL_AIR_TUBED_MM	2 mm	Airspeed sensor tube diameter. Only used for the Tube Pressure Drop Compensation
CAL_AIR_TUBELEN	0 m	Airspeed sensor tube length
CAL_MAG_ROT_AUTO	1	Automatically set external rotations
CAL_MAG_SIDES	Six side calibration	Bitfield selecting mag sides for calibration
IMU_ACCEL_CUTOFF	30 Hz	Low pass filter cutoff frequency for accel
IMU_DGYRO_CUTOFF	30 Hz	Cutoff frequency for angular acceleration (D-Term filter)
IMU_GYRO_CAL_EN	1	IMU gyro auto calibration enable
IMU_GYRO_CUTOFF	30 Hz	Low pass filter cutoff frequency for gyro
IMU_GYRO_FFT_EN	1	
IMU_GYRO_FFT_LEN	1024	
IMU_GYRO_FFT_MAX	192 Hz	
IMU_GYRO_FFT_MIN	32 Hz	
IMU_GYRO_NF_BW	20 Hz	
IMU_GYRO_NF_FREQ	0 Hz	
IMU_GYRO_RATEMAX	400 Hz	
IMU_INTEG_RATE	200 Hz	
SENS_BARO_QNH	1013 hPa	
SENS_BARO_RATE	20 Hz	

Figure 5.14: Low-pass filter cutoff frequency tuning (left) and verification (right)

blocks, some model configurations need to be done by following these steps as shown in Fig. 5.15:

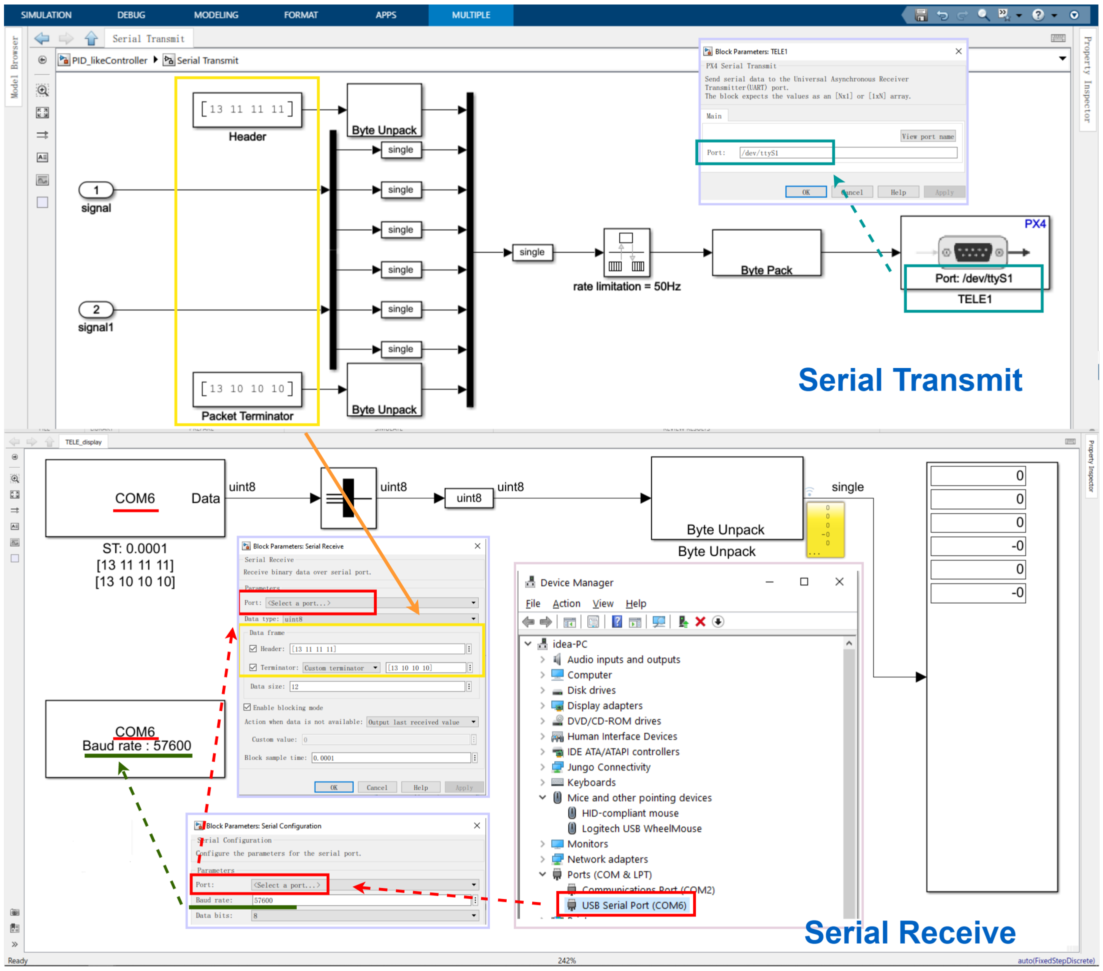


Hardware board: PX4 Pixhawk 4
Code Generation system target file: ert.tlc
Device vendor: ARM Compatible
Device type: ARM Cortex
Enable MAVLink on /dev/ttyACM0
Baud rate (in bits/s): 57600
Parity: None
Stop bits: 1

Labels	UART/USART port number
USB	/dev/ttyACM0
TELEM 1	/dev/ttyS1
TELEM 2	/dev/ttyS2

Figure 5.15: Configure the model for Pixhawk hardware

1. In the **Modeling** tab, click **Model Settings**.
2. In the **Configuration Parameters** dialog box, navigate to the **Hardware Implementation** pane, and set the **Hardware board** to **Pixhawk 4**.
3. In the **Target Hardware Resources** section: open the **MAVLink** pane, and select **Enable MAVLink on /dev/ttyACM0**;
4. Open the **/dev/ttyS1** pane.
5. To know or verify the mapping between the serial ports, click **View port map**.
6. Set the **Baud rate** to **57600** (for TELE1 port), **Parity** to **None**, and **Stop bits** to **1**.
7. Click **Apply**, and then **OK** to close the dialog box.



Block Parameters: TELE1

PX4 Serial Transmit

Send serial data to the Universal Asynchronous Receiver Transmitter(UART) port.
The block expects the values as an [Nx1] or [1xN] array.

Main

Port: /dev/ttyS1

Block Parameters: Serial Receive

Serial Receive

Receive binary data over serial port.

Parameters

Port: <Select a port...>

Data type: uint8

Data frame

☑ Header: [13 11 11 11]

☑ Terminator: Custom terminator [13 10 10 10]

Data size: 12

☑ Enable blocking mode

Action when data is not available: Output last received value

Custom value: 0

Block sample time: 0.0001

Block Parameters: Serial Configuration

Serial Configuration

Configure the parameters for the serial port.

Parameters

Port: <Select a port...>

Baud rate: 57600

Data bits: 8

Figure 5.16: Serial transmit and serial receive models with the block configurations

The serial transmit and serial receive models that we use for logging the real-time flight data are shown in Fig. 5.16.

In the serial transmit model, the **TELE1** PX4 Serial Transmit block is connected to the signals we are interested in and sends the chosen data to the ground station computer. By double-clicking this block, we can verify that the Port we are accessing is the **/dev/ttyS1** which is the same as we can find in the port map. The header and terminator are set to be [13 11 11 11] and [13 10 10 10] respectively, which will be used in the serial receive block configuration to build a link between the two models.

In the serial receive model, the Serial Configuration block and the Serial Receive block work in pairs to receive the data sent from the quadrotor. By checking the USB Serial Port in Device Manager on Windows, we can figure out the correct port to use for both of the blocks. The Baud rate is set to 57600 when using the TELE1 port with telemetry radios. This serial receive model can run in real time and log the chosen data while doing the flight test.

uORB

The uORB, a micro (symbolized as u) sized Object Request Broker, provides a data structure for data distribution [75]. PX4 uses it as an asynchronous publish/subscribe messaging API for inter-thread/inter-process communication. UORB works like a librarian who keeps track of information while communicating with people. Customers have a notion of ”Topic” as the book they are writing or reading from. But only the librarian can actually touch, write, and read the book. Therefore customers never actually get to interact with the physical book itself [78, 79]. In Simulink, the device driver block (for example, PX4 uORB Read block) requests sensor data over uORB from the IO server built into the Pixhawk hardware. In our Simulink program, we get IUM data by using the Accelerometer, Gyroscope, and Magnetometer blocks provided in the UAV Toolbox Support Package for PX4 Autopilots, shown in Fig. 5.17. Each of the blocks is a PX4 uORB Read block, which can be checked by clicking the arrow at the bottom left corner of the PX4 Accelerometer block, see in Fig. 5.18.

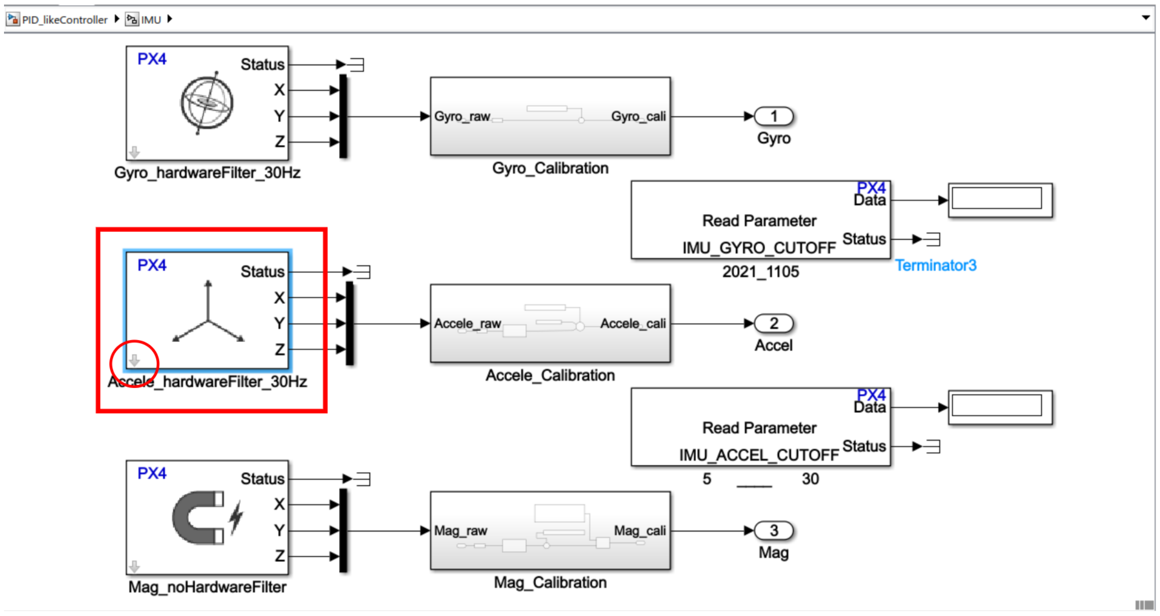
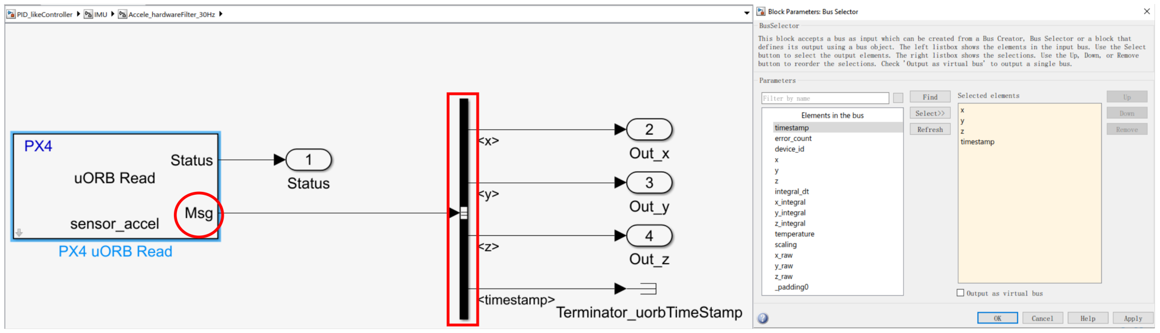


Figure 5.17: IUM model for reading the IMU data from Pixhawk 4

UORB has four main concepts: message, topic, module, and publish/subscribe. The PX4 Accelemeter block, as shown in Fig. 5.17, is taken as an example to explain these four concepts.

- Message defines the basic information format with a name and internal values, like language grammar. A message describing the accelerometer could be named *sensor_accel*, and store information such as timestamp, device ID, reference frames, temperature, and the number of raw samples, etc. The timestamp is mandatory for all messages which tells when the message was published and is used by subscribers to determine the age of the message. A uORB message is represented as a bus signal in Simulink, and the PX4 *sensor_accel* uORB Read block, shown in Fig. 5.18, outputs a Simulink bus signal corresponding to its defined uORB message.



This block accepts a bus as input which can be created from a Bus Creator, Bus Selector or a block that defines its output using a bus object. The left listbox shows the elements in the input bus. Use the Select button to select the output elements. The right listbox shows the selections. Use the Up, Down, or Remove button to reorder the selections. Check 'Output as virtual bus' to output a single bus.

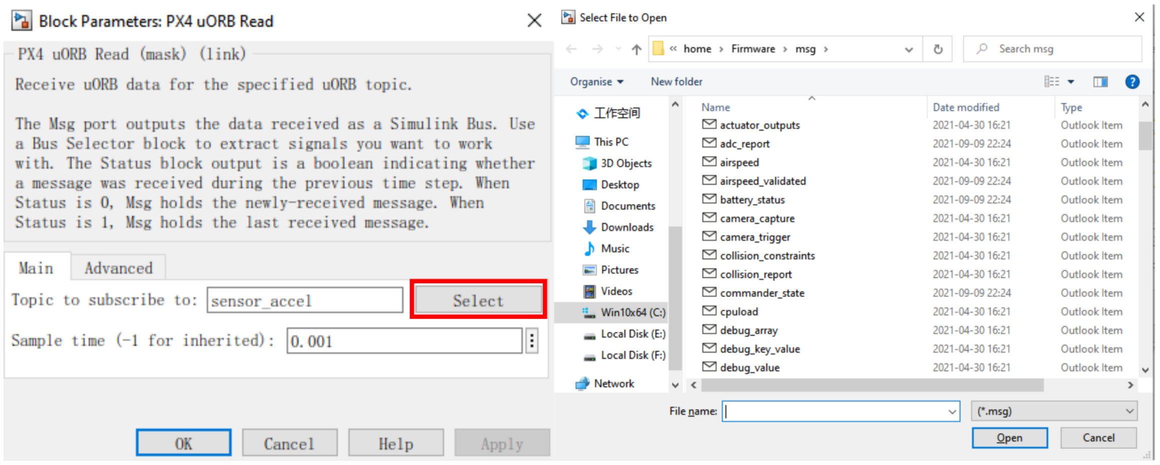
Elements in the bus: timestamp, error_count, device_id, x, y, z, integral_dt, x_integral, y_integral, z_integral, temperature, scaling, x_raw, y_raw, z_raw, _padding0

Selected elements: x, y, z, timestamp

☐ Output as virtual bus

Figure 5.18: PX4 *sensor_accel* uORB Read block (left) and BusSelector parameter block (right)

- Topic is a communication funnel where the message gets sent and received. Different topics are separated from each other, and a message can be used nested in other messages. New uORB topics can be added either within the main PX4/PX4-Autopilot repository or can be added in out-of-tree message definitions. Message files can not be used directly in the code, so these message files are analyzed and converted into a $C/C++$ struct during the build time to make it possible for the codebase to use the message. The build system by default automatically registers the topics with an identical name as the message, this way the topic name can always be used as the name of the message since it is always registered. By double-clicking the PX4 *sensor_accel* uORB Read block in Fig. 5.18, the message subscribed to this topic can be changed, shown in Fig. 5.19. And a list of messages can be found by clicking the select button in the PX4 *sensor_accel* uORB Read Parameter block.



PX4 uORB Read (mask) (link)

Receive uORB data for the specified uORB topic.

The Msg port outputs the data received as a Simulink Bus. Use a Bus Selector block to extract signals you want to work with. The Status block output is a boolean indicating whether a message was received during the previous time step. When Status is 0, Msg holds the newly-received message. When Status is 1, Msg holds the last received message.

Topic to subscribe to: sensor_accel

Sample time (-1 for inherited): 0.001

Name	Date modified	Type
actuator_outputs	2021-04-30 16:21	Outlook Item
adc_report	2021-09-09 22:24	Outlook Item
airspeed	2021-04-30 16:21	Outlook Item
airspeed_validated	2021-09-09 22:24	Outlook Item
battery_status	2021-09-09 22:24	Outlook Item
camera_capture	2021-04-30 16:21	Outlook Item
camera_trigger	2021-04-30 16:21	Outlook Item
collision_constraints	2021-04-30 16:21	Outlook Item
collision_report	2021-04-30 16:21	Outlook Item
commander_state	2021-09-09 22:24	Outlook Item
cpuload	2021-04-30 16:21	Outlook Item
debug_array	2021-04-30 16:21	Outlook Item
debug_key_value	2021-04-30 16:21	Outlook Item
debug_value	2021-04-30 16:21	Outlook Item

File name: (*.msg)

Figure 5.19: PX4 *sensor_accel* uORB Read Parameter block (left) and a select-able message list (right)

- Modules use uORB topics to communicate with each other. Our Simulink module, shown in Fig. 5.17, reads the onboard sensors values and publishes data using the corresponding uORB topics over uORB message bus.

- Publish/Subscribe: Sending out messages on a topic is called publishing, and listening to a topic is called subscribing. Every topic, therefore, has at least one publisher and subscriber. The PX4 uORB Read block is a subscriber that receives the message sent to the topic, which can be checked by clicking the arrow at the bottom left corner of the PX4 *sensor_accel* uORB Read block, details are provided in Fig. 5.20. The selected signals on the top right of Fig. 5.20 are extracted from the subscribed message. As a result, the accelerometer data can be read from Pixhawk 4.

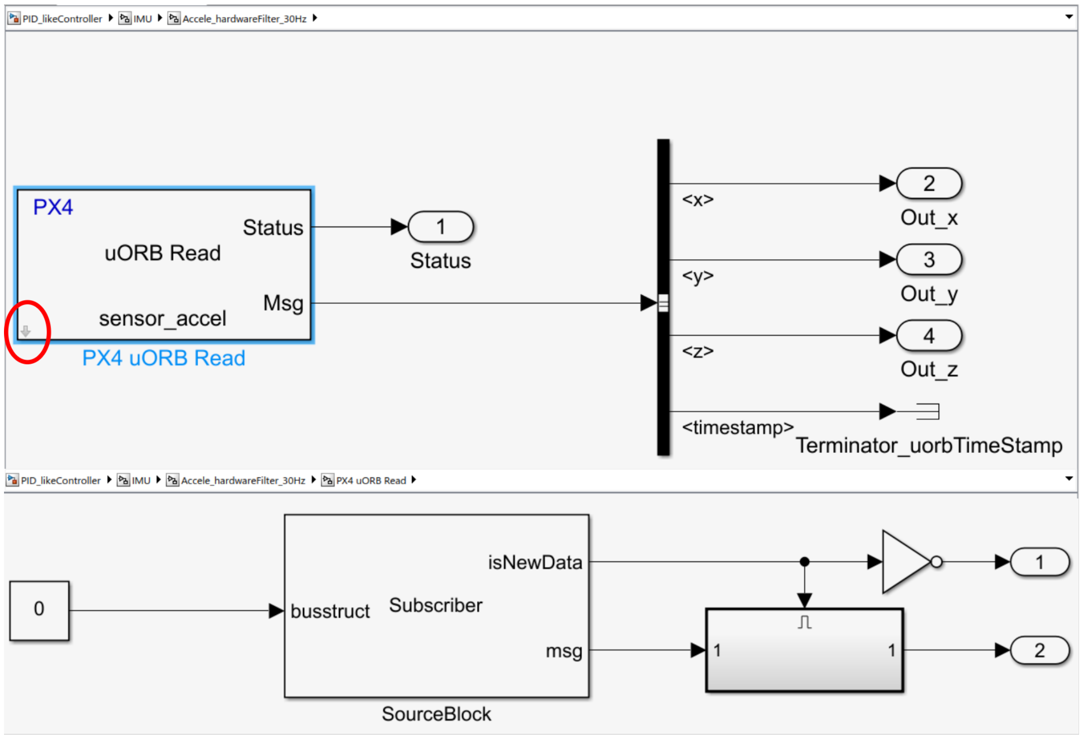


Figure 5.20: PX4 *sensor_accel* uORB Read block (top) and the block details (bottom)

The most basic and essential concept in uORB is the topic. Topics are at the heart of all the interfaces to uORB. For more details, please refer to [74, 78, 79].

Simulation Data Inspector

The Simulation Data Inspector (SDI) is used to visualize the flight performance and record the data in our flight testing, shown in Fig. 5.21. It integrates with data logging in Simulink models, works well for visualizing many signals throughout a model, and manages incoming simulation data using the archive. By default, the previous run moves to the archive when a new simulation is started. Signals from the archive can be plotted, and runs of interest can be dragged back into the work area [80, 81].

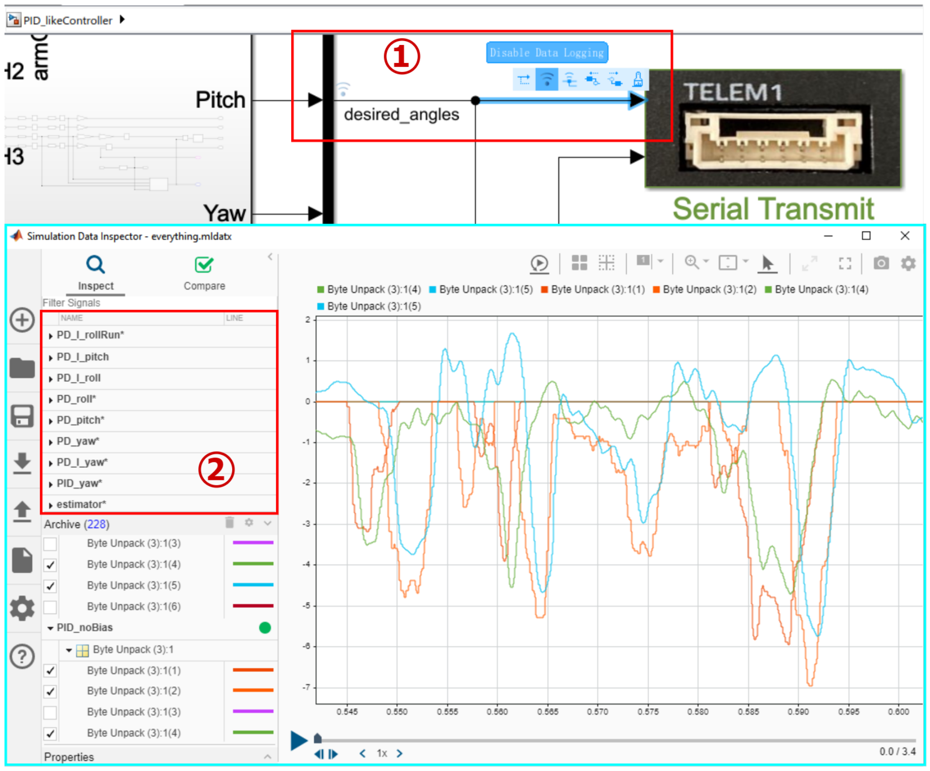


Figure 5.21: Open the Simulation Data Inspector in Simulink (top) and Simulation Data Inspector (bottom)

The Simulation Data Inspector supports:

- Viewing signals during simulation
- Logging, importing, and exporting data, the imported data from the workspace or a MAT-file will be seen in the second highlight area in Fig. 5.21
- Configurable subplot layouts and visualization settings
- Viewing data using multiple visualization options, including maps and XY plots
- Post-processing and data analysis using comparisons with tolerance values
- Saving plots and data to share or archive results

There are three ways to open the Simulation Data Inspector:

- Simulink Toolstrip: On the Simulation tab, under Review Results, click Data Inspector.
- Select the signal in the model, then click the logging budge. When the logging badge appears above the signal marked for logging, as shown in the first highlight area in Fig. 5.21, signal-click it.
- MATLAB command prompt: Enter *Simulink.sdi.view*.

5.3 Overview of the Quadrotor Implementation Phases

The quadrotor implementation includes the following three phases, as shown in Fig. 5.22:

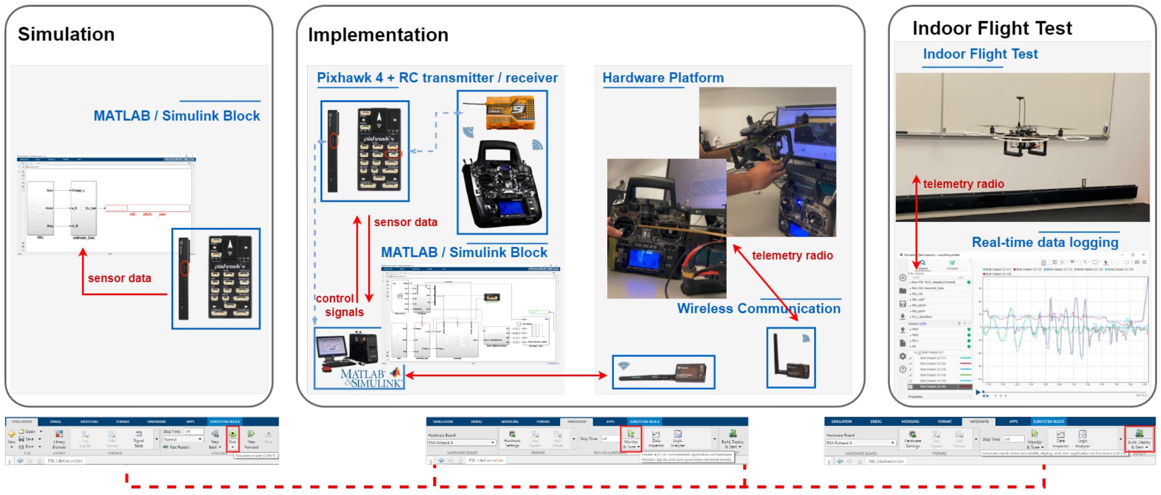


Figure 5.22: The quadrotor implementation phases

- Testing the attitude estimation with hardware in the loop

  The estimator's efficiency and accuracy are tested and verified in this phase. The IMU is calibrated using the method in section 4.2 and a rough gain tuning of the estimator is done after. In this phase, the Pixhawk 4 is connected to the host computer using a USB cable, the IMU data are read from the Pixhawk board, and the estimation program is running on the host computer. To run the program in the simulation mode, we click Run in the Simulink Toolstrip on the Simulation tab.

- Implementation of the complete program

  This phase is the core of our whole testing, where the configuration of each component and the majority of the parameter tuning are performed. In this phase, the complete Simulink program is used. Two independent models are included in this Simulink program, a flight controller model and a serial communication model. The former is used to conduct the test in this phase, while the latter is built and set up to accomplish real-time wireless data logging for the indoor flight test in the following phase. During the test, the battery is connected and the propellers are detached from the motors, which are capable of rotating when a command is given. Pixhawk 4 is connected to the host computer using a USB cable and the Monitor & Tune mode is used. By clicking Monitor & Tune from the Run on Hardware section of the Hardware tab in the Simulink Toolstrip, the C code will be generated for the controller model and automatically deployed to the Pixhawk board. Pixhawk 4 communicates with the host computer over Monitor & Tune mode, which allows tuning parameters in real time on Pixhawk and logging real-time signals in Simulink.

  1. RC setups and configurations:

   (a) Specify the channels to which the transmitter sticks (for roll, pitch, yaw, and throttle) are mapped;
   (b) Calibrate the minimum, maximum, trim, and reverse settings for all transmitter channels by using the method in Section 5.1.2;
   (c) Set a safety switch, an arming gesture, and a trigger that resets the orientation of the drone once it has been armed or disarmed.

2. Fine-tune the estimator by adjusting the cutoff frequencies of the hardware and the software low-pass filters until the desired attitude performance is achieved.
3. Test the control algorithms in Chapter 6 and check the real-time performance from the Simulink Data Inspector (SDI). With the battery connected and the RC orientation set to zero, lift the drone with hands and apply orientations to test the motor feedback and the control logic. Rough-tune the control gains until achieving satisfactory plots from the SDI. This group of gains will be used as a starting point for the indoor flight test.
4. Build and set up a serial wireless communication system (see 5.1.2) to realize real-time wireless data logging (see 5.2.2) for the indoor flight test.

The implementation is done in the monitor and tune implementation mode which enables real-time parameter tuning. As soon as the parameter values change in the Simulink model, the modified values are communicated to the target hardware immediately. In addition, compared to the simulation mode, this testing can help to identify issues related to the communication channels and the magnetic field error caused by battery current. This phase facilitates an efficient parameter tuning process and serves as appropriate safety preparation for the real flight test.

- Indoor flight test

  In this phase, the flight controller model is built, deployed, and run on Pixhawk 4 to fly the drone. By clicking Build, Deploy, & Start from the Deploy section of the Hardware tab in the Simulink Toolstrip, MATLAB generates C-code from the model. The generated C-code is then deployed on Pixhawk 4 and starts execution. The serial communication model is run on the host computer during the test to log the real-time flight data via telemetry radios. The flight data are saved and plotted in order to analyze the flight performance. Control gains are fine-tuned until satisfactory flight performance is achieved.

Chapter 6

Attitude Control

Attitude control for quadrotors has evolved from several different classes of controllers over the past decades; linear controllers have been proven to be sufficient for obtaining stable flights. The concept of PID control with its simplicity has been explored extensively as a classical approach to yielding reliable and consistent tracking performance. A number of research groups have adopted PID controllers specifically for quadrotor systems and due to their efforts, PID controllers are now widely used for commercial quadrotors. Three different PID-type controllers are introduced in this chapter, which have already been successfully implemented on the PX4 autopilot system. The experimental results are presented and the flight performance are compared.

6.1 PD-Like Controller

Let Q_d be the desired attitude to be tracked, generated by

$$\dot{Q}_d = \begin{bmatrix} -q_d^\top \\ \eta_d I_3 + S(q_d) \end{bmatrix} \omega_d \tag{6.1}$$

where $Q_d := [\eta_d,\ q_d]^\top$. The attitude tracking error is represented by $R_e := R_d^\top R$, which corresponds to the unit quaternion $Q_e := Q_d^{-1} \odot Q$ and is defined as

$$Q_e = \begin{bmatrix} \eta_e \\ q_e \end{bmatrix} = \begin{bmatrix} \eta_d \eta + q^\top q_d \\ \eta_d q - \eta q_d - q_d \times q \end{bmatrix} \tag{6.2}$$

Differentiating both sides of equation $Q_d \odot Q_e = Q$ with respect to time, one has

$$\dot{Q}_d \odot Q_e + Q_d \odot \dot{Q}_e = \dot{Q} \tag{6.3}$$

Then the time derivative of Q_e can be derived as follows:

$$\begin{aligned}
\dot{Q}_e &= Q_d^{-1} \odot \dot{Q} - Q_d^{-1} \odot \dot{Q}_d \odot Q_e \\
&= \frac{1}{2} Q_d^{-1} \odot Q \odot Q_\omega - \frac{1}{2} Q_d^{-1} \odot Q_d \odot Q_{\omega_d} \odot Q_e \\
&= \frac{1}{2} Q_e \odot Q_\omega - \frac{1}{2} Q_{\omega_d} \odot Q_e \\
&= \frac{1}{2} Q_e \odot Q_\omega - \frac{1}{2} Q_e \odot Q_e^{-1} \odot Q_{\omega_d} \odot Q_e \\
&= \frac{1}{2} Q_e \odot Q_\omega - \frac{1}{2} Q_e \odot Q_{\tilde{\omega}_d} \\
&= \frac{1}{2} Q_e \odot Q_{\omega_e} \qquad (6.4)
\end{aligned}$$

Using the fact that $Q_e^{-1} \odot Q_{\omega_d} \odot Q_e = Q_{\tilde{\omega}_d}$ where $\tilde{\omega}_d = R_e^\top \omega_d$ and $\omega_e = \omega - \tilde{\omega}_d$, and assuming that Q_d is constant (or slowly varying), *i.e.*, $\omega_d \approx 0$, the attitude error dynamics can be rewritten as

$$\dot{Q}_e = \begin{bmatrix} -\frac{1}{2} q_e^\top \omega \\ \frac{1}{2}(\eta_e I_3 + S(q_e))\omega \end{bmatrix} \qquad (6.5)$$

using the fact $\omega_d = 0$ and $\omega_e = \omega$. The PD-like attitude stabilization control law is given by [39]

$$\tau = -\alpha q_e - \Gamma_1 \omega \qquad (6.6)$$

Under the proposed control law (6.6), the closed loop dynamics are given by

$$\begin{aligned}
\dot{q}_e &= \frac{1}{2}(\eta_e I_3 + S(q_e))\omega \\
I_f \dot{\omega} &= -\omega \times I_f \omega - \alpha q_e - \Gamma_1 \omega \qquad (6.7)
\end{aligned}$$

Theorem 6.1. [39] Consider (3.11) under the control law (6.6). Then, the equilibrium point $(\eta_e = 1, q_e = 0, \omega = 0)$ is almost globally asymptotically stable.

Proof. The time derivative of the following Lyapunov function candidate:

$$V = \alpha q_e^\top q_e + \alpha(1 - \eta_e)^2 + \frac{1}{2}\omega^\top I_f \omega \qquad (6.8)$$

can be written as follows

$$\begin{aligned}
\dot{V} &= 2\alpha q_e^\top \dot{q}_e + 2\alpha(\eta_e - 1)\dot{\eta}_e + \omega^\top I_f \dot{\omega} \\
&= \alpha q_e^\top ((\eta_e I_3 + S(q_e))\omega) - \alpha(\eta_e - 1)\left(q_e^\top \omega\right) + \omega^\top (\tau - S(\omega) I_f \omega) \\
&= \alpha \eta_e q_e^\top \omega - \alpha \omega^\top S(q_e) q_e - \alpha \eta_e q_e^\top \omega + \alpha q_e^\top \omega - \alpha \omega^\top q_e - \omega^\top \Gamma_1 \omega - \omega^\top S(\omega) I_f \omega \\
&= -\omega^\top \Gamma_1 \omega \qquad (6.9)
\end{aligned}$$

which implies that $\omega(t), q_e(t)$ and η_e are bounded. Applying the LaSalle's invariance theorem, for $\dot{V} = 0$ one can get $\omega = 0$, which implies $\dot{\omega} = 0$. Consequently, in view of the second equation in (6.7), one has $q_e(t) = 0$. Using the fact that $\eta_e^2 + q_e^\top q_e = 1$, one can easily show that $\eta_e = \pm 1$. One can also show that the equilibrium point $(q_e = 0, \eta_e = 1, \omega = 0)$ is an attractor while the equilibrium point $(q_e = 0, \eta_e = -1, \omega = 0)$ is a repeller (*i.e.*, unstable). □

Remark 1. Since the attitude of the quadrotor can not be directly measured. The real attitude Q will be replaced by the estimated attitude $\hat{Q}$ as discussed in the previous section. On the other hand, the angular velocity measurements can be corrected using the estimated gyro bias $\hat{b}$. Therefore, the PD-like control law (6.6) can be modified using the estimated attitude and gyro bias as:

$$\tau = -\alpha \hat{q}_e - \Gamma_1(\omega_{\mathcal{B}} - \hat{b}) \tag{6.10}$$

where, $\hat{Q}_e := [\hat{\eta}_e \;\; \hat{q}_e]^\top = Q_d^{-1} \odot \hat{Q}$. This modified controller will be implemented in our real-time experimental tests.

6.1.1 Experimental Results

In this experiment, two tests have been performed, a test for attitude stabilization and a test for attitude tracking. The quadrotor is taken off the ground, and its attitude is stabilized to $R = I_3$ (*i.e.*, $\phi = 0\,$deg, $\theta = 0\,$deg and $\psi = 0\,$deg) during the hover time. In the case of hovering, only the thrust commands are applied to the motors, and no orientation commands from the remote are introduced into the system. The experiment is performed with the attitude estimator (4.7-4.8), and the attitude control law (6.10). The Simulink program is given in Fig. 6.1.

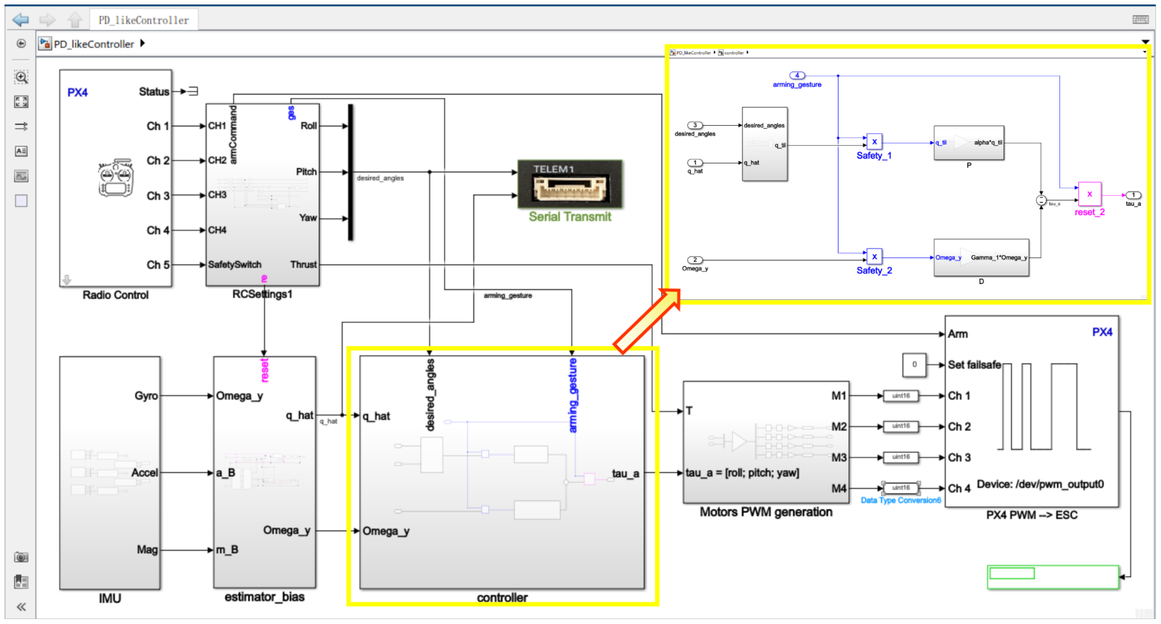


Figure 6.1: Simulink model for the PD-like controller

The gains involved in the controller are chosen as follows by trial and error:

$$\alpha = 4.5 \qquad \Gamma_1 = \mathrm{diag}([0.375,\ 0.375,\ 0.375]) \tag{6.11}$$

The real-time attitude stabilization performance is shown in Fig. 6.2, and the attitude tracking performance for the roll, pitch, and yaw respectively are shown in Fig. 6.3. Fig.

6.3 shows that the PD-like controller has the ability to control the orientation and track the reference commands closely. In Fig. 6.2, with a flight of 10 seconds, one can see that the PD-like controller has the ability to stabilize the orientation, where the roll is maintained between ± 3 deg, the pitch between ± 5 deg, and the yaw between ± 2 deg. One can also notice that a small command has been given to the pitch during the hover time to adjust for the quadrotor drift. To enhance the flight performance an integral action is added to this PD-like controller.

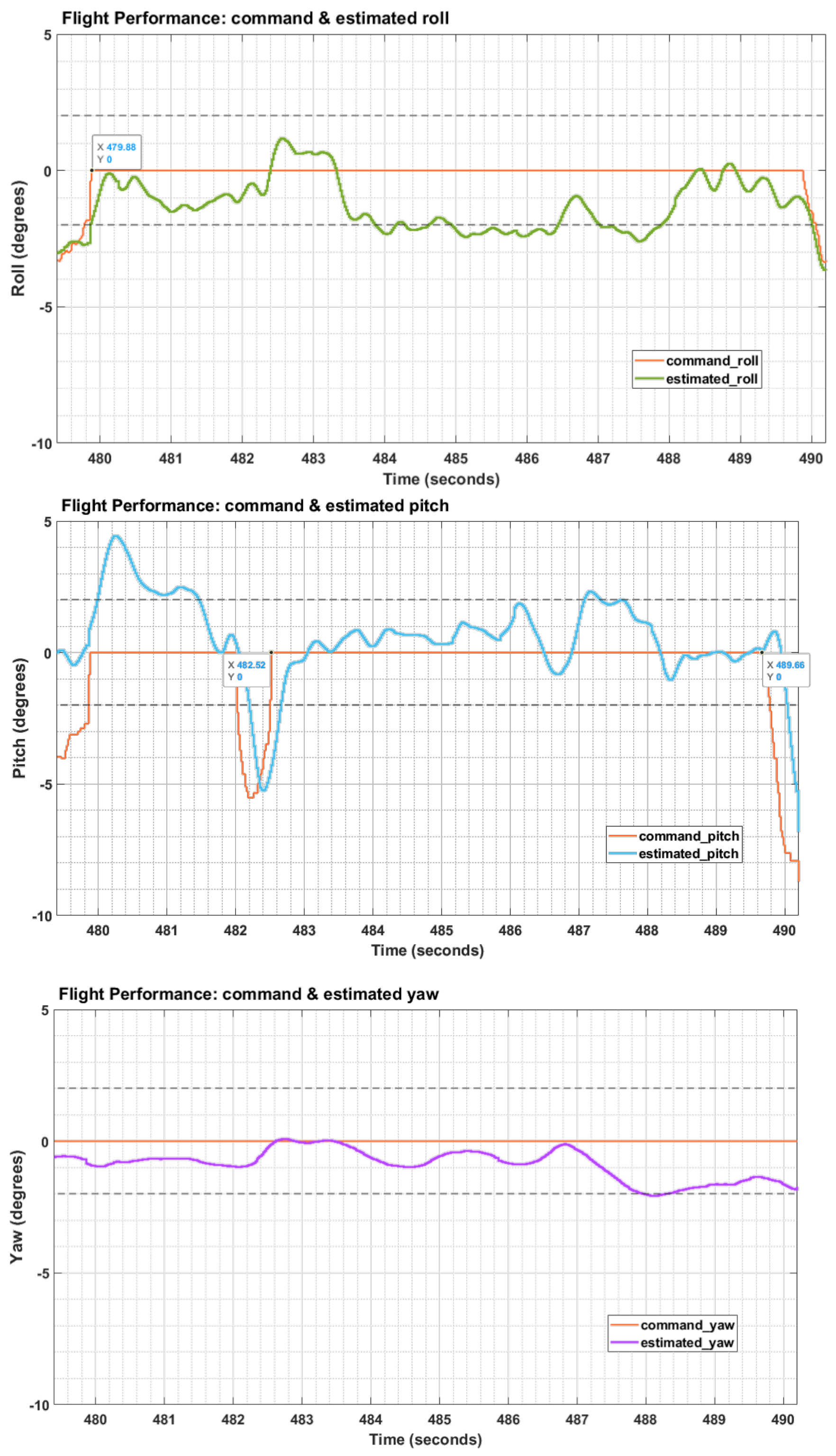


Figure 6.2: Real-time flight performance for attitude stabilization using PD-like controller

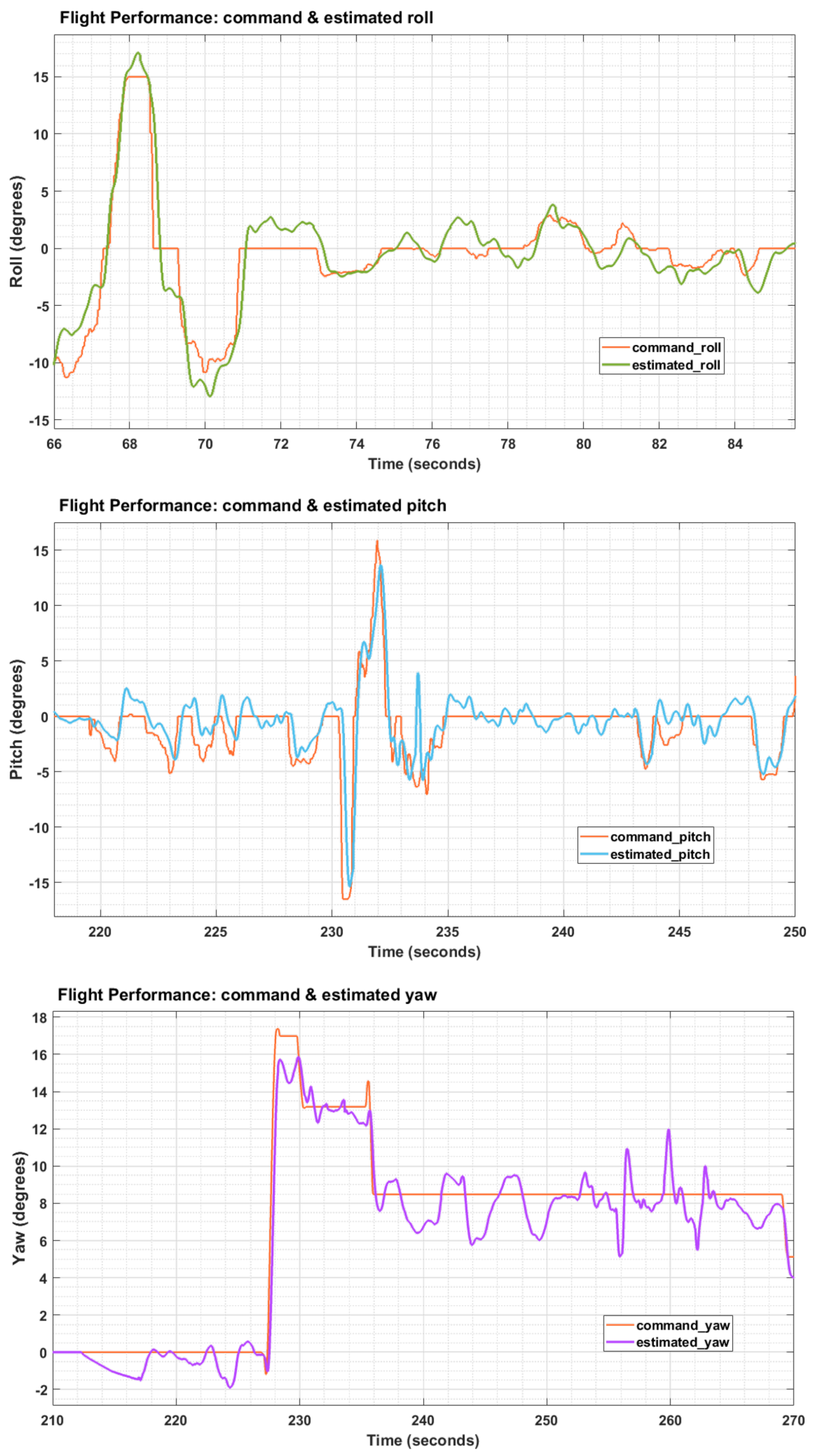


Figure 6.3: Real-time flight performance for attitude tracking using PD-like controller

6.2 PID-Like Controller

The PID-like attitude stabilization control law is given by

$$\tau = -\alpha q_e - \Gamma_1 \omega - k_1 \int_0^t \omega \mathrm{d}t - k_2 \int_0^t q_e \mathrm{d}t \tag{6.12}$$

Considering the fact that the attitude of the quadrotor can not be directly measured. The real attitude Q will be replaced by the estimated attitude $\hat{Q}$ as discussed in the previous section. On the other hand, the angular velocity measurements can be corrected using the estimated gyro bias $\hat{b}$. Therefore, the PID-like control law (6.12) can be modified using the estimated attitude and gyro bias as:

$$\tau = -\alpha \hat{q}_e - \Gamma_1(\omega_{\mathcal{B}} - \hat{b}) - k_1 \int_0^t (\omega_{\mathcal{B}} - \hat{b}) \mathrm{d}t - k_2 \int_0^t \hat{q}_e \mathrm{d}t \tag{6.13}$$

where, $\hat{Q}_e := [\hat{\eta}_e \ \hat{q}_e]^\top = Q_d^{-1} \odot \hat{Q}$. This modified controller will be implemented in our real-time experimental tests.

6.2.1 Experimental Results

In this experiment, two tests have been performed, a test for attitude stabilization and a test for attitude tracking. The quadrotor is taken off the ground, and its attitude is stabilized to $R = I_3$ (*i.e.*, $\phi = 0\,\mathrm{deg}$, $\theta = 0\,\mathrm{deg}$ and $\psi = 0\,\mathrm{deg}$) during the hover time. In the case of hovering, only the thrust commands are applied to the motors, and no orientation commands from the remote are introduced into the system. The Simulink program is given in Fig. 6.4.

The gains involved in the controller are chosen as follows by trial and error:

$$\alpha = 4.5 \qquad \Gamma_1 = \mathrm{diag}([0.375,\ 0.375,\ 0.375]) \qquad k1 = 0.6 \qquad k2 = 1.5 \tag{6.14}$$

The real-time attitude stabilization performance is shown in Fig. 6.5, and the attitude tracking performance is shown in Fig. 6.6. Fig. 6.6 shows that the PID-like controller has the ability to control the orientation and track the reference commands closely. In Fig. 6.5, with a flight of 10 seconds, one can see that the PID-like controller has the ability to stabilize the quadrotor and maintain the roll between ± 3 deg, the pitch between ± 2 deg, and the yaw between ± 2 deg. A small command has been given to the roll during the hover time to adjust for the quadrotor drift. Compared to the PD-like controller, the steady-state error is smaller, which provides a better tracking performance as well as better stabilization.

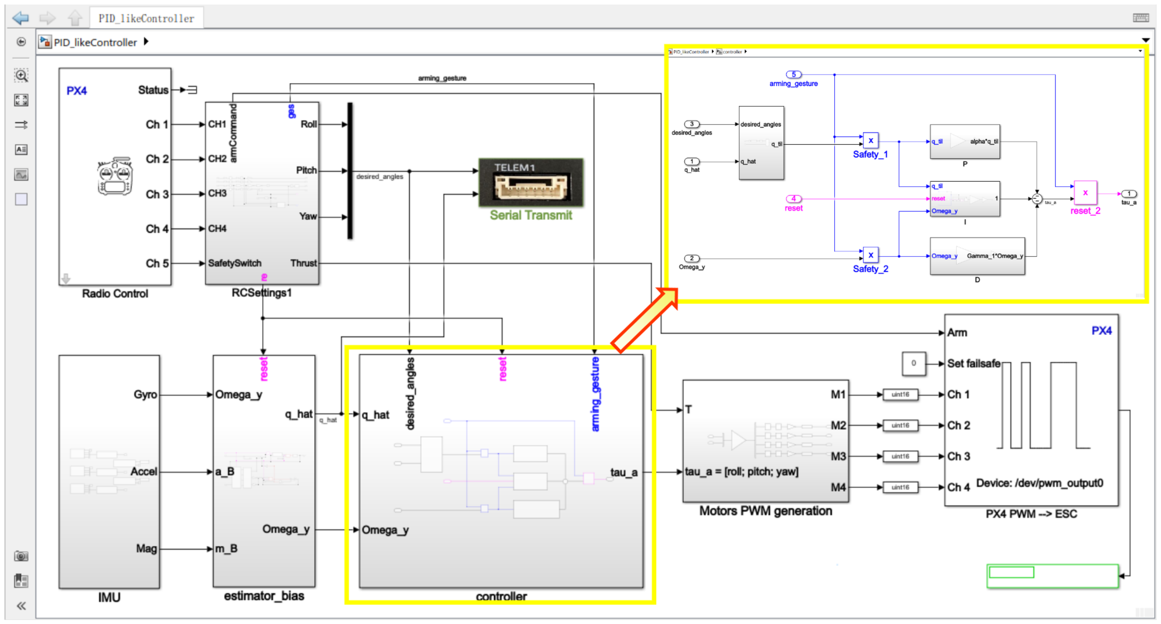


Figure 6.4: Simulink model for the PID-like controller

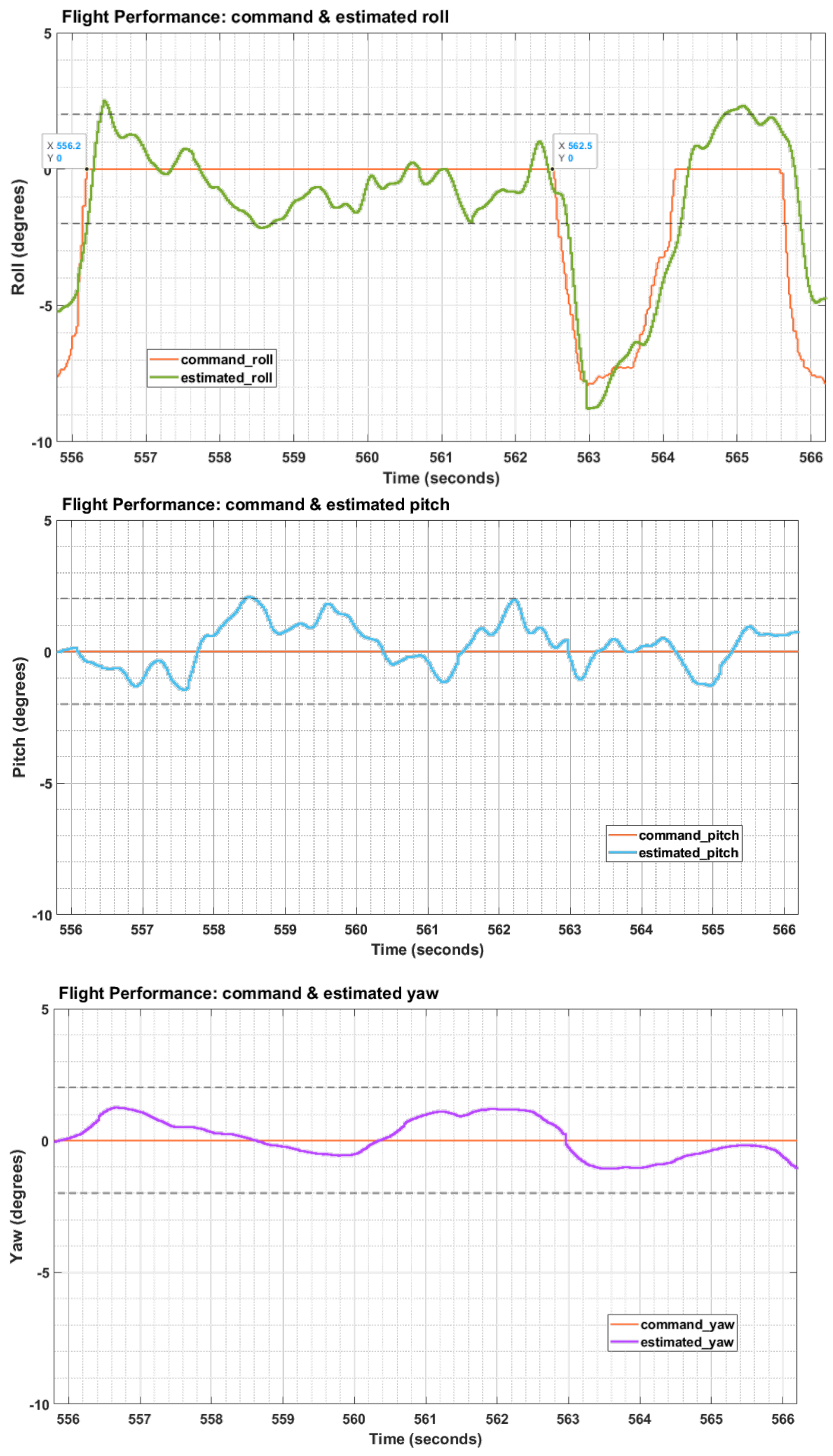


Figure 6.5: Real-time flight performance for attitude stabilization using PID-like controller

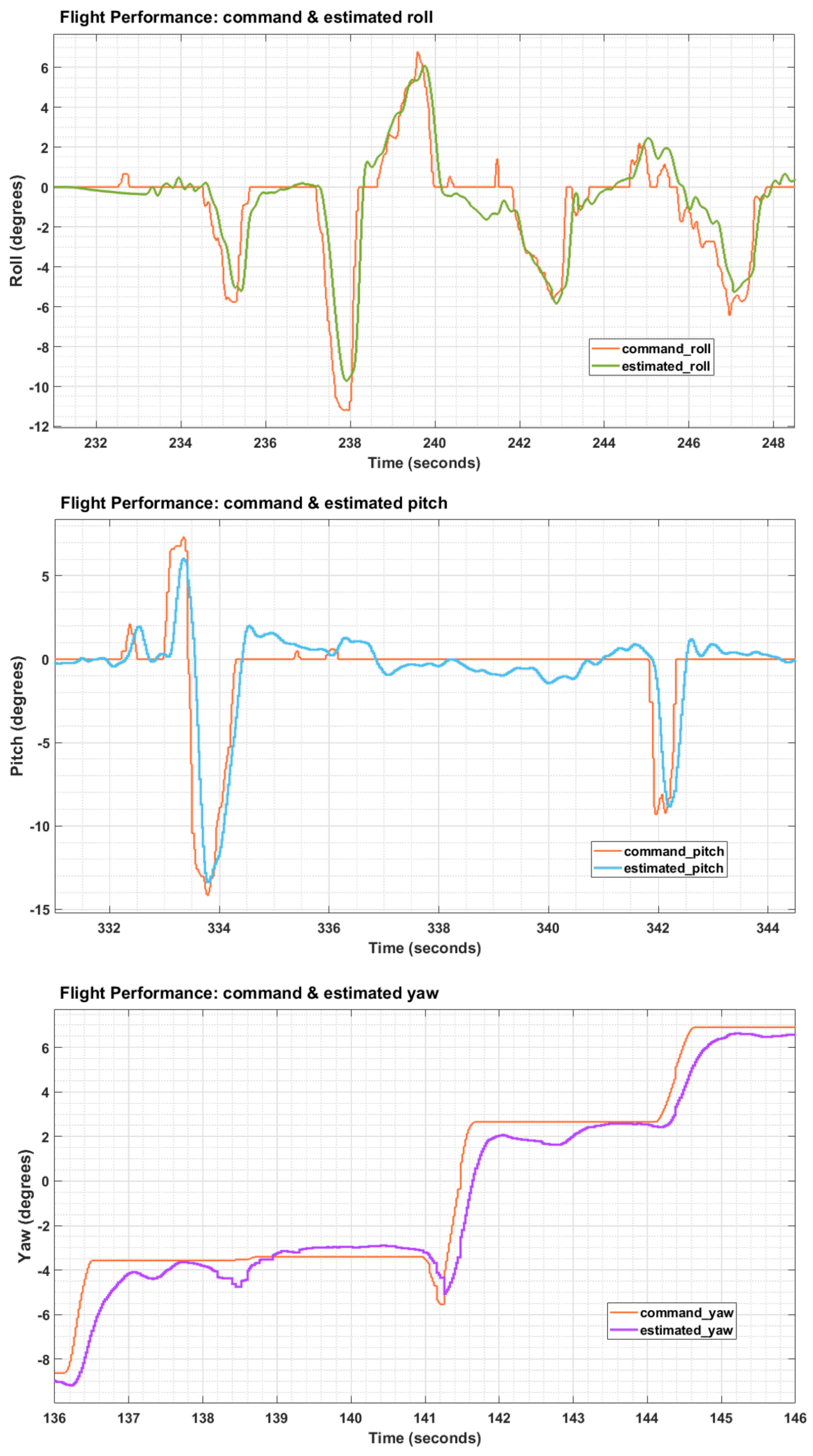


Figure 6.6: Real-time flight performance for attitude tracking using PID-like controller

6.3 Nested PID-like Controller

Aiming at improving the previous two attitude control schemes, we introduce a PID-like controller with two nested loops. The control scheme is shown in Fig. 6.7.

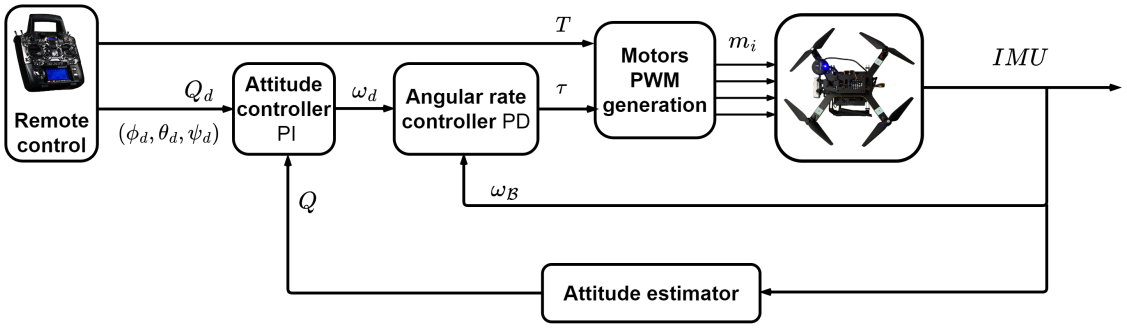


Figure 6.7: Control scheme for the nested PID-like controller

The attitude error dynamics can be rewritten as

$$\dot{Q}_e = \begin{bmatrix} \dot{\eta}_e \\ \dot{q}_e \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} q_e^\top (\omega + d\omega) \\ \frac{1}{2}(\eta_e I_3 + S(q_e))(\omega + d\omega) \end{bmatrix} \tag{6.15}$$

where ω is the angular velocity, $d\omega$ is constant gyro bias. The attitude control law is given by

$$\omega = -\eta_e K q_e - \hat{\theta} \tag{6.16}$$

with $\dot{\hat{\theta}} = \Gamma \eta_e q_e$, $K = K^T > 0$ and $\Gamma = \Gamma^T > 0$. Consider the following Lyapunov function candidate:

$$V = q_e^\top q_e + \frac{1}{2} \tilde{\theta}^\top \Gamma^{-1} \tilde{\theta} \tag{6.17}$$

where $\tilde{\theta} = \hat{\theta} - d\omega$. The time derivative of V can be written as follows:

$$\begin{aligned} \dot{V} &= 2 q_e^\top \dot{q}_e + \tilde{\theta}^\top \Gamma^{-1} \dot{\hat{\theta}} \\ &= q_e^\top \left[(\eta_e I_3 + S(q_e))(\omega_d + d\omega)\right] + \tilde{\theta}^\top \Gamma^{-1} \dot{\hat{\theta}} \\ &= q_e^\top \eta_e (\omega_d + d\omega) + \tilde{\theta}^\top \Gamma^{-1} \dot{\hat{\theta}} \\ &= q_e^\top \eta_e (-\eta_e K q_e - \hat{\theta} + d\omega) + \tilde{\theta}^\top \Gamma^{-1} \dot{\hat{\theta}} \\ &= -\eta_e^2 q_e^\top K q_e \end{aligned} \tag{6.18}$$

which implies that $\tilde{\theta}(t), q_e(t)$ and η_e are bounded. As per LaSalle's invariance theorem, setting $\dot{V} = 0$ implies either $\eta_e = 0$ or $q_e = 0$. One can show that the (undesired) equilibrium characterized by $\eta_e = 0$ is unstable while the desired equilibrium characterized by $q_e = 0$ is asymptotically stable.

Remark 2. Since the attitude of the quadrotor can not be directly measured. The real attitude Q will be replaced by the estimated attitude $\hat{Q}$ as discussed in the previous section.

On the other hand, the angular velocity measurements can be corrected using the estimated gyro bias $\hat{b}$. Therefore, the attitude control law (6.16) can be modified using the estimated attitude and gyro bias as:

$$\omega = -\hat{\eta}_e K \hat{q}_e - \Gamma \int_0^t \hat{\eta}_e \hat{q}_e \, \mathrm{d}t \tag{6.19}$$

where, $\hat{Q}_e := [\hat{\eta}_e \ \hat{q}_e]^\top = Q_d^{-1} \odot \hat{Q}$. And the angular rate controller which is the inner loop PD controller is given by

$$\tau = K_p[\omega - (\omega_{\mathcal{B}} - \hat{b})] + K_d \frac{\mathrm{d}[\omega - (\omega_{\mathcal{B}} - \hat{b})]}{\mathrm{d}t} \tag{6.20}$$

This modified attitude controller will be implemented in our real-time experimental tests.

6.3.1 Experimental Results

In this experiment, two tests have been performed, a test for attitude stabilization and a test for attitude tracking. The quadrotor is taken off the ground, and its attitude is stabilized to $R = I_3$ (*i.e.*, $\phi = 0\,$deg, $\theta = 0\,$deg and $\psi = 0\,$deg) during the hover time. In the case of hovering, only the thrust commands are applied to the motors, and no orientation commands from the remote are introduced into the system. The Simulink program is given in Fig. 6.8.

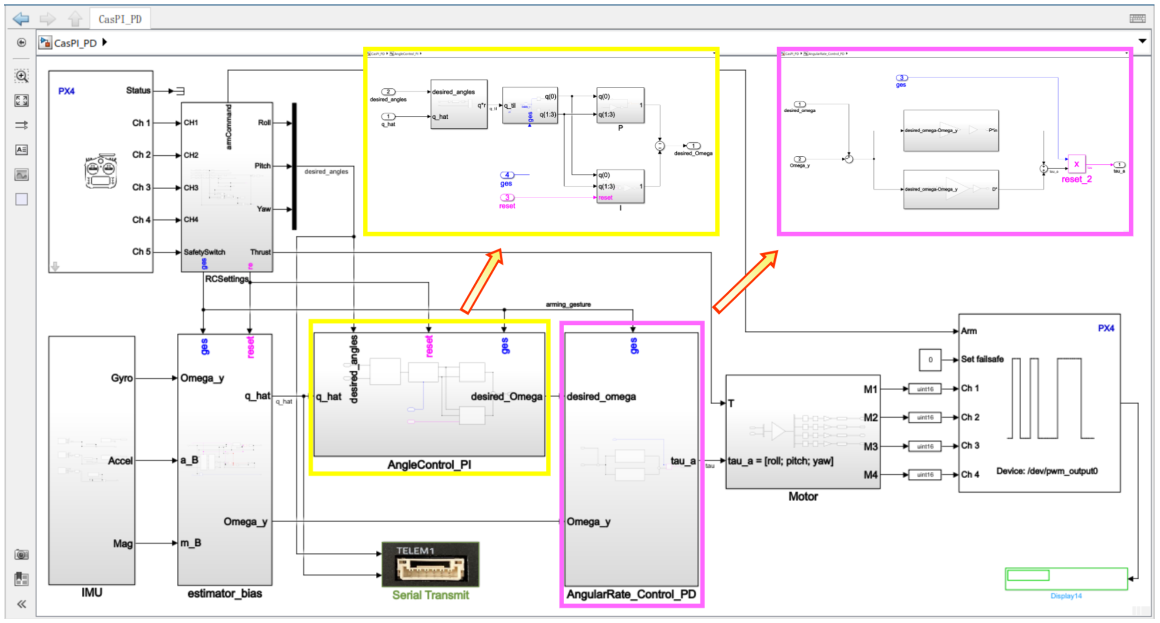


Figure 6.8: Simulink model for the nested PID like controller

The gains involved in the controller are chosen as follows by trial and error:

$$K = 18 \qquad \Gamma = 2.5 \qquad K_p = 0.2 \qquad K_d = 0.001 \tag{6.21}$$

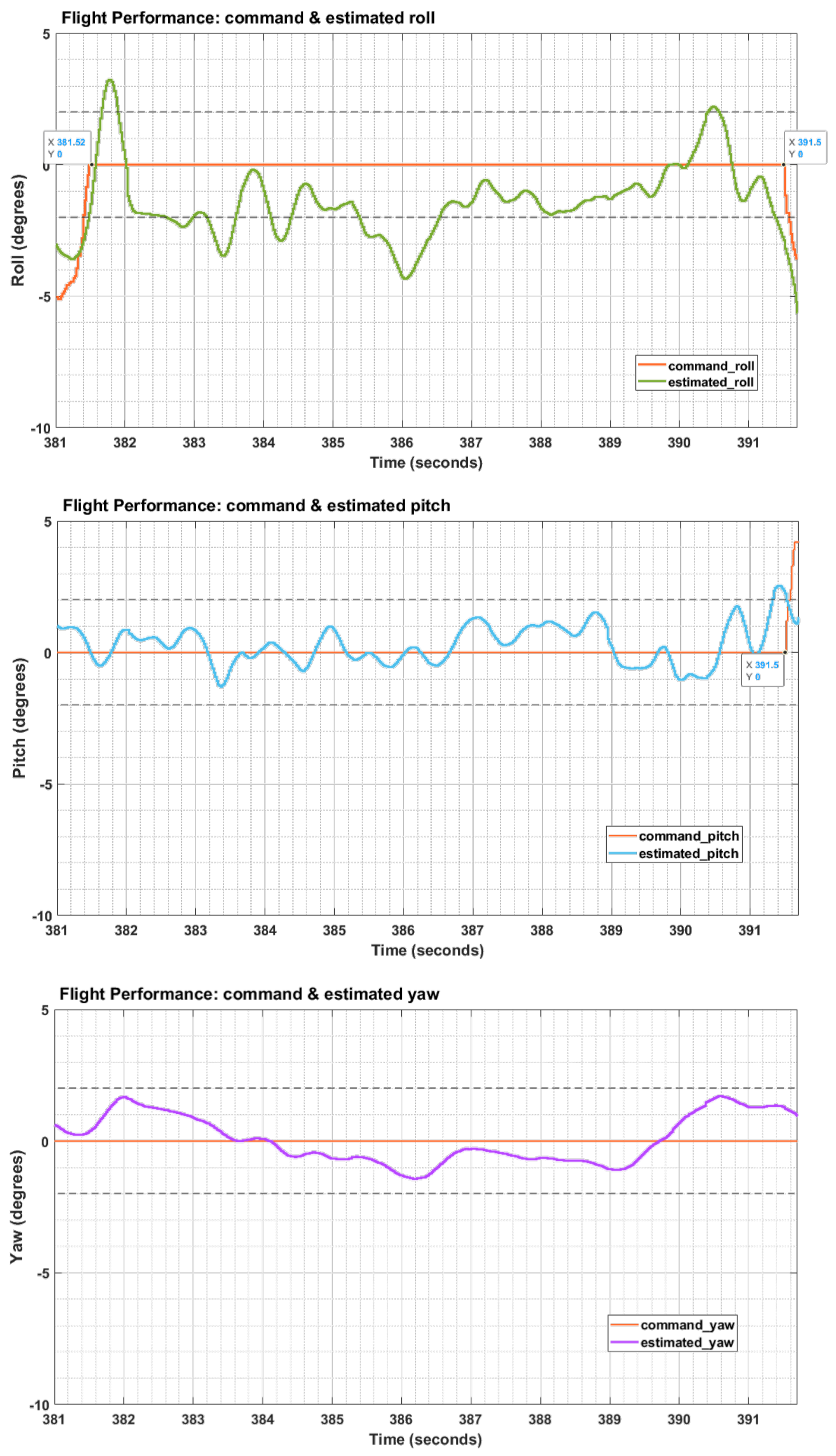
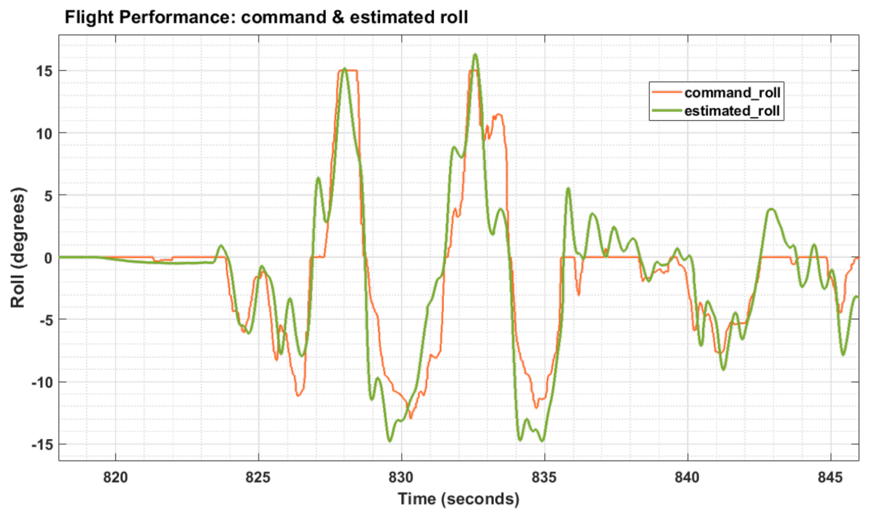
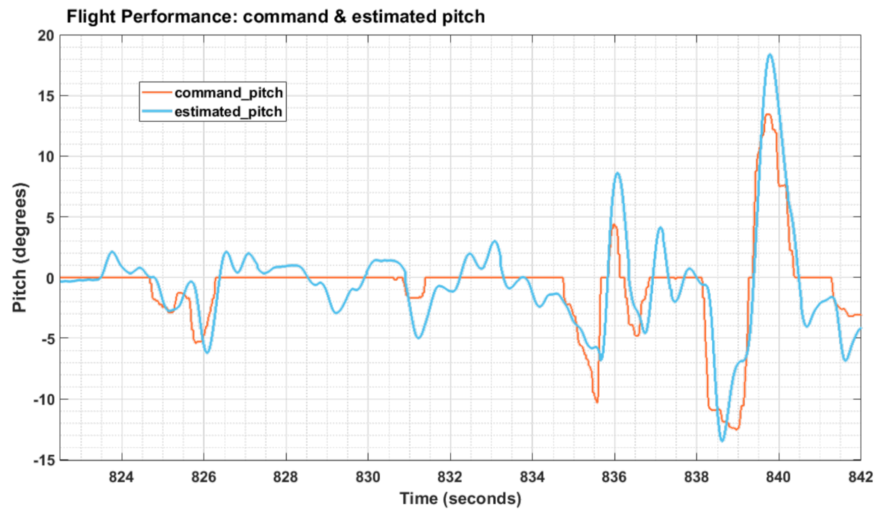


Figure 6.9: Real-time flight performance for attitude stabilization using nested PID-like controller

The real-time attitude stabilization performance is shown in Fig. 6.9, and the attitude tracking performance is shown in Fig. 6.10. Fig. 6.10 shows that the nested PID-like controller has the ability to control the orientation and track the reference commands closely. In Fig. 6.9, with a flight of 10 seconds, one can see that the nested PID-like controller has the ability to stabilize the quadrotor and maintain the roll between ± 5 deg, the pitch between ± 2 deg, and the yaw between ± 3 deg. No orientation commands have been given to the system to adjust the position within the 10 seconds. Moreover, compared to the PD-like and PID-like controllers, the motion achieved with this controller is relatively smoother and less aggressive.





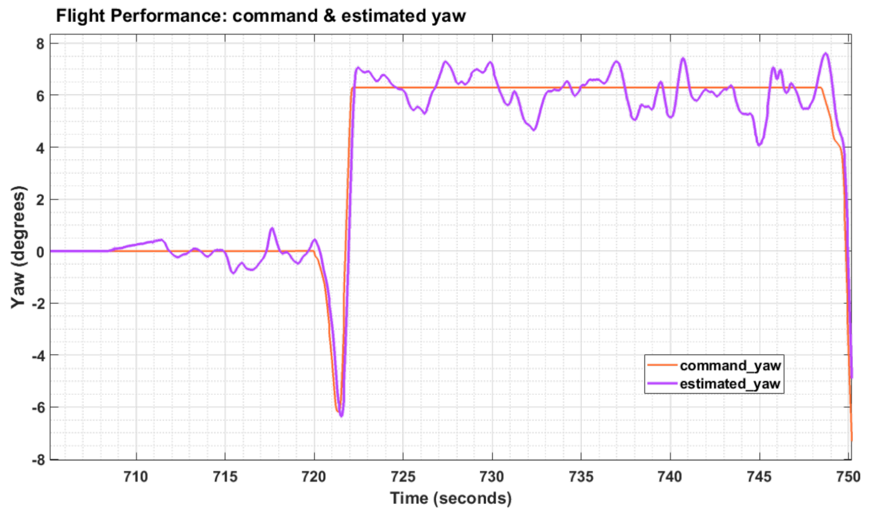


Figure 6.10: Real-time flight performance for attitude tracking using nested PID-like controller

6.4 Comparison

Due to the unavailability of sensors that directly measure the attitude, a suitable attitude observer is used to reconstruct the attitude from the angular velocity and inertial vector measurements provided by the IMU. Three different PID-type controllers are implemented and proved to have the ability to control the orientation of the quadrotor with satisfactory performance. To compare the performance of the three controllers a group of flight tests of roughly 30-50 seconds is conducted.

The comparison of the flight performance is presented in table. 6.1, and shown in Fig. 6.11-6.13.

PD	Pros	Responsive
	Cons	Steady state error, shaking aggressively during takeoff
PID	Pros	Stable performance, precise
	Cons	Little aggressive responses, needs extra attention to fly
Nested PID	Pros	Stable, easy to fly with smooth performance
	Cons	Over-correction occurs when hovering for too long, sometimes causes shaking or small oscillation when this happens

Table 6.1: Comparison of the flight performance with the three controllers

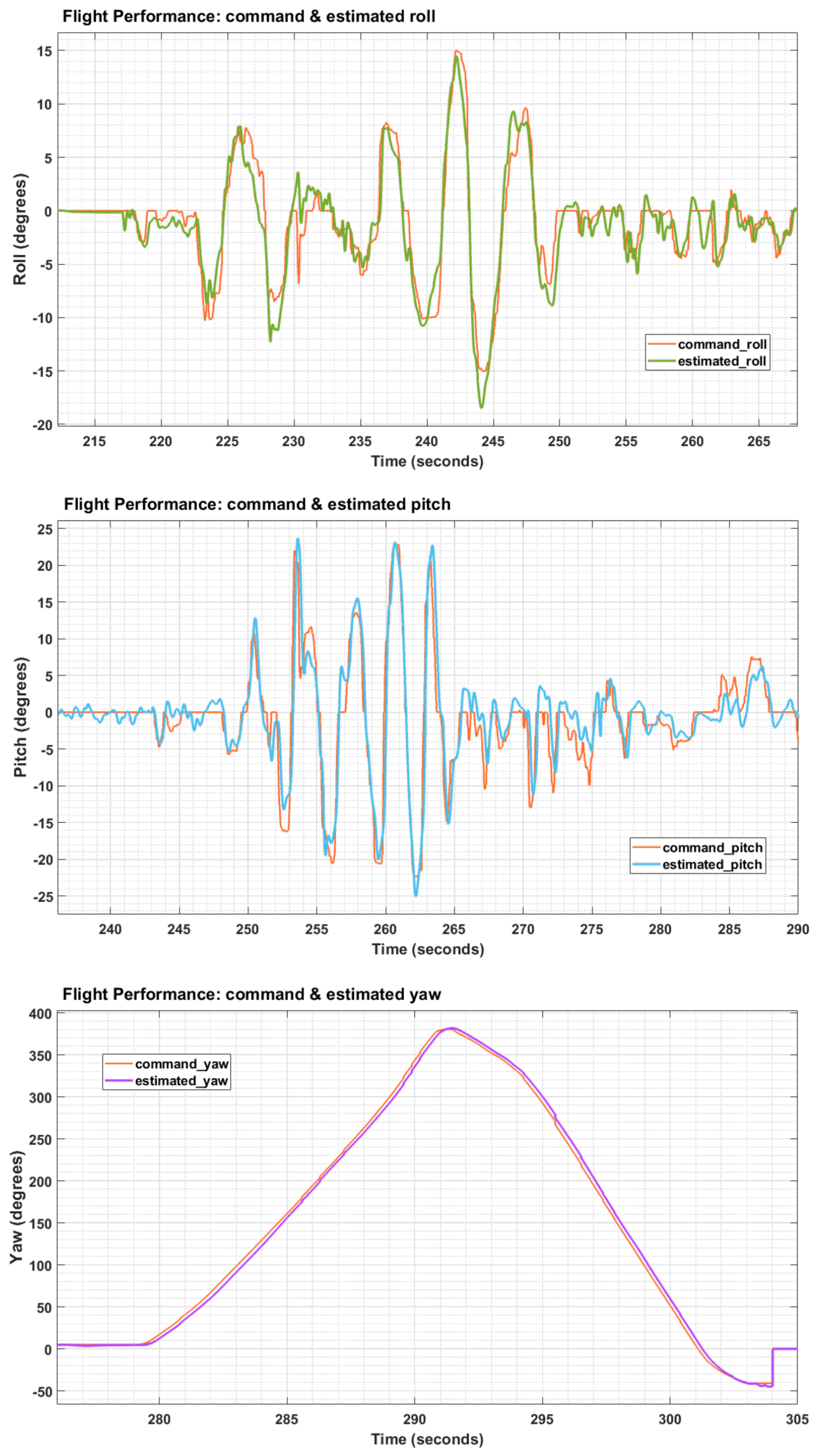


Figure 6.11: Real-time flight performance with orientation commands for the PD-like controller

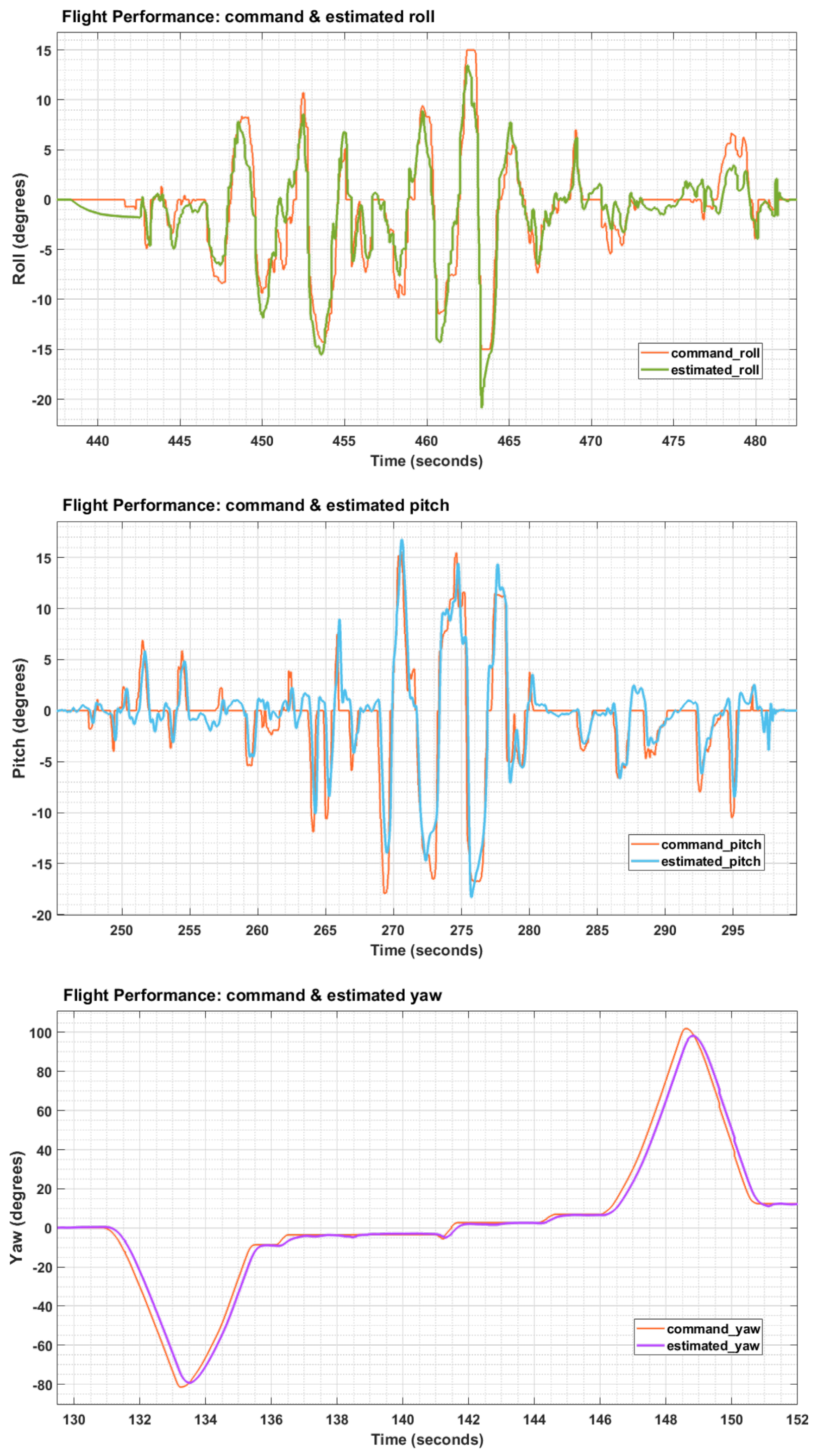


Figure 6.12: Real-time flight performance with orientation commands for the PID-like controller

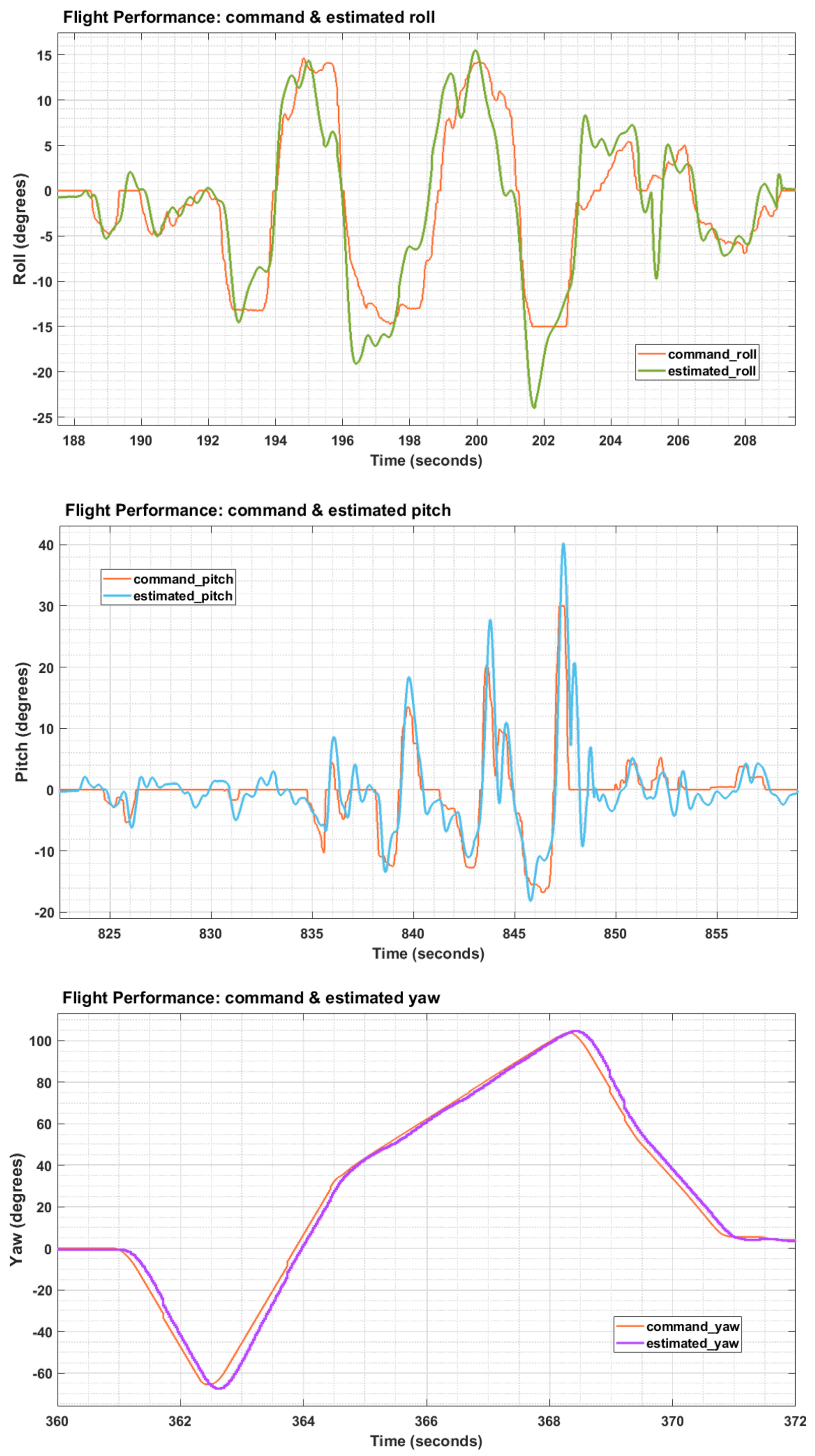


Figure 6.13: Real-time flight performance with orientation commands for the nested PID-like controller

Chapter 7

Conclusion

A rapid-prototyping experimental platform (hardware and software), using Pixhawk 4 and Matlab-Simulink environment, has been developed for the real-time implementation of attitude estimation and control algorithms for quadrotor UAVs. The estimation and control algorithms can be easily implemented using Simulink blocks from which the real-time code is automatically generated and uploaded onto the PX4 autopilot hardware. This mitigates the cost and challenges of quadrotor development as it enables users to devote more time to their control algorithm rather than to the details of programming. A nonlinear attitude estimation algorithm as well as different attitude controllers have been successfully implemented and evaluated.

As future work, we are interested in developing and experimentally validating an attitude control scheme that uses directly the vector measurements without attitude estimation, as well as an attitude control scheme that relies on a single vector measurement.

Bibliography

[1] E. Ebeid, M. Skriver, K. H. Terkildsen, K. Jensen, and U. P. Schultz, "A survey of open-source uav flight controllers and flight simulators," *Microprocessors and Microsystems*, vol. 61, pp. 11–20, 2018.

[2] J. L. Crassidis, F. L. Markley, and Y. Cheng, "Survey of nonlinear attitude estimation methods," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 1, pp. 12–28, 2007.

[3] M. Zamani, J. Trumpf, and R. Mahony, "Nonlinear attitude filtering: A comparison study," *arXiv preprint* arXiv:1502.03990, 2015.

[4] G. Wahba, "A least squares estimate of satellite attitude," *SIAM review*, vol. 7, no. 3, pp. 409–409, 1965.

[5] M. D. Shuster and S. Oh, "Three-axis attitude determination from vector observations," *Journal of Guidance, Control, and Dynamics*, vol. 4, no. 1, pp. 70–77, 1981.

[6] F. L. Markley, "Attitude determination using vector observations and the singular value decomposition," *The Journal of the Astronautical Sciences*, vol. 36, no. 3, pp. 245–258, 1988.

[7] S. F. Schmidt, "The Kalman filter - Its recognition and development for aerospace applications," *Journal of Guidance, Control, and Dynamics*, vol. 4, no. 1, pp. 4–7, 1981.

[8] J. L. Farrell, "Attitude determination by Kalman filtering," *Automatica*, vol. 6, no. 3, pp. 419–430, 1970.

[9] E. J. Lefferts, F. L. Markley, and M. D. Shuster, "Kalman filtering for spacecraft attitude estimation," *Journal of Guidance, Control, and Dynamics*, vol. 5, no. 5, pp. 417–429, 1982.

[10] F. L. Markley, "Attitude error representations for kalman filtering," *Journal of guidance, control, and dynamics*, vol. 26, no. 2, pp. 311–317, 2003.

[11] I. Bar-Itzhack and Y. Oshman, "Attitude determination from vector observations: quaternion estimation," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 321, no. 1, pp. 128–136, 1985.

[12] S. Salcudean, "A globally convergent angular velocity observer for rigid body motion," *Automatic Control, IEEE Transactions on*, vol. 36, no. 12, pp. 1493–1497, 1991.

[13] N. Metni, J.-M. Pflimlin, T. Hamel, and P. Souères, "Attitude and gyro bias estimation for a flying uav," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 1114–1120, IEEE, 2005.

[14] T. Hamel and R. Mahony, "Attitude estimation on SO(3) based on direct inertial measurements," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 2170–2175, IEEE, 2006.

[15] N. Metni, J.-M. Pflimlin, T. Hamel, and P. Souères, "Attitude and gyro bias estimation for a VTOL UAV," *Control Engineering Practice*, vol. 14, no. 12, pp. 1511–1520, 2006.

[16] A. Tayebi, S. McGilvray, A. Roberts, and M. Moallem, "Attitude estimation and stabilization of a rigid body using low-cost sensors," in *Decision and Control, 2007 46th IEEE Conference on*, pp. 6424–6429, IEEE, 2007.

[17] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *Automatic Control, IEEE Transactions on*, vol. 53, no. 5, pp. 1203–1218, 2008.

[18] R. Mahony, T. Hamel, J. Trumpf, and C. Lageman, "Nonlinear attitude observers on SO(3) for complementary and compatible measurements: A theoretical study," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pp. 6407–6412, IEEE, 2009.

[19] P. Martin and E. Salaün, "Design and implementation of a low-cost observer-based attitude and heading reference system," *Control Engineering Practice*, vol. 18, no. 7, pp. 712–722, 2010.

[20] A. Tayebi, A. Roberts, and A. Benallegue, "Inertial measurements based dynamic attitude estimation and velocity-free attitude stabilization," in *American Control Conference (ACC), 2011*, pp. 1027–1032, IEEE, 2011.

[21] A. Roberts and A. Tayebi, "On the attitude estimation of accelerating rigid-bodies using GPS and IMU measurements," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pp. 8088–8093, IEEE, 2011.

[22] M.-D. Hua, G. Ducard, T. Hamel, R. Mahony, and K. Rudin, "Implementation of a nonlinear attitude estimator for aerial robotic vehicles," *Control Systems Technology, IEEE Transactions on*, vol. 22, no. 1, pp. 201–213, 2014.

[23] M.-D. Hua, K. Rudin, G. Ducard, T. Hamel, and R. Mahony, "Nonlinear attitude estimation with measurement decoupling and anti-windup gyro-bias compensation," in *IFAC World Congress*, pp. 2972–2978, 2011.

[24] S. Berkane, A. Abdessameud, and A. Tayebi, "Hybrid attitude and gyro-bias observer design on $so(3)$," *IEEE Transactions on Automatic Control*, vol. 62, no. 11, pp. 6044–6050, 2017.

[25] J.-Y. Wen and K. Kreutz-Delgado, "The attitude control problem," *Automatic Control, IEEE Transactions on*, vol. 36, no. 10, pp. 1148–1162, 1991.

[26] N. Chaturvedi, A. K. Sanyal, and N. H. McClamroch, “Rigid-body attitude control,” *Control Systems Magazine, IEEE*, vol. 31, no. 3, pp. 30–51, 2011.

[27] R. Mahony, V. Kumar, and P. Corke, “Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor,” *IEEE Robotics and Automation Magazine*, no. 19, pp. 20–32, 2012.

[28] S. Bouabdallah, A. Noth, and R. Siegwart, “PID vs LQ control techniques applied to an indoor micro quadrotor,” in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2451–2456, IEEE, 2004.

[29] J. Li and Y. Li, “Dynamic analysis and PID control for a quadrotor,” in *Mechatronics and Automation (ICMA), 2011 International Conference on*, pp. 573–578, IEEE, 2011.

[30] E. Reyes-Valeria, R. Enriquez-Caldera, S. Camacho-Lara, and J. Guichard, “LQR control for a quadrotor using unit quaternions: Modeling and simulation,” in *Electronics, Communications and Computing (CONIELECOMP), 2013 International Conference on*, pp. 172–178, IEEE, 2013.

[31] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D’Andrea, “A platform for aerial robotics research and demonstration: The flying machine arena,” *Mechatronics*, vol. 24, no. 1, pp. 41–54, 2014.

[32] A. Benallegue, A. Mokhtari, and L. Fridman, “Feedback linearization and high order sliding mode observer for a quadrotor UAV,” in *Variable Structure Systems, 2006. VSS’06. International Workshop on*, pp. 365–372, IEEE, 2006.

[33] I. Voos, “Nonlinear control of a quadrotor micro-UAV using feedback-linearization,” in *Mechatronics, 2009. ICM 2009. IEEE International Conference on*, pp. 1–6, IEEE, 2009.

[34] D. Lee, H. J. Kim, and S. Sastry, “Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter,” *International Journal of control, Automation and systems*, vol. 7, no. 3, pp. 419–428, 2009.

[35] S. Bouabdallah and R. Siegwart, “Backstepping and sliding-mode techniques applied to an indoor micro quadrotor,” in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 2247–2252, IEEE, 2005.

[36] T. Madani and A. Benallegue, “Backstepping control for a quadrotor helicopter,” in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 3255–3260, IEEE, 2006.

[37] Y.-C. Choi and H.-S. Ahn, “Nonlinear control of quadrotor for point tracking: Actual implementation and experimental tests,” *Mechatronics, IEEE/ASME Transactions on*, vol. 20, no. 3, pp. 1179–1192, 2015.

[38] F. Goodarzi, D. Lee, and T. Lee, “Geometric nonlinear PID control of a quadrotor UAV on SE(3),” in *Control Conference (ECC), 2013 European*, pp. 3845–3850, IEEE, 2013.

[39] A. Tayebi and S. McGilvray, "Attitude stabilization of a VTOL quadrotor aircraft," *Control Systems Technology, IEEE Transactions on*, vol. 14, no. 3, pp. 562–571, 2006.

[40] F. Lizarralde and J. T. Wen, "Attitude control without angular velocity measurement: A passivity approach," *Automatic Control, IEEE Transactions on*, vol. 41, no. 3, pp. 468–472, 1996.

[41] P. Tsiotras, "Further passivity results for the attitude control problem," *IEEE Transactions on Automatic Control*, vol. 43, no. 11, pp. 1597–1600, 1998.

[42] M. R. Akella, "Rigid body attitude tracking without angular velocity feedback," *Systems & Control Letters*, vol. 42, no. 4, pp. 321–326, 2001.

[43] A. Tayebi, "Unit quaternion-based output feedback for the attitude tracking problem," *Automatic Control, IEEE Transactions on*, vol. 53, no. 6, pp. 1516–1520, 2008.

[44] A. Tayebi, A. Roberts, and A. Benallegue, "Inertial vector measurements based velocity-free attitude stabilization," *Automatic Control, IEEE Transactions on*, vol. 58, no. 11, pp. 2893–2898, 2013.

[45] D. Thakur and M. R. Akella, "Gyro-free rigid-body attitude stabilization using only vector measurements," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 811–818, 2014.

[46] L. Benziane, A. Benallegue, and A. Tayebi, "Attitude stabilization without angular velocity measurements," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 3116–3121, IEEE, 2014.

[47] L. Benziane, A. Benallegue, Y. Chitour, and A. Tayebi, "Velocity-free attitude stabilization with inertial vector measurements," *International Journal of Robust and Nonlinear Control*, in press, 2015.

[48] M. D. Shuster, "A survey of attitude representations," *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.

[49] P. Singla, D. Mortari, and J. L. Junkins, "How to avoid singularity for Euler angle set?," in *Proceedings of the AAS Space Flight Mechanics Conference, Hawaii*, 2004.

[50] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, pp. 15–16, 2006.

[51] S. P. Bhat and D. S. Bernstein, "A topological obstruction to continuous global stabilization of rotational motion and the unwinding phenomenon," *Systems & Control Letters*, vol. 39, no. 1, pp. 63–70, 2000.

[52] R. Mahony and T. Hamel, "Adaptive compensation of aerodynamic effects during takeoff and landing manoeuvres for a scale model autonomous helicopter," *European Journal of Control*, vol. 7, no. 1, pp. 43–57, 2001.

[53] T. Hamel, R. Mahony, R. Lozano, and J. Ostrowski, "Dynamic modeling and configuration stabilization for an X4-flyer," *Proceedings of the 15th IFAC World Congress*, vol. 15, no. 1, pp. 846–846, 2002.

[54] P. E. Crouch, “Spacecraft attitude control and stabilization: Applications of geometric control theory to rigid body models,” *Automatic Control, IEEE Transactions on*, vol. 29, no. 4, pp. 321–331, 1984.

[55] B. Mccormick, *Aerodynamics, Aeronautics, and Flight Mechanics.* Wiley, Second Edition, 1995.

[56] J. G. Leishman, *Principles of Helicopter Aerodynamics with CD Extra.* Cambridge, U.K: Cambridge Univ. Press., 2006.

[57] M. Krznar, D. Kotarski, P. Piljek, and D. Pavković, “On-line inertia measurement of unmanned aerial vehicles using on board sensors and bifilar pendulum,” *Interdisciplinary Description of Complex Systems*, vol. 16, pp. 149–161, 01 2018.

[58] A. Shaheen and M. S. Anwar, “A doubly suspended pendulum,” 2017.

[59] C. C. Foster and G. H. Elkaim, “Extension of a two-step calibration methodology to include nonorthogonal sensor axes,” *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 44, no. 3, pp. 1070–1078, 2008.

[60] M. Barczyk, *Nonlinear state estimation and modeling of a helicopter UAV.* PhD thesis, University of Alberta, 2012.

[61] [PX4 User Guide]. https://docs.px4.io/master/en/.

[62] [Pixhawk4 User Guide]. https://docs.px4.io/master/en/flight_controller/pixhawk4.html.

[63] [UAV Toolbox User Guide]. https://www.mathworks.com/help/pdf_doc/supportpkg/px4/px4_ug.pdf.

[64] [QGroundControl Developer Guide]. http://qgroundcontrol.com/.

[65] [FMU Processer STM32F765 Datasheet]. https://www.mouser.ca/datasheet/2/389/stm32f765bi-2956130.pdf.

[66] [IO Processor STM32F100 Datasheet]. https://www.st.com/en/microcontrollers-microprocessors/stm32f100rb.html.

[67] [Accel/Gyro ICM-20689 Datasheet]. https://www.ic-components.hk/files/db/ICM-20689.pdf.

[68] [Accel/Gyro BMI055 Datasheet]. https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi055/.

[69] [Mag IST8310 Datasheet]. https://intofpv.com/attachment.php?aid=8104.

[70] [GPS User Guide]. http://www.holybro.com/manual/Pixhawk4-GPS-Quick-Start-Guide.pdf.

[71] [ESC Calibration Process]. http://copter.ardupilot.com/wiki/esc-calibration/.

[72] [Dronecode User Guide]. https://www.dronecode.org/.

[73] [MAVLink Developer Guide]. https://mavlink.io/en/.

[74] [uORB Developer Guide]. https://docs.px4.io/main/en/middleware/uorb.html.

[75] L. Meier, D. Honegger, and M. Pollefeys, “Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6235–6240, 2015.

[76] [MAVLink Messaging Tutorial]. https://docs.px4.io/main/en/middleware/mavlink.html.

[77] [QGC Parameter List]. https://docs.px4.io/main/en/advanced_config/parameters.html#parameter-not-in-firmware.

[78] [uORB Tutorial Part1]. https://px4.io/px4-uorb-explained-part-2/.

[79] [uORB Tutorial Part2]. https://px4.io/px4-uorb-explained-part-1/.

[80] [SDI User Guide Part1]. https://www.mathworks.com/help/simulink/slref/simulationdatainspector.htm

[81] [SDI User Guide Part2]. https://www.mathworks.com/help/simulink/ug/viewing-output-trajectories.html.