drones

MDPI

Article

Design and Flight Simulation Verification of the Dragonfly eVTOL Aircraft

Wen Zhao [1], Yingqi Wang [1,*], Liqiao Li [1,*], Fenghua Huang [2,*], Hanwen Zhan [1], Yiqi Fu [1] and Yunfei Song [1]

[1] School of Civil Aviation, Northwestern Polytechnical University, Xi'an 710072, China; zhaowen@nwpu.edu.cn (W.Z.); zhanhanwen@mail.nwpu.edu.cn (H.Z.); fuyiqi@mail.nwpu.edu.cn (Y.F.); 2021264509@mail.nwpu.edu.cn (Y.S.)

[2] Fujian Key Laboratory of Spatial Information Perception and Intelligent Processing, Yango University, Fuzhou 350015, China

* Correspondence: wangyingqi@mail.nwpu.edu.cn (Y.W.); anthony_llq_2@mail.nwpu.edu.cn (L.L.); fhhuang@ygu.edu.cn (F.H.); Tel.: +86-18700492743 (Y.W.)

Abstract: Recently, electric vertical take-off and landing (eVTOL) aircraft have become a top priority for urban air transportation due to their ability to overcome urban ground traffic congestion. In this research, a new type of scaled lift–cruise 'Dragonfly' has been designed. The 'Dragonfly' combines the characteristics of an octocopter and a fixed-wing aircraft. Compared with the same type of eVTOL aircraft, it has a longer wingspan and a more stable aircraft structure, it can not only take off and land vertically without the need for a runway, but also fly quickly in a straight line and hover in mid-air. In order to ensure the success of the flight test, it was also simulated in this paper. A simulation scenario highly fitting with the flight test environment of eVTOL is designed in the Gazebo simulation platform, and then combined with the PX4 flight control platform, the system SITL of the constructed aircraft simulation model is carried out on the Gazebo platform, Finally, simulation flight test data for accurate analysis are obtained, the accuracy and stability of the control algorithm are fed back, and scientific support for the follow-up 'Dragonfly' aircraft hardware-in-the-loop simulation and physical flight test is provided.

Keywords: drone; eVTOL; aircraft design; simulation verification

Citation: Zhao, W.; Wang, Y.; Li, L.; Huang, F.; Zhan, H.; Fu, Y.; Song, Y. Design and Flight Simulation Verification of the Dragonfly eVTOL Aircraft. *Drones* **2024**, *8*, 311. https://doi.org/10.3390/drones8070311

Academic Editor: Abdessattar Abdelkefi

Received: 22 May 2024
Revised: 24 June 2024
Accepted: 5 July 2024
Published: 9 July 2024

1. Introduction

Recently, in order to overcome urban ground traffic congestion, reduce travel time, make effective use of urban airspace, and ultimately achieve safe and efficient passenger transportation in urban areas, electric vertical take-off and landing (eVTOL) aircraft have been proposed as a typical urban air transport vehicle. The aircraft has the characteristics of vertical take-off and landing, high-speed cruising, and short-distance flight. Compared to traditional aircraft, it is pollution-free, low-noise, greener, and sustainable [1–3]. This makes it an ideal product for the future air traffic system and a key solution to many problems in urban transportation. At present, the main types of eVTOL aircraft being developed worldwide can be categorized into three types: vector propulsion, multi-rotor, and lift–cruise.

A vector propulsion eVTOL is a low-altitude aircraft with high commercial value at present. Compared to other configurations, it has a heavy load, long range, and higher cruising speed. And it does not need to change the thrust direction to achieve vertical take-off, landing, and cruising. At present, the main vector propulsion eVTOLs are the 'Vahana' (Boeing) as shown in Figure 1a and the 'E20' aircraft (Tcab Technology Co., Ltd, China) as shown in Figure 1b. A multi-rotor aircraft has the same propulsion device, lacks aerodynamic force, features a simple design, operates at low speeds, carries small loads, and has a limited range. Recently, the typical products mainly include 'Velocity' (Volocopter) as shown in Figure 1c and 'ehang-216' (NASDAQ/EH) aircraft as shown in Figure 1d. A

lift–cruise eVTOL combines the characteristics of rotorcraft and fixed-wing aircraft. The market access time is delayed, and the overall performance combines the advantages of the two mentioned above. The typical prototypes include 'Boing-PAV' (Boeing) as shown in Figure 1e and 'V1500M' (Autoflight) as shown in Figure 1f [4–6].

Figure 1. (**a**) 'Vahana' (Airbus), (**b**) 'E20' (Tcab Tech. Co., Ltd.), (**c**) 'VoloCity' (Volocopter), (**d**) 'ehang-216' (NASDAQ/EH), (**e**) 'Boeing-PAV' (Boeing), (**f**) 'V1500M' (Autoflight).

In this research, the scaled verification 'Dragonfly' eVTOL, which is a new type of lift–cruise aircraft, is proposed and designed. The prototype design of the 'Dragonfly' is shown in Figure 2a, and the physical drawing is shown in Figure 2b. 'Dragonfly' is designed for inter-city transportation. Its unique structure combines the characteristics of eight-rotor and fixed-wing aircraft. Compared with the same type of eVTOL aircraft, it has a longer wingspan and a more stable aircraft structure; it can not only take off and land vertically without the need for a runway, but also fly quickly in a straight line and hover in mid-air. In the future, it will be able to carry two people and hand luggage. At present, the scaled verification aircraft is fully electric-powered, with a range of 15 km, a maximum speed of 20 m/s, eight lift propellers, one thrust propeller, and a maximum endurance of 1 h. The characteristic information of the prototype is shown in Table 1.

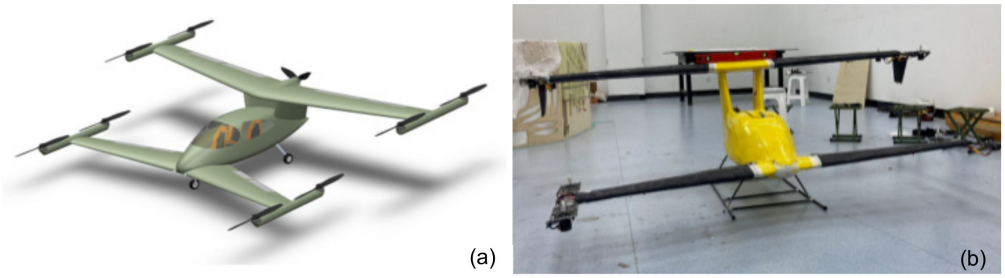

Figure 2. (**a**) The prototype design of the 'Dragonfly' eVTOL; (**b**) the prototype design of the 'Dragonfly' eVTOL.

The overall design of the 'Dragonfly' includes aircraft design, avionics, information and communication, sensors, and materials technology. When the overall, aerodynamic, structural design, and assembly work for the 'Dragonfly' is completed, issues such as improper operation, poor flight control laws, and low robustness during take-off, cruising, and landing due to the lengthy and frequent testing process will inevitably be encountered. At this time, the aircraft under test may experience a sudden drop, posing unpredictable dangers. This situation is not conducive to testing various parameters of the aircraft and may even hinder the progress of the entire project.

Table 1. Basic specifications of Dragonfly aircraft.

Category	Specifications
Type	Lift–cruise
Power	Electricity
Maximum speed	≈20 m/s
Flight range	≈15 km
Effective load	≈1 kg
Propeller	Eight lift propellers and one thrust propeller
Endurance time	≈1 h
Aircraft materials	Compound material

Aircraft simulation software, such as Matlab/Simulink, or flight simulators, like FlightGear, are frequently utilized by developers and researchers to simulate rotor and fixed-wing EVTOL systems. On the one hand, although these simulation tools provide a simulation framework, a lot of work still needs to be performed for advanced control simulation, including sensors [7]. On the other hand, commercial simulators have the limitation of providing users with only limited functions and restricting the modification or enhancement of these functions. Due to these limitations, ROS/Gazebo provides a modular and standardized simulation framework in the development of robotics. It also has an active open-source community that promotes the integration of resources. ROS/Gazebo is a powerful tool that can assist researchers in easily reconfiguring and simulating the availability of hardware components, such as sensors. It also allows for flexible environment settings, all at a low cost, low risk, and high reliability. Gazebo is widely used in the field of ground robot simulation, particularly for manipulator control in industrial settings. Through the modeling of the manipulator in Gazebo and the utilization of ROS programming technology, the manipulator is able to perform grasping, rotation, and other actions. The effectiveness of the algorithm is verified by outputting dynamic simulation results [8]. In addition, ROS/Gazebo can easily build a working scene where service robots are used in shopping malls, airports, banks, and other places. It enables the simulation of task planning, obstacle avoidance, and navigation control in this environment [9]. It is critical and necessary to use ROS/Gazebo simulation technology to conduct simulation tests prior to the flight testing of eVTOL aircraft.

Recently, with the development of EVTOL technology and the increasing design requirements, ROS and Gazebo have become popular open-source and free tools for EVTOL development. They are widely used in various areas such as EVTOL control system, navigation system, and power system development and simulation. Unmanned aerial vehicles can also be regarded as a special kind of aerial robot. Compared to ground-based mobile robots, the impact of aerodynamics on aircraft in high-speed, high-altitude flight cannot be ignored, in addition to their basic kinematic properties. Gazebo, on the other hand, is widely used in simulation experiments of EVTOL with aerodynamic requirements because it can be set up with relevant physics engines, so it can provide more realistic feedback on the physical scene. Relevant research in China includes the use of ROS/Gazebo technology to simulate the autonomous landing and control system of EVTOLs [10], the simulation of multi-rotor EVTOL cluster formation control systems [11], and the simulation of flight control and obstacle avoidance systems [12]. Meanwhile, foreign researchers K. D. Nguyen et al. used ROS/Gazebo to conduct a vision-based EVTOL software-in-the-loop simulation [13]. C. McCord et al. conducted a distributed formation control simulation of multiple EVTOLs in the ROS/Gazebo environment [14]. S. Khaliq et al. used Gazebo to simulate the multi-platform hardware-in-the-loop simulation of multi-EVTOL distributed group communication [15]. In brief, ROS/Gazebo is widely used by the majority of aircraft researchers and designers due to its open-source characteristics and powerful functions.

This platform is of great significance in simulating eVTOL aircraft before conducting flight tests.

The focus of this research was on the design and simulation of a new eVTOL aircraft called 'Dragonfly'. The simulation model and scene are built using ROS/Gazebo. In the simulation scenario, the system software uses the flight control algorithm based on quaternion PID and the Px4 flight control platform to simulate the aircraft model in the loop-in-the-loop simulation. Through the Gazebo platform and ROS visualization program, real-time sensor and attitude data can be obtained. In this paper, a new type of eVTOL aircraft with a longer wingspan, more stable structure, stronger endurance, and better control performance is designed, and a safer and more reliable simulation experiment method combined with the Gazebo&PX4 platform is designed. This research method is very effective and necessary for the low-cost design and testing of future aircraft.

2. ROS/Gazebo Simulation Platform

Gazebo is a free and open-source 3D robot simulator that can be utilized for algorithm testing and robot design. It also allows for the creation of intricate indoor and outdoor environments for robots. Furthermore, the platform features a powerful physics engine, high-quality graphics display capability, practical programming, and a graphical interface.

ROS is a flexible framework for robot programming that can integrate multiple plugins and provide a communication architecture for it. There are three levels of ROS architecture: the OS layer, the middle layer, and the application layer. The OS layer of ROS relies on the Linux operating system, so the Ubuntu system, which is officially supported by ROS, is widely used in the OS layer. In the middle layer, ROS primarily relies on the TCP/UDP ROS communication system to facilitate data transmission using various communication mechanisms. Based on the communication mechanism, ROS provides various libraries for robot development, including coordinate transformation and motion control. The smallest processing unit in ROS is the node. In the application layer, ROS designates the master as the central management center for the nodes. The master is responsible for ensuring the proper functioning of each node and the entire system. The communication mechanism between nodes is divided into topic-based publish/subscribe communication, service-based publish/subscribe communication, and an RPC-based parameter server.

The eVTOL aircraft integrates aircraft design technology, avionics technology, information and communication technology, sensor technology, and material technology, and it is a high-tech, complex-design, and expensive aircraft. When conducting tests such as take-off, cruise, and landing on eVTOL aircraft that have completed the overall, aerodynamic, structural design and assembly, due to the long and frequent testing process, it is inevitable that there will be improper operation, poor flight control law, and low robustness, which may lead to the fall of the aircraft under test, causing unpredictable dangers, which is not conducive to the testing of various parameters of the aircraft and even slows down the progress of the entire engineering project.

Aircraft simulation software (e.g., Matlab/Simulink) or flight simulators (e.g., FlightGear) are often used by developers or researchers to simulate and simulate rotor- and fixed-wing EVTOL systems. On the one hand, although these simulation tools provide a framework for simulation, there is still a lot of work to be performed for advanced control simulation, including sensors. Business simulators, on the other hand, are characterized by offering only limited features to users and restricting modifications or enhancements. Due to these limitations, ROS/Gazebo provides a modular standard simulation framework in the development of the robotics field and has an active open-source community that facilitates the integration of resources. ROS/Gazebo is a powerful tool that allows researchers to easily reconfigure and simulate the availability of hardware components such as sensors and flexibly set up environments with low cost, low risk, and high reliability. It is crucial and necessary to use ROS/Gazebo simulation technology to simulate the whole robot before the flight test of the eVTOL aircraft.

Robot developers use the URDF function package provided by ROS to simulate and develop robots. This function includes a C++ parser for parsing URDF files. URDF is a file in XML format, composed of special XML tags. These XML tags are used to describe the components, joints, and dimensions of the robot, as well as the kinematics and dynamic description, visual shape, and collision model of the robot [16–18].

When using URDF to describe a robot, the rigid links of the robot can be seen as a tree structure. This means that the robot consists of rigid links connected by joints. However, URDF cannot represent flexible links. The XML tag of URDF can support the modeling of the robot. Therefore, a file must be created to define the relationship between each rigid link and joint in the robot. The file should be saved with the extension '.urdf'. The following are commonly used tags to create a URDF robot model [19]:

(a) <link> label: It represents the rigid body of a part of the robot. Using this tag for 'Dragonfly', the appearance and physical properties of the robot's rigid body can be established including size, shape, and color, and a 3D mesh can even be imported to represent the appearance of the rigid body. The <link> can also provide rigid body inertia parameters and collision attributes. One <link> includes three attributes: <visual>, <inertial>, and <collision>. Among them, <visual> can describe the appearance of the link and represent the real link of a robot. < Inertial> can describe the inertia parameters of the link. The area around the real link is the <collision> attribute, and this attribute is wrapped with the real link, to support aircraft in detecting the collision before hitting the link.

(b) <joint> label: It refers to an aircraft model joint, which is used to describe the kinematic and dynamic characteristics of the joint. At the same time, it can also determine the position and speed of the joint's motion based on the parent link and the child link. <Joint> tags support various types of joint motion, including revolution, continuous, prismatic, fixed, floating, and planar.

(c) <robot> label: The <robot> tag represents the whole aircraft model and belongs to the top tag of the model. Both <link> and <joint> tags are required to be included in the <robot> tag, which can be considered as a sub-tag of <robot>. In addition, a complete robot is composed of a series of <link> and <joint> tags.

(d) <gazebo> label: URDF can only specify the kinematic and dynamic characteristics of a single aircraft model. It cannot specify the position of the model itself in the Gazebo environment and lacks attributes such as friction and elasticity. Moreover, URDF cannot describe attributes such as lighting and altitude maps. Therefore, Gazebo provides a Simulation Description Format (SDF) to solve the shortcomings of URDF. SDF is the default format supported by Gazebo. Therefore, to use URDF files in Gazebo, some additional special simulation labels must be added to ensure proper functionality with Gazebo. The labels typically describe the parameters necessary for simulation in Gazebo, including model materials, plugins, etc.

3. Simulation Model Construction

A. Three-Dimensional and URDF Model

In order to accurately depict the external features of the 'Dragonfly' aircraft on a simulation platform, it is necessary to first design the 3D model of the 'Dragonfly' in order to construct the aircraft simulation model. When using Gazebo to simulate the 'Dragonfly', its internal detailed structures, such as wing ribs, trusses, and connectors, can be ignored. Referring to the overall and structural design of the aircraft, the geometric parameters of the aircraft's fuselage, wings, control surfaces, propulsion systems, and other components can be determined. The 3D model of the aircraft was created using solid filling in SolidWorks 2014 software, as shown in Figure 3.

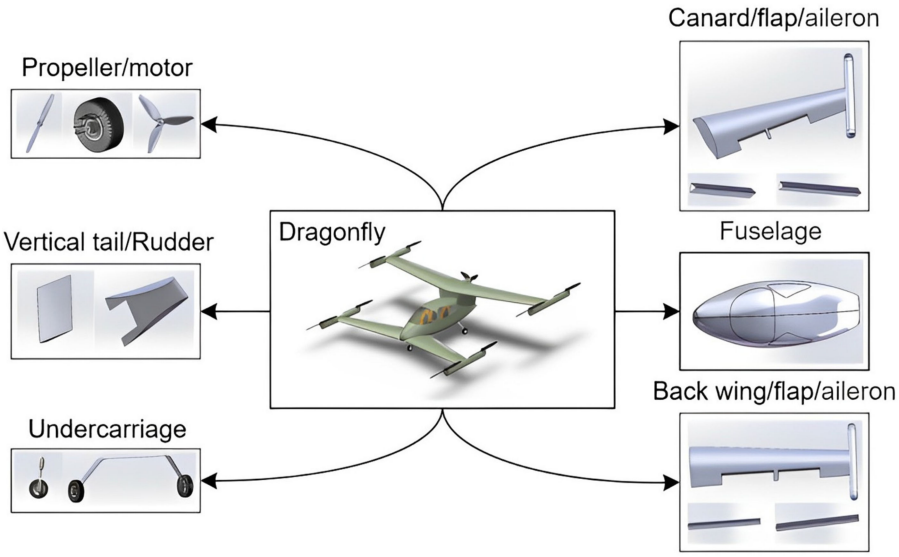

Figure 3. Three-dimensional model diagram of 'Dragonfly' aircraft.

Similar to robots, aircraft can also be described using URDF. The system consists of various complex mechanisms, including actuators, drive systems, sensor systems, and control systems. Meanwhile, it also has the ability of perception, planning, action, and coordination.

In order to enhance simulation speed and accuracy, the 3D structure of the aircraft should be further simplified. Except for the propeller and actuating surfaces, all other structures are considered part of a rigid body. The simplified basic components include the body, propeller, flap, aileron, and rudder, as shown in Figure 4.

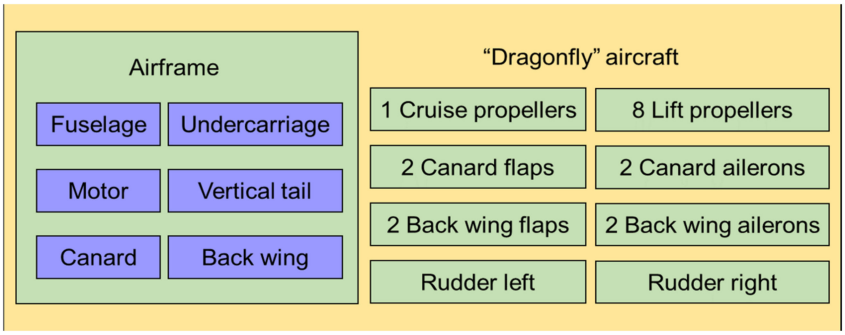

Figure 4. Structural components of simplified aircraft model.

Based on the simplified model, the reference axis of each rigid body is further set in SolidWorks. Additionally, the rigid body model file can be exported in the 'stl' file format. Among them, each rigid body is described with a <link> label. The 'stl' file is an important parameter of the sub-tag <mesh> of the visual tag <visual> in the <link> tag. The connection, relationship, and motion type of rigid bodies can be described using the term "joint". The control surface of the aircraft rotates at a specific angle as a result of the propeller's continuous rotation. As a result, the primary types of joint motion used are mainly continuous and rotational. According to the simplified URDF model, the body is referred to as the 'base_link'. Propellers, flaps, ailerons, and other structures are considered as sub-links, which are connected to the 'base_link' through a <joint>. According to the overall structure of the URDF model of the 'Dragonfly' aircraft shown in Figure 5, the entire body consists of 20 links and 19 joints.

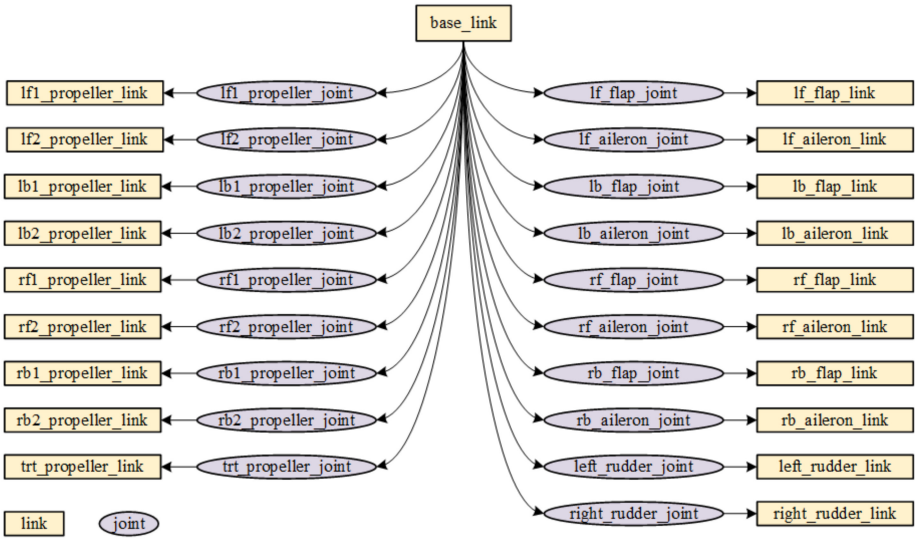

Figure 5. Overall structure diagram of aircraft URDF model.

Taking the connection between 'base_link' and 'lf1_propeller_link' as an example, XML tags are used to describe the framework and code architecture of the <visual> visual attributes, as shown in Figure 6. Other links and joints of the aircraft URDF model are implemented in the same logical manner as described above.

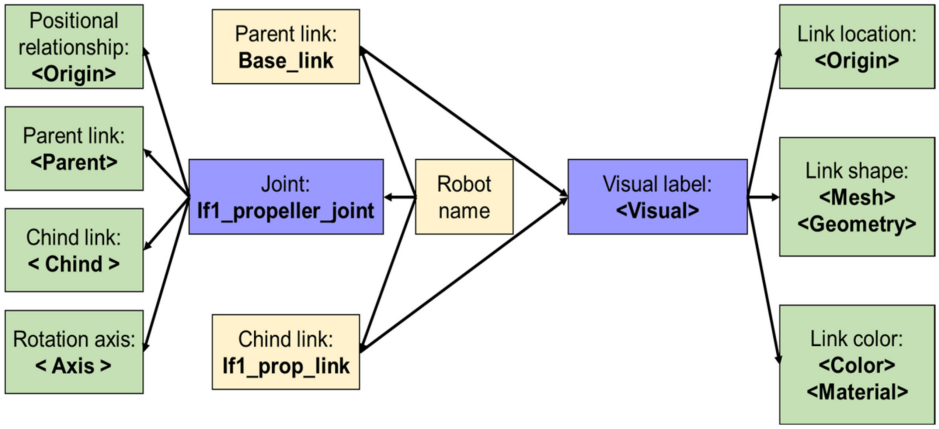

Figure 6. 'base_ link' and 'lf1_ prop_ link' connection diagram.

B. Improvements to the 'Dragonfly' URDF model

In order to accurately display and simulate the model of a dragonfly aircraft in Gazebo, inertia and collision attributes must be added to the URDF model. In particular, the label <inertia> must be added, and the configuration of inertia parameters primarily involves mass and inertia matrix.

According to the hierarchical relationship between the labels of the URDF model shown in Figure 7, all rigid body links are added with <inertia> and <collision> labels at the same level as the <visual> label. The URDF model of the aircraft is basically completed, and it is loaded into Gazebo as shown in Figure 8. In addition, Gazebo does not support URDF's description of its material properties; that is, the <material> and <color> labels do not work.

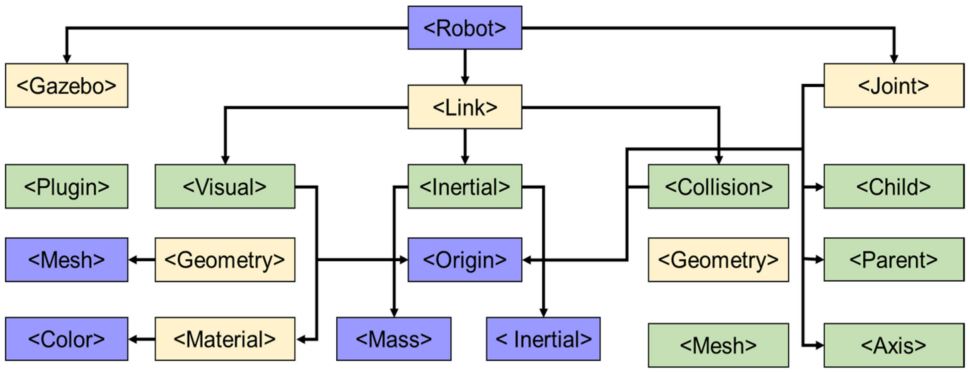

Figure 7. URDF model label hierarchy of aircraft.

Figure 8. Display interface of aircraft simulation model in Gazebo.

In order to address the issue of lengthy description code in URDF models, ROS offers an enhanced version of URDF called Xacro. Xacro is a compact, reusable, and modular description format. Xacro can define constants and function blocks by creating macro definitions. It also supports programmable interfaces, such as constants, variables, mathematical formulas, and conditional statements. This allows for code reuse, reducing the amount of code and making the model code modular and readable. Therefore, in this article, we use the Xacro format to describe the 'Dragonfly' aircraft.

C. Dynamic Model

For the 'Dragonfly' aircraft, there is a close relationship between the dynamic system and the control system. A more accurate dynamic model must be established to ensure the effectiveness of the data output by the control system during the development and testing process.

For dynamic simulation, Gazebo provides a motor model plugin called 'gazebo_motor _model' for common rotorcraft, and a pneumatic model plugin called 'liftdragplugin' for fixed-wing aircraft. In particular, the 'gazebo_motor_model' involves parameters such as the pull coefficient, torque coefficient, and maximum speed of the propeller, which are provided by the propeller. Through macro definition, 'gazebo_motor_model' is defined as a standard module, allowing the aircraft simulation model to be used in rotor mode.

In order to obtain the position information of the 'Dragonfly' aircraft, the dynamic equation of the center of mass movement of the EVTOL is established in the geographic system as follows:

$$m\begin{bmatrix}\dot{V}_{xg}\\ \dot{V}_{yg}\\ \dot{V}_{zg}\end{bmatrix} = L_b^B\begin{bmatrix}T_x\\ T_y\\ -T_z\end{bmatrix} + L_a^s\begin{bmatrix}-D\\ Y\\ -L\end{bmatrix} + \begin{bmatrix}0\\ 0\\ mg\end{bmatrix} \tag{1}$$

In Equation (1), m is the mass of the drone; $\dot{V}_{xg}, \dot{V}_{yg}, \dot{V}_{zB}$ are the triaxial components of the velocity of the 'Dragonfly' aircraft in the geographic system, respectively. L_b^g, L_a^g represent the transformation matrices from the body coordinate system to the ground coordinate system and the velocity coordinate system to the ground coordinate system, respectively, as follows:

$$L_b^{\varepsilon} = \begin{bmatrix} c_\theta c_\phi & c_\theta s_\phi & -s_\theta \\ -c_\varphi s_\phi + s_\varphi s_\theta c_\phi & c_\varphi c_\phi + s_\varphi s_\theta s_\phi & s_\varphi c_\theta \\ s_\varphi s_\phi + c_\varphi s_\theta c_\phi & -s_\varphi c_\phi + c_\varphi s_\theta s_\phi & c_\varphi c_\theta \end{bmatrix} \tag{2}$$

$$L_a^E = \begin{bmatrix} c_\gamma c_\chi & -s_\gamma c_\chi c + s_\chi s & s_\gamma c_\chi s + s_\chi c \\ s_\gamma & c_\gamma c & -c_\gamma s \\ -c_\gamma s_\chi & s_\gamma s_X c + c_\chi s & -s_\gamma s_X s + c_\chi c \end{bmatrix} \tag{3}$$

According to the moment of momentum theorem, the dynamic equation for the rotation of the 'Dragonfly' aircraft around the center of mass in the coordinate system of the body is

$$\begin{cases} p = I_x^{-1}\left[\left(I_y - I_z\right)qr + l_a + l_p\right] \\ q = I_y^{-1}\left[\left(I_z - I_x\right)pr + m_a + m_p\right] \\ r = I_z^{-1}\left[\left(I_x - I_y\right)pq + n_a + n_p\right] \end{cases} \tag{4}$$

In Equation (4), p, q, r are the roll, pitch, and yaw angular velocities of the 'Dragonfly' aircraft in the coordinate system of the fuselage, respectively. I_x, I_y, I_z are the moments of inertia of the EVTOL.

From the mutual conversion relationship between coordinate systems, the following kinematic equations can be derived:

$$\begin{bmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & \sin\varphi\tan\theta & \cos\varphi\tan\theta\\ 0 & \cos\varphi & -\sin\varphi\\ 0 & \sin\varphi\sec\theta & \cos\varphi\sec\theta\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{5}$$

$$\begin{cases} V = \sqrt{V_{xg}^2 + V_{yg}^2 + V_{zg}^2} \\ \gamma = \sin^{-1}\left(V_{zg}/\sqrt{V_{xg}^2 + V_{yg}^2 + V_{zg}^2}\right) \\ \gamma = \tan^{-1}\left(V_{yg}/V_{xg}\right) \end{cases} \tag{6}$$

$$\begin{cases} s_\beta = c_\chi c_\gamma\left(c_\phi s_\theta s_\varphi - s_\psi c_\varphi\right) + s_\chi c_\gamma\left(s_\phi s_\theta s_\varphi + c_\phi c_\varphi\right) - \\ s_\gamma c_\theta s_\varphi \\ s_\alpha c_\beta = c_\chi c_\gamma\left(c_\phi s_\theta c_\varphi - s_\phi s_\varphi\right) + s_\chi c_\gamma\left(s_\phi s_\theta c_\varphi - c_\psi s_\varphi\right) - \\ s_\gamma c_\theta c_\varphi \\ s = \left[c_\alpha s_\beta s_\theta - \left(s_\alpha s_\beta c_\varphi - c_\beta s_\varphi\right)c_\theta\right]/c_\gamma \end{cases} \tag{7}$$

In Equations (5)–(7), φ, θ, ψ are the roll, pitch, and yaw angles of the 'Dragonfly' aircraft, respectively; p, q, r are the triaxial components of the angular velocity vector of the EVTOL in the coordinate system of the airframe, respectively. α, β, are the angle of attack, sideslip angle, and track roll angle of the EVTOL, respectively; γ, χ are the trajectory inclination and the trajectory declination of the drone, respectively.

Figure 9 shows the angle of attack (AOA) and coefficient of lift (C) curves of the Dragonfly after the installation of the 'liftdragplugin' plugin, where the red is the transformation of the lift coefficient with the angle of attack, and the green color fits the curve into two segments; the first half is the normal condition, and the second half is the stall condition. Here, the key parameters are as follows:

CLa: lift curve slope;
$C_{La,stall}$: slope of the lift curve after stalling;
α_{stall}: stall angle of attack.

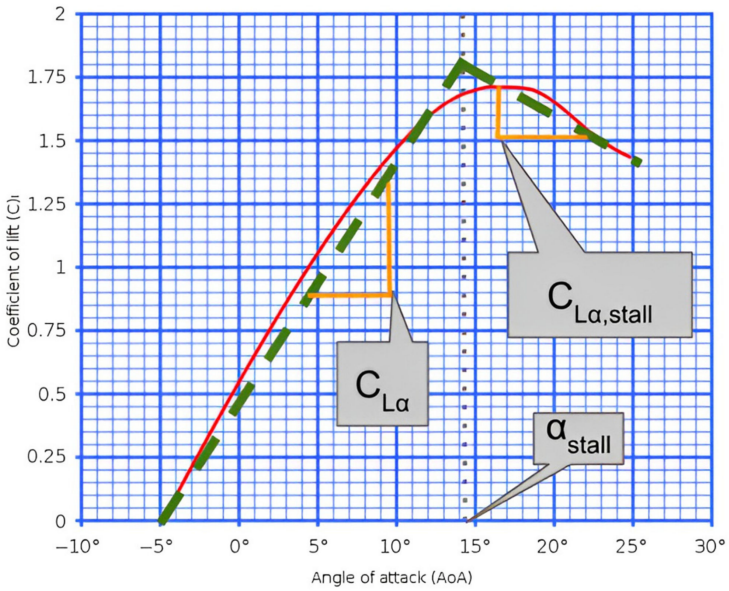

Figure 9. Diagram of the angle of attack and coefficient of lift.

In the small angle of attack range, the lift coefficient and torque coefficient of the aircraft exhibit a linear relationship with the angle of attack, whereas the drag coefficient demonstrates a nonlinear variation with the angle of attack. When the angle of attack increases beyond a certain value, the aircraft will stall.

According to this principle, the 'liftdragplugin' provided by Gazebo linearizes the curve of the aerodynamic coefficient of the airfoil, which varies with the angle of attack, into two segments. This linearization occurs at the position where the stall angle of attack occurs.

In this research, the RONCZ1082 airfoil is used for the wing of the aircraft, while the NACA0015 airfoil is used for the vertical tail. According to the aerodynamic characteristic curve of the airfoil, the overall and aerodynamic design of the aircraft calculates the lift coefficient, drag coefficient, torque coefficient, slope of the angle, and other parameters in the 'liftdragplugin'. Then, the plugins and input parameters are added to the URDF model file of this aircraft through the <gazebo> and <plugin> labels [20,21].

The "Dragonfly" flight control system comprises two main flight control computers that are redundant and heterogeneous with each other and two auxiliary flight control computers that are redundant and heterogeneous with each other, and also includes a flight mission module, a state performance detection module, a flight mode switching module, a real-time flight guidance module, a flight protection module, and a control law matching module connected with the main flight control computer and the auxiliary flight control computer. This is shown in Figure 10.

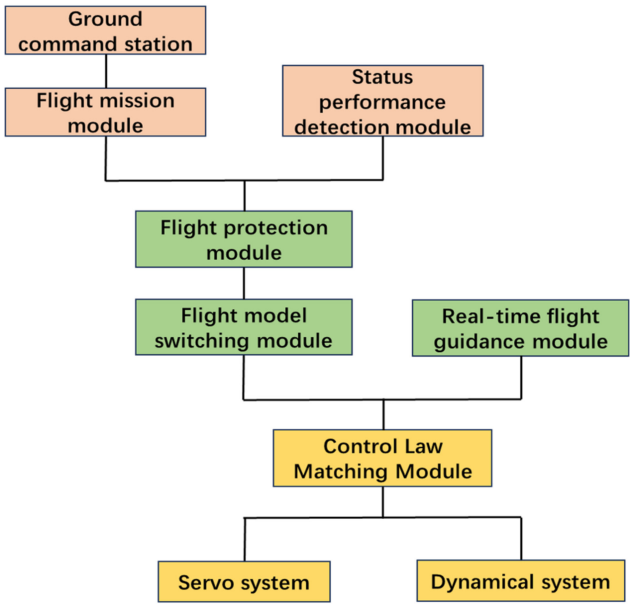

Figure 10. Schematic diagram of the principal architecture of the "Dragonfly" flight control system.

The flight mission module is used to generate flight route planning tasks and conduct virtual demonstrations of the generated flight route planning tasks to obtain flight route mission instructions that do not conflict with the flight environment.

The state performance detection module is used to detect the state performance parameters of the EVTOL aircraft.

The flight protection module is used to calculate whether the state performance parameters of the EVTOL aircraft meet the requirements of the flight route mission command and optimize the flight route mission command.

The flight mode switching module switches the EVTOL aircraft to the corresponding flight mode according to the flight mode requirements in the flight route mission command.

The real-time flight guidance module collects the real-time flight parameters and external environment data of the EVTOL aircraft and generates real-time guidance instructions according to the real-time flight parameters and external environment data.

The control law matching module comprehensively solves the real-time guidance instructions and flight route mission instructions to generate control law instructions and controls the servo system and power system of the EVTOL aircraft in real time through the control law instructions.

D. Sensor Plugin

When the 'Dragonfly' aircraft flies in a real physical environment, the control system requires sensor data, such as the current attitude, speed, position, temperature, air pressure, and other parameters from the environment, to be obtained. After being processed by the control system, the control commands are outputted to control the aircraft and complete the flight task according to the planned route. In this research, simulating control in the Gazebo environment also necessitates sensor data. Therefore, it is essential to incorporate analog sensors into the aircraft simulation model.

The Gazebo environment supports the simulation of sensor data and noise simultaneously. In this environment, the observation model of the sensor can be described by the following equations:

$$V_m = V_r + E_d + N_0, \tag{8}$$

$$E_d = N_b \tag{9}$$

Here, V_m refers to the measured value, V_r indicates the actual value, E_d represents the drift error, N_0 represents Gaussian noise, and N_b represents the rate of change in drift error over time.

Gazebo provides eight types of sensor plugins, including magnetometer, IMU, barometer, and more, as shown in Table 2. 'Xacro' is used to define the input parameters of these sensor data in the form of a macro definition. This method makes it easy to modify the input parameters of the sensor plugin, and it is also convenient for using other simulation models to enhance the multiplex code rate. According to the labels shown in Figure 11, the IMU is connected to the simulation model of the aircraft using the <gazebo> and <plugin> labels.

Table 2. Sensor plugin provided by Gazebo.

Category	Specifications
Inertial measurement unit (IMU)	gazebo_imu_plugin
GPS	gazebo_gps_plugin
Barometer	gazebo_barometer_plugin
Magnetometer	gazebo_magnetormeter_plugin
Camera	gazebo_ros_camera
Lidar	gazebo_ros_laser
Binocular camera	gazebo_ros_multicamera
Depth camera	gazebo_ros_openni_kinect

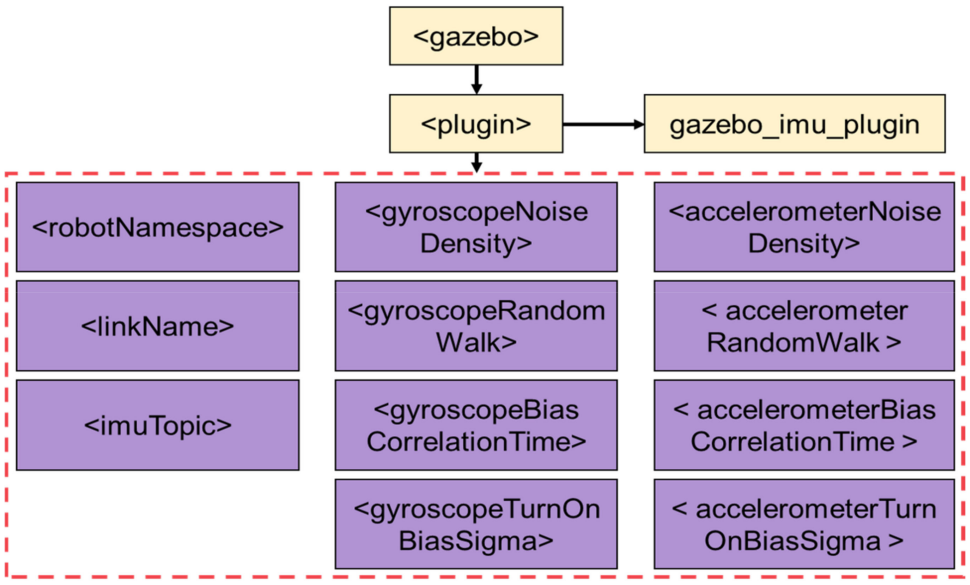

Figure 11. Add sensors by using <gazebo> and <plugin> labels.

As shown in Figure 11, the contents within the <plugin> tag are the input parameters for the sensor plugin. These parameters include the name of the published IMU topic, the link name, the noise value, etc. In addition, airspeed meters, magnetometers, and barometers can be directly used through the <gazebo> and <plugin> labels. In particular, the noise settings and parameter meanings of gyroscopes, accelerometers, and magnetometers are associated with the parameters in the sensor model, as illustrated in Table 3.

Table 3. Characteristics of sensor noise parameters.

Parameters	Specifications
Noise Density	Standard deviation of Gaussian noise
Bias Correlation Time	Relevant time of drift error
Turn-On Bias Sigma	Initial standard deviation of drift error

After adding sensors, it is necessary to load the simulation model of the aircraft into the Gazebo 11 software and select the sensor topic interface in Gazebo. The interface is user-friendly for real-time observation and data evaluation. For the creation of documents, the sensor data need to be exported and visualized using MATLAB or other drawing tools.

E. Data Transmission Plugin

Various sensors added to the aircraft simulation model should communicate with the flight controller through a communication protocol in order to sense external environmental information or receive control commands. In this research, the adoption of Mavlink for the communication between Gazebo and the Px4 flight controller [22,23] is recommended. Mavlink allows for the interaction with control commands and sensor information.

The 'gazebo_mavlink_interface' provided by the Gazebo plugin can serve as a communication link with Px4. For example, in the control of motors and actuators, Mavlink provides control commands, and Px4 maps the 'actuator_control' from the controller to the 'actuator_output' message. The simulator module then subscribes to the message, converts it into a Mavlink format message, and sends it to the 'gazebo_mavlink_interface' plugin.

For motors, the 'gazebo_mavlink_interface' plugin can send parameters to the 'gazebo _motor_model' plugin through the 'gztopic' of Gazebo for dynamic model calculation. Here, the reference value of the motor's actual speed needs to be processed by a first-order filter in order to simulate the dynamic process of motor acceleration and deceleration.

For the steering motors, the 'gazebo_mavlink_interface' plugin will determine whether to use the dynamic model based on the 'joint_control_type' parameter. When set to 'position_kinematic', the actuator's angle is directly set as the reference value. While setting the 'position', the actuator is a position servo system controlled by a PID controller.

In addition, the IMU, GPS, barometer, and other sensors can communicate between the 'simulator' module and the 'gazebo_mavlink_interface' plugin. Many parameters in this plugin can be set through the <gazebo> and <plugin> tags provided by URDF, such as the TCP/UDP communication port, subscription topic, and actuator parameters.

4. Simulation and Analysis

The eVTOL 'Dragonfly' system contains sensing, control, and propulsion systems. Among them, the sensing system is responsible for obtaining the current position, geomagnetic data, three-axis angular velocity, three-axis acceleration, altitude, and airspeed data of aircraft for input into the control system. Then, through analysis, calculation, and processing of these data, the control system can obtain the attitude, position, and other information of the aircraft. The control system can receive control commands through human–machine interaction or automatic driving and convert them into PWM signals for the driving system. Finally, the driving system can control the attitude and position of the eVTOL.

A. Building of Simulation Scenarios

As a new type of eVTOL aircraft designed for short-distance urban traffic, the 'Dragonfly' is greatly affected by the urban environment, which in turn impacts its performance. For the system simulation analysis of this aircraft, it is essential to have a simulation scenario that closely resembles the actual physical environment. The speed and altitude data, as well as the aircraft's pitch and yaw angle, can reflect the influence of various environmental factors such as wind speed, temperature, air pressure, and buildings on the

eVTOL in urban environments. For example, a collision between an aircraft and buildings in the environment may cause the aircraft to fall or lose control. The overall design of the simulation scene is based on city buildings, which are divided into several blocks and arranged according to their distance. This method can approximate the actual distribution of urban buildings, better reflect the real working environment of aircraft, and enhance the accuracy of the simulation.

Gazebo offers a range of methods for constructing simulation scenes, including the use of Blender 3.5, SolidWorks, or other 3D software to design environment models. The models are packaged to create a library of Gazebo models, which can be easily used and placed within the Gazebo environment.

Compared with using the 'building editor' model editor provided by Gazebo to directly construct the model, the model library provided by the open-source community not only enables the creation of intricate urban model scenes, but also decreases the time required to build the simulation environment. Ultimately, it enables the creation of simulation scenes that closely resemble real environments.

B. Software-in-the-Loop (SITL) Simulation

The software-in-the-loop (SITL) simulation serves as the foundation for both hardware-in-the-loop (HITL) simulation and flight testing. In SITL, the parameters of the algorithm can be adjusted to meet the requirements of the HITL and outdoor flight tests later. In addition, SITL can also be used to test the accuracy and stability of the algorithm.

In this research, ROS/Gazebo and Px4 combined with a quaternion-based PID control algorithm were used to simulate the eVTOL system. The architecture of the system software-in-the-loop simulation is depicted in Figure 12. Mavros is a tool provided by PX4 to send and receive MAVLink messages within the ROS environment. It can not only send Mavlink messages to the PX4 platform to control the aircraft but also read topic data (position, speed, attitude, etc.) from PX4.

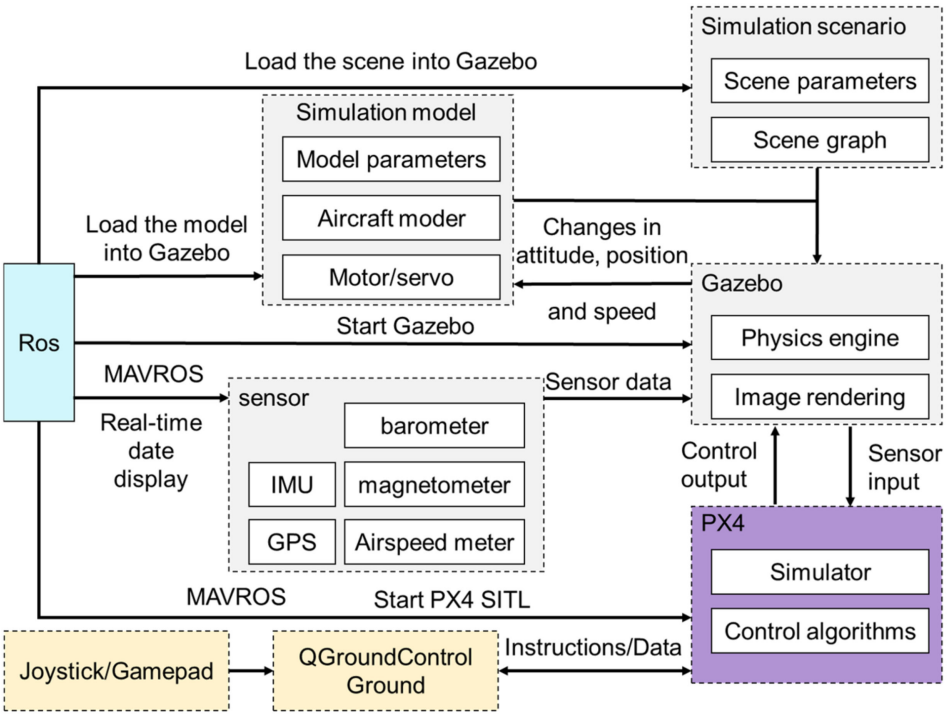

Figure 12. The architecture of system software-in-the-loop simulation.

In order to obtain real-time data on the attitude of the aircraft, it is necessary to launch multiple software programs in the system simulation at the same time. The launch file provided by ROS can start multiple ROS node programs simultaneously, eliminating the

need to use multiple terminals. The 'roslaunch' command, which is used to execute the launch file, allows access to all components of the simulation architecture. The node relationship diagram is shown in Figure 13. According to Figure 12, the SITL simulation of the 'Dragonfly' aircraft system can be conducted in the following steps: Firstly, Gazebo is started and loaded into the simulation scene called 'world'. Then, using the 'gazebo_ros' function package, the URDF model of the aircraft is generated. Secondly, the port address, Px4 configuration, Px4 SITL, and other related parameters are set. Then, the control algorithm is specified, and finally, SITL and Mavros are started. The 'QGroundControl' interface is adopted to connect Px4 and issue flight commands such as take-off, landing, and cruise flight. The process also includes using the remote control handle and connecting "QGroundControl" to transmit control commands for flight tests.

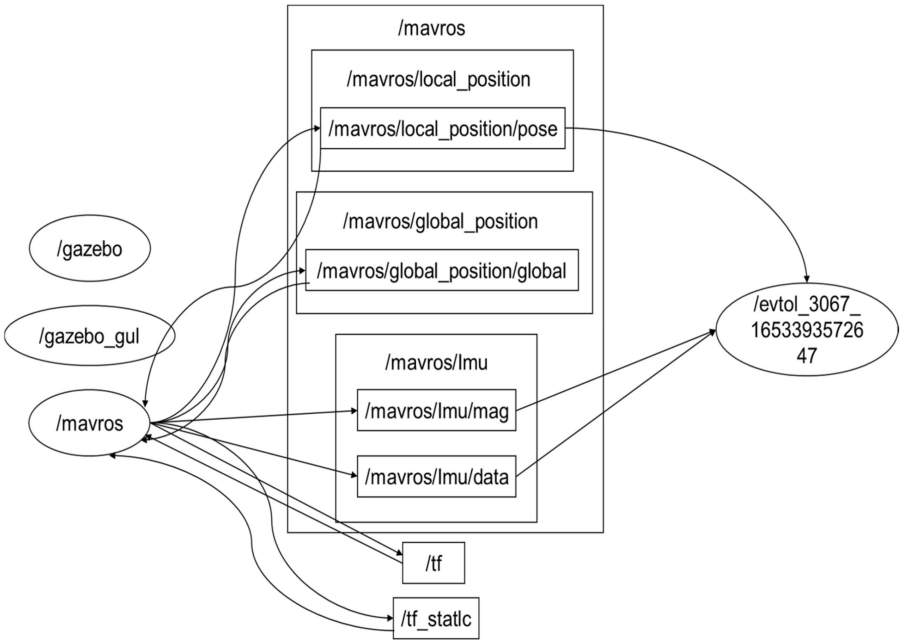

Figure 13. Diagram of ROS nodes.

C. Analysis of Simulation Results

As illustrated in Figure 14, a typical flight path has been planned to verify the aircraft's flight performance. This path includes vertical take-off/landing, mode switching, climb, cruise, and glide. Compared to other traditional aircraft, the 'Dragonfly' aircraft can integrate the characteristics of vertical take-off and landing of rotorcraft with the high-speed cruise of fixed-wing aircraft. It operates in rotor mode during take-off and landing, and it transitions to fixed-wing mode for cruising once it reaches a specific altitude and speed. Figure 15 demonstrates the flight simulation test diagram of 'Dragonfly' throughout the entire flight profile, as well as the flight trajectory of the aircraft.

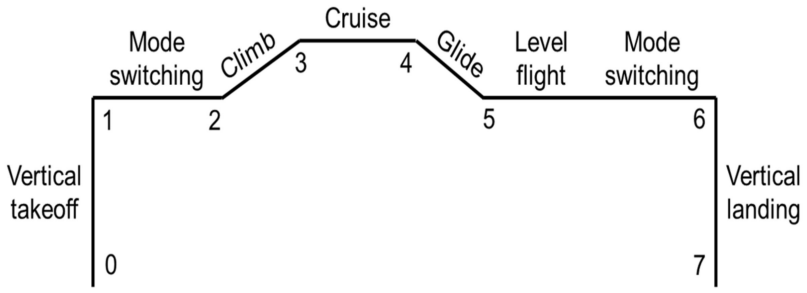

Figure 14. The planned flight path.

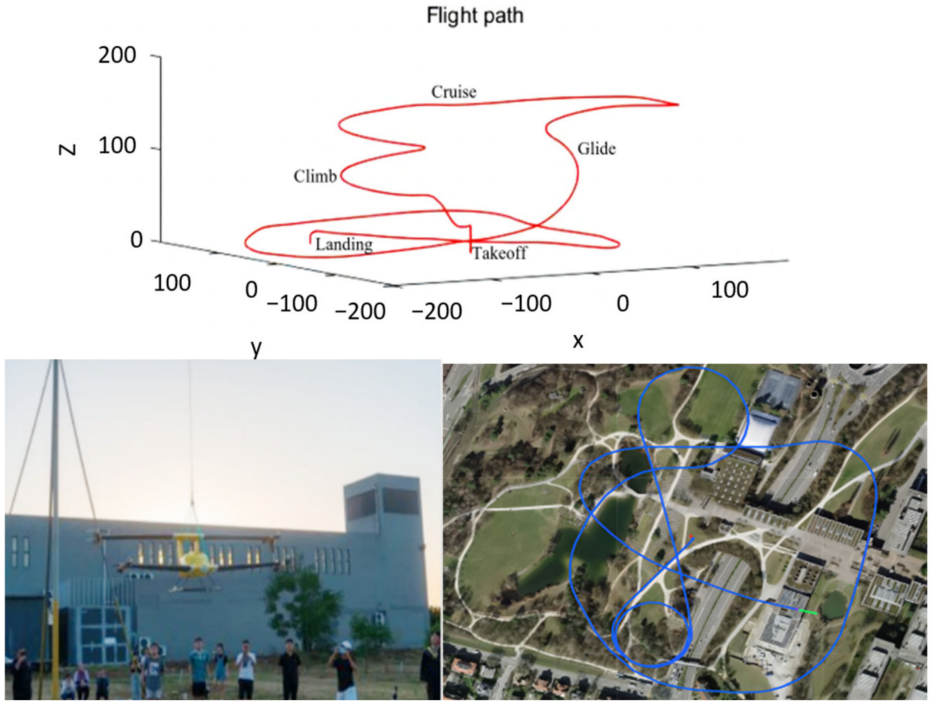

Figure 15. The flight simulation test diagram in the whole flight profile.

As shown in Figure 16, the SITL simulation of the system software provides information on the take-off climb, cruise, glide down, and landing. The flight log files in ulog format can be obtained from 'Q Ground Control' and converted into CSV files using the 'ulog2csv' tool with MATLAB. Combined with real-time data and the visualization function of the Gazebo platform, the changes in attitude, altitude, and speed of the aircraft throughout the entire flight profile can be analyzed.

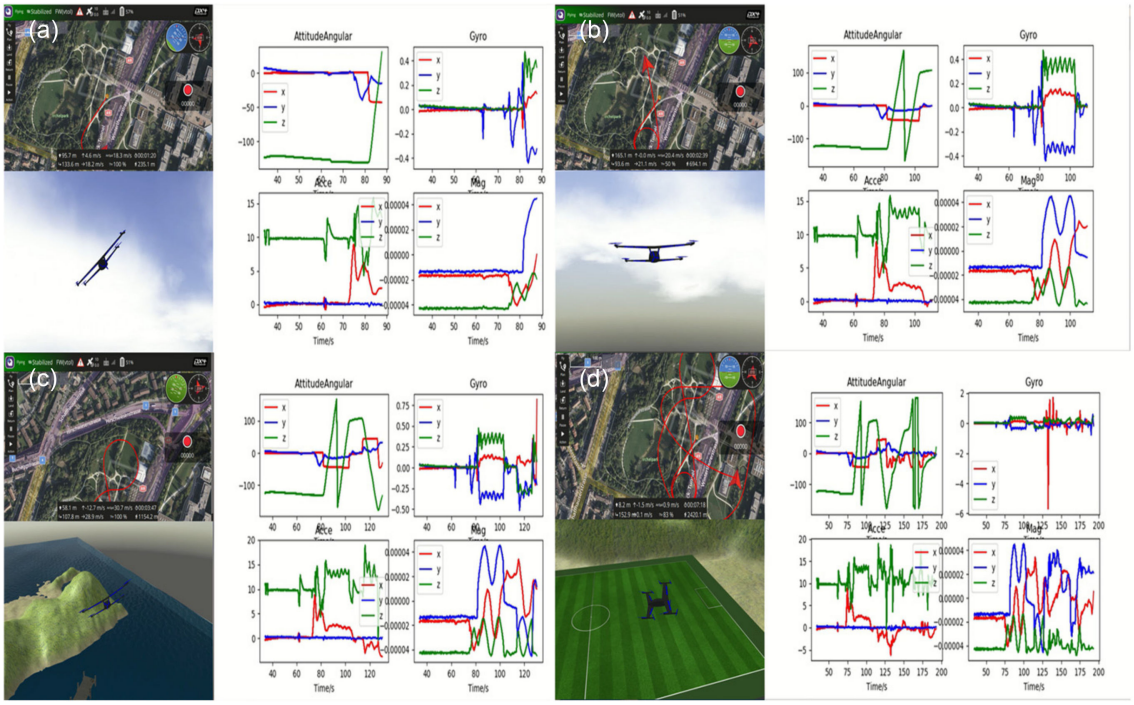

Figure 16. The flight simulation diagram at each stage: (**a**) climb, (**b**) cruise, (**c**) glide, (**d**) landing.

Figure 17 shows the raw data output from the IMU and magnetometer sensors. At the beginning of loading the aircraft model into the Gazebo environment, there is significant vibration in the data from the IMU and magnetometer sensors. When the URDF model is loaded into Gazebo, the collision attribute of the simulation model affects the aircraft's balance. As a result, the aircraft collides with the building in the simulation scene, causing it to enter an unstable state and experience strong shaking.

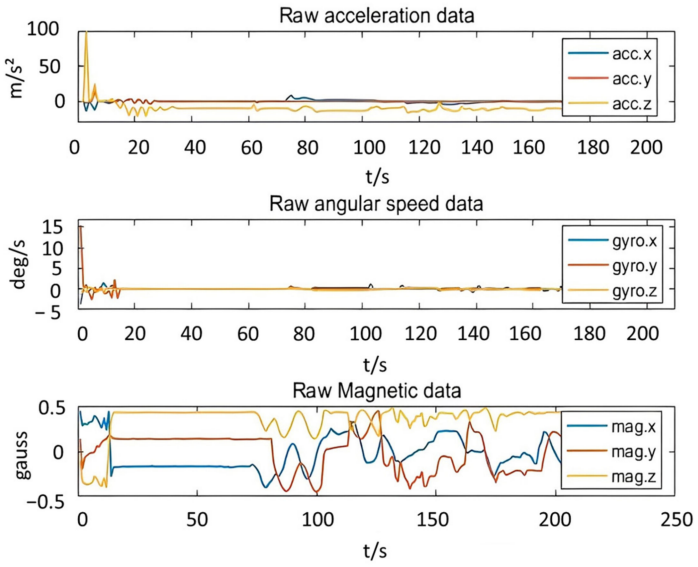

Figure 17. The raw data output by IMU and magnetometer sensors.

Figure 18 shows the change in flight attitude of the aircraft throughout the entire flight trajectory. The orange curve represents the attitude estimation value, while the blue curve represents the expected value. The horizontal axis indicates the flight time. The raw data from IMU and magnetometer sensors are inputted into the position/attitude predictor of the Px4 flight control stack. The attitude data are obtained using the data fusion algorithm of Px4. Figure 19 reflects the changes in barometric/absolute altitude and indicated/vacuum airspeed throughout the entire flight. Combined with the data changes in Figures 18 and 19, it can be observed that the aircraft collides with objects in the scene before take-off and produces strong vibrations. In particular, the maximum error caused by the vibration of the attitude angle is greater than 50°. Over time, the vibration gradually stabilizes, with the pitch and roll angles tending towards 0°, while the yaw angle settles at a stable value of 40°.

(a) Taking-off status: As shown in Figure 18a–c, the aircraft began to take off vertically at a speed of 1 m/s at 64 s, and its attitude maintained a pitch angle and roll angle of 0° and a yaw angle of 40°. During this time, the attitude was closely monitored. As shown in Figure 19, at 72 s, the aircraft reached a flying altitude of about 28.2 m, and the barometric altitude gradually increased to approximately 519 m as the barometric pressure decreased. At this time, the aircraft transitioned from rotor mode to fixed-wing mode. During this time, the flight altitude dropped to 27 m. Furthermore, in this scenario, there is a noticeable discrepancy between the air pressure altitude, which is calculated based on the standard air pressure surface, and the absolute altitude of the aircraft, which is measured relative to sea level.

(b) Climbing status: According to Figures 18a–c and 19, when the mode switched, the propulsion motor activated, the elevator deflected upward, the aircraft pitched up, and the pitch angle increased to 40°. As a result, the maximum speed decreased to 17 m/s. When the aircraft climbed to approximately 52.5 m, it reduced its speed to 15 m/s and entered a stable condition. At 82 s, the aircraft hovered and climbed with a roll angle of 42.5°, a pitch angle of 16°, and a speed of 18 m/s. As observed

from Figure 19, during the climb from 82 s to 102 s, the speed fluctuated at 18 m/s. However, the performance of maintaining a consistent speed was slightly weak. The interface of the climbing status is depicted in Figure 16a.

(c) Cruising status: Figures 18 and 19 indicate that the aircraft reached a steady altitude of approximately 164 m after 110 s and encountered varying cruising speeds. The stability of the aircraft was slightly poor, at approximately 21 m/s. However, the attitude path can be accurately tracked. Figure 16b illustrates the interface of the cruise status.

(d) Sliding status: The aircraft began to descend at 116 s by adjusting its flight attitude and speed, as observed in Figures 18a–c and 19. During the sliding phase, the maximum speed reached 32 m/s, and the flight altitude also rapidly decreased to about 13.5 m at 189 s. At this point, it transitioned from fixed-wing mode to rotor mode. Judging from the change in attitude data during the glide, the aircraft's attitude estimation has successfully tracked the target. Figure 16c illustrates the interface of the slide status.

(e) Landing status: In this period, the aircraft switched to rotor mode at 191 s, and the pitch and roll angles were close to 0°. The aircraft adjusted its yaw angle and landed at a speed of 1.5 m/s. At this time, the attitude estimation can be tracked quickly. The interface of the landing status is shown in Figure 16d.

From the entire flight simulation analysis, it can be concluded that the attitude estimation of the 'Dragonfly' aircraft achieved good tracking performance after the change in attitude. However, the stability of speed and height maintenance was slightly poor. It still requires further parameter modification. The simulation results demonstrate that this aircraft can successfully complete the basic flight task in SITL. This provides valuable technical support for future HITL and actual flight tests.

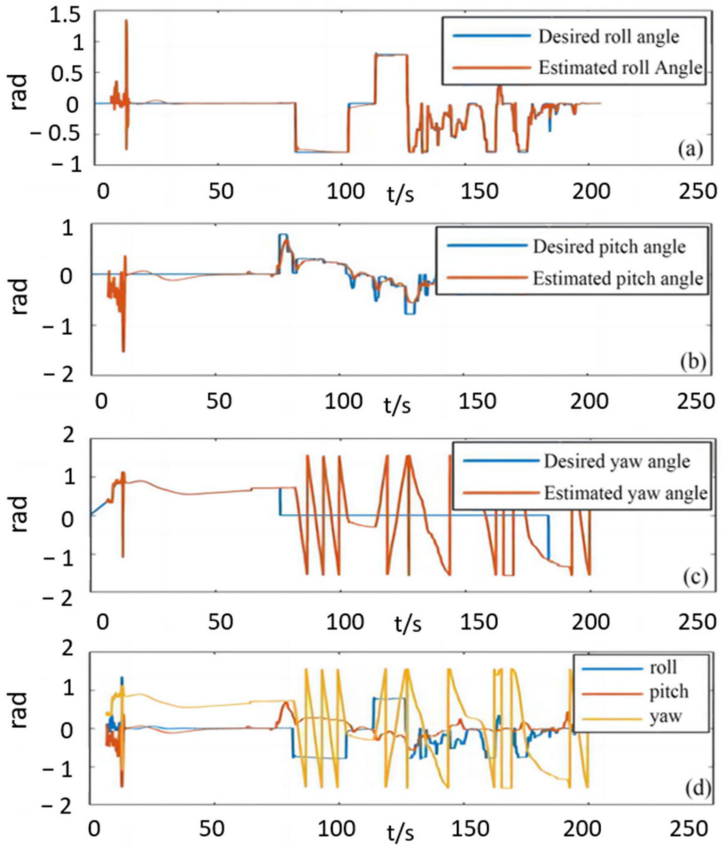

Figure 18. The change in flight attitude: (**a**) roll, (**b**) pitch, (**c**) yaw, (**d**) three-axis attitude angle.

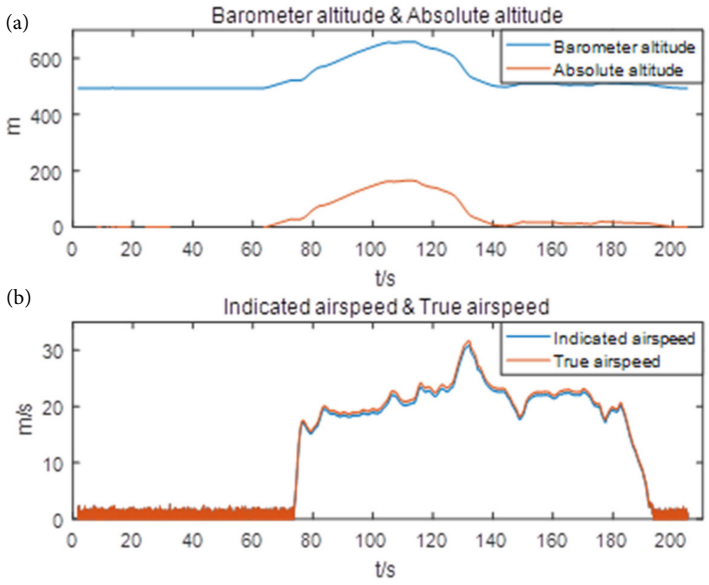

Figure 19. (**a**) The changes in barometric/absolute altitude; (**b**) the changes in indicated/vacuum airspeed.

5. Conclusions and Future Works

In this study, a simulation of a new small eVTOL aircraft called "Dragonfly" and the design of a real aircraft were completed. Compared to similar eVTOL aircraft, the wingspan is longer, and the aircraft structure is more stable; it can not only take off and land vertically without the need for a runway, but also fly quickly in a straight line and hover in mid-air. In addition, a novel semi-physical simulation test based on Gazebo and PX4 is proposed to verify the reading of aircraft sensor data, such as data from attitude sensors and magnetometers. In addition, in the take-off stage, climb stage, cruise stage, taxiing stage, landing stage, and five other flight stages, the real-time display of aircraft altitude, speed, heading angle, pitch angle, yaw angle, and other status information is verified. At the same time, the accuracy of the control algorithm is also verified. The co-simulation test based on PX4 and ROS/Gazebo proposed in this paper can provide a safer, lower cost, and more scientific aircraft simulation experiment than traditional physical experiments and provide a reference for aircraft manufacturing and design. Compared with the same type of Matlab/Simulink and FlightGear simulation software, it has the advantages of being more scientific, more intuitive, and more referential. Firstly, the simulation model of the aircraft in the Gazebo environment has been completed using the URDF robot modeling method provided by ROS. Secondly, a simulation scene highly fitting the working environment of the aircraft is designed. Finally, the framework for the in-loop simulation of the aircraft system software has been established. The flight test, conducted under the flight profile, has been carried out, and the test results have been analyzed. Based on the above research, the following conclusions can be obtained: (1) By utilizing the parameters derived from the overall and structural design of the aircraft, along with the relevant plugins available on the Gazebo platform, we can create a URDF model that closely resembles the physical properties of the aircraft. This model can then be simulated on the Gazebo platform. (2) The in-loop simulation framework of the aircraft system software has been built and completed. The control algorithm within the framework can be easily transplanted, and the parameters can be adjusted. (3) According to the software-in-the-loop simulation framework, the system simulation allows for qualitative analysis of changes in flight attitude and other data. This analysis is performed through a compiled real-time data display program and the visualization provided by the Gazebo platform. Additionally, the simulation results can be quantitatively analyzed, and the feasibility of the model can be preliminarily verified by the simulation flight experiment established by PX4 and Gazebo.

There are still some problems in this research that need to be addressed in future studies:

(1) The number of control channels of the 'Dragonfly' aircraft drive/actuator is insufficient. For this problem, the method for modifying the mixer controller in the Px4 firmware should be adopted. The mixer controls the motor or steering motor by receiving commands and converting them into actual actuator commands. The control channel of the hybrid controller should be added to operate the aircraft's actuator.
(2) In the simulation scenario, in order to verify the flight control module related to the power and control of the aircraft, only collision attributes and other influencing factors were added. Wind or other plugins can be incorporated into the simulation scenario as needed to simulate more severe weather conditions.

Author Contributions: Conceptualization, W.Z. and Y.W.; methodology, Y.W.; software, L.L.; validation, F.H., H.Z. and Y.F.; formal analysis, Y.S.; investigation, Y.W.; resources, W.Z.; data curation, Y.W.; writing—original draft preparation, Y.W.; writing—review and editing, W.Z.; visualization, L.L.; supervision, W.Z.; project administration, Y.W.; funding acquisition, W.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the Fund of National Key Laboratory of Helicopter Aeromechanics (Grant No. 2024-ZSJ-LB-02-03), in part by the Open Project of Fujian Key Laboratory of Spatial Information Perception and Intelligent Processing (Yango University, Grant No. FKL-SIPIP1026), in part by the Fund of Robot Technology Used for Special Environment Key Laboratory of Sichuan Province (Grant No. 23kftk01), in part by the National Natural Science Foundation of China (Grant No. 62303379), in part by the Natural Science Foundation of Shaanxi Province, China (Grant No. 2023-JC-QN-0665), in part by the Industry-University-Research Innovation Fund of Ministry of Education for Chinese Universities (Grant No. 2022IT189), and in part by Fundamental Research Funds for the Central Universities (Grant No. G2022WD01017).

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on reasonable request.

Acknowledgments: The author acknowledges the School of Civil Aviation, Northwestern Polytechnical University, thanks to the strong support for this project and the laboratory environmental conditions and the hardware equipment provided.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Waterman, B.M.; Pasculano, B.H.; Goldsberry, J.A.; Barker, Q.; Jack, R.; Bergeron, T.P.; Weiss, T.M.; Zolotarevsky, Z.J. Vertical Take-Off and Landing Autonomous Aircraft Design. Undergraduate Major Qualifying Project). Retrieved from Worcester Polytechnic Institute Scanned Projects Collection. 2019. Available online: http://www.wpi.edu/Pubs/E-project/Available/Eproject-032119-131229/unrestricted/MQP_Report_Submission.pdf (accessed on 3 June 2022.).
2. Nielsen, S.M. A Visually Realistic Simulator for Autonomous eVTOL Aircraft. Master's Thesis, Brigham Young University, Provo, UT, USA, 2021.
3. Craighead, J.; Murphy, R.; Burke, J.; Goldiez, B. A Survey of Commercial & Open Source Unmanned Vehicle Simulators. In Proceedings of the IEEE International Conference on Robotics and Automation, Rome, Italy, 10–14 April 2007; pp. 852–857.
4. Kinjo, H. *Development Trends and Prospects for Evtol: A New Mode of Air Mobility*; Mitsui & Co. Global Strategic Studies Institute: Tokyo, Japan, 2018; pp. 2–3.
5. Kleinbekman, I.C.; Mitici, M.A.; Wei, P. eVTOL Arrival Sequencing and Scheduling for On-demand Urban Air Mobility. In Proceedings of the 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC), London, UK, 23–27 September 2018; pp. 1–7.
6. Zintl, M.; Marb, M.M.; Wechner, M.A. Development of a Virtual Reality Simulator for eVTOL Flight Testing. In Proceedings of the AIAA Aviation Forum, Chicago, IL, USA/Virtual, 27 June–1 July 2022; p. 3941.
7. Zhang, J.; Geng, Q.; Fei, Q. EVTOL Flight Control System Modeling and Simulation based on FlightGear. In Proceedings of the International Conference on Automatic Control and Artificial Intelligence (ACAI 2012), Xiamen, China, 3–5 March 2012; pp. 2231–2234.
8. Hentati, A.I.; Krichen, L.; Fourati, M.; Fourati, L.C. Simulation Tools, Environments and Frameworks for EVTOL Systems Performance Analysis. In Proceedings of the 14th International Wireless Communications & Mobile Computing Conference (IWCMC), Limassol, Cyprus, 25–29 June 2018; pp. 1495–1500.
9. Koenig, N.; Howard, A. Design and Use Paradigms for Gazebo, An Open-source Multi-robot Simulator. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan, 28 September–2 October 2004; pp. 2149–2154.

10. Preiss, J.A.; Honig, W.; Sukhatme, G.S. Crazyswarm: A Large Nano-Quadcopter Swarm. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3299–3304.
11. Campion, M.; Ranganathan, P.; Faruque, S. A Review and Future Directions of EVTOL Swarm Communication Architectures. In Proceedings of the 2018 IEEE International Conference on Electro Information Technology (EIT), Grand Forks, ND, USA, 3–5 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 903–908.
12. Yang, H.; Lee, Y.; Jeon, S.Y. Multi-rotor Drone Tutorial: Systems, Mechanics, Control and State Estimation. *Intell. Serv. Robot.* **2017**, *10*, 79–93. [CrossRef]
13. Nguyen, K.D.; Nguyen, T. Vision-Based Software-in-the-Loop-Simulation for Unmanned Aerial Vehicles Using Gazebo and PX4 Open Source. In Proceedings of the International Conference on System Science and Engineering (ICSSE), Dong Hoi, Vietnam, 20–21 July 2019; pp. 429–432.
14. McCord, C.; Queralta, J.P.; Gia, T.N.; Westerlund, T. Distributed Progressive Formation Control for Multi-Agent Systems: 2D and 3D deployment of EVTOLs in ROS/Gazebo with Rotors. In Proceedings of the 2019 European Conference on Mobile Robots (ECMR), Prague, Czech Republic, 4–6 September 2019; pp. 1–6.
15. Khaliq, S.; Ahsan, S.; Nisar, M.D. Multi-Platform Hardware in the Loop (HIL) Simulation for Decentralized Swarm Communication Using ROS and GAZEBO. In Proceedings of the IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Pisa, Italy, 7–11 June 2021; pp. 310–315.
16. Crick, C.; Jay, G.; Osentoski, S. ROS and ROS bridge: Roboticists out of the Loop. In Proceedings of the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction, Originally Sapporo, Hokkaido Interaction, Sapporo, Japan, 7–10 March 2022; pp. 493–494.
17. Canas, J.M.; Perdices, E.; Garcia-Perez, L. A ROS-based Open Tool for Intelligent Robotics Education. *Appl. Sci.* **2020**, *10*, 7419. [CrossRef]
18. Beinschob, P.; Reinke, C. Graph SLAM based Mapping for AGV Localization in Large-scale Warehouses. In Proceedings of the IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 3–5 September 2015; pp. 245–248.
19. An, Z.; Hao, L.; Liu, Y. Development of Mobile Robot SLAM based on ROS. *Int. J. Mech. Eng. Robot. Res.* **2016**, *5*, 47–51. [CrossRef]
20. Hernandez Mendez, S.; Maldonado Mendez, C.; Marin Hernandez, A. Design and Implementation of a Robotic Arm using ROS and MoveIt. In Proceedings of the IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), Ixtapa, Mexico, 8–10 November 2017; pp. 1–6.
21. Tsolakis, N.; Bechtsis, D.; Bochtis, D. Agros: A Robot Operating System based Emulation Tool for Agricultural Robotics. *Agronomy* **2019**, *9*, 403. [CrossRef]
22. Pietrzik, S.; Chandrasekaran, B. Setting up and Using ROS-Kinetic and Gazebo for Educational Robotic Projects and Learning. In *Journal of Physics: Conference Series, Proceedings of the 2019 3rd International Conference on Control Engineering and Artificial Intelligence (CCEAI 2019), Los Angeles, CA, USA, 24–26 January 2019*; IOP Publishing: London, UK, 2019; Volume 1207, p. 12019.
23. Jang, J.T.; Santamaria-Navarro, A.; Lopez, B.T. Analysis of State Estimation Drift on a MAV using PX4 Autopilot and Mems IMU during Dead-reckoning. In Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, 7–14 March 2020; pp. 1–11.