applied sciences

MDPI

Article

Hardware-In-the-Loop Simulation of Time-Delayed Anti-Swing Controller for Quadrotor with Suspended Load

Hanafy M. Omar

Department of Mechanical Engineering, College of Engineering, Qassim University, Buraydah 51452, Saudi Arabia; hanafy@qec.edu.sa

Abstract: Quadrotor flying vehicles with suspended loads have a lot of applications, such as mine detection and transferring loads for delivery, or through places where ground vehicles cannot reach. In these applications, the payload will be suspended underneath the vehicle and it is subjected to large oscillations due to external disturbances or the acceleration of the vehicle itself. These oscillations add additional forces and moments to the quadrotor, and this deteriorates its performance and stability. In this paper, we developed a hardware-in-the-loop (HIL) simulation platform for a quadrotor with a suspended load system using Gazebo. The developed platform can be used to understand the dynamics of the system and design control systems to suppress the oscillation of the suspended load. An anti-swing controller that is based on time-delayed feedback of the load swing angles is implemented in the designed HIL environment to show the usefulness of the newly developed system and the effectiveness of the anti-swing controller.

Keywords: quadrotor; suspended load; anti-swing; hardware-in-the-loop—HIL; PX4; autopilot

Citation: Omar, H.M. Hardware-In-the-Loop Simulation of Time-Delayed Anti-Swing Controller for Quadrotor with Suspended Load. *Appl. Sci.* **2022**, *12*, 1706. https://doi.org/10.3390/app12031706

Academic Editors: Luigi Fortuna and Yosoon Choi

Received: 26 November 2021
Accepted: 31 January 2022
Published: 7 February 2022

1. Introduction

In recent years, the demanding nature of civilian and military applications has brought significant attention to autonomous vehicles. Moreover, to provide a solution to distinct applications which cannot be handled using land or water, flying vehicles are the only choice. Furthermore, applications that require hovering and vertical take-off further limit solutions for autonomous vehicles and rotary flying vehicles, such as helicopters and quadrotors [1].

The quadrotor vehicle has more advantages compared with the helicopter, such that the presence of four rotors allows for load distribution and enables the quadrotor to have smaller rotor diameters, and hence smaller vehicle size, as compared with the single rotor helicopter. Furthermore, the smaller rotors store less kinetic energy per rotor and thus reduce the extent of damage in the event that the vehicle is involved in a crash. In addition, rotors can be protected by enclosing them within a frame.

Unmanned quadrotors, with the capability of handling suspended payloads with a cable, can be used in situations where normal land vehicles either cannot reach or are not recommended to be used. Applications such as tracking moving targets, photographing for surveillance, rescue operations for natural disaster areas, discovering mines, or inspecting gas and oil pipelines that exist in remote or hazardous areas are better operated with aerial vehicles [2].

A vehicle with a suspended load system can be treated as a multi-body dynamical system. The pendulous motion of a suspended payload with a cable can limit the applicability by either slowing down the flying vehicle or can damage the load itself. Moreover, the pendulum oscillation may prevent an accurate alignment during the pickup and/or placement of the load. In reality, a violent suspended payload swing due to a wind gust or pilot input can cause an accident [3]. It is thus critical to limit the payload swing to improve safety and performance [4].

In recent years, the delivery of packages has become an attractive application for multi-rotor vehicles, especially quadrotors. Many companies started offering this service, such as Amazon [5]. The payload can be attached to the quadrotor with a rigid connection or it can be suspended underneath the quadrotor by a cable. The rigid connection is the most common because of its practicality and simplicity. However, it is not suitable for heavy and large size packages, and in some situations, it may not be possible to land the vehicle. Moreover, with this configuration, the inertia of the quadrotor will increase significantly, which will reduce its maneuverability [6]. With a suspended load, large packages can be carried with one or more quadrotors. This configuration allows delivering the package without the necessity to land the vehicle, which improves the mission efficiency. However, it will complicate the dynamics and the performance of the quadrotor, especially when controlling the position of the payload in windy scenarios. Therefore, many researchers have recently started to develop control systems for the quadrotor with suspended load to stabilize the vehicle and suppress the oscillations of the load, alongside attaining accurate pickup and positioning of the load.

It is expensive and time-consuming to develop and test algorithms for autonomous vehicles in the actual world. Quadrotor simulations must be conducted prior to the UAV's actual flight to limit the danger of property damage caused by aircraft crashes, system failure, or controller malfunction. Different simulations, such as numerical simulations, software-in-the-loop simulation (SIL), hardware-in-the-loop (HIL) simulation, and actual tests on testbeds, can be used to avoid these problems from arising; moreover, these methods make it possible to test and build multirotor vehicles safely and cost-effectively.

SIL simulation relies mainly on a separate computer where the flight code is completely simulated, along with a sensor model and vehicle dynamics to provide similar results to those of a real-life experiment; SIL is mainly used during the development stage of the simulation model. The SIL platform relies on UDP and TCP communication among its various parts, this communication is mainly governed by what is known as Lockstep, which is a technique developed to synchronize all the simulator parts together to avoid any packet loss or desynchronization between them. Lockstep issues a command for each cycle, performed by the controller to mimic the operation in real life [7,8].

HIL is a good tool for evaluating the performance of deployed models and algorithms on real-world hardware. The HIL simulations run the flight code and algorithms on the real-time computer, models the vehicle dynamics on the host computer, feeds data through a serial connection, and then visualizes the received responses [8–10].

Most HIL systems are developed for simulating the quadrotor only [11–14]. Recently, SIL simulations with Gazebo and PX4 were conducted by Graham [6] for a quadrotor with a suspended load. The intentions were to test his design for a landing system. However, no HIL simulations for the quadrotor have been demonstrated; therefore, there is a gap which must be filled by researchers.

Pixhawk hardware has been widely used for UAV applications in recent years [15]. It offers high-quality autopilot hardware at a low cost and with a high level of availability. Furthermore, the open-source code PX4, which is autopilot firmware for driving unmanned aerial and ground vehicles, can be installed on this device. Pixhawk can be used in conjunction with simulation software, such as Gazebo, to create a HIL system. [16–18].

The Gazebo software is popular in the robotics community and is completely open-source so that the user can easily define 3D virtual worlds, sensor models, and communication protocols. In particular, this software contains the open dynamics engine (ODE), which can present a system model robot with high accuracy in real-time conditions. ODE can provide real-time dynamical simulations and the corresponding sensor readings for the UAVs [18,19].

The present paper is organized as follows: The quadrotor model of the quadrotor and the suspended load is given in Section 2; the suggested controller is given briefly in Section 3, followed by the procedure to develop the HIL environment in Section 4. The simulation results are discussed in Section 5. The conclusion is given in Section 6.

2. Quadrotor–Load Model

The quadcopter used has a cross structure, as shown in Figure 1. It consists of a frame with four motors that provide the forces that lift and move the vehicle. The suspended load is attached to the center of the frame.

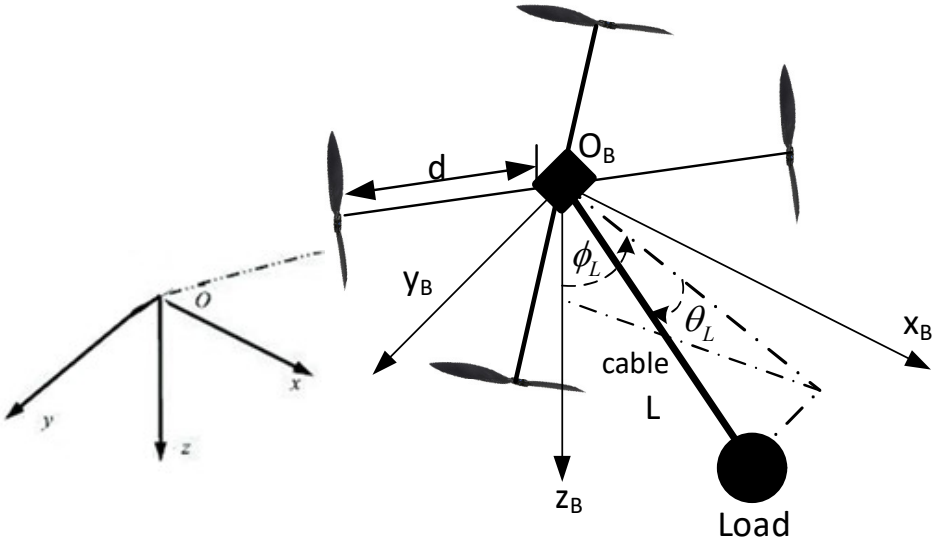

Figure 1. Quadrotor with a suspended load.

The quadcopter with a suspended load system can be considered as a multi-body dynamical system [20]. Two frames are typically used to describe the movements of a quadrotor vehicle. The earth frame, also known as the inertial frame, is the first frame. The second frame is a body-fixed frame, in which the origin OB represents the quadrotor's center of mass.

The position of the quadrotor body frame with respect to the earth frame is given by x, y, and z, which represent the longitudinal, lateral, and vertical motions, respectively. The orientation of the body frame with respect to the earth frame is given by the angles ϕ, θ, and ψ, which represent the roll, pitch, and yaw, respectively [21].

The suspended load can be described as a point mass that behaves as a spherical pendulum. Therefore, its motion can be described by two angles, θ_L and ϕ_L, where ϕ_L is the load oscillation angle in the xz plane, and θ_L is the load oscillation angle out of the xz plane.

3. Quadrotor Controller

The configuration of the proposed controller which will be implemented in the developed HIL environment is shown in Figure 2. It consists of two control systems: an anti-swing controller and a tracking controller [22].

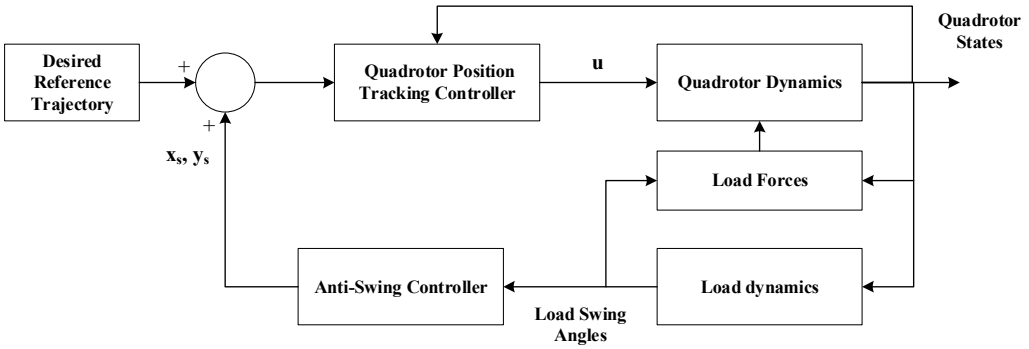

Figure 2. Proposed control loop.

3.1. Tracking Controller

The objective of the tracking controller is to follow the original trajectory of the quadcopter and the additional trajectory that is generated from the anti-swing controller. So, to follow the quadcopter trajectory, the six outputs of the quadrotor $(x, y, z, \phi, \theta, \psi)$ must be controlled. However, selecting all these six outputs will induce an under-actuated system, as there are only four inputs. To overcome this problem, a double-loop architecture method is adopted as shown in Figure 3 [23,24]. The function of the inner loop is to control the attitude angles and the altitude (ϕ, θ, ψ, z), while the function of the outer loop is to follow the desired trajectory in the x and y directions by providing the inner loop with the desired angle values. As a result, the quadrotor is stabilized at a desired height and yaw angle by the inner loop controller, while the outer loop stabilizes it in the x and y directions. To achieve stability, the inner loop must be substantially quicker than the outer loop.

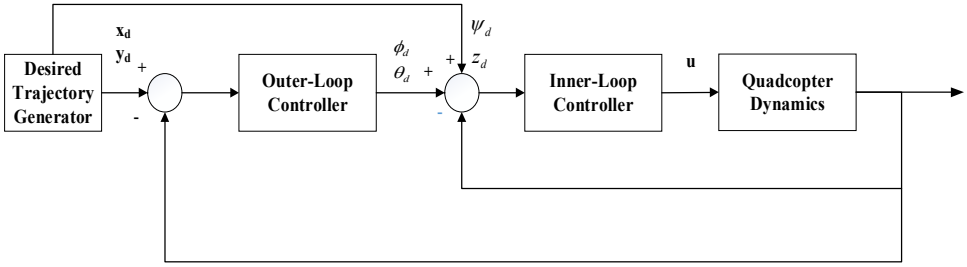

Figure 3. Structure of the quadcopter tracking controller.

The controller used in the open-source flight controller (PX4) is the PID controller. More details about the structure and design of the tracking controller implemented in the open-source PX4 autopilot can be found in [25]. In this controller, it is assumed that the motion is just a perturbation from the hovering condition. The PX4 firmware used in this work is the V1.12.3 release [26].

3.2. Anti-Swing Controller (ASC)

The objective of the ASC is to suppress the oscillations of the suspended. The time-delayed feedback gives a correction trajectory, which adds to the reference trajectory [27,28]. This correction trajectory is given by the following equations:

$$x_s = -K_{sx} L \, \phi_L(t - t_{sx}) \tag{1}$$

$$y_s = -K_{sy} L \, \theta_L(t - t_{sy}) \tag{2}$$

where K_{sx} and K_{sy} are the feedback gain and t_{sx} and t_{sy} are the time delay of the swing angles in the x and y directions, respectively. By proper selection of these values, the load swing angles will stabilize and vanish. This feedback scheme is easy to implement. Moreover, only minimal knowledge of the system is required. The only quantity of the system that needs to be known is the oscillation period TL of the load which is given by the following:

$$T_L = \frac{2\pi}{\sqrt{g/L}} \tag{3}$$

The parameters of the ASC can be tuned to give the best damping for the suspended load. To measure the level of damping, the integration of the swing time history (*ISH*) is calculated according to the following equation:

$$ISH = \int \left(\phi_L^2 + \theta_L^2 \right) dt \tag{4}$$

According to Equation (4), the level of damping will increase with a decrease in the *ISH* index.

3.3. Swing Angle Sensor

The ASC is suggested to be implemented in Pixhawk autopilot board. The implementation needs the measurement of the load swing angles which can be performed by two methods. In the first method, the swing sensor can be connected directed to the pixhawk analog to a digital (ADC) port, while the other is a non-contact one, where an inclination sensor is placed on the load and emits its signal either through wired connection or Bluetooth. In both cases, the sensor data is routed through the Pixhawk board to ensure the readings are synchronized to the main flight controller readings and then processed after relay on the companion computer. In this work, the non-contact measurement method will be used, which is based on an inclinometer that takes its measurements through an inertial measurement unit (IMU).

4. HIL Simulation

The HIL simulation is carried out to test the validity and performance of the controller and its impact on the autopilot performance in operation. The simulation system is separated into three main parts:

1. The quadrotor and load dynamics simulation;
2. The autopilot sensors simulation;
3. The communication and synchronization between the different parts.

The quadrotor dynamics and sensor simulation is handled by Gazebo, which is an open-source robotics simulation framework that provides flexible and extensible environments for modeling and simulating various types of vehicles and environments, including our platform, a quadrotor, and its onboard sensors [29].

The communication is based on the Mavlink protocol, which is the standard communication protocol for micro aerial vehicles, and is handled by an open-source robotics framework for communication and data transfer between multiple components and systems, that relays stamped messages, states, commands, and other data through a serial connection, such as USB, UART, UDP, or TCP remote connection through WIFI [30]. Figure 4 shows the communication diagram during the HIL simulation [31].

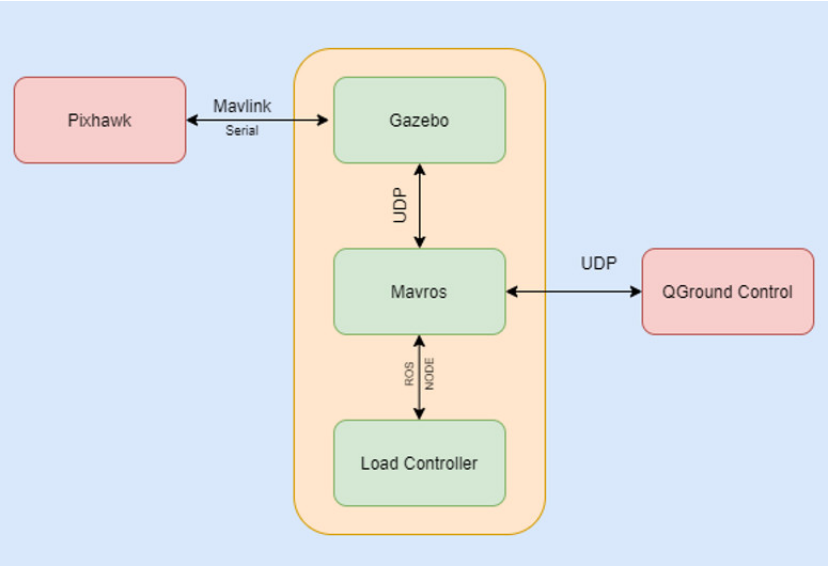

Figure 4. Hardware-in-the-loop (HIL) communication diagram.

4.1. Load Model

The suspended load is connected to the quadrotor on a ball and socket joint which allows rotation in the three axes with no translation. For simplicity, the rotation around the central axis "z" is disabled since it is not the focus of the control point of view. The connecting rod is then fixed to the load and the sensor is attached to it.

The joint structure in Gazebo is built up from a chain of simple joints, mostly fixed joints and one DOF joint. These simple joints are hierarchically connected by what are called virtual or dummy joints. These dummy joints serve the purpose of defining the allowed degrees of freedom individually. Therefore, the ball joint can be described in the Gazebo as two virtual one-DOF joints in the in the-plane swing direction (ϕ) and the out-of-plane swing direction (θ) with a fixed joint at the end that attaches to the load, as shown in Figures 5 and 6.

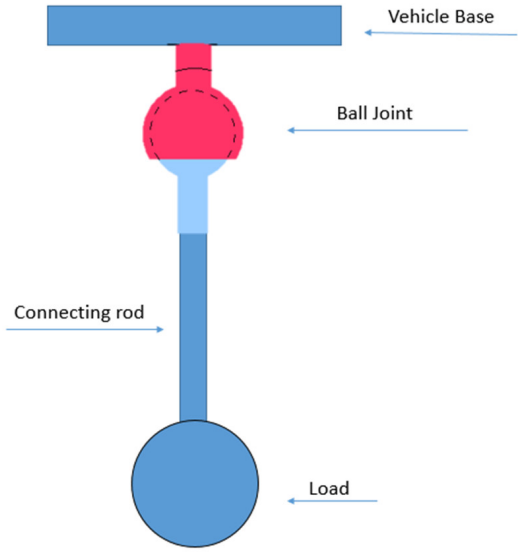

Figure 5. Load fixation.

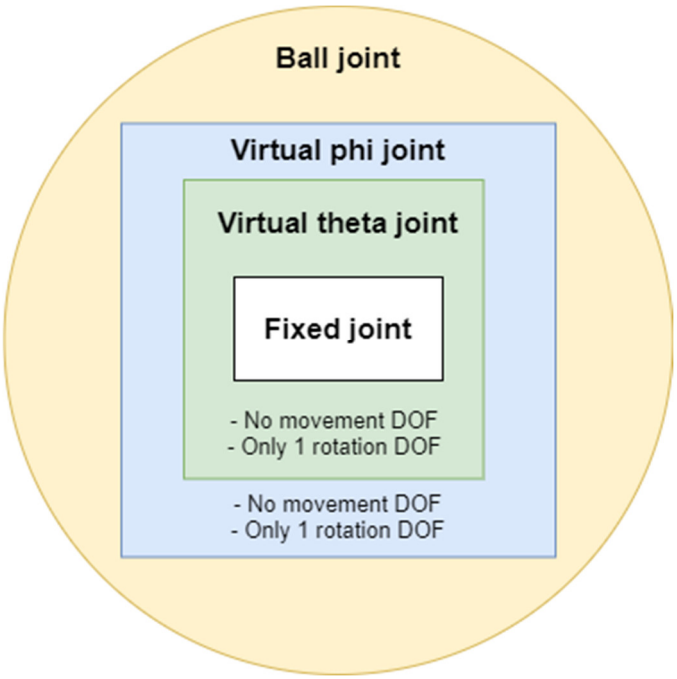

Figure 6. Ball joint structure in Gazebo.

The joint configuration and drone joint model is displayed in Rviz, which is an open-source visualizer tool for robotics applications [32], as shown in Figure 7. This helps in checking for joint interaction and limits in a clear visual manner.

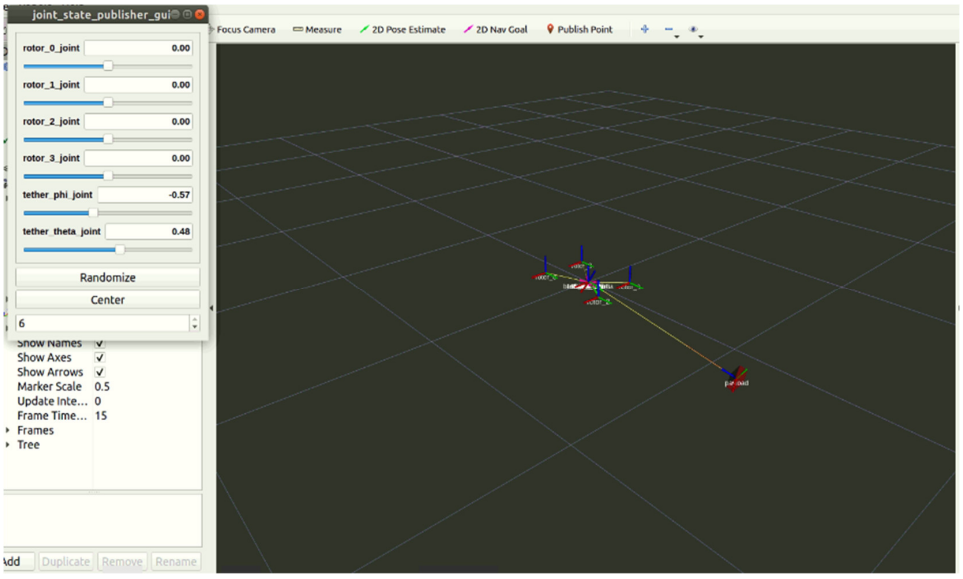

Figure 7. RVIZ display of the quadrotor with the suspended load.

4.2. Gazebo Implementation

The Gazebo platform relies on an XML-based file format named SDF, which stands for Simulation Description Format—this is either written directly or generated from a scripted format under the name URDF (Universal Robot Description Format), or from macro format under the name XACRO. These formats ease the building of complex robots as they rely on defining the main components of the robots and connecting them through joints and links. Complex robotics rely mainly on templates generated from the XACRO format to produce URDF for defining the robot and then automatically converting it to SDF using ROS. This benefits in the ability to define joint controllers and ROS controllers, which allow the control of the robot and manipulation of its parts using ROS API that manipulates the Gazebo simulation.

The hinge is composed of two single DOF joints as stated before that rotate in the $\phi - \theta$ planes. Empty links are attached to these joints then fixed to the main link. This link represents the rod that is fixed to the suspended load using a fixed joint to avoid any unnecessary movements or rotations. Figure 8 shows a simple hierarchy tree for the drone and load model that is displayed in its ground state in Figure 9, and in its flying state in Figure 10.

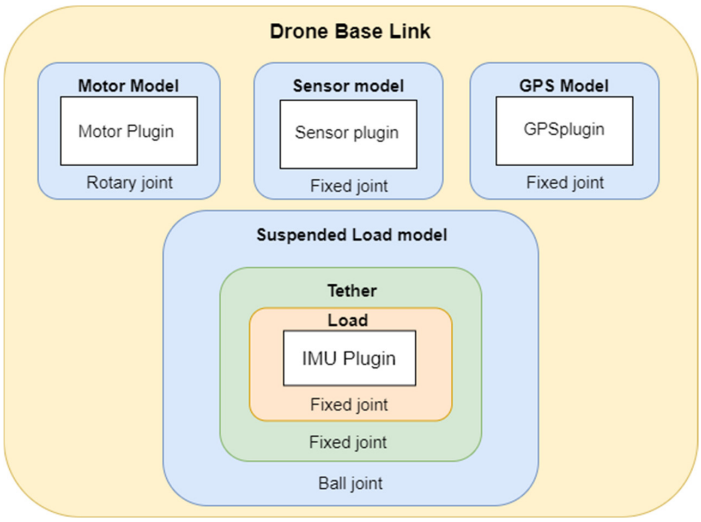

Figure 8. Gazebo model hierarchy.

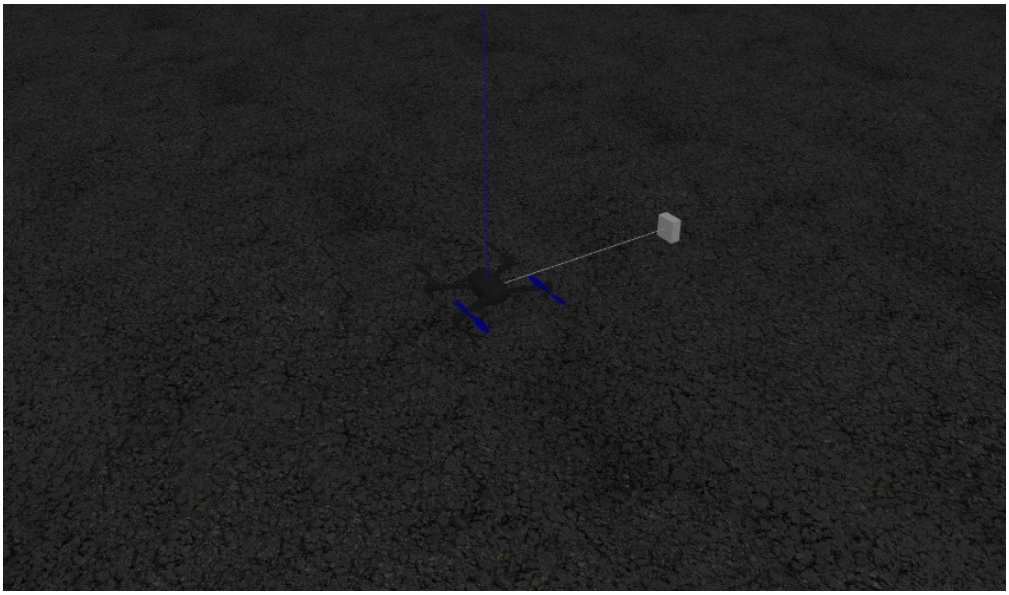

Figure 9. Quadrotor with the suspended load on the ground.

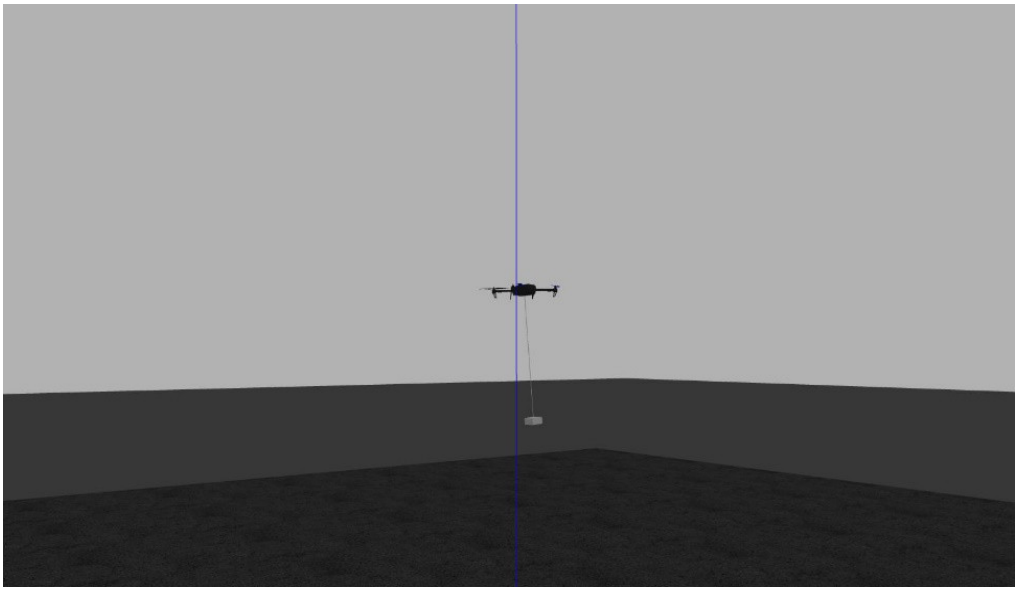

Figure 10. Flying quadrotor with the suspended load.

4.3. Sensor Details

To simulate the inclination sensor, a gazebo sensor plugin for an inertial measurement unit (IMU) is used and placed, using a fixed joint, on the load part. This sensor allows the measurement of the inclination angles of the load, relative to their fixation point on the quadrotor and their rates. This data is published on both a Gazebo link state and a separate sensor ROS topic, so it can be later processed. Below is the code snippet for the sensor and defined parameters that are appended to the SDF. The sensor adds a Gaussian noise to simulate real-life performance and has a switch to turn on or off, in case it is needed to simulate a failsafe. The update and publish rate is also variable and can be easily adjusted to different rates to suit the performance needs.

```
<gazebo>
<plugin name="imu_plugin" filename="libgazebo_ROS_imu.so">
<alwaysOn>true</alwaysOn>
```

```
<bodyName>base_footprint</bodyName>
<topicName>imu</topicName>
<serviceName>imu_service</serviceName>
<gaussianNoise>0.0</gaussianNoise>
<updateRate>20.0</updateRate>
</plugin>
</gazebo>
```

This plugin publishes its reading to a ROS topic defined by the parameter "<topicName>", this data is then synchronized with the PX4 using a Mavlink message through MAVROS then sent to the control algorithm.

The plugin approach can be replicated by reading the link state directly from gazebo, but it is better to use external plugins to measure drone behavior on the readings.

4.4. MAVROS Plugin

MAVROS is a communication node between ROS and the Mavlink protocol to facilitate algorithm and application development. The Mavlink system sends data in a structured format called "messages", where the message is instantiated as an object and its contents are parameters to that object. The code below shows the structure of the "swing_angles" message that was created specifically for this work.

```
uint64 timestamp # time since system start (microSeconds)
float32 phi # Phi swing angle
float32 theta # Theta swing angle
float32 phid # Phi swing angle rate of change
float32 thetad # Theta swing angle rate of change
```

The message contains a timestamp header to ensure synchronization with the main IMU in the Pixhawk. The plugin performs two main functions: during simulation, it is used as a relay for the simulated sensor from the Gazebo to the controller and the PX4 module through Mavlink; meanwhile, in real testing, the plugin receives the sensor message from the PX4 and relays it back to the companion computer and then publishes it through ROS for the controller. The first function is useful in determining the suitable update rates that can function correctly with the PX4 firmware and various Pixhawk boards, without overloading the processor with a large data stream.

4.5. Mavlink Message Definition

The Mavlink message sent by Mavros should be defined in the Mavlink headers of both the sender and receiver; thus, the Mavlink headers in both Mavros and PX4 are regenerated with the newly enumerated message so it can be transmitted. This enumeration follows the Mavlink message template definition so it can be implemented to any Mavlink-supported flight stack. The message definition snippet is shown below.

```
<message id="228" name="SWING_ANGLES">
<description> Transmit swing angles from suspended payload sensor to Pixhawk
</description>
<field type="uint64_t" name="time_usec" units="us">Timestamp (synced to
UNIX time or since system boot).</field>
<field type="float" name="phi">Swing angle Phi</field>
<field type="float" name="theta">Swing angle theta </field>
<field type="float" name="phid">Swing angle rate Phi</field>
<field type="float" name="thetad">Swing angle rate theta </field>
```

```
</message>
```

4.6. PX4 Application for Real-Time Data Viewing

PX4 is developed upon the NuttX, a Linux real-time operating system (RTOS), and it contains a console for control and debugging. Using this console, several apps can be developed to be used directly on the PX4 without the need for external computers. An app has been developed to read and view the "swing_angles" topic, that is published on the PX4, so the data can be quickly debugged and monitored on the fly. The app can also be set up to infinitely display data upon request. Below in Figure 11 is the output of the NuttX console when the application is running in real-time on a Pixhawk 1 board using the HIL simulation.

```
pxh> swing_angles
INFO  [swing_angles] Hello Dr Omar!
1000 | Phi | Theta | dPhi | dTheta
---------------------------------
+0.08  |  -0.02  |  +0.55  |  -0.22
+0.17  |  -0.05  |  +0.25  |  -0.06
+0.18  |  -0.06  |  -0.15  |  +0.06
+0.11  |  -0.04  |  -0.47  |  +0.11
-0.00  |  -0.01  |  -0.54  |  +0.17
-0.10  |  +0.03  |  -0.37  |  +0.14
-0.15  |  +0.05  |  -0.03  |  +0.12
-0.12  |  +0.05  |  +0.26  |  -0.06
-0.04  |  +0.03  |  +0.45  |  -0.11
+0.05  |  -0.01  |  +0.43  |  -0.13
+0.12  |  -0.04  |  +0.23  |  -0.01
+0.13  |  -0.05  |  -0.11  |  -0.02
+0.08  |  -0.04  |  -0.35  |  +0.08
+0.00  |  -0.01  |  -0.38  |  +0.13
-0.07  |  +0.02  |  -0.24  |  +0.13
-0.10  |  +0.04  |  -0.04  |  +0.04
-0.08  |  +0.04  |  +0.17  |  -0.08
```

Figure 11. Real-time reading from PX4 terminal.

The application can later be extended to include the ASC controller as a part of the PX4 software stack to eliminate the need for a companion computer. This eliminates the external delay that can occur due to the transmission and external processing delay.

4.7. Data Logging Interface

An interface was designed using Python and Qt, a dynamic robust framework for developing graphical user interfaces (GUI) applications, to facilitate the process of simulation and data logging, as shown in Figure 12. The interface provides a real-time representation of the quadrotor states and the load states directly from the sensors. It also provides the ability to quickly modify controller gains and drone parameters. The interface is also connected to RQT [33], an open-source data logging system and visualizer for ROS-based applications, to provide more control and visualization tools for the simulation. The main features are developed based on ROS, and the drone model modification capabilities rely on parsing the SDF files and launch files. The interface provides the ability to run any configuration through a launch file and the ability to add or customize any feature, such as running a simulation multiple times or launching pre-configured instances of other software, such as RQT or RViz, for more features [32]. Another addition is the ability to log and save a synchronized version of the states that are used later in for post-processing, debugging and/or analysis of the data. This is a handy feature that really saves the trouble of frame matching the data readings and multiple simulation clocks between nodes and topics.

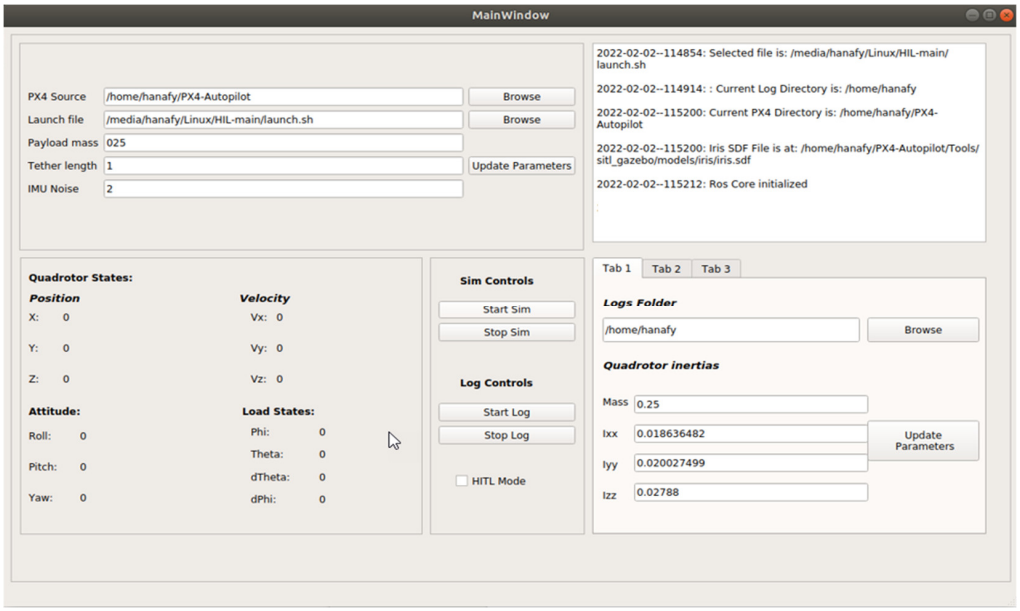

Figure 12. Simulation GUI interface.

The interface also has the capability of modifying the internal parameters of PX4 on the fly through its use of Mavros as shown in Figure 13. Further, the choice of Qt for GUI development provides the freedom for customization and ease of simulation and testing and extension to include embedded visualization tools as shown in Figure 14.

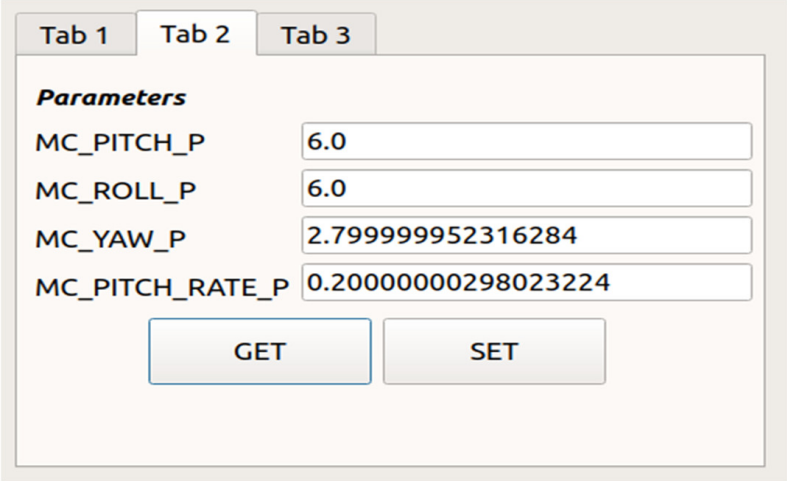

Figure 13. PX4 parameters manipulator.

4.8. Controller Implementation

The controller was written in Python for ease in development; it receives the discretized analog signals from the Pixhawk and then converts them to the desired angles after applying noise filters if required or calibration biases. The data is then used to calculate the delay values. The controller node operation is delayed by the desired value that is calculated dynamically through operation and then the generated trajectory is calculated and then added to the main vehicle trajectory, to be then published as position setpoints that the quadrotor can track. Figure 15 shows the code flow of the controller implementation.

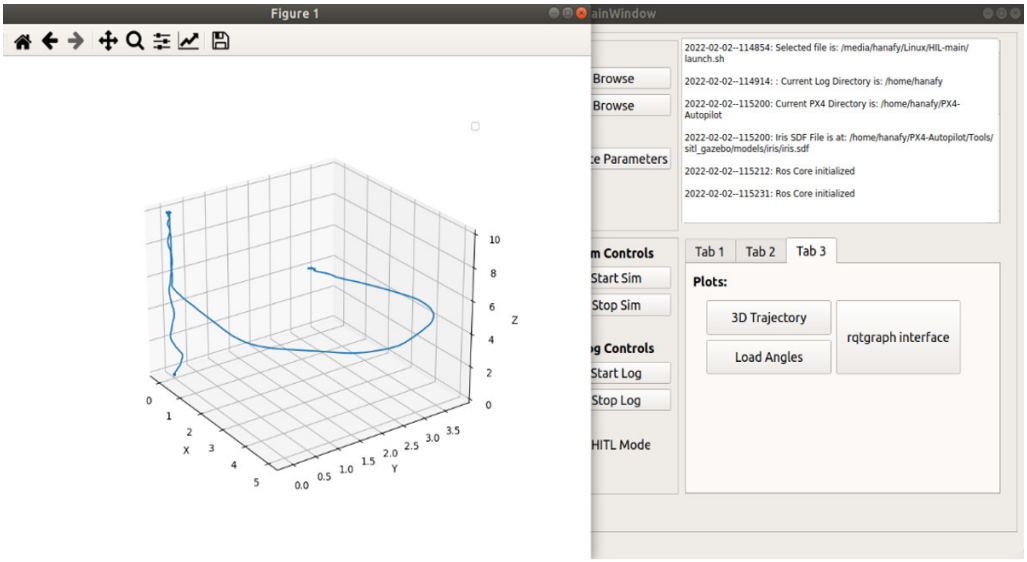

Figure 14. Trajectory plotter.

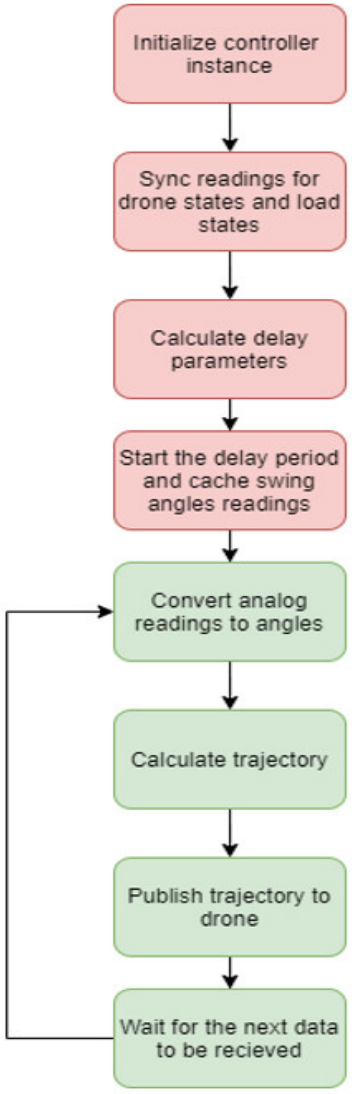

Figure 15. Controller code flowchart.

5. Simulation Results

The inertia and propulsion system properties of the quadrotor model used in the simulation are listed in Table 1.

Table 1. Parameters of the quadcopter and the suspended load.

Symbol	Description	Value	Unit
m	Quadrotor mass	1.585	Kg
I_{xx}	Quadrotor moment of inertia in the x, y, and z directions	0.018636482	Kg·m^2
I_{yy}		0.020027499	
I_{zz}		0.02788	
d	Quadrotor arm length	0.252	m
C_m	Motor moment coefficient	1.39×10^{-7}	
C_t	Motor thrust coefficient	9.79×10^{-6}	
C_{df}	Quadrotor fuselage drag coefficient	0.055	
C_{dm}	Motor drag coefficient	0.003	
m_L	Suspended load mass	0.27	Kg
L	Suspended load cable length	1	m
(x_h, y_h, z_h)	Hook suspension point	(0,0,–0.1)	m

5.1. Hover Flight

The suspended load is subjected to a disturbance of 30 degrees in both directions. This is carried out by directly changing the joint state in the Gazebo simulation or by enforcing force disturbances on the load, but the first yielded better and more accurate state changes. The time history of the load swing angles is shown in Figure 16, while the time history of the quadrotor position in the forward and lateral directions is shown in Figure 17, without including the ASC. It is observed that the quadrotor tracking controller can dampen the oscillations of the suspended load due to the moments generated on the quadrotor from the motion of the suspended load. The tracking controller implemented in PX4 was able to return the quadrotor to its initial position. This indicated that the system is stable with this controller. The ISH determined for this flight maneuver is 0.6752.

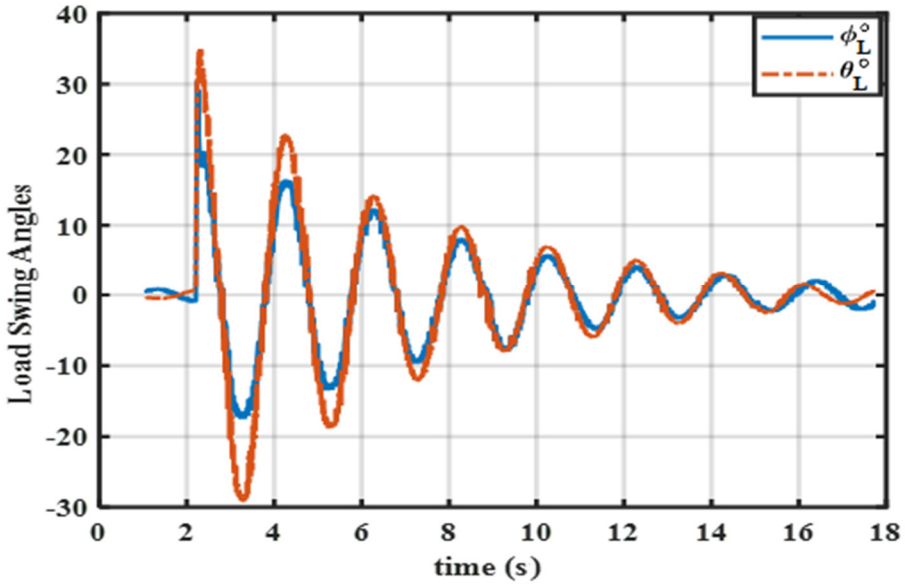

Figure 16. Time history of the load swing angle during hover without the ASC.

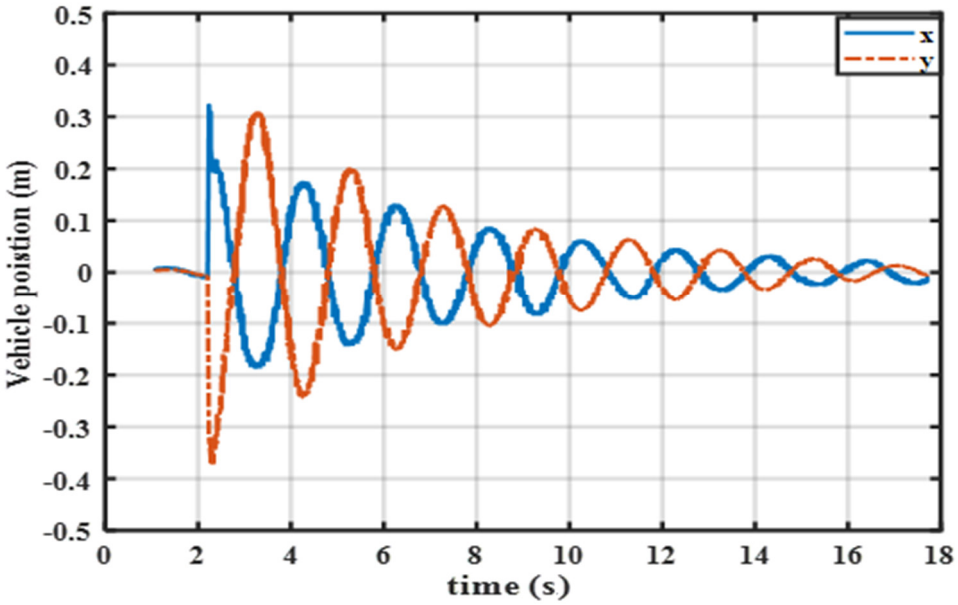

Figure 17. Time history of the quadrotor position during hover without the ASC.

In the second maneuver, the ASC is implemented as described in the previous section with the following parameters $K_{sx} = K_{sy} = 0.6L$ and n. After including the ASC in the control loop, it is observed that the oscillations history of the suspended load was improved in both directions, as shown in Figure 18. The deviation of the quadrotor from the hover is smaller compared with the uncontrolled case, as shown in Figure 19. The ISH determined for this maneuver is 0.4271. Therefore, the reduction is 36.7%.

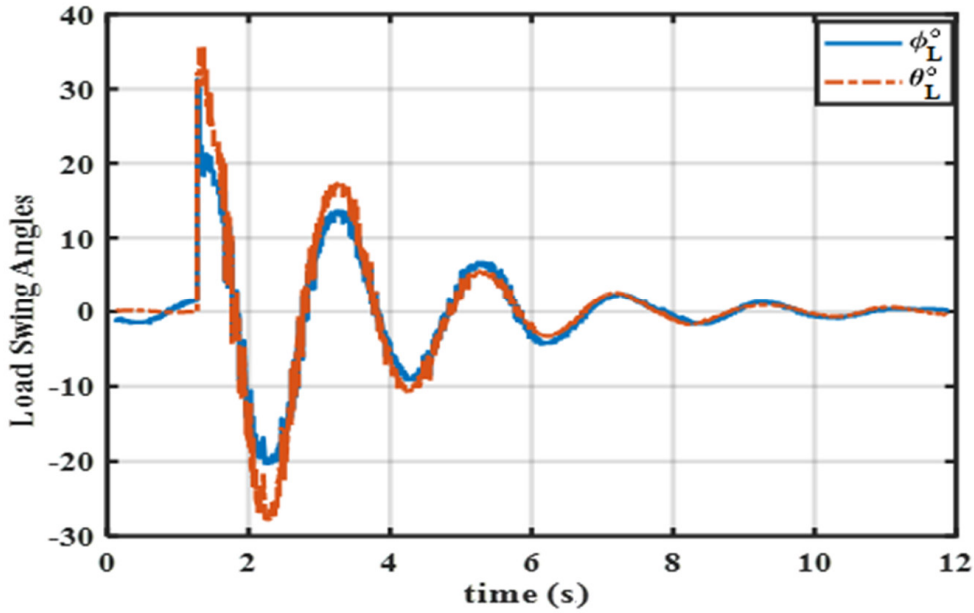

Figure 18. Time history of the load swing angle during hover with the ASC.

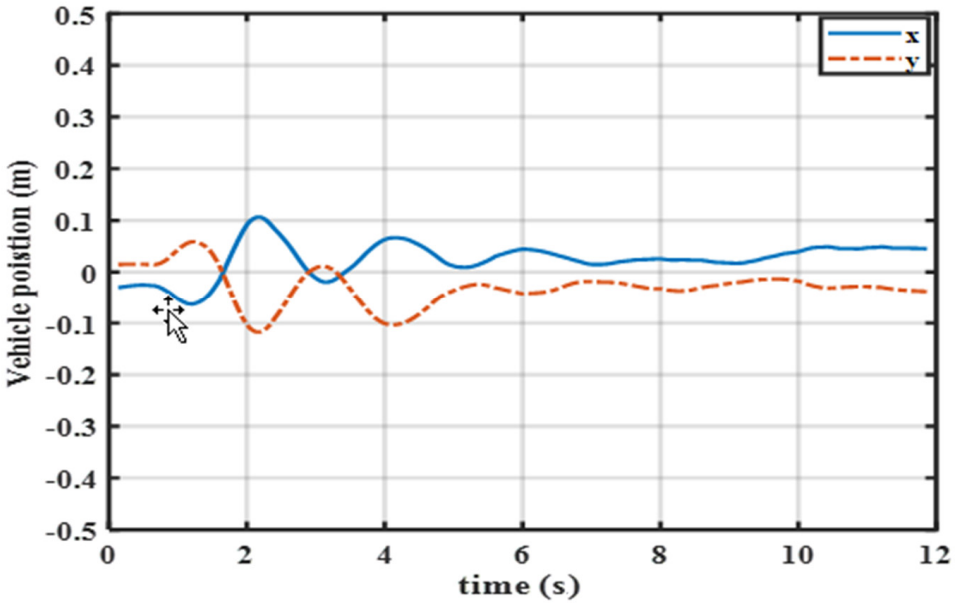

Figure 19. Time history of the quadrotor position during hover with the ASC.

5.2. Travel Flight

To measure the effectiveness of the ASC during travel flight, the quadrotor is subjected to a trajectory pattern that involves moving the quadrotor a step in the forward direction (x) by a distance of 3 m, then in the lateral direction (y) by a similar distance, and finally returning the vehicle to the hovering point. The ASC parameters are the same as the ones used for the hovering flight.

The time histories of the load swing angles are shown in Figures 20 and 21 with and without the ASC. The quadrotor trajectories in the x and y directions with and without the ASC are shown in Figures 22 and 23, respectively. It is observed that the oscillations were reduced when using the ASC. The ISH values are decreased from 3.7238 to 1.8930 when the ASC is implemented amounting to a reduction of about 49.2%. Moreover, the effects on the quadrotor trajectories are very small.

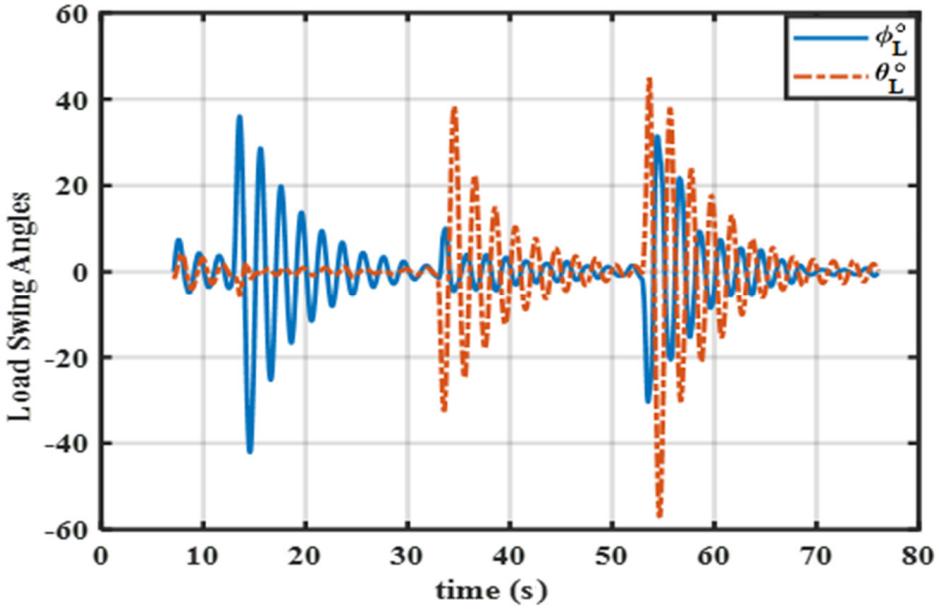

Figure 20. Time history of the load swing angle during travel maneuver without the ASC.

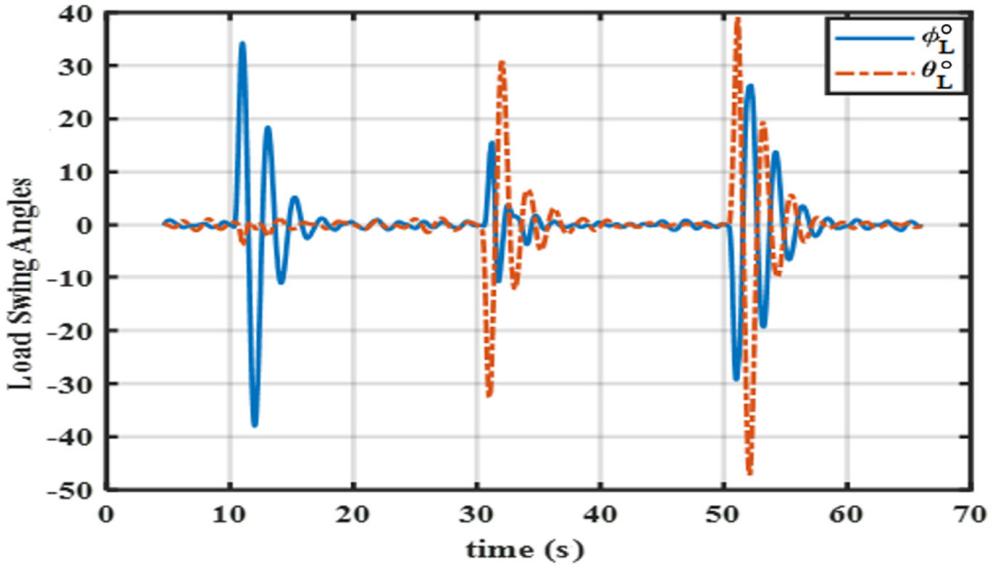

Figure 21. Time history of the load swing angle during travel maneuver with the ASC.

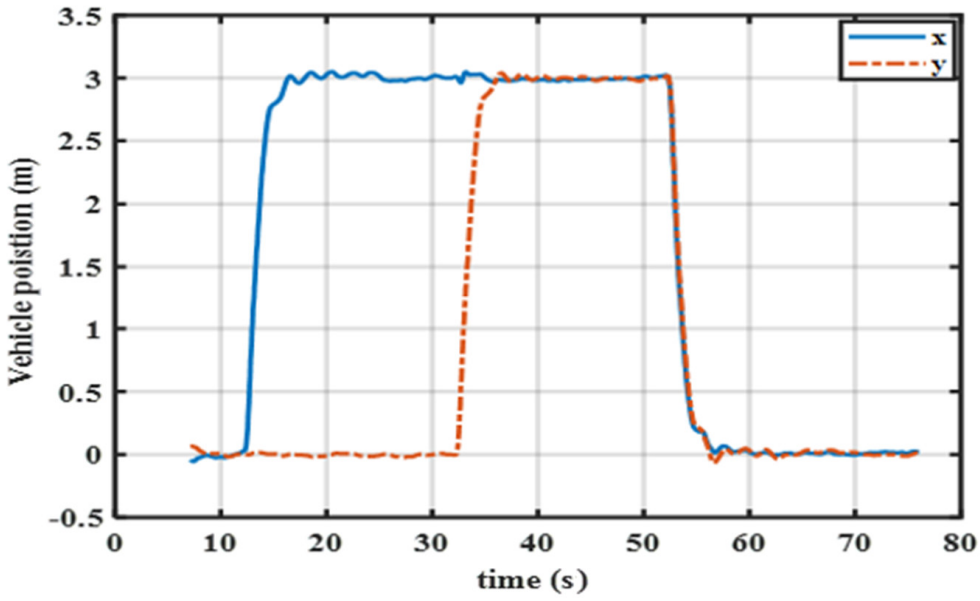

Figure 22. Time history of the quadrotor position during travel maneuver without the ASC.

5.3. Numerical Simulation

To check the results obtained from the developed testing platform, the proposed ASC is simulated numerically with a tracking controller that is similar to that one implemented in PX4 firmware, Figure 24. The time histories of the load swing angles with and without the ASC, due to an initial disturbance in hover flight, are shown in Figures 25 and 26, respectively. We can observe the same trend—the ASC can improve the damping of the suspended load oscillations. The differences in the initial swing angles are due to the method of generating these initials values. The same trend can be observed also in the forward flight.

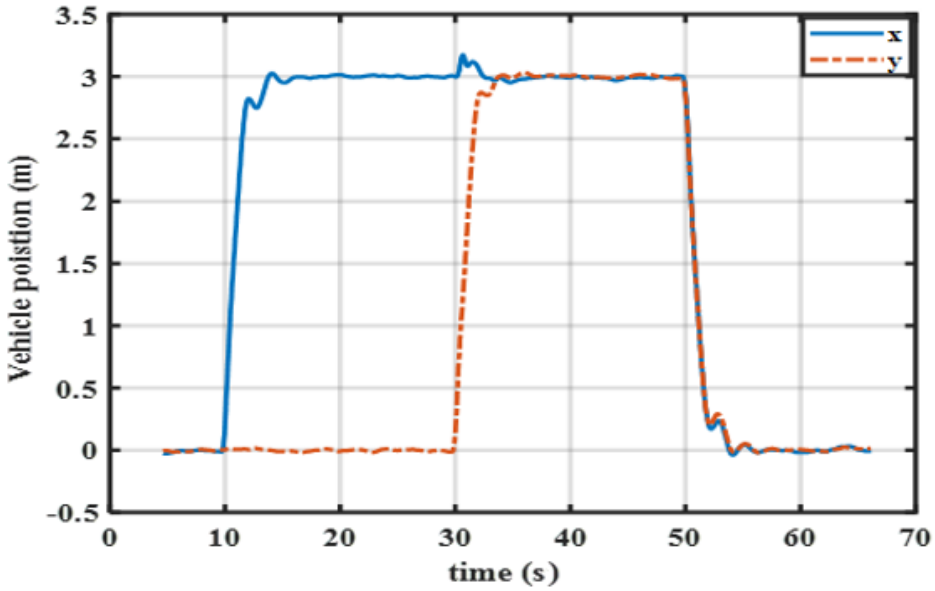

Figure 23. Time history of the quadrotor position during travel maneuver with the ASC.

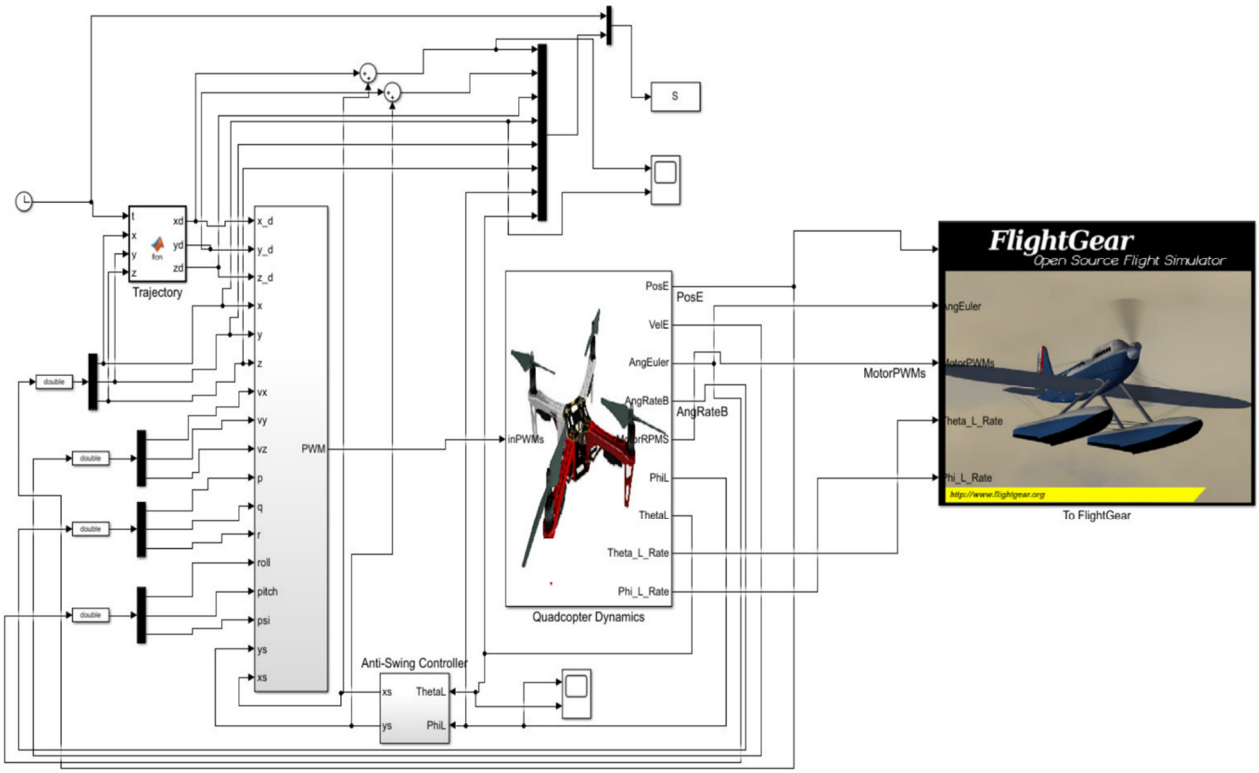

Figure 24. Simulink model of the anti-swing control system.

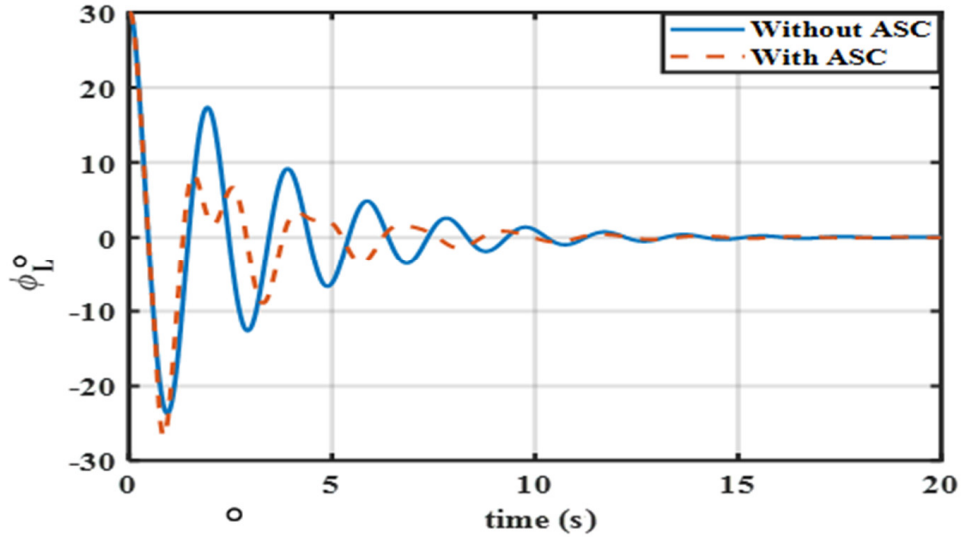

Figure 25. Time history of the in-plane load swing angle during hover using numerical simulations.

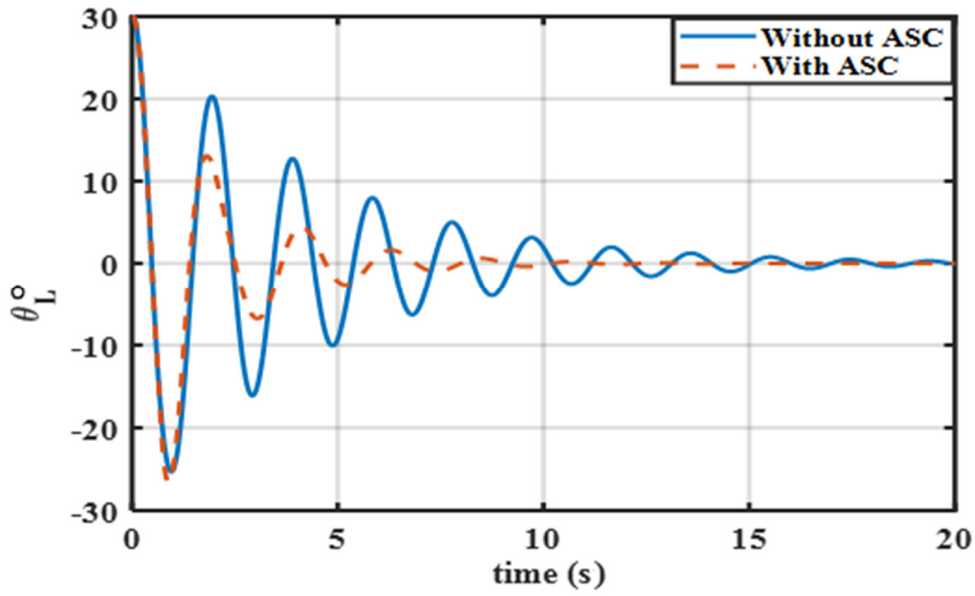

Figure 26. Time history of the out-of-plane load swing angle during hover using numerical simulations.

6. Conclusions

A robust testing platform was developed for quadrotors with suspended loads that is based on the hardware-in-the-loop simulation concept. The new platform allows the researchers to develop new controllers and test them safely and inexpensively for various flight scenarios and maneuvers, with the capability to simulate the disturbances that occur in a real flight. The Gazebo simulator provides researchers with a good tool to build the physical model with all the necessary sensors of the flight vehicles, without the need to deal with the complexity of the mathematical modeling. The usage of ROS facilitates real-time monitoring and logging of the flight.

The developed platform was used successfully to implement and test an anti-swing controller for the quadrotor suspended load system. This controller is based on the time-delayed feedback of the load swing angles. It was observed that the controller was able to improve the damping of the load oscillations in different flight scenarios. The testing platform was verified by comparing its results with the numerical simulations using Simulink modeling.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The researcher would like to thank the Deanship of Scientific Research, Qassim University, for funding the publication of this project.

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

References

1. Farid, G.; Hongwei, M.; Ali, S.M.; Liwei, Q. A review on linear and nonlinear control techniques for position and attitude control of a quadrotor. *Control Intell. Syst.* **2017**, *45*, 43–57.
2. Gupte, S.; Mohandas, P.I.T.; Conrad, J.M. A survey of quadrotor unmanned aerial vehicles. In Proceedings of the 2012 Proceedings of IEEE Southeastcon, Orlando, FL, USA, 15–18 March 2012; pp. 1–6.
3. Bucolo, M.; Buscarino, A.; Fortuna, L.; Gagliano, S. Bifurcation scenarios for pilot induced oscillations. *Aerosp. Sci. Technol.* **2020**, *106*, 106194. [CrossRef]
4. Bernard, M.; Kondak, K.; Maza, I.; Ollero, A. Autonomous transportation and deployment with aerial robots for search and rescue missions. *J. Field Robot.* **2011**, *28*, 914–931. [CrossRef]
5. Jung, S.; Kim, H. Analysis of amazon prime air UAV delivery service. *J. Knowl. Inf. Technol. Syst.* **2017**, *12*, 253–266.
6. Qian, L.; Graham, S.; Liu, H.H.-T. Guidance and Control Law Design for a Slung Payload in Autonomous Landing: A Drone Delivery Case Study. *IEEE/ASME Trans. Mechatron.* **2020**, *25*, 1773–1782. [CrossRef]
7. Nguyen, K.D.; Nguyen, T.-T. Vision-based software-in-the-loop-simulation for Unmanned Aerial Vehicles using gazebo and PX4 open source. In Proceedings of the 2019 International Conference on System Science and Engineering (ICSSE), Dong Hoi, Vietnam, 20–21 July 2019; pp. 429–432.
8. Silano, G.; Oppido, P.; Iannelli, L. Software-in-the-loop simulation for improving flight control system design: A quadrotor case study. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 466–471.
9. Taimoor, M.; Aijun, L. *Autonomous Flight of Unmanned Aerial Vehicle (UAV)*; LAP LAMBERT Academic Publishing: Saarbrücken, Germany, 2017.
10. Khan, H.S.; Kadri, M.B. Position control of quadrotor by embedded PID control with hardware in loop simulation. In Proceedings of the 17th IEEE International Multi Topic Conference 2014, Karachi, Pakistan, 8–10 December 2014; pp. 395–400.
11. Nguyen, K.D.; Ha, C. Development of Hardware-in-the-Loop Simulation Based on Gazebo and Pixhawk for Unmanned Aerial Vehicles. *Int. J. Aeronaut. Space Sci.* **2018**, *19*, 238–249. [CrossRef]
12. Bhargava, A. Development of a Quadrotor Testbed for Control and Sensor Development. Master's Thesis, Clemson University, Clemson, SC, USA, 2008.
13. Hancer, M.; Bitirgen, R.; Bayezit, I. Designing 3-DOF hardware-in-the-loop test platform controlling multirotor vehicles. In IFAC PapersOnLine, Proceedings of the 3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control PID 2018, Ghent, Belgium, 9–11 May 2018; Volume 51, pp. 119–124.
14. Wang, H.; Azaizia, D.; Lu, C.; Zhang, B.; Zhao, X.; Liu, Y. Hardware in the loop based 6DoF test platform for multi-rotor UAV. In Proceedings of the 4th International Conference on Systems and Informatics (ICSAI) 2017, Hangzhou, China, 11–13 November 2017; pp. 1693–1697.
15. Quan, Q.; Dai, X.; Wang, S. *Multicopter Design and Control Practice: A Series Experiments Based on MATLAB and Pixhawk*; Springer Nature: Singapore, 2020.
16. Meyer, J.; Sendobry, A.; Kohlbrecher, S.; Klingauf, U.; Von Stryk, O. Comprehensive simulation of quadrotor UAVS using ROS and gazebo. In *Simulation, Modeling, and Programming for Autonomous Robots, Proceedings of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Tsukuba, Japan, 5–8 November 2012*; Noda, I., Ando, N., Brugali, D., Kuffner, J.J., Eds.; Springer: Berlin, Germany, 2012; pp. 400–411.
17. Zhang, M.; Qin, H.; Lan, M.; Lin, J.; Wang, S.; Liu, K.; Lin, F.; Chen, B.M. A high fidelity simulator for a quadrotor UAV using ROS and Gazebo. In Proceedings of the IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society, Yokohama, Japan, 9–12 November 2015; pp. 002846–002851.
18. Alborzi, Y.; Jalal, B.S.; Najafi, E. ROS-based SLAM and navigation for a gazebo-simulated autonomous quadrotor. In Proceedings of the 2020 21st International Conference on Research and Education in Mechatronics (REM), Krakow, Poland, 9–11 December 2020; pp. 1–5.
19. Takaya, K.; Asai, T.; Kroumov, V.; Smarandache, F. Simulation environment for mobile robots testing using ROS and Gazebo. In Proceedings of the 2016 20th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 13–15 October 2016; pp. 96–101.

20. El-Ferik, S.; Syed, A.H.; Omar, H.M.; Deriche, M.A. Nonlinear forward path tracking controller for helicopter with slung load. *Aerosp. Sci. Technol.* **2017**, *69*, 602–608. [CrossRef]
21. Jazar, R.N. *Advanced Dynamics: Rigid Body, Multibody, and Aerospace Applications*; Wiley: Hoboken, NJ, USA, 2013.
22. Omar, H.M. Designing anti-swing fuzzy controller for helicopter slung-load system near hover by particle swarms. *Aerosp. Sci. Technol.* **2013**, *29*, 223–234. [CrossRef]
23. Freddi, A.; Lanzon, A.; Longhi, S. A Feedback Linearization Approach to Fault Tolerance in Quadrotor Vehicles. In IFAC Proceedings Volumes, Proceedings of the 18th IFAC World Congress, Milano, Italy, 28 August–2 September 2011; Volume 44, pp. 5413–5418. [CrossRef]
24. Voos, H. Nonlinear control of a quadrotor micro-UAV using feedback-linearization. In Proceedings of the 2009 IEEE International Conference on Mechatronics, Malaga, Spain, 14–17 April 2009; pp. 1–6.
25. Quan, Q. *Introduction to Multicopter Design and Control*; Springer Nature: Singapore, 2018.
26. PX4. PX4-Autopilot. Available online: https://github.com/PX4/PX4-Autopilot (accessed on 16 October 2021).
27. El Ferik, S.; Ahmed, G.; Omar, H.M. Load swing control for an Unmanned Aerial Vehicle with a slung load. In Proceedings of the IEEE 11th International Multi-Conference on Systems, Signals, and Devices, Barcelona, Spain, 11–14 February 2014; pp. 1–9.
28. Hanafy, O.; Saad, M. Integrating Anti-Swing Controller with PX4 Autopilot for Quadrotor with Suspended Load. *J. Mech. Sci. Technol.* **2022**, *36*. not published.
29. Gazebo. Gazebo Simulator. Available online: http://gazebosim.org (accessed on 15 October 2021).
30. Lamping, A.P.; Ouwerkerk, J.N.; Cohen, K. Multi-UAV control and supervision with ROS. In Proceedings of the 2018 Aviation Technology, Integration, and Operations Conference, Atlanta, GA, USA, 25–29 June 2018; p. 4245.
31. Lee, H.; Yoon, J.; Jang, M.-S.; Park, K.-J. A Robot Operating System Framework for Secure UAV Communications. *Sensors* **2021**, *21*, 1369. [CrossRef] [PubMed]
32. Rviz. Robotic Operating System. Available online: http://wiki.ros.org/rviz (accessed on 16 October 2021).
33. RQT. Robotic Operating System. Available online: http://wiki.ros.org/rqt (accessed on 16 October 2021).