International Journal of Aeronautical & Space Sciences (2018) 19:238–249
https://doi.org/10.1007/s42405-018-0012-8

IJASS International Journal of Aeronautical and Space Sciences

ORIGINAL PAPER

Development of Hardware-in-the-Loop Simulation Based on Gazebo and Pixhawk for Unmanned Aerial Vehicles

Khoa Dang Nguyen[1] · Cheolkeun Ha[1]

Received: 14 July 2017 / Revised: 3 February 2018 / Accepted: 5 February 2018 / Published online: 4 April 2018
© The Korean Society for Aeronautical & Space Sciences and Springer Nature Singapore Pte Ltd. 2018

Abstract
Hardware-in-the-loop simulation (HILS) is well known as an effective approach in the design of unmanned aerial vehicles (UAV) systems, enabling engineers to test the control algorithm on a hardware board with a UAV model on the software. Performance of HILS is determined by performances of the control algorithm, the developed model, and the signal transfer between the hardware and software. The result of HILS is degraded if any signal could not be transferred to the correct destination. Therefore, this paper aims to develop a middleware software to secure communications in HILS system for testing the operation of a quad-rotor UAV. In our HILS, the Gazebo software is used to generate a nonlinear six-degrees-of-freedom (6DOF) model, sensor model, and 3D visualization for the quad-rotor UAV. Meanwhile, the flight control algorithm is designed and implemented on the Pixhawk hardware. New middleware software, referred to as the control application software (CAS), is proposed to ensure the connection and data transfer between Gazebo and Pixhawk using the multithread structure in Qt Creator. The CAS provides a graphical user interface (GUI), allowing the user to monitor the status of packet transfer, and perform the flight control commands and the real-time tuning parameters for the quad-rotor UAV. Numerical implementations have been performed to prove the effectiveness of the middleware software CAS suggested in this paper.

Keywords HILS · Quad-rotor · Pixhawk · Gazebo

1 Introduction

In the unmanned systems community, UAVs are becoming more and more popular in the agriculture, transportation, navigation, and military fields. There are many types of UAVs, including fixed-wing [1], quad-rotor [2], hexacopter [3], and helicopter [4]. Among them, the quad-rotor UAV attracts significant interest from researchers due to its advanced characteristics such as simple structure, easy assembly and potential to hover, takeoff, and land in small areas.

Prior to the real flight of the UAV, quad-rotor UAV simulations need to be performed to reduce the risk of property damage resulting due to airframe crashes, system failure, or controller malfunction. To prevent these problems from occurring, HILS is an excellent solution to test the UAVs system on a real-time platform, combining a hardware device with simulation software. The hardware is used to install control algorithms on the microchip, while a simulation software provides the nonlinear 6DOF model of UAVs, sensor models, environment models, and visualization to simulate the UAV operation.

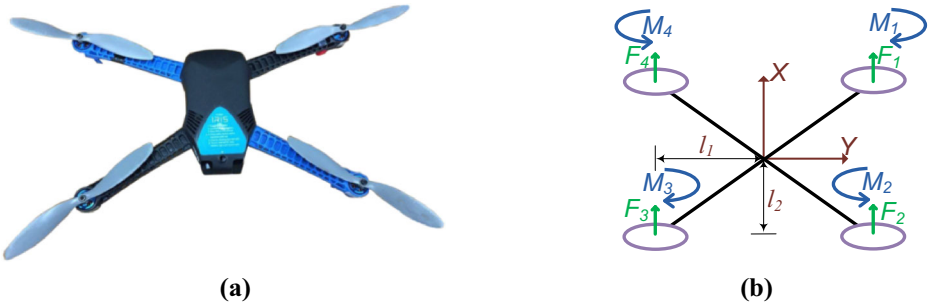
Some hardware, such as the AuRoRa board-STM32F3 Discovery board [5], PhycoreMPC5200B-tiny embedded board [6], dSpace dsll03 board [7], Rabbit board [8], and FPGA microchip [9], was chosen to develop HILS for UAVs. Although these boards provided the real-time platform, these solutions are limited due to the complicated hardware components, which are inflexible, expensive, and difficult to establish for various simulations.

In recent years, the Pixhawk hardware [10,11] has been widely used for UAV applications. It provides high-end autopilot hardware with low costs and high availability. Furthermore, the control algorithms in the open-source code PX4 [12,13], which is autopilot firmware used to drive unmanned aerial and ground vehicles, can be installed on this hardware.

Pixhawk can be coupled with some simulation software to establish the HILS system. The jMAVSim [14] has been used to develop HILS for a multi-rotor UAV simulation. However, it has limitations with regard to complex model. Researchers

✉ Cheolkeun Ha
cheolkeun@gmail.com

[1] School of Mechanical and Automotive Engineering, University of Ulsan, Ulsan 44610, Republic of Korea



(a) (b)

Fig. 1 Description of the quad-rotor UAV. **a** IRIS quad-rotor UAV. **b** Geometry of the quad-rotor UAV

have thus used other simulation software such as MAV3DSim [15], GEFS [16], FlightGear [17], and Xplane [18]. Although they provide a simulation framework, such software still requires significant amount of effort when testing sensor-based high-level control [19]. The cost of the commercial simulators needs to be considered in the development process.

To overcome these limitations, the Gazebo software [20] 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 [21]. ODE can provide real-time dynamical simulations and the corresponding sensor readings for the UAVs [22,23].

In the traditional HILS setup, the algorithm installed on the hardware is connected to a 6DOF model on the simulation software via either a direct connection or using middleware software. In the first case, the parts of HILS are connected without any information regarding the state of line transfer within the HILS system [14,17,18]. Subsequently, the communication of HILS is conflicted, the time responses are slow and the signals between the hardware and simulation software are lost. Therefore, the HILS performance is totally deteriorated. In the second case, the middleware software robot operating system (ROS) can be used to overcome the limitations of the first case [22,24]. To achieve this, the ROS needs to integrate all parts of the HILS, including the hardware and software. Before transferring a signal to another part, the data need to be attached to an ROS message. Consequently, the HILS system becomes more complex, limiting its applicability.

In this paper, a middleware software called the control application software (CAS) is designed to secure communications in the HILS system. Unlike the ROS, the CAS does not need to integrate the different HILS components. For configuration, three components are used to verify the performance of the CAS. First, the Gazebo software is used to construct the nonlinear 6DOF model and 3D visualization of the quad-rotor UAV based on 3D CAD model from the SolidWorks software. The sensor models are integrated with the Gazebo software through the plugins. Second, the flight control algorithm for the quad-rotor UAV is designed and implemented on Pixhawk using the open-source code PX4. Third, the CAS is robustly developed with a multithread structure in Qt Creator to synchronize signals, accelerate time responses, and reduce signal losses of the communication between the Gazebo and Pixhawk. Furthermore, the CAS can create line connections to any software and hardware using TCP, UDP, and serial protocol. Therefore, it does not need to integrate the parts of HILS as the ROS. The CAS provides a GUI that allows users to monitor the status of packet transfer and perform the tasks of a ground control station(GCS) software.

The remainder of this paper is organized as follows: Sect. 2 includes the problem description and the system modeling of a quad-rotor UAV, while the proposed HILS is introduced in Sect. 3. Numerical simulations are performed in Sect. 4 to illustrate the effectiveness of the CAS in HILS. Finally, some concluding remarks are given in Sect. 5.

2 Problem Description and System Modeling

2.1 General Configuration of the Quad-rotor UAV

In this paper, a quad-rotor UAV [25], which is shown in Fig. 1, is used for HILS development. The UAV contains a body frame that is connected with four electric motors. Let us use motors 1 and 3 as rotating in the clockwise direction. Similarly, motors 2 and 4 are set to rotate in the counter-clockwise direction. l_1 and l_2 are the distances along the axes to UAV's center gravity.

2.2 Mathematical Model of the Quad-rotor UAV

The quad-rotor UAV is controlled by the angular velocity of the four motors. Each motor produces forces and moments in the direction of the motor axis. Let us define F_i, M_i $(i = 1, 2, 3, 4)$ to be the force and moment of the ith motor, respectively. These can be defined as

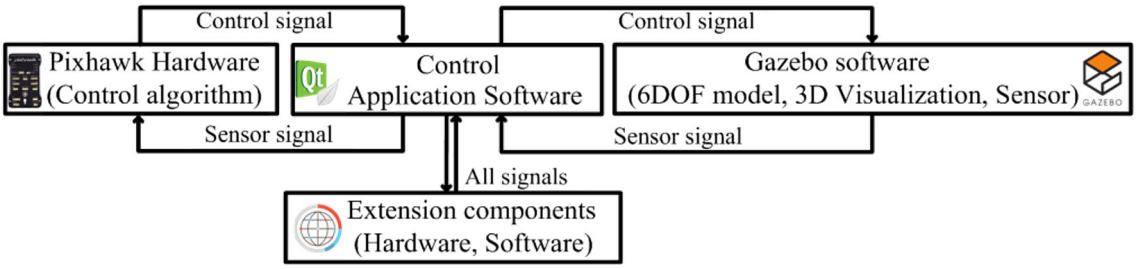


Fig. 2 Component Schema of HILS

$$F_i = K_f \omega_i^2 \tag{1}$$

$$M_i = K_m \omega_i^2 \tag{2}$$

where ω_i is the angular velocity of the ith motor; K_f and K_m are constant values of rotor thrust coefficient and rotor torque coefficient, respectively.

Suppose that the control inputs (U_1, U_2, U_3 and U_4) of the quad-rotor UAV are defined as follows:

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} K_f & K_f & K_f & K_f \\ -l_1 K_f & -l_1 K_f & l_1 K_f & l_1 K_f \\ l_2 K_f & -l_2 K_f & -l_2 K_f & l_2 K_f \\ -K_m & K_m & -K_m & K_m \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{3}$$

Based on these control inputs, the motion of the quad-rotor UAV can be divided into two subsystems, rotational subsystem roll (ϕ), pitch (θ) and yaw (ψ) and translational subsystem (x, y, z). The dynamic model of the quad-rotor UAV can be implemented as [26]

$$\begin{cases} \ddot{\phi} = \frac{1}{I_{xx}} \left(U_2 - J_r \dot{\theta} \omega_r + \left(I_{yy} - I_{zz} \right) \dot{\Psi} \dot{\theta} \right) \\ \ddot{\theta} = \frac{1}{I_{yy}} \left(U_3 - J_r \dot{\phi} \omega_r + \left(I_{zz} - I_{xx} \right) \dot{\phi} \dot{\Psi} \right) \\ \ddot{\Psi} = \frac{1}{I_{zz}} \left(U_4 + \left(I_{xx} - I_{yy} \right) \dot{\theta} \dot{\phi} \right) \\ \ddot{x} = \left(\cos(\phi) \cos(\psi) \sin(\theta) + \sin(\phi) \sin(\psi) \right) \left(\frac{U_1}{m} \right) \\ \ddot{y} = \left(\cos(\phi) \sin(\psi) \sin(\theta) - \sin(\phi) \cos(\psi) \right) \left(\frac{U_1}{m} \right) \\ \ddot{z} = \left(\cos(\phi) \cos(\theta) \right) \left(\frac{U_1}{m} \right) - g \end{cases} \tag{4}$$

where $\omega_r = \omega_1 - \omega_2 + \omega_3 - \omega_4$; I_{xx}, I_{yy}, I_{zz} are the moments of inertia about the principle axes in the body frame; J_r denotes the inertia of the motor; m is the total mass of quad-rotor UAV; and g is the gravitational constant.

3 HILS Design

The goal of HILS is to simulate the operation of a robot driven by the control algorithm in real-time conditions. Utilizing the mathematical model of the quad-rotor UAV presented in Sect. 2, a flight controller is designed and installed on the hardware board. The simulation software is required to configure the 6DOF model of the quad-rotor UAV, sensor models, and environment models to provide the state of the UAV as well as a 3D visualization for HILS. In fact, the hardware board and the simulation software are developed and work in different platforms. Communication conflicts can exist between them due to data transfer without knowing information about the line data transfer. To solve this problem, a control communication unit is indispensable to ensure connection between the hardware board and the simulation software in HILS.

Therefore, this section explains the component schema of HILS, as shown in Fig. 2. Herein, the Gazebo software is used to present the functions of the simulation software, while the Pixhawk hardware board is used to install flight control algorithms for the quad-rotor UAV. Based on the 6DOF model of the quad-rotor UAV in Gazebo software, the data measured from sensor models are generated and sent to the Pixhawk hardware board. Similarly, the control signals from the controller in Pixhawk are sent to the Gazebo software. The CAS is developed to ensure the connection and data transfer between Gazebo and Pixhawk. Some extension components in HILS, such as QGroundControl, other softwares and hardwares can connect to the CAS though a custom connection. This software uses the multithread method to avoid communication confliction, accelerate time response, and reduce package loss during communication.

3.1 Simulation Software Design

In this section, the Gazebo software is developed in three steps to construct the 6DOF model, sensor models and 3D visualization for simulating the operations of the quad-rotor UAV. The components of the Gazebo software are shown in Fig. 3 [20].

First, 3D CAD of the quad-rotor UAV was drawn in SolidWorks for the IRIS quad-rotor UAV from 3DR [27]. Herein, the user can define the parameters by material and color, and then, the mass and moment of inertia of the UAV are calcu-

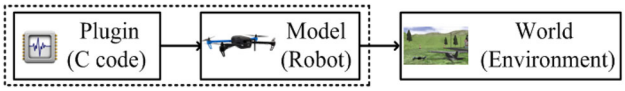


Fig. 3 Components of the gazebo software

lated automatically. Based on the configuration of the UAV in SolidWorks, the SW2URDF toolbox in SolidWorks [28] was used to generate a URDF file which describes all elements, the kinematic and dynamic properties of a single UAV in isolation. It does not, however, specify the pose of the robot itself within a world. To simulate this in the Gazebo, the pose of the UAV in the world needs to be added to the URDF file to create a new file called the SDF file, of which an example is shown in Fig. 4. The SDF file is loaded into the Gazebo to make a UAV model which has the 3D visualization, as shown in Fig. 5. This model can be applied the forces and moments in the 6DOF model using the open dynamics engine library in the gazebo [20].

Second, the sensor models and the motor control functions are added to the UAV model using the C programming language of the plugin. Here, the sensor models measure the state of the UAV to provide feedback data to the flight controllers in Pixhawk. The motor control functions are used to set forces and moments for the 6DOF model of the UAV based on the angular velocity of each motor, which is given from the flight controllers. It is also used to visualize the flight motion of the UAV in Gazebo.

The sensor models of the inertial measurement unit (IMU) and the global positioning system (GPS) are selected for installation on the quad-rotor UAV. The IMU sensor data contain information on the orientation (ξ), linear acceleration (a), and angular velocity (v_a). The GPS sensor model provides the latitude (δ), longitude (λ) and altitude (z). The Gazebo software is run to provide the IMU data (ξ, a, v_a), the current position (x, y, z), and the initial location (δ_0, λ_0, z_0) using the physics library. The current location (δ_1, λ_1, z_1) of the UAV can be calculated as [29]

Fig. 5 Visualization of the quad-rotor UAV

$$\begin{cases} \delta_1 = \arcsin\left(\cos(c)\sin(\delta_0) + \frac{x}{\rho}\sin(c)\cos(\delta_0)\right) \\ \lambda_1 = \lambda_0 + \arctan\left(\frac{y\sin(c)}{\rho\cos(c)\cos(\delta_0) - x\sin(c)\sin(\delta_0)}\right) \\ z_1 = z_0 + z \end{cases} \tag{5}$$

where R is the radius of Earth, and $c = \frac{\sqrt{x^2+y^2}}{R}$, $\rho = \sqrt{x^2+y^2}$

Based on the sensor models of IMU and GPS, the sensor data are generated and the disturbances are added to them in two individual plugins. In this simulation, the Gaussian noise and white noise are placed in each sensor, as shown in Table 1 [30,31]. To present and operate the real sensors in real-time conditions, the GPS and IMU signals are performed at the frequencies of 5 and 500 Hz, respectively [24]. Because the sensors are achieved in different plugins, they are collected in a general interface plugin to publish to the controllers. Sensor

```
<pose frame="">0 0 0 0 -0 0</pose>   Add pose for the UAV

<inertial>                          Original URDF file
  <origin
    xyz="0.025982 3.77287740202584E-18 0.061733"
    rpy="0 0 0" />
  <mass
    value="0.002477" />
  <inertia
    ixx="1.61247820352629E-06"
    ixy="5.61993810631176E-07"
    ixz="3.10511981052143E-24"
    iyy="1.28805193648675E-22"
    iyz="-3.65987310616101E-06"
    izz="2.22548464616382E-22" />
</inertial>
<visual>
  <origin
    xyz="1.5707963267949 0 0"
    rpy="0 0 0" />
  <geometry>
    <mesh
      filename="package://UAV/meshes/Quad.STL" />
  </geometry>
  <material
    name="">
    <color
      rgba="1 1 1 1" />
  </material>
</visual>
```

Fig. 4 Description of the SDF file

Table 1 Parameters Gaussian noise for IMU and GPS sensors

Sensor	Mean	Standard deviation
GPS	0	Lon/Lan/Alt = 0.3 mG
IMU	0	Gyro$X/Y/Z$ = 0.67/0.78/0.12 (°/s); Acc$X/Y/Z$ = 0.04/0.06/0.07 (m/s^2)

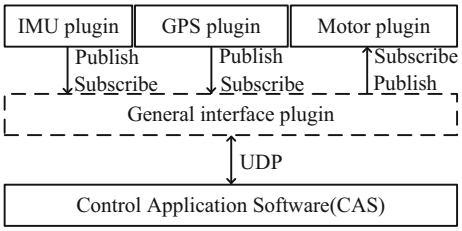


Fig. 6 Flowchart describing the send/receive task between the plugins in Gazebo

plugins can send the information of sensors through the [*transport::PublisherPtr*] class with the *Publish* method to the general interface plugin which can receive the data through the [*transport::SubscriberPtr*] class with the *Subscribe* method. For example, the code in the listing below describes for the *Publish* and *Subscribe* methods in the plugins.

For any operation of the quad-rotor UAV, the motors need to be set with the angular velocity that was described in Sect. 2. The motor control functions are defined in a motor plugin via two methods. The first method is to receive the signal control from the controller and use the inverse function of Eq. (3) to calculate the angular velocity for each motor. Then, this angular velocity is used to visualize the rotation of the propeller. The second method is to calculate the forces and moments in Eqs. (1, 2) for each motor, and then set to the 6DOF model of the quad-rotor UAV. Similar to the IMU and GPS plugins, the motor plugin also needs to use the general interface plugin to send/receive the data with the applications outside the Gazebo. A flowchart describing the send/receive task between the plugins in Gazebo is shown in Fig. 6. Without loss of generality, the UDP protocol is selected for transferring and receiving the signals of Gazebo in the HILS system. All signals are encoded in the MAVLink message packet [22]. The IMU, GPS and motor messages are defined in the list below.

```
//IMU and GPS plugin code
    transport::PublisherPtr  imu_pub_;  imu_pub_->Publish(imu_data_);
    transport::PublisherPtr  gps_pub_;  gps_pub_->Publish(gps_data_);
//General interface plugin code
    transport::SubscriberPtr  imu_sub_;  transport::SubscriberPtr  gps_sub_;
    imu_sub_ = node_handle_->Subscribe(imu_sub_topic_, &GazeboMavlinkInterface::ImuCallback, this);
    gps_sub_ = node_handle_->Subscribe(gps_sub_topic_, &GazeboMavlinkInterface::GpsCallback, this);
```

```
#pragma once                    // MESSAGE IMU (HIL_SENSOR)
#define MAVLINK_MSG_ID_HIL_SENSOR 107
MAVPACKED(typedef struct __mavlink_hil_sensor_t {  uint64_t time_usec;
float  xacc,yacc,zacc,xgyro,ygyro,zgyro, xmag,ymag,zmag; }) mavlink_hil_sensor_t;
#pragma once                    // MESSAGE GPS (HIL_GPS)
#define MAVLINK_MSG_ID_HIL_GPS 113
MAVPACKED(typedef struct __mavlink_hil_gps_t { uint64_t time_usec;
 int32_t  lat,lon,alt; }) mavlink_hil_gps_t;
#pragma once                    // MESSAGE MOTOR (HIL_CONTROLS)
#define MAVLINK_MSG_ID_HIL_CONTROLS 91
MAVPACKED(typedef struct __mavlink_hil_controls_t { uint64_t time_usec;
 float  roll_aileron, pitch_elevator,  yaw_rudder, thottle;  //U2//U3//U4//U1
}) mavlink_hil_controls_t;
```

Fig. 7 Visualization of the world file in the Gazebo software

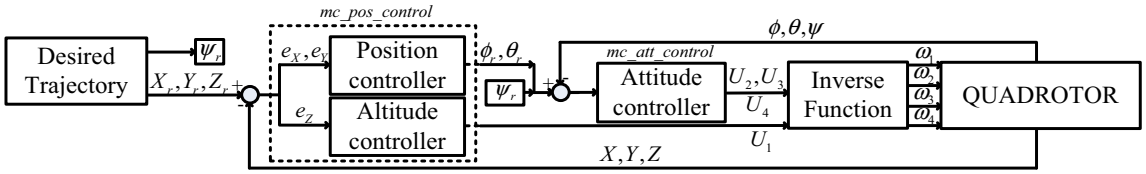


Fig. 8 Block diagram of a tracking controller for the quad-rotor UAV

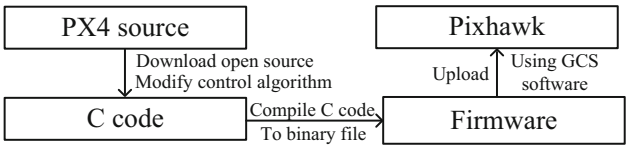


Fig. 9 Compiling and uploading the firmware to Pixhawk

Third, the quad-rotor UAV model and the plugins are put into a world file to setup the working environment for the simulations. In the world file, we can design other models, including trees, wind, houses and cars. Real-time conditions are also configured in this file based on the ODE. An example display of the world file in the Gazebo software is shown in Fig. 7.

3.2 Pixhawk Development

The Pixhawk hardware board is used to install the flight control algorithm for the quad-rotor UAV, providing the real-time environment based on the NuttX real-time operation system [10]. A tracking control algorithm is designed, as shown in Fig. 8.

Here, the quad-rotor UAV is driven to follow the trajectory described by the position (X_r, Y_r, Z_r) and yaw (ψ_r). The altitude, position, and attitude controller are designed to generate the force and moment (u_1, u_2, u_3 and u_4), as presented in Sect. 2. The altitude controller refers to the error distance (e_z) as the input and results in the u_1 signal. Meanwhile, the input of the position controller uses the error position (e_x, e_y) and its output provides roll (ϕ_r) and pitch (θ_r) which are combined with the yaw reference to make desired inputs to the attitude controller. The output signals of this controller are put into the inverse function of Eq. (3) to obtain angular velocities (w_1, w_2, w_3, and w_4) for the motors in the quad-rotor UAV.

To implement the control algorithm, the PID controller [13] is selected for the altitude, position, and attitude controller. The IMU and GPS sensor models in Gazebo are used to generate feedback signals(ϕ, θ, ψ; X, Y, Z). Based on the libraries of the open-source PX4, the controllers are written in the *mc_pos_control* and *mc_att_control* files using C programming language, and these codes are compiled in firmware that is loaded to the microchip of the Pixhawk hardware. The process from PX4 source to uploading the firmware to Pixhawk is shown in Fig. 9.

In our research, the softwares that provide the editor code, compiler code and upload firmware, are shown in Table 2.

3.3 Control Application Software Design

The Gazebo software is designed to simulate the tasks of the quad-rotor UAV under the flight controllers which are installed in the Pixhawk hardware. The CAS is developed to ensure data communication between the Gazebo and the Pixhawk. It is designed with custom connections and service that is required from the hardwares and softwares in the HILS

Table 2 List softwares use in the proposed HILS

Name	Version	Description
PX4 source	v1.5.0	Open source for developing controllers
Qt creator	v5.7.0	Editor code, installed on Ubuntu OS
Ground control station (GCS)	v3.0.1	Upload firmware, named QGroundControl

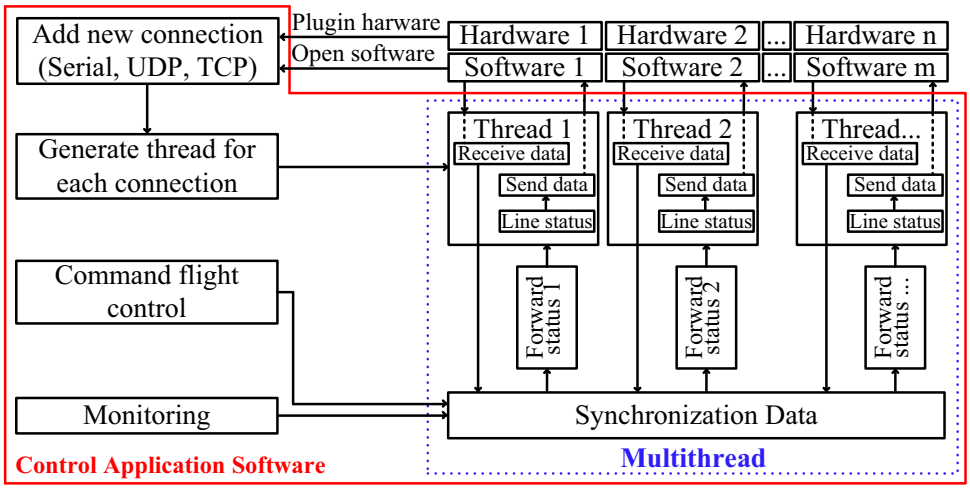


Fig. 10 Structure of the CAS

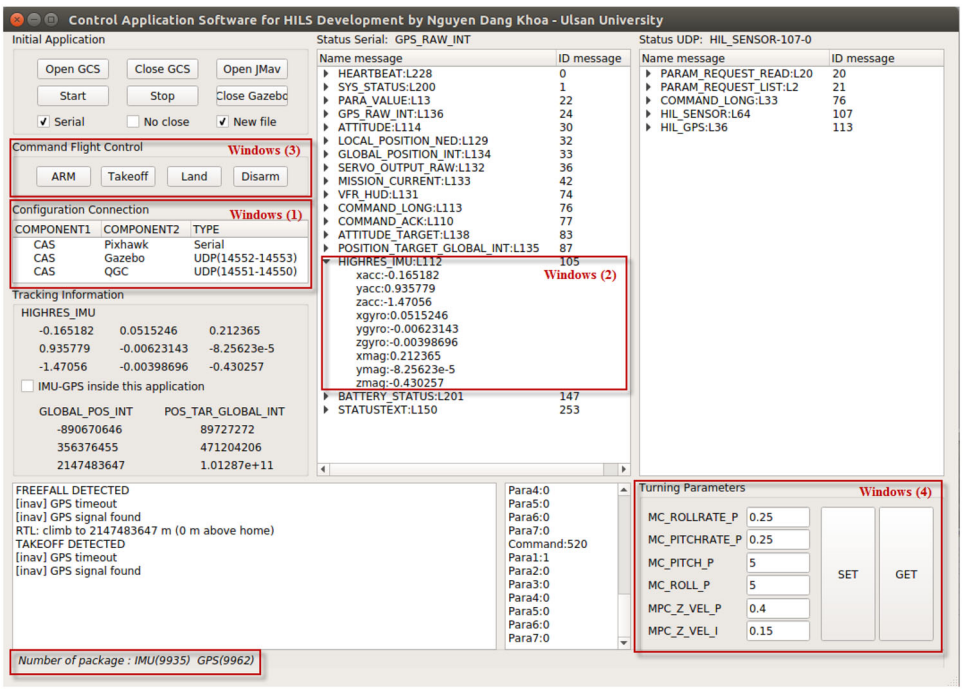


Fig. 11 GUI of the CAS

system. The detailed structure of the CAS proposed in this paper is presented in Fig. 10.

Here, the CAS includes a connection module, thread module, command flight control module and monitoring module. The connection module is used to define the interface between the CAS and other applications on the hardware (software) through the serial, UDP and TCP connections. Therefore, the middleware software CAS does not need to be integrated with the other components of the HILS like the ROS. To up speed for the communication in HILS, the thread module provides the threads corresponding to each connection. The threads can be activated in parallel at the same time to control the process of receiving and sending data with the applications. To avoid conflicts on each thread and on all systems, the synchronization method is used to synchronize data, and the threads often check the line status before transferring data to the applications. Furthermore, the CAS defines the flag *forward status* to limit the number of packets to each thread, because the applications need only some of the information that is transferred in the HILS system. The monitoring and the command flight control modules allow the users either to track the state of the HILS system or to send custom commands and messages to the threads.

The CAS is implemented based on the C programming language in Qt Creator. The GUI of CAS is shown in Fig. 11.

The GUI in Fig. 11 provides monitoring information of the packet in HILS. For example, the identification, name, and value of the packets in the serial and UDP connections

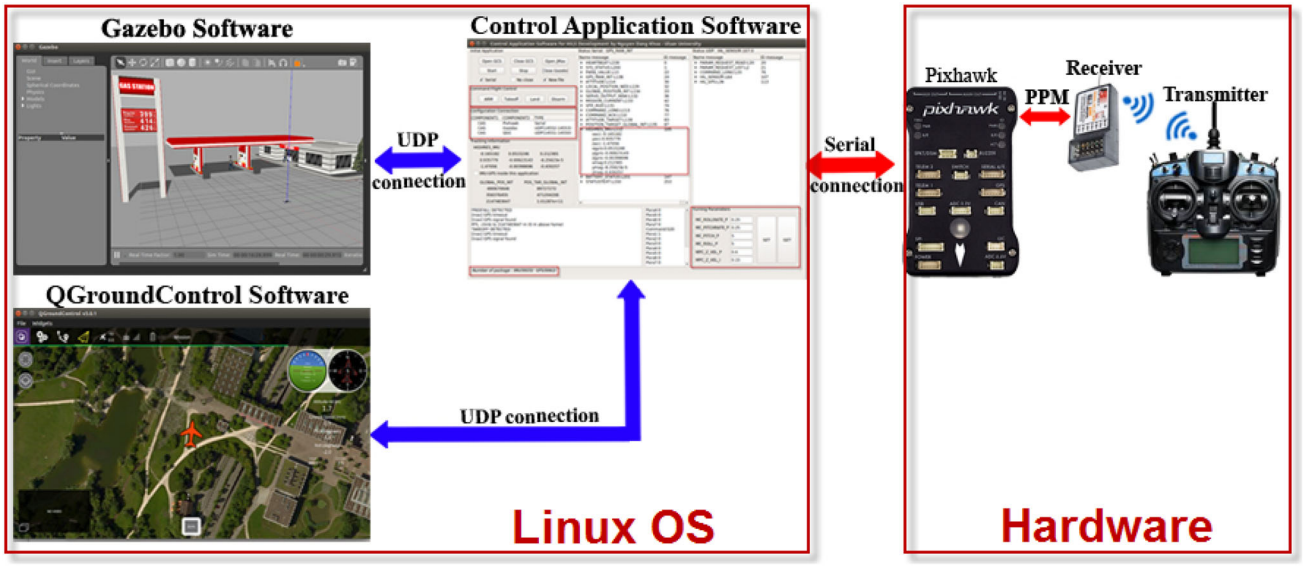


Fig. 12 Layout of the overall HILS

Fig. 13 HILS setup

Table 3 Setting frequency for the signal in HILS

Component name	Frequency (Hz)	Description
IMU sensor	500	HIL_SENSORS – Generation from Gazebo
GPS sensor	5	HIL_GPS – Generation from Gazebo
Actuator control	400	HIL_CONTROLS – Generation from Pixhawk

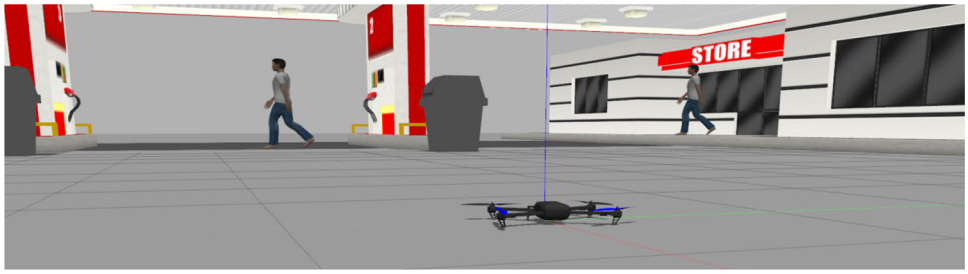


Fig. 14 Workspace of the quad-rotor UAV in Gazebo

are shown on windows (1) and (2) in Fig. 11. Furthermore, in window (3), the flight control commands, such as arm/disarm motors, takeoff, and land, can be utilized in the CAS to control the quad-rotor UAV. The gains *(P, I, D)* for the PID flight controllers are also tuned in real time to improve the performance of the controllers in window (4).

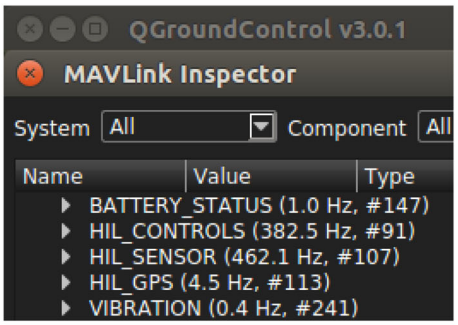
Based on the functional effectiveness of the CAS, HILS for the quad-rotor UAV is built, as shown in Fig. 12.

In the configuration of HILS in Fig. 12, the CAS, Gazebo, and GCS software (QGroundControl) are performed on a computer based on the Linux operating system. Meanwhile, the hardware includes the Pixhawk board, transmitter, and receiver. For the communication protocol, the CAS creates a serial to Pixhawk and two UDP connections to the Gazebo and QGroundControl software. This means that three threads are made on the CAS to maintain data flow in the HILS system. All packets in HILS are transferred through the serial and UDP connections using the MAVLink message, which is popular in UAV applications.

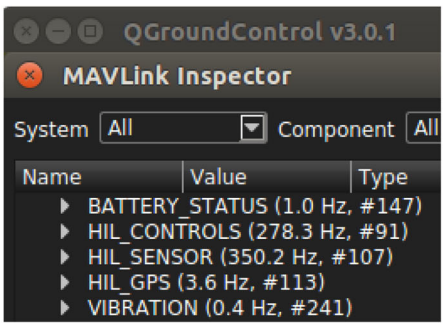
4 Implementation of HILS and Results

In this section, the effectiveness of the proposed CAS as well as HILS, implemented in the setup, as shown in Fig. 13, is demonstrated. To assess the capability of the CAS, the quad-rotor UAV flights are performed in station-keeping hovering mode which has the desired position $X = 0$ (m), $Y = 0$ (m), $Z = 2.5$ (m) and the desired yaw angle is 0°. The dimensions of the rotor to the mass center of the quad-rotor UAV were calculated based on the experimental setup [27] with $L = 55$ (cm). The sampling frequency of the sensors and the actuator controls are given in Table 3, and are also given for the real quad-rotor UAV [22]. As shown in Fig. 14, a workspace of the quad-rotor UAV is created in the Gazebo software, where the initial position of the quad-rotor UAV is set on the ground. The simulation environment looks like the natural world operated in real time.

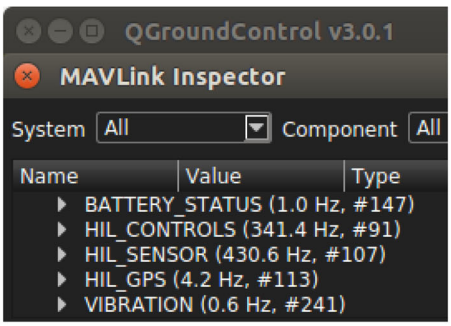
For comparison, the proposed CAS (denoted TCAS) is investigated using software-in-the-loop simulation (SITL) [32] and traditional HILS with single thread (denoted TS). Herein, SITL is simulated by PX4 source, Gazebo, and QGC.



(a)



(b)



(c)

Fig. 15 Frequency in QGroundControl. **a** SITL. **b** TS. **c** TCAS

For communication in SITL, we use the same packet of HILS: HIL_SENSORS, HIL_GPS, and HIL_CONTROLS.

Based on the above configuration, the SITL, TS, and TCAS are performed to evaluate communication performance. For each simulation, QGC software is used to measure the signal frequencies in the system, which is popu-

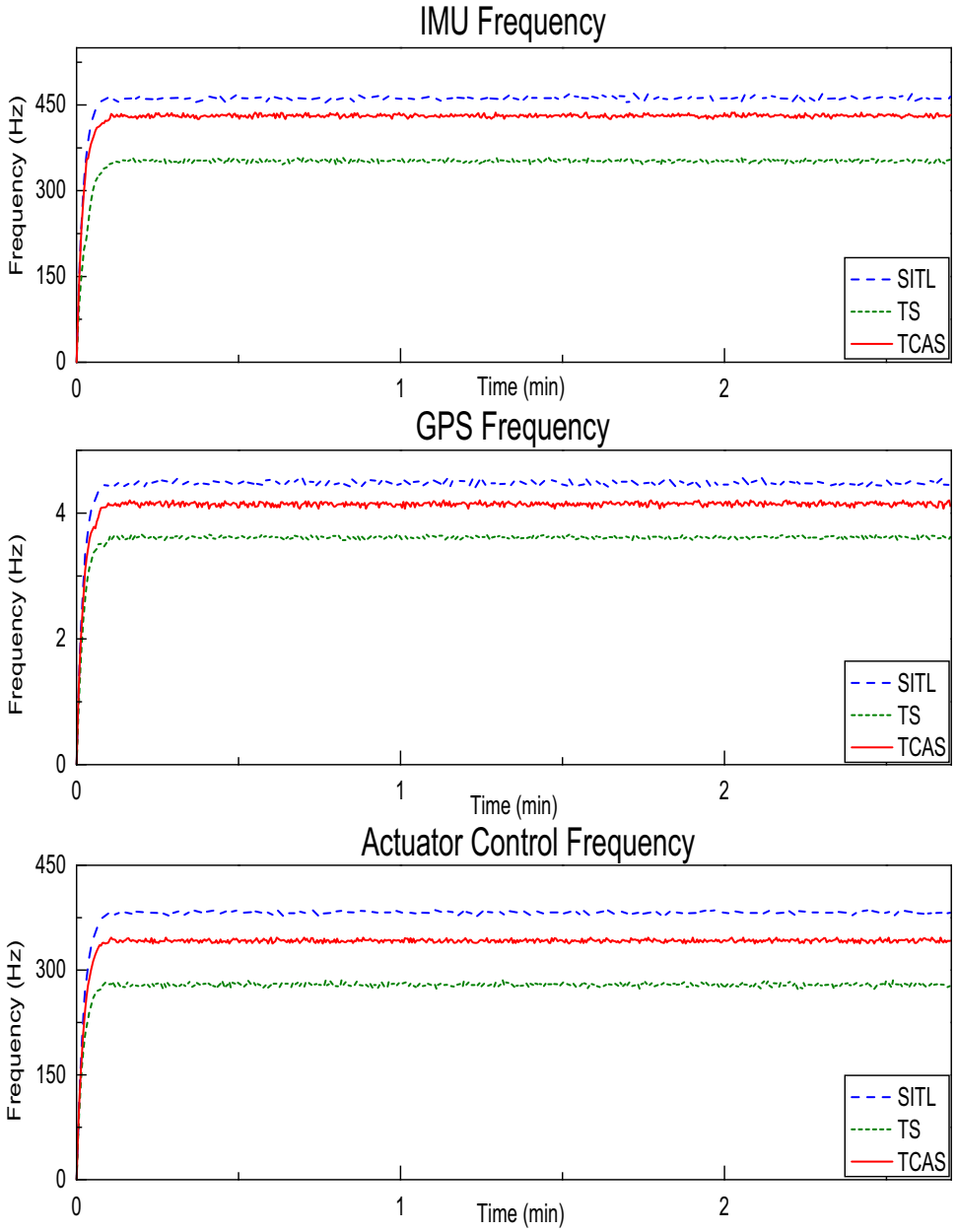


Fig. 16 Frequency in QGroundControl

lar for the ground control station. As displayed in Fig. 15, the sampling frequencies of the sensor signals (HIL_SENSORS, HIL_GPS) and actuator control signals (HIL_CONTROLS) do not reach the frequency source in Table 3.

The different receive frequencies are shown clearly in Fig. 16 and Table 4. The results indicate that the TS has the worst performance signal due to the lack of management signals between the components of HILS. Subsequently, conflict communication, signal losses, and time responses are influenced by the superior performance signal in SITL and TCAS compared to TS. This is because SITL is built based on all of the softwares. The communication between the parts has the best performance due to the support of the operating system (OS). The TCAS with the management signal in the CAS can resolve the limitations in TS using multithread architecture. The sampling frequency of the signal is improved in HILS. Due to significant degradation in the sampling frequency, the performance of the quad-rotor UAV deteriorates. As shown in Fig. 17 and Table 5, the quad-rotor UAV in the TS simulation did not reach the desired goal. The SITL and TCAS provide superior performance to TS. All results convincingly indicate that the proposed CAS approach could establish a better HILS in terms of communication speed and performance in the quad-rotor UAV.

Table 4 Frequency and ratio errors in simulations

Name	IMU		GPS		Control	
Frequency source	500 Hz		5 Hz		400 Hz	
	Average Hz	Ratio error (%)	Average Hz	Ratio error (%)	Average Hz	Ratio error (%)
SITL	462.7284	7.4543	4.4901	10.1972	382.9965	4.2509
TS	352.8847	29.4231	3.6218	27.5650	279.4541	30.1365
TCAS	431.8999	13.6200	4.1525	16.9509	342.6261	14.3435

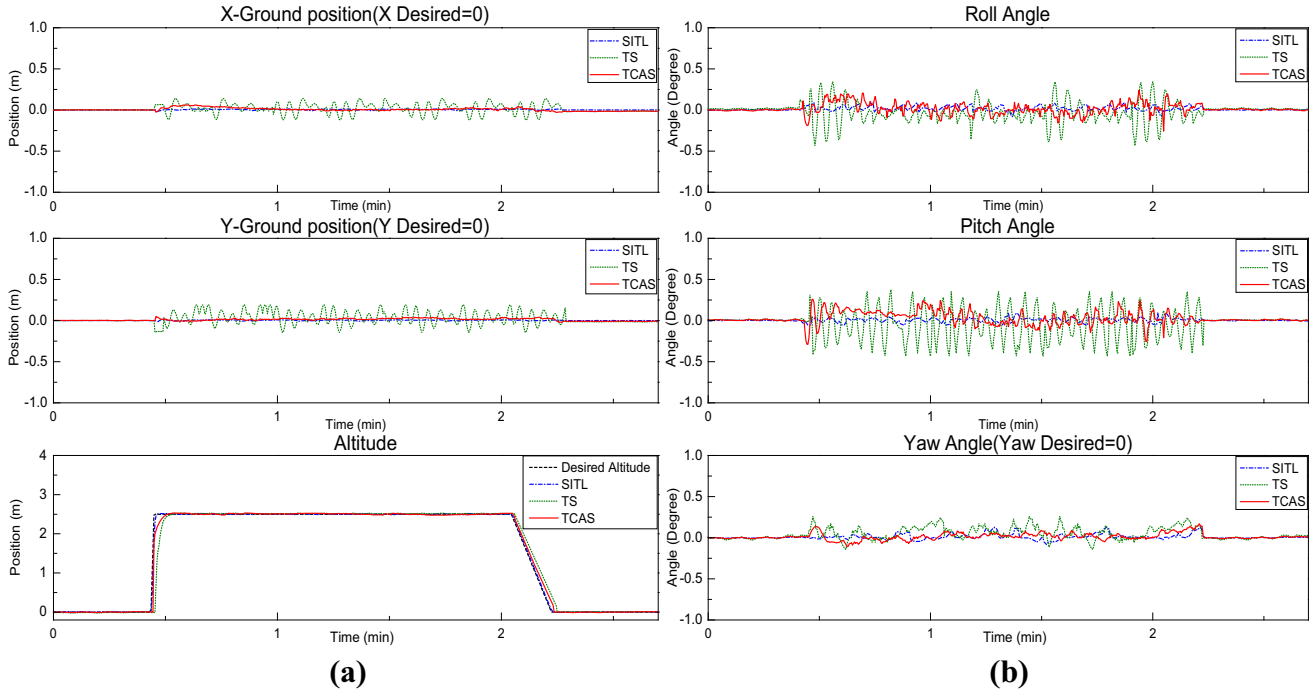


Fig. 17 Response of quad-rotor UAV. **a** Position. **b** Attitude

Table 5 RMS error for the performance of quad-rotor UAV

RMS error	X	Y	Z	Yaw angle
SITL	0.0068	0.0050	0.0449	0.0342
TS	0.0581	0.0790	0.2131	0.0866
TCAS	0.0196	0.0179	0.0996	0.0429

5 Conclusion

This paper proposed the CAS to control and secure communications in HILS system. A HILS setup was presented to evaluate flight control algorithms for a quad-rotor UAV. The proposed HILS, which is combined with three components, the Gazebo software, the Pixhawk hardware and the CAS, worked well. Here, the Gazebo software is developed to construct the 6DOF model, sensor models, 3D visualization and environment workspace of the UAV, while the Pixhawk hardware is used to install the flight control algorithms based on the open-source code PX4. The CAS is developed to ensure communication among the Gazebo, Pixhawk, and the extension components in HILS. The CAS is designed to reduce the time response and signal loss in the data transfer process using the multithread architecture. The CAS allows the user to send flight control commands and enter the real-time tuning parameters for the quad-rotor UAV as a GCS. The CAS has good performance in the management signal in HILS. It was also compared with the SITL and the traditional HILS setup. Furthermore, the HILS setup was evaluated under real-time conditions in a quad-rotor UAV operated by the flight control algorithm.

Acknowledgements This work was supported by the 2015 Research Fund of University of Ulsan.

References

1. Arifianto O, Farhood M (2015) Optimal control of a small fixed-wing UAV about concatenated trajectories. Control Eng Pract 40:113–132

2. Raharja GB, Beom KG, Yoon KJ (2010) Design of an autonomous hover control system for a small quadrotor. Int J Aeronaut Space Sci 11(4):338–344. https://doi.org/10.5139/IJASS.2010.11.4.338
3. Lee J, Choi HS (2016) Fault tolerant control of hexacopter for actuator faults using time delay control method. Int J Aeronaut Space Sci 17:54–63. https://doi.org/10.5139/IJASS.2016.17.1.54
4. Do-Hyung K, Tae-Joo K (2016) Test and simulation of an active vibration control system for helicopter applications. Int J Aeronaut Space Sci 17(3):442–453. https://doi.org/10.5139/IJASS.2016.17.3.442
5. Pizetta IHB, Brandao AS, Sarcinelli-Filho M (2016) A hardware-in-the-loop platform for rotary-wing unmanned aerial vehicles. J Intell Robot Syst 84:725–743. https://doi.org/10.1007/s10846-016-0357-9
6. Simon D (2011) Hardware-in-the-loop test-bed of an unmanned aerial vehicle using Orccad. In: 6th National conference on control architectures of robots, France
7. Bayrakceken MK, Yalcin MK, Arisoy A, Karamancioglu A (2011) HIL simulation setup for attitude control of a quadrotor. In: Proceedings of the 2011 IEEE international conference on mechatronics, Turkey, pp 354–357
8. Santos SRB, Sidney SNGJ, Cairo LNJ, Adriano B, Neusa MFO (2011) Modeling of a hardware-in-the-loop simulator for UAV autopilot controllers. In: Proceedings of COBEM, Brazil
9. Khan HS, Kadri MB (2014) Position control of quadrotor by embedded PID control with hardware in loop simulation. In: IEEE 17th international multi-topic conference, Pakistan
10. Meier L, Honegger D, Pollefeys M (2015) PX4: a node-based multithread open source robotics framework for deeply embedded platforms. In: IEEE international conference on robotics and automation, USA, pp 6235–6240
11. Fraundorfer F, Heng L, Honegger D, Lee GH, Meier L, Tanskanen P, Pollefeys M (2012) Vision-based autonomous mapping and exploration using a quadrotor MAV. In: IEEE/RSJ international conference on intelligent robots and systems, Portugal, pp 4557–4564
12. PX4 Team (2013) Pixhawk px4 autopilot. https://pixhawk.ethz.ch/px4/en/start
13. Bolandi H, Rezaei M, Mohsenipour R, Nemati H, Smailzadeh SM (2013) Attitude control of a quadrotor with optimized PID controller. Intell Control Autom 4:335–342
14. Rendy W, Bambang RT, Egi H (2016) Hardware-in-the-loop simulation of UAV hexacopter for chemical hazard monitoring mission. In: 2016 6th international conference on system engineering and technology, Indonesia, pp 189–193
15. Cardenas IL, Salazar S, Lozano R (2016) The MAV3DSim hardware in the loop simulation platform for research and validation of UAV controller. In: International conference on unmanned aircraft systems, USA, pp 1335–1341
16. GeoFS (2017) Gefs: free online flight simulator. http://www.gefsonline.com
17. Prabowo YA, Trilaksono BR, Triputra FR (2015) Hardware in the loop simulation for visual servoing of fixed wing UAV. In: International conference on electrical engineering and informatics, pp 247–252
18. Korkmaz H, Ertin OB, Kasnakoglu C, Kaynak U (2013) Design of a flight stabilizer system for a small fixed wing unmanned aerial vehicle using system identification. In: 1st IFAC workshop on advances in control and automation theory for transportation applications, vol 46, no. 25, pp 145–149
19. Mengmi Z, Hailong Q, Menglu L, Jiaxin L, Shuai W, Kaijun L, Feng L, Ben MC (2015) A high fidelity simulator for a quadrotor UAV using ROS and Gazebo. In: IECON 2015—41st annual conference of the IEEE industrial electronics society, pp 2846–2851
20. Sourceforge (2017) Player/stage/gazebo. http://sourceforge.net/projects/playerstage
21. Carlos EA, Nate K, Ian C, Hugo B, Steven P, John H, Brian G, Steffi P, Jose LR, Justin M, Eric K, Gill P (2015) Inside the virtual robotics challenge simulating real-time robotic disaster response. IEEE Trans Autom Sci Eng 12(2):494–506
22. Odelga M, Stegagno P, Heinrich HB, Ahmad A (2015) A setup for multi-UAV hardware in the loop simulations. In: 2015 workshop on research education and development of unmanned aerial systems, Mexico, pp 204–210
23. Thomio W, Gustavo N, Romulo C, Tiago T, Marco R, Sylvain J, Jan A (2015) The rock-gazebo integration and a real-time AUV simulation. In: 2015 12th Latin American robotics symposium and 2015 third Brazilian symposium on robotic, Brazil, pp 132–138
24. Qing B, Fuhua W, Zhen X, Qinhu R, Jianhua Z, Sheng L (2015) General simulation platform for vision based UAV testing. In: 2015 IEEE international conference on information and automation, China, pp 2512–2516
25. Yang Y, Yan Y (2016) Attitude regulation for unmanned quadrotors using adaptive fuzzy gain scheduling sliding mode control. Aerosp Sci Technol 54:208–217
26. Xiong JJ, Zhang GB (2017) Global fast dynamic terminal sliding mode control for a quadrotor UAV. ISA Trans 66:233–240
27. 3DR company (2017) Support IRIS quad-rotor. https://3dr.com/support/articles/iris/
28. Gazebosim (2017) Gazebo. http://gazebosim.org
29. Snyder JP (1987) Map projections—a working manual. Government Printing Office, Washington
30. Nirmal K, Sreejith AM, Mathew J, Mayuresh S, Suresh A, Prakash A, Safonova M, Murthy J (2016) Noise modeling and analysis of an IMU-based attitude sensor: improvement of performance by filtering and sensor fusion. In: SPIE astronomical telescopes and instrumentation symposium, UK
31. Stewart SM, Holt GN (2003) Real-time attitude determination of a nanosatellite using GPS signal-to-noise ratio observations. University of Texas at Austin, Austin
32. Gazebo simulation (2017). https://dev.px4.io/en/simulation/gazebo.html