A Model-Based Design to Implement Controllers for Quadrotor Unmanned Aerial Vehicles

HIEN N.V., DIEM P.G.
Department of Aeronautical and Space Engineering, School of Transportation Engineering
Hanoi University of Science and Technology
N^{o}1, Dai Co Viet, Hai Ba Trung, Hanoi
VIETNAM
[hien.ngovan, diem.phamgia]@hust.edu.vn http://www.hust.edu.vn

Abstract: - This paper presents a model-driven control design, which is based on the specialization of Model-Based Systems Engineering (MBSE) approach combined with the real-time UML/MARTE, hybrid automata and the Extended Kalman Filter (EKF) algorithm in order to conveniently implement and deploy controllers for quadrotor Unmanned Aerial Vehicles (UAVs). This model also creates a real-time communication pattern, which can permit the designed control components of a developed quadrotor UAV to be customized and reused in new control applications of various UAV typed Vertical Take Off and Landing (VTOL). To achieve this goal, the study is stepwise carried out as follows: the physical and dynamic model together with control structure of a quadrotor UAV are firstly adapted for developing entirely a quadrotor UAV controller. The use-case model combined with the realization hypotheses of hybrid automata and the EKF algorithm are then specialized to gather the requirement analysis of control. The specializations of real-time UML/MARTE's features such as the 'capsules, ports and protocols' notation combined with the timing concurrency of evolution are next realized to precisely design structures and behaviors for the controller. The detailed design model is then converted into the implementation model by using object-oriented and open-source platforms in order to quickly simulate and realize this controller. Finally, a trajectory-tracking controller is developed and tested that permits an autonomous quadrotor UAV to reach and follow a reference trajectory in the *Cartesian* space with good reliability.

Key-Words: - UAV control; autonomous flying robot; model-based control design; systems engineering; hybrid automata; EKF; real-time UML/MARTE; MBSE.

Nomenclature

CLF	Control *Lyapunov* Functions	MDE	Model-Driven Engineering
DoF	Degrees of Freedom	OMG	Object Management Group
EKF	Extended *Kalman* Filters	OO	Object-Oriented
GPS	Global Positioning System	OOA	Object-Oriented Analysis
HA	Hybrid Automata	OOD	Object-Oriented Design
HDS	Hybrid Dynamic Systems	OOImpl	Object-Oriented Implementation
IB	Integral Backstepping	PID	Proportional-Integral-Derivative
IDE	Implementation Development Environment	SMC	Sliding-Mode Control
IGCB	Instantaneous Global Continuous Behavior	SysML	System Modeling Language
IMU	Inertial Measurement Unit	UAV	Unmanned Aerial Vehicles
LQ	Linear Quadratic	UML	Unified Modeling Language
LOS	Line-Of-Sight	VTOL	Vertical Take Off and Landing
MARTE	Modeling and Analysis of Real-Time and Embedded Systems	WP	Way-Point
MBSE	Model-Based Systems Engineering		

1 Introduction

Unmanned Aerial Vehicles (UAVs) have seen unprecedented levels of growth over the last decade, especially the quadrotor UAV. Even though UAVs have been mainly used for military applications, there is a considerable and increasing interest for civilian applications. It is postulated that UAVs will be used extensively in the future for environmental monitoring, traffic monitoring, mapping, surveillance, etc. The quadrotor UAV is one of the

miniature UAV types of which operating modes are capable of Vertical Take Off and Landing (VTOL), hovering and horizontal flight; so they can be more easily handled in turbulences such as wind and are easier to design and realize by using a compact airframe [1-3].

The problem of designing autonomous flight controllers for quadrotor UAVs is equally challenging because these controllers are closely connected with the dynamic models. Therefore, these control systems must take account of models with discrete events and continuous behaviors that can be called Hybrid Dynamic Systems (HDS) described in [4-6]. These behaviors are distributed on different operating modes, which are associated with processes related to the interactivity with users such as the designer, supervisor, maintainer etc. Furthermore, controlled systems do not always have the same behavior because they are associated with validity hypotheses to check at any moment; the security requirement forces to envisage events, and behaviors different from nominal behaviors. The behaviors of such systems are thus complex; in this paper, the quadrotor UAV controllers are considered as HDS of which behaviors can be modeled by Hybrid Automata (HA) [7-9].

In addition, the immersion in an industrial control context makes that designers and programmers must take into account costs and existing standards in order to effectively design and deploy the developed systems with reasonable cost. The customization and reusability are also factors to be associated with the production of new applications in order to reduce their costs, resources and time development. According to the Object Management Group (OMG) [10], UML appeared to us to be essential for its visual object-oriented design support, which has been largely spread and appreciated in the software industry. Furthermore, the System Modeling Language (SysML) [11], which is a UML profile for systems engineering, has been standardized by OMG. SysML supports the specification, analysis, design, verification and validation of a broad range of complex systems. However, both UML and SysML lack the constructs for modeling time and duration constraints of the developed system. Thus, the real-time UML/MARTE version [12-14] is chosen to model in detail the analysis and design artifacts for real-time and embedded control systems, e.g. the quadrotor UAV controller. This version also includes the 'capsules, ports, protocols, connectors' concepts that can be adapted by specializing a set of control capsules in precise behaviors and structures of the quadrotor UAV controller.

The paper aims to implement a control model integrating the quadrotor UAV dynamics into MDA combined with the real-time object paradigms and the specialization of HA features, which can permit us to conveniently realize and deploy the quadrotor UAV controller, and also allow the designed and implemented control elements to be closely customizable and re-usable in the realization of new applications for various UAV types of which operating modes are capable of VTOL. In our current model, the quadrotor dynamics and control architecture are also adapted for the control that are then combined with the models as follows: The Object-Oriented (OO) Analysis (OOA), OO Design (OOD) and OO Implementation (OOImpl) models; this control system permits a quadrotor UAV to track a reference trajectory in the *Cartesian* space. Here, the OOA includes the use-case model specialized closely with an implementable function block diagram, the EKF algorithm, HA and its evolution hypotheses of realization to precisely capture the requirement analysis for a quadrotor UAV controller; the OOD model is built on the identified OOA model by specifying the real-time UML/MARTE to entirely design the real-time control capsules with their timing concurrency of evolutions in detail. The detailed OOD elements is then converted into OOImpl models by using open-source platforms such as *OpenModelica* [15] and *Arduino* [16] in order to quickly simulate, realize and deploy the quadrotor UAV controller. Finally, a trajectory-tracking controller of an application of autonomous quadrotor UAV was completely deployed and tested.

The paper is structured as follows: The second section brings the related works that have inspired us to define a model-based design for quadrotor UAV controllers. The quadrotor UAV dynamic model and control architecture are introduced in the third section. The fourth section presents the details of model-driven development to intensively realize quadrotor UAV controllers, including the OOA, OOD and OOImpl components. Following this described model, in the fifth section, it is applied to a case study. Conclusions and future works are reported in the final section.

2 Related work

The autonomy architecture of general UAV basically consists of three main sub-systems as follows: the guidance sub-system, navigation sub-system and control system. All three of these sub-systems have their own individual tasks to complete, yet must also work cooperatively in order to reliably

allow a UAV to complete its objectives. In present design and construction of quadrotor UAV controllers, there were many applications that have used the basic control methods combined with soft computing approaches [17-22] to make them more effective for control systems. For example, a model-based adaptive controller for quadrotor UAV with different payload has introduced by Tsay [23]; it is a gain stabilized control technique in which the large gain is used for large tracking error to get fast response, the small gain is used between large and small tracking error for good performance. The large gain is also used for small tracking error to cope with disturbance in this developed model. A Binary Observers (BO)-based linear feedback Sliding Mode Control (SMC) for quadrotor's robust flight has been presented by Zhang, Zhou, Li, et al. [24]; in this study, the quadrotor has been divided into two subsystems, i.e. the rotational subsystem sensitive to unknown disturbances and the translational one subjected to both disturbances and measurement delay. The disturbance observer and delayed output observer were then designed with their interactions being considered, and disturbance observer based controller and BO-based controller were hierarchically proposed for attitude and position control respectively. The disturbances and real outputs were converged exponentially by observers' estimations, which were compensated in the feedback control loop. A robust controller has been proposed by Liu, Li, Zuo, et al. [25] to address the attitude control problem for quadrotors with uncertainty in the input delays; their designed controller included a nominal controller to achieve desired tracking for the nominal system and a robust compensator to achieve the robust stability of the uncertain system with input delays. In particular, the different controllers based on *Lyapunov* theory, PID, Linear Quadratic (LQ), Backstepping and SMC techniques were implemented to the control design of a miniature quadrotor UAV, and were compared for attitude control that could be found in [26].

However, the above guidance and control models were based on the structural implementations. Thus, the designed control elements could be difficult to customize and reuse for realizing controllers of different UAV types of which operating modes are capable of VTOL, and for deploying appropriately into various software and hardware platforms. To achieve this goal, the Model-Based Systems Engineering (MBSE) approach can be specialized in order to intensively performing the whole of development lifecycle for quadrotor UAV controllers. The MBSE is the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases [27]. MBSE is intended to facilitate systems engineering activities that have traditionally been performed using the document-based approach and result in the enhanced specification and design quality, reuse of system specification and design artefacts, and communications among the development team. Following the MBSE approach, Barbieri, Fantuzzi and Borsari [28] have introduced a model-based design methodology, which supports the integrated development of complex mechatronic devices and the problem of devices interchangeability; this method was based on the W model [29] and on the SysML identification: the generation of a system model constantly refined and updated, during the design process, and whose information can be exported to or imported from domain specific models. A practical approach to the application of advanced software engineering methodologies for the design and realization of automatic machineries for smart manufacturing processes have been presented in [30]; the patterns presented in this work have driven control software programmers in the development stage of real-world systems, in order to cope with the increasing system complexity, preserving efficiency and reusability of mechatronic components. Herrera, Posadas, Peñil, et al. [31] have introduced the COMPLEX UML/MARTE methodology for Design Space Exploration of embedded systems, which was based on a novel combination of Model-Driven Engineering (MDE), Electronic System Level and design exploration technologies; this framework could enable capturing the set of possible design solutions, that is, the design space, in an abstract, standard and graphical way by relying on UML and the standard MARTE profile. In particular, the statistical findings from a complete survey about the use of real-time UML/MARTE and model-driven approaches for the design of real-time embedded systems can be seen in [32, 33].

In this section, we illustrated the structural implementations for quadrotor UAV controllers and the model-driven methodologies for real-time and embedded systems that could be used to make up a model-driven implementation to conveniently develop the quadrotor UAV controller.

3 Quadrotor UAV dynamics and control structure

3.1 Overview of quadrotor UAV dynamics for control

From the large field of guidance, navigation and control of aerial vehicles in [18, 26, 34-39], the six DoF dynamic model of a quadrotor UAV in body coordinate frame can be written in equation systems (1).

$$\begin{cases} m\ddot{x} = (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)\sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{xi} - \frac{1}{2} C_x A_c \rho \dot{x}|\dot{x}| \\ m\ddot{y} = (-\cos\psi\cos\phi + \sin\psi\sin\theta\cos\phi)\sum_{i=1}^{4} T_i - \sum_{i=1}^{4} H_{yi} - \frac{1}{2} C_y A_c \rho \dot{y}|\dot{y}| \\ m\ddot{z} = mg - (\cos\psi\cos\phi)\sum_{i=1}^{4} T_i \\ I_{xx}\ddot{\phi} = \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) + J_r\dot{\theta}\Omega_r + l(-T_2 + T_4) - h\left(\sum_{i=1}^{4} H_{yi}\right) + (-1)^{i+1}\sum_{i=1}^{4} R_{mxi} \\ I_{yy}\ddot{\theta} = \dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) - J_r\dot{\phi}\Omega_r + l(T_1 - T_3) - h\left(\sum_{i=1}^{4} H_{xi}\right) + (-1)^{i+1}\sum_{i=1}^{4} R_{myi} \\ I_{zz}\ddot{\psi} = \dot{\theta}\dot{\phi}(I_{xx} - I_{yy}) + J_r\dot{\Omega}_r + (-1)^{i}\sum_{i=1}^{4} Q_i + l(H_{x2} - H_{x4}) + l(H_{y3} - H_{y1}) \end{cases} \quad (1)$$

Where: I_{xx}, I_{yy} and I_{zz} are inertia moments; ϕ, θ, ψ are respectively Roll, Pitch, Yaw (RPY) angles; J_r presents the rotor inertia; H is a set of hub forces; R_m is a set of rolling moments; T_i presents the thrust force ($i = \overline{1,4}$); Ω_r is the overall residual propeller angular speed; A_c is fuselage area; C is the propulsion group cost factor; ρ is the air density; Q_i presents the drag moment; h and l are respectively vertical distance and horizontal distance: propeller center to Center Of Gravity (CoG); x, y and z define the position in body coordinate frame.

To develop controllers of quadrotor UAVs, it is advisable to simplify the model in order to comply with the real-time constraints of the embedded control loop. In our case study, we proposed that hub forces and rolling moments are neglected and thrust and drag coefficients are supposed constant. The system can be rewritten in state-space form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ with $\mathbf{u}$ inputs vector and $\mathbf{x}$ state vector.

$$\mathbf{u} = [u_1, u_2, u_3, u_4]^T \quad (2)$$

Here, u_i is the control input ($i = \overline{1,4}$) and described by a set of equations (3); b and d are respectively the thrust and drag factors.

$$\begin{cases} u_1 = b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ u_2 = b(-\omega_2^2 + \omega_4^2) \\ u_3 = b(-\omega_1^2 + \omega_3^2) \\ u_4 = d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{cases} \quad (3)$$

$\mathbf{x}$ is a 12-dimensional state vector for describing the motion of quadrotor UAV that is written as follows:

$$\mathbf{x} = \left[x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}\right]^T \quad (4)$$

A discrete state-space representation is required for modeling the quadrotor UAV controller in order to use a recursive estimation filter of motion states, e.g. the EKF; the developed system can be then described by a set of equations (5).

$$\begin{cases} \mathbf{x_k} = \mathbf{f_{k-1}}(\mathbf{x_{k-1}}, \mathbf{u_{k-1}}) + \mathbf{w_{k-1}} \\ \mathbf{y_k} = \mathbf{h_k}(\mathbf{x_k}) + \mathbf{v_k} \end{cases} \quad (5)$$

Where: $\mathbf{x_k}$ is the vector of state variables at the k^{th} instant of $\mathbf{x}$; $\mathbf{u_k}$ and $\mathbf{y_k}$ are respectively the inputs and outputs of the system; $\mathbf{w_k}$ and $\mathbf{v_k}$ are the additive process and measurement noise; the first equation in (5) is called the system's evolution equation, while the second one is called the measurement equation.

3.2 Using control structure and technique

Within the autonomy architecture of quadrotor UAVs are three main systems. These are the guidance system, which is responsible for generating the trajectory for the vehicle to follow; the navigation system, which produces an estimation of the current state of the vehicle; and the control system, which calculates and applies the appropriate forces to manoeuvre the vehicle. Fig. 1 shows out a functional block diagram, which captures how these sub-systems interact.

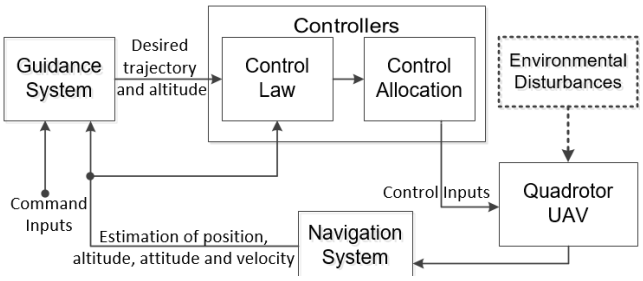

Fig. 1. Block diagram of guidance, navigation and control for quadrotor UAVs.

Here, the guidance system is responsible for producing the desired trajectory for the vehicle to follow. This task can be completed by implementing a common guidance approach based on the generations of way-points that take the desired way-points defined pre-mission and, with the possible inclusion of external environmental disturbances, generates a path for the quadrotor UAV to follow in order to reach each successive way-point. An alternative approach is based on Line-Of-Sight (LOS) guidance [40-42]; in this case, the heading control is computed by considering as input the angle formed by the vector from the quadrotor UAV to the next way-point rather than requiring the

quadrotor UAV to exactly follow the line segment between the current and the following way-point. The navigation system addresses the task of determining the current state of the quadrotor UAV; for airborne vehicles, Global Positioning System (GPS) is readily available and is often used to provide continuous accurate positioning information to the navigation system. Overall, the task of the navigation system block is to provide the best estimate of the current state of the quadrotor UAV, regardless of what sensor information is available. The controllers are responsible for providing the corrective signals and events to enable the quadrotor UAV to follow the desired trajectory. This is achieved by receiving the desired state of the quadrotor UAV from the guidance system, and its current state combining the altitude, position, attitude and velocity from the navigation system. It then calculates and applies correcting forces and moments, through use of the various actuators on the quadrotor UAV, to minimize the difference between desired and current states. This allows the quadrotor UAV to track a desired trajectory even in the presence of unknown disturbances. Overall, the controllers of any quadrotor UAV composes of a sub-block of control law and a sub-block of control allocation. The first is responsible for generating the generalized forces and moments in six DoF based on current and desired states, while the second is responsible for distributing this generalized forces and moments amongst the actuators of quadrotor UAV in order to realize the navigation task allocation.

From the above described quadrotor UAV dynamic model and general control structure together with characteristics of HDS in [4, 5, 43], we find that controllers of quadrotor UAVs are HDS whose dynamic behaviors can be modeled by HA. These control systems have the *continuous*/*discrete* parts and their interactions such as the motional components, e.g. *horizontal transferring*, *VTOL*, *rotation*, *roll*, *pitch* and *yaw*, and external interacting events from the guidance and navigation system, and environmental disturbances. In our model, we are interested in developing the trajectory-tracking controller of quadrotor UAVs, so we can use this hybrid dynamic model to find out the control algorithms with a specific guidance law such as the Line-Of-Sight (LOS) guidance implemented in [40, 41, 44].

Furthermore, Bouabdallah [26] has made the theoretical simulation and experiment evaluations for a quadrotor UAV controller with several control techniques using different approaches such as: *Lyapunov* theory, Proportional Integral Derivative (PID) control, optimal control theory, Backstepping and SMC; it clearly brought out that the way is to follow a combination between PID and Backstepping into the so-called Integral Backstepping (IB) [45]. The goal of this approach was to bring together the robustness against disturbances offered by Backstepping and robustness against model uncertainties offered by the integral action. The stability analysis is performed by using *Lyapunov* theory; it means that IB is a recursive design technique using Control *Lyapunov* Functions (CLF). The CLF concept is a generalization of *Lyapunov* design results by, for instance, [46-48]. Hence, this can permit us to design the control law of quadrotor UAV controllers combined with the IB technique, the specialization of HA features and CLF for more complex flight maneuvers than a simple hovering.

4 Model-driven development for a quadrotor UAV controller

As the previously stated, the real-time UML/MARTE version was chosen to model in detail the analysis and design artifacts for real-time and embedded control systems, e.g. the quadrotor UAV controller. Starting from the above adapted quadrotor UAV dynamics and control structures and real-time UML/MARTE features, we develop in detail a model-driven implementation for the quadrotor UAV controller, which includes three models as follows: OOA, OOD and OOImpl model so separate the specification of the operation of the system from the details of the way that system uses the capabilities of its platform. There are also transformation rules, which allow the OOD model be converted into various OOImpl models.

i) In OOA model, the use case model in object collaboration is specialized by the implemented functional block diagram, HA and its realization hypotheses to closely capture the requirements analysis for a quadrotor UAV controller.

ii) OOD model is built up by specifying the real-time control capsules, ports, protocols and their timing concurrency of evolutions together with the realization hypotheses of HA in order to model the precise behaviors and structures of quadrotor UAV controller.

iii) OOD model is then converted into OOImpl model as follows: Object-Oriented simulation models are carried out in the OOImpl model in order to simulate the defined design model into the specific software platforms that permits us to theoretically evaluate the control performance and

functionalities, and to easily identify control design elements of this system before we decide to realize and deploy it. The OO realization models are also performed in the OOImpl model in order to carry out its implementation phase with specific platforms, which can support object-oriented programming languages such as C++, Java, Ada, etc., and upload the implemented control program to compatible microcontrollers to completely deploy the quadrotor UAV controller.

4.1 OOA model for a quadrotor UAV controller

From the above dynamic model (1), general control structure (Fig. 1) of quadrotor UAVs together with LOS guidance [41], we have developed the main use case model of controllers as shown in Fig. 2 combined with an example of trajectory-tracking scenarios and local state machine of the "*Track a desired trajectory*" use case which are respectively shown in Fig. 3a and Fig. 3b. Here, MDS is the *Measurement and Display System* combined with the guidance and navigation system; AES is the *Air Environment System* including disturbances generated by the weather. Here, it is necessary to provide industrial control constraints, e.g. the maximum tilted angle, velocity, altitude and other safe flying modes of the developed quadrotor UAV in order to ensure the operational safety of this system.

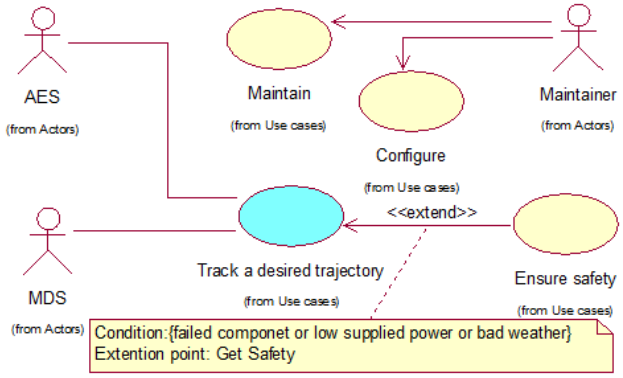

Fig. 2. Main use case model for a quadrotor UAV controller.

Actually, the real-time UML/MARTE version lacks the constructs for modeling internal continuous behaviors for each state on the state machine diagram, an implemented functional block diagram must be then defined in order to model continuous behaviors of the quadrotor UAV controller with events issued from outside. Starting from the considered dynamic model of quadrotor UAV, the general architecture and the identified use case model with LOS guidance, we propose an implemented functional diagram for the quadrotor UAV controller as shown in Fig. 4.

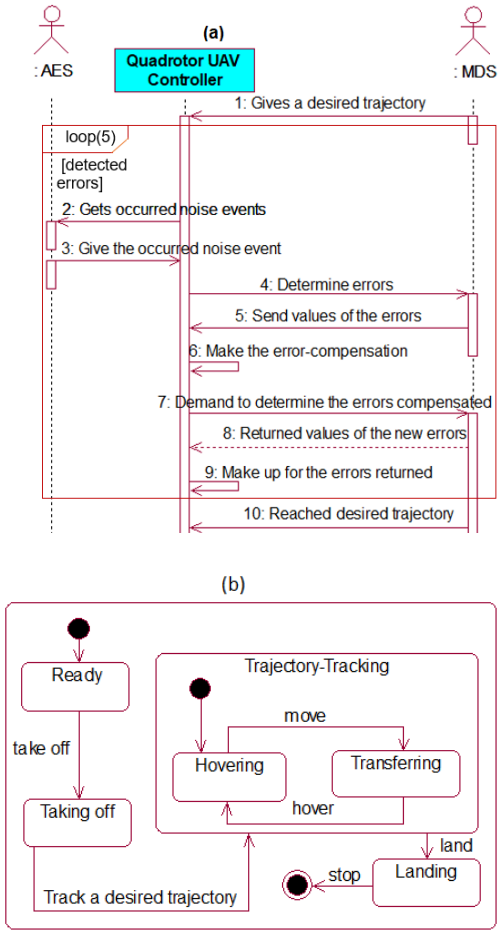

Fig. 3. An example of trajectory-tracking scenarios (**a**) and the local state machine of the "*Track a desired trajectory*" use case (**b**) for a quadrotor UAV controller.

Here, *Desired trajectory* and *Take-off/Landing* actions respectively give the desired position (x_d, y_d) altitude (z_d) to the position and altitude controller; ΣT_d is the desired overall thrust; the position controller receives the quadrotor UAV's position (x, y) and desired thrust, it outputs desired roll (ϕ_d) and pitch (θ_d) while desired yaw (Ψ_d) comes directly from the guidance system; the attitude controller gives then the desired motor speeds (ω_{d1}, ω_{d2}, ω_{d3}, ω_{d4}) to the motor controllers; $\tau_{\phi,\theta,\Psi}$ and ΣT are respectively the overall torque and thrust acting on the quadrotor UAV. In our current model, the IB technique combined with the EKF algorithm are used for altitude, position, attitude and velocity control. The IB expansions combined with CLF for quadrotor UAV controllers were well known by

many quadrotor UAV control applications, for instance, [26, 46, 49-51]. The state-space models (2), (4) and (5) are used for the estimation and prediction of the altitude, position, attitude and velocity corresponding to the sensors installed on the quadrotor UAV that are implemented in the *Navigation System* block. The standard navigation filter is based on EKF that composes of the *predict/update* scheme shown in Algorithm 1 for estimating the altitude, position, attitude and velocity of the quadrotor UAV. In Algorithm 1, $\hat{\cdot}$ denotes an estimate; P is the state covariance; Q and R are respectively the covariance matrices of process and measurement noise, assumed as zero mean stationary white noises with zero cross-correlation; the state is recursively estimated starting from the assumed initial conditions as follows: $\hat{\mathbf{x}}_{0|0} = \mathbf{x}_0$ and $P_{0|0} = \mathbf{0}_{12\times12}$. When the EKF does not use directly the measures of the accelerometer, as it was deemed too noisy for the position estimation; in the case in which data from one or more sensors are not available for a period, the algorithm then uses the last available measure, but it gives a high value to the corresponding coefficient; the measure will be thus characterized by a low weight in the estimation process.

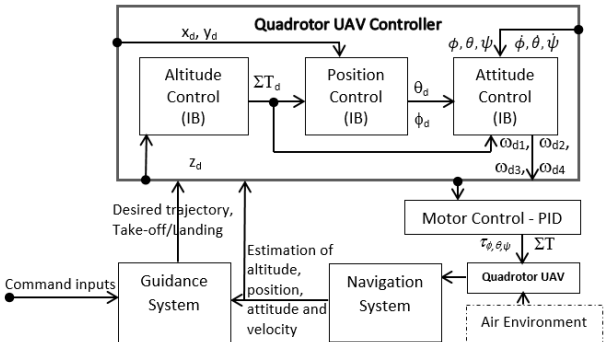

Fig. 4. An implemented functional block diagram for the quadrotor UAV controller.

Algorithm 1. Standard navigation filter based on EKF for a quadrotor UAV controller.

Function *EKF algorithm*

Step *EKF predict*

Data : $\hat{\mathbf{x}}_{k-1|k-1}, P_{k-1|k-1}, \mathbf{f}_{k-1}(.)$

Result : $\hat{\mathbf{x}}_{k|k-1}, P_{k|k-1}$

$F_{k-1} = \frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_{k-1|k-1}\mathbf{u}_{k-1}}$;

$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1|k-1})$;

$P_{k|k-1} = F_{k-1}, P_{k-1|k-1}F_{k-1}^T + Q_{k-1}$;

end

Step *EKF update*

Data : $\hat{\mathbf{x}}_{k|k-1}, P_{k|k-1}, \mathbf{h}_k(.)$

Result : $\hat{\mathbf{x}}_{k|k}, P_{k|k}$

$H_k = \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_{k|k-1}}$;

$S_k = R_k + H_k P_{k|k-1} H_k^T$;

$L_k = P_{k|k-1} H_k^T S_k^{-1}$;

$\mathbf{e}_k = \mathbf{y}_k - \mathbf{h}_k(\hat{\mathbf{x}}_{k|k-1})$;

$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + L_k \mathbf{e}_k$;

$P_{k|k} = P_{k|k-1} - L_k S_k L_{k-1}^T$;

End

In OOA model, HA are specialized to describe mathematical behaviors, i.e. the dynamic model of quadrotor UAV including terms as *Situations*, *Continuous State Variables*, *Event*, *Transition*, *Global Continuous Behavior* and *Invariants*. A HA of the quadrotor UAV controller is defined in equation (6).

$$\boldsymbol{H_{Quadrotor\ UAV} = (Q, X, \sum, A, Inv, F, q_o, x_o)} \quad (6)$$

Where:

- $\boldsymbol{Q}$ is a set of states describing flying modes of $\boldsymbol{H_{Quadrotor\ UAV}}$, e.g. the motion in *horizontal translations*, *hovering*, *VTOL*, *rotations*, *roll*, *pitch*, and *yaw*, which are combined with the local state machine oriented towards control modes (Fig. 3b) in permutations. $\boldsymbol{Q}$ can be called situations of the quadrotor UAV controller; $\boldsymbol{q_o}$ is the initial situation.
- $\boldsymbol{X}$ presents the continuous state space of $\boldsymbol{H_{Quadrotor\ UAV}}$, $\boldsymbol{X \subset \Re^n}$, $\boldsymbol{x_{co}}$ is the initial value of this space, e.g. continuous components of the quadrotor UAV controller; $\boldsymbol{x_c \in X}$ presents variables of which values are average in the occurred noises.
- $\boldsymbol{\sum}$ is a finite set of events, e.g. external interacting events from the guidance and navigation system, and environmental disturbances.
- $\boldsymbol{A}$ is a set of transitions defined by **(*q, Guard, σ, Jump, q'*)** and represented by an arc between situations, here: $\boldsymbol{q \in Q}$, $\boldsymbol{q' \in Q}$; ***Guard*** is a subset of the state space in which the continuous state must be, so that the transition can be crossed; ***Jump*** represents the continuous state transformation during the change of situation; it is expressed by a state value function, whose result is affected like initial value of the continuous state in the new situation; $\boldsymbol{\sigma \in \sum}$ presents the event being associated in the transition; this association does not imply to give an input or output direction to the event.
- ***Inv*** is an application, which associates a subset of the state space in each situation; it is called the

invariant of the situation, in which the continuous state must remain, when the situation is q, the continuous state must verify $x_c \in Inv(q)$.

- F is defined by using the six DoF dynamic model of quadrotor UAV, and the implemented functional block diagram (Fig. 4); the evolution of continuous state is occurred when the situation is activated.

The constraints as follows: $\sigma \in \Sigma$ is considered in term of inputs/outputs and internality/externality; X contains input/output signals were applied to globally perform the HA evolution of a quadrotor UAV controller. In addition, the realization hypotheses of the HA evolution, which permit the invariant Inv and guard control $Guard$ combined with transitions can generate internal events for the quadrotor UAV controller, are assumed in Table 1.

Table 1. Realization hypotheses of the HA evolution for a quadrotor UAV controller.

Inv and ***Guard***	System evolution
If $x_q \notin Inv(q)$ and $Guard(a)=True$, $a \in A$,	then there is a generated internal event, and the system changes to the situation q' described in a set of situations of the quadrotor UAV, with $Jump_{q'}$, identified by the initial value of the continuous fluid $F_{q'}$. This evolution is realized by the local state machines, e.g. the local state machine of the "*Track a desired trajectory*" use case (Fig. 3b).
If $x_q \in Inv(q)$ and $Guard(a)=True$, $a \in A$,	then the system remains its actual situation q.
If $x_q \in Inv(q)$ and $Guard(a)=False$, $a \in A$,	then the system remains its actual situation q.
If $x_q \notin Inv(q)$ and $Guard(a)=False$, $a \in A$,	then there is a generated internal event; the system changes into the situation q'', which is called the irreversible default situation.

4.2 OOD model for a quadrotor UAV controller

From the authors' approach [52, 53], we have specialized the 5 main control capsules, which take part in the HA realization of the quadrotor UAV being developed: the continuous part's capsule, discrete part's capsule, internal interface's capsule, external interface's capsule and Instantaneous Global Continuous Behavior (IGCB's capsule). Fig. 5 and Fig. 6 indicate respectively the real-time communication pattern, its structure and behaviors of main control capsules by using the real-time UML/MARTE's collaboration and sequence diagrams.

- *The discrete part's capsule* contains a set of situations Q and transitions A in HA of the quadrotor UAV controller (i.e. the *macro*-motion in *horizontal translations*, *hovering*, *VTOL*, and *rotations: roll*, *pitch*, and *yaw*, which are combined with the local state machine (Fig. 3b) in permutations.

- *The continuous part's capsule* is combined with the continuous state space X in the implemented functional block diagram. The sequential evolution of continuous elements is carried out by specifying the *Rendezvous* pattern presented by [54] with two sub-capsules called *RendezVous* and *Semaphore*.

- *The IGCB's capsule* contains concrete global continuous behaviors of the quadrotor UAV being developed at time given just as $f \in F$ in its HA. f is derived from (1), (5) and the implemented functional block diagram (Fig. 4). This is also taken part into Algorithm 1 to estimate the position, depth, attitude and velocity of the quadrotor UAV. Each global continuous behavior corresponds to a situation in this HA.

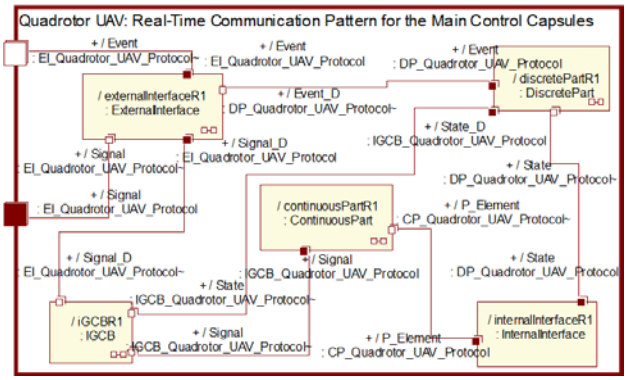

Fig. 5. Real-time communication pattern for the main control capsules for a quadrotor UAV controller.

- *The internal interface's capsule* verifies the Inv in HA of the quadrotor UAV controller, and generates internal events; so that the discrete part's capsule can make its own evolution by these events.

- *The external interface's capsule* is an intermediary, which receives or sends episodic events and periodic signals between the developed quadrotor UAV and their interacted systems such as AES and MDS in our model.

In Fig. 6, the messages exchanging between the main

control capsules are synchronous, and the interval between two adjacent *timeout* messages indicates the sampling period (ΔT) of the IGCB's capsule. The external interface's capsule receives period signals coming from external continuous components. It then gives the *ContinuousElement* message to the IGCB's capsule so that the IGCB's capsule can call all of the continuous elements corresponding to the concrete '*IGCB: IGCB$_1$*'. During the call of the IGCB's capsule, the external interface's capsule can receive an event named '*3: InputEvent*' issued from the MDS or AES, and gives this event named '*ee1: DetectedEvent*' to the discrete part's capsule. The discrete part's capsule then memorizes and later processes this event. If the IGCB's capsule receives the *LastContinuousElement* message coming from the continuous part's capsule, then it gives the *ContinuousEvolution* message to the continuous part's capsule so that the internal interface's capsule can receive all updated variables. The internal interface's capsule then verifies the invariant (***Inv***) of the situation q_2; in this case, there is a generated internal event. The internal interface's capsule gives this event to the discrete part's capsule that permits the IGCB's capsule to identify the concrete '*IGCB: IGCB$_2$*' and give output signals to the external interface's capsule. At the end of this sampling period, the external interface's capsule gives the output event and control signals to the external environment of the quadrotor UAV operating with its concrete '*IGCB: IGCB$_2$*'.

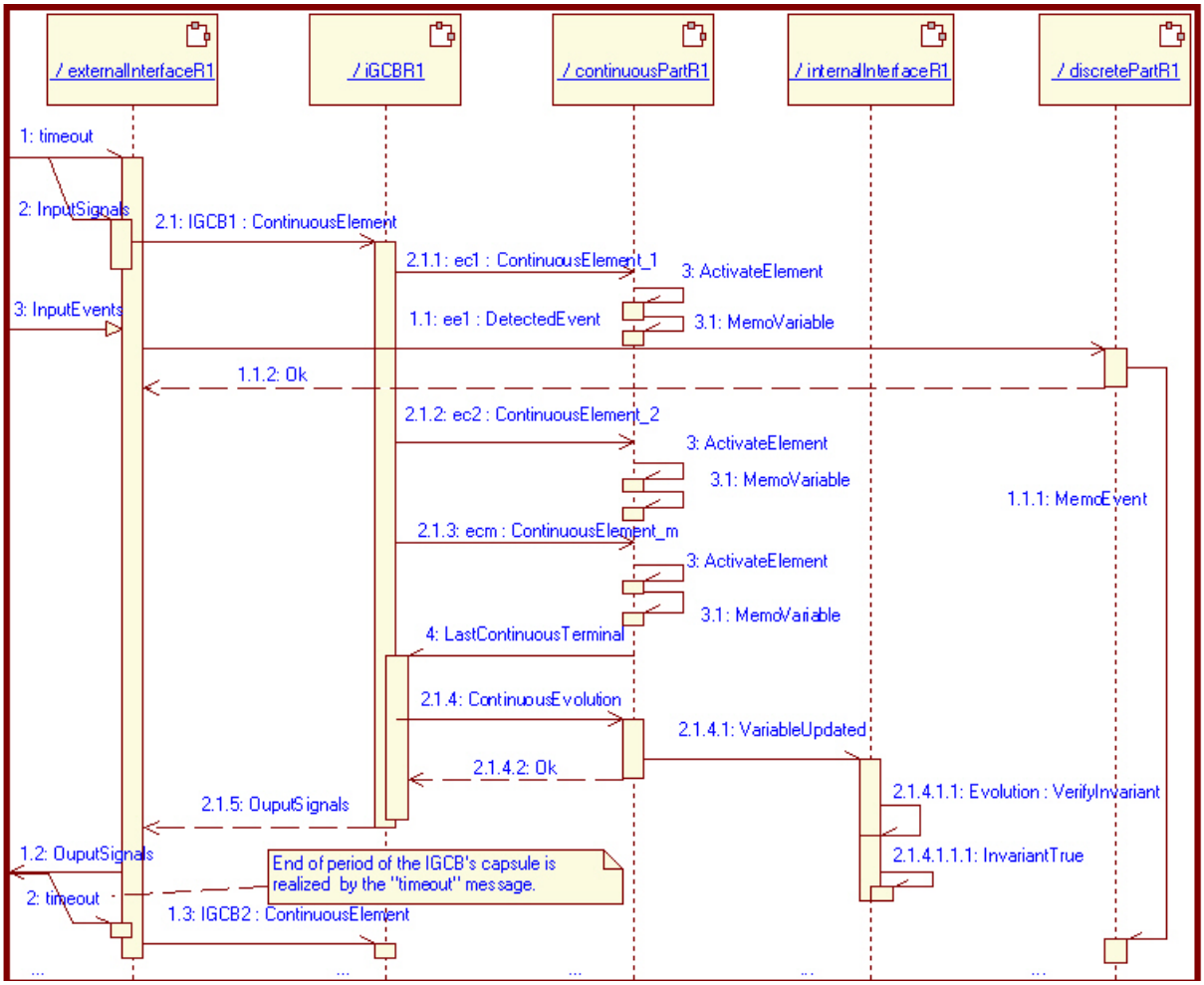

Fig. 6. Sequential comunication messages in one sampling period (ΔT) of main control capsules for a quadrotor UAV controller.

Fig. 7 describes in detail the timing concurrency of evolutions for the above real-time communication pattern of main control capsules. Here, Ee_1, Ee_2, Ee_3… are the external events coming from the external interface's capsule; Ei_1, Ei_2, Ei_3… are internal events issued by the evolution of the internal interface's capsule; q_1, q_2, q_3… indicate the concrete situations in HA of the quadrotor UAV controller being developed; ec_1, ec_2, ec_3… represent the evolutions of continuous elements in the

continuous part's capsule; and ΔT is a sampling period of the IGCB's capsule. The realization hypotheses of timing concurrency for capsule evolutions in Fig. 7 are applied as follows:
- If the end of the discrete part's evolution is located before or just at the sampling date of the IGCB's capsule, then the current IGCB model will change to the new IGCB model corresponding to this evolution;
- If the end of the discrete part's evolution is located after of appearing sampling date (ΔT), then the current IGCB model is not commutated;
- If an event appears during the evolution of the local state machine of UAV application (e.g. Fig. 3b), then this event is immediately memorized and solved later on;
- All of the external and internal events have the same process by the discrete part's capsule;
- During the sampling period of the IGCB's capsule, the continue part's capsule, internal interface's capsule and discrete part's capsule make their own evolutions to possibly commutate to a new IGCB model, the IGCB continuous model remains in its current mode for this period;

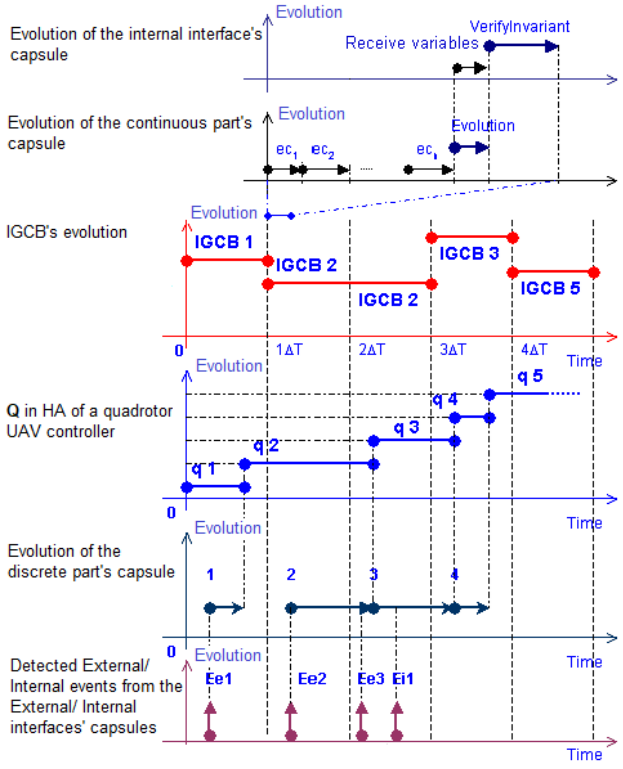

Fig. 7. Timing concurrency of evolutions for main control capsules of a quadrotor UAV controller.

- So during the period of the IGCB's capsule, the current IGCB model can detect two or more appeared situations (i.e. ***Q*** in HA), then the IGCB's capsule synchronizes all these situations just at the end of this period with the null timing duration; the current IGCB model subsequently changes to a new IGCB model, which corresponds to the last appeared situation during this period.

In addition, the reusability is very important to implement controllers for different applications of UAVs typed VTOL because it permits the development time and cost to be reduced. The various reusable views in the development phase are considered as follows: The reusable view is based on the virtual mechanism of objects, classes or class hierarchies; the other re-use view can be based on design components, e.g. the implemented functional block diagram, the local state machine of quadrotor UAV controller and generic state machine of main control capsules that can be specified to develop various control applications of quadrotor UAVs using the same technique. The specializations, which permit the capsule collaboration of a developed quadrotor UAV to can be customizable and reusable in the new control application for various UAVs typed VTOL, can be seen in the author's thesis [55].

4.3 OOImpl models for a quadrotor UAV controllers

To carry out the quadrotor UAV controller, the OOD model is firstly implemented to the OOImpl model (i.e. the simulation model) that is transformed from the above built OOD by using tools such as *IBM Rational Rose RealTime* or *IBM Rational Software Architect RealTime* [56] and *Modelica/OpenModelica* [15, 57]. Here, *IBM Rational*'s leading role in defining the real-time UML is widely acknowledged, as is the pre-eminence of the *IBM Rational Rose RealTime* or *IBM Rational Software Architect RealTime* products in implementing UML to support the architecting of large-scale real-time and embedded software systems. It combines a rich modeling environment with a code-oriented tool set to create a comprehensive practitioner desktop for creating solutions in a variety of architectural styles, and targeted at specific runtime infrastructures. Many other important lifecycle artifacts also benefit from this tool (e.g. requirements lists, test cases and build scripts) to entirely cover development phases for the quadrotor UAV controller. Besides, *OpenModelica* is an open source modelling and simulation environment intended for industrial and academic usage. It is an object- oriented declarative multi-domain modelling language for complex dynamic systems. The *OpenModelica* environment allows most of the expression, algorithm, and function

parts of *Modelica* to be executed interactively, as well as equation models and *Modelica* functions to be compiled into efficient C/C++ codes. The generated C/C++ codes are combined with a library of utility functions, a run-time library and a numerical Differential Algebraic Equation (DAE) solver. The obtained simulation results in *OpenModelica* permits us to theoretically evaluate the control performance and functionalities, and to easily optimize the control design elements before they are implemented and deployed. Then, the OOD model with the optimized control elements of simulation model is adapted to obtain the new updated OOD model for realization models of quadrotor UAV that will be called OOD* model. Finally, this OOD* model is converted into the new OOImpl* model (i.e. the realization model) by using different specific platforms, which are based on the object-oriented Implementation Development Environment (IDE), e.g. the *Arduino*'s IDE [16] in order to completely realize the quadrotor UAV controller with compatible microcontrollers, e.g. *ATMEGA32-U2* and *STM32 Cortex-M4* microcontrollers [16]. A sketch of the above described model transformations is shown as Fig. 8. Here, the transformations are performed through the round-trip engineering (i.e. the forward and reverse engineering) of the intermediate C++ codes including about 80% of the generated codes and 20% of the hand-crafted codes, which are issued from the models depicted in *IBM Rational Rose RealTime* or *IBM Rational Software Architect RealTime*, *OpenModelica* tools and the *Arduino*'s IDE.

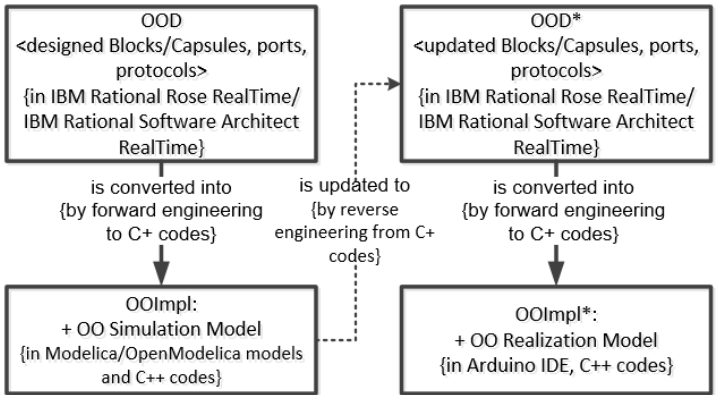

Fig. 8. Sketch of OOD-OOImpl model transformation for quadrotor UAV controllers.

5 Application

Based on the above described model, a trajectory-tracking controller was developed that permits an autonomous quadrotor UAV reach and follow a geometric reference path in the *Cartesian* space starting from a given initial configuration. The main characteristics of this quadrotor UAV are resumed in Table 2.

Table 2. Characteristics of the developed quadrotor UAV.

Parameter	Value
Weight	4.50 kg
Maximum payload	1.55 kg
Autonomy	25 minutes
Power Li-Po battery	22.2 V, 20000 mAh
Maximum Take-off speed	7.5 m/s
Maximum horizontal translation speed	8.5 m/s
Maximum altitude	500 m
Maximum radius of action	4900 m
Inertia moment on x-axis I_{xx}	51.34e-3 kg.m^2
Inertia moment on y-axis I_{yy}	51.34e-3 kg.m^2
Inertia moment on z-axis I_{zz}	11.18e-2 kg.m^2
Rotor inertia of motor J_r	36.24e-5 kg.m^2
Horizontal distance: propeller center to center of gravity l	0.515 m

All of artifacts of the analysis and design models have been created to entirely implement the trajectory-tracking controller of this quadrotor UAV that could be found in the author's thesis [55]. We present here some of the control simulation results supposed as follows: the guidance system addresses a drive event of *Taking-off* to the quadrotor UAV with a desired altitude of 15 m; the transient control response in z-direction (z-Altitude) is shown as Fig. 9a; Fig. 9b brings out the transient control response in x-direction, when the quadrotor UAV received a drive event of *Transferring* with a desired distance of 15 m in x-direction (x-Position) from the current position. All of the obtained simulation results

permit us to theoretically evaluate the control performance of this system within the control criteria such as the admissible timing response, transition and static errors. From that point, we can decide to choose the designed control elements and their parametric values to perform the realization phase of this application.

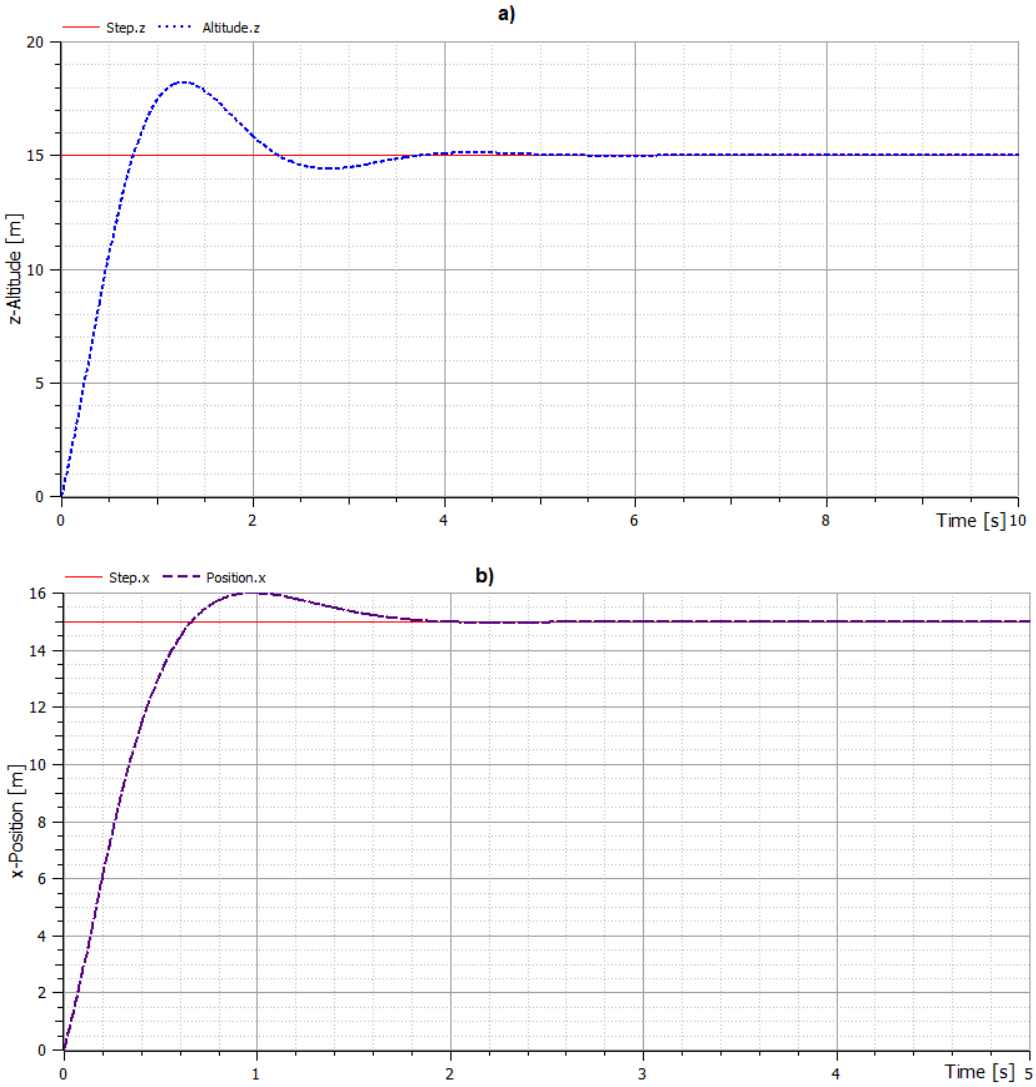

Fig. 9. Transient control responses in z-direction (**a**) and in x-direction (**b**).

The *Arduino* platform [16] was also used to quickly deploy the realization model for the controller. This platform can sense the environment by receiving input from a variety of sensors such as the pressure and magnetometer sensors, Inertial Measurement Unit (IMU), GPS, e.g. *MPU6000* with working frequency 100Hz [58], *Ublox Neo 6M* with working frequency 10Hz [59], etc. and can affect its surroundings by controlled actuators. *ATMEGA32-U2* and *STM32 Cortex-M4* microcontrollers [16] have been used on the board, and can be programmed by using the *Arduino* programming language based on C++ and the object-oriented embedded programming C++.

The trial flights were performed to test the realization model of this application (Fig. 10a); the test scenarios were mainly based on the local state machine (Fig. 3b). We present here some of main test cases and obtained results for controlling the quadrotor UAV as follows:

- 1st test scenario: the quadrotor UAV autonomously reaches and follows a desired rectangle-shaped trajectory with steer angles about 90°, the maximum trajectory error is about 1.8 m against each Way-Point (WP) position at these steer angles (Fig. 10b).
- 2nd test scenario: the quadrotor UAV autonomously reaches and follows a desired quadrangular-shaped trajectory with steer angles about 120°, 30° and 150°, the maximum trajectory error is about 1.9 m against each WP position at these steer angles (Fig. 10c).

- 3rd test case: the quadrotor UAV is autonomously capable of Vertical Take Off and Landing (VTOL) with a maximum altitude of 490 m; the landed position error is 0.5 m against the initial taken off position.

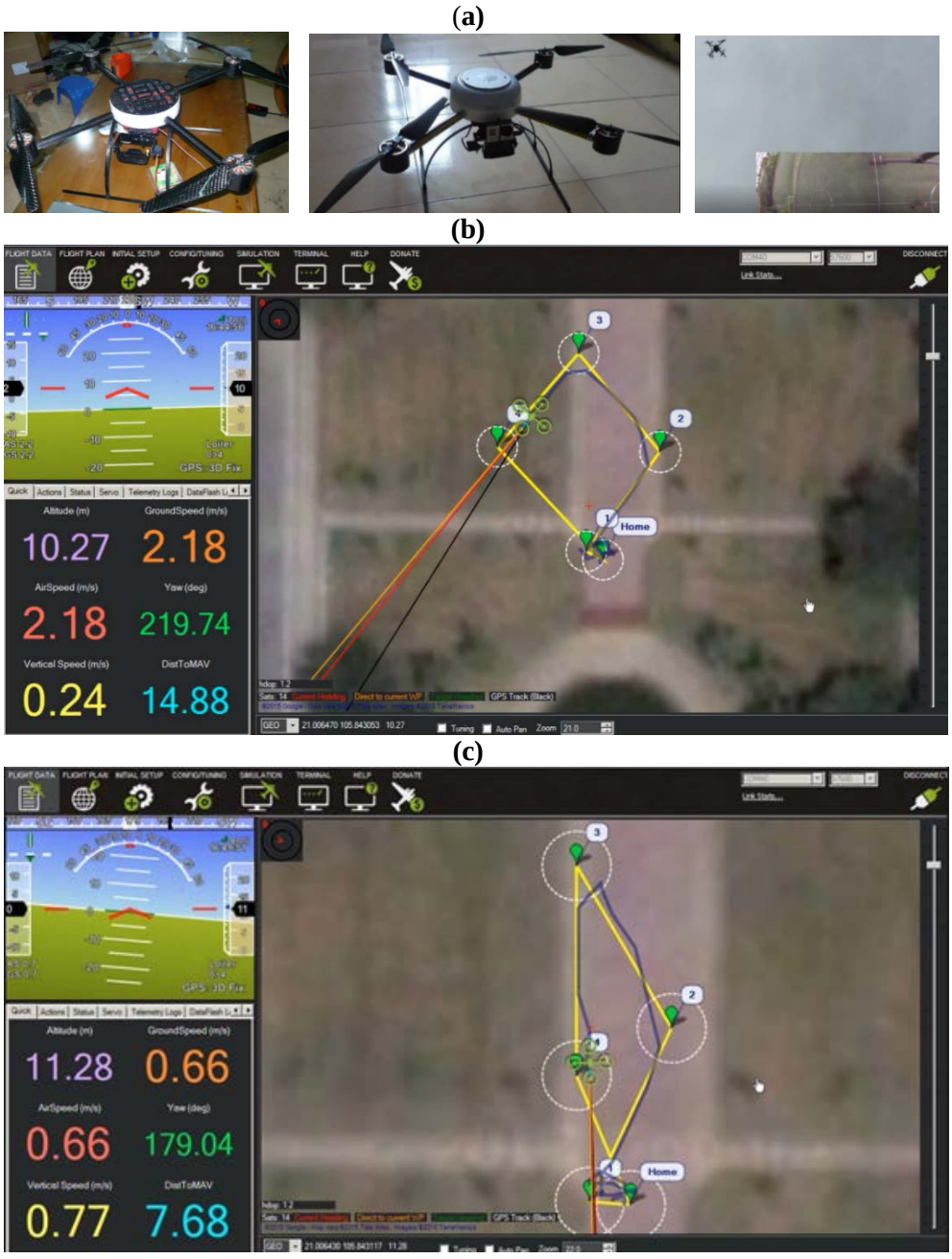

Fig. 10. Setting up and testing the controller for the developed autonomous quadrotor UAV (**a**); Quadrotor UAV reaches and follows a desired rectangle-shaped trajectory with steer angles about 90° (**b**); Quadrotor UAV reaches and follows a desired quadrangular-shaped trajectory with steer angles about 120°, 30° and 150° (**c**).

The trajectory-tracking controller of this autonomous quadrotor UAV was satisfied with performance requirements, e.g. the admissible control duration, transition and static errors that were based on the comparison between all of the experimental data corresponding with the obtained simulation results. The obtained results of control performance were improved in comparison to the alone utilization of standard control methods such as the SMC, PID, LQ, IB or model predictive control. In this application, the standard control method of IB and the EKF algorithm were used for the altitude, position, and attitude control and the PID regulators were applied to the block of *Motor Control* that

combined with HA and its evolution hypotheses of realization introduced in Table 1 to implement the functional block diagram (Fig. 4) for building up the ICCB's capsule of OOD model. Within limit of the paper, we do not present here all the artefacts of the analysis, design and implementation model, and test results of the quadrotor UAV controller, so detailed results can be seen in the author's thesis [55].

6 Conclusions and future works

This paper presented a model-based implementation to intensively realize controllers for quadrotor UAVs whose global behaviors can be considered as the HDS. This model is based on the specializations of MDA's features combined with the real-time UML/MARTE, HA and EKF algorithm to closely analyze, design, implement and realize the control parts of system. No single formalism or language of an engineering process can possible capture all the knowledge and information needed to solve complex control systems such as the quadrotor UAV controller. The main points of this paper have been mentioned as follows:

- The quadrotor UAV dynamics and control structure are adapted for the control that are combined with the specialization of real-time object features including the OOA, OOD and OOImpl components.
- In the OOA model, the use case model is specialized with the implementable functional block diagram, EKF algorithm and HA together with its evolution hypotheses of realization (Table 1) to closely capture the requirements analysis for a quadrotor UAV controller.
- The OOD model is built for obtaining the detailed design model by specifying the real-time control capsules, ports, protocols enclosed with their timing concurrency of evolutions in order to model in detail the behaviors and structures of quadrotor UAV controllers. The developed OOD components are also used to create a real-time communication pattern that can be customized and reused in new UAV applications of which operating modes are capable of VTOL, hovering and horizontal flight.
- The OOD model with the optimized control elements is then adapted to obtain the new updated OOD model for the realization model. This updated OOD model is converted into new OOImpl models by using different object-oriented specific platforms in order to completely realize the quadrotor UAV controller with compatible microcontrollers.
- Based on this model, a trajectory-tracking controller of a low-cost quadrotor was completely deployed and tested out *Arduino ATMEGA U2* and *STM32-Cortex-M4* microcontrollers.

Up to now, we applied the above developed model just in the low-cost quadrotor UAV controller and intend to implement it in the new control applications for autonomous coordinated UAV. Eventually, if we get positive feedbacks, we will investigate in our application strategy to extend it more effective in order to completely make up controllers for balancing search and target response in cooperative team of various UAVs typed VTOL.

Acknowledgements

The authors would like to thank the material supports of the research project coded *T2017-PC-058* at *Hanoi University of Science and Technology, Vietnam.*

References:

[1] N.P. Hung, N.V. Hien, P.G. Diem, N.P. Khanh, V.Q. Huy, H.T.K. Dung, L.X. Truong, D.T. Hung, Research on, design and manufacture a micro-unmanned aerial flying autonomously at desired trajectories, Final report of research project, code: KC03.TN03/11-15, Hanoi University of Science and Technology., Hanoi, Vietnam, 2013.

[2] S.K. Phang, K. Li, B.M. Chen, T.H. Lee, Systematic Design Methodology and Construction of Micro Aerial Quadrotor Vehicles, in: K. P. Valavanis, G.J. Vachtsevanos (Eds.) Handbook of Unmanned Aerial Vehicles, Springer, Dordrecht, Heidelberg, New York, London, 2015, pp. 181-206.

[3] K. Dalamagkidis, Classification of UAVs, in: K. P. Valavanis, G.J. Vachtsevanos (Eds.) Handbook of Unmanned Aerial Vehicles, Springer, Dordrecht, Heidelberg, New York, London, 2015, pp. 83-91.

[4] L.P. Carloni, R. Passerone, A. Pinto, A.L. Sangiovanni-Vincentelli, Languages and Tools for Hybrid Systems Design, now Publishers Inc., Boston, 2006.

[5] P.A. Fishwick (Ed.), Handbook of Dynamic System Modeling, Taylor & Francis Group, USA, 2007.

[6] T. Soriano, A. Sghaier, N.V. Hien, Mechatronics Design from an Object-Oriented Point of View, WSEAS Transactions on Communications, ISSN 1109-2742, 3, 2004, pp. 282-287.

[7] T. A. Henzinger, P. W. Kopke, A. Puri, P. Varaiya, What's Decidable about Hybrid Automata?, Journal of Computer and System Sciences, Elsevier, ISSN 0022-0000, 57, 1998, pp. 94-124.

[8] N.V. Hien, "Une Méthode Industrielle de Conception de Commande par Automate Hybride Développée en Objets", Thèse de Doctorat, Univertsité de Marseille III, France, 2001.

[9] P.G. Diem, P.H. Anh, N.P. Khanh, N.P. Hung, N.V. Hien, A Hybrid Control Model to Develop the Trajectory-Tracking Controller for a Quadrotor UAV, Advanced Materials Research, Trans Tech Publications, ISSN 1022-6680, 1016, 2014, pp. 678-685.

[10] OMG, Documents Associated With Unified Modeling Language™ (UML® Version 2.5), OMG formal/15-03-01, http://www.omg.org/spec/UML/2.5/2015.

[11] OMG, SysML Specifications Version 1.4, OMG formal/2015-06-03, http://www.omg.org/spec/SysML/1.4/2015.

[12] OMG, UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems, OMG Formal Version. http://www.omg.org/spec/MARTE/2011.

[13] B.P. Douglass, Real-Time UML Workshop for Embedded Systems (2nd Edition), Elsevier, Oxford, UK, 2014.

[14] B. Selic, S. Gerard, Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE, Elsevier, USA, 2014.

[15] OpenModelica. *OpenModelica Software, Version 1.12*. Available: https://www.openmodelica.org/, 2018 (accessed July 2018).

[16] Arduino. *Open-source electronics prototyping platform for hardware and software*. Available: http://www.arduino.cc/, 2018 (accessed April 2018).

[17] M. Salichon, K. Tumer, A neuro-evolutionary approach to control surface segmentation for micro aerial vehicles, International Journal of General Systems, Taylor & Francis, ISSN 0308-1079, 4, 2013, pp. 793–805.

[18] K. P. Valavanis, G.J. Vachtsevanos (Eds), Handbook of Unmanned Aerial Vehicles, Springer, Dordrecht, Heidelberg, New York, London, 2015.

[19] M. Saad, A. Ramadhan, Sliding Mode Control of Nonlinear Coupled Tank System, WSEAS Transactions on Systems, Print ISSN: 1109-2777, E-ISSN: 2224-2678, 18, 2019, pp. 35-44.

[20] M. Herrera, M. Sarzosa, I. Paredes, O. Camacho, Optimal Control Based on Fuzzy Estimation of Takagi-Sugeno Model for the Furuta Pendulum: Experimental Results, WSEAS Transactions on Systems, Print ISSN: 1109-2777, E-ISSN: 2224-2678, 18, 2019, pp. 12-24.

[21] T. Emami, Mixed Sensitivity Design of Discrete Time PID Controllers, WSEAS Transactions on Systems, Print ISSN: 1109-2777, E-ISSN: 2224-2678, 17, 2018, pp. 137-145.

[22] Y. Maeda, N. Ishibashi, Control Scheme for SCARA by Recurrent Neural Network Using Simultaneous Perturbation, WSEAS Transactions on Systems, Print ISSN: 1109-2777, E-ISSN: 2224-2678, 17, 2018, pp. 146-155.

[23] T.S. Tsay, Model Based Adaptive Controller for Quadrotor UAV with Different Payload, WSEAS Transactions on Systems, Print ISSN: 1109-2777, E-ISSN: 2224-2678, 17, 2018, pp. 276-283.

[24] J. Zhang, L. Zhou, C. Li, B. Wen, Binary observers based control for quadrotor unmanned aerial vehicle with disturbances and measurement delay, Journal of Aerospace Engineering, Institution of Mechanical Engineers, SAGE Publishing, ISSN 0954-4100, https://doi.org/10.1177/0954410017717286, 2017, pp. 1-14.

[25] H. Liu, D. Li, Z. Zuo, Y. Zhong, Robust attitude control for quadrotors with input time delays, Control Engineering Practice, Elsevier, ISSN 0967-0661, 58, 2017, pp. 142-149.

[26] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying", PhD Thesis, École Polytechnique Fédérale de Lausanne, Suisse, 2007.

[27] INCOSE, Systems Engineering Vision 2025, INCOSE, San Diego, C.A. 92111-2222, USA, 2014.

[28] G. Barbieri, C. Fantuzzi, R. Borsari, A model-based design methodology for the development of mechatronic systems, Mechatronics - The Science of Intelligent Machines, Elsevier, ISSN 0957-4158, 24, 2014, pp. 833–843.

[29] VDI, Design methodology for mechatronic systems VDI 2206, Verein Deutscher Ingenieure, VDI, Düsseldorf, 2004.

[30] M. Bonfè, C. Fantuzzi, C. Secchi, Design patterns for model-based automation software design and implementation, Control Engineering Practice, Elsevier, ISSN 0967-0661, 21, 2013, pp. 1608–1619.

[31] F. Herrera, H. Posadas, P. Peñil, E. Villar, F. Ferrero, R. Valencia, G. Palermo, The COMPLEX methodology for UML/MARTE Modeling and design space exploration of embedded systems, Journal of Systems Architecture - Embedded Software Design, Elsevier, ISSN 1383-7621, 60, 2014, pp. 55–78.

[32] L.T.W. Agner, I.W. Soares, P.C. Stadzisz, J.M. Simão, A Brazilian survey on UML and model-driven practices for embedded software development, Journal of Systems and Software, Elsevier, ISSN 0164-1212, 86, 2014, pp. 997-1005.

[33] M. Rashid, M.W. Anwar, A.M. Khan, Toward the tools selection in model based system engineering for embedded systems—A systematic literature review, Journal of Systems and Software, Elsevier, ISSN 0164-1212, 106, 2015, pp. 150-114.

[34] R.C. Nelson, Flight Stability and Automatic Control (2nd Edition), McGraw Hill, Singapore, 1998.

[35] G.J.J. Ducard, Fault-tolerant Flight Control and Guidance Systems: Practical Methods for Small Unmanned Aerial Vehicles, Springer, London, UK, 2009.

[36] L.R.G. Carrillo, A.E.D. López, R. Lozano, C. Pégard, Quad Rotorcraft Control: Vision-Based Hovering and Navigation, Springer, London, Heidelberg, New York, Dordrecht, 2013.

[37] A. Tewari, Advanced Control of Aircraft, Spacecraft and Rockets, John Wiley & Sons, UK, 2011.

[38] T.I. Fossen, Mathematical Models for Control of Aircraft and Satellites (3rd Edition), Department of Engineering Cybernetics, NTNU, Norway, 2013.

[39] Q. Zhang, X. Wang, X. Xiao, C. Pei, Design of a fault detection and diagnose system for intelligent unmanned aerial vehicle navigation system, Journal of Mechanical Engineering Science, SAGE Publishing, ISSN 0954-4062 https://doi.org/10.1177/0954406218780508, 2018, pp. 1-7.

[40] H. Choset, K.M. Lynch, S. Hutchinson, G.A. Kantor, W. Burgard, L.E. Kavraki, S. Thrun, Principles of Robot Motion: Theory, Algorithms, and Implementation, The MIT Press, Cambridge MA 02142-1209, USA; London, W1W 6AN, UK, 2005.

[41] A.M. Lekkas, T.I. Fossen, Integral LOS Path Following for Curved Paths Based on a Monotone Cubic Hermite Spline Parametrization, IEEE Transactions on Control Systems Technology, ISSN 1063-6536, 22, 2014, pp. 2287 - 2301.

[42] Tsay T.S, Intelligent Guidance and Control Laws for an Autonomous Underwater Vehicle, WSEAS Transactions on Systems, Print ISSN: 1109-2777, E-ISSN: 2224-2678, 9, 2010, pp. 463-475.

[43] N.V. Hien, T. Soriano, Implementing Hybrid Automata for Developing Industrial Control Systems, 8th IEEE-ETFA, IEEE, Antibes - Juan les Pins, France, 2001, pp. 129-137.

[44] J.H. Liu, J.Y. Shan, Q. Liu, Optimal pulsed guidance law with terminal impact angle constraint, Journal of Aerospace Engineering, Institution of Mechanical Engineers, SAGE Publishing, ISSN 0954-4100, 231, 2017, pp. 1993–2005.

[45] M. Krstic, L. Kanellakopoulos, P.V. Kokotovic, Nonlinear and Adaptive Control Design, John Wiley & Sons, USA, 1995.

[46] J.J. Xiong, E.H. Zheng, Position and attitude tracking control for a quadrotor UAV, ISA Transactions®, Elsevier, ISSN 0019-0578, 53, 2014, pp. 725–731.

[47] H.J. Jayakrishnan, Position and Attitude control of a Quadrotor UAV using Super Twisting Sliding Mode, IFAC-PapersOnLine, Elsevier, ISSN 2405-8963, 49, 2016, pp. 284-289.

[48] D. Cabecinhas, R. Cunha, C. Silvestre, A nonlinear quadrotor trajectory tracking controller with disturbance rejection, Control Engineering Practice, Elsevier, ISSN 0967-0661, 26, 2014, pp. 1-10.

[49] M. Bouchoucha, S. Seghour, H. Osmani, M. Bouri, Integral Backstepping for Attitude Tracking of a Quadrotor System, Electronics and Electrical Engineering, ISSN 1392-1215, 116, 2011, pp. 75-80.

[50] Z. Fang, W. Gao, Adaptive integral backstepping control of a Micro-Quadrotor, IEEE International Conference on Intelligent Control and Information Processing (ICICIP),, IEEE, Harbin Institute of Technology Harbin, China, 2011, pp. 910-915.

[51] X. Huo, M. Huo, H.R. Karimi, Attitude Stabilization Control of a Quadrotor UAV by Using Backstepping Approach, Mathematical Problems in Engineering, Hindawi, ISSN 1024-123X, 2014, 2014, pp. 9 pages.

[52] N.V. Hien, N.V. He, P.G. Diem, A model-driven implementation to realize controllers for Autonomous Underwater Vehicles, Applied Ocean Research, Elsevier, ISSN 0141-1187, 78, 2018, pp. 307-319.

[53] T. Soriano, N.V. Hien, K.M. Tuan, T.V. Anh, An object-unified approach to develop controllers for autonomous underwater vehicles, Mechatronics - The Science of Intelligent Machines, Elsevier, ISSN 0957-4158, 35, 2016, pp. 54-70.

[54] B.P. Douglass, Design Patterns for Embedded Systems in C - an Embedded Software Engineering Toolkit (1st edition), Elsevier, Oxford, UK, 2011.

[55] P.G. Diem, "An Object-Oriented Design Method for Controllers of Quadrotor Unmanned Aerial Vehicles (UAV)", Ph.D. Thesis, Hanoi University of Science and Technology, Vietnam, 2017.

[56] IBM. *IBM Rational software and Training kit*. Available: https://ibm.onthehub.com, 2018 (accessed April 2018).

[57] P. Fritzson, Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica, Wiley & Sons, UK, 2011.

[58] InvenSense. *Sensor System on Chip*. Available: http://www.invensense.com/, 2018 (accessed September 2018).

[59] u-blox. *Gobal leader in wireless communications and positioning semiconductors and modules for the industrial, automotive and consumer markets*. Available: https://www.u-blox.com, 2018 (accessed June 2018).