The Design and Implementation of Quadrotor UAV

Petr Gabrlik[1], Vlastimil Kriz[2], Jan Vomocil[1], and Ludek Zalud[2]

[1] Central European Institute of Technology, Brno University of Technology, Czech Republic
{petr.gabrlik,jan.vomocil}@ceitec.vutbr.cz
[2] Faculty of Electrical Engineering and Communication,
Brno University of Technology, Czech Republic
xkrizv00@stud.feec.vutbr.cz, zalud@feec.vutbr.cz

Abstract. The design and implementation of the four-rotor aerial mobile robot called quadrotor is described in this paper. The beginning is focused on the mechanical construction of the robot body and the hardware implementation of the main control board. The next part describes the simplified mathematical model of the quadrotor. On the base of the created model, a state space controller was designed and implemented. As a result, every quadrotor axis is controlled independently as in the case of the use of separated PI or PD controllers. The last part of the paper deals with a software solution. This can be divided into three parts: the first part describes the application for the onboard microcontroller, the second part focuses on the solution of the base station. The special part of software which solves the localization of the quadrotor using a camera is described at the end of the article.

1 Introduction

The quadrotors are a very popular type of unmanned aerial vehicles because of their mechanic simplicity in comparison with other flying robots. The construction shown on Fig. 1 is very simple. It is formed by four beams which are orthogonal to each other. At the end of each beam, there is a BLDC (Brushless DC) engine with a propeller. Common helicopters have very similar flight characteristics as quadrotors, but quadrotors have different methods of flight control.

The paper deals with the modelling and realization of quadrotor. This robot is used for many purposes. It can be used in civil or military applications. This type of robot allows for very fast takeoff and its operation is very low-cost. It can carry many devices, e.g. chemical sensors for inspection of air pollution during fire.

This work is divided into several parts. First of all, necessary electronics are described. This section is followed by the description of the mathematical model of the robot. This model describes a relation between forces and torques affecting the quadrotor body. Following parts deal with the design of the state space controller. The next chapter explains software solution and implementation of the microcontroller. A computer vision system for automated landing is described at the end of the paper.

P. Sinčák et al. (eds.), *Emergent Trends in Robotics and Intelligent Systems*,
Advances in Intelligent Systems and Computing 316, DOI: 10.1007/978-3-319-10783-7_5

Fig. 1. Quadrotor prototype

2 Hardware

The auxiliary stabilization must be done by electronics because this robot cannot be controlled only by man. For this reason a necessary control unit was created. The control unit was developed just for this application. It can be divided into several parts. The main part is the microcontroller and power supply modules. Another part of the control unit is the wireless communication module. On Fig. 2, there is a block diagram of the control unit and ground station with wireless modules and a laptop computer.

All electronic devices on this robot are powered by the Li-Pol accumulator. In this construction a 3-cell accumulator with a nominal voltage 11.1 V is used. All engines use this voltage, while other electronics are supplied by 3.3 V and 5 V. Voltage level reduction is done by switching and LDO (Low-dropout Regulator) stabilizers.

The base of the control unit is the microcontroller LM3S8269. It is a 32-bit ARM with Cortex M3 architecture and operating up to 50 MHz. This device provides relatively high performance in comparison with similar projects (e.g. [1]), which is useful for testing various control algorithms. The microcontroller is also equipped with required communication buses (SPI, UART, I2C) and internal A/D converters.

The main sensor is an IMU (Inertial Measurement Unit) module Vectornav VN-100. This device provides us data about acceleration, angular speed and magnetic field in each axis, required for orientation determination [2]. Its advantage is the integrated Kalman filter. Next, an ultrasonic sensor SRF10 is used for measuring the distance to the ground in low altitudes.

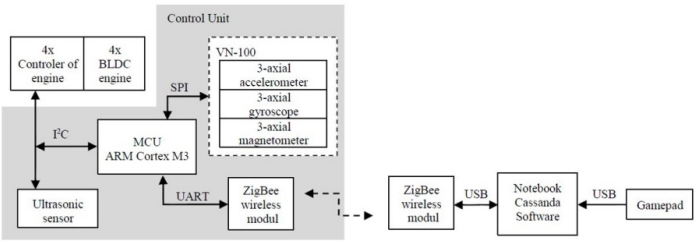

Fig. 2. The block diagram of control unit and ground station with user interface

3 Quadrotor Model

The mathematical model of the quadrotor was divided into two parts. The first part handles relations between the speed of rotors and forces and torques affecting the quadrotor rigid body. The second one handles the dynamics of the rigid body including the transformation between frames and the effect of gravity force. This separation allows to design the controller with better dynamics, because the first part can be easily linearized in the controller.

The forces and torques caused by propellers of rotors affecting the quadrotor rigid body were noted as u_1 to u_4. The meaning is following: u_1 - torque around x axis caused by different thrusts of rotors 2 and 4,u_2 - torque around y axis caused by different thrusts of rotors 1 and 3, u_3 - torque around z axis caused by different reaction torques from rotors rotating in the opposite direction, u_4 - force in z axis caused by the common thrust of all rotors.

Relation between the thrust and propeller speed of the rotor is

$$F_T = k_T \cdot n^2 \tag{1}$$

and the relation between the reaction torque and propeller speed of the rotor is

$$M_R = k_M \cdot n^2. \tag{2}$$

k_T and k_M are constantly determined by measurement. The first part of the model:

$$u_1 = \left(-n_2^2 + n_4^2\right)k_T l \tag{3}$$

$$u_2 = \left(n_1^2 - n_3^2\right)k_T l \tag{4}$$

$$u_3 = (-n_1 + n_2 - n3 + n_4)k_M \tag{5}$$

$$u_4 = \left(-n_1^2 + n_2^2 - n_3^2 + n_4^2\right)k_T \tag{6}$$

l denotes the distance between the rotors and the center of the gravity of the quadrotor. The second part of the model can be further divided into three parts: Equations that describe rotational movement, equations of linear movement, equations describing transformation between frames and equations expressing the effect of gravity force.

Rotational movement:

$$\dot{\omega}_x = \frac{u_1 + (I_{yy} - I_{zz})\omega_y\omega_z + u_1}{I_{xx}} \tag{7}$$

$$\dot{\omega}_y = \frac{u_2 + (I_{zz} - I_{xx})\omega_x\omega_z + u_2}{I_y y} \tag{8}$$

$$\dot{\omega}_z = \frac{u_3 + (I_{xx} - I_{yy})\omega_z\omega_y + u_3}{I_z z} \tag{9}$$

I_{xx}, I_{yy} and I_{zz} represent moments of inertia around given axis.
Translation movement:

$$\dot{v}_x = \frac{m\omega_y v_z + m\omega_z v_y + G_x}{m} \tag{10}$$

$$\dot{v}_y = \frac{m\omega_x v_z + m\omega_z v_x + G_y}{m} \tag{11}$$

$$\dot{v}_z = \frac{m\omega_x v_y + m\omega_y v_x + G_z - u_4}{m} \tag{12}$$

G_x, G_y and G_z are gravity forces component in given axis:

$$G_x = -mg \sin\ \theta \tag{13}$$

$$G_y = mg \cos\ \theta \sin\ \phi \tag{14}$$

$$G_z = mg \cos\ \theta \cos\ \phi \tag{15}$$

Transformation between robot frame and inertial frame [3]:

$$\dot{\phi} = \omega_x + \omega_y \sin\ \phi \tan\ \theta + \omega_z \cos\ \theta \tan\ \theta \tag{16}$$

$$\dot{\theta} = \omega_y \cos\ \theta - \omega_z \sin\ \phi \tag{17}$$

$$\dot{\psi} = \omega_y \frac{sin\ \phi}{cos\ \theta} + \omega_z \frac{cos\ \phi}{cos\ \theta} \tag{18}$$

$$\dot{x} = v_z(\sin\ \phi \sin\ \psi + \cos\ \phi \cos\ \psi \sin\ \theta) -$$
$$-v_y(\cos\ \phi \sin\ \psi - \cos\ \psi \sin\ \phi \sin\ \theta) + v_x \cos\ \theta \cos\ \psi \tag{19}$$

$$\dot{y} = v_y(\cos\ \phi \cos\ \psi + \sin\ \phi \sin\ \theta \sin\ \psi) -$$
$$-v_z(\cos\ \psi \sin\ \phi - \cos\ \phi \sin\ \theta \sin\ \psi) + v_x \cos\ \theta \cos\ \psi \tag{20}$$

$$\dot{z} = v_z \cos\ \phi \cos\ \theta - v_x \sin\ \theta + v_y \cos\ \theta \sin\ \phi \tag{21}$$

4 Controller Design

To stabilize the quadrotor, the controller based on the state space representation of the quadrotor was designed.

As it was shown in Sect. 3, the model of the quadrotor was divided into 2 parts. The first part handles the non-linear relation between the rotation speed of the propel-

lers and introduced variables u_1 - $u_4$with the meaning of forces and torques affecting the rigid body. Because in this part of model there are no dynamics, this relation can be expressed by purely static Equations (3). In the controller there is an inserted block with inverse function (8).

$$n_1 = \sqrt{\frac{2k_M u_2 - k_T l u_3 + k_M l u_4}{4k_M k_T l}} \tag{22}$$

$$n_2 = \sqrt{\frac{-2k_M u_1 + k_T l u_3 + k_M l u_4}{4k_M k_T l}} \tag{23}$$

$$n_3 = \sqrt{\frac{-2k_M u_2 - k_T l u_3 + k_M l u_4}{4k_M k_T l}} \tag{24}$$

$$n_4 = \sqrt{\frac{2k_M u_1 + k_T l u_3 + k_M l u_4}{4k_M k_T l}} \tag{25}$$

This block together with the first part of the model will act as the linear section with the unitary transfer function (while operating in the working range of rotor drivers). This allows direct use of variables u_1 - u_4 as inputs to the system and treats those parts of system as linear.

The schema of the system with the controller is on Fig. 3. The classical state controller is modified with a few improvements. The first one has added bias (u_{4-0}) to input u_4 that has the meaning of thrust needed to keep the quadrotor in hover flight. The next one is adding new state I_Z which is integral of error in altitude z. The purpose of this is to achieve a zero steady-state error while hovering, because it is almost impossible to set up the thrust that exactly compensates gravity force.

The design of the state space controller was based on the linearized model of the rigid part of the quadrotor. Equations (6) and (7) were linearized around the working point which is hovering. Following equation applies for the equilibrium point

$$\phi = \theta = \omega_x = \omega_y = 0. \tag{26}$$

To reach this state, the thrust of rotors must compensate the gravity force in hover flight acting in z axis. Thus

$$u_{4-0} = mg. \tag{27}$$

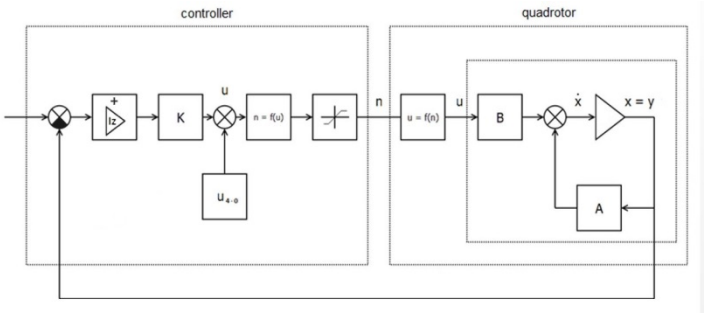

Fig. 3. Designed controller

Matrix **A** derived from linearized Equations (4), (5), (6) and (7):

	ω_x	ω_y	ω_z	v_x	v_y	v_z	ϕ	θ	ψ	x	y	z
ω_x	0	0	0	0	0	0	0	0	0	0	0	0
ω_y	0	0	0	0	0	0	0	0	0	0	0	0
ω_z	0	0	0	0	0	0	0	0	0	0	0	0
v_x	0	0	0	0	0	0	0	−9.81	0	0	0	0
v_y	0	0	0	0	0	0	9.81	0	0	0	0	0
v_z	0	0	0	0	0	0	0	0	0	0	0	0
ϕ	1	0	0	0	0	0	0	0	0	0	0	0
θ	0	1	0	0	0	0	0	0	0	0	0	0
ψ	0	0	1	0	0	0	0	0	0	0	0	0
x	0	0	0	1	0	0	0	0	0	0	0	0
y	0	0	0	0	1	0	0	0	0	0	0	0
z	0	0	0	0	0	1	0	0	0	0	0	0

Matrix **B**:

	u_1	u_2	u_3	u_4
ω_x	6.67	0	0	0
ω_y	0	6.67	0	0
ω_z	0	0	4.55	0
v_x	0	0	0	0
v_y	0	0	0	0
v_z	0	0	0	−1.23
ϕ	0	0	0	0
θ	0	0	0	0
ψ	0	0	0	0
x	0	0	0	0
y	0	0	0	0
z	0	0	0	0

Matrix C is identity matrix and matrix D is zero. Matrix K that determines control law was obtained experimentally by pole placement method.

Matrix K:

$$\begin{matrix} 1.05 & 0 & 0 & 0 & 0.29 & 0 & 2.66 & 0 & 0 & 0 & 0.12 & 0 & 0 \\ 0 & 1.05 & 0 & -0.29 & 0 & 0 & 0 & 2.66 & 0 & -0.12 & 0 & 0 & 0 \\ 0 & 0 & 0.66 & 0 & 0 & 0 & 0 & 0 & 0.44 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4.08 & 0 & 0 & 0 & 0 & 0 & -6.32 & -3.06 \end{matrix}$$

As seen above, the linearized state space description of the system regarded the quadrotor as an independent system in every axis and the controller handles it in this manner. In fact, when looking at altitude control, the state space controller can be understood as a set of separated PI or PD controllers. The results prove that this approach using the space controller and PID controllers gives very comparable results in this mode of flight.

5 Software Solution

The software solution of the quadrotor can be divided into several parts (see Fig. 4). Stabilizing algorithms are computed onboard to preserve real-time control and due to operation with no radio signal. Base station, which standardly consists of a PC or laptop, is equipped with the Cassandra system. The system allows the remote control of the group of mobile robots and visualizes various information. Quadrotor can also communicate with another base station, for example for the need of image data processing and other laboratory experiments.

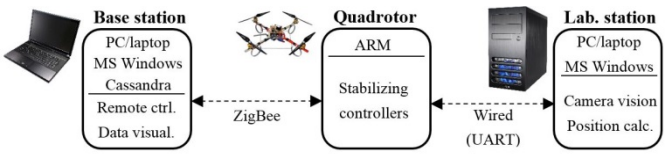

Fig. 4. Block diagram describing software solution

5.1 ARM Application

The main computing unit of the quadrotor is using the LM3S8962microcontroller which is based on the ARM Cortex-M3 architecture. The control program was written by the development environment Code Composer Studio v4 using C programming language. The microcontroller application does not use any operating system, but all tasks are handled in interrupt routines. To achieve a real-time behaviour, a system timer with the period of 1 ms was used, which is the shortest possible period to handle any application task.

The main quadrotor stabilization (angular rotations) is computed with a period of 5 ms due to the maximum output frequency of the inertial measurement unit VN-100, which is 200 Hz. With the same frequency, actuators speeds are updated. The sample

period of altitude stabilization depends on the used sensor, for example it is period of 25 ms for ultrasonic sensor.

Data from the base station are asynchronously handled by UART interrupt routine and then they are stored in a software ring buffer. For the reason of data synchronization and verification, data is transmitted in defined messages. The physical layer of communication is formed by asynchronous serial interface UART and Zig-Bee modules.

The application also checks special states, for example radio signal lost detection or battery low voltage detection and it is able to perform pre-programmed actions.

5.2 Base Station

The base station typically consists of a PC or laptop with a Microsoft Windows TM operating system. The computer must be equipped with a controller (gamepad or joystick), an appropriate communication interface and the Cassandra system [4].

Cassandra is a real-time robot control system for reconnaissance of previously unknown environments through a group of heterogeneous robots. It represents a relatively universal user interface program capable to control various robots in the similar way.

The user interface was designed to allow the operator to concentrate on the mission and to show the relevant information. The operator must not be flooded by the not-so-important data. The user interface, shown on Fig. 5, typically consists of the main camera image with a series of transparent overlay displays with important data.

Fig. 5. The screen of the user interface of the Cassandra operator system

5.3 Computer Vision System

This project also deals with an automated landing system. The aim is a solution that allows landing on a target. This target can be placed, e.g., on another robot on the ground. The main objective is the recognition and localization of landing mark on the image from the camera. Found coordinates of the mark are used to subsequently compute position stabilization.

In this case a camera is mounted on the flying robot. The better solution is an application with a mechanical stabilization system based on servos. The computer vision

system is realized on the board. In the laboratory it is possible to use a more simple configuration. The camera is on the ground, the recognition mark is on the quadrotor and the computer vision system is realized on classic PC.

The landing mark must enable the system to find out a position, orientation and altitude.

The selected mark is comprised of a big square with three sub-squares [5]. The position is calculated by the centre of the big square. Orientation is detectable by sub-squares and the altitude is calculated by the size of the mark in the image.

Image processing includes several operations with the image. It is considered landing only during day. Pre-processing includes an elimination of a disturbance by a convolution mask. The next operation is thresholding. In this case the daylight causes the brightness value of white parts on the mark to reach up to 255. The global threshold value is sufficed. After this, the edges pixels with high gradient are founded (Figure 6). An approximation of the edges causes linking of many lines with the same direction to one solid line. The desired square is described by four orthogonal lines. Lines that are orthogonal to each other are found and corners are stored as a potential area of the landing mark. The landing mark contains three small sub-squares. The method of search is the same as in the previous case, only the area of the application is inside the big square. Around the sub-squares are circumscribed circles. The centres of these circles are also the centres of the squares. The orientation point is the corner close to these squares.

All necessary data are known and the desired information is calculated. The altitude, x, y positions and the orientation are known.

Fig. 6. Edges after image processing and sub-squares

6 Conclusion

This article described current results of the development of the quadrotor type aerial robot at the Department of Control and Instrumentation of BUT. The aim is to develop a complete robot platform which will be independent from third party products and solutions.

The applied aluminium construction is suitable for testing thanks to its strength and variability, but it will be replaced by a carbon construction in the final version because of its low weight. The created control board, equipped with the ARM Cortex-M3 microcontroller, provides sufficient computing performance for various stabilizing algorithm testing and hardware resources to work with miscellaneous peripherals.

The robot can be remotely controlled from the base station and integration to the robotic system Cassandra will allow it to participate in various robotic missions in the

group of mobile robots in the future. The camera position system which is currently tested in the laboratory should facilitate autonomous landings.

The paper also briefly described the simplified mathematical model of the quadrotor which was divided into two parts. This allowed the linearization of the first part of the model and on its base the state space controller was designed. The complete form of the matrixes A, B, C and D which describe quadrotor behaviour and the form of matrix K which determines control law are shown in the paper. The resulting altitude stabilization behaves very similarly as a set of separated PI or PD controllers.

Acknowledgments. The work was supported by the Technology Agency of the Czech Republic under the project TE01020197 "Centre for Applied Cybernetics 3" under project CEITEC - Central European Institute of Technology (CZ.1.05/1.1.00/02.0068-) from the European Regional Development Fund.

References

1. Gurdan, D., Stumpf, J., Achtelik, M., Doth, K.-M., Hirzinger, G., Rus, D.: Energy-efficient Autonomous Four-rotor Flying Robot Controlled at 1 kHz. In: 2007 IEEE International Conference on Robotics and Automation (2007)
2. Silhan, M.: Odhad orientace UAV v prostoru. Diploma thesis - Czech Technical University in Prague, Faculty of Electrical Engineering (2010)
3. Solc, F.: Modelling and Control of a Quadrocopter. Advances in Military Technology 5(2), 29–38 (2010)
4. Zalud, L., Florian, T., Kopecny, L., Burian, F.: Cassandra - Heterogeneous Reconnaissance Robotic System for Dangerous Environments. In: Proceedings of 2011 IEEE/ SICE International Symposium on System Integration, pp. 1–6 (2011)
5. Sharp, C.S., Shakernia, O., Sastry, S.S.: A vision system for landing an unmanned aerial vehicle. In: IEEE International Conference Proceedings ICRA 2001, vol. 2, p. 1720 (2001)