Control of unmanned aerial vehicle based on XMOS platform

Conference Paper · November 2013

DOI: 10.1109/TELFOR.2013.6716374

4 authors:

Dejan Bokan
RT-RK Computer Based Systems

Nenad Četić
University of Novi Sad

Jelena Kovacevic
RT-RK Computer Based Systems

Velibor Ilic
The Institute for Artificial Intelligence Research and Development of Serbia

Control of unmanned aerial vehicle based on XMOS platform

Dejan Bokan, Nenad Četić, Jelena Kovačević, *Member, IEEE,* Velibor Ilić

***Abstract* — This paper describes one solution for quadrotor aircraft control based on XMOS platform. It describes implementation of systems for orientation and position estimation in 3-D space, and algorithm for reaching and maintaining requested position and orientation based on PID regulation. It also contains description of quadrotor physical model built for the purpose of system verification.**

***Keywords* — Unmanned aerial vehicle, XMOS, xCORE, PID regulation, quadrotor**

I. Introduction

UNMANNED aerial vehicle (UAV), also known as a drone, is an aircraft without a human pilot on board. Its flight is controlled either autonomously by computer or under the remote control of a pilot on the ground [1].

In this paper, we studied the behavior of the quadrotor. This physical configuration has been chosen for this research for its low dimension, good maneuverability, simple mechanics and payload capability. The quadrotor is an aircraft with four propellers placed around a body frame. The power source and control hardware are placed on a main frame. The vehicle is controlled by four rotors. Independence of rotors rotational speeds makes it possible to control the attitude of the vehicle. Movement is produced by the total thrust of the rotors. Direction of movement varies according to quadrotor orientation [2].

The quadrotor control in this paper was implemented using xCORE architecture developed by XMOS. xCORE is a class of microcontroller that has multiple processor cores, flexible I/O, and a unique timing deterministic architecture. Using xCORE, you can easily design embedded electronic systems which exactly meet the exact timing. Unlike conventional microcontrollers, the xCORE multicore microcontroller is able to run multiple real - time tasks simultaneously. It is made up from Tiles that contain multiple 32bit logical processor cores which execute the program using RISC instructions. Each logical processor core can execute tasks running computational code, DSP code, control software (including taking logic decisions and executing a state machine) or software that handles I/O operations. xCORE microcontrollers support high - level programming [3].

Dejan Bokan, Faculty of Technical Sciences Novi Sad, Trg Dositeja Obradovića 6, Novi Sad, Serbia (e-mail: dejan.bokan@rt-rk.com).

Nenad Četić, Faculty of Technical Sciences Novi Sad, Trg Dositeja Obradovića 6, Novi Sad, Serbia (e-mail: nenad.cetic@rt-rk.com).

Jelena Kovačević, Faculty of Technical Sciences Novi Sad, Trg Dositeja Obradovića 6, Novi Sad, Serbia (e-mail: jelena.kovacevic@rt-rk.com).

Velibor Ilić, RT-RK Institute for Computer Based Systems LLC, Novi Sad, Serbia (e-mail: velibor.ilic@rt-rk.com)

II. Quadrotor Flight Dynamics

In this section, the basics of the quadrotor flight dynamics will be described. It is assumed that the configuration of the aircraft is as follows: There is one rotor at the front and one at the rear, while the other two are on the sides. The front and rear rotors turn clockwise whilst the left and right rotors turn the opposite way in order to cancel the torques. Entire quadrotor movement can be presented through four basic movements: pitch, roll, yaw and altitude change, as seen in Fig.1. To pitch up, thrust in the front rotor is increased while thrust in the rear one is decreased in the same quantity. This produces a pitch movement while global thrust and torque are kept unchanged. Pitching down is analogous. Roll control is performed in a similar manner, by increasing thrust in one of the side rotors and decreasing in the other. Yaw movement is produced by increasing thrust in rotors that rotate same way and decreasing the opposite pair. For instance, to yaw to the right, thrust and therefore torque are reduced in the rotors rotating clockwise and increased in those rotating counter clockwise. This is also needs to keep the global thrust unchanged. The direction of reaction torque and the direction in which the rotors turn are opposite.

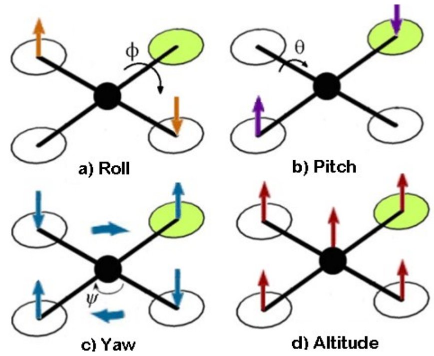

Fig. 1. Quadrotor basic movements

From a steady hovering state, to ascend or descend, thrust in the all four rotors is changed for the same amount. To produce a forward or backward movement a certain pitch angle has to be maintained. Lateral translation is analogous [4].

III. Solution Concept

In order to implement quadrotor control algorithm, system which consists of the following units, is developed:

- Main control unit
- System for estimation of quadrotor attitude
- System for estimation of quadrotor position
- System that represents user interface

A. Main control unit

This unit contains control algorithm implementation separated in four blocks presented in Fig.2.

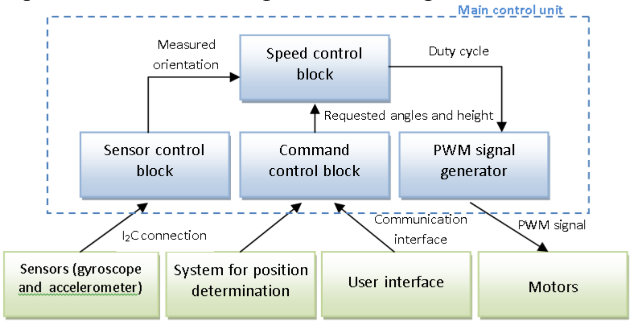

Fig. 2. Main control unit diagram

Sensor control block is connected to sensors mounted on quadrotor body via I2C interface. Set of sensors consists of 3 – axis gyroscope that measures the rotation around the axes, and 3 – axis accelerometer that measures proper acceleration by axes. Main tasks of this block are configuration and initialization of sensors, sensor calibration and receiving and processing data. Received data contains information about current orientation in sensor frame, so it is necessary to perform certain mathematical transformations to calculate the orientation in body frame of quadrotor. Calculated data is presented in the form of Euclid angles (yaw, pitch and roll) and forwarded to speed control block.

Command control block is in charge for receiving commands related to the position of the quadrotor, parsing commands, processing and transfer to the speed control block. Types of command messages that can be received are: information about the current position of the aircraft in the XY plane, information on a requested position in the XY plane, and information about the required change of aircraft height. Each message begins with an 8-bit flag that indicates the type of message. Message that contains current position contains the coordinates of two points on a quadrotor body. Two points are needed so that measured and requested position coordinates can be presented in the coordinate system that corresponds to quadrotor body frame. Desired values of required roll and pitch angles are being calculated based on the position error, using proportional regulator with limited range of output values, in order to prevent excessive slope and loss of flight stability. These values are being forwarded to speed control block. Message with requested height change is forwarded directly to the speed control block.

Speed control block calculates the appropriate value of duty cycle for each of four motors, based on information received from sensor control block and command control block. Computation is performed by applying speed control algorithm. Orientation error that represents difference between current and requested orientation, together with requested height velocity makes a set of input parameters for speed control algorithm. Algorithm outputs are new values of duty cycles for every motor and are being forwarded to block for generating PWM signals.

PWM signal generator has the task of continuously generating an output signal using pulse width modulation technique. Frequency and resolution of signals are defined during initialization of the block, as the initial cycle. During execution, speed control block can change the value of duty cycle of each motor.

B. Speed control algorithm

Quadrotor dynamics is described by the following system of equations:

$$\begin{cases} \ddot{Z} = -g + (\cos\theta\cos\phi)\dfrac{U_1}{m} \\ \ddot{\phi} = \dfrac{U_2}{I_{XX}} \\ \ddot{\theta} = \dfrac{U_3}{I_{YY}} \\ \ddot{\psi} = \dfrac{U_4}{I_{ZZ}} \end{cases} \tag{1}$$

where $\ddot{Z}$ represents height velocity, and $\ddot{\varphi}$, $\ddot{\theta}$ and $\ddot{\psi}$ angular velocity for angles of pitch, roll and yaw. Gravitational acceleration is marked with g, aircraft mass with m, and Ixx, Iyy and Izz are moments of inertia around axis. System input variables (U_1, U_2, U_3, U_4) are combinations of propellers squared speeds, and are defined by the following equitation system:

$$\begin{cases} U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_2 = lb(-\Omega_2^2 + \Omega_4^2) \\ U_3 = lb(-\Omega_1^2 + \Omega_3^2) \\ U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ \Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \end{cases} \tag{2}$$

where Ω is propellers angular speed, b thrust factor constant, d drag factor constant, and l is distance between center of quadrotor and center of propeller [5].

The control can be divided into three components:

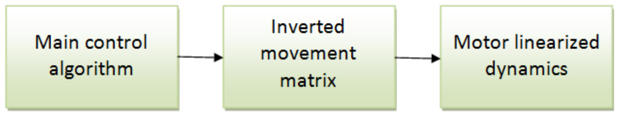

Fig. 3. Speed control algorithm

Main control algorithm represents the core of the speed control algorithm. It processes the requested and the sensors data and provides a signal for each basic movement which balances the position error. Equation (1) is used in this block to transfer an acceleration command to a basic movement one. The control rules used to estimate the acceleration commands are PID techniques. Entire process consists of four regulators. The first one is used for roll angle data and is shown on Fig. 4. $\varphi^d[rad]$ represents the desired roll angle, φ $[rad]$ is the measured roll angle, e_φ $[rad]$ is the roll error and U_2 $[N\ m]$ is the required roll torque. $K_{p\varphi}$ $[s^{-2}]$, $K_{I\varphi}$ $[s^{-3}]$ and $K_{d\varphi}$ $[s^{-1}]$ are the three control parameters. At last I_{XX} $[N\ m]$ is the body moment of inertia around the x-axis.

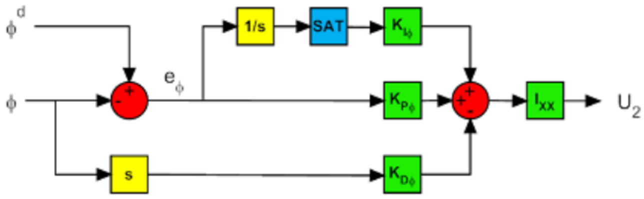

Fig. 4. Roll control PID regulator

Pitch control regulator has a same structure, but the input values are related to pitch angle instead of roll, and the output is U_3 $[N\ m]$ control signal.

Yaw control regulator structure presents a few differences from the previous two. Input, ψ^d $[rad\ s^{-1}]$

represents the desired yaw angle velocity, instead of just desired angle. The new block, *folder,* represents a code which allows taking into account the discontinuity of the yaw angle at ±π, e.g. if the input of the block is equal to 3π/2, it outputs −π/2. The block *sign coherence* is a little bit more complex. It is needed to consider the sign of the two samples for the derivate computation. Once again, the purpose of this algorithm is used to avoid error during the transition π to -π and vice versa.

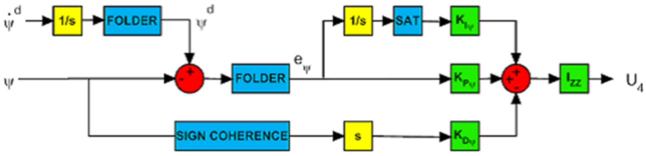

Fig. 5. Yaw control PID regulator

Height control regulator is presented in Fig.6. Input $\dot{Z}^d$ [m] represents the desired height velocity, and output U_1 [N] is the required thrust. At last, g [$m\ s^{-2}$] is the acceleration due to gravity, and m [kg] is the mass of the quadrotor. In order to achieve greater stability due to existence of differences in the motor characteristics, the force required to achieve hover state will not be calculated by the formula $m*g$. The value will be determined experimentally for each motor.

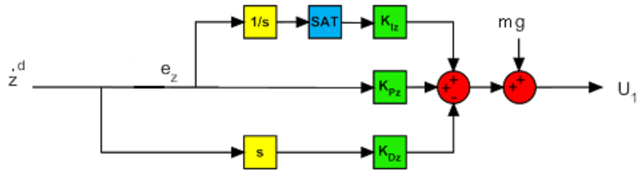

Fig. 6. Height control PID regulator

Inverted movement matrix is the second block in the control chain. It is used to compute the propellers squared speed from the four basic movement signals. It is calculated by inverting the movement matrix (2):

$$\begin{cases} \Omega_1^2 = \dfrac{1}{4b}U_1 - \dfrac{1}{2bl}U_3 - \dfrac{1}{4d}U_4 \\ \Omega_2^2 = \dfrac{1}{4b}U_1 - \dfrac{1}{2bl}U_2 + \dfrac{1}{4d}U_4 \\ \Omega_3^2 = \dfrac{1}{4b}U_1 + \dfrac{1}{2bl}U_3 - \dfrac{1}{4d}U_4 \\ \Omega_4^2 = \dfrac{1}{4b}U_1 + \dfrac{1}{2bl}U_2 + \dfrac{1}{4d}U_4 \end{cases} \quad (3)$$

Motor linearized dynamics is the third block in the control chain. This block is responsible for calculating the required voltage level on the engines to achieve requested propellers speed. The characteristic of DC motor is nonlinear differential equations. For the sake of simplification of the block, we found a linear dependence of these two quantities. Linearization is achieved by experimental measurements, and linear approximation of relation propellers speed to requested input voltage. This block determines adequate voltage values, and calculates PWM duty cycles for every engine based on it.

C. *System for estimation of quadrotor attitude*

Set of sensors sufficient for accurate estimation of quadrotor attitude consists of 3-axis gyroscope and 3-axis accelerometer. Gyroscope measures angular speed velocity around 3 axes (roll, pitch and yaw). In ideal conditions, by calculating integral of these values, based on the initial orientation, the exact orientation of the aircraft can be determined at any time. But in normal conditions, a problem appears in form of noise at gyroscope output that is later manifested as a drift in integral calculation output. Therefore, we are required to perform error correction using output value of accelerometer.

Accelerometer is a device used for detecting the value of inertial forces acting on the body of aircraft along the x, y and z axis. The accelerometer output also contains noise, so these values alone cannot provide precise orientation estimation either. There are several different ways to combine these two devices.

In this paper we used chip MPU-6050 which contains 3-axis gyroscope, 3-axis accelerometer and Digital Motion Processor™ [6]. MPU-6050 is connected to microcontroller via I2C bus. DMP™ contains implemented algorithm for combing these two devices. Estimated orientation is obtained as an output from DMP™, in form of quaternion. Quaternion Q (w, x, y, z) in mathematics represents four dimensional variable. Orientation presentation in form of quaternion consists of three dimensional vector which is the axis of object rotation, and a scalar value that represents the value of angle for which the object is rotated.

D. *System for estimation of quadrotor position*

For the purpose of estimating the position of quadrotor a system is designed based on object detection in video. The system consists of two cameras that capture two perpendicular planes. The implemented algorithm for object detection is executed on a PC.

The camera that is positioned above the space predicted for quadrotor flight captures the XY-plane, and based on that footage the x and y coordinates of the aircraft are determined. The second camera captures the XZ-plane. Based on this image z-coordinate of the aircraft is detected. It is necessary to calculate true height value based on detected pixel value. For this we need to obtain distance between quadrotor and camera 2, based on video from camera 1. Once we have this information, the actual height is calculated as follows:

$$H = \left(H_{det} + {}^{H_{max}}/_{2}\right) * {}^{D}/_{D_{max}} - {}^{H_{max}}/_{2} \quad (1)$$

where H is the actual height in pixels, H_{det} is measured height in pixels, H_{max} is maximal possible height value, D is distance of aircraft from the camera 2, and D_{max} is maximum distance.

The picture, on which the algorithm is applied, is in HSV format. Before applying algorithm, it is necessary to experimentally determine ranges of object color components. First step is smoothing the image by applying low pass filter (Gaussian). Then, binary image is formed where requested color pixels are marked with 1. The next step is detection of *white island* (group of pixels with value 1) and calculation of its surface. After that, it is checked if calculated surface is greater than given threshold to avoid detection of noise as an object. Next the coordinates of island center are determined, and they represent the object position. The algorithm is implemented using *OpenCV* library [7].

E. *User interface*

In this section a simple user interface is implemented. This system enables user to specify the position where he/she wants quadrotor to be located. Just like the previous one, this unit runs on PC and it uses video from two cameras.

GUI consists of two windows as seen in Fig.7. In each of them a video from one of the cameras is shown. Current position of quadrotor is marked with red circle. The green circle represents requested XY position, while the green line represents requested height. User sets the new requested values by pressing left mouse button.

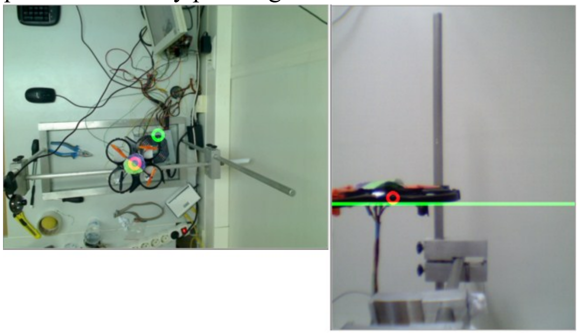

Fig. 7. Graphical user interface

IV. TESTING AND VERIFICATION

For the purpose of testing and verification, a physical model of the system was formed. Diagram of entire system is presented in Fig.8. As aircraft body, plastic frame and DC brushed motors took from *UDI RC U816A* aircraft are used. Control algorithm is being executed on *XMOS Slice Kit* development board, which contains *xCORE L-16* microcontroller [3]. *InvenSense MPU-6050* chip is used as orientation sensors. Motor control is done by two dual half bridges *ST L298N* [8]. Every half bridge controls two motors. The size of the aircraft is relatively small, so *Slice Kit* board was too heavy for quadrotor to carry. Because of that, quadrotor was controlled from the ground. Only sensor chip was placed on the aircraft body. Physical connection between ground chips and aircraft was established using long thin wires.

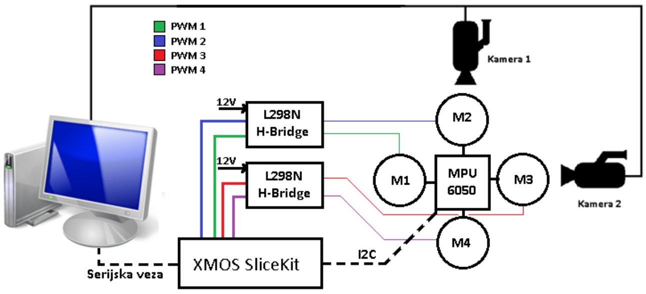

Fig. 8. Diagram of entire system

Testing of solution was carried out in several stages. First tests covered testing of every system unit individually. For the purpose of testing, additional applications were developed. Test results showed that units for position estimation, orientation estimation and PWM signal generator are working properly.

Next phase is testing of PID controller and angles error correction. To enable testing of error correction of each angle separately, quadrotor was fixed to a special stand so it can move around only one axis. First we tested roll angle then pitch angle correction. Results were recorded in real-time usnig *XMOS* tool called *xScope* wich allows system monitoring and data access without affecting the execution of application. The results of orientation maintaining are presented in Fig.9.

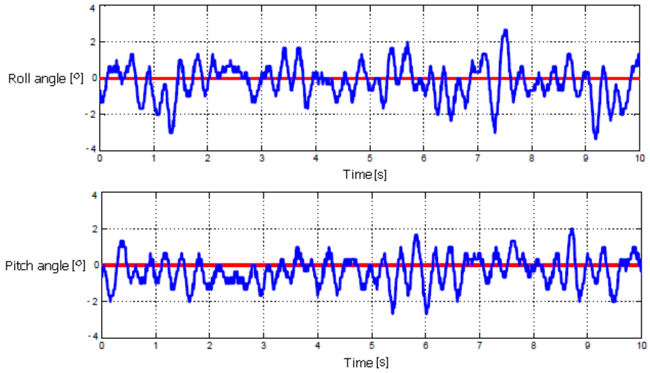

Fig. 9. Results of maintaining orientation arround one axis

Requested value is presented with red line, and measured with blue. Small oscillations around stable point are the impact of the string which is used to fix quadrotor to a stand. The results of test showed the correctness of the PID controller, and successful maintaining of orientation.

Next test represents an attempt of stable autonomous indoor flight of a quadrotor. The aim of the test is to make aircraft take off, reach certain position requested. Successful flight was achieved, and so was the maintaining of height, however maintaining of the position failed. The main causes of the test failure are the physical limitations of constructed physical model. Used engines were too weak, to lift the aircraft, so voltage, higher than maximal allowed by specification, was used. It caused increased vibration of quadrotor body. By the influence of this vibration, an error appeared on the sensor output. This error could not be predicted nor eliminated.

V. CONCLUSION

Platform xCORE proved its ability to handle demanding real-time tasks, like UAV control. To proceed with further testing and development it is required to build better physical model of quadrotor.

ACKNOWLEDGMENTS

This work was partially supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia under Grant TR-32031.

REFERENCES

[1] K. Valavanis, *Advances in unmanned aerial vehicles state of the art and the road to autonomy*, Springer, Dordrecht, 2007.

[2] Atheer L. Salih, M. Moghavvemi, Haider A. F. Mohamed and Khalaf Sallom Gaeid, *Flight PID controller design for a UAV quadrotor*, Scientific Research and Essays Vol. 5(23), pp. 3660-3667, 4 December, 2010.

[3] David May, *The XMOS XS1 Architecture,* XMOS Limited, UK, 2009.

[4] Vicente Martinez Martinez, *Modelling of the Flight Dynamics of a Quadrotor Helicopter*, A MSc Thesis in Cranfield University, Septembar 2007.

[5] Samir Bouabdallah, *Design and control of quadrotors with application to autonomous flying*, Ecole Polytechnique Federale de Lausanne, 2007.

[6] *MPU-6000 and MPU-6050 Product Specification Revision 3.3*, InvenSense Inc., June 2012.

[7] *OpenCV (Open Source Computer Vision Library),* http://opencv.org/documentation.html, July 2013.

[8] *L298 dual full bridge driver datasheet,* STMicroelectronics, January 2000.