Crazyflie 2.0 Quadrotor as a Platform for Research and Education in Robotics and Control Engineering

Wojciech Giernacki, Mateusz Skwierczyński, Wojciech Witwicki, Paweł Wroński, Piotr Kozierski
Faculty of Electrical Engineering, Institute of Control and Information Engineering
Poznan University of Technology
Piotrowo 3A, 60-965 Poznan, Poland
e-mail: wojciech.giernacki@put.poznan.pl

***Abstract*—In this paper a *Crazyflie 2.0* nano quadrotor helicopter (quadcopter) as an open source experimental platform for research and education in robotics and control engineering has been presented. This low cost, easily expandable and upgradeable flying robot is here characterized in terms of hardware and software. Three aspects, which demonstrate the potential of broad use of this unmanned aerial vehicle (UAV) by researchers and students, are discussed in the paper. The first one is an acquisition of measurement data from test flights by the proposed, freely available "black-box" software. The second is the use of a new, advanced *4FLY Simulator* in order to utilize the *MATLAB®/Simulink* environment to easily implement a mathematical model of *Crazyflie 2.0* dynamics, as well as for a synthesis of various types of controllers with support of *OpenGL* cross-language in the visualization of simulations results. The *4FLY Simulator* allows to test autonomous flights (and landings) with obstacles avoidance and to conduct learning and teaching the basics of *Crazyflie 2.0* piloting. In the third aspect the authors outlined promising, preliminary results obtained in control of flying robot by pointing device (positioner) and with the support of a vision system, which basis only on a single *Kinect* sensor.**

Keywords—Crazyflie 2.0; nano quadrotor helicopter; nano quadcopter; flight simulator; autonomous flights; obstacles avoidance; autonomous landing; vision-based tracking

I. INTRODUCTION

There are a number of unmanned aerial vehicles (UAVs) construction types. The most widely used are: fixed-wing [1] and multirotors [2]. The second group is preferred here due to possibility of vertical take-off and landing (VTOL). In recent years quadrotor helicopters achieved popularity due to mechanical simplicity of the robot's construction – change of UAV's position and orientation in the air is the result of solely speed changes of particular propulsion units. The quadrotor is inherently unstable, multidimensional, underactuated object, with highly non-linear dynamics, and its parameters can be generally considered as non-stationary in time. Therefore it may be an interesting platform for the part of scientific community and due to the focused properties UAVs are usable in many areas of robotics and control engineering research. An access to low-cost and compact size flying robots is still desirable in conducting research and projects with students – especially in laboratories with limited space. In many countries outdoor flights using these lightweight (micro or nano) UAVs still rarely require special permissions regulated by law, but they must primarily ensure the level of security that allows to use them to fly in environment close to people. In recent years, *AR.Drone 2.0* is widely used [3] due to compact size, mechanical strength of its construction during unexpected emergency landings and extensive facilities in a number of sensors (e.g., two on-board video cameras). Unfortunately, this UAV is commercial, developed for consumer entertainment (mostly for augmented reality games). This fact introduces difficulties and limitations of using it as an open platform for research and development (there is a lack of full/open documentation and scientific community support for *AR.Drone 2.0*).

Fig. 1. *The Crazyflie 2.0* quadrotor helicopter 3D model

An alternative solution, i.e., the *Crazyflie 2.0* flying robot (see Fig. 1) is proposed to use in this paper. The novelty and scientific contribution is here a development of dedicated, open source research and education tools. On their examples one demonstrated (to the authors' best knowledge for the first time in world literature) the effectiveness of proposed control system synthesis for the robot from [6] with introduced mechanisms of an autonomous soft landing and obstacles avoidance. The paper presents also the preliminary results of the research on autonomous tracking of the reference changes of the robot's position and orientation in 3D space by the use of the low-cost vision system [4].

In the second section, the hardware and software of *Crazyflie 2.0* is presented, as well as a state of the art in world research based on this robot. Section III contains basic information about *Crazyflie Java Client*. In Section IV one may find a description of the *4FLY Simulator* project and proposed algorithms of an autonomous soft landing and obstacles avoidance (verified in tests). In Section V the preliminary results in control of *Crazyflie 2.0* by pointing device and just one *Kinect* sensor are shown.

II. CRAZYFLIE 2.0 FLYING ROBOT

A. Background

Nowadays, several of world's known research centers and technical universities use the possibilities offered by the *Crazyflie 2.0*. Research is being conducted in the context of parametric identification methods for the mathematical model of its dynamics [5, 6], as well as for designing of position and trajectory control algorithms for a single robot [6], especially operating in cluttered environments [7]. In [8], the use of UAV for creation of stippled prints on canvas using an ink soaked sponge, was proposed. Moreover, nano size predefines *Crazyflie 2.0* also for work in groups (swarms) of robots – such as: online trajectory planning in cluttered environments [9], large, 49-vehicles formation flights [10] or to explore the field of mixed reality [11]. In order to more outline the area of UAV potential applications, a brief introduction to the *Crazyflie 2.0* is presented below.

B. Hardware

Crazyflie 2.0 is a nano quadrotor helicopter equipped with four 7x16 mm coreless DC motors (Kv: 14000 rpm/V, rated voltage: 4.2 V, rated current: 1000mA, weight: 2,7 g, shaft length: 3,5 mm, shaft diameter: 0.8 mm) and 45 mm plastic propellers. Propulsion units are attached to the circuit-board frame of the UAV. Thus it measures only 92 mm between diagonally opposed motor shafts and is 29 mm height. Total weight is contained in 27 g. Payload is limited to 15 g. *Crazyflie 2.0* is supplied by 1 Cell (3.7 V), 20x30x7 mm, 240 mAh LiPo battery (weight: 7.1 g). It provides energy up to 7 minutes of continuous flight.

This nano flying robot is based on two microcontrollers: main and additional (32 MHz *nRF51822 ARM Cortex-M0* processor for power energy and radio communication management). The main microcontroller is an ARM 32-bit *STM32F405 Cortex*-M4 embedded processor (with floating-point unit) running at 168 MHz with 192kb of SRAM. It communicates with ground station (PC computer with USB dongle) and being controlled over the 2.4 GHz *Crazyradio PA* (with *Nordic Semiconductor nRF24LU1+*) in up to 1 km range line-of-sight (with transmission up to 2 Mb/sec in 32-byte packets). As a manual controller, a gamepad with four analog axes may be used. *Crazyflie 2.0* can be control also using a smartphone or tablet via *Bluetooth* (default by changes of *roll*, *pitch*, *yaw* and *thrust* values).

The robot on-board sensors system is based on 10-DOF (degrees of freedom) Inertial Measurement Unit (IMU), i.e., *MPU-9250* with three axis: gyro, accelerometer and magnetometer, as well as additional high precision pressure sensor (*LPS25H*). All sensors provide measurements to stabilize the flight. By the expansion interface the users have a direct access to such buses as *UART*, *I2C* and *SPI*. Moreover, the whole hardware architecture of the robot is open source, as well as software. More information may be found in [12, 13].

C. Software

The beginning of *Crazyflie* project was in year 2009. Swedish developers from *Bitcraze AB* company, aimed whole effort to propose versatile flying development platform ("*Made by developers for developers*"). Thus now, from December 2015, science community is able to work with *Crazyflie 2.0* source code and final product documentation. Researchers and students can modify the firmware and the software for variety of research and educational purposes, as well as share the knowledge on the forum [14] with the community.

The UAV's firmware is based on the *FreeRTOS* open source real-time operating system. The default software (*Crazyflie Python Client – CPC*) is written in the *Python* programming language, but there are several alternatives for *Linux*, *Mac OS* and *Windows* operating systems (in *Ruby*, *C*, *C++*, *C#*, *Javascript* and *Java* languages) – available at the *GitHub* hosting service [13]. The *CPC* software offers a number of useful capabilities for both beginners and advanced users. Primarily, it enables quadrotor control using the predefined controller with a preview in real time for the most important flight parameters and an attitude indicator. The software, in the *Parameters* menu, allows to preview and set the values of particular controllers. Moreover, a control signal saturation level can be introduced (in the *Advance* mode). *The Crazyflie 2.0* flight can be recorded and presented on plots (*Plotter* menu). Attached library in *Python* language encourages the further personalization of the *CPC* [13].

D. State of the art in mathematical modeling and control

Due to limited volume of the article, the authors decided only to briefly specify references to valuable papers, where a necessary information, extending below issues, are provided. Respectively, in [5] on page 75, one can find identified inertia matrix for 28 g weight *Crazyflie 2.0*. This matrix is used in mathematical models of quadrotors. Moreover, the author provided information about moment of inertia of a point mass with distance from axis that approximately equals a half of the robot diameter (0.092 m). Motor parameters, e.g., discrete transfer function between input command given to the motors and produced thrust for UAV's motors, thrust and rotor angular velocity as functions of input command, as well as information about particular drag coefficients and their proposed models, have been presented. A classical, well-established in literature [15], parametrized model of the *Crazyflie 2.0* dynamics (with motion capture *Vicon* system's markers on-board) is explained in detail in [6] and [7], where one can find the thrust and torque coefficient values. In [6] the author proposed the linearized model in a state space representation. In this paper the robot mathematical model from [15] has been used. The sets of controllers [6] are presented in Table 1 (default from *Bitcraze* are bold). Naturally, they are non-optimal sets, but only a reference for own tuning.

TABLE I. *CRAZYFLIE 2.0* SETS OF CONTROLLERS FROM [6]

	Roll & Pitch Rate	Yaw Rate	Roll & Pitch Rate	Yaw Attitude	Altitude	X & Y
K_P	**70**	**70**	**3.5**	**0** or 3	11000	30
K_I	**0**	**50** or 16.7	**2**	**0**	3500	2
K_D	**0**	**0**	**0**	**0**	9000	0

III. *CRAZYFLIE JAVA CLIENT*

The *Crazyflie Python Client* software and library [16] were the inspiration for development of *Crazyflie Java Client* (CJC) [17]. It was created to provide to scientific community and students a simpler tool predefined for two purposes: a preview on basic flight parameters (Fig. 2) and for the archiving of all possible data from *Crazyflie 2.0* (especially from its sensors). After connecting the robot via USB to the ground station, it is possible to explain to students on the artificial horizon how to control the quadrotor in the preview mode (and meaning of *roll*, *pitch* and *yaw* angles, as well as *thrust*). In flight mode one can observe changes of speeds of individual propulsion units while maneuvering, and also one can monitor the battery charge level. Introduced minimalism enables to use the software primarily as data logger for research and student projects such as a development of state vector estimators and an identification of UAV model parameters. Fusion of sensors data is also possible, as well as an analysis of robot dynamics characteristics and the testing of the controllers synthesized in the *4FLY Simulator* – presented in the following section.

The *CJC* is available as an open source software among others in *MMD database* [18], where its capabilities are widely demonstrated. The use of *Java* language in conjunction with *Apache Maven* and *Java FX* enabled to propose software for multiple platforms and operation systems (*CJC* requires only a *Java Virtual Machine* on a particular device). The software allows to record data from flights (up to 6 different parameters simultaneously) directly to *.csv* file or to one of two databases: relational – *MySQL* (*Java DataBase Connectivity*) and non-relational – *MongoDP* (as a form of objects in a format similar to *JSON*).

IV. *4FLY SIMULATOR*

There are a number of ideas and approaches to UAVs simulation. Popular among control engineers *MATLAB*® environment is dominated by solutions which use a blocks architecture of the *Simulink* library and related toolboxes, such as the open source *Robotics Toolbox*. By using of *m*-files and *s*-functions, conducting of a control system synthesis, an optimization, a path planning and a swarm intelligence of UAVs is possible, but the presentation of results in the form of basic time courses or limited, simple animations focused attention of scientific community (mainly roboticists) in direction of solutions commonly used under *Robot Operating System* on *Linux* and in *C* programming language. The software such as GAZEBO enable to integrate functionalities (e.g., sensory data from the real environment and robot in 3D virtual reality [11]). The use of virtual environments like UNITY 3D or V-REP is also widespread in this context. In this paper, the application of the *OpenGL* multiplatform library (with *GLFW*, *GLEW* and *GLM* packages) and CAD-type (Computer Aided Design) software are proposed for tasks conducted in *MATLAB*® environment – for example the *4FLY Simulator* dedicated mainly to multirotor flying robots, e.g., *Crazyflie 2.0* [19]. The idea and some functionality blocks of the simulator are shown in Fig. 3. Figure 4 presents the integration of used environments. The proposed communication via *MATLAB*® *Engine* is a more efficient alternative to commonly used *User Datagram Protocol* (UDP).

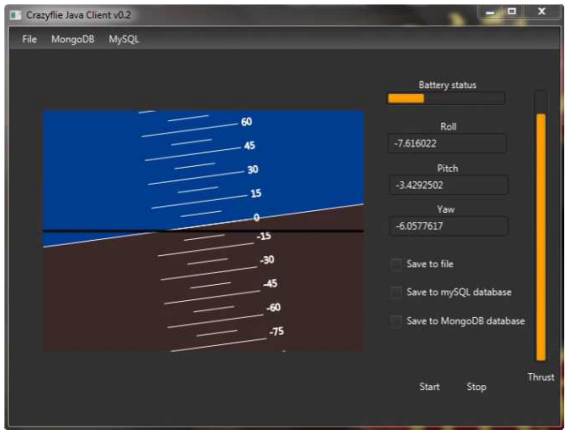

Fig. 2. *Crazyflie Java Client interface*

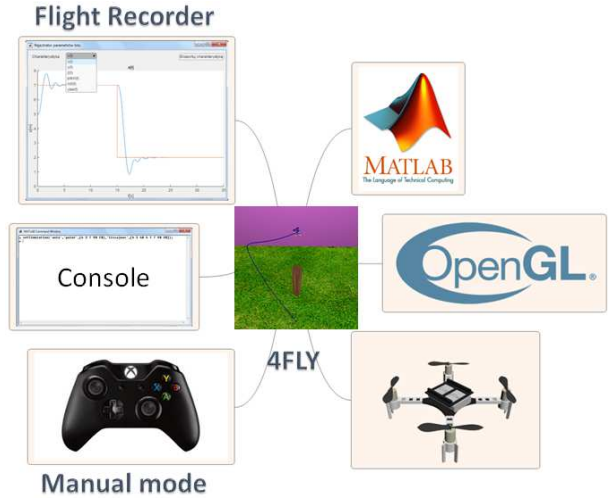

Fig. 3. *FLY Simulator* idea

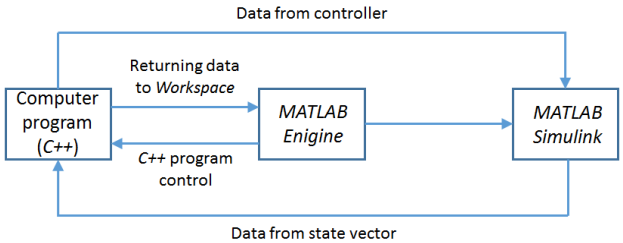

Fig. 4. Software and signals integration in *4FLY Simulator*

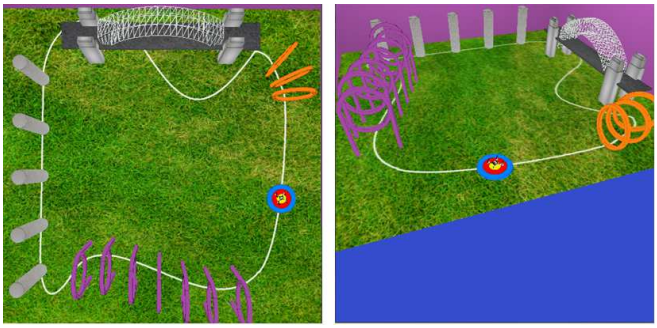

Fig. 5. The example of scene used in dydactics of *Crazyflie 2.0* manual control in *4FLY Simulator* – views from above and side of the scene

Examples of flight simulator capabilities are widely presented on the *MMD database* website [18]. The *4Fly Simulator* has been written in the *C++* programming language. The use of *Assimp* library enables to import CAD models (of the *Crazyflie 2.0* or any other flying robot saved in *.obj* format) to the simulator in *OpenGL* graphical environment. The graphics engine allows to visualize the results of work performed in real time in *MATLAB®/Simulink* or simulate manual control of the UAV in one of the preset scenes consisting of a set route and a number of obstacles (see Fig. 5). The *4FLY Simulator* enables also the use of scenes prepared in CAD programs. In 'manual' mode, the camera is directed on quadrotor, so it will always track the UAV flight in TPP (third-person perspective). Such solution enables a convenient quadrotor control using the controller. The dynamic lighting usage enhances the effect of a scene depth. In the mode dedicated to autonomous flight simulations, the camera track the center of scene, so the user has a convenient preview of the entire contents. The authors proposed for this operation mode a transparent graphical interface and control from *Console*. The simulator enables the analysis of tests results and export them to files after its end. The proposed method of flight paths setting is discussed below.

a) 'Auto' mode

The 'Auto' mode is characterized by the possibility of manually defining all the flight parameters from the control panel level (*Command Window* in *MATLAB Engine*). In this mode, it is possible to predefine individual points of a particular path, as well as trajectories in the form of Lissajous curves. There is also the option to combine points with advanced trajectories to build full mission scenarios.

b) 'Waypoint' mode

A flight path is determined graphically using a computer mouse. In windows of *4FLY simulator* one needs to set desired coordinates in *X*, *Y* axes, as well as altitude (*Z*) and *yaw* values.

The use of *MATLAB®/Simulink* as a modeling tool of an autonomous control architecture, quadrotor dynamics, as well as a parametrization for simulation conditions makes *4FLY Simulator* a complete research tool. In general, it allows a synthesis of any type of UAV control system (low- and high-level), as well as tests with mathematical models of multirotor dynamics. To demonstrate research aspects it has been decided to present the results of implementation in *4FLY Simulator* of two selected mechanisms which have been described below.

a) Obstacles avoidance

For the reference path from Fig. 6 with the object on the UAV flight path, it is necessary to propose an efficient obstacles avoidance algorithm. For this purpose, it is desired to use a distance measure from obstacle:

$$|d| = \sqrt{(X_{iout} - X_{ob})^2 + (Y_{iout} - Y_{ob})^2}\,, \tag{1}$$

where (X_{iout},Y_{iout}) is a current (i-th) position of the UAV (from state vector) and $i=1,\ldots,k$, where k is a simulation horizont. Moreover, (X_{ob},Y_{ob}) describes position of obstacle and r is its radius. Thus formula for the activation threshold of obstacles avoidance mechanism can be used:

$$|d| < r + \Delta\,, \tag{2}$$

where Δ is a desired, minimal flight distance from obstacle. The i-th coordinates of flight (X_i,Y_i) through which the UAV must fly can be described iteratively using following conditions:

$$(X_i, Y_i) = \begin{cases} (X_{ref}, Y_{ref}) & |d| \geq r + \Delta \\ (X_{ref} + \cos\theta, Y_{ref} + \sin\theta) & |d| < r + \Delta \end{cases}, \tag{3}$$

where θ is a desired angle of deviation of the flight direction from original, set direction of flight (for a non-avoiding of obstacles version), and (X_{ref},Y_{ref}) is a flight destination point.

b) Autonomous mode of soft landing

Upon reaching the last point of the reference path it is proposed that the quadrotor should return to the starting point with the soft landing, according to the formula:

$$Z_i = \frac{Z_0}{t_i^2 + 1} + \Gamma\,, \tag{4}$$

where Z_i – altitude (above the level of the Earth) in i-th time step from the start of landing procedure, Z_0 – initial altitude (the last point of the mission), t_i – accumulated time from the start of landing procedure to i-th moment of time, Γ – altitude on which the UAV has to finish the soft landing (optional parameter for soft landings, e.g., on environment elements that are height different from the starting point).

The example results, obtained during the test of autonomous, soft landing for *Crazyflie 2.0* model from [6], have been presented in Fig. 7.

V. Position Control via Pointing Device

The last example of the *Crazyflie 2.0* use as a platform for research and education, is a proposal for approach to synthesis of the autonomous tracking system of reference paths by flying robot, where the track is set by pointing device/positioner (see Figure 9). The main difference between solution from [20] and presented in this paper is that instead of using an expensive motion capture system (with many video cameras) like *Vicon* or *OptiTrack*, only the single, low cost motion sensing input device, i.e., *Kinect* (from *Microsoft Xbox 360* games console) has been used for current pointing device position estimation.

In the first stage [4] of research presented here, the use of *Crazyflie 2.0* with attached marker (green ping-pong ball) has been proposed to determine the current position of robot in 3D space. To determine the orientation, only on-board sensory data, transmitted to the ground station (see Fig. 8), are used. Balls illuminated with LEDs in blue color (1 ball) and red (2 balls) are used for the construction of positioner (see Fig. 9).

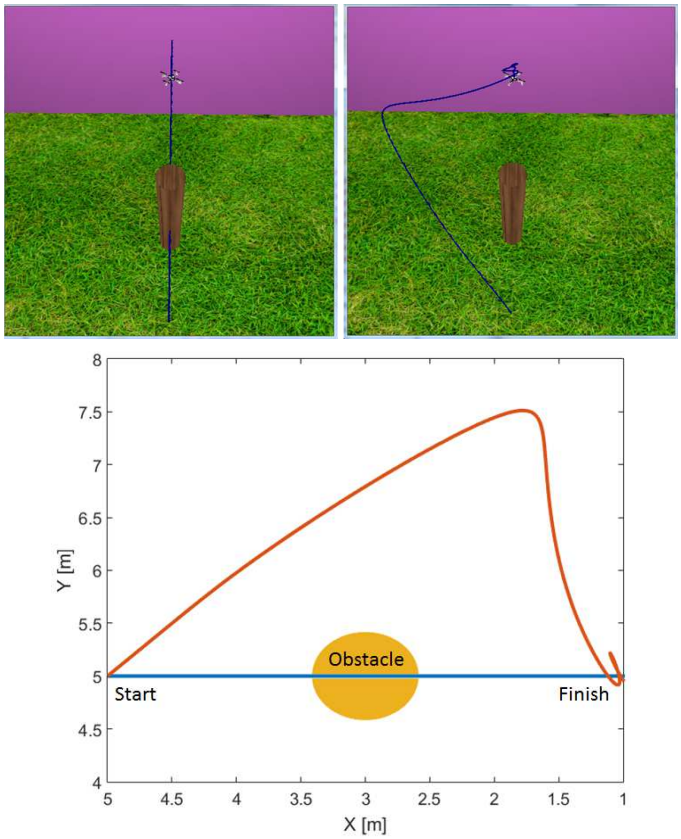

Fig. 6. Exemplary test results with *Crazyflie 2.0* model [6] implemented in *4FLY Simulator*: flight without obstacles avoidance algorithm (left figure) and with (right figure). Recorded flight paths in *X-Y* surface (below)

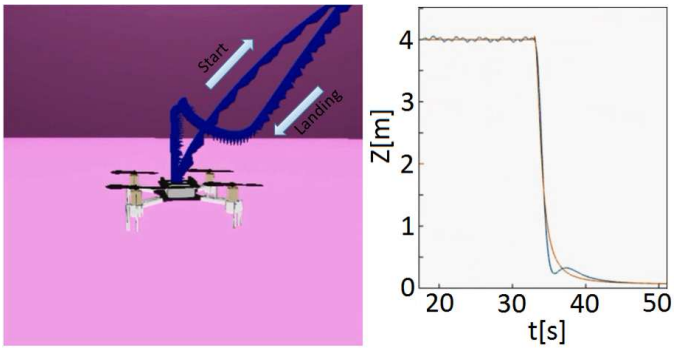

Fig. 7. Test result with *Crazyflie 2.0* model [6] impemented in *4FLY Simulator*: autonomous, soft landing (left figure – zoom on robot) and its time course in the *Z* axis (right figure)

Fixed distances between them are preserved, allowing based on the RGB image processing from the *Kinect* explicitly designate the set reference changes of position and orientation in 3D space (through which the UAV should move). In [4], a detailed description of the algorithm implemented in *MATLAB*® for image processing is presented. This solution basis on the isolation of red, blue and green colors from pictures, filtering of images using the median filter, further binarization, and recognition of extracted objects (their centers). In the mentioned thesis the necessary mathematical formulas, which enable to assign the position of markers on the binary image to their real life counterparts, are given.

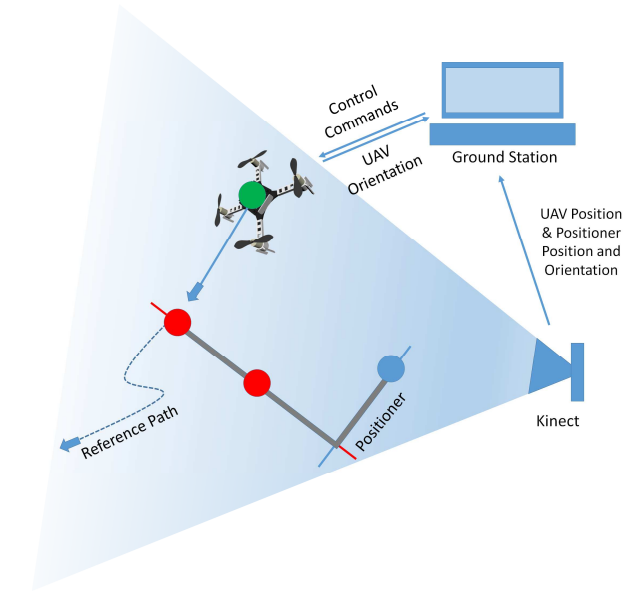

Fig. 8. Control framework

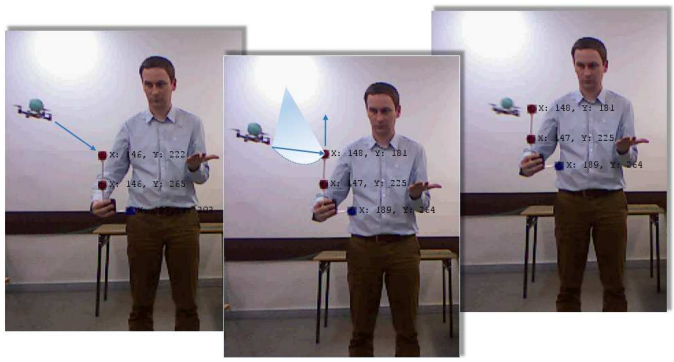

Fig. 9. *Crazyflie 2.0* position control using pointing device (tracking based on the objects recognition by the use of *Kinect* sensor)

The processed image of the scene is an input for the calculation software with the implemented control algorithm for the UAV autonomous flight after the path drawn by positioner. PID controllers are used to correct the global position in *X,Y,Z* axes in cascade control system, while the default *Crazyflie 2.0* controllers (see Table 1) allow to correct its orientation. The communication between the UAV and ground station is achieved using the *CrazyflieDotNet* library [21] – written in *C#* for the first generation of *Crazyflie* UAV. The software code has been changed [4]. It was desirable that the data transmitted to the robot (control) were not set, but automatically retrieved from the *MATLAB*® processed images recorded with *Kinect*. In the first phase of tests it was decided to limit the control of robot's global position in *Z* axis (rotational speeds of its drive units) and to control the orientation around the *X* axis (*roll* angle), without the use of support from the depth sensor. Preliminary results may be found in the *MMD database* webpage [18].

As it has been shown in [7, 8], multilayer UAV control architecture generally requires the use of a software-based speed controller due to an unregulated power supply used for propulsion units of the *Crazyflie 2.0*. The torques and trusts

produced by the DC motors generally do not perfectly reflect the commands (duty cycles U_D), which are sent to the hardware. This fact is a problem in position tracking efficiency (obtaining reliable thrust control). To solve it the feedback from the recent measured battery voltage (V_{actual}) is adapted. Thus one can use formula for command input to the motors (as a function of V_{actual} and desired angular velocities of the motors):

$$U_D = \frac{V_{\max}}{V_{actual}} \sqrt{\omega^2 - \beta} + \alpha, \tag{5}$$

where V_{max} is the nominal voltage of the battery, α – the minimum duty cycle that must be sent to the hardware in order to get any angular velocity ω at the motors. The parameter β accounts for the fact that the propellers start out at the certain non-zero velocity [7].

VI. Conclusion

In this paper the *Crazyflie 2.0* flying robot was introduced as the platform for research and education in robotics and control engineering. The state of the art and the references to most valuable papers regarding this quadrotor were provided – i.e., to describe its potential use, the modeling of the UAV dynamics, the available control architectures, *etc*. The *Crazyflie 2.0* hardware and software were also briefly discussed. On that background, three related projects were described. Based the first – to enhance the potential use of robot in education and research, one presented a freely available *Crazyflie Java Client* software package, which allows among others the flight data acquisition. The second project outlined the capabilities of the *4FLY Simulator* as the tool developed mostly for designing and initial testing of the *Crazyflie 2.0* control system with a variety of controller types. For example, the two selected test results were provided for proposed mechanisms of the soft landing and the obstacles avoidance. In the last example, one showed some preliminary results, how this nano quadrotor can be used in the context of vision-based control.

The presented experiments in which the flying robot has been utilized, highlighted interesting further research perspective and educational usage. The proposed *4FLY Simulator* can be a starting point in the development of swarms intelligence algorithms for missions of several *Crazyflie 2.0* UAVs before tests with real robots. A recorded data from the *CJC* can be useful to simulate real flight conditions of particular swarm. Moreover, by the use of the motion capture algorithms, one may use positioner as "the leader" to introduce the reference path for the swarm, which is flying with mechanisms of collisions and obstacles avoidance. In education, the UAV can be very useful in projects for students studying subjects like: basics of dynamical modeling and control, real-time systems, sensors and vision systems, automatic control, embedded control systems, signal processing and many other.

Acknowledgment

The authors are grateful to engineers: Mr Sebastian Korcz and Mr Adrian Olbrzymek for the development and the implementation of the solutions presented in Section 5 (used there in order to show the diverse of the applications in *Crazyflie 2.0* flying robot).

References

[1] P. Parada, T. Espinoza, and A. Dzul, "Nonlinear observers applied to fixed-wing UAVs", Proceedings of the 2014 International Conference on Unmanned Aerial Systems (ICUAS), pp. 780-790, May 2014.

[2] P. Castillo, R. Lozano, and A. Dzul, Modelling and Control of Mini-Flying Machines, Springer-Verlag, London, 2005.

[3] T. Krajnik, V. Vonasek, D. Fiser, and J. Faigl, "AR-Drone as a Platform for Robotic Research and Education", Proceedings of the Research and Education in Robotics – EUROBOT 2011, pp. 172-186, 2011.

[4] S. Korcz, and A. Olbrzymek, Multirotor Flying Robot Control in Relation to Positioner, B.Sc. Thesis in polish, Poznan University of Technology, 2017.

[5] J. Forster, System identification of the Crazyflie 2.0 Nano Quadrocopter, Bachelor Thesis, Institute for Dynamic Systems and Control, Swiss Federal Institute of Technology (ETH), Zurich, 2015.

[6] C. Luis, Design of a Trajectory Tracking Controller for a nano-quadcopter, Techn. Report, Ecole Polytechnique de Montreal, 2016.

[7] B. Landry, Planning and Control for Quadrotor Flight through Cluttered Environments, Master Thesis, MIT Institute of Technology, 2014.

[8] B. Galea, E. Kia, N. Aird, and P.G. Kry, "Stippling with aerial robots", Computational Aesthetics in Graphics, Visualization, and Imaging, online, 2016.

[9] L. Campos-Macias, D. Gomez-Gutierrez, R. Aldana-Lopez, R. de la Guardia, and J.I. Parra Vilchis, "A Hybrid Method for Online Trajectory Planning of Mobile Robots in Cluttered Environments", IEEE Robotics and Automation Letters, DOI: 10.1109/LRA.2017.2655145, 2017.

[10] J.A. Preiss, W. Honig, G.S. Sukhatme, and N. Ayanian, "Crazyswarm: A Large Nano-Quadcopter Swarm", Extended Abstract for IROS 2016.

[11] W. Honig, Ch. Milanes, L. Scaria, T. Phan, M. Bolas, and N. Ayanian, "Mixed Reality for Robotics", 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5382-5387, 2015.

[12] Bitcraze Wiki, https://wiki.bitcraze.io/, access: 6.02.2017.

[13] Bitcraze GitHub, https://github.com/bitcraze, access: 6.02.2017.

[14] Bitcraze Forums, The Bitcraze web forum, https://forum.bitcraze.io/.

[15] S. Bouabdallah, and R. Siegwart, "Full Control of a Quadrotor", 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 153-158, 2007.

[16] Java library https://github.com/fredg02/se.bitcraze.crazyflie.lib for Crazyflie 2.0 (Bitcraze GitHub), access: 6.02.2017.

[17] P. Wroński, The application for acquisition of flight data from Crazyflie 2.0 multirotor flying platform, B.Sc. Thesis in polish, Poznan University of Technology, 2017.

[18] W. Giernacki, D. Horla, and T. Sadalla, "Mathematical Models Database (MMD ver. 1.0). Non-Commercial Proposal for Researchers", Proceedings of the 21st International Conference on Methods and Models in Automation and Robotics (MMAR), pp. 555-558, 2016, DOI: 10.1109/MMAR.2016.7575196. http://mathematicalmodels.put.poznan.pl

[19] M. Skwierczyński, and W. Witwicki, Simulation environment in OpenGL for control purposes of a multi-rotor flying platfrom, B.Sc. Thesis in polish, Poznan University of Technology, 2017.

[20] Crazyflie – Position Control, University of Augsburg, https://www.youtube.com/watch?v=QjxF9zUlIx0, access: 6.02.2017.

[21] Ch. Karcz, CrazyflieDotNet, https://github.com/ckarcz/CrazyflieDotNet, access: 6.02.2017.