Design and Implementation of A Mini Quadrotor Control System in GPS Denied Environments

Chang Liu[1], *Member, IEEE* and Stephen D. Prior[1]

***Abstract*— This paper presents the design of quadrotor control architecture, based on crowd-sourcing electronics. The aim of this quadrotor is to provide a test-bed for vision-based autonomous navigation system in GPS denied environments. The control architecture consists of a cascaded structure, where an attitude controller nested in velocity and altitude controllers. The sub-controllers are all linear controllers with feedforward term to linearize the quadrotor dynamics. The control and sensor fusion algorithm is developed under Arduino compatible open source electronics, whereas the complete design also includes an additional downward facing optical flow sensor (PX4FLOW camera) for horizontal velocity estimation and vehicle altitude estimation, and a separate Linux embedded computer (Odriod-U3) for future Simultaneous Localization And Mapping (SLAM) vision algorithm development. In current stage, by utilizing the PX4FLOW sensor, it is capable of horizontal velocity control and altitude hold. Besides, a ground station GUI software is developed in MATLAB® for two-way telemetry visualization and in-air parameter tuning.**

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) are being considered in an increasing number of defence-related applications, for the purpose of reducing downside risk and rising confidence in mission success, however, the civilian market is predicted to rapidly expand over the next decade [1]. Quadrotor is one of the most popular subset of UAVs. Because of its agile manoeuvrability, as well as ability of vertical take-off and landing (VTOL) and stable hovering, it is commonly agreed to be an ideal candidate for search and rescue, surveillance, exploration, agriculture, monitoring and military applications in both indoor and outdoor environments.

Over the last decades, Global Positioning System (GPS) has been the key to enabling the autonomy of UAVs. It provides global localization service with the best accuracy of 1-2 metres. However, Recently, due to the proven weakness of GPS signal and rapid development of onboard sensing and computation capability, there has been growing interest in developing and researching alternative navigation methods for UAVs in GPS denied environments [2]–[8]. The successful implementations will not only improve system robustness under GPS failure, but also enable a new range of applications out of GPS coverage.

A mini quadrotor is defined by [1] to carry under 2 kg payload, which is sufficient for light weight perception sensors (such as cameras, laser scanner, radar and ultrasonic sensor) and embedded computer, which are essential for an autonomous navigation. Additionally, because they are low cost, easy to maintain, and safe to operate, these makes them very good test-beds for research and development [9]. The most popular research platform in this category is Hummingbird quadrotor sold by Ascending Technologies [10], and the state-of-the-art quadrotor autonomous control theory is described in [9]. Thus, extensive research has been conducted on mini quadrotor development [11]–[14].

The vision-based method is believed, in this paper, to be the optimal sensor for navigation. The reason is that a camera has significant advantages over other sensors, such as low mass, low power consumption, low price, adjustable field of view (FOV), high accuracy, additional colour information and long range. Especially in the past five years, world top research institutes paid high attention to developing advanced visual-based simultaneous localization and mapping (vSLAM) algorithms based on structure from motion (SFM) theory [15]–[23], which are suitable to modern onboard embedded computer. Moreover, the visual scale problem, which was the main challenge of involving vision in control loop, is addressed to various extent by fusing onboard inertial measurements (accelerometer and gyroscope), which is named visual inertial navigation system (VINS) [6], [24]–[30]. However, it still has limitations, since visual signal generally requires high computation to percept, relies on visual features in the scene, and is sensitive to lighting conditions. Therefore, this mini quadrotor, as shown in Fig. 1, is designed and implemented aiming to provide a test-bed

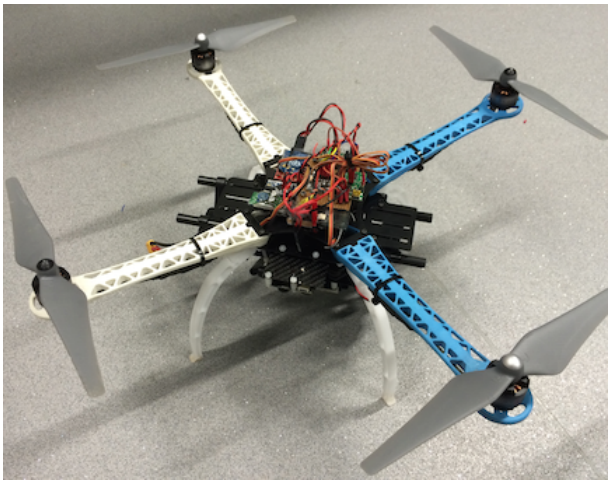


Fig. 1: The developed quadrotor

*This work was supported by Faculty of Engineering and the Environment, University of Southampton

[1]Chang Liu (cl21g11@soton.ac.uk) and Stephen D. Prior (S.D.Prior@soton.ac.uk) are with Faculty of Engineering and the Environment, University of Southampton, Southampton, SO17 1BJ, UK.

for developing similar algorithms in the near future.

The rest of this paper is formed as follows: in Section II, it explains the modelling of the quadrotor dynamics, and Section III describes the control architecture design based on the dynamic model. Then, Section IV summarizes quadrotor implementation details, and then, Section V shows the test data to demonstrate the system performance. Lastly, Section VI concludes and proposes future works.

II. QUADROTOR DYNAMICS MODELLING

This section presents the nonlinear dynamic model of the mini quadrotor, which formes the basis for the controller synthesis in Section III.

The coordinate frames and system setup is indicated in Fig. 2. Quadrotor body frame is fixed to the quadrotor body with X_b–axis pointing forward, Y_b–axis pointing right, Y_b–axis pointing down. World frame is fixed to the world, and is defined to have the same origin with body frame at the moment when the quadrotor connects to a battery. Z_w–axis points to the direction of gravity. The angles defined in the system follow the right hand rule. Fig. 2 also shows that the quadrotor has the cross configuration and four motors are numbered 1–4, with spinning directions as indicated.

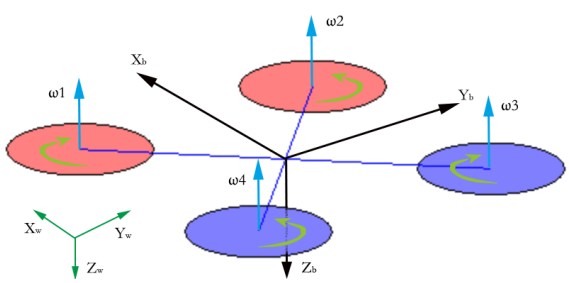


Fig. 2: Coordinate system and quadrotor setup.

The overview of the nonlinear model is shown in Fig. 3, where inputs of the model, δ_n, are the normalized pulse width modulation (PWM) command signal to the electronic speed controller (ESC) of motors, and outputs of the model are 3 dimensional position vector $\mathbf{P_w}$ $(= (x, y, z)^\top)$ in world frame and Euler angles vector $\mathbf{\Theta}$ $(= (\phi, \theta, \psi)^\top)$, in aerospace sequence $(\psi \rightarrow \theta \rightarrow \phi)$. The following subsections will explain the included components individually.

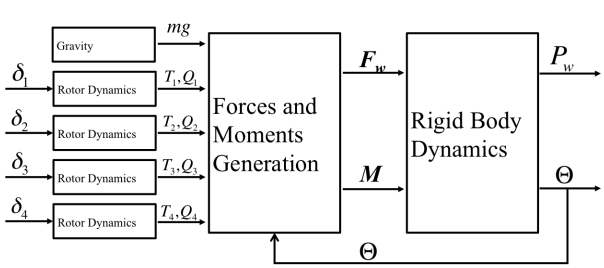


Fig. 3: Overview of quadrotor dynamics model.

A. Rotor Dynamics

The propulsion system of the quadrotor includes two pairs of counter-rotating ESC-motor-propeller systems. The dynamics of the four systems are identical and are approximated by the rotor dynamics model. The model receives the normalized PWM command δ and outputs thrust, T, in g and torque, Q, in Nm. If a propeller, with diameter D, rotates at ω angular velocity in free air, whose density is ρ, the thrust T and torque Q that it produces can be modelled as:

$$T = \rho\omega^2 D^4 C_T \tag{1}$$

$$Q = \rho\omega^2 D^5 C_Q, \tag{2}$$

where C_T and C_Q are thrust and torque coefficients respectively, which depend on propeller geometry and profile. Furthermore, by assuming an ideal ESC-motor system, which spins the propeller at the angular velocity that is linear to the PWM pulse width command, u, in unit μs, without mechanical delay:

$$\omega = ku - c, \tag{3}$$

where k and c are constants. Note that in our implementation, we use normalized PWM, δ, to command ESC. The PWM command signal is normalised by:

$$u = \frac{1856}{180}\delta + 544, \tag{4}$$

which maps 544–1856 μs PWM to 0–180. This corresponds to standard servo angle position.

Therefore, to model the propulsion system, we substitute (4), (3) into (1), (2). However, in practice, to model the propulsion system with given parameters (ρ, D, C_T and C_Q), the derived equations can be simplified as:

$$T = c_T(\delta - c_o)^2, \tag{5}$$

$$Q = c_Q T, \tag{6}$$

where c_T, c_Q and c_o are constants, and c_T and c_o can be easy obtained from bench static thrust test.

B. Force and Moment Generation

All the forces and moments applied to the quadrotor result in movement. As mentioned in [31], they are generally generated by four different sources, i.e., the gravitational force, the rotors thrust and moment, their reaction torques, and their gyroscopic effects. However, the last two have insignificant effect on overall forces and moments, thus we only consider the former two. Therefore, this module converts all the forces and moments into a 3 dimensional force vector, $\mathbf{F_w}$ in world frame, and a 3 dimensional moment vector, $\mathbf{M}$ in body frame.

The gravitational force in world frame only applies to positive Z_w axis, which yields:

$$\mathbf{F_{gravity}} = \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix}, \tag{7}$$

where m is quadrotor mass and g is standard gravitational acceleration.

Besides, each of the four rotors on the quadrotor generates thrust, T_n, and torque, Q_n, where $n = 1, 2, 3, 4$ (the numbering order is indicated in Fig. 2). The force generated by the four rotors applies to the negative Z_b axis in body frame. By rotating it to world frame, we get:

$$\mathbf{F_{thrust}} = \mathbf{R_{w/b}} \begin{pmatrix} 0 \\ 0 \\ -(T_1 + T_2 + T_3 + T_4) \end{pmatrix}, \tag{8}$$

where $\mathbf{R_{w/b}}$ is the rotation matrix, which rotates vector from body frame to world frame, which is defined as:

$$\mathbf{R_{w/b}} = \begin{bmatrix} c_\theta c_\psi & -c_\theta s_\psi & s_\theta \\ s_\phi s_\theta c_\psi + c_\phi s_\psi & -s_\phi s_\theta s_\psi + c_\phi c_\psi & -s_\phi c_\theta \\ -c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi + s_\phi c_\psi & c_\phi c_\theta \end{bmatrix}, \tag{9}$$

where $s_\star = sin(\star), c_\star = cos(\star)$, and ϕ, θ, ψ can be obtained from the feedback Euler angles vector, Θ.

Therefore, the total force applied onto the quadrotor in world frame can be derived as:

$$\mathbf{F_w} = \mathbf{F_{gravity}} + \mathbf{F_{thrust}} = \begin{pmatrix} F_{wx} \\ F_{wy} \\ F_{wz} \end{pmatrix}, \tag{10}$$

where:

$$F_{wx} = -s_\theta T_{total}, \tag{11}$$
$$F_{wy} = s_\phi c_\theta T_{total}, \tag{12}$$
$$F_{wz} = mg - c_\phi c_\theta T_{total}, \tag{13}$$
$$T_{total} = T_1 + T_2 + T_3 + T_4, \tag{14}$$

and where T_{total} is the total thrust provided by the four rotors.

The other output from the module is the 3 dimensional moment vector in body frame, which is approximated in this paper to be generate by the thrusts and torques of the four rotors. Roll moment is contributed by the thrust difference between rotors 1, 4 and 2, 3. Pitch moment is contributed by the thrust difference between rotors 1, 2 and 3, 4. Yaw moment is contributed by the torque difference between rotors 1, 3 and 2, 4. Thus, by also substituting (6), it then can be formulated as:

$$\mathbf{M} = \mathbf{BCT}, \tag{15}$$

where:

$$\mathbf{B} = \begin{bmatrix} \frac{\sqrt{2}}{2}l & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2}l & 0 \\ 0 & 0 & c_Q \end{bmatrix}, \tag{16}$$

$$\mathbf{C} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix}, \tag{17}$$

$$\mathbf{T} = (T_1, T_2, T_3, T_4)^\top, \tag{18}$$

The thrust vector, $\mathbf{T}$, represents the thrusts generated by the four rotors, and l is the distance between rotors and quadrotor centre of mass.

C. Rigid-body Dynamics

Rigid-body dynamics formularises the translational and rotational dynamics of the quadrotor, by utilising the Newton-Euler formalism. Therefore, the resulting 3 dimensional position vector, $\mathbf{P_w}$ in world frame, and 3 dimensional Euler angle vector, $\mathbf{\Theta}$, can be obtained as:

$$m\ddot{\mathbf{P}}_\mathbf{w} = \mathbf{F_w}, \tag{19}$$
$$\mathbf{J}\ddot{\mathbf{\Theta}} + \dot{\mathbf{\Theta}} \times (\mathbf{J}\dot{\mathbf{\Theta}}) = \mathbf{M}, \tag{20}$$

where $\mathbf{J}$ is the inertia matrix of the quadrotor, and since our quadrotor is approximately four way symmetrical, $\mathbf{J}$ is assumed to be a diagonal matrix.

III. CONTROLLER DESIGN

Base on the dynamics model developed in the last section, a nested controller structure is designed to ultimately control the quadrotor altitude and horizontal velocity. As shown in Fig. 4, three sub-controllers: attitude controller, altitude controller and velocity controller, are developed. It receives user commands, including altitude command, $z^\star$, and horizontal velocity command in world frame, $(\dot{x}^\star, \dot{y}^\star)$, and desired yaw command, $\psi^\star$. And controls the quadrotor accordingly by taking the feedback measurement from inertial measurement unit (IMU), altitude and velocity sensor onboard.

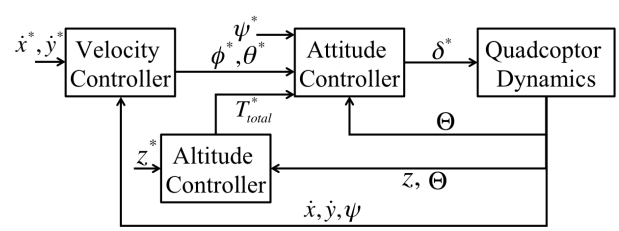


Fig. 4: Overview of nested controller.

A. Attitude Controller

The attitude controller nested inside the altitude and velocity controllers, it receives desired roll $\phi^\star$, desired pitch, $\theta^\star$, and total thrust command, $T^\star_{total}$, from them, as well as the desired yaw command, $\psi^\star$, from the user. Then with the help of attitude feedback measurement, $\mathbf{\Theta}$, from IMU, it commands normalised PWM signals, $\delta^\star_n$, to the four ESCs. It is designed to minimise the attitude error, $\mathbf{e}_\theta$, while maintaining the total thrust, T_{total}, as commanded. The attitude error, $\mathbf{e}_\theta$, is defined as:

$$\mathbf{e}_\theta = \mathbf{\Theta}^\star - \mathbf{\Theta} = \begin{pmatrix} \Delta\phi \\ \Delta\theta \\ \Delta\psi \end{pmatrix}, \tag{21}$$

where desired attitude is $\mathbf{\Theta}^\star = (\phi^\star, \theta^\star, \psi^\star)^\top$ and angle deviations are $\Delta\phi = \phi^\star - \phi$, $\Delta\theta = \theta^\star - \theta$, $\Delta\psi = \psi^\star - \psi$. This expression of error guarantees stability only when angle deviations are small [9]. A more sophisticated error expression will be implemented in future work.

Then, we can apply the proportional, integral and derivative error to obtain the PID control law, and attitude correction vector, $\mathbf{u}_\theta$, can be computed by:

$$\mathbf{u}_\theta = \mathbf{K_p}\mathbf{e}_\theta + \mathbf{K_i}\int \mathbf{e}_\theta \, \mathrm{d}\mathbf{t} + \mathbf{K_d}\dot{\mathbf{e}}_\theta, \tag{22}$$

where gain vectors, $\mathbf{K}_\star = (k_{\star\phi}, k_{\star\theta}, k_{\star\psi})^\top$, can be tuned, depending on the aggressiveness of the required manoeuvre.

This correction vector must be linear to the moment, that we should apply to quadrotor to compensate the attitude error. Therefore, we can define the moment command to quadrotor, $\mathbf{M}^\star$, is:

$$\mathbf{M}^\star = 4\mathbf{B}\mathbf{u}_\theta, \tag{23}$$

where $\mathbf{B}$ is defined in (16). And then we add total thrust control. Thus, by combining (14) and (15) we can say:

$$\begin{bmatrix} T^\star_{total} \\ 4\mathbf{u}_\theta \end{bmatrix} = \begin{bmatrix} \mathbf{1_{1\times 4}} \\ \mathbf{C} \end{bmatrix} \mathbf{T}^\star, \tag{24}$$

where matrix $\mathbf{C}$ is defined in (17), and $\mathbf{T}^\star = (T^\star_1, T^\star_2, T^\star_3, T^\star_4)^\top$ is desired thrust vector, which represents the desired thrust command to each rotor. Therefore, the thrust command for individual rotors can be computed by reversing (24), :

$$\mathbf{T}^\star = \begin{bmatrix} \mathbf{1_{4\times 1}} & \mathbf{C}^\top \end{bmatrix} \begin{bmatrix} T^\star_{total}/4 \\ \mathbf{u}_\theta \end{bmatrix}, \tag{25}$$

This expression of thrust commands not only applies the desired moment to quadrotor, but also ensures the total thrust provided by the four rotors is equal to $T^\star_{total}$.

Finally, to generate the normalised PWM signal command, $\delta^\star_n$, for individual rotor, simply apply inverted (5) on $T^\star_n$. Then we get:

$$\delta^\star_n = \sqrt{\frac{T^\star_n}{c_T}} + c_o. \tag{26}$$

B. Altitude Controller

The altitude controller receives altitude command, $z^\star$, from the user, altitude measurement, z, from sensor, and attitude measurement, $\mathbf{\Theta}$, from IMU. And then It minimises altitude error, e_z, by varying the total thrust command, $T^\star_{total}$. The altitude error is defined as:

$$e_z = z^\star - z. \tag{27}$$

Then, one can apply PID control law, which yields the altitude correction value, u_z:

$$u_z = K_{pz}e_z + K_{iz}\int e_z \, \mathrm{d}t + K_{dz}\dot{e_z}, \tag{28}$$

where K_{pz}, K_{iz}, K_{dz} are gains to be tuned for altitude controller.

This correction value must be linear to the acceleration command in Z_w–axis, $\ddot{z}^\star$ to compensate altitude error. Thus we can define:

$$\ddot{z}^\star = \frac{u_z}{m}. \tag{29}$$

Therefore, based on (19) and (13), the total thrust command, $T^\star_{total}$, can be computed by:

$$T^\star_{total} = \frac{mg - u_z}{c_\phi c_\theta}. \tag{30}$$

C. Velocity Controller

The velocity controller minimises the error between the measured 2 dimensional velocity, $\mathbf{v} = (\dot{x}, \dot{y})^\top$, and user velocity command in world frame, $\mathbf{v}^\star = (\dot{x}^\star, \dot{y}^\star)^\top$, and then, also by referencing the yaw angle, ψ, reading from IMU, it gives desired roll and pitch command, $(\phi^\star, \theta^\star)^\top$, to attitude controller. The velocity error, $\mathbf{e_v}$, is defined as:

$$\mathbf{e_v} = \mathbf{v}^\star - \mathbf{v}. \tag{31}$$

Then we apply PID control law, which yields the velocity correction vector, $\mathbf{u_v}$:

$$\mathbf{u_v} = K_{pv}\mathbf{e_v} + K_{iv}\int \mathbf{e_v} \, \mathrm{d}t + K_{dv}\dot{\mathbf{e_v}}, \tag{32}$$

where K_{pv}, K_{iv}, K_{dv} are gains to be tuned for altitude controller.

This correction value must be linear to the commanded acceleration in X_w–axis and Y_w–axis to compensate velocity error. Thus we can define:

$$\dot{\mathbf{v}}^\star = \frac{\mathbf{u_v}}{m}. \tag{33}$$

Therefore, based on (19), (11) and (12), we get:

$$\mathbf{u_v} = \begin{pmatrix} u_{vx} \\ u_{vy} \end{pmatrix} = T^\star_{total} \begin{pmatrix} -s^\star_\theta \\ s^\star_\phi c^\star_\theta \end{pmatrix}, \tag{34}$$

where $T^\star_{total}$ has been computed from the altitude controller above. Then we solve this equation as:

$$\phi^\star = sin^{-1}(\frac{u_{vy}}{\sqrt{1 - u^2_{vx}}}), \tag{35}$$

$$\theta^\star = -sin^{-1}(u_{vx}). \tag{36}$$

IV. IMPLEMENTATIONS

This section summarises the implementation details for the working quadrotor system, including mechanical setup, and autopilot electronics and software description. The quadrotor basic details are summarised in Table. I.

TABLE I: Quadrotor basic details

Quantity name	Value	Unit
Total mass	1158	g
Length from motor to center of mass	250	mm
Propeller size	9.4×4.3	$inch$
Motor Kv	920	RPM/V

A. Chassis

We selected S500 glass-fiber quadrotor frame[1] for industrial standard design suitable to fast prototyping applications. The 500 mm arm span also gives a wide range of suitable propeller sizes (9–12 inch).

B. Propulsion system

We selected DJI E300 tuned propulsion system[2] as our off-the-shelf solution. We then did bench static thrust test on the system and obtained c_T and c_o in (5) by conducting linear regression on square root of T versus δ. The obtained c_o is 48.1385 and c_T is 0.09376, which gives 95% accuracy, as shown in Fig. 5, where the blue curve is the measured data from thrust test, and the red curve is generated from (5) where c_T and c_o have above values.

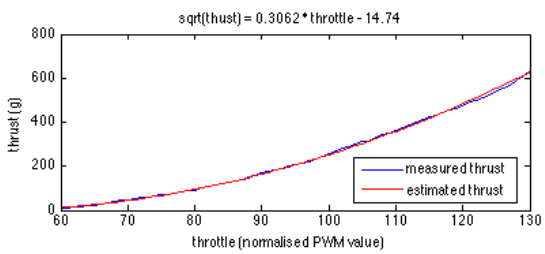
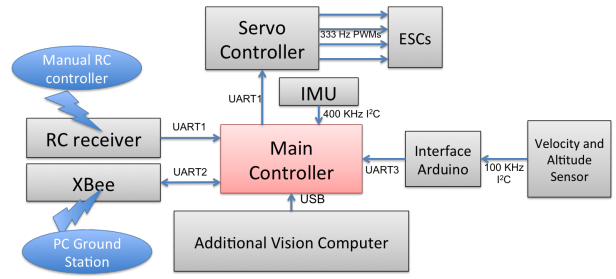


Fig. 5: Propulsion system model.

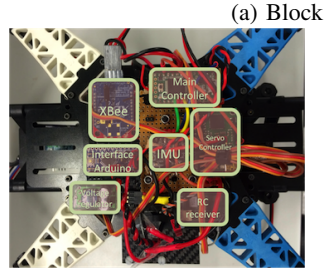
C. Flight Control Implementation

Thanks to the high speed Teensy 3.1 processor and a dedicated servo controller, the attitude control loop, including IMU sensor fusion, PID control law and PWM signal update, executes within 3 ms. As well as the altitude and velocity controller executes within 10 ms. The physical layout is shown in Fig. 6b and Fig. 6c. And the block diagram in Fig. 6a shows the interactions between components.

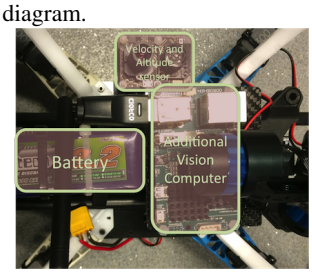
- **Main Controller Board** is based on Teensy 3.1 board[3]. It is an ARM based Arduino compatible development board, which features very small form factor (35×18 mm) and fast processor (ARM Cortex–M4 with up to 96 MHz clock speed). It is ideal for a flight controller.
- **Servo Controller** is based on Pololu Mini Maestro Servo Controller board[4]. It is a dedicated servo controller board, which features high resolution (0.25 μs) servo PWM output to 12 channels, with update rate up to 333 kHz, and the fast UART Serial protocol makes it easy to receive command from main controller board.
- **Inertial Measurement Unit** is based on FreeIMU open source project [32], which not only provides a small (21×22 mm) 10 degree-of-freedom sensor board, but also published an open source IMU sensor fusion library for Arduino environment, based on a nonlinear complementary filter [33]. Recently, the updated library[5] (with Teensy support) has been published and is used in this paper.
- **Altitude and Velocity Sensor** is based on PX4FLOW camera[6] [34]. It employs an ultrasonic sensor to sense flight altitude by measuring ground distance within 5 m, and at the same time thanks to the onboard CMOS high speed vision sensor, it measures the optical flow at 250 Hz. Then it obtains the ground velocity relative to quadrotor by fusing optical flow, ground distance. Moreover, by subtracting the scaled gyroscope rate, it compensates the optical flow caused by roll and pitch rotation. Note that the sensed velocity is relative to the quadrotor, so the sensed velocity needs to be pre-rotated to world frame before (31).



(a) Block diagram.



(b) Top down view.



(c) Bottom up view.

Fig. 6: System layout.

D. Ground Station Software

A ground control station graphical user interface (GUI) software is develop in MATLAB® along with the quadrotor development, for the purpose of monitoring real time sensor measurement, in-air parameter tuning, flight data logging and post-processing. The wireless communication is realised by XBee low power RF module as shown in Fig. 6a and Fig. 6b. A customised bidirectional serial protocol is developed for reliable transmission and minimised data package. Two frame bytes are used at beginning and end of the package, also one byte checksum is used for each data package. A hand-shake procedure is followed for in-air parameter tuning, so that data transmission only happens when updating new parameters.

[1] http://www.hobbyking.co.uk/hobbyking/store/__57154__S500_Glass_Fiber_Quadcopter_Frame_with_PDB_480mm.html
[2] http://www.dji.com/product/tuned-propulsion-system
[3] https://www.pjrc.com/teensy/teensy31.html
[4] http://www.pololu.com/product/1352
[5] https://github.com/mjs513/FreeIMU-Updates
[6] https://pixhawk.org/modules/px4flow

E. Additional Onboard Vision Computer

A medium level embedded computer is installed onboard for future high level vision algorithm development, which is mainly focuses on vision based Simultaneous Localization and Mapping (SLAM) solutions. The Odroid–U3[7] embedded computer is selected as the vision computer as shown in Fig. 6c. It features small form factor (83×48 mm), light weight (48 g), 1.7 GHz Quad-Core processor with latest linux support. Robotic Operating System (ROS) can be easily installed in it, which allows reuse of the code from other research groups.

V. TEST RESULTS

Outdoor flight test results are shown below to demonstrate the control performance and validate the theory. Note that the weather forecast states 8 mph wind speed at the time of testing. The manual tuning was conducted in advance of this trial and the following tuning parameters in Table. II was used in this trial.

TABLE II: PID parameters

Controller	K_p	K_i	K_d
Roll and Pitch	4.7	2	0.79
Yaw	20	0	6
Altitude	90	10	200
X and Y Velocity	1.4	0.3	0.3

Two sessions was carried out in this test: The first session tests the attitude controller only, where the user control signals are feed directly to the attitude controller as angle commands through RC transmitter. The obtained results are indicated in Fig. 7, which shows the command-responds graph of roll, pitch, yaw angle individually. The second session enables all three controllers, with user commanding horizontal velocity, altitude and yaw angle through RC transmitter. The obtained results are indicated in Fig. 8, which shows the velocity command-response in Y_w–axis in the first sub-plot, altitude command-response in the third sub-plot and yaw hold in the forth sub-plot. An additional second sub-plot shows the intermediate command-response signal between velocity controller and attitude controller.

VI. CONCLUSIONS AND FUTURE WORK

This paper has shown the quadrotor modelling and controller design principle, as well as implementation details. The flight test result showed a good attitude and altitude hold and an acceptable velocity control performance, even with significant wind. The cascaded control architecture of the developed quadrotor is suitable for testing vision based localization algorithms, and testing new control strategies. The fully customised design makes it easy to integrate new sensors and manipulating controller.

The future work includes: implement more sophisticated error vector expression for attitude controller; design and manufacture customised printed circuit board (PCB) to interface all the onboard components; develop and test vision based localization solutions.

[7] http://www.hardkernel.com/main/products/prdt_info.php?g_code=g138745696275

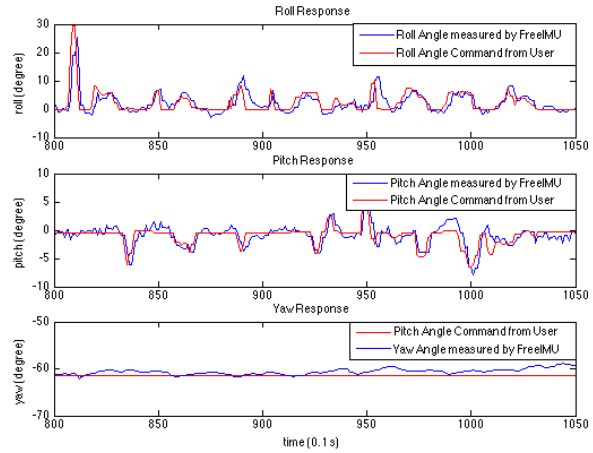


Fig. 7: Attitude command-response graph with manual control.

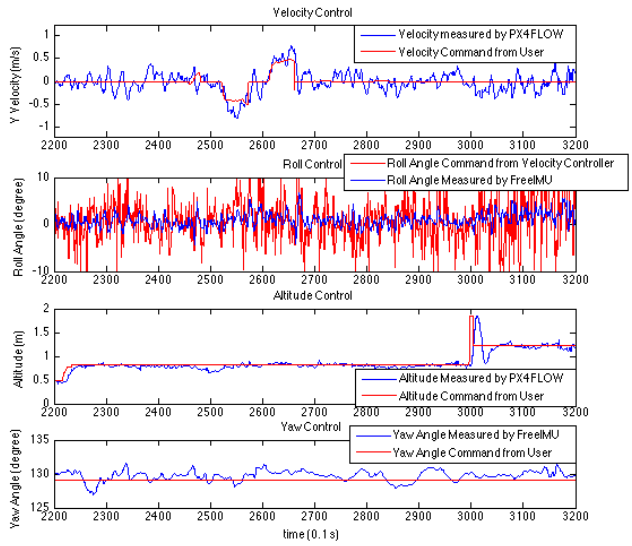


Fig. 8: velocity and altitude command-response graph.

ACKNOWLEDGMENT

I would like to thank Mantas Brazinskas and Mehmet Ali Erbil for their continued technical support and encouragement. I offer my sincere appreciation for the learning opportunities provided by them.

REFERENCES

[1] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, 2012.

[2] A. Bachrach and S. Prentice, "RANGERobust autonomous navigation in GPSdenied environments," *Journal of Field Robotics*, 2011.

[3] A. Bry, "State estimation for aggressive flight in GPS-denied environments using onboard sensing," *International Conference on Robotics and Automation*, no. Icra, pp. 1–8, May 2012.

[4] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," *Intelligent Robots and Systems*, pp. 2815–2821, Oct. 2012.

[5] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-Based State Estimation and Trajectory Control Towards High-Speed Flight with a Quadrotor." *Robotics: Science and Systems*, 2013.

[6] E. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *The International Journal of Robotics*, pp. 1–38, 2011.

[7] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular SLAM based navigation for autonomous micro helicopters in GPS-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[8] M. W. Achtelik, S. Lynen, S. Weiss, L. Kneip, M. Chli, and R. Siegwart, "Visual-inertial SLAM for a small helicopter in large outdoor environments," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2651–2652, Oct. 2012.

[9] R. Mahony, V. Kumar, and P. Corke, "Multirotor Aerial Vehicle Modeling, Estimation, and Control of Quadrotor," no. SEPTEMBER, 2012.

[10] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, July 2013.

[11] S. Jeong and S. Jung, "Design, Control, and Implementation of Small Quad-Rotor System Under Practical Limitation of Cost Effectiveness," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 13, no. 4, pp. 324–335, Dec. 2013.

[12] J. Jiang, J. Qi, D. Song, and J. Han, "Control platform design and experiment of a quadrotor," *Control Conference (CCC)*, pp. 2974–2979, 2013.

[13] H. Fernando, A. D. Silva, and M. D. Zoysa, "Modelling , Simulation and Implementation of a Quadrotor UAV," *IEEE International Conference on Industrial and Information Systems (ICIIS)*, pp. 207–212, 2014.

[14] M. Elsamanty, a. Khalifa, M. Fanni, a. Ramadan, and a. Abo-Ismail, "Methodology for identifying quadrotor parameters, attitude estimation and control," *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1343–1348, July 2013.

[15] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1–10, Nov. 2007.

[16] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," *Computer VisionECCV 2014*, pp. 1–16, 2014.

[17] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast Semi-Direct Monocular Visual Odometry," *Proc. IEEE Intl. Conf. on Robotics and Automation*, 2014.

[18] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE : Probabilistic , Monocular Dense Reconstruction in Real Time."

[19] G. Vogiatzis and C. Hernández, "Video-based, real-time multi-view stereo," *Image and Vision Computing*, vol. 29, no. 7, pp. 434–441, June 2011.

[20] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," *2011 International Conference on Computer Vision*, pp. 2320–2327, Nov. 2011.

[21] C. Roussillon, A. Gonzalez, and J. Solà, "RT-SLAM: a generic and real-time visual SLAM implementation," *Computer Vision Systems*, 2011.

[22] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges," *International Joint Conference on Artificial Intelligence*, vol. 18, pp. 1151–1156, 2003.

[23] J. Engel and D. Cremers, "Semi-Dense Visual Odometry for a Monocular Camera," *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[24] J. Kelly and G. S. Sukhatme, "Visual-inertial simultaneous localization, mapping and sensor-to-sensor self-calibration," *2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation - (CIRA)*, pp. 360–368, Dec. 2009.

[25] J. Kelly and G. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *The International Journal of Robotics Research*, 2011.

[26] J. Lobo and J. Dias, "Vision and inertial sensor cooperation using gravity as a vertical reference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, 2003.

[27] O. Dunkley, J. Engel, and D. Cremers, "Visual-Inertial Navigation for a Camera-Equipped 25 g Nano-Quadrotor," pp. 4–5.

[28] M. Li and a. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, June 2013.

[29] S. Weiss, M. W. Achtelik, M. Chli, and R. Siegwart, "Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV," *2012 IEEE International Conference on Robotics and Automation*, pp. 31–38, May 2012.

[30] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation for autonomous rotorcraft MAVs in complex environments," *2013 IEEE International Conference on Robotics and Automation*, pp. 1758–1764, May 2013.

[31] S. K. Phang, K. Li, K. H. Yu, B. M. Chen, and T. H. Lee, "Systematic Design and Implementation of a Micro Unmanned Quadrotor System," *Unmanned Systems*, vol. 02, no. 02, pp. 121–141, Apr. 2014.

[32] F. Varesano, "FreeIMU: An Open Hardware Framework for Orientation and Motion Sensing," *arXiv preprint*, 2013.

[33] R. Mahony, T. Hamel, and J. Pflimlin, "Nonlinear Complementary Filters on the Special Orthogonal Group," *Automatic Control, IEEE Transactions*, 2008.

[34] D. Honegger and L. Meier, "An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications," *Intl. Conf. Robotics and Automation (ICRA 2013)*, 2013.