2017 International Conference on
Unmanned Aircraft Systems (ICUAS)
June 13-16, 2017, Miami, FL, USA

Altitude and Attitude Cascade Controller for a Smartphone-based Quadcopter

Alejandro Astudillo, Pedro Muñoz, Fredy Álvarez, and Esteban Rosero

***Abstract*—This paper presents the design and implementation of Android-based cascade PID controller structures to control the altitude and attitude of a quadcopter, and the sensor fusion algorithms implemented to estimate its flight dynamics. The main goal of this research is to stabilize the attitude of an unmanned aerial vehicle (UAV), such as a quadcopter, and control its altitude using exclusively the sensors and processor of a smartphone on-board. As the sensors embedded in the smartphones are not accurate enough to measure the altitude of the quadcopter, a linear Kalman filter for relative altitude estimation was designed and implemented. Here, it is described precisely the hardware that was used to build the test platform, the non-linear an linearized quadcopter model, the software structure to execute the controllers in the smartphone, the sensor fusion algorithms implemented to obtain reliable data from the smartphone sensors, and the cascade PID controllers design. Finally, the success of the proposed system is evidenced in the results of a set of experimental tests. In these tests, the quadcopter attitude was regulated after some disturbances were applied to the system and its altitude was controlled after the reference was changed.**

Keywords: Quadcopter, Altitude, Attitude, Control, Smartphone, Android-based controller, Cascade PID controllers

I. Introduction

Current smartphone processors are able to perform complex calculations such as those required in the implementation of real time control strategies. There are many ongoing research related to the possibility of using smartphones to implement control strategies, such as [1], as configuration and monitoring interfaces in control systems as seen in [2], [3], and as a tool in both education and design of control strategies seen in [4], [5]. Following this trend, in the Universidad del Valle, it was developed a smartphone-based platform for monitoring, control and communication in portable laboratories, where a controller for a pendulum based in the Lego Mindstorms EV3 platform was implemented [6].

In recent years, the interest in aerial robotics research has increased substantially. This is because this type of robotics offers several potential new services such as search and rescue, observation, mapping, inspection, etc. On the other hand, smartphones have become essential devices for humans and easily acquirable development tools. The interaction between these two technologies allow the development of low cost quadcopters based on an everyday item such as the smartphones, facilitating the distribution of the quadcopter control software and its implementation by other researchers.

All the authors are with the Research Group in Industrial Control of the School of Electrical and Electronic Engineering, Universidad del Valle, Cali, Colombia *(e-mail: {alejandro.astudillo, pedro.munoz, fredy.alvarez, esteban.rosero}@correounivalle.edu.co).*

Fig. 1. Assembled quadcopter used in this research with the on-board smartphone on the top center of it.

In the University of Pennsylvania, in [7], was developed a quadcopter using a last generation smartphone as a flight controller and an additional processing system for image-based positioning. The state estimation algorithms, control and planning were firstly implemented in a ODROID-XU board with additional sensors, but then, in [8], this algorithms were ported to the Qualcomm processor in the phone due to the Qualcomm colaboration in that project.

Current research focuses on the development of aerial robots potentiated by the use of smartphones, as seen in [9]–[12]. In the last years, computing capacity and sensor technology in smartphones has decreased in price but increased in performance. Smartphones have become an inexpensive tool capable of commanding an UAV. The challenge then, is to use smartphones as quadcopter flight controllers for autonomous flights following specific missions, taking advantage of the fact that the phones today are very powerful computers that include elements of sensing, processing and signal communication.

978-1-5090-4494-8/17/$31.00 ©2017 IEEE

In this paper, the implementation of a quadcopter with a smartphone acting as its flight controller while using exclusively the sensors and processor in the smartphone, is shown. The controller keeps the attitude of the quadcopter stabilized while making the quadcopter to hover at an altitude reference. It is presented the detailed composition of the test platform (quadcopter) used, integrating it with its dynamic model in addition to the quadcopter altitude and attitude estimation strategies using sensor fusion algorithms.

In the next section, a detailed description of the hardware used in the quadcopter is presented. After that, the dynamic non-linear and linearized quadcopter model is shown in Section III. In Section IV, all the aspects related to the altitude and attitude cascade controllers design are presented. Then, in Section V the sensor fusion algorithm that is used to estimate the altitude of the quadcopter, and how the orientation of the system is estimated, is described. In Section VI, the implementation in Android of all the necessary software with its results is provided. Finally, Section VII concludes this paper taking into account the results of this research and the future work to be done.

II. Hardware Overview

In this section, it is detailed the hardware used to build the quadcopter. To control the quadcopter it is necessary to sense variables such as orientation and position, and define control signals that will command the actuators. This tasks are done completely by the smartphone. The control signal goes from the smartphone, through an Arduino Mega ADK that acts as a gateway, and to the Electronic Speed Controllers (ESCs) that will set the motors speed depending on the control signal they receive.

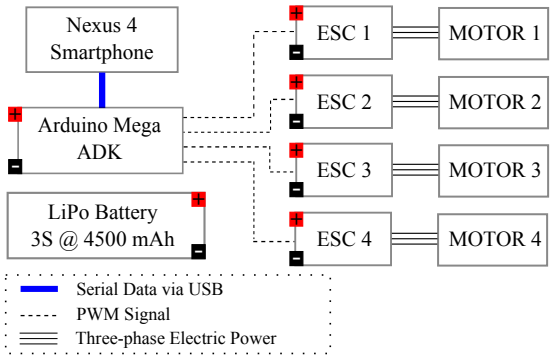

Fig. 2. Quadcopter hardware overview with signals detailing.

A. Nexus 4 Smartphone

The Nexus 4 is a smartphone developed by Google and assembled by LG, released in 2012 with Android 4.2 OS. This phone has a quad-core Qualcomm Snapdragon APQ8064 1.5 GHz processor, 2 GB of RAM memory and a total mass of 139 g. Its instrumentation includes accelerometers, gyroscopes, magnetometers, GNSS and barometer. It handles Bluetooth 4.0, WiFi, USB and GSM connectivities.

All the sensorial, estimation and control algorithms are executed in the Nexus 4 Smartphone, and it sends the control signals to the Arduino Mega ADK through an USB cable.

B. Arduino Mega ADK

The Arduino Mega ADK is a development board based on the Atmel 8-bit AVR RISC-based ATmega2560 microcontroller. The main use advantage of this board is that it handles a MAX3421E USB host chip which let us connect this board as a USB host to a Smartphone, while using the USB Host Library for Arduino.
This board receives the control signals through its USB port and translate them into PWM signals that will command each of the four ESCs.

C. LiPo Battery

To power all the hardware that makes up this quadcopter, it is used a 3 cells lithium-ion polymer battery with a nominal current capacity of 4500 mAh and a nominal voltage of 11.1 V. This battery can deliver up to 20 times its nominal current, so it can deliver 90 A continuously. As seen in Fig. 2 the battery is directly connected to the Arduino Mega ADK and to the high power supply of all the four ESCs.

D. Electronic Speed Controllers

To vary the speed of a R/C-type brushless motor it is necessary to use an ESC. This controllers receive a PWM signal, a low power supply and a high power supply, and output a three-phase electric power directly to the brushless motor. Four 30 A Simonk ESCs where used. Each of the ESCs receives the PWM signal and the low power supply from the Arduino Mega ADK board, while the high power supply is received directly from the LiPo battery.

E. Brushless Motors

The brushless motors are the actuators in any multirotor. The speed of each motor is controlled by the three-phase electric signal sent by the ESCs. In this quadcopter we used four DJI 2212 920 KV motors with 10” length and 4.5” pitch propellers.

III. Dynamic Model of the Quadcopter

In this section, the derivation of the quadcopter dynamic model is presented. First, there are shown the equations of motion that describe precisely the way the quadcopter is affected on its angular and linear accelerations. Then, a linearized model where just the altitude and attitude dynamics are taken into account is got.

A. Nonlinear Quadcopter Model

The nonlinear model is defined as a function of the state and input vectors X and U as $\dot{X} = f(X, U)$, where the state vector is $X = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} & \psi & \dot{\psi} & \theta & \dot{\theta} & \phi & \dot{\phi} \end{bmatrix}^T$, the input vector is set as $U = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T = \begin{bmatrix} u & \tau_\psi & \tau_\theta & \tau_\phi \end{bmatrix}^T$, and

$$\begin{aligned} u &= F_1 + F_2 + F_3 + F_4, \\ \tau_\psi &= T_2 + T_4 - T_1 - T_3, \\ \tau_\theta &= L(F_4 + F_1 - F_2 - F_3), \\ \tau_\phi &= L(F_1 + F_2 - F_3 - F_4), \end{aligned} \tag{1}$$

where x, y and z represent the quadcopter position in three dimensions, ψ is the yaw angle, θ is the pitch angle, ϕ is the roll angle, u is the throttle input and τ_ϕ, τ_θ and τ_ψ the rolling, pitching and yawing moment, F_i is the thrust force applied by the i-th motor, T_i is the torque each motor exert around the z-axis, and L is the distance between the center of mass of the quadcopter and the rotor of the motors. There are used two orthogonal geometrical frames for reference known as body frame (composed by x, y and z) and the inertial frame (composed by x_{world}, y_{world} and z_{world}). These variables are shown in Fig. 3.

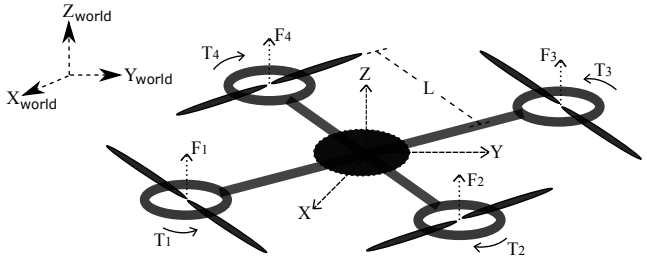

Fig. 3. Quadcopter scheme with movement axis, thrust forces, motor torques and reference frame.

The equations of motion that define the dynamics of the quadcopter are defined as

$$\begin{aligned} \ddot{x} &= \frac{u_1}{m}(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)) \\ \ddot{y} &= \frac{u_1}{m}(\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi)) \\ \ddot{z} &= \frac{u_1}{m}(\cos(\phi)\cos(\theta)) - g \\ \ddot{\psi} &= \dot{\phi}\dot{\theta}\frac{J_{xx} - J_{yy}}{J_{zz}} + \frac{u_2}{J_{zz}} \\ \ddot{\theta} &= \dot{\phi}\dot{\psi}\frac{J_{zz} - J_{xx}}{J_{yy}} + \frac{u_3}{J_{yy}} \\ \ddot{\phi} &= \dot{\theta}\dot{\psi}\frac{J_{yy} - J_{zz}}{J_{xx}} + \frac{u_4}{J_{xx}} \end{aligned} \tag{2}$$

where $J_{\bullet\bullet}$ are the moments of inertia of the quadcopter around each of its body axes [13], [14]. This is a simplified model of a real quadcopter; a complete quadcopter model can be found in [15].

B. Linearized Quadcopter Model

As the main objective of this project is to control just the altitude and attitude of the quadcopter, the state variables of the system are

$$X = \begin{bmatrix} z & \dot{z} & \psi & \dot{\psi} & \theta & \dot{\theta} & \phi & \dot{\phi} \end{bmatrix}^T. \tag{3}$$

The equations of motion are then reduced to four equations, neglecting the dynamics related to the linear movement in the x and y axes. The dynamic model is linearized using the Jacobian approach around the hover position where the velocity in the z direction and the angular velocities and positions tend to zero, representing a small-gain model.

$$\begin{aligned} \ddot{z} &= u_1/m, \\ \ddot{\psi} &= u_2/J_{zz}, \\ \ddot{\theta} &= u_3/J_{yy}, \\ \ddot{\phi} &= u_4/J_{xx}. \end{aligned} \tag{4}$$

The parameters of the quadcopter that was used in this project are shown in table I.

TABLE I
PARAMETERS OF THE QUADCOPTER

Parameter	Value
m	$1.152\ kg$
L	$0.24\ m$
J_{xx}	$0.0151\ kg \cdot m^2$
J_{yy}	$0.0151\ kg \cdot m^2$
J_{zz}	$0.0247\ kg \cdot m^2$

IV. CONTROLLER DESIGN

Here, it is presented the altitude and attitude controllers architecture that was implemented in Android to be executed in the on-board smartphone. The objective of this controllers is to regulate the quadcopter pose while hovering at a desired altitude.

A. Cascade PID Controllers

As explained in [16] and [17], one disadvantage of using a classic PID controller is related to the lack of robustness when the system suffers a large error accumulation due to the effect of a disturbance, affecting the settling time and the steady-state response of the controller. An approach to improve the stability and reduce the disturbance sensibility of the control system is to implement a cascade PID controller. In this case, there are designed four cascade PID controllers, one for each of the desired variables to be controlled (z, ψ, θ, ϕ). The overall scheme of the designed controller is shown in Fig. 4.

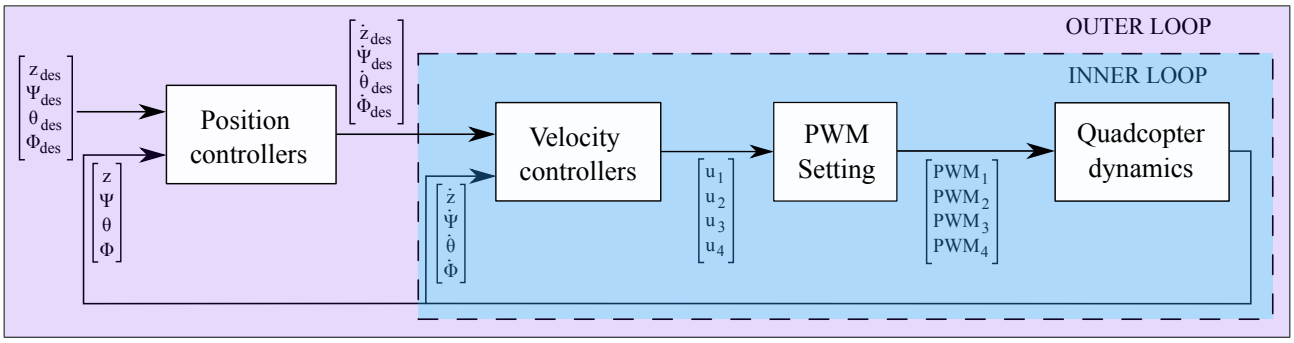

Fig. 4. Inner and outer loops of the cascade PID controllers implemented for altitude and attitude control.

The control signals u_1, u_2, u_4, u_4 are

$$\begin{aligned} u_1 &= k_{p\dot{z}} e_{\dot{z}} + k_{d\dot{z}} \dot{e}_{\dot{z}} + k_{i\dot{z}} \int e_{\dot{z}}, \\ u_2 &= k_{p\dot{\psi}} e_{\dot{\psi}} + k_{d\dot{\psi}} \dot{e}_{\dot{\psi}} + k_{i\dot{\psi}} \int e_{\dot{\psi}}, \\ u_3 &= k_{p\dot{\theta}} e_{\dot{\theta}} + k_{d\dot{\theta}} \dot{e}_{\dot{\theta}} + k_{i\dot{\theta}} \int e_{\dot{\theta}}, \\ u_4 &= k_{p\dot{\phi}} e_{\dot{\phi}} + k_{d\dot{\phi}} \dot{e}_{\dot{\phi}} + k_{i\dot{\phi}} \int e_{\dot{\phi}}, \end{aligned} \tag{5}$$

with $k_{p\bullet}$ as proportional gains, $k_{i_\bullet}$ as integral gains and $k_{d\bullet}$ as derivative gains of the Inner PID controllers, while $e_\bullet$ is each controller error. This errors are defined as

$$\begin{aligned} e_{\dot{z}} &= \dot{z}_{des} - \dot{z}, \\ e_{\dot{\psi}} &= \dot{\psi}_{des} - \dot{\psi}, \\ e_{\dot{\theta}} &= \dot{\theta}_{des} - \dot{\theta}, \\ e_{\dot{\phi}} &= \dot{\phi}_{des} - \dot{\phi}, \\ e_z &= z_{des} - z, \\ e_\psi &= \psi_{des} - \psi, \\ e_\theta &= \theta_{des} - \theta, \\ e_\phi &= \phi_{des} - \phi, \end{aligned} \tag{6}$$

where z_{des}, ψ_{des}, θ_{des} and ϕ_{des} are the desired values of the controlled variables, and

$$\begin{aligned} \dot{z}_{des} &= k_{pz} e_z + k_{dz} \dot{e}_z + k_{iz} \int e_z, \\ \dot{\psi}_{des} &= k_{p\psi} e_\psi + k_{d\psi} \dot{e}_\psi + k_{i\psi} \int e_\psi, \\ \dot{\theta}_{des} &= k_{p\theta} e_\theta + k_{d\theta} \dot{e}_\theta + k_{i\theta} \int e_\theta, \\ \dot{\phi}_{des} &= k_{p\phi} e_\phi + k_{d\phi} \dot{e}_\phi + k_{i\phi} \int e_\phi. \end{aligned} \tag{7}$$

As the execution of the controllers is done periodically by an Android thread, it was necessary to take into account that these controllers must be discretized using the sample time of the controller thread. Finally, the gains set for the PID controllers are shown in Table II. These gains were initially set using the Ziegler-Nichols tuning method, but during the implementation it was necessary to heuristically modify these parameters until achieving a stable flight.

TABLE II
PID GAIN VALUES

Controlled Variable	k_p	k_i	k_d
ψ	0.6	0.03	0.025
$\dot{\psi}$	1.3	0.1	0.24
θ	0.4	0.03	0.02
$\dot{\theta}$	1.3	0.1	0.24
ϕ	0.5	0.03	0.025
$\dot{\phi}$	1.3	0.1	0.24
z	0.8	-	-
$\dot{z}$	4	1.5	4

B. PWM Signal Setting

The ESCs inputs are PWM signals and it is necessary to calculate the PWM value for each motor ESC so that it represents the control signals delivered by the controllers. The PWM values can be set as

$$\begin{aligned} PWM_1 &= u - \tau_\theta - \tau_\phi - \tau_\psi, \\ PWM_2 &= u - \tau_\theta + \tau_\phi + \tau_\psi, \\ PWM_3 &= u + \tau_\theta + \tau_\phi - \tau_\psi, \\ PWM_4 &= u + \tau_\theta - \tau_\phi + \tau_\psi. \end{aligned} \tag{8}$$

V. ALTITUDE AND ATTITUDE ESTIMATION

The estimation algorithms that are implemented to get reliable data that describe the quadcopter flight dynamic, are shown in this section. The angular position relative to the three axis is obtained from the Rotation Vector virtual sensor in the smartphone, while it is shown that a linear Kalman filter is implemented to make an estimation of the quadcopter altitude.

A. Euler Angles Estimation

To get precise estimations of the Euler angles ϕ, θ and ψ, it is necessary to use a sensor fusion algorithm that takes into account the raw data generated from the accelerometers, gyroscopes and magnetometers.

Android smartphones have a sensor type called *Orientation Sensor* which outputs the three Euler angles directly. However the Android SDK 2.3.4 deprecated this sensor in its systems. The last Android SDKs have included a sensor type called *Rotation Vector (Q_s)* that replaces the deprecated sensor, delivering a Quaternion representation of the absolute orientation as the result of a Kalman filter that implements a fusion of the magnetometers, accelerometers and gyroscopes raw data. As shown in [18] the Q_s delivers a 4-component vector described by

$$Q_s = \begin{bmatrix} u\sin(\alpha/2) \\ v\sin(\alpha/2) \\ w\sin(\alpha/2) \\ \cos(\alpha/2) \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}, \tag{9}$$

where α is the amount of degrees the Quaternion is rotated through around the rotation axis $u\vec{i} + v\vec{j} + w\vec{k}$. The Euler angles can be obtained from the Quaternion representation of Q_s as

$$\begin{bmatrix} \psi \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} atan2(2(q_3q_2 + q_0q1), 1 - 2(q_1^2 + q_2^2)) \\ arcsin(2(q_3q_1 - q_2q_0)) \\ atan2(2(q_3q_0 + q_1q2), 1 - 2(q_0^2 + q_1^2)) \end{bmatrix}. \tag{10}$$

The function *atan2* is the arctangent computing function with two arguments that produces results in range $(-\pi, \pi]$ but is undefined for $(0, 0)$ arguments.

In the Android environment, the Euler angles are obtained using some methods from the *SensorManager* class [19]. The method *getRotationMatrixFromVector* generate a rotation matrix from the quaternion. The *remapCoordinateSystem* method rotates the generated rotation matrix to the world coordinate frame. Finally, the *getOrientation* method computes the orientation vector $[\psi,\ \theta,\ \phi]^T$ from the remapped rotation matrix. This orientations will be used as feedback to the angular position controllers while the angular velocities acquired from the gyroscope will be used as measurements for the angular velocities controllers.

B. Altitude Estimation

As the quadcopter dynamics are sensed exclusively using the sensors embedded in the on-board smartphone, one completely depends on the accuracy of these sensors. In [20] the importance of adopting a sensor fusion algorithm that improves the navigation capabilities of an UAV was demonstrated; thus, it is necessary to improve the accuracy of the smartphone sensors with algorithms such as the Kalman Filter.

The barometer on the smartphones are not accurate enough, so a linear Kalman filter for altitude tracking was designed. This Kalman filter will use the position and velocity (from the barometer) and the linear acceleration (from the accelerometer and the orientation estimations) measurements to improve the position estimation.

As seen in [21], the barometric pressure measurement is transformed in a elevation measurement as

$$h_{bar}(k) = 44330(1 - \frac{p(k)}{p_0}^{1/5.255}), \tag{11}$$

where $p(k)$ is the current barometric pressure and p_0 is the atmospheric pressure at sea level in hPa.
The initial elevation measurement is expressed as $h_{bar}(0)$. Then the relative altitude estimated with just the barometer measurements, are calculated as

$$h_{bar\_rel}(k) = h_{bar}(k) - h_{bar}(0). \tag{12}$$

Thus, the altitude velocity of the barometer measurements is obtained by a simple discrete derivation as

$$v_{bar} = \frac{h_{bar\_rel}(k) - h_{bar\_rel}(k-1)}{\delta t}, \tag{13}$$

where δt is the sample time.

On the other hand, to get the accelerometer measurement in the world z-axis direction, it is necessary to transform the accelerations from the telephone coordinates frame to the world coordinates frame. This transformation is done by rotating the accelerometer vector in the telephone coordinates $a_T(k)$ using the quaternion Q_s as

$$a_W(k) = Q_s a_T(k) Q_s', \tag{14}$$

where $Q_s' = \begin{bmatrix} -q_0 & -q_1 & -q_2 & q_3 \end{bmatrix}^T$ is the quaternion conjugate of Q_s. The z-axis world coordinate acceleration was obtained as

$$a_{WZ}(k) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} a_W(k). \tag{15}$$

Once the vector $\begin{bmatrix} h_{bar\_rel} & v_{bar} & a_{WZ} \end{bmatrix}^T$ is calculated, it must be taken as the measurement vector of the linear Kalman filter shown below to get the filtered altitude estimation. The Kalman filter starts making a prognosis of the states $E(k)$ and the covariance $P(k)$, based on the initial conditions set ($\hat{x}(0)$ and $P(0)$), as

$$\hat{E}^-(k) = \Gamma \hat{E}(k-1) + \Sigma F(k), \tag{16}$$

$$P^-(k) = \Gamma P(k-1) \Gamma^T + Q(k), \tag{17}$$

where Γ is the system matrix, $F(k)$ is the input vector, Σ is the input matrix and $Q(k)$ is the process variance at sample k. The upper index " $-$ " means that the variable is based on previous estimations.

The system states will correspond to the position, velocity and acceleration in the world z-axis direction, thus the

matrices Γ and Σ and the state vector $E(k)$, can be defined as

$$\Gamma = \begin{bmatrix} 1 & \delta t & \frac{(\delta t)^2}{2} \\ 0 & 1 & \delta t \\ 0 & 0 & 1 \end{bmatrix}, \tag{18}$$

$$\Sigma = 0_{3\times 1}, \tag{19}$$

$$E(k) = \begin{bmatrix} z(k) & \dot{z}(k) & \ddot{z}(k) \end{bmatrix}^T. \tag{20}$$

After the prediction is completed, the states are corrected using the actual measurements of the process as

$$\begin{aligned} K(k) &= P^-(k)H^T(HP^-(k)H^T + R)^{-1}, \\ \hat{E}(k) &= \hat{E}^-(k) + K(k)(N(k) - H\hat{E}^-(k)), \\ P(k) &= (I - K(k)H)P^-(k), \end{aligned} \tag{21}$$

where R is the measurement covariance matrix, $K(k)$ is the Kalman gain, $N(k)$ the measurement vector and I is identity matrix. Matrix $H = I_{3\times 3}$ was built taking into account that the vector of measurements is $N(k) = \begin{bmatrix} h_{bar\_rel} & v_{bar} & a_{WZ} \end{bmatrix}^T$ and

$$N(k) = HE(k) + v, \tag{22}$$

where v is a zero mean white noise. Finally, the estimated altitude and velocity in the world z-axis direction can be got from the first and second component of the $\hat{E}(k)$ vector respectively.

VI. Implementation and Results

In this section, it is evaluated the performance of the designed controllers implemented in Android to control the built quadcopter.

A. Software Limitations

As shown in [11], Android can not provide real-time data processing while its Real Time Operating System features are at a prototype stage. This means that the different threads running under Android, like the control loop and estimation algorithms, can not assure a constant sample time during all the execution. Also, the sample time of the sensors is variable, although different sample frequencies can be set (50, 100 or 200 Hz) this values are just nominal and are not achieved precisely. However, it was decided to use a smartphone as development platform because of the large processing capacity, the variety of embedded sensors they have, and the wired and wireless communication interfaces these devices have.

Taking into account this limitations, it was implemented a temporized thread that executes the control loop each 10 ms, reading the output measurements and estimations just before running the control algorithms. This sampling frequency is chosen to ensure that each time the control action is executed, new data will be available from the sensors, in addition to maintaining an almost constant execution frequency. In Fig. 5 it is shown the histogram of sample time logged while an experimental test on the real quadcopter was developed.

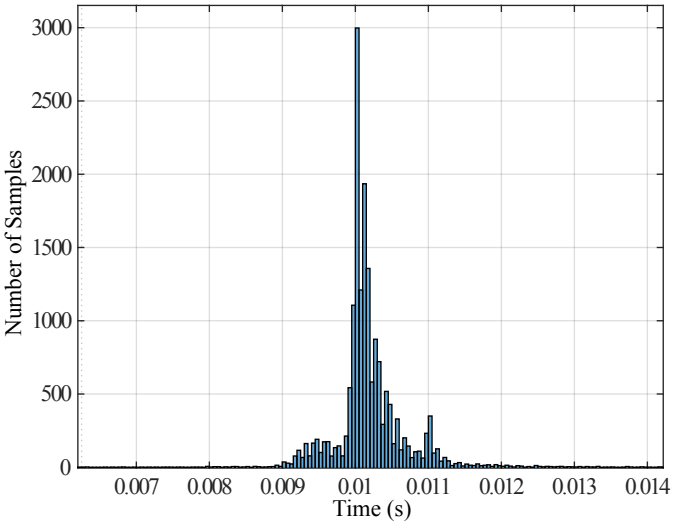

Fig. 5. Histogram of the measured sample time after a test execution.

As can be seen, the temporized thread is mainly executed each $10.2\ ms$, but it has a standard deviation of $0.56\ ms$. The following tests were developed using this sample time variation profile.

B. Controller Test Results

To test the designed controllers performance, there were developed some tests while the quadcopter was flying, as shown in Fig. 6. These tests include applying disturbances to the roll, pitch and yaw dynamics, and changing the reference of the altitude and yaw angle dynamics.

Fig. 6. Quadcopter flying during tests.

In Fig. 7 and 8 is presented how the designed controller is able to reject the effect of disturbances in the dynamics

of the roll and pitch angle, respectively. These disturbances were manually applied while flying pulling the quadcopter from its landing gear to change its pose. Also, it is shown the control signals excursion reacting to the applied disturbances.

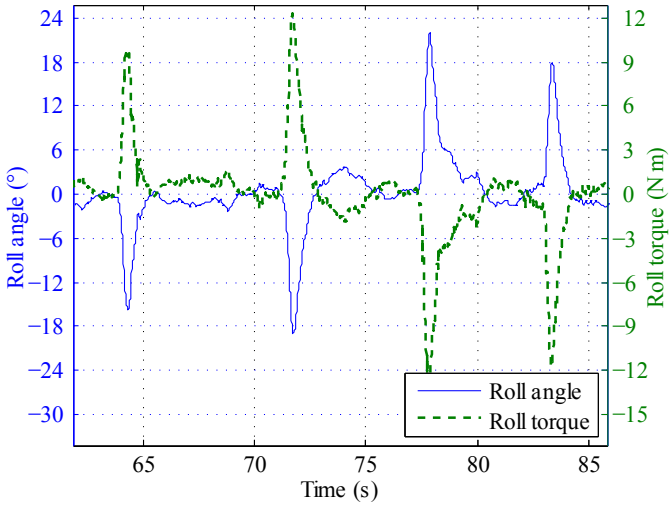

Fig. 7. Roll angle controller response after being subjected to disturbances.

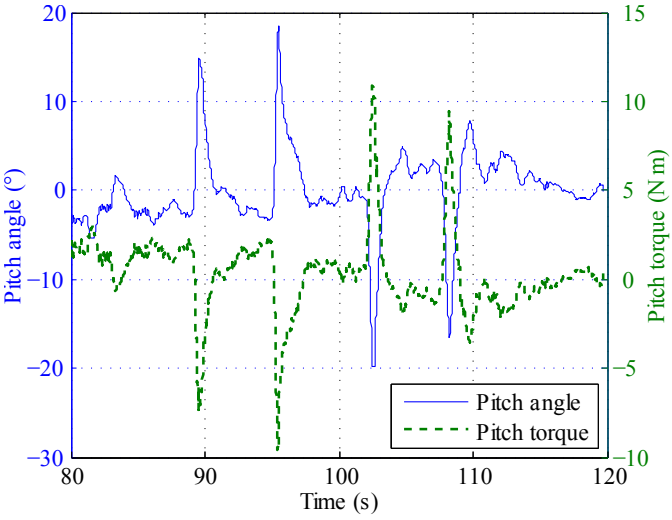

Fig. 8. Pitch angle controller response after being subjected to disturbances.

The yaw angle cascade controller response is shown in Fig. 9. To test this controller, there were manually imposed torques around the z-axis as disturbances and a step in the yaw reference signal was applied.

In Fig. 10, it is plotted the altitude controller response after the reference in the altitude is changed from 2 to 4 m. This response has an overshoot of 16.95 % and a rise time of 7 s.

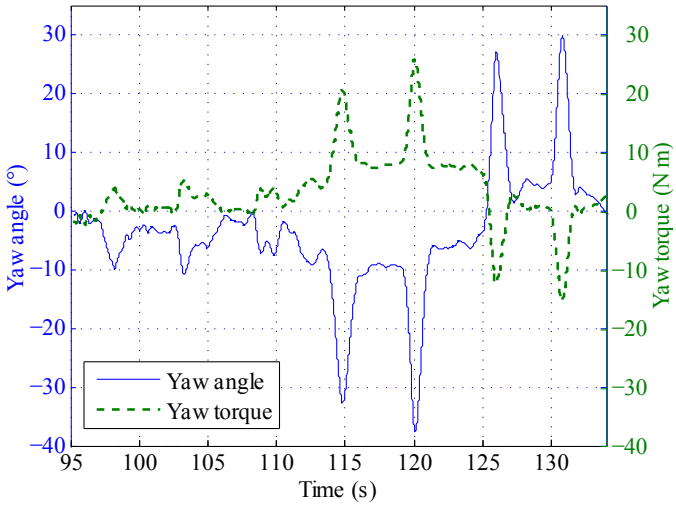

Fig. 9. Yaw angle controller response after being subjected to disturbances.

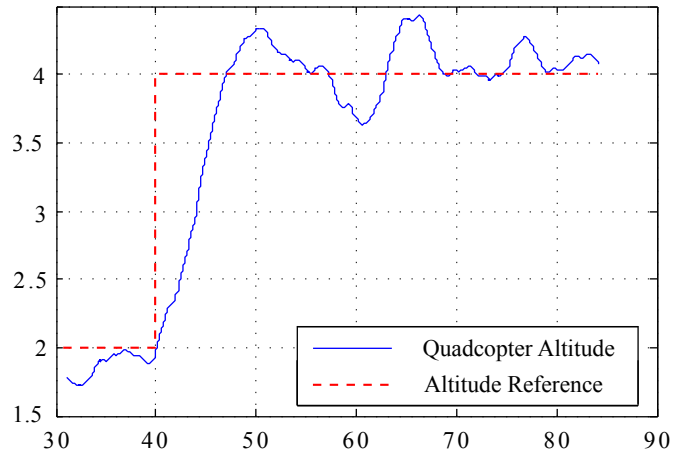

Fig. 10. Altitude controller response after a change in the relative altitude reference.

The controller response was also tested with a force disturbance that affected the altitude of the quadcopter by pulling it to the ground at time 38 s, as can be seen in Fig. 11.

VII. Conclusion

This paper presented the development of a quadcopter controlled exclusively using the sensors and the computational capacity of an on-board smartphone. Analysing the developed tests, the proposed cascade PID controllers showed a good performance in disturbance rejection of the orientation and reference changes in the altitude and heading of the quadcopter. Despite the fact that the orientation angles delivered by the Android SensorManager class use a generic Kalman filter that is not tuned for each of the smartphones running Android, the proposed controller was able to regulate these angles stabilizing the quadcopter and allowing it to fly. The relative

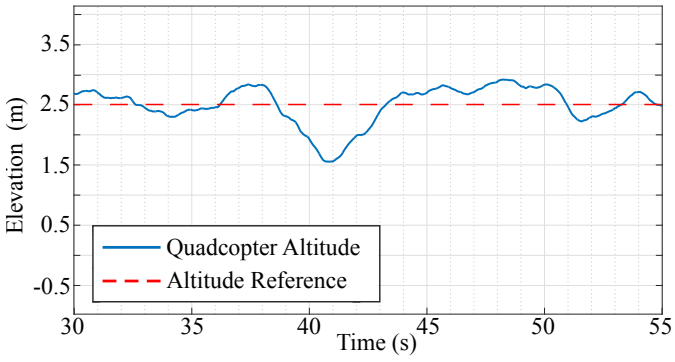

Fig. 11. Altitude controller response after a disturbance in 38 s is applied.

altitude estimation was achieved due to a designed and implemented linear Kalman filter. This Kalman filter was feed with the data acquired from the barometer and the accelerometer after being converted to world coordinates. The feedback outputs were taken from the altitude Kalman filter, the orientation delivered by the Androd SensorManager class and the angular velocities from the gyroscope.

The future work will be oriented toward adding the x and y position controllers to the implementation and improving the quadcopter response implementing better estimation algorithms and more advanced controllers such as a LQG or H∞ controller for reference tracking and regulation.

REFERENCES

[1] A. Drumea, "Control of Industrial Systems Using Android-Based Devices," *36th Int. Spring Seminar on Electronics Technology*, pp. 405–408, 2013.

[2] K.-w. Lin, Z.-h. Wu, and Y.-l. Lin, "Applications of Temperature Control Based on Android Platform," *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. II, 2014.

[3] N.-V. Truong and D.-L. Vu, "Remote monitoring and control of industrial process via wireless network and Android platform," *2012 International Conference on Control, Automation and Information Sciences (ICCAIS)*, no. 1, pp. 340–343, 2012.

[4] L. F. Aristizabal, D. F. Almario, and J. A. Lopez, "Development of an Android App as a learning tool of dynamic systems and automatic control," in *2014 III International Congress of Engineering Mechatronics and Automation (CIIMA)*. IEEE, oct 2014, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/6983438/

[5] S. Wu Wu, "Sintonización de controladores PI/PID mediante el desarrollo de una aplicación en Android," Ph.D. dissertation, Universidad de Costa Rica, 2013.

[6] J. García Téllez and D. F. Ochoa Calambás, *Desarrollo de una plataforma de monitorización, control y comunicación con teléfonos inteligentes, para laboratorios portátiles de soporte al área de señales y sistemas*. Cali: Universidad del Valle, 2015.

[7] G. Loianno, G. Cross, C. Qu, Y. Mulgaonkar, J. A. Hesch, and V. Kumar, "Flying Smartphones: Automated flight enabled by consumer electronics," *IEEE Robotics Automation Magazine*, vol. Vol. 22, pp. pages 24 – 32, 2015.

[8] G. Loianno, Y. Mulgaonkar, C. Brunner, D. Ahuja, A. Ramanandan, M. Chari, S. Diaz, and V. Kumar, "Smartphones power flying robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, sep 2015, pp. 1256–1263. [Online]. Available: http://ieeexplore.ieee.org/document/7353530/

[9] C. Pearce, M. Guckenberg, B. Holden, A. Leach, R. Hughes, C. Xie, M. Hassett, A. Adderley, L. E. Barnes, M. Sherriff, and G. C. Lewin, "Designing a spatially aware, automated quadcopter using an Android control system," in *2014 Systems and Information Engineering Design Symposium (SIEDS)*, vol. 00, no. c. IEEE, apr 2014, pp. 23–28. [Online]. Available: http://ieeexplore.ieee.org/document/6829921/

[10] A. Bjälemark and H. Bergkvist, "Quadcopter control using Android based sensing," *Advances in Electrical and Computer Engineering*, pp. 15–21, 2014.

[11] L. Aldrovandi, M. Hayajneh, M. Melega, M. Furci, R. Naldi, and L. Marconi, "A smartphone based quadrotor: Attitude and position estimation," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2015, pp. 1251–1259. [Online]. Available: http://ieeexplore.ieee.org/document/7152418/

[12] P. Bryant, G. Gradwell, and D. Claveau, "Autonomous UAS controlled by onboard smartphone," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2015, pp. 451–454. [Online]. Available: http://ieeexplore.ieee.org/document/7152322/

[13] M. Emam and A. Fakharian, "Attitude tracking of quadrotor UAV via mixed H2/H¡inf¿¡/inf¿ controller: An LMI based approach," in *2016 24th Mediterranean Conference on Control and Automation (MED)*. IEEE, jun 2016, pp. 390–395. [Online]. Available: http://ieeexplore.ieee.org/document/7535919/

[14] S. Badr, O. Mehrez, and A. E. Kabeel, "A novel modification for a quadrotor design," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2016, pp. 702–710. [Online]. Available: http://ieeexplore.ieee.org/document/7502536/

[15] S. Bouabdallah, "Design and Control of Quadrotors With Application To Autonomous Flying," *École Polytechnique Fédérale De Lausanne, À La Faculté Des Sciences Et Techniques De L'Ingénieur*, vol. 3727, no. 3727, p. 61, 2007.

[16] G. L. Raja and A. Ali, "Series cascade control: An outline survey," in *2017 Indian Control Conference (ICC)*, no. Icc. IEEE, jan 2017, pp. 409–414. [Online]. Available: http://ieeexplore.ieee.org/document/7846509/

[17] P. Wang, Z. Man, Z. Cao, J. Zheng, and Y. Zhao, "Dynamics modelling and linear control of quadcopter," in *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*. IEEE, nov 2016, pp. 498–503.

[18] Android Developer, "SensorEvent," 2015. [Online]. Available: https://developer.android.com/

[19] ——, "Sensormanager," 2015. [Online]. Available: https://developer.android.com/reference/android/hardware/SensorManager.html

[20] F. Cappello, S. Ramasamy, R. Sabatini, and J. Liu, "Low-cost sensors based Multi-Sensor Data Fusion techniques for RPAS Navigation and Guidance," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2015, pp. 714–722.

[21] K. S. Lauszus, "Flight Controller for Quad Rotor Helicopter in X-configuration," Ph.D. dissertation, Kongens Lyngby, Denmark, 2015.