UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

TRABALHO DE CONCLUSÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Implementation of a fixed-wing UAV autopilot in Snapdragon Flight board

Author: Alexandre Loeblein Heinen

Advisors: Carlos Eduardo Pereira
Edison Pignaton de Freitas

Porto Alegre, July 27, 2017

Contents

Abstract

In our days, developing high-end flight controllers is a challenge in industry and research. This rapid growth of the variety of applications using autonomous vehicles is mainly due to the recent advances is solid state electronics, which allows the miniaturization of the embedded devices, such as sensors, microprocessors and antennas. Thus, as new technologies are available, developing cost-effective UAV systems is also being emphasized.

The goal of this work is to propose a new platform which embeds a professional solution autopilot – based on the PX4 project – in the Snapdragon Flight board, a high-end onboard computer introduced by Qualcomm, which aims the development of new autonomous vehicles for consumer, enterprise and hobbyist markets. Since it has very appropriated imaging and stability capabilities, a fixed-wing airframe is chosen, more specifically, Zangão V professional solution UAV by SkyDrones.

After a brief introduction about the main concepts of a fixed-wing aircraft flight, we present a reliable platform architecture, which should support the further development of new applications for fixed-wing UAVs, by taking advantage of the high processing capacity of Snapdragon Flight board. At last, some remarks on the platform potentials, limitations and eventual future projects are made at the end of this report.

Resumo

Nos dias atuais, o desenvolvimento de controladores de voo de ponta é um desafio tanto para a indústria, quanto para a pesquisa. O rápido aumento do número de aplicações que utilizam veículos autônomos é principalmente devido aos recentes avanços na eletrônica de estado sólido, que permite a miniaturização dos disponitivos embarcados, como sensores, microprocessadores e antenas. Assim, à medida que novas tecnologias estão disponíveis, o desenvolvimento de sistemas de VANTs (veículos aéreos não-tripulados) rentáveis também vem sendo enfatizado.

O objetivo deste trabaho é propor uma nova plataforma que embarca um piloto automático profissional – beseado no projeto PX4 – na plataforma Snapdragon Flight, um computador de bordo de ponta introduzido pela Qualcomm que visa o desenvolvimento de novos veículos autônomos para os mercados consumidores, empresariais e amadoristas. Uma vez que esta configuração tem boas propriedades de captura de imagens e de estabilidade, optou-se por um estrutura de asa fixa, mais especificamente, o VANT profissional Zangão V da empresa SkyDrones.

Após uma breve introdução sobre os principais conceitos de voo de uma aeronave de asa fixa, apresentamos uma arquitetura de plataforma fiável, que ambiciona o desenvolvimento futuro de novas aplicações para VANTs de asa fixa, tirando proveito assim da alta capacidade de processamento da plataforma Snapdragon Flight. Por fim, alguns comentários sobre os potenciais, limitações e eventuais trabalhos futuros da plataforma são tecidos no final deste relatório.

List of Figures

List of Tables

List of Abbreviations

ADB	Android Debug Bridge. 25
AHRS	Attitude Heading and Reference Systems. 11
AIMC	adaptive internal model control. 12
AOA	angle of attack. 7, 11
BAM	Bus Access Manager/Module. 17
BEC	Battery eliminator circuit. 30, 32
BLSP	BAM Low Speed Peripheral. 17, 45
CPU	central processing unit. 17
CS	control surface. 7, 15
DSP	Digital Signal Processor. 17, 25
ECL	Estimation and Control Library. 20, 21, 26
EKF	Extended Kalman Filter. 11, 21, 22, 27, 45
ESC	Electronic Speed Control. 15, 30
FIFO	First In, First Out. 21
GCS	ground control station. 17, 19, 24, 32, 35
GPS	Global Positioning System. 1, 11, 36
GPU	graphics processing unit. 17
IMU	Inertial Measurement Unit. 11, 18, 21, 27
ISR	intelligence, surveillance and reconnaissance. 1
KF	Kalman Filter. 11, 22, 45
LQR	Linear-Quadratic Regulator. 11
MAVLink	Micro Air Vehicle Communication Protocol. 16, 19, 26, 35
OS	Operating System. 2, 25
OSI	Open Systems Interconnect. 16

PID	proportional-integral-derivative. 11
POSIX	Portable Operating System Interface. 17, 25
PSO	Particle Swarm Optimization. 11, 12
PWM	pulse width modulation. 16, 19, 22, 24, 28
R/C	radio control. 4, 23, 29
RTOS	Real-Time Operating System. 17
S.Bus	Futaba Serial Bus. 16, 23, 24, 29
SLAM	Simultaneous localization and mapping. 36
SSA	slideslip angle. 7, 11
UAV	Unmanned Aerial Vehicle. 1, 5, 10
V/STOL	Vertical/Short Take-Off and Landing. 1

List of Symbols

$\boldsymbol{F} \triangleq (X \quad Y \quad Z)^T$	aerodynamics force components. 6
$\boldsymbol{M} \triangleq (L \quad M \quad N)^T$	aerodynamics moment components. 7
$\boldsymbol{V} \triangleq (u \quad v \quad w)^T$	linear velocity. 6
$\boldsymbol{\Omega} \triangleq (p \quad q \quad r)^T$	angular rate/velocity. 6
$\boldsymbol{\Phi} \triangleq (\phi \quad \theta \quad \psi)^T$	Euler angles: *roll* (ϕ), *pitch* (θ) and *yaw* (ψ). 6
ρ	air density at sea level and 15°C, approx. $1,225\text{kg/m}^3$. 7
$\hat{x}(t)$	estimate of the state variable x at continuous time instant t. 5
$\hat{x}(n\|m)$	estimate of x at discrete time instant n given observations up to and including at time $m \leq n$.. 45
$x_r(t)$	reference for state variable x at continuous time instant t. 5
$\tilde{x}(t)$	control error on state variable x at continuous time instant t. 5

1 Introduction

Unmanned Aerial Vehicles (UAVs) have been widely employed for numerous emerging technologies thanks to their flight characteristics (rapid dynamics, easy and risk-free maneuverability) and low cost, if compared with conventional manned aircrafts (Sharma et al., 2014), (Lee et al., 2016). Consequently, UAVs are being increasingly used in activities in a range of fields:

- In military operations like intelligence, surveillance and reconnaissance (ISR), assisted decision-making and rescue missions. Indeed, UAVs can make low-altitude and slow-speed flights, and thus, they can provide good stealth and endurance characteristics (Becker and Sheffler, 2016), (Orfanus et al., 2016), (Lee et al., 2016).
- For power line inspection, by image based navigation algorithms that detects power lines in cameras images and guide the UAV navigation closely along them (de Avila Wieczorek, 2017), (Jones, 2005), (Zhou et al., 2016). Compared to other classes of inspections, the UAV option reduces drastically the operation costs (Sharma et al., 2014).
- When integrated with multispectral cameras and Global Positioning System (GPS), UAVs are widely employed for mapping, industrial imaging and precision agriculture. Indeed, this latter application improves the production process (phenotype) and decision-making to improve crops' health (Habib et al., 2017), (Flores et al., 2017).

Different UAV configurations exist depending on the propulsion principle that is used. For instance, Hoffer et al., 2014, divides them into five groups: helicopter, fixed-wing, multirotor, flapping-wing, and lighter-than-air. However, new UAV airframes such as Vertical/Short Take-Off and Landing (V/STOL) aircrafts – which transit from hovering to airplane flight – are increasingly studied in the literature (Naldi and Marconi, 2009), (Becker and Sheffler, 2016).

The main interest of this work will be the **fixed-wing UAVs**, more particularly the **Zangão V** UAV solution (Section 1.2).

To ensure reliable navigation of a fixed-wing UAV, modern modeling, estimation and control techniques are deployed. The implementation of such algorithms became possible due to recent advances in solid state electronic components and processing unities (Briod et al., 2016).

1.1 Motivations and objectives

The main goal of the present work is the implementation of an autopilot for a fixed-wing UAV, namely Zangão V. The deployment of a such high-end platform requires many technical skills, including C/C++ programming, deep understanding of control theory and microprocessors architecture, and broadly based knowledge about electronics, remote sensing and and foremost, aeronautics.

Nowadays, interest in both civilian and commercial use of UAV – usually referred as drones –

has continued to grow rapidly. This phenomenon was mainly due to the high-end technologies companies, research institutes and drone-building community, which created many software and hardware projects under open licenses that allow the development of end-to-end platforms.

Nonetheless, the development of high-end devices encounters many difficulties in Brazil, especially due to the lack of skilled human resources and R&D (research and development) activities. This handicap can be seen in recent governmental reports like Secretaria de Desenvolvimento e Competitividade Industrial (SDCI), 2016, which describes the current efforts aiming at the development and regulation of this sector in Brazil.

The present work is thus within the scope of the contribution of both SkyDrones Tecnologia Aviônica and the Universidade Federal do Rio Grande do Sul (UFRGS) for the sector, as it can be seen in sections 2.1.4 and 2.8.1 of Secretaria de Desenvolvimento e Competitividade Industrial (SDCI), 2016.

Based on a solid background, this work proposes a functional platform that implements the basic features of a fixed-wing aircraft.

To do so, the existing Zangão V airframe is used, however, electronic components, control and navigation algorithms are reviewed and the new architecture is fully based on the Snapdragon Flight board, by Qualcomm.

In other words, the main activities can be summarized as:

1. Compile a base autopilot in a Linux OS, by configuring the environment and by linking the firmware with compiler, assembler and lower-level libraries;
2. Study the autopilot structure in order to identify and understand its modules;
3. Define all modifications required to adapt the existing algorithms/drivers to the new airframe;
4. Implement all the softwares updates previously defined;
5. Assemble all the electronic components;
6. Validate the firmware behavior regarding its flight capabilities;
7. Fix any remaining bugs and unforeseen circumstances that turn up during the project advancements.

This concluding project (in Portuguese, *Trabalho de Conclusão de Curso*) is a requirement for obtaining the bachelor's degree in Control and Automation Engineering at the Universidade Federal do Rio Grande do Sul (UFRGS) and it was held in partnership with SkyDrones Tecnologia Aviônica.

1.2 About Zangão V

The fixed-wing UAV professional solution Zangão V (Figure 1) embeds a set of sensors that ensures a high stability flight, which requires reduced pilotage training. Oppositely to adaptations of another

existing UAVs, Zangão V was fully designed to carry dedicated cameras. Fields of applications include mapping of agricultural areas, topography/surveying, power lines inspection, environmental inspection, just to name a few.

Some of Zangão V's important features:

- Automatic stabilization, based on autonomous actioning of its control surfaces.
- Easy take-off and landing by means of a catapult and a parachute, respectively.
- Programmable flight planning over a predetermined area, aiming image acquisition and aerial observation.
- High payload capacity and robust framework, which enable the usage of different cameras and sensors.

Figure 1: Zangão V fixed-wing UAV

1.3 Work outline

To accomplish the objectives introduced in Section 1.1, firstly, some general concepts in aeronautics are introduced in Section 2.1, as these basic notions are very important to understand the control of fixed-wing UAVs. Afterwards, Section 2.2 presents a state of art of modern control techniques, whereas Section 2.3 states the more important open-source autopilot solutions applied to this class of aircrafts.

Based on these definitions, in Section 3, the specifications for this project are set and we provide a design overview of the components that are used to accomplish these goals.

Once all the components are set and well understood, in Section 4, all the main modules are described along with all eventual modifications required to adapt it to Zangão V UAV.

In a nutshell, the autopilot is based on an open-source professional solution, the PX4 autopilot. It has to be built, configured and then embedded in Snapdragon Flight board, a high-end flight controller released by Qualcomm Technologies (Section 4.1). These components ensure a platform that integrates both PX4 and Snapdragon Flight and which is able to implement the control laws deployed by PX4 Flight Stack (Section 4.2).

Furthermore, the internal logic must be adapted to Zangão V structure, allowing it to mix the right

efforts into the control surfaces (Section 4.3) and receive R/C commands (Section 4.4). Finally, all these modules are assembled into a final prototype (Section 4.5).

The relevant results are then presented and analyzed in Section 5. Finally, the current work is concluded by Section 6, where the abovementioned modules and results are discussed aiming at a flying drone. A set of future works is presented based on the author perspective of the project follow-up.

2 Literature review

The so-called "Atmospheric flight" is a general topic that embraces three major disciplines: performance, flight dynamics and aeroeslaticity (Nelson, 1989).

While performance deals with airplane indices such as endurance, take-off/landing distances and flight plan optimization; aeroeslaticity describes how inertial, elastics and aerodynamics forces interact with the aircraft structure. Both McCormick, 1994, and Pamadi, 2004, have an extended description of a whole set of phenomena that influence the flight performance and aeroeslaticity.

However, this section will mainly discuss how UAVs – and more generally aircrafts – flight dynamics can be modeled. This third topic deals with the motions of an airplane in a 3D space due to internal and external efforts and disturbances. Indeed, we will be namely interested in UAV stability and control capabilities.

Although, another crucial point that must be highlighted is that, due to the flexibility of modern vehicles structures, the interplay between those three subjects cannot be completely ignored. Thus, some particular points of aerodynamics modeling will be discussed in this section, even though, as noted above, performance and aeroeslaticity are not a direct interest of the current work.

2.1 Basic concepts

Throughout this work, scalars are represented by regular italic roman, Greek, lower or uppercase letters, like x, L or α, while vectors and quaternions (see Appendix D) are represented by bold symbols, like $\boldsymbol{q}$ or $\boldsymbol{\Omega}$. Matrices are less frequently employed, however they will appear as stylized symbols as $\mathfrak{J}$ and $\mathfrak{H}$.

Regarding control and estimation purposes, given a continuous time instant t and a state variable x,

- $\hat{x}(t)$ represents the estimate of the state variable x,
- $x_r(t)$ is the reference input of the controller and
- $\tilde{x}(t)$ is the control error

$$\tilde{x}(t) = x_r(t) - \hat{x}(t) \tag{1}$$

2.1.1 Rigid body dynamics

For the purpose of simplicity, the nomenclature employed in this report is mainly the one used by Nelson, 1989, with a few modifications that aim to clarify the text reading and somewhat avoid excessive definitions.

There are two main coordinate systems used in aircraft motion models:

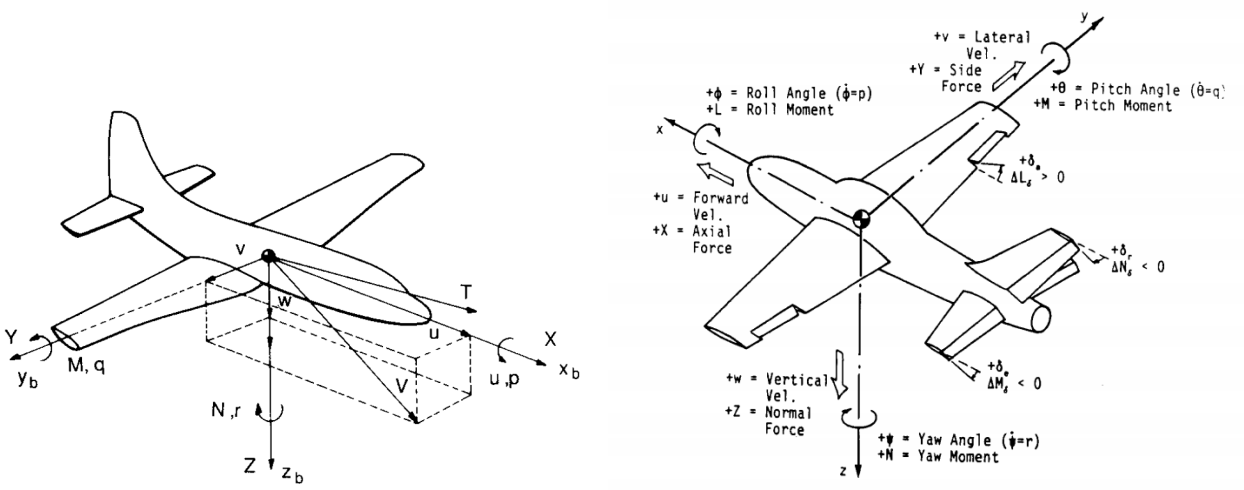
- The body frame (or coordinate system) is fixed to the aircraft and variables related to this

frame are symbolized with an subscribed "b", namely the axes (x_b, y_b, z_b).

- The earth or NED frame (which stands for North-East-Down) is fixed to the earth. Variable are indicated by a subscript "e", namely the axis (x_e, y_e, z_e).

For both coordinates systems, the z-axis is always pointed downwards and, if no subscript is indicated, the variable is considered as in the body frame. This concept is represented in Figure 2a.

In body frame, the (x, y, z) axes are respectively called roll, pitch and yaw axes. Thus, it is usual to reference a body on a tridimensional space by its roll, pitch and yaw angles (the so-called Euler angles), which relate body and earth frames.



(a) Body frame and the definition of vectors. (b) Control surfaces and axes of motion.

Figure 2: Motion-related definitions for an aircraft.

Source: Nelson, 1989

The representation by Euler angles is intuitive and easy to develop, but computationally intense. Yet, another alternative is to represent the body orientation by a quaternion.

This representation is based on Euler's rotational theorem which states that the relative orientation of two coordinate systems can be described by only one rotation about a fixed axis. A reference can be made to Appendix D to relate quaternions properties and their relation with Euler angles.

There is a set of vectors which will be useful later and which are closely related to the coordinate system that is being used, in particular

- Euler angles: *roll* (ϕ), *pitch* (θ) and *yaw* (ψ), $\mathbf{\Phi} \triangleq (\phi \quad \theta \quad \psi)^T$,
- linear velocity, $\boldsymbol{V} \triangleq (u \quad v \quad w)^T$,
- angular rate/velocity, $\mathbf{\Omega} \triangleq (p \quad q \quad r)^T$,
- aerodynamics force components, $\boldsymbol{F} \triangleq (X \quad Y \quad Z)^T$ and

- aerodynamics moment components, $\boldsymbol{M} \triangleq (L \quad M \quad N)^T$.

The mass moments of inertia are I_x, I_y and I_z, whereas the product of inertia are I_{yz}, I_{xz} and I_{xy}.

In addition, Section 2.1.5 describes how aerodynamics efforts (forces and moments) are modeled and how they are related to the aircraft profile, state (velocities and orientation) and control variables.

2.1.2 Aeronautics-related definitions

In a regular airplane, different control devices can act on the vehicle stability, such as (1) ailerons, (2) elevators, (3) rudders, (4) thrust vectoring by an engine, (5) speedbrakes/spoilers, (6) canard surface, among others. Nevertheless, it is assumed that once the principle of control sensitivity is understood, additional control approaches can be introduced as alternative controllers of aircraft dynamics. Hence, the conventional airplane controls consider only the ailerons, elevators and rudders.

These are the control surfaces (CS) that produce the aerodynamic forces and moments due to a change in camber of their lifting surface (Schmidt, 1998). This assumption is interesting because each control surface is then responsible by the control one, and only one, axis of motion as indicated in Figure 2b. Figure 2b also defines the sign convention of the control deflections.

We define the angle of attack (AOA) α and the slideslip angle (SSA) β in terms of the velocity components. Let V be the Euclidean norm of the vector $\boldsymbol{V}$, the equations of α and β are given by

$$\alpha = \tan^{-1} \frac{w}{u} \tag{2}$$

$$\beta = \tan^{-1} \frac{v}{V}, \tag{3}$$

The aerodynamic forces and moments are generally defined in terms of dimensionless coefficients (or aerodynamic coefficients) C, the flight dynamic pressure Q, a reference area S and a characteristic length l as

$$X = C_x QS \qquad Y = C_y QS \qquad Z = C_z QS \tag{4}$$

$$L = C_l QS\, l \qquad M = C_m QS\, l \qquad N = C_n QS\, l \tag{5}$$

The dynamic pressure is actually related to the flow around the airframe by the air speed $V_a = ||\boldsymbol{V} - \boldsymbol{V}_w||$ (being $\boldsymbol{V}_w$ the wind speed) as

$$Q = \frac{1}{2}\rho V_a^2, \tag{6}$$

where ρ is the air density at sea level and 15°C, approx. $1,225\text{kg/m}^3$.

Alternatively, one may define the drag ($C_D = -C_x$) and the lift ($C_L = -C_z$) coefficients. To complement the aircraft-related definitions, the Appendix C states how aerodynamic coefficients vary in function of the abovementioned state variables.

2.1.3 Fixed-wing UAV dynamics equations

For a sake of simplicity and coherence, the complete set of non-linear motion equations for an aircraft is not presented in this text, but it is given in Appendix A. Indeed, these equations are widely presented in the literature. For control purposes, there is only a limited interest in these equations, although, it can be very important to understand the flight dynamics principles involved in their deduction, as it will introduced in Section 2.1.4.

2.1.4 Longitudinal and lateral-directional dynamics

Even though an aircraft can change its flight configuration very quickly due to aggressive maneuvers, establishing a linearized system relative to a fixed flight condition still very convenient to determine the airframe's small amplitude dynamics. Schmidt, 1998, does a complete deduction of these equations by stating all the relevant assumptions for the perturbation analysis. The representation of the linear system in presented in detail in Appendix B.

Normally, contributions to the Taylor series expansion of the axial force ΔX term (see Appendix B) due to the influence of β, $\dot{\beta}$, p and r will not be a factor for fixed-wing aircraft. The coupling with the lateral-directional related motions by the longitudinally oriented force (or moment) term would only be of import when considering a rotary-winged vehicle or high angles of attack (Schmidt, 1998).

The linearized equations can be thus separated into two sets on the assumption of a fixed-wing aircraft: longitudinal-directional (equations 30, 32 and 34) and lateral-directional (equations 31, 33 and 35).

Longitudinal dynamics The aircraft's linearized longitudinal dynamics normally will consist of two pairs of complex conjugate roots, which will correspond to a fast (short-period) mode and a slow (long-period) mode. The slow mode is also known as phugoid (Schmidt, 1998).

Lateral dynamics Unlike the longitudinal system, the lateral system will most often yield a complex conjugate pair of roots, known as Dutch roll mode, and two real roots related to the pure rolling and spiral modes (Schmidt, 1998). It will be seen later in Section 2.1.5 that the lateral-directional control input must be simplified (reduced) as UAVs do not possesses rudder control.

2.1.5 Production of aerodynamic forces and moments

Although one may find in the literature a complete description of how different phenomena influence in the aircrafts aerodynamics, for UAV dynamic control they are not specially relevant because it is almost impossible to take all those factors in account while doing real-time calculations. Thus, the main books that were referenced until now are not very useful anymore. This point is where the study of airplanes and fixed-wing drones somewhat drift apart, as the real-time computation

capability of small aircrafts become more and more constraining.

As discussed above, there are several components that contribute, along with its geometry/profile, to the production of lift and drag in an airplane, such as flaps, ailerons, elevators and rudders (control surfaces). However, a fixed-wing UAV does not have all these control surfaces. In fact, this kind of drones usually exhibits a pair of elevons that combine the functions of the elevators and the ailerons. Figure 3 shows how elevons can generate both roll and pitch movements.

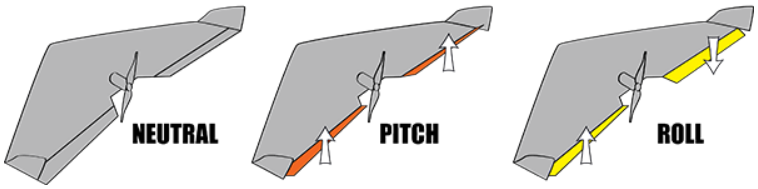


Figure 3: Elevons operating principle.

Even though the ailerons and elevators deflections are still denoted δ_a and δ_e on the following equations, they are only separated for the sake of clarity. In practice they are mixed into left and right elevons deflections, as

$$\delta_L = \delta_a + \delta_e \tag{7}$$

$$\delta_R = -\delta_a - \delta_e \tag{8}$$

2.1.6 Production of thrust

The thrust is the propelling force that balances mainly the aerodynamic drag on the vehicle. In an airplane it can be produced by a propeller, turbojet or a rocket engine (McCormick, 1994). However, in UAVs, only the use of propeller-based thrust generation is feasible.

In order to model an existing propeller accurately, it is necessary to examine the aerodynamics of blade by means of the "blade element theory". In overall terms, the propulsion is a simple force generated by a propeller coupled with a DC motor dynamics. A control variable $\delta_t \in [0, 1]$ is associated to the thrust/throttle control. For this end, two articles propose very interesting and somewhat different approaches.

First, Lesprier et al., 2015, suggests to model the motor's rotation speed ω_p and the thrust force $(F_x)_{prop}$ by two concatenated second-order polynomial interpolations. The DC motor time delay is taken in account by filtering δ_t by a first-order unitary DC gain filter with the same time constant of the servo motor. The drawback is that, whereas the motor speed model can be calculated using experimental data, it is more difficult to do this for thrust force.

In the other hand, Grano-Romero et al., 2016, develops a model by applying Bernoulli's principle, Kirchhoff's and Newton's law. In fact, the difference between upstream and downstream pressures of the propeller (which are function of ω_p and V_a) generates the thrust force, while the DC motor dynamics can be described by its consecrated model.

2.2 State of art on fixed-wing UAV control

This section outlines a state of art of modern control techniques applied to UAVs. Indeed, the main issues of UAV control are (1) the definition a reliable mathematical model (and its parameters), (2) the state estimation (usually by Kalman filtering) and (3) a robust control law implementation. However, before describe how recent works approach UAV control problems, one need to understand the principle of airplane stabilization. For instance, Nelson, 1989, in chapters 7 and 8, describes how root locus, frequency domain, state feedback and state reconstruction techniques can be applied to fixed-wing aircrafts.

As stated in Section 2.1.4, under certain assumptions, the dynamics of longitudinal and lateral-directional can be decoupled and then controlled separately. Figure 4 presents a general control diagram which can be applied to both systems. The cascade control loop is the most usual configuration, although the state variables fed back for inner and outer loops may vary, as it will be shown hereinafter. In a double loop cascade configuration, the inner loop response typically needs to be faster than the outer loop (Poksawat et al., 2016).

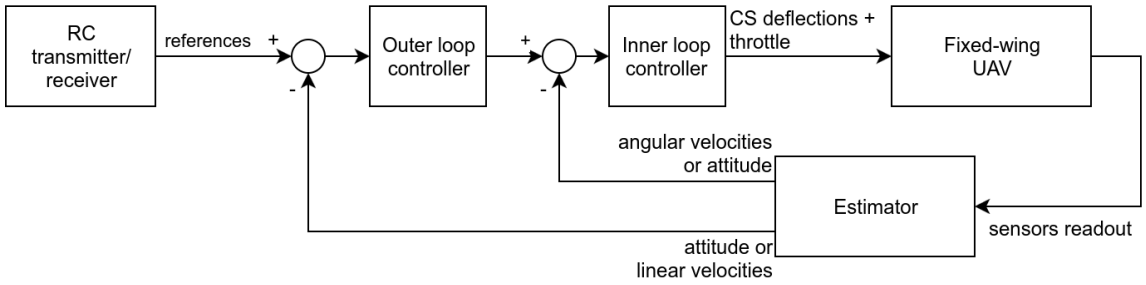


Figure 4: General control schematic.

A reliable mathematical model is required to simulate aircraft behavior, preflight test and control law synthesis. One approach is to calculate the stability derivatives using, for instance, computational fluid dynamics tools such as XFLR5[1] (Lesprier et al., 2015). The aerodynamic coefficients can be calculated in real-time or obtained experimentally and then given in terms of a look-up table, like in Callegati et al., 2016.

Sometimes an accurate UAV model is inaccessible for due to custom design of airframe or unavailable computational tools. To deal whit this, Jamil et al., 2015, and Ahmad et al., 2015, propose system identification approaches for, respectively, longitudinal and lateral dynamics.

First, Jamil et al., 2015, presents a grey-box system identification, i.e., order and structure of the linearized system were predefined by a reference model using UAV geometrical information, then the parameters were estimated based on recorded data. In the other hand, Ahmad et al., 2015, uses recorded data to estimate black box identification based transfer functions related to aileron and

[1] `http://www.xflr5.com`

rudder inputs. Both studies show that the identified models well predict the longitudinal and lateral dynamics modes of the UAVs.

Once a suitable model is defined, a commonly proposed estimation approach is the use of an Extended Kalman Filter (EKF) based on the non-linear kinematics, a measurement model and possibly also an aerodynamic model (Johansen et al., 2015). Indeed, the state of a UAV is composed by motion (position, velocity and acceleration) and attitude variables (angles, angular velocity and acceleration). To assure a proper closed loop behavior, a UAV must possesses a set of sensors that assure a reliable estimate of the state vector (de Souza Gonçalves and Rosa, 2017). They are mainly an Inertial Measurement Unit (IMU), a compass, a barometer, a Global Positioning System (GPS) and perhaps a Pitot tube (Johansen et al., 2015) or a thermal/optical flow camera (Helgesen et al., 2016).

Long and Song, 2009, claims a velocity, AOA and SSA estimation based on data fusion where the aerodynamic model is directly taken in account to calculate the forces, including wind and other disturbances. However, as stated in Johansen et al., 2015, Heinen et al., 2016, it is feasible to use only kinematics relationships with a Kalman Filter (KF) algorithm, avoiding an aerodynamic model. Attitude estimation is actually independent of the UAV configuration (fixed-wing or rotary-wing), and an Attitude Heading and Reference Systems (AHRS) can be based on an EKF or on a non-linear estimator.

Finally, from state estimates and systems models, the next challenging task is the design of control laws.

Different techniques are explored in the literature to obtain the inner and outer controllers (Figure 4). The main controller classes are introduced throughout the following paragraphs, along with particular applications to UAVs.

PID controllers Due to its simplicity and low computational intensity, proportional-integral-derivative (PID) controllers have become one of the most widely used controllers in autopilots. Poksawat et al., 2016, and Kumar and Jain, 2014, propose somewhat the similar approaches. Decoupled control loops which stabilize on specific state variable by a PID controller.

Whereas Kumar and Jain, 2014, uses Tyreus-Luyben tuning method without an explicit cascade architecture, Poksawat et al., 2016, uses an automatic tuning method for fixed wing UAVs with cascade control system structure for roll axis. In practice, the controlled variables vary according to the flight zone where the UAV is. Grano-Romero et al., 2016, switches between different controllers depending on if the UAV is on a descend, altitude hold, climb or take-off zone.

State feedback It the 80's Nelson, 1989, already explained how state feedback (or modal) control can be used to assure airplanes stability. Nonetheless, it does not assume a cascade loop, like nowadays. Indeed, a pole placement pole is suitable to control longitudinal-directional dynamics as shown in Lesprier et al., 2015. In similar way, to design a lateral-directional Linear-Quadratic Regulator (LQR) controller, in (Chollom et al., 2016), a multi-objective Particle Swarm Optimization

(PSO) algorithm is demonstrated for the optimization of the tuning of controller weighting matrices.

H_∞ and loop shaping Loop shaping controllers (or H_∞-based controllers) improve the robust stability properties of the control laws (Lesprier et al., 2015) and provides more straightforward design equations than modal/optimal control (Gadewadikar et al., 2007). H_∞ structure uses directly the available measurements, the input references and the disturbance estimates to minimize the H_∞ transfers between disturbance and longitudinal and lateral-directional states.

Adaptive/Predictive/non-linear control These are less explored controllers in the literature, mainly because of the success and simplicity of the previous approaches. The main advantages of theses controller classes is dealing with model imperfections and strong disturbances, where a robust controller is crucial. A essential robust control strategy based on an adaptive internal model control (AIMC) controller is used in Gao and Jia, 2017, for autonomous landing.

In the other hand, Callegati et al., 2016, achieves robust regulation of an fixed-wing UAV using a non-linear controller based on real aerodynamic coefficients. However, it employees a numerical algorithm to solve the output inversion problem which increase slightly the computing cost.

Finally, a third approach proposed by Triwiyatno et al., 2015, explore a fuzzy controller in order to control the longitudinal dynamics. Input to the fuzzy controller is the pitch angle of the complementary filter output and the output rate of the pitch sensor, while the output is a change in the control signal to adjust elevator servo deflection.

2.3 Open-source autopilots

In order to implement the abovementioned control strategies, various projects have been developed. Among them, the open-source solutions are particularly attractive because they deploy sophisticated hardware and software architectures to assure that any developer can build a controllable UAV. Some well-known projects are PX4 (PX4, 2017), Ardupilot (Ardupilot, 2017), Paparazzi UAV (Paparazzi UAV, 2017) and INAV (INAV, 2017).

In fact, PX4 is part of the Dronecode project (Dronecode, 2017), a shared and collaborative Linux Foundation-sponsored project working to build a common open-source platform for UAV development. The platform includes: PX4, MAVLink, and QGroundControl (PX4, 2017).

All abovementioned projects have some common visions and goals, which are recurrent among this class of project, like the strong emphasis on autonomous flight – rather than manual flight modes – and on an end-to-end workflow. Moreover, autopilots share a commonality in architecture, as their functionality can be divided into three distinct layers: real-time operating system, middleware and flight control.

However, one of the main reasons why Dronesmith Technologies, 2017, chose PX4 over Ardupilot is that PX4 is far more advanced, with more modern architecture: it supports a much larger number of peripherals, it can integrate with hybrid systems and it has a software-in-the-loop (SITL) simulation

that is much more developed and matured. For similar reasons, PX4 flight controller is used as the base to the present work implementation, yet the main one is the great support from Snapdragon Flight developer, Qualcomm. Indeed, Qualcomm is one of the “platinum members” of Dronecode project.

Thus, in Section 3, more details of this system architecture are given, as well as an overview of the components needed for a fully controllable prototype.

3 Architectural design

Based on the objectives stated in Section 1.1 and the concepts introduced in Section 2, this section proposes a system which can ensure a reliable flight of a fixed-wing UAV. Starting by a set of specifications that defines the bounds and the general features of the project, the system architecture is then introduced by setting up the modules that will be further developed in Section 4.

3.1 Requirements specification

In this section, the bounds of the projects are set. The following requirements ensure that the final platform will be able to respect all the project goals and that it has trustworthy properties regarding stability and extensibility for future implementations.

In this point, airframe characteristics, system properties and user interface are set by the following specifications:

- **Airframe characteristics:** As the flight controller is designed for Zangão V UAV, the system must have a minimal fixed-wing architecture, i.e. it must be controlled by two elevons and one propeller. Further, take-off and landing mechanisms must be integrated.
- **System integration and module independence:** It must be a fully integrated system. In other words, all devices must work as stand-alone modules, nonetheless, they must be able to retrieve all relevant signal/information from another module by a given communication path.
- **Stability and control capabilities:** It must ensure that the UAV can be stabilized. Although the final prototype might not be an stable version, a proper configuration and adaptation should ensure a reliable flight.
- **User interface and command:** One or more platforms must provide a final user interface both for inputs (remote control) and outputs (supervisory system).

That being said, in the following section, the main modules are described aiming at the deployment of the abovementioned features.

3.2 Design overview

Starting by the lower-level components and progressing towards the higher levels, the system can be designed by the following components:

- The control surfaces (CSs) – the elevons – that are controlled by two servo motors (i.e. position controlled), and a throttle/propeller, which is controlled in velocity by an Electronic Speed Control. As introduced in Section 2.1.2, it ensures a minimal fixed-wing configuration.
- Snapdragon Flight board, which features sensors, communication modules and signals outputs

to all peripherals. It will be the center device of this project, and it will distribute the information to all peripheral devices. It also runs a real-time operating system which allows the implementation of PX4 modules.

- The PX4-based autopilot. It defines all communication protocols (drivers), control/estimation algorithms and real-time tasks. It will be responsible to implement control and estimation techniques described in Section 2.2.
- Reference inputs can be sent by a radio control. Different devices are available in the industry. Once again, the chosen model is the one that is already used with Zangão V, namely, FrSky Taranis X8R.
- A ground control station that allows the user to define some flight parameters, perform calibrations, make flight plans and it displays useful flight data (telemetry). It will be the main output channel.

All these modules and their layer organization are summarized in Figure 5. Considering the OSI model (Microsoft, 2017), Snapdragon Flight implements lower-level layers, i.e. mainly physical and data link layers, while PX4 is responsible by higher-level layers.

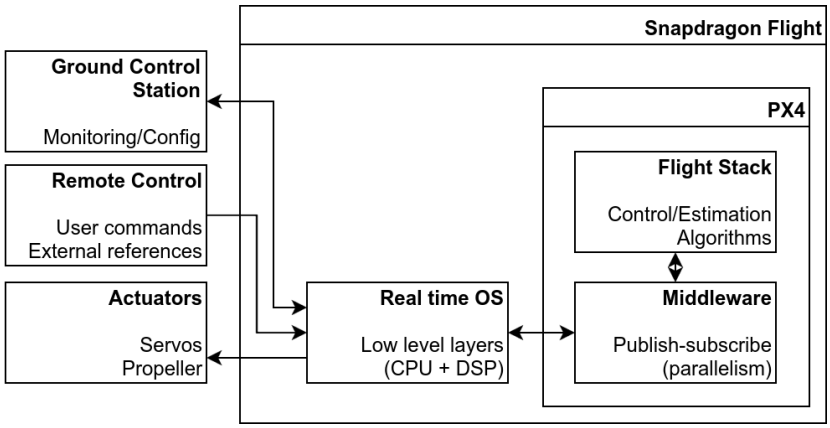


Figure 5: System layers distribution and peripherals.

One may also notice that arrows between two modules represent a (different) communication protocol. Even though a state-of-art on the communication protocols used for UAV applications is not presented in this text, the understanding of some protocols is very important, namely MAVLink, S.Bus, PWM, uORB, ULog, among others. Their role in the system architecture will be further described throughout sections 4 and 5.

4 Development

Once that the system requirements and its general architecture are well assimilated, the definition of which platforms will be responsible by the core functions can be addressed.

First, in Section 4.1, the core components of this application are introduced: PX4 firmware and Snapdragon Flight flight controller. Afterwards, the flight control techniques are described from a control engineering point-of-view, i.e. which control laws and estimation algorithms are implemented by PX4 flight stack (Section 4.2), and how efforts are mixed into control surfaces and throttle (Section 4.3). The implementation of a R/C device is described in Section 4.4, whereas the integration of all components is discussed in Section 4.5

4.1 Architecture overview

The goal of this section is to summarize the main features of Snapdragon Flight, PX4 autopilot and QGroundControl GCS. These are important concepts to understand the system architecture, since they will perform different tasks regarding data processing and communication.

4.1.1 Snapdragon Flight

The Snapdragon Flight board was introduced by Qualcomm targeting specifically autonomous vehicles platforms. It is based on a Qualcomm Snapdragon 801 processor, which features threes units

- CPU: Qualcomm Krait quad core (2.26 GHz)
- GPU: Qualcomm Adreno 330
- DSP: Qualcomm Hexagon

It means that applications can be built aiming different processing units, depending on its purpose. Also, the code is built using Hexagon proprietary development tools, which include C/C++ compilers, assembler, linker, debugger, among others.

Beside that, Hexagon DSP runs its own Real-Time Operating System (RTOS), called QuRT and Qualcomm provides a DSP abstraction layer (DSPAL) which ensures the POSIX compatibility. DSPAL includes all sensor interface and BAM flow control protocols. Indeed, Snapdragon Flight has the BLSP ports represented in Table 3 (Appendix F).

These ports can be used to interface the board with general peripherals, as it will be discussed in Section 4.3. For more technical specifications, please see Snapdragon Flight datasheet (Qualcomm Technologies, 2016).

Furthermore, the board features a set of flight management functions, including a sensors like:

- IMU + magnetometer: Invensense MPU-9250 9-Axis Sensor, 3x3mm QFN (via SPI interface).
- Barometric pressure sensor: Bosch BMP280.
- 640x480 Optic Flow Camera.
- 4K High Res camera.

Both cameras (optical flow and high resolution) are not exploited in this work. Nonetheless, this must be done in future works, even to improve control capabilities using the optical flow measurements.

4.1.2 PX4

As introduced in Section 2.3, autopilots follow a general structure. For the PX4/Snapdragon Flight combination the system architecture is represented in Figure 5. In other words, PX4 consists in two main layers:

- A Flight Stack, which is the set of guidance, navigation, control and attitude/position estimation algorithms for multirotor, fixed-wing and VTOL UAVs, summarized in Figure 6.
- A publish-subscribe based Middleware which interfaces sensors and peripherals to applications running the flight control. A uORB protocol based publish-subscribe scheme allows an asynchronous and fully parallelized message passing, meaning that any module can update and consume data from anywhere in a thread-safe mode.

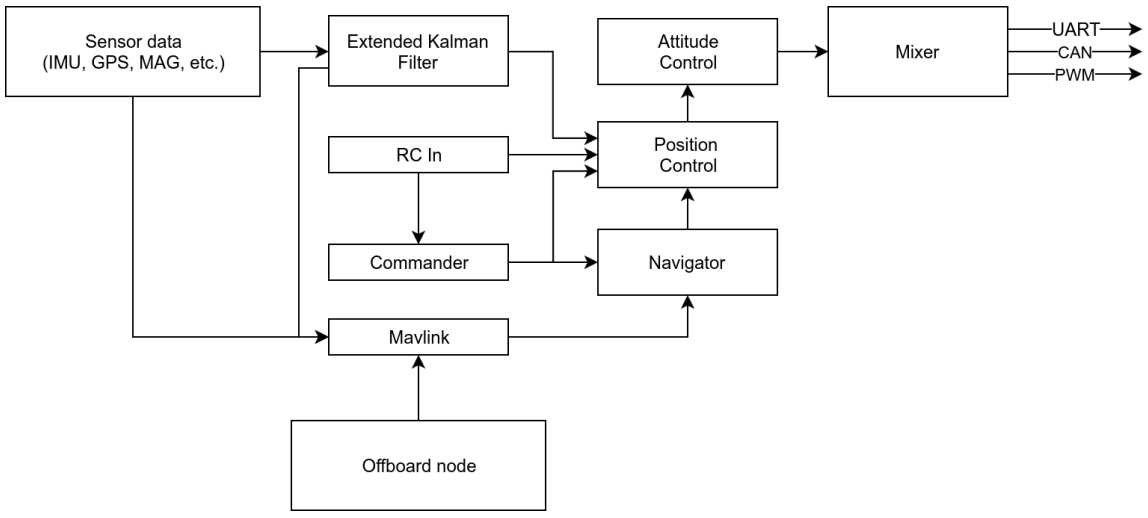


Figure 6: PX4 architecture.

Source: PX4, 2017

PX4 uses control groups to the core flight controls. An output group is one physical bus and has 8 commands ports which range mainly from -1 to 1. For instance, the so-called "Group 0" controls the actuators outputs, thus, this group will be the main focus. The outputs are then scaled by an output driver into an adequate wave format based on the output protocol (e.g. PWM).

In fact, PX4 firmware allows to configure J13 port into four PWM outputs among with supply and ground, as discussed in Appendix F. Since Zangão V needs three PWM outputs to flight (two elevons servos and the propeller), this port was chosen to control the UAV flight.

In practice, PX4 firmware source code is structured in the following directories:

- `/drivers`: These applications define interface channels with the peripherals.
- `/firmware`: Compilation guidelines.
- `/modules`: These applications are independent of the platform of choice, such as navigation and high level protocols (MAVLink/uORB).
- `/platforms`: PX4 main application itself and links with DSPAL. It manages all abovementioned applications.
- `/libs`, `/include`: External libraries and includes (`.h`).

Each application runs as an independent task and PX4 executes them using a pooling strategy depending on their priority. That being said, as applications can run in different units, a middleware is necessary to create a communication link between all modules. It uses uORB messaging: a publisher-subscriber based protocol.

The uORB module is started early on the bootup and it creates a set of topics that can be assign by any application. Each topic is then related to a meaningful data structure, such as attitude, R/C input, battery status, sensor data and many other useful information.

4.1.3 QGroundControl

The QGroundControl ground control station (GCS) is used as supervisory system, commander and navigator. It communicates with the embedded system by Micro Air Vehicle Communication Protocol (MAVLink), a very lightweight, header-only message marshalling library for micro air vehicles (QGroundControl, 2017). Indeed, the messages are translated by the software into uORB compatible data structures, which avoids PX4 depending directly on MAVLink related libraries.

QGroundControl allows to log the UAV states, as well as some operating controls like arming/disarming the UAV motors, and mission planing (the software turns them into lower-level navigation instructions).

Generally speaking, QGroundControl is able to plot and log all the MAVLink messages that are sent by PX4, such as orientation, position, motors outputs, sensors readouts and flight state. In addition, two other important features are:

- Motors arming: when that "arm" message is sent by MAVLink protocol, PX4 enables the output driver to send valid voltage signal to servos and ESC device. In the opposite direction, "disarm" command stops all outputs.
- IMU and R/C calibration: it allows to compensate eventual drifts and inconsistencies from nominal operation. These trim values are stored in PX4 configuration files.

4.2 Estimators and controllers

The following explanations are based on the PX4 general architecture scheme in Figure 6. In blocks "Extended Kalman Filter", "Attitude Control" and "Position Control", ECL implements both estimation and control algorithms.

Whereas the controllers and mixer are specific to a particular airframe, a fixed-wing in the present case, the higher-level management blocks like "Commander" or "Navigator". The navigator is responsible by the trajectories followed by the UAV while executing general tasks, like the flight plan. The commander is the general state machine. In fact, depending on the current state, the UAV applies different control strategy. This approach is constant across all modern autopilots, thus, it is hard to draw a visual representation of the control law.

However, while in "flight mode" (after take off and under normal circumstances) and controlled by R/C, the control schematic is similar to what is represented in Figure 4.

As introduced in Section 2, roll, pitch and yaw dynamics can be controller separately, thus, there are three independent control loops. Nonetheless, it is a generic control schematic. In other words, there is no yaw control, however, the PX4 flight stack must implement it anyway.

The output from the estimators (see Section 4.2.1) feedbacks the attitude angles to the outer loop and the angular velocities to the inner loop. Therefore, the references are

$$\boldsymbol{r} = \begin{pmatrix} \phi_r & \theta_r & \psi_r & \delta_{tr} \end{pmatrix}, \tag{9}$$

i.e. the Euler angles and throttle. Even though it would be natural to think that the throttle is directly bypassed to the output, some compensations are made aiming the UAV stabilization.

By the way of example, the roll control loop is analyzed. Pitch and yaw loops have similar characteristics. The outer controller is simply a P controller, therefore, the control law is

$$p_r(t) = \frac{1}{\tau_c}\left(\phi_r(t) - \hat{\phi}(t)\right) = \frac{1}{\tau_c}\tilde{\phi}(t), \tag{10}$$

where τ_c is the controller time constant and, thus, p_r is the reference of the inner controller whose control law is a PI-FF (PI controller with feedforward action on the rate reference). In other words, let $\tilde{p}(t)$ be the angular rate error (see Equation 1),

$$\delta_a(t) = k_p(t)\ \tilde{p}(t) + k_i \int_0^t \tilde{p}(\tau)\ d\tau + k_{ff}\ p_r(t). \tag{11}$$

The controller gains k_i and k_{ff} are constant, however, the particularity of the PI component is that

$k_p(t)$ is not constant overtime, but

$$k_p(t) = (k_{air}(t))^2 k_p^*, \tag{12}$$

where k_p^* is the conceptual proportional gain and $k_{air}(t)$ compensates the airspeed, i.e.

$$k_{air}(t) = \frac{V_{ref}}{V_a(t)} \tag{13}$$

where V_{ref} is a reference velocity.

In addition to these controllers, some compensations are implemented in order to avoid disturbances and improve robustness. Let δ^* be the ideal command

- The yaw (or rudder) control is increased by the roll feedforward component
$$\delta_r = \delta_r^* + k_{yaw}^{roll}\, \delta_a. \tag{14}$$
- When the UAV battery is low, the throttle control is decreased
$$\delta_t = k_{bat}\, \delta_t^*, \qquad k_{bat} \in [0, 1]. \tag{15}$$
- An anti-windup technique is implemented by resetting controllers integrators in some situations.
- Almost every variable has an upper and a lower limit (e.g. airspeed) to avoid unexpected behaviors and any instability due to calculation issues.

4.2.1 Extended Kalman Filter

As one may observe in the previous section, the controllers do not have access to the real values of angular speeds and attitude. Actually, the ECL deploys an EKF algorithm to process sensor measurements and provide an estimate of the following states:

- Quaternion defining the rotation from North, East, Down local earth frame to X,Y,Z body frame.
- Velocity at the IMU - NED (m/s).
- Position at the IMU - NED (m).
- IMU delta angle bias estimates - X,Y,Z (rad).
- IMU delta velocity bias estimates - X,Y,Z (m/s).
- Earth Magnetic field components - NED (Gauss).
- Vehicle body frame magnetic field bias - X,Y,Z (Gauss).
- Wind velocity - North, East (m/s).

The EKF runs on a delayed "fusion time horizon" to allow different time delays on each measurement relative to the IMU (Section 4.1.1). Data for each sensor is FIFO buffered and retrieved from the buffer by the EKF to be used at the correct time. The minimum set of measurements that the EKF needs is (1) the IMU data, (2) a source of yaw information (like magnetometer) and (3) a source of

height data (barometer). For more information, one might check the "ecl EKF" section of the PX4 Dev Guide (PX4, 2017).

In Appendix E, the Kalman filtering theory is briefly introduced, stating the main assumptions about the system representation and noises properties, nonetheless, an interested reader can refer to Rudolph Kalman's original work (Kalman, 1960). Further, Welch and Bishop, 2006 gives a straightforward introduction to KF and EKF theory.

It also important to know that EKF-based navigation approaches are almost a constant among all recent works and professional autopilots due to its relatively easy implementation and understanding. Namely, de Souza Gonçalves and Rosa, 2017, proposes the deployment of an embedded system for a fixed-wing UAV based on an EKF in order to achieve localization on a mission. Furthermore, not only PX4, but other solutions cited in Section 2.3 also use some kind of EKF-based strategy.

4.3 Mixers, propelling and control surfaces

Mixing means to take the controllers output – δ_a, δ_e, δ_r and δ_t – and translate them into actuator commands which control motors or servos. In our case, it means mix the roll and pitch/yaw commands into right and left elevons and axial external force $(\boldsymbol{F}_x)_{prop}$ (Appendix A) into a throttle motor command. The PWM output driver then scales it into PWM signal and sends it to the output port J13.

The Control Group 0 is organized as stated in Table 5 (Appendix G), thus, it will be convenient to associate elevons to PWM outputs 1 and 2, and motor to PWM/ESC output 4.

In the PX4 firmware, mixers are defined by text files (`.mix` extension). There are mainly three types of mixers: multirotor, simple and null mixers.

A simple fixed-wing mixer combines zero or more control inputs into a single actuator output. Inputs are scaled, and the mixing function sums the result before applying an output scaler. A null mixer is useful when the input is not used. Indeed, it is used for yaw control, which is irrelevant for Zangão V. Obviously, multirotor mixers are not used in this work.

For the general syntax of mixer files, see sections "Mixing and Actuators" and "Adding a new Airframe" of the the PX4 Dev Guide (PX4, 2017).

For attitude control, the strategy consists in mix roll and pitch controls into right and elevons (see Figure 3). In other words, the relation between the input and output deflections is

$$\begin{pmatrix} \delta_L \\ \delta_R \\ \delta_t \end{pmatrix} = \begin{pmatrix} k_a & k_e & 0 & 0 \\ k_a & -k_e & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \delta_a \\ \delta_e \\ \delta_r \\ \delta_t \end{pmatrix}, \tag{16}$$

where $k_a, k_e \in [0, 1]$ is the mixing gain for roll (aileron) and pitch (elevator).

For instance, the control surfaces can chosen to take maximum 60% deflection from roll ($k_a = 0,6$)

and 65% deflection from pitch ($k_e = 0,65$). A particularity of PX4 mixing strategy is that, as it is over-committed with 125% total deflection for maximum pitch and roll, it means the first channel (roll, here) has priority over the second channel/scaler (pitch). It avoids unnecessary saturation of the PWM outputs and increases control robustness. Finally, the thrust control is simply bypassed by the mixer (unitary mixer) to the PWM driver.

4.4 Radio control

The input vector $\boldsymbol{r}$ must be provided by some external source, i.e. either from the "Position Control" module (see Figure 6), when a flight plan is consider, or by a radio control (R/C) input.

Generally speaking, a R/C device uses radio waves to remotely send some kind of control to the UAV. From an fixed-wing UAV point-of-view, it sends roll, pitch, yaw and throttle commands, among with auxiliary commands, like parachute trigger, flight status (manual, take-off, landing), etc. Each command is sent over a specific channel. A R/C system is then composed by a transmitter, in which the user inputs his commands, and a receiver that decodes the radio waves into another electric signal modulation.

For this purpose, S.Bus is a serial protocol was introduced by Futaba (Futaba, 2017), but is commonly used by many FrSky products, too. Even though S.Bus is a proprietary protocol, several articles attempt to reverse-engineer it.

It is an one-wire serial asymmetric (3,3 V) protocol with 100000 baud rate. It uses a 8E2 inverted 25 bytes-long encoding. The general structure of a S.Bus frame is presented in Table 1. In regular mode, each frame takes about 14 ms to be transmitted.

Table 1: S.Bus frame

Byte(s)	Comment	Value
1	start byte	`0x0f`
2-23	18 channels, 11 bits per channel	-
24-25	2 stop bytes	`0x00`

To interface the R/C module with the autopilot, the `sbus_rc` driver is introduced. It might read the S.Bus output from the R/C receiver, process the information by validating the information feasibility, e.g. if R/C is loss or inactive during a longtime and then, publish the input into the `INPUT_RC` uORB topic. The `INPUT_RC` message structure is explained in more detail in Appendix H.

4.5 Prototyping and system integration

The abovementioned modules must then be assembled into a final prototype. This integration required a deep understanding of each component. Figure 7 shows the relation between all components.

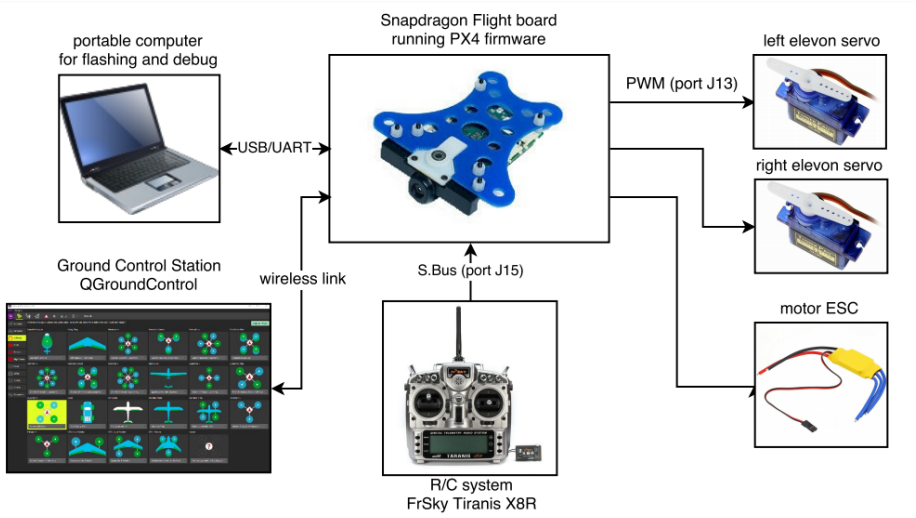


Figure 7: Final architecture overview

First, Snapdragon Flight board was configured for the new airframe. To do so, the existing `snapdragon_pwm_out` driver is updated, because, originally, this module was supposed to accept only multirotor configuration. as these are the regular configuration for Snapdragon Flight prototype. The modification consists in update the mixer format and the way that the commands are sent to the PWM driver (port J13).

For the R/C, FrSky Taranis X8R R/C system is the chosen one. It is a very common R/C transmitter and thus, a lot of online support is available. The X8R R/C receiver is connected to the port J15 (`/dev/tty-1`). Pins 1 and 5 are respectively 5V and GND, while the TX signal is sent by pin 3.

It is important to notice that, as S.Bus is an inverted protocol, a simple voltage inverter circuit is introduced between the receiver and Snapdragon Flight.

To take into account the new airframe, QGroundControl GCS must be rebuilt from source in order to add into it the parameters for the new aircraft. Once it was done, the software should accept and display a new configuration, meaning that PX4 can be entirely configured from QGroundControl.

Indeed, there are several parameters that can be personalized into the PX4 firmware, from controllers gain, until flags to run or not certain modules. In this section, only the controllers gains (attitude and Euler's rates) are of interest. In Appendix I, there are the main parameters, which were set in PX4 configuration file for both roll and pitch control.

Since the platform runs in an asynchronous mode, different modules can run in different frequencies. The sensor readout is performed between 1 and 8 kHz. The data validation module rate (which checks the readout consistency) is from 250 to 500 Hz. The controller tasks run in lower frequencies, the inner loop controllers and estimators update at 200 Hz, while the ESC/PWM (pulse width modulation) outputs are at 60 Hz.

5 Results

For debug and development purposes, it is possible to connect over Snapdragon Flight board using a Portable Operating System Interface (POSIX) and a USB connection by the Android Debug Bridge (ADB) tool. It allows the user to execute the POSIX modules/functions directly over the console, including push and pull files into the board's memory.

In addition, when connected to the POSIX side of the OS, the interaction with the QuRT (DSP) is enabled using the `qshell` application. On Snapdragon Flight, `mini-dm` interface outputs debug messages directly from the DSP modules.

For our standard prototype, applications can be classified into two groups presented in Table 2. The `start` flag just explicitly starts the module. Another default flags are `stop`, `status`, etc. Each of these applications runs as an independent task either in the POSIX or in the QuRT part.

Table 2: QuRT and POSIX applications

QuRT	POSIX

`uorb start`
`qshell start`
`gps start -d /dev/tty-4`
`df_hmc5883_wrapper start`
`df_mpu9250_wrapper start`
`df_bmp280_wrapper start`
`df_trone_wrapper start`
`sensors start`
`commander start`
`ekf2 start`
`land_detector start fixedwing`
`mc_pos_control start`
`mc_att_control start`
`snapdragon_pwm_out start`
`sbus_rc start`
	`uorb start`
	`muorb start`
	`logger start -t -b 200`
	`dataman start`
	`navigator start`
	`mavlink start -u 1`

That being said, `mini-dm` allows us, for instance, to oversee all the tasks that are being run, as well as the instantiated uORB topics, as shown in Figure 8. For the sake of clarity, just a few uORB topics are shown this figure, however, by default, there are at least 54 topics.

It also shows that no critical error, such as segmentation fault, occurs during a standard execution of the PX4 main module (i.e. during its boot-up and manual flight mode) and the system have not seen any unexpected interruption.

In a similar way, an ADB terminal can monitor the POSIX tasks. An important module is the

```
[08500/00]  01:13.035  HAP:61:DF Devices:  0573  vdev.cpp
[08500/00]  01:13.035  HAP:61:   /dev/mag0  0583  vdev.cpp
[08500/00]  01:13.035  HAP:61:   /dev/imu0  0583  vdev.cpp
[08500/00]  01:13.035  HAP:61:   /dev/baro0  0583  vdev.cpp
[08500/00]  01:13.035  HAP:61:   /dev/trone0  0583  vdev.cpp
[08500/00]  01:13.035  HAP:61:Files:  0609  vdev.cpp
[08500/00]  01:13.035  HAP:61:Active Tasks:  0314  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:   dspal      2097663  0318  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:   hpwork     2097661  0318  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:   lpwork     2097660  0318  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:   wkr_hrt    2097659  0318  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:   qshell     2097658  0318  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:   gps        2097657  0318  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:   sensors    2097656  0318  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:   commander  2097655  0318  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:   ekf2       2097653  0318  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:   mc_pos_control 2097652  0318  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:   mc_att_control 2097651  0318  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:   snapdragon_pwm_out_main 2097650  0318  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:   sbus_rc_main 2097649  0318  px4_qurt_tasks.cpp
[08500/00]  01:13.035  HAP:61:Devices:  0593  vdev.cpp
[08500/00]  01:13.035  HAP:61:   /obj/_obj_  0599  vdev.cpp
[08500/00]  01:13.035  HAP:61:   /obj/qshell_req0  0599  vdev.cpp
[08500/00]  01:13.035  HAP:61:   /obj/parameter_update0  0599  vdev.cpp
[08500/00]  01:13.035  HAP:61:   /obj/gps_inject_data0  0599  vdev.cpp
[08500/00]  01:13.035  HAP:61:   /obj/sensor_accel0  0599  vdev.cpp
[08500/00]  01:13.035  HAP:61:   /obj/sensor_gyro0  0599  vdev.cpp
[08500/00]  01:13.035  HAP:61:   /obj/sensor_mag0  0599  vdev.cpp
[08500/00]  01:13.035  HAP:61:   /obj/input_rc0  0599  vdev.cpp
```

Figure 8: Example of `mini-dm` debug tool output.

`logger`, which logs the MAVLink messages into a ULog file in board's memory (or alternatively into a SD card). It is very useful for further analysis of estimators, controllers and actuators behaviors.

ULog is the file format used for logging system data. The format is self-describing, i.e. it contains the format and message types that are logged. It can be used for logging device inputs (sensors, etc.), internal states (CPU load, attitude, etc.) and print log messages. The format uses Little Endian for all binary types.

For the results presented in sections 5.1 and 5.2, a regular test is proposed:

- Angle references are kept at zero (null setpoint), i.e. the control laws tend to recover the flat flight mode;
- First, until approximately 20 s, a roll movement keeping pitch almost null is performed;
- After, a roll maneuver is performed, and pitch angles is kept close to zero;
- The logs from these movements are then uploaded to a personal computer to be further compared to the control surfaces and throttle behavior.

5.1 Estimators

To analyze the ECL estimates (and further, the actuators outputs), log files are imported into MATLAB software. It allows to post-process these signals and plot them into convenient formats.

For example, Figure 9 presents the Euler angles' estimates for the abovementioned regular test.

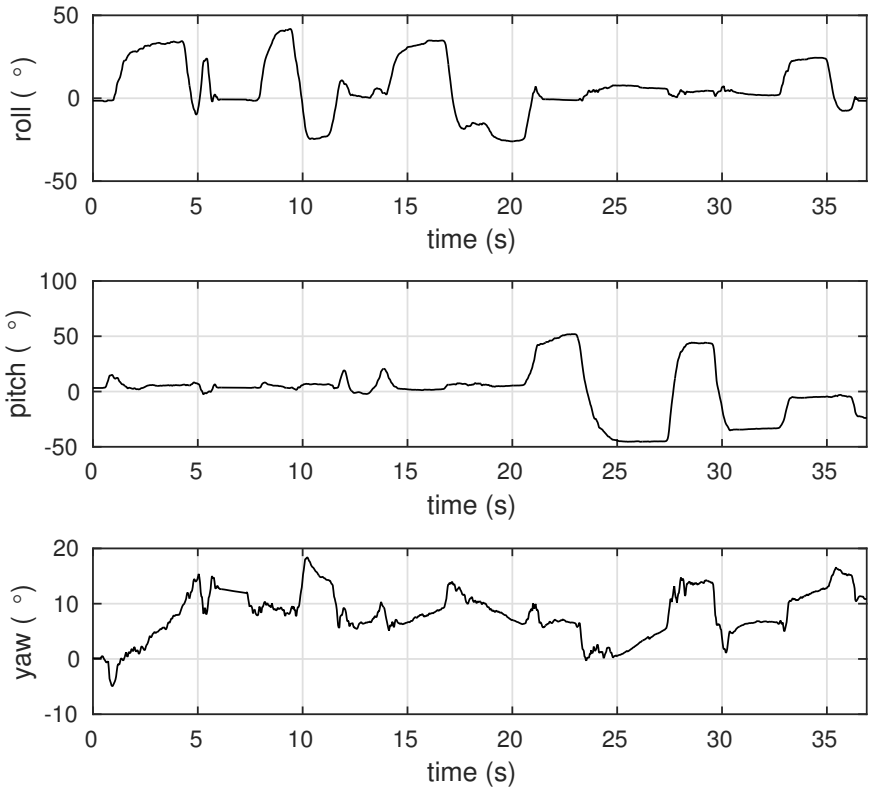


Figure 9: Euler angles estimates for a regular test.

It is hard to quantify the estimation error of the values shown in Figure 9 since there is no ground truth measurement to compare nor a proper testbench was designed. Nonetheless, a qualitative analysis lead us to assume that the estimators work fine and the estimates are a reliable representation of the board orientation.

It is important to notice that these estimates feature almost no noise and are very stable while in steady state, yet no delay between the actual maneuver and the EKF response is observed. These characteristics are good indicators that no abnormal phenomenon is occurring during the estimation algorithm execution.

Furthermore, the yaw estimates are slightly noisier than the roll and pitch components. This is due to the IMU operating principle which also requires magnetometer observations to provide trustworthy values.

5.2 Actuators outputs

For the same maneuvers presented in Figure 9 (regular test), Figure 10 shows the actuators outputs for Group 0 (see Table 5) which are later sent to port J13.

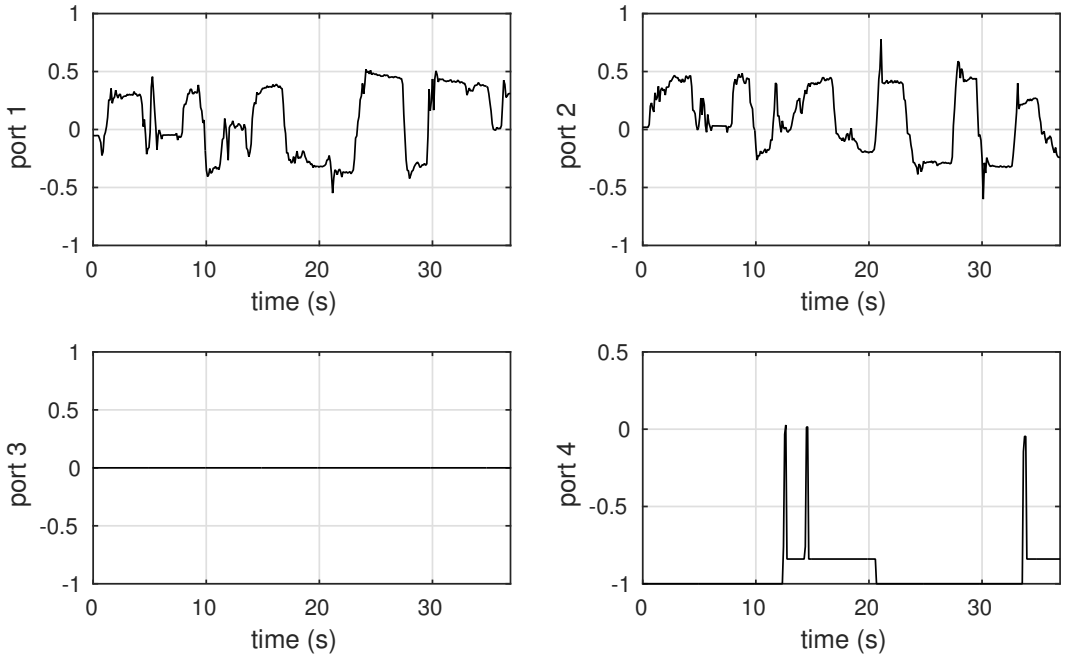
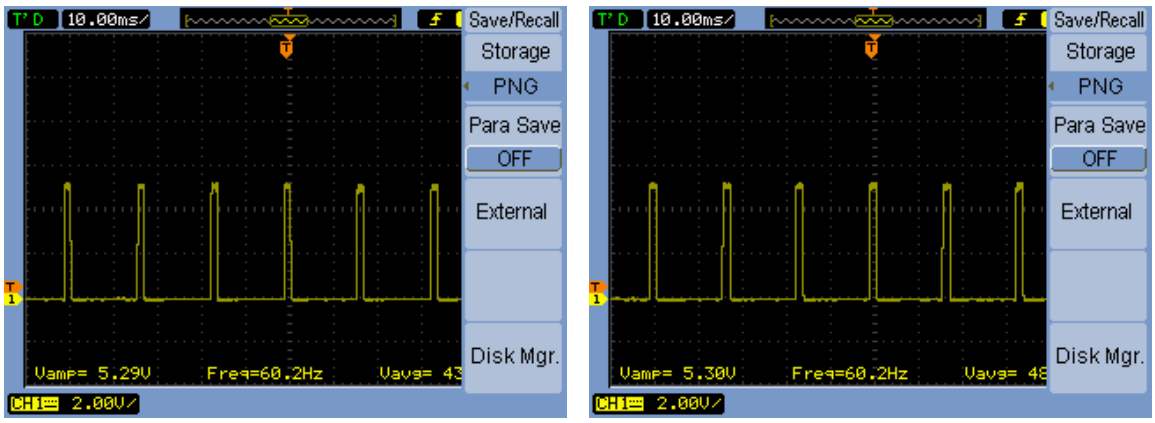


Figure 10: Outputs for a regular test.

As it should be expected, port 3 is always zero, as it represents a null mixer (Section 4.3). Ports 1 and 2 are related, respectively, to left and right elevons. Is important to notice that, on the assumption that the two elevon servos are physically reversed, the pitch input is inverted between the two servos. Thus, the scaling factor for roll inputs is adjusted to implement differential travel for the elevons. Finally, the throttle command is directly output into port 4.

In the other hand, one can observe the behavior of PWM signals. Figure 11 shows an example of the J13 port output for pin 2 (left servo). It is also to observe that the output frequency is 60 Hz, just like configured in the source code.

When servos are disarmed, the output driver sets a PWM value of 900. Whereas, when armed, PWM outputs range from 1000 to 2000, describing the full rotation of its axis, being 1500 the value of the centered position. It is the reason why, as it will be seen in the following section, the R/C inputs use the same scale.



(a) PWM output when servos are disarmed. (b) PWM output for a negative pitch attitude.

Figure 11: J13 port pin 2 sample output.

5.3 R/C integration

For the R/C integration, first, a S.Bus driver interface is needed. In Figure 12, a sample of the S.Bus protocol signal in exhibited. The cursors show the baud rate frequency of $1/|\Delta X| = 100$ kHz and the signal amplitude $\Delta Y = 4,24$ V.

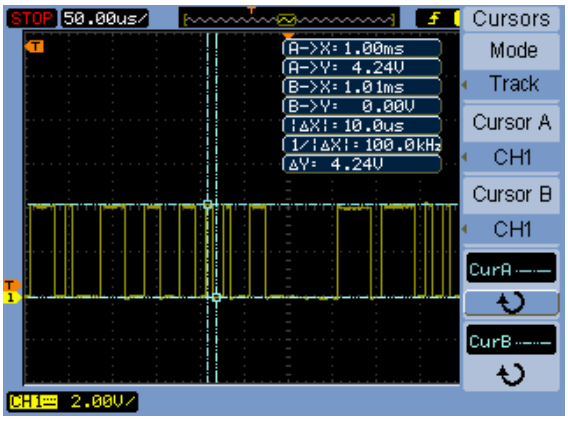


Figure 12: Sample S.Bus protocol signal.

A major inconvenient during the project execution took place during this integration since Snapdragon Flight does not accept the 100000 baud rate, but only 115200 and 67600 baud rate which are divisors of the board fundamental frequency.

Thus, a workaround had to be introduced to synchronize S.Bus device and Snapdragon Flight. a middleware microprocessor was necessary between them. The microprocessor receives serial data in a 100000 baud rate and repeats it into a 115200 baud rate waveform, in order to ensure that the original signal is not sub-sampled.

Although, this solution can not be sustained for the final version, as the introduction of an extra microprocessor just to establish the S.Bus interface is not feasible.

Once this middleware is implemented, by monitoring the `INPUT_RC` uOBR topic, it can verified that the R/C commands are well published (Figure 13).

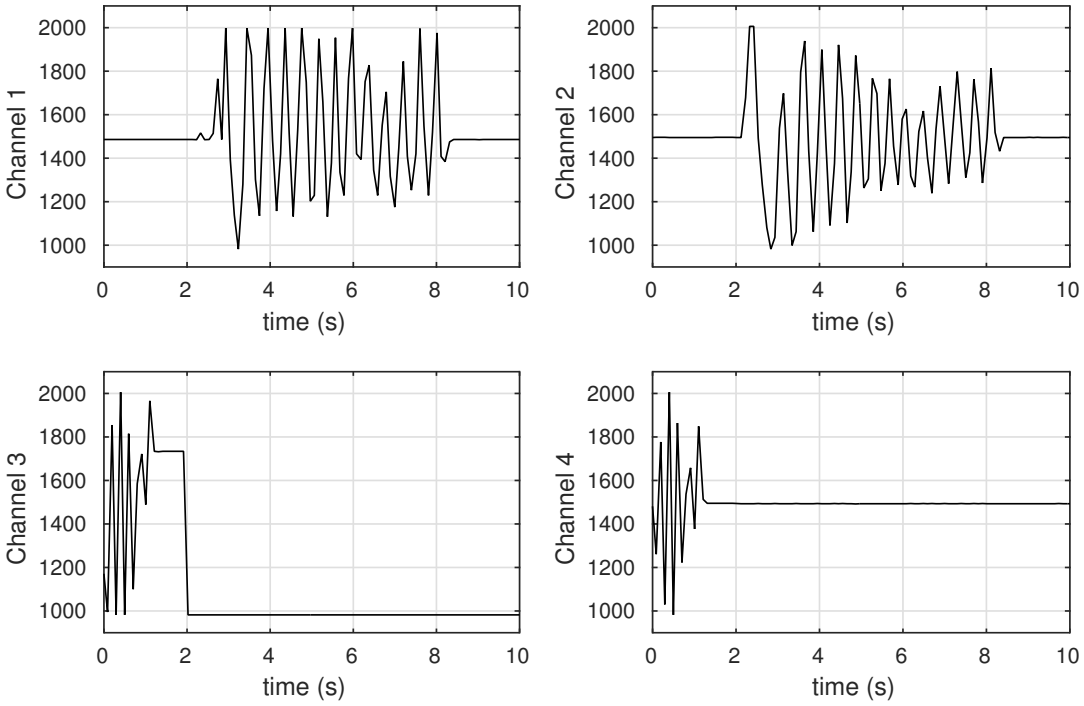


Figure 13: R/C inputs

In this test, no particular procedure was followed, and FrSky Taranis was configured to send roll, pitch, throttle and yaw command in channels 1, 2, 3 and 4, respectively. In other words, channels were just tested by moving them from minimum to maximum value repeatedly. As mentioned above, inputs range from 1000 to 2000, and, when the commands are centered, a constant value of 1500 is given.

5.4 Prototype

Finally, all modules were assembled into a final prototype within Zangão V aircraft. Figure 14 shows the components that were fixed at the airframe structure.

Figure 14 also highlights the main parts, namely (1) Snapdragon Flight board, (2) Snapdragon Flight power circuit, (3) BEC, (4) FrSky 8XR R/C module, (5) ESC device and (6) Arduino

Mega microcontroller, used to interface the S.Bus protocol. In Appendix J, more precisely in tables 9 and 10, all these modules are presented along with the input/output connections A to E.

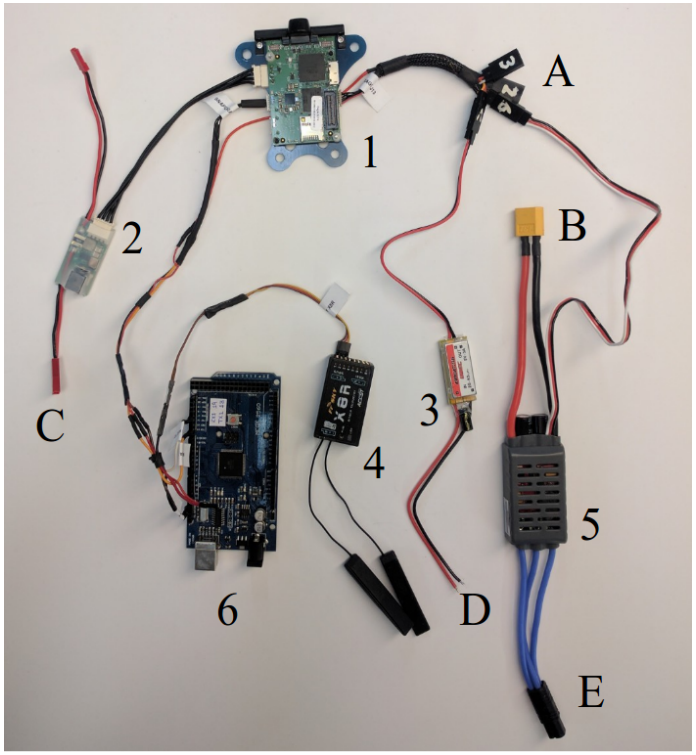


Figure 14: Zangão V embedded electronics.

The top view of Figure 15 shows both control surfaces and their respective actioning parts, namely, (1) right and (3) left elevons, as well as (2) the propeller. The servos are actually inside the UAV's wings, which can be removed to ease its shipping and boxing. The propeller blades can also be unfixed from the motor axis. For security reasons, this is very helpful during the earlier tests.

The best configuration should be to match both board and airframe centers of mass. It will avoid any correction on the estimators implementation. However, there is no room for the board in this position. Hence, a workable workaround is to place the board over the roll axis, but in the UAV's rear side. It causes a misalignment on the plan (y, z), however, a higher mass at the aircraft rear side has less unwelcome consequences regarding the airframe stability (Nelson, 1989). Hereafter, the IMU measurements must be adjusted using the Parallel axis theorem (or Huygens–Steiner theorem) to retrieve trustworthy UAV velocities.

The prototype seems to work fine, and the system responds to R/C commands like expected. The elevons behavior is coherent as they go in different ways when a roll maneuver is required and they go in the same way for pitch commands.

Although the reliable behavior, it is still hard to say whether this model can flight or not, because

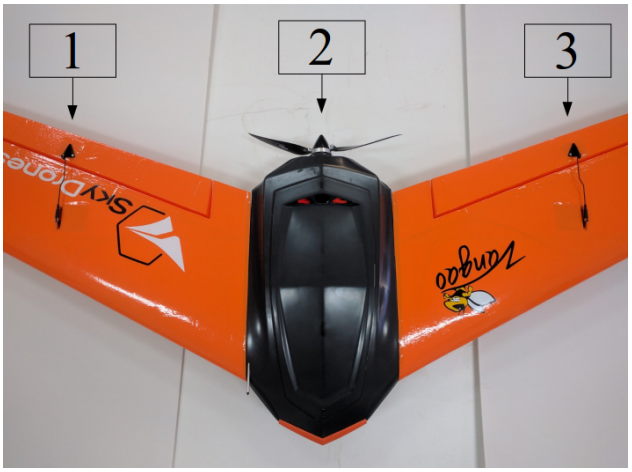


Figure 15: Top view of the aircraft.

several tests to properly tune all estimators and controllers are still needed. Another alternative could be a further modeling of the UAV to retrieve the parameters of the equations of motion, and then, a design of suitable controllers. These approaches will be discussed later in Section 6.2.

To confirm that QGroundControl interface is working fine, it can be observed that the right parameters are set and that "Zangao Skydrones" airframe is selectable in the Airframe's menu (Figure 16). In addition, the GCS can be used to calibrate IMU and R/C. The user should calibrate the platform as often as possible to ensure that any drift from the original configuration do not deteriorate the flight performance.

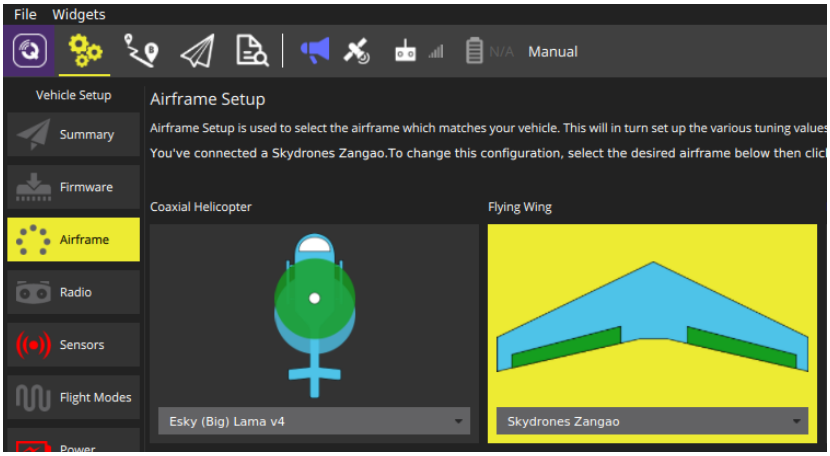


Figure 16: QGroundControl airframe's menu showing the Zangão V airframe.

Previously, even though it was not explicitly mentioned, all modules were powered from the electric grid by 5 V sources. Yet, for the new prototype, the power sources are Li-ion batteries and then Battery eliminator circuits (BECs) are used.

Another further tests include aerodynamic stability due to wind (i.e. controllers robustness), IMU drifts, power isolation (e.g. from ground loops), estimation coherence with ground truth measurements, among others.

6 Conclusions

As defined during the project specification phase (see Section 3), the project was mainly aiming at a fixed-wing configurable prototype that was not necessarily functional regarding its flight capability, but which could be piloted from an external source, and which could potentially fly after proper configurations and adjustments.

Regarding these main objectives, the project was successful. In the end, a fully integrated system is presented: R/C commands are well received by the autopilot, the servos have the exact behavior determined by the mixer outputs, the ground control station can receive and log all MAVLink messages, arm the motors and can be used to calibrate sensors and R/C, etc.

In addition, no critical error was identified during the firmware execution in QuRT, i.e. any unexpected error message that could lead the system to an emergency stop (due to a segmentation fault, for instance) was identified.

The interface microprocessor introduced to ensure the S.Bus connection can not be avoided yet, since only the open-source part of the code can be edited. A feasible workaround might modify the DSPAL, which is a set of proprietary libraries from Qualcomm.

6.1 Lessons learned and main contributions

Foremost, this project represented an extensive research work in terms of autopilots architecture, programming/code structure and protocols. These are very useful concepts for an engineer that intends to act on the UAV market.

Moreover, organizational aspects were also very important. A reliable set of specification and a prior understanding of the upcoming activities eased the project progress, beside that, a great autonomy enabled its rapid advancement.

In order to develop such a complex system, a strong technical background and experience in control theory were great assets, as it avoided many misunderstandings during the implementation.

In a world that is constantly going more autonomous, the understanding of “intelligent vehicles”, their capabilities and limitations, is somewhat primordial as this is a tendency in almost every single engineering field.

6.2 Future works

Once a basic, well-executed and configurable prototype is done, a broad range of opportunities and activities becomes achievable.

It is possible to envisage projects aiming at practical and theoretical studies, regarding testbenches for the prototype, control capabilities enhancement, sensing and imaging, development of new

algorithms, fields of application, development of high-level applications, among others.

To exemplify the huge amount of new horizon, just a few essential future work that would ensure an high quality high-end product are listed below:

1. **Further modeling** of the UAV to determine or, in some cases, enhance the precision of the UAV model. Some parameters like the most part of aerodynamics coefficients, weight and moments of inertia are already known.
2. **Estimators (namely EKF) weighting** to improve the estimation performance. At the moment, there is no precise information about the quality of the system estimates. Further, a review of the estimation strategy could also be considered.
3. **Optical flow and GPS integration** to improve the speed estimation and allow position estimation for flight plans. A proper optical flow algorithm could enhance significantly the stability, since they give a good speed estimate if fused with an altitude source (barometer or LIDAR).
4. **Further controllers tuning** based on a reliable model and, further, the study of different control strategies. Many examples are given in the previous state-of-art (see Section 2.2).
5. **Flight plans execution**, which requires an stable communication with QGroundControl and also a good tuning of the position controller. Normally, once all the lower-level functions are implemented and GPS signal is available flight pans should be easily performed
6. **Withdraw of the middleware microcontroller** for S.Bus interface. Indeed, it is not a feasible solution in a long-term project. A correction inside the DSPAL could be a reliable workaround.
7. **Integration of 4K and professional cameras** for image acquisition. In fact, multispectral cameras are already used in Zangão V's original configuration.
8. **Optimization of the components disposal** inside the airframe. It will more than likely improve the flight performance.
9. **Flight tests** to study control robustness, stability, vibrations, disturbance (e.g. wind drag) rejection, flying range (i.e. battery lifetime), behavior regarding adverse situations, etc.

The abovementioned activities are nonetheless just "low-level" applications, since they aim to ensure a reliable and robust flight performance and to provide convenient flight monitoring.

However, it is important to notice that one of the main upsides of Snapdragon Flight board is its tiny size and its high processing capacity. Indeed, nowadays, Snapdragon 801 processor is used in high-end portable devices, such as smartphones.

Thus, future applications could also include the development of new algorithms, especially for imaging and image processing, which can exploit Qualcomm Adreno GPU in very useful ways.

Just to name a few, nowadays, obstacle avoidance and SLAM techniques are widely under research. Thus, they could also be very interesting algorithms to be implemented in Snapdragon Flight GPU.

7 References

Ahmad, U., Ahsan, M., Qazi, A. I., Choudhry, M. A., Dec 2015. Modeling of lateral dynamics of a uav using system identification approach. In: 2015 International Conference on Information and Communication Technologies (ICICT). pp. 1–5.

Ardupilot, 2017. Ardupilot development site.
URL http://ardupilot.org/dev/index.html

Becker, M., Sheffler, D., April 2016. Designing a high speed, stealthy, and payload-focused vtol uav. In: 2016 IEEE Systems and Information Engineering Design Symposium (SIEDS). pp. 176–180.

Briod, A., Zufferey, J.-C., Floreano, D., 2016. A method for ego-motion estimation in micro-hovering platforms flying in very cluttered environments. Autonomous Robots 40 (5), 789–803.
URL http://dx.doi.org/10.1007/s10514-015-9494-4

Callegati, F., Naldi, R., Melega, M., Marconi, L., June 2016. Robust nmiscar control of miniature fixed-wing uavs. In: 2016 European Control Conference (ECC). pp. 2584–2589.

Chollom, T. D., Ofodile, N., Ubadike, O., Aug 2016. Application techniques of multi-objective particle swarm optimization: Aircraft flight control. In: 2016 UKACC 11th International Conference on Control (CONTROL). pp. 1–6.

de Avila Wieczorek, I., 2017. A power line detection algorithm to support a fine grain UAV movemtn guidance. Master's thesis, PPGEE-UFRGS, Porto Alegre.

de Souza Gonçalves, E., Rosa, P. F. F., March 2017. Cointegration analysis for imu in a fixed-wing uav. In: 2017 IEEE International Conference on Industrial Technology (ICIT). pp. 755–760.

Dronecode, 2017. Dronecode: The open source uav platform.
URL https://www.dronecode.org/

Dronesmith Technologies, 2017. Why we chose px4 (vs apm) as luci's default firmware.
URL https://medium.com/@Dronesmith/why-we-chose-px4-vs-apm-as-lucis-default-firmware-ea39f4514bef

Flores, D. A., Saito, C., Paredes, J. A., Trujillano, F., Feb 2017. Multispectral imaging of crops in the peruvian highlands through a fixed-wing uav system. In: 2017 IEEE International Conference on Mechatronics (ICM). pp. 399–403.

Futaba, 2017. S.bus system.
URL http://www.futabarc.com/sbus/

Gadewadikar, J., Lewis, F., Subbarao, K., Chen, B. M., May 2007. Attitude control system design for unmanned aerial vehicles using h-infinity and loop-shaping methods. In: 2007 IEEE International Conference on Control and Automation. pp. 1174–1179.

Gao, J.-z., Jia, H.-g., 2017. Adaptive internal model control research in autonomous landing phase for a fixed-wing uav. CEAS Aeronautical Journal 8 (1), 45–51.
URL http://dx.doi.org/10.1007/s13272-016-0216-1

Grano-Romero, C., García-Juárez, M., Guerrero-Castellanos, J. F., Guerrero-Sánchez, W. F., Ambrosio-Lázaro, R. C., Mino-Aguilar, G., June 2016. Modeling and control of a fixed-wing uav powered by solar

energy: An electric array reconfiguration approach. In: 2016 13th International Conference on Power Electronics (CIEP). pp. 52–57.

Groÿekatthöfer, K., Yoon, Z., May 2012. Introduction into quaternions for spacecraft attitude representation. Tech. rep., Technical University of Berlin, Berlin, Germany.

Habib, A., Xiong, W., He, F., Yang, H. L., Crawford, M., Jan 2017. Improving orthorectification of uav-based push-broom scanner imagery using derived orthophotos from frame cameras. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 10 (1), 262–276.

Heinen, A. L., Moriceau, N., Michel, N., 3 2016. Trajectory tracking by a multi-rotor for radar applications: further modeling, control laws development and observer design. Tech. rep., Ecole CentraleSupélec, Gif-sur-Yvette, France.

Helgesen, H. H., Leira, F. S., Johansen, T. A., Fossen, T. I., June 2016. Tracking of marine surface objects from unmanned aerial vehicles with a pan/tilt unit using a thermal camera and optical flow. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 107–117.

Hoffer, N. V., Coopmans, C., Jensen, A. M., Chen, Y., 2014. A survey and categorization of small low-cost unmanned aerial vehicle system identification. Journal of Intelligent & Robotic Systems 74 (1), 129–145. URL http://dx.doi.org/10.1007/s10846-013-9931-6

INAV, 2017. Inav: Drone autopilot software.
URL http://inavflight.com/

Jamil, M. A., Ahsan, M., Ahsan, M. J., Choudhry, M. A., Dec 2015. Time domain system identification of longitudinal dynamics of a uav: A grey box approach. In: 2015 International Conference on Emerging Technologies (ICET). pp. 1–6.

Johansen, T. A., Cristofaro, A., Sørensen, K., Hansen, J. M., Fossen, T. I., June 2015. On estimation of wind velocity, angle-of-attack and sideslip angle of small uavs using standard sensors. In: 2015 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 510–519.

Jones, D., Nov 2005. Power line inspection - a uav concept. In: 2005 The IEE Forum on Autonomous Systems (Ref. No. 2005/11271). pp. 8 pp.–.

Kalman, R. E., 1960. A New Approach to Linear Filtering and Prediction Problems. Transactions of the ASME–Journal of Basic Engineering 82 (Series D), 35–45.

Kumar, N., Jain, S., 6 2014. Identification, Modeling and Control of Unmanned Aerial Vehicles. International Journal of Advanced Science and Technology 67 (1), 1–10.

Lee, S., Bang, H., Lee, D., June 2016. Predictive ground collision avoidance system for uav applications: Pgcas design for fixed-wing uavs and processor in the loop simulation. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 1287–1292.

Lesprier, J., Biannic, J. M., Roos, C., Sept 2015. Modeling and robust nmiscar control of a fixed-wing uav. In: 2015 IEEE Conference on Control Applications (CCA). pp. 1334–1339.

Long, H., Song, S., Oct 2009. Method of estimating angle-of-attack and sideslip angel based on data fusion. In: 2009 Second International Conference on Intelligent Computation Technology and Automation. Vol. 1. pp. 641–644.

McCormick, B. W., 1994. Aerodynamics, Aeronautics, and Flight Mechanics, 2nd Edition. Wiley.

Microsoft, 2017. The OSI Model's Seven Layers Defined and Functions Explained.
URL https://support.microsoft.com/en-us/help/103884/the-osi-model-s-seven-layers-defined-and-functions-explained

Naldi, R., Marconi, L., Aug 2009. Optimal transition maneuver for a class of v/stol aircrafts. In: 2009 European Control Conference (ECC). pp. 1842–1847.

Nelson, R. C., 1989. Flight stability and automatic control. McGraw-Hill.

Orfanus, D., de Freitas, E. P., Eliassen, F., April 2016. Self-organization as a supporting paradigm for military uav relay networks. IEEE Communications Letters 20 (4), 804–807.

Pamadi, B. N., 2004. Performance, Stability, Dynamics, and Control of Airplanes. AIAA.

Paparazzi UAV, 2017. Paparazzi: The free autopilot.
URL http://wiki.paparazziuav.org/wiki/Main_Page

Poksawat, P., Wang, L., Mohamed, A., 2016. Automatic tuning of attitude control system for fixed-wing unmanned aerial vehicles. IET Control Theory Applications 10 (17), 2233–2242.

PX4, 2017. Px4 development guide.
URL https://dev.px4.io/

QGroundControl, 2017. Qgroundcontrol user guide.
URL https://docs.qgroundcontrol.com/en/

Qualcomm Technologies, May 2016. Snapdragon™ Flight™ Kit : Developer's Edition. Intrinsyc, rev. 1.4.

Schmidt, L. V., 1998. Introduction to Aircraft Flight Dynamics. AIAA Education Series. American Institute of Aeronautics and Astronautics.

Secretaria de Desenvolvimento e Competitividade Industrial (SDCI), Jan 2016. Estudo Sobre a Indústria Brasileira e Europeia de Veículos Aéreos Não Tripulados. Tech. rep., Diálogos Setoriais União Europeia - Brasil, Brasília, Brazil.

Sharma, H., Bhujade, R., Adithya, V., Balamuralidhar, P., Feb 2014. Vision-based detection of power distribution lines in complex remote surroundings. In: 2014 Twentieth National Conference on Communications (NCC). pp. 1–6.

Triwiyatno, A., Syafei, W. A., Prakoso, T., Setiyono, B., Wijaya, A. P., Oct 2015. Design of self balancing pitch control in fixed wing unmanned aerial vehicle with fuzzy logic controller. In: 2015 2nd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE). pp. 211–215.

Welch, G., Bishop, G., July 2006. An Introduction to the Kalman Filter. Tech. rep., University of North Carolina, Chapel Hill, United States of America.

Zhou, G., Yuan, J., Yen, I. L., Bastani, F., Sept 2016. Robust real-time uav based power line detection and tracking. In: 2016 IEEE International Conference on Image Processing (ICIP). pp. 744–748.

Appendixes

A Airframe equations of motion

In this section we will mainly state the equations of motion for a fixed-wing aircraft. Although, there will be no discussion about which assumptions are made to achieve each simplification or equation system.

For this purpose, both Nelson, 1989, and Schmidt, 1998, do an extensive description of how rigid body dynamics and aerodynamic equations can be deduced.

The equations of motion are a consequence of the conservation laws for linear and angular momentum for a rigid body. Indeed, the first equations are directly derived from Newton's second law ($\boldsymbol{F} = d(m\boldsymbol{V})/dt$) and are related with the force and moment vectors applied on the airplane center of mass.

The aerodynamic force is decomposed in external (propulsion/drag/lift) and gravitational (weight) components, respectively,

$$F_x - mg\ \sin\theta = m(\dot{u} + qw - rv) \tag{17}$$

$$F_y + mg\ \cos\theta\sin\phi = m(\dot{v} + ru - pw) \tag{18}$$

$$F_z + mg\ \cos\theta\cos\phi = m(\dot{w} + pv - qu) \tag{19}$$

The external forces $(\boldsymbol{F})_{ext} = \begin{pmatrix} F_x & F_y & F_z \end{pmatrix}^T$ are then a sum of three different components

$$(\boldsymbol{F})_{ext} = (\boldsymbol{F})_{prop} + (\boldsymbol{F})_{drag} + (\boldsymbol{F})_{lift} \tag{20}$$

Moreover, some suitable assumptions are made to model these three components:

1. Propulsion acts only on x-axis, i.e.,
$$(\boldsymbol{F})_{prop} = \begin{pmatrix} (F_x)_{prop} & 0 & 0 \end{pmatrix}^T. \tag{21}$$
2. Drag acts over axis x and y by the drag and lateral coefficients, i.e.,
$$(\boldsymbol{F})_{drag} = -\begin{pmatrix} C_D QS & C_y QS & 0 \end{pmatrix}^T. \tag{22}$$
3. The lift is the only force that reacts to the weight, thus
$$(\boldsymbol{F})_{lift} = \begin{pmatrix} 0 & 0 & C_L QS \end{pmatrix}^T. \tag{23}$$

Although, the weight does not result in any moment about the center of mass. Therefore, the moments equations are

$$L = I_x\dot{p} - I_{xz}\dot{r} + qr(I_z - I_y) - I_{xz}pq \tag{24}$$

$$M = I_y\dot{q} + rq(I_x - I_z) + I_{xz}(p^2 - r^2) \tag{25}$$

$$N = -I_{xz}\dot{p} + I_z\dot{r} + pq(I_y - I_x) + I_{xz}qr \tag{26}$$

The relationship between the angular velocity in the body frame $\boldsymbol{\Omega}$ and the Euler angles rate $\dot{\boldsymbol{\Phi}}$

is given by Equation 27. By integrating the angles rates overtime, one can determine the aircraft orientation.

$$\begin{pmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix} = \underbrace{\begin{pmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta\end{pmatrix}}_{\mathfrak{T}(\mathbf{\Phi})}\begin{pmatrix}p\\ q\\ r\end{pmatrix} \tag{27}$$

Finally, the absolute velocity is related to the velocity in the body frame by a frame rotation matrix $\Re(\mathbf{\Phi})$ as

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{pmatrix} = \underbrace{\begin{pmatrix}\cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi & \cos\phi\sin\theta\cos\psi+\sin\phi\sin\psi\\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi & \cos\phi\sin\theta\sin\psi-\sin\phi\sin\psi\\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta\end{pmatrix}}_{\Re(\mathbf{\Phi})}\begin{pmatrix}u\\ v\\ w\end{pmatrix} \tag{28}$$

The rotation matrix depends though directly on the Euler's angles $\mathbf{\Phi}$.

B Linearized small-disturbance equations

For the linearization process, any state variable $\xi(t)$ is decomposed in a constant mean value ξ_0 and a null mean perturbation $\Delta\xi(t)$ that varies overtime, i.e.

$$\xi(t) = \xi_0 + \Delta\xi(t) \tag{29}$$

In this work, we are just interested in the final linearized "small perturbation equations of motion", which are

$$\Delta X(t) - (mg\cos\theta_0)\,\theta(t) = mV_0\,(\dot{u}/V_0) \tag{30}$$

$$\Delta Y(t) + (mg\cos\theta_0)\,\phi(t) = mV_0\,(\dot{\beta} + r) \tag{31}$$

$$\Delta Z(t) - (mg\sin\theta_0)\,\theta(t) = mV_0\,(\dot{\alpha} - q) \tag{32}$$

$$\Delta L(t) = I_x\dot{p} - I_{xz}\dot{r} \tag{33}$$

$$\Delta M(t) = I_y\dot{q} \tag{34}$$

$$\Delta N(t) = I_z\dot{r} - I_{xz}\dot{p} \tag{35}$$

The aerodynamic force and moment terms remaining in equations 30 to 35 can be established using Taylor series expansions of the perturbation variables as typically indicated by

$$\Delta X(t) = \Delta X\left[(u/V_0), \alpha, \dot{\alpha}, q, \delta, \beta, \dot{\beta}, p, r, \ldots\right] \tag{36}$$

$$\Delta Y(t) = \Delta Y\left[(u/V_0), \alpha, \dot{\alpha}, q, \delta, \beta, \dot{\beta}, p, r, \ldots\right] \tag{37}$$

and so on. The coefficients of the Taylor series are then called "stability derivatives" (or simply "partial derivatives terms"). These derivatives can be determined either by analytic, semi-empirical, computational fluid dynamics or experimental methods (Pamadi, 2004).

As discussed in Section 2.1.3, as longitudinal and lateral-directional dynamics can be decoupled, we might use any linear system representation to describe them. For instance, Nelson, 1989 uses mainly a set of transfer functions, while Schmidt, 1998, prefers a state-space representation.

When a state-space system is employed, the general structure becomes

$$\mathfrak{I}_n \dot{\boldsymbol{x}} = \mathfrak{A}_n \boldsymbol{x} + \mathfrak{B}_n u \tag{38}$$

Therefore, for the **longitudinal-directional system**, the state vector is defined as $\boldsymbol{x} = \begin{pmatrix} u/V_0 & \alpha & q & \theta \end{pmatrix}^T$ along with a single control term $u = \delta_e$. In fact, sometimes a thrust control variable δ_t is added, and then we have $\boldsymbol{u} = \begin{pmatrix} \delta_e & \delta_t \end{pmatrix}^T$.

In the other hand, we defined the **lateral-directional system** state vector by $\boldsymbol{x} = \begin{pmatrix} \beta & p & \phi & r \end{pmatrix}^T$, whereas the control input is given as $\boldsymbol{u} = \begin{pmatrix} \delta_r & \delta_a \end{pmatrix}^T$. However, for fixed-wing UAVs, the control term is reduced, as they do not actually have a rudder control, then $u = \delta_a$.

C Aerodynamic coefficients for a fixed-wing UAV

The force and moments coefficients as a function of the aircraft state variables are given by the equations below (Lesprier et al., 2015)

$$C_L = C_{L0} + C_{L\alpha}\alpha + C_{Lq}\frac{q}{V_a} + C_{L\delta_e}\delta_e \tag{39}$$

$$C_y = C_{y\beta}\beta + C_{yp}\frac{p}{V_a} + C_{yr}\frac{r}{V_a} + C_{y\delta_a}\delta_a \tag{40}$$

$$C_D = C_{D0} + C_{DC_L}C_L^2 \tag{41}$$

$$C_l = C_{l\beta}\beta + \frac{L_a}{V_a}(C_{lp}p + C_{lr}r) + C_{l\delta_a}\delta_a \tag{42}$$

$$C_m = C_{m0} + C_{m\alpha}\alpha + \frac{L_o}{V_a}C_{mq}q + C_{m\delta_e}\delta_e \tag{43}$$

$$C_n = C_{n\beta}\beta + \frac{L_a}{V_a}(C_{np}p + C_{nr}r) + C_{n\delta_a}\delta_a \tag{44}$$

All "C_i" coefficients of these Taylor series expansions are the abovementioned stability derivatives.

D Quaternions and Euler angles

A quaternion $\boldsymbol{q}$ is a 4×1 vector which consists of a scalar part s and a vector part $\boldsymbol{v}$ and it represents a coordinate transformation around a normalized rotational axis $\boldsymbol{e}$. Let γ be the transformation

angle, we can write

$$\boldsymbol{q} = \begin{pmatrix} s \\ \boldsymbol{v} \end{pmatrix} = \begin{pmatrix} q_s \\ q_x \\ q_y \\ q_z \end{pmatrix} = \begin{pmatrix} \cos\gamma/2 \\ \boldsymbol{e}\ \sin\gamma/2 \end{pmatrix} \tag{45}$$

Quaternions are a complex mathematical theory, however Groÿekatthöfer and Yoon, 2012 provide a straight-forward and practical introduction to quaternion operation and calculation for rigid-body attitude representation.

Some useful operations with quaternions are

- Conjugate: $\boldsymbol{q}^* = \begin{pmatrix} s \\ -\boldsymbol{v} \end{pmatrix}$
- Multiplication: $\boldsymbol{q}_1\boldsymbol{q}_2 = \begin{pmatrix} s_1 s_2 - \boldsymbol{v}_1^T\boldsymbol{v}_2 \\ s_1\boldsymbol{v}_1 + s_1\boldsymbol{v}_2 + \boldsymbol{v}_1 \times \boldsymbol{v}_2 \end{pmatrix}$, where $\times$ is the cross product.
- Norm: $|\boldsymbol{q}| = \sqrt{\boldsymbol{q}\boldsymbol{q}^*} = \sqrt{\boldsymbol{q}^*\boldsymbol{q}} = \sqrt{q_s^2 + q_x^2 + q_y^2 + q_z^2}$
- Inverse: $\boldsymbol{q}^{-1} = \dfrac{\boldsymbol{q}^*}{|\boldsymbol{q}|}$

In terms of these operations, the transformation of a vector $\boldsymbol{v}$ from coordinate system A to B is defined, as function of the transformation quaternion $\boldsymbol{q}_{BA}$, as

$$\begin{pmatrix} 0 \\ \boldsymbol{v}_B \end{pmatrix} = \boldsymbol{q}_{BA} \begin{pmatrix} 0 \\ \boldsymbol{v}_A \end{pmatrix} \boldsymbol{q}_{BA}^{-1} \tag{46}$$

D.1 Conversion between representations

To obtain the equivalent quaternion representation, we proceed by three axes transformation as defined in Equation 45. Therefore

$$\boldsymbol{q} = \begin{pmatrix} \cos\psi/2 \\ 0 \\ 0 \\ \sin\psi/2 \end{pmatrix} \begin{pmatrix} \cos\theta/2 \\ 0 \\ \sin\theta/2 \\ 0 \end{pmatrix} \begin{pmatrix} \cos\phi/2 \\ \sin\phi/2 \\ 0 \\ 0 \end{pmatrix} \tag{47}$$

In the opposite direction, the Euler angles can be obtained from the quaternions by

$$\begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} = \begin{pmatrix} \arctan\left(\dfrac{2(q_s q_x + q_y q_z)}{1 - 2(q_x^2 + q_y^2)}\right) \\ \arcsin\left(2(q_s q_y - q_z q_x)\right) \\ \arctan\left(\dfrac{2(q_s q_z + q_x q_y)}{1 - 2(q_y^2 + q_z^2)}\right) \end{pmatrix} \tag{48}$$

E Kalman filtering principles

A Kalman Filter (KF) is a powerful estimation algorithm named after Rudolf E. Kálmán. This filter addresses the general problem of trying to estimate the state $\boldsymbol{x}(k) \in \mathbb{R}^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$\boldsymbol{x}(k+1) = F\ \boldsymbol{x}(k) + G\ \boldsymbol{u}(k) + \boldsymbol{v}(k) \tag{49}$$

$$\boldsymbol{z}(k) = H\ \boldsymbol{x}(k) + \boldsymbol{w}(k) \tag{50}$$

by means of observations of the output $\boldsymbol{y}(k) \in \mathbb{R}^m$. Matrices F, G and H have coherent dimensions that make the above multiplications possible. In this section, for a sake of simplicity, matrices are represented by uppercase letters.

Noises $\boldsymbol{v}$ and $\boldsymbol{w}$ are stochastic variables that model process and measurement uncertainties. They are assumed to be independent, i.e $E[\boldsymbol{v}^T \boldsymbol{w}] = 0$, and Gaussian white noises. Their covariance matrices are respectively Q and R.

The general filter evolution equation (see $\hat{x}(n|m)$) are then

$$\hat{\boldsymbol{x}}(k+1) = F\ \hat{\boldsymbol{x}}(k) + G\ \boldsymbol{u}(k) + K\ (\boldsymbol{z}(k) - \hat{\boldsymbol{z}}(k)) \tag{51}$$

$$\hat{\boldsymbol{z}}(k) = H\hat{\boldsymbol{x}}(k) \tag{52}$$

where K is the (Kalman) filter gain and $\hat{\boldsymbol{z}}(k)$ is the set of available measurements.

Let $P(k)$ be the estimation error covariance matrix, the Kalman Filter is the optimal filter that minimizes the mean of the quadratic estimation error. In a nutshell, the Kalman filter can be written as recursive set of equations composed by two distinct phases: *prediction* and *observation* (equations 53 to 57).

Prediction

$$\hat{\boldsymbol{x}}(k|k-1) = F\hat{\boldsymbol{x}}(k-1|k-1) + G\ \boldsymbol{u}(k) \tag{53}$$

$$P(k|k-1) = FP(k-1|k-1)F^T + Q \tag{54}$$

Observation

$$K(k) = P(k|k-1)H^T\ \left(H\ P(k|k-1)\ H^T + R\right)^{-1} \tag{55}$$

$$\hat{\boldsymbol{x}}(k|k) = \hat{\boldsymbol{x}}(k|k) + K(k)\ \ (\boldsymbol{z}(k) - H\hat{\boldsymbol{x}}(k|k-1)) \tag{56}$$

$$P(k|k) = (I_n - K(k)H)\ P(k|k-1) \tag{57}$$

These equations can be extended to nonlinear time invariant systems, leading to the Extended Kalman Filter (EKF) (Welch and Bishop, 2006).

F Snapdragon Flight ports

In this section, the BLSP ports (Table 3) are presented. Moreover, the J13 port pins are described in Table 4.

Table 3: Snapdragon Flight BLSP ports

Device	Port	Usage
`/dev/tty-1`	J15	R/C device
–	J14	Power
`/dev/tty-2`	J13	Actuators PWM/ESC
`/dev/tty-3`	J12	Gimbal
`/dev/tty-4`	J9	External GPS

Table 4: Port J13 pins in PWM ESC outputs configuration

Pin	ESC	Comment
1	none	ESC/servos already have supply
2	1	PWM signal
3	2	PWM signal
4	3	PWM signal
5	GND	GND for all ESCs
6	4	PWM signal

G PX4 Control Group 0 ports

The Control Group 0, as well as it ports are presented in Table 5.

Table 5: Control Group 0 command ports.

Port	Variable	Range
0	roll	$[-1 \dots 1]$
1	pitch	$[-1 \dots 1]$
2	yaw	$[-1 \dots 1]$
3	throttle	$[0 \dots 1]$ or $[-1 \dots 1]$
4	flaps	$[-1 \dots 1]$
5	spoilers	$[-1 \dots 1]$
6	airbrakes	$[-1 \dots 1]$
7	landing gear	$[-1 \dots 1]$

H uORB messages format

In Table 6, we describe in detail the structure associated to the `INPUT_RC` uORB topic.

Table 6: `INPUT_RC` message

Variable	Description
timestamp	last valid reception time
channel counter	number of channels actually being seen
RSSI	receive signal strength indicator
failsafe mode	explicit failsafe flag
R/C lost	R/C receiver connection status
R/C lost frames counter	number of lost RC frames
R/C total frames counter	number of total RC frames
R/C PPM frame length	length of a single PPM frame. Zero for non-PPM systems
input source	input source
values	measured pulse widths for each of the supported channels

I Controllers parameters

Tables 7 and 8 present controller parameters for roll and pitch control, respectively.

Table 7: Roll controller parameters

Parameter	Value	Comment	Unit
`FW_RR_FF`	0.6000	k_{ff}	%/rad/s
`FW_RR_I`	0.1000	k_i	%/rad
`FW_RR_IMAX`	0.2000	integrator saturation	
`FW_RR_P`	0.0020	k_p	%/rad/s
`FW_RSP_OFF`	0.0000	ref offset	deg
`FW_R_LIM`	50.0000	max roll	deg
`FW_R_RMAX`	70.0000	max roll rate	deg/s
`FW_R_TC`	1.8000	τ_c	s

All parameters have intuitive names, for instance, the `FW_` prefix means that these parameters are only used by fixed-wing controllers, which are different from the multirotor controllers, whereas, `RR` and `PR` stand for "roll rate" and "pitch rate" parameters, respectively.

J Prototype modules and connections

Tables 9 and 10 refer to numbering introduced in Figure 14[2].

[2]More information about Snapdragon Flight development kit, see `https://www.intrinsyc.com/getting-started-intrinsyc-qualcomm-snapdragon-flight-kit/`

Table 8: Pitch controller parameters

Parameter	Value	Comment	Unit
FW_PR_FF	0.3500	k_{ff}	%/rad/s
FW_PR_I	0.1200	k_i	%/rad
FW_PR_IMAX	0.4000	integrator saturation	
FW_PR_P	0.0010	k_p	%/rad/s
FW_PSP_OFF	0.0000	ref offset	deg
FW_P_LIM_MAX	45.0000	max pitch	deg
FW_P_LIM_MIN	-45.0000	min pitch	deg
FW_P_RMAX_NEG	60.0000	max negative pitch rate	deg/s
FW_P_RMAX_POS	60.0000	max positive pitch rate	deg/s
FW_P_TC	1.5000	τ_c	s

Table 9: Zangão V electronic devices.

Number	Comment
1	Snapdragon Flight board without GPS and WLAN antennas
2	Power adapter board (APM)
3	5 V - 3 A BEC
4	FrSky X8R R/C receiver and voltage inverter
5	ESC for throttle
6	Arduino Mega middleware microcontroller

Table 10: Prototype connections.

Letter	Comment
A	Outputs to servos (2-3) and ESC (6), along with voltage input (V)
B	From battery
C	From external source/battery
D	From external source/battery
E	To propeller engine