From control requirements to PIL test: development of a structure to Autopilot implementation

Marcelo H. Santos 1, Neusa M. F. Oliveira 2, and Roberto d’Amore 3

1 IACIT, São José dos Campos – SP 12240-420 Brazil
2 Electrical Engineering Department, ITA, São José dos Campos – SP 12228-900 Brazil
3 Electrical Engineering Department, ITA, São José dos Campos – SP 12228-900 Brazil

Corresponding author: Neusa M. F. Oliveira (e-mail: neusa@ita.br).

ABSTRACT This article presents the development of Autopilot (AP) blocks for a fixed-wing aircraft and the implementation of a Processor In the Loop (PIL) test platform. The AP entails the control, guidance, and navigation blocks. The control block design employed classical control theory considering a Piper ¼ aircraft model, and the MILF-8785C design requirements standard. The PIL test platform is composed of an AP running in a microcontroller and a simulated aircraft on a PC (Personal Computer). The results of three missions carried out on the PIL platform are presented. In these missions, the performed maneuvers are such that lateral and longitudinal controls are required. In all three cases, all predefined waypoints are achieved, indicating the success of the developed AP project. The AP test implemented in a microcontroller that can be embedded onto an aircraft for a real flight mission indicates the usefulness of the developed PIL platform.

INDEX TERMS Autopilot, PIL, RPA, Unmanned Aerial Systems.

I. INTRODUCTION

An Unmanned Aerial Vehicle (UAV) can be autonomous or remotely controlled and, in several countries, a human pilot on the ground ready to take on the mission via radio control is required. However, despite these restrictions, research related to autonomous vehicles is being carried out, whether related to the physical structure of the vehicle [1], [2], either to the applications of these vehicles, such as the study of the feasibility of using UAVs in VOR (VHF Omnidirectional Range) inspection [3], the use of UAVs as base stations to offer wireless connectivity for users without the necessity of new terrestrial infrastructure [4], application of UAVs in precision agriculture [5]. ITS (Intelligent Transportation Systems) is another example of an application applied to flying accident report agents, flying roadside units (RSUs), flying speed cameras, flying police eyes, and flying dynamic traffic signals [6]. The coordination of activities of these vehicles in a swarm can be also mentioned: developing HIL platforms to test classical UAV swarm control application scenarios [7], exploring the use of distributed control in UAV swarm for the purpose of terrain mapping [8], developing design and software-in-the-loop implementation of adaptive formation controllers for fixed-wing unmanned aerial vehicles (UAVs) [9], or applying coordinated control method to continuously track a moving target using a group of UAVs [10], to name a few.

In order to operate autonomously, a UAV must have an AutoPilot (AP) for carrying out a mission. Several commercial AP are available on the market, but the addition of new features is not an easy task. In purpose to accelerate the development of new systems, references [11] and [12] propose the use of commercial APs with high-level APs to add new features to a matured system.

Once a new AP or subsystem is developed, bench testing is required before field testing. Processor-In-the-Loop (PIL) tests provide an effective platform for the development and test of the embedded system [13], [14]. PIL should be adopted whenever possible to validate hardware and software under the most realistic conditions.

This work presents the design of an AP and the tasks involved in its development: aircraft modeling, aircraft control design, and the implementation of the guidance.
algorithm and control loops. It also addresses the development of a test platform for an AP for a fixed-wing UAV.

On a test platform, it is possible to use two types of aircraft models: an aircraft simulated by a flight simulator, such as the X-Plane [15], [16], or an aircraft simulated by its mathematical model such as, for instance, in [9], in which the mathematical models of fixed wing UAVs are simulated in MATLAB, for implement an adaptive formation control of these vehicles. Reference [17] uses an accurate nonlinear model of the aircraft dynamics to investigate an approach of a loop-shaping for distressed fixed-wing aircraft experiencing control loss due to surface or power failure. Reference [18] proposes an AP and Path Planning for a UAV in flying wing configuration. Analytical aerodynamics computer programs were used to find the aerodynamic coefficients and, with them, the mathematical equations of the UAV were simulated. Reference [19] shows a prototype of a gun-launched micro air vehicle system (GLMAV) developed for long-distance observation capabilities. Based on wind-tunnel measurements, a model of the transient phase (the phase in which the UAV is changing its configuration from projectile to a rotary-wing micro air vehicle) is built, allowing the design and simulation of control laws for this phase. In both cases, obtaining the model of a specific aircraft requires extensive work, either through the use of design tools (for X-Plane) or mathematical equations. In this work, the identification of an aircraft model was not carried out. A mathematical model available in the literature [20] was used and relevant information is presented about the mathematical model of a fixed-wing aircraft and its use on a test platform as proposed. In this text, the term "aircraft" refers to the aerial platform (on which the AP will be shipped), while "UAV" refers to the platform with the AP onboard, capable of performing an autonomous mission. In many situations, however, these terms may be interchangeable. The term "Unmanned Aerial System" is used for the complete system consisting of aircraft, AP, payload, and ground station.

Depending on the problem to be solved and the platform to be used, there are several proposals for the control loops. For instance, [21] presents an overview of controller design based on chosen specifications for a mini-helicopter. Reference [22] investigates the control problem of a quadricopter considering uncertain parameters, external disturbances, and input saturation. The solution proposes different control approaches such as robust controller and adaptive switching gain. Reference [23] applies a model-independent control strategy called high-order differential feedback control to a quadrotor unmanned aerial vehicle. Reference [24] main concern is the control of a system facing an external disturbance corresponding to sensor failures. The solution employs a PID Sliding Mode Control (SMC), and the controlled system is a quadrotor dynamic model. Reference [25] deals with the problem of altitude control of the variable load quadrotor unmanned aerial vehicle, and a robust sliding mode controller (SMC) is proposed. Reference [26] discusses the development of a control architecture for hybrid UAVs, those which combine the beneficial features of fixed-wing UAVs and rotary wings UAVs. The control architecture is done based on model-free control principles. For fixed-wing aircraft, the control loops are typically well established. However, flight qualities definitions for UAV, which result in specification of parameters to be used in the control loops design for UAVs, does not exist [27] or, at least, are not established. One possibility is to use the parameters employed for human-piloted aircrafts [28]. In this work, the Longitudinal and Latero Directional control loops are designed using classical control theory, employing the aircraft mathematical model. The design of the control loops uses the definitions of Aircraft Classes, Flight Quality, and Flight Phase [29] for the specifications of the design parameters of the aircraft flight modes.

The development of an AP in a modularized way allows the development of each subsystem independently [30], [31]. In addition, a bench test platform that allows testing new proposals for each subsystem is quite appropriate for the rapid development of APs using new algorithms in their subsystems. In academic and business environments, a platform such as the proposed can be used for the development and implementation of new control, guidance and navigation algorithms. The new algorithm can be introduced in the complete AP, embedded in a microprocessor, and the virtual flight mission test can be executed. So, the performance of the new algorithm (as part of an AP) can be verified in a faster way.

In this work, the control block was implemented by the control loops designed using the specification parameters in [29], as stated before. The designed guidance algorithm follows the proposal made by [32], which interacts with the navigation and the control blocks. The navigation block resides in the platform's AP, where algorithms to filter data read from sensors or for sensor data fusion can be implemented. However, in this work, no algorithm is implemented in this block. The information received from the aircraft model is passed directly to the guidance and control blocks. Since in this work the algorithms are encapsulated in their respective blocks, changing a specific algorithm does not impose an alteration in other blocks in the system. In this way, developers of new AP functionalities can benefit from the development carried out in a modular way.

For the implementation of the PIL test platform, it is necessary to define the devices that compose the platform and to establish communication among them (simulated aircraft and embedded AP). The aircraft is simulated in MATLAB and the information of its current state is sent to the AP embedded in a microcontroller. The AP calculates the control surface commands to be imposed on the aircraft and returns these data to the aircraft.

With the platform implemented, initial PIL tests were carried out with the control loops embedded in the
The Control subsystem receives information of the current aircraft state, and the maneuver to be performed to computes which commands will be given to the vehicle's control actuators. The Control Surface Manager is responsible for sending deflection commands to the aircraft's control surfaces, ailerons, rudder, and elevator.

The Navigation system encompasses the sensors that read the status of the aircraft and the algorithms that process the signals, in order to provide the data to be used by the AP subsystems. In this work, this module only receives the information from the simulated aircraft and sends it to the microcontroller, without performing any processing.

The Ground Control Station keeps the status of the aircraft mission up to date for the ground team and can send new commands to the AP. The Control Surface Manager and Control Station subsystems will not be explored in this work.

Figure 1. Flowchart showing the approach.

III. THE AIRCRAFT MODEL

The aircraft modeling and identification obtain the mathematical model necessary to design an aircraft's control loops using classic or modern control theories. The use of small-scale aircraft to test the control design of large aircraft is a technique that is receiving attention nowadays [36]. Thus, before performing tests on the target aircraft, tests are performed on a similar, but smaller, aircraft [37],...
The aircraft used in the PIL platform is a nonlinear mathematical model of a Piper J-3 Cub ¼ scale [20]. The aircraft model and movement equations were implemented in MATLAB® Simulink®. The Earth was considered an inertial referential.

The rigid body dynamic equations of a fixed-wing aircraft, with six degrees of freedom (6-DOF), describe the translational and rotational dynamics. The adopted referential to these differential equations is the body system coordinates \((x_b, y_b, z_b)\), pointing to the aircraft nose; right wing, and down). The aircraft is considered to be symmetrical in the \(xz\) plane. This consideration simplifies the moment of inertia matrix, for which the elements corresponding to products of inertia are null.

To perform a nonlinear simulation of the aircraft, it is also necessary to include expressions to keep updated the aircraft position \((x_n, y_n, z_n)\) and attitude \((\phi, \theta, \psi)\), in the navigation coordinate system (the coordinate system chosen to track the aircraft mission), North–East–Down frame (NED). In this study, the wind effects will be neglected, and the Earth will be represented as a flat surface [39].

A simplified nonlinear model for a 6-DOF aircraft is shown in Figure 3. The atmospheric model used was the International Standard Atmosphere (ISA), but disturbances such as gusts of wind on the three axes of the aircraft are disregarded.

In the implemented model, the aerodynamic coefficients are input parameters to the aerodynamic subsystem. The dynamic system for simulation is given by:

\[
\begin{align*}
\dot{x} &= f(x, \dot{u}) \\
\dot{y} &= g(x, \dot{u})
\end{align*}
\]

where, \(f\) represents the rigid body dynamic and \(g\) the output equations. The state vector is:

\[
x = (u \ v \ w \ p \ q \ r \ \phi \ \theta \ \psi \ \text{n}_x \ \text{n}_y \ \text{n}_z)^T
\]

where, \(u, v,\) and \(w\) are the components of velocity in the NED frame, \(x_n, y_n,\) and \(z_n\) axis, respectively; \(p, q,\) and \(r\) are the angular velocity in the NED frame, \(x_n, y_n,\) and \(z_n,\) respectively; \(\phi, \theta,\) and \(\psi\) are the Euler angles representing the aircraft's attitude; \(x_n, y_n,\) and \(z_n\) are the aircraft's coordinate represented in the NED frame. The control vector is:

\[
\dot{u} = (\Pi \ \delta_e \ \delta_a \ \delta_r)^T
\]

where, \(\Pi\) is the variation in throttle; \(\delta_e\) the deflection in the elevator command surface; \(\delta_a\) the deflection in the ailerons command surface; \(\delta_r\) the deflection in the rudder command surface. The output vector:

\[
\dot{y} = (V_T \ \alpha \ \beta \ \text{n}_x \ \text{n}_y \ \text{n}_z)^T
\]

where, \(V_T\) is the air velocity, \(\alpha\) is the angle of attack and \(\beta\) is the sideslip angle. The other elements were defined before.

A dynamic system as presented in (1), with the state vector, command vector, and output vector, may be used to represent a typical fixed-wing aircraft. The parameters used in the dynamic system must be related to the specific aircraft to be simulated. In this way, to implement a mathematical model of an aircraft, it is necessary to know...
its geometric parameters and aerodynamic coefficients.

A. PIPER J-3 CUB ¼ MODEL

The main geometric parameters of the Piper J-3 Cub ¼ are presented in Table I. The inertia parameters, mass, and moments of inertia matrix use the data given by [41]. The inertia matrix \( J \) was computed by the method of approximate numerical integration. In general, in the case of conventional fixed-wing aircrafts, it is assumed that the aircraft is symmetrical related to the \( xz \) plane. Where \( x \) is the longitudinal axis pointing to the aircraft's nose, \( z \) is the axis pointing to the aircraft's belly (both with origin in the center of gravity of the aircraft).

To complete the nonlinear model implementation that describes the aircraft dynamic movement, the aerodynamic derivatives, and control coefficients of the aircraft are necessary. In [20], the parameters of a Piper J-3 Cub ¼ were determined for the aircraft flying in different velocities (\( V_T \)) by software Athena Vortex Lattice [AVL] and USAF Stability and Control Digital DATCOM. In this work, it was used the aerodynamic derivatives and control coefficients related to \( V_T = 20 \text{ m/s} \).

Table II lists the parameters used for \( V_T = 20 \text{ m/s} \) according to [20]. The longitudinal aerodynamic derivatives and control coefficients are: \( CL0, CD0, CL\alpha, CM\alpha, CLq, CMq, CL\delta e, \) and \( CM\delta e \). Table III lists the Latero Directional aerodynamic derivatives and control coefficients are: \( CY\beta, CL\beta, CN\beta, CYp, CLp, CNp, CYr, CLr, CNr, CL\delta a, CN\delta a, CY\delta r, CL\delta r\) and \( CN\delta r \). Stability derivatives and control derivatives measure how forces and moments in an aircraft change as other parameters, related to stability, modify. Parameters such as airspeed, altitude, and angle of attack as examples. They are used in the nonlinear aircraft model implementation [39].

B. LINEARIZED LONGITUDINAL AND LATERO DIRECTIONAL MODELS

To design the controllers using classic control theory, it is necessary to linearize the nonlinear model. This can be done with the aircraft in an equilibrium condition. The trim condition considers the aircraft control surfaces set for a straight and leveled flight. In this condition, all translational and rotational accelerations are null, and this equilibrium condition is given by:

\[
0 = \vec{x} = f(\vec{x}_0, \vec{u}_0) \quad (5)
\]

This condition provides an initial state to the flight simulation, and the nonlinear equations can be linearized. In this work, the trim point in the flight envelop is \( V_{T0} = 20 \text{ m/s} \) and \( H_0 = 400 \text{ m} \), the aircraft altitude. The equilibrium states and input were computed by MATLAB\textsuperscript{®} and are presented in Table IV.

Using the solution for the equilibrium state, it is possible to linearize the nonlinear mathematic model. The resulting linearized dynamic system is represented as in (6), and this representation can be used for both, longitudinal and Latero Directional modes.

\[
\begin{align*}
\Delta \ddot{x} &= A\Delta \dot{x} + B\Delta u \\
\dot{y} &= C\Delta \dot{x} + D\Delta u
\end{align*}
\]

In general, the \( \Delta \) notation is neglected, but the new \( \vec{x} \) and \( \vec{u} \) variables are understood as deviations from the equilibrium.

The linearized dynamic system to the Longitudinal mode has the state vector \( x \), the control vector \( u \), and output vector \( y \):

\[
\begin{align*}
\ddot{x} &= (u \ w \ q \ \theta \ z_n)^T \\
\dot{u} &= (\Pi \ \delta_e)^T \\
\dot{y} &= (V_T \ \alpha \ q \ \theta \ \Pi)^T
\end{align*}
\]

The matrices \( A, B, C \) and \( D \) are computed by MATLAB\textsuperscript{®} using a developed script:

\[
A = \begin{bmatrix}
-0.0963 & 0.5176 & -1.2715 & -9.7938 & 0 \\
-0.6680 & -6.1282 & 15.977 & -0.5009 & -0.0009 \\
0.4485 & -8.7745 & -26.529 & 0 & 0 \\
-0.0511 & 0.9987 & 0 & -20.000 & 0 \\
19.2141 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
127.712 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1.0000 & 0 \\
0 & 0 & 0 & 0 & -1.0000
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
19.2141 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
127.712 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1.0000 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1.0000
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1.0000 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1.0000
\end{bmatrix}
\]

\[
D = (0)_{5 \times 2}
\]

The linearized dynamic system to the Latero Directional has the state space vectors:

\[
\begin{align*}
\dot{x} &= (v \ p \ q \ \phi \ \psi)^T \\
\dot{u} &= (\delta_a \ \delta_e)^T \\
\dot{y} &= (\beta \ p \ q \ \phi \ \psi)^T
\end{align*}
\]

Proceeding in a similar way to the longitudinal case, the state space matrices can be found:

\[
A = \begin{bmatrix}
-0.2120 & 1.2162 & -19.551 & 9.7938 & 0 \\
-1.9852 & -49.630 & 11.560 & 0 & 0 \\
1.2963 & -1.7111 & -3.9050 & 0 & 0 \\
0 & 1.0000 & 0.0511 & 0 & 0 \\
0 & 0 & 1.1001 & 0 & 0
\end{bmatrix}
\]
C. STABILITY AND FLYING QUALITIES

It is necessary to evaluate the aircraft's stability and control, and flying qualities before applying classic control theory to design control loops. Stability and control comprehend how an aircraft flies and can be controlled. Flight qualities are the characteristics of stability and control behavior of the aircraft [39].

Once the system is linearized, it is possible to analyze the stability parameters through the poles of the characteristic equation of the system. In addition, control loops with desired behaviors can be obtained by allocating the poles. In order to carry out these analyzes, each linearized dynamic system (Longitudinal and Latero Directional) must be studied according to its flight modes [42].

For the Longitudinal Dynamic, it was considered the: Phugoid Mode (pair of complex poles conjugated closest to the origin); and Short Period Mode (pair of conjugated complex poles furthest from the origin). For the Latero Directional Dynamic, it was considered the: Dutch Rolling Mode (pair of conjugated complex poles); Pure Rolling Mode (real pole in the left semi-plane, very stable); and Spiral Mode (real pole close to the origin, often slightly unstable).

IV. CONTROL LOOP DESIGN

To perform the control loop design of a system, it is necessary to define the requirements. This work uses the flying qualities according to the MIL-8785C specification [29]. This document classifies the aircrafts according to the geometric and inertial parameters. The specifications for the control design consider also the flight phase for which the design is considered. By the MIL-8785C, the Piper J-3 Cub \( \frac{1}{4} \), can be classified as a small, light airplane, class I. Regarding the flight phase, category C is more adequate, since the control flight system proposed has the objective to allows an autonomous flight following a pre-defined trajectory. Finally, once the proposed flight control design should provide a small margin of error, Level 1 flight quality has been adopted. For an aircraft with these classifications, the flight quality specifications are as in Table V.

Using the linearized model, the control loops for the Longitudinal and Latero Directional movements were designed, allowing the system to operate in points or conditions near the equilibrium. The actuators’ dynamics were considered ideal, so they were not modeled.

The controllers’ design was not the main scope of this work, so, the details will not be presented. The loops and controllers design follow the approach presented in [43]. The simulations in the time domain were performed using MATLAB® Simulink® with the integration routine \( \text{ode4} \) (Runge-Kutta) and sample time \( T_s \) equal to 10 ms.

A. LONGITUDINAL CONTROL SYSTEM

The longitudinal control design for the Piper J-3 1/4, was designed following [43] and its block diagram is presented in Figure 4. The aircraft is the linearized Longitudinal Dynamic system determined in Section 3. The pitch stability augmentation system (Loop1) is used when the aircraft is in level flight. The velocity control loop (Loop 2) tracks the reference velocity during the flight. It is the combination of the controlled pitch angle mode and the velocity feedback (\( V_T \)), and a dynamic compensator for the
throttle \( G_{c, VT} \). The altitude control system (Loop 3) allows the aircraft to be maintained at a fixed altitude. In the proposed longitudinal controller, the velocity and altitude controllers act in parallel with each other, preventing the control system to perform sudden movements and, after reaching the desired reference values, the aircraft returns to the equilibrium state.

The determination of the compensators' gains considered first Loop 1, and then Loop 2 and Loop 3. The gains of compensators PI (proportional and integral), \( G_{c, \theta}, G_{c, VT} \) and \( G_{c, H} \) were obtained with Matlab's PID Tuner. The \( k_q \) parameter was designed using the root locus method applied to the transfer function \( \theta(s)/\delta_e(s) \), which can be obtained from the aircraft longitudinal linearized dynamic. To the \( k_{VT} \) and \( k_{H} \) parameters, it was heuristically assigned a value equal to 1 (one), which means that ideal sensors are being considered for the measurement of altitude and velocity. Table VI shows the designed gains.

### TABLE I.
**GEOMETRIC PARAMETERS** [41]

| Parameter | Description               | Value  |
|-----------|---------------------------|--------|
| \( S \)   | Flat wing area            | 0.8169 m² |
| \( b \)   | wingspan                  | 2.34 m  |
| \( \bar{c} \) | Mean aerodynamic chord    | 0.3491 m |
| \( AR \)  | Aspect ratio              | 6.7030 |
| \( x_{CG} \) | Position x of the center of gravity (CG) | 0.46 m |
| \( y_{CG} \) | Position y of the center of gravity (CG) | 0.00 m |
| \( z_{CG} \) | Position z of the center of gravity (CG) | -0.18 m |

### TABLE II.
**LONGITUDINAL AERODYNAMIC DERIVATIVES AND CONTROL COEFFICIENTS FOR \( V_T=20 \text{ m/s} \)** [20]

| \( CL_0 \) | \( CD_0 \) | \( CL_{\alpha} \) | \( CM_{\alpha} \) | \( CL_q \) | \( CM_q \) | \( CL_{\delta e} \) | \( CM_{\delta e} \) |
|------------|------------|------------------|------------------|------------|------------|------------------|------------------|
| 0.14143    | 0.02269    | 4.7998           | -1.6559          | 7.8822     | -12.6926   | 0.3275           | -1.2032          |

### TABLE III.
**LATERAL DIRECTIONAL AERODYNAMIC DERIVATIVES AND CONTROL COEFFICIENTS FOR \( V_T=20 \text{ m/s} \)** [20].

| \( CY_{\beta} \) | \( CL_{\beta} \) | \( CN_{\beta} \) | \( CY_{p} \) | \( CL_{p} \) | \( CN_{p} \) | \( CY_{\delta r} \) | \( CL_{\delta r} \) | \( CN_{\delta r} \) |
|------------------|------------------|------------------|------------|------------|------------|------------------|------------------|------------------|
| -0.1362          | -0.0494          | 0.0631            | 0.0657     | -0.5200    | -0.0237    | 0.1429           |                   | |

### TABLE IV.
**EQUILIBRIUM STATES.**

| \( u_0 \) (m/s) | \( w_0 \) (m/s) | \( \theta_0 \) (rad) | \( H_0 \) (m) | \( H_0 \) (%) | \( \delta_{\alpha} \) (rad) | \( \delta_{\alpha} \) (rad) | \( \delta_{\alpha} \) (rad) |
|------------------|-----------------|----------------------|--------------|--------------|-----------------------------|-----------------------------|-----------------------------|
| 19.9739          | 1.0219          | 0.0511               | 400          | 4.16         | -0.0703                     | 0                           | 0                           |

### TABLE V.
**SPECIFICATIONS FOR THE FLIGHT CONTROL DESIGN. LATERAL DIRECTIONAL MODES SPECIFICATIONS:**

- \( \zeta_{min} \) (minimum damping ratio);
- \( \zeta_{0a} \) (minimum); \( \omega_n \) (minimum natural frequency); \( \tau \) (maximum time constant);
- \( \zeta \) (damping ratio).

**LONGITUDINAL MODES SPECIFICATIONS:**

| \( \zeta \) | \( \zeta_{min} \) | \( \zeta_{0a} \) | \( \omega_n \) (Rad/s) | \( \delta_{\alpha} \) (rad) | \( \delta_{\alpha} \) (rad) | \( \delta_{\alpha} \) (rad) |
|-------------|-------------------|------------------|------------------------|-----------------------------|-----------------------------|-----------------------------|
| \( \zeta \) | 0.08              | 0.15             | 1.0                    | -0.0703                     | 0                           | 0                           |

| \( \tau \) (s) | \( \zeta \) | \( \zeta_{min} \) | \( \zeta_{0a} \) | \( \omega_n \) (Rad/s) | \( \delta_{\alpha} \) (rad) | \( \delta_{\alpha} \) (rad) | \( \delta_{\alpha} \) (rad) |
|---------------|-------------|------------------|------------------|------------------------|-----------------------------|-----------------------------|-----------------------------|
| 1.0           | \( \zeta \) | 0.35 < \( \zeta \) < 1.3 |                      |                         |                             |                             |                             |

| \( \zeta \) | \( \zeta_{min} \) | \( \zeta_{0a} \) | \( \omega_n \) (Rad/s) | \( \delta_{\alpha} \) (rad) | \( \delta_{\alpha} \) (rad) | \( \delta_{\alpha} \) (rad) |
|-------------|-------------------|------------------|------------------------|-----------------------------|-----------------------------|-----------------------------|
| \( \zeta \) | 0.08              | 0.15             | 1.0                    | -0.0703                     | 0                           | 0                           |

| \( \tau \) (s) | \( \zeta \) | \( \zeta_{min} \) | \( \zeta_{0a} \) | \( \omega_n \) (Rad/s) | \( \delta_{\alpha} \) (rad) | \( \delta_{\alpha} \) (rad) | \( \delta_{\alpha} \) (rad) |
|---------------|-------------|------------------|------------------|------------------------|-----------------------------|-----------------------------|-----------------------------|
| 1.0           | \( \zeta \) | 0.35 < \( \zeta \) < 1.3 |                      |                         |                             |                             |                             |
Figure 5 shows the temporal longitudinal response in closed-loop, of the designed controllers. The reference used was a 25 m doublet in $H_r$. The control system switches between the Altitude control and the velocity and pitch angle controls, depending on the flight situation. The switching points are indicated by $A$ switching when the Altitude control assumes the system command, and $B$ switching when velocity and pitch angle control. In case of altitude changing, the Altitude control should be triggered when the aircraft is at a distance up to 5 m from the reference altitude ($A$ switching). Otherwise, the velocity and pitch angle autopilots are triggered until the actual aircraft altitude is 5 m from the reference altitude ($B$ switching).

The simulations in time domain were performed using MATLAB® Simulink® with the integration routine ode4 (Runge-Kutta) and sample time $T_s$ equal to 10 ms.

Figure 5. Temporal longitudinal response in closed loop, of the designed

B. Latero Directional controller

In general, for a UAV, the Latero Directional controller is designed to keep the aircraft pointing in the desired direction (compass heading). To control the yaw angle, a strategy commonly used is to keep the roll angle in a pre-determined position until the aircraft reaches the desired reference. Therefore, the conventional method to implement this autopilot is to feedback the yaw angle ($\psi$) and use the roll angle ($\phi$) autopilot as an internal loop in the yaw angle autopilot.

The Latero directional control design implemented for the Piper J-3 1/4 has the diagram shown in Figure 6 and incorporates the highlighted systems. The Latero directional stability system augmentation (Loop 1) is a combination of roll and yaw stability systems, as these movements are generally coupled. The roll angle control system (Loop 2) acts, in its simplest form, as a wing leveler. Finally, the yaw angle controller uses the roll angle control system as an internal loop to point the aircraft in a certain reference direction.

The aircraft is the linearized Latero directional Dynamic system determined in Section 3. The loops designs are made in sequence, Loop-1, Loop-2 and finally Loop-3. Loop 1 is the Latero directional stability system augmentation, which contemplates stability augmentation of the yaw angle and the roll angle. For yaw stability augmentation using yaw rate feedback, $r$, the control is performed so that the yaw angle error should be zero, damping the Dutch Roll movement. However, in coordinated flight (turn maneuver), the yaw rate must be constant and non-zero. Thus, to meet these two commitments, a usual solution is to use a washout filter, $G_w$, with the form:

$$ G_w = s \tau_w / (1 + s \tau_w) \quad (21) $$

The Latero Directional controller was designed following the proposal of [43]. The gains of the PI compensator, $G_{c,\phi}$, were obtained with the Matlab PID Tuner. The parameters $k_p$ and $k_i$ were designed using the Root Locus method, and $k_o$ was chosen equal unit, which means that ideal sensor is being considered for the measurement of roll angle. To calculate $\phi_{ref}$ from the yaw angle, $\psi$, it was used the expression $u_0 / (\tau g)$, in which $u_0$ is the nominal velocity of the aircraft (20m/s), $g$ is the gravity acceleration, and $\tau$ equal 1 (a value

| Pitch stability augmentation $G_{c,\theta}$ | Velocity and pitch angle $G_{c,VT}$ | Altitude $G_{c,H}$ |
|---------------------------------------------|-----------------------------------|-------------------|
| $K_\theta$ | $K_{\psi,\theta}$ | $K_{\phi,\theta}$  |
| $K_{VT}$ | $K_{VT,\phi}$ | $K_{VT,\psi}$  |
| $K_H$ | $K_{H,\phi}$ | $K_{H,\psi}$  |
| 1 | 0.0877 | -0.45319 | -0.66623 | 1 | 0.030269 | 0.022003 | 1 | 0.0172410 | 0.0059742 |
heuristically assigned). In Table VII the designed gains and some parameters used are presented. With the Latero Directional control designed, the system behavior was studied. For the designed system, in case of a change in the yaw reference angle, the aircraft should maneuver to reach the new yaw angle. During the maneuver, the roll angle changes, and when the desired yaw is reached, it returns to the null position, with the aircraft with leveled wings.

Figure 7 shows the temporal Latero Directional response in closed-loop for the complete Latero Directional control system. The reference used was a $45^\circ$ amplitude doublet in $\psi_r$, at $t=50$ s. The graphics show that the linearized and the nonlinear aircraft models (both using the control system designed using the linearized model dynamics), have a similar response. The yaw angle follows the input reference, $\psi_r$, with an accommodation time of 32.50 s.

V. GUIDANCE ALGORITHM
In the proposed AP, the aircraft mission is specified by a waypoint matrix, composed of the geographic coordinates (NED frame) of the points to be reached during the mission and the aerodynamic velocity planned for each waypoint.

The guidance system generates the reference heading angle by accessing the waypoint matrix during the mission, and repeat the processes for each waypoint (WP) in the waypoint matrix. The altitude and velocity are provided in the waypoint matrix, and the guidance system does not need to calculate them. The guidance algorithm work with the information relative to the next WP to be reached and the current state of the aircraft and generates the references to be used by the control system. As already commented, the implemented algorithm is an adaptation of the one proposed by [32]. It deals with the heading generator, the monitoring and control of the Cross-track error, and the switching waypoint.

A. HEADING GENERATOR
The reference heading angle, the heading for which the aircraft should point, is computed based on the current UAV position ($N_{UAV}$, $E_{UAV}$), and its desired position ($N_{WP(n+1)}$, $E_{WP(n+1)}$), defined by the next waypoint, $WP_{(n+1)}$, both positions given in the NED frame. Consider Figure 8, and a triangle formed by the line parallel to the East axis ($N_{WP(n+1)}$ coordinate), the line parallel to the North axis ($E_{UAV}$ coordinate), and the line connecting the UAV to the WP. In the triangle, $\psi_{ref}$ is the angle at the vertex in which the aircraft is located. So, $\psi_{ref}$ is computed as:

$$\psi_{ref} = \tan^{-1}\left(\frac{R_{WP(n+1)} - R_{UAV}}{N_{WP(n+1)} - N_{UAV}}\right) \quad (22)$$

The reference yaw angle, $\psi_{ref}$, is sent as a reference to the AP control system.
Figure 8. Coordinates representation in the heading angle computation.

B. MONITORING OF THE CROSS-TRACK ERROR

During the straight and level flight, when the aircraft is flying between two WPs (but not maneuvering for a WP switching), to avoid continuous correction in the trajectory and the flight parameters, it is defined a region around the segment connecting the two waypoints in which region no correction is executed. In this way, yaw angle corrections are imposed only when the UAV cross-path error is greater than an assigned value, i.e. the maximum acceptable transversal error [33].

The region where no correction is done is defined as an isosceles trapezoid, whose axis of symmetry is defined by the segment that connects a waypoint to the next one (WP\(n\) to WP\((n+1)\)), Figure 9. The distance between the two WPs is \(d_s\), \(P_{\text{ref}}\) point is the projection of the aircraft's position on the line connecting the two consecutive waypoints; \(d_{\text{ref}}\) is the distance from \(P_{\text{ref}}\) to the next waypoint; and \(\varepsilon_r\) is the cross-track error and \(K\) and \(\alpha\) are intermediate variables. The maximum acceptable cross-track error, \(\varepsilon_{\text{max}}\), is variable and decreases over segments \(a\) and \(b\) (longer and shorter bases). Consider Figure 8. The inclination of a leg of the isosceles trapezoid can be computed as:

\[
\tan \alpha = \frac{(a/2) - (b/2)}{d_s}
\]  

(23)

By triangle likeness:

\[
\frac{(a/2) - (b/2)}{d_s} = \frac{K}{d_{\text{ref}}} \Rightarrow K = d_{\text{ref}} \left(\frac{(a/2) - (b/2)}{d_s}\right)
\]  

(24)

And then, the maximum acceptable cross-track error, \(\varepsilon_{\text{max}}\), is given by:

\[
\varepsilon_{\text{max}} = \frac{b}{2} + K = \frac{b}{2} + d_{\text{ref}} \left(\frac{(a/2) - (b/2)}{d_s}\right)
\]  

(25)

If the UAV cross-track error is less than \(\varepsilon_{\text{max}}\) during the direct flight phase, no correction to the aircraft heading should be made. Otherwise, if \(\varepsilon_r \geq \varepsilon_{\text{max}}\), the new reference direction angle will be the angle between the current UAV position and the next WP, computed by the heading generator.

When the distance between the aircraft and the next waypoint is smaller than the minimum distance defined, it is considered that the WP was reached. Then, the aircraft now should be led to the next WP, and the guidance system switches to the next point in the waypoint matrix.

VI. EMBEDDING THE AP IN A MICROCONTROLLER

The validation of the guidance algorithm and the control laws which run in the microcontroller was performed in a PIL simulation environment. The test platform was implemented using the LM3S6965 microcontroller, an ARM® based controller [45].

To implement the AP in the hardware, it was necessary to define a communication protocol for data exchange among modules. It consists of a header and a terminator that indicate the beginning and end of transmission; a register that defines what information will be transmitted and a data set with the information to be transmitted.

In order to implement the control system developed in continuous time domain, it is necessary to discretize the control. The difference equations were determined by using the discrete equivalent technique, with a bilinear transformation (Tustin Method). The sampling period to solve the non-linear equations that describe the dynamics of the aircraft's movement was \(Ts = 0.01\ s\), the same used in SIMULINK.

A. PIL TEST PLATFORM

The defined data packet structure establishes communication between Simulink and the microcontroller platform. A local area network between the embedded AP and the simulated aircraft was created using Ethernet ports, and a specific IP address for each device was assigned.
During each step of simulation of the aircraft model, the input and output data interface is responsible for receiving and decoding the data packet sent by the microcontroller. In the codification and decodification of the data packets, it was used the S-Function Builder, a Simulink® block able to integrate codes in C/C++ Programming Language.

For the Reception and Decryption of the Data Pack coming from the microcontroller, the interface uses the UDP Receive block available in the Instrument Control Toolbox™. Then, it decodes (unpack) and makes the received variables available for subsequent processing of the nonlinear simulation model.

The data entry interface implemented in the test platform is responsible for receiving the data packets sent by the microcontroller, decoding, and making the received variables available for subsequent processing of the nonlinear simulation model. The Simulink® blocks used in the implementation of this interface are shown in Figure 10.

The UDP Receive block configures and opens an interface for a specific IP address using the UDP protocol. The Unpack block is responsible for dividing the received data packets into a one-dimensional vector. The Byte Reversal block performs the necessary inversion in the order of bytes, since the microcontroller uses the big-endian data format and the MATLAB® uses the little-endian. The S-Function Builder block is responsible for decoding the data packet received from the hardware, which was assembled using the defined data packet structure for the communication between Simulink and the microcontroller platform.

In summary, at each simulation step, the routine for data input is called and then analyze the structure of the received data packet, group the bytes, extract the values of the variables being transmitted, and, finally, make these variables available to the input vector of the aircraft simulation model.

For the encoding and sending the data package to the microcontroller the data output interface implemented in the test platform, is responsible for coding and sending the data packet to the microcontroller. The data packet sent, in each simulation step, must contain the output vector variables of the aircraft nonlinear simulation model. The Simulink® blocks used in the implementation of this interface are shown in Figure 11.

The Data Type Conversion block is responsible for converting the variables to be sent, from the double-precision to the single-precision floating-point format. The S-Function Builder block is responsible for converting the output vector variables from the non-linear aircraft simulation model. The Pack block is used to convert the received one-dimensional vector into a data packet with 77 bytes in sequence. Finally, the UDP Send block is responsible for sending the data packet to the AP embedded in the microcontroller, to the specified IP address and local port, using the UDP protocol.

B. PIL TESTS OF THE LONGITUDINAL AND LATERO DIRECTIONAL CONTROL LOOPS

Figure 12 shows the closed-loop responses of the Latero Directional and Longitudinal Dynamics controllers running in the microcontroller. The figure also shows the responses of the nonlinear model controlled by the control loops implemented in continuous time, simulated in MATLAB.

The responses of the aircraft controlled by the control loops embedded in the microcontroller are very close to those performed by the continuous time control loops, indicating that the discrete control was performed satisfactorily. The final test of the designed control block is done when embedding the AP (control and guidance) blocks to perform a mission.

VII. PIL MISSION TESTS

The missions defined to tests are those in which the UAV must pass through predetermined geographical coordinates, or waypoints. These missions were simulated to assess the performance of the guidance and control systems developed in this work.

The results obtained with the PIL simulations performed using the developed test platform will be presented. Three missions were executed with the UAV traversing trajectories of predefined waypoints.

After reaching the last desired point, the mission is considered completed and the simulation is ended. The mapping of the desired waypoints, for each trajectory, in the local navigation coordinate system (NED frame) is shown in Table VIII. The definition of the waypoints was done in the MATLAB® environment.
Figure 12. Closed loop responses. Left the Latero directional. Right the Longitudinal.

| Latero directional | Roll angle | Yaw angle |
|--------------------|------------|-----------|
|                     | $G_{r}$    | $K_r$     | $K_{\phi}$ | $k_{f,r}$ | $k_{f,\phi}$ | $k_{i,r}$ | $k_{i,\phi}$ | $k_{i,\phi}$ | gravity acceleration |
|                     | 0.105      | 1         | 0.119      | -0.090681 | -0.019958 | 9.8       |

TABLE VIII. Waypoints coordinates in NED coordinates system

| Trajectory A | Trajectory B | Trajectory C |
|--------------|--------------|--------------|
| WP1          | (0, 0, 400)  | (0, 0, 400)  |
| WP2          | (1000, 0, 400) | (1000, 0, 400) | (309, 951, 400) |
| WP3          | (309, 951, 400) | (-809, 588, 420) | (-809, 588, 425) |
| WP4          | (-809, 588, 400) | (309, 951, 380) | (309, 951, 375) |
| WP5          | (-809, -588, 400) | (309, 951, 420) | (-809, -588, 350) |
| WP6          | (309, -951, 400) | (-809, -588, 380) | (309, -951, 400) |
| WP7          | (1000, 0, 400) | (1000, 0, 400) | - |
The autopilot interacts with the aircraft simulation dynamics implemented in MATLAB® Simulink® environment, which employs the ode4 simulation algorithm (Runge-Kutta method) with a sampling period equal to 0.01 s. The simulations consist of performing flight trajectories considering the following conditions:

- The atmospheric model does not take into account disturbances such as gusts of wind on the axes of the aircraft;
- The mathematical modeling and the inertial and geometric parameters do not present errors;
- The sensors do not have typical errors such as bias, scale factor, and nonlinearity.

Figure 13 shows the platform in execution. During the execution of the PIL platform, twenty-four libraries were used in the code embedded in the microcontroller, for instance "lib/gpio.h", "lib/ethernet.h", "app/eth/eth_udp.h" for communication. For the missions presented in the article, the execution time in Matlab was, on average, 559 s for the microcontroller running at 50MHz.

The codes necessary to reproduce the proposed platform as presented in this paper, with a short video showing the platform being executed, are available as a supplementary material posted with this article.

The codes to reproduce the proposed platform in microcontrollers from the Arduino family, specifically the microcontrollers ATmega328P (Arduino UNO) from Microchip, Stm32f103c8 from STM Microeletronics, Esp8266 and Esp32 from Expressif Systems, are also available as a supplementary material posted with this article.

A. TRAJECTORY A

Figure 14 presents the result after the simulated mission, where it is possible to verify that the aircraft has traveled all the pre-defined waypoints for Trajectory A. The simulation starts at Waypoint 1 (WP1) and ends at Waypoint (WP7). The aircraft traveled a route of approximately 7 km in total length and the duration of the mission was 5 min and 58 seconds (358 s). The figure also shows the responses for the aerodynamic velocity, altitude, and yaw angle of the aircraft. These are variables affected by the reference signals generated by the guidance system during the simulation and tracked by the designed control system. Notice that Trajectory A is a bidimensional trajectory, given that there is no variation in altitude.

B. TRAJECTORY B

Figure 15 shows the result after the simulated mission showing that the aircraft has traversed all the pre-defined waypoints for Path B. The simulation starts at Waypoint 1 (WP1) and ends at Waypoint (WP7). The aircraft traveled a route of approximately 11 km in total length and the duration of the mission was 9 minutes and 22 seconds (562 s). The figure also shows the aircraft's airspeed, altitude, and yaw angle responses. As already commented, these are variables affected by the reference signals generated by the guidance system during the simulation and tracked by the designed control system.

In Trajectory B mission, there is altitude variation. In the aerodynamic velocity response, it is observed that, even though the reference velocity is constant, the velocity of the aircraft is not. This is because there are coupled movements in the aircraft dynamic. Changing altitude, the aircraft will change velocity. The velocity control was designed to keep the velocity as constant as possible. The results show that the error between the reference velocity and the aircraft velocity was less than 14% in two points of the trajectory. During most part of the flight, the velocity error is less than 5%. The altitude and yaw angle responses indicate that the aircraft performed the predetermined mission autonomously.

C. TRAJECTORY C

Figure 16 shows the result after the simulated mission, where it is possible to verify that the aircraft has traversed all the pre-defined waypoints for Trajectory C. The simulation starts at Waypoint 1 (WP1) and ends at Waypoint (WP7). The aircraft traveled a route of approximately 8 km in total length and the duration of the mission was 6 minutes and 23 seconds (383 s). The figure also shows the aerodynamic velocity, altitude, and yaw angle of the aircraft, corresponding to the reference signals generated by the guidance system during the simulation and tracked by the designed control system.
Figure 14. Reached waypoints in mission Trajectory A (left up); Aircraft aerodynamic velocity response (right up); Aircraft altitude response (left down); Yaw angle of the aircraft during mission Trajectory A (right down).

Figure 15. Reached waypoints in mission Trajectory B (left up); Aircraft aerodynamic velocity response (right up); Aircraft altitude response (left down); Yaw angle of the aircraft during mission Trajectory A (right down).
Figure 16. Reached waypoints in mission Trajectory C (left up); Aircraft aerodynamic velocity response (right up); Aircraft altitude response (left down); Yaw angle of the aircraft during mission (right down).

In the Trajectory C mission, there is also altitude variation Similar to Trajectory B, the maximum aerodynamic velocity error was 16%, remaining less than 5% during most of the flight. In trajectory C, the yaw angle suffers a decrease in its value, a maneuver that was not done in the previous missions. As before, in Trajectory C the aircraft fulfilled the mission.

VIII. CONCLUSION
The stages of development for control and guidance blocks of an AP for a fixed-wing aircraft were presented. The implementation of a PIL test platform is carried out, by implementing the developed AP in a microcontroller, executing the aircraft model on a PC and establishing the communication between these two elements. Thus, the AP is responsible for piloting the simulated aircraft, leading it to perform a mission established by WPs. Tests on the PIL platform were presented, in which the aircraft performs the missions, reaching all the WP previously defined. The tests presented indicate the success of the developed AP project, as well as the functionality of the implemented PIL platform.

In this work, we consider an AP consisting of three blocks: navigation, guidance, and control. These blocks were developed in a modular way allowing individual modifications.

The design of control loops using classical control theory needs the aircraft's mathematical model, and a Piper ¼ scale mathematical model was employed. As design parameters, the MILF-8785C standard was used (since flight quality standards for autonomous aircraft are not yet fully established). For the design of the control loops, the linearized model was computed for a fixed speed. However, if it is desired to work with the aircraft in different flight conditions, linearization at different flight conditions can be performed, control loops for each condition can be designed and then bank of controllers can be used during the flight mission, in which each controller is used in the respective flight condition for which it was designed. The bank of controller may be implemented in the platform and the PIL test may be performed. The guidance block was performed using an algorithm available in the literature. No algorithms have been implemented in the navigation block, however, the block structure is present, available for use.

The PIL platform implemented can be used for designing an AP for another aircraft model. For this, it will be necessary just substitute the present model by mathematical model of the desired aircraft since the communication between microcontroller and PC is carried out. If the new aircraft is fixed wing, the control loop’s structure can be used, being necessary just to design the controllers based in the new mathematical model. As the development of the PA was done in a modularized way, the implementation of new functionalities in one block is possible without changing other blocks. As the proposed platform is of low
cost and can be used for the development of several algorithms, even though it is not possible to accurately estimate the cost effectiveness, it is understood that it is adequate given the time savings and accident prevention, as a bench test stage.

There is a work in progress related to the navigation block. Models of sensor errors are being introduced and algorithms for filtering the signals provided by them will be implemented in the navigation block. As a possible future work, the problem of detecting and avoiding obstacle can be addressed. For this, it must be developed the introduction of obstacle information, and sensors to detect these obstacles must be implemented in the navigation block. In the guidance block, an algorithm to use the information of the obstacle detection should be developed, so that the resulting AP will be capable of altering its mission during the flight. The evolution of the platform to a real-time system is being planned. The first step to be taken in this direction will be the use of Matlab’s real time pacer function, to synchronize its time with the real time elapsed. For the real-time system, a more detailed study of aspects such as runtime and computational complexity will be made.

Since the PIL tests are always one of the last tests to be carried out before flight tests, it is considered that the developed PIL test platform is very useful for groups working with the development of PAs for autonomous aircraft.

REFERENCES

[1] A. S. Saeed, et al., “A survey of hybrid Unmanned Aerial Vehicles,” Progress in Aerospace Sciences, Vol. 98, pp. 91–105, 2018. https://doi.org/10.1016/j.paerosci.2018.03.007
[2] A. Sobester, et al., “Conceptual Design of UAV Airframes Using a Generic Geometry Service,” Infotech/Aerospace AIAA Meeting Papers, Arlington, Virginia, Sept. 26 – 29, 2005. https://doi.org/10.2514/6.2005-7079
[3] D. O. Costa, N. M. F. Oliveira and R. d’Amore “The feasibility of Remotely-Piloted Aircrafts for VOR flight inspection. Sensors 2020, 20(7) DOI: 10.3390/s20071947
[4] X.Zuo, G. Han and X. Mu, “Joint Optimization of UAV Trajectory Statistical Precoding and User Scheduling,” IEEE Access, Vol. 8, pp. 73232 – 73240, 14 April 2020, DOI: 10.1109/ACCESS.2020.2987967
[5] P. Radoglou-Grammatikis, et al., “A compilation of UAV applications for precision agriculture,” Computer Networks, Vol. 172, 107148, 2020, DOI: 10.1016/j.comnet.2020.107148
[6] H. Menouar, et al., “UAV-Enabled Intelligent Transportation Systems for the Smart City: Applications and Challenges,” IEEE Communications Magazine, Vol. 55, Issue: 3, March 2017, DOI: 10.1109/MCOM.2017.1600238CM
[7] Z. Zhang, et al., “Hardware-in-the-loop Simulation Platform for Unmanned Aerial Vehicle Swarm System: Architecture and Application,” in Proc. of the 39th Chinese Control Conference, Shenyang, China, July 27-29, 2020. DOI: 10.23919/CCCS2018.2019.0185541
[8] T. Sherman and S. Boskovich, “Application of a Modified Boids Algorithm For Navigation and Mapping with a System of Small UAVs,” in AIAA SciTech Forum, San Diego, California, USA, Jan. 7-11, 2019. https://doi.org/10.2514/6.2019-2287
[9] J. Yang, et al., “A software-in-the-loop implementation of adaptive formation control for fixed-wing UAVs,” IEEE/CAA Journal of Automatica Sinica, Vol. 6, Issue: 5, Sept. 2019. DOI: 10.1109/JAS.2019.1911702
[10] Z. Xu, et al., “Coordinated Standoff Target Tracking Guidance Method for UAVs,” IEEE Access, Vol. 6, pp. 59853 – 59859, Oct. 2018. DOI: 10.1109/ACCESS.2018.2875787
[11] A. R. Kuroswiski, N. M. F. Oliveira, E. H. Shiguemori,”Autonomous long-range navigation in GNSS-denied environment with low cost UAV platform”, in 2019 IEEE Latin America Transactions (ISyCon) Vancouver, BC, Canada, DOI: 10.1109/ISYCON.2018.8369592
[12] K. D. Nguyen, C. Ha “Development of Hardware-in-the-Loop Simulation Based on Gazebo and Fixhawk for Unmanned Aerial Vehicles,” International Journal of Aeronautical and Space Sciences vol. 19, 238–249 (2018). https://doi.org/10.1016/j.ijass.2018.08.001
[13] M. H. Santos, N. M. F. Oliveira, B. C. O. Maciel, “UAV control loops design tested in an HIL platform”, In: ABCM Proceedings of 24th ABCM International Congress of Mechanical Engineering, Curitiba, PR, Brazil, 2017.
[14] F.Splendor, N. A. Martins, I. M. S. Gimenes and J. A. Martini, “Design of an Autopilot for Cessna 182,” IEEE Latin America Transaction, Vol. 13, No. 1, pp. 27 – 36, Jan. 2015. DOI: 10.1109/TLA.2015.7040624
[15] A. Bittar, N. M. F. Oliveira, H. V. Figueiredo, “Hardware-In-the-Loop Simulation with X-Plane of Attitude Control of a SUAV Exploring Atmospheric Conditions”, Journal of Intelligent & Robotic Systems, January 2014, Volume 73, Issue 1, pp 271-287 DOI 10.1007/s10886-013-9905-8
[16] F. Garcia, L. Barnes, “Multi-UAV Simulator Utilizing X-Plane”, Journal of Intelligent and Robotic Systems, Vol. 57, pg 393–406, 2010
[17] Ş. A., B. Kürkçü, Ü. Kaynak, C. Kasnakoğlu “Control Loss Recovery Autopilot Design for Fixed-Wing Aircraft”, in: IFAC-PapersOnLine, Vol. 49, Issue 9, 2016, Pages 117-123 DOI: 10.1016/j.ifacol.2016.07.509
[18] H. Grankvist, “Autopilot Design and Path Planning for a UAV”, FOI – Swedish Defence Research Agency Defence and Security, Systems and Technology, Scientific report FOI-R–2224–SE, ISSN-1650-1942 December 2006
[19] P. Giemmi, et al., “Flight Phases With Tests of a Projectile-Drone Hybrid System”, IEEE Transaction on Control Systems Technology, Vol. 26, No. 6, pp. 2091– 2105, Nov. 2018. DOI: 10.1109/TCST.2017.2749559
[20] Y. Du, “Development of real-time flight control system for low-cost vehicle”, M.S. thesis, Cranfield University, 2011.
[21] D. V. Pandey, T. Uppal, S. M. Ratanekar, D. Bhattacharya, “Controller Development for a Rotary UAV using Physics Based Non-Linear Helicopter Model”, in: IFAC-PapersOnLine, Vol. 51, Issue 1, 2018, DOI: 10.1016/j.ifacol.2018.05.029.
[22] K. Eiker, S. Grouni, M. Tadjine, W. Zhang “Practical finite time adaptive robust flight control system for quad-copter UAVs”, Aerospace Science and Technology, Vol. 98, March 2020, DOI: 10.1016/j.ast.2020.105708
[23] Guoyuan Qi, et al., “High-Order Differential Feedback Control for Quadrotor UAV: Theory and Experimentation”, Electronics, Vol. 9 (12), 2001. Available: https://doi.org/10.3390/electronics9122001
[24] O. Mofid, S. Mohayen and W. Wong, “Adaptive Terminal Sliding Mode Control for Attitude and Position Tracking Control of Quadrotor UAVs in the Existence of External Disturbance”, IEEE Access, Vol. 9, pp. 3428 – 3440, Dec. 2020, DOI: 10.1109/ACCESS.2020.3047659
[25] Zhenbao Liu, et al., “Altitude Control for Variable Load Quadrotor via Learning Rate Based Robust Sliding Mode Controller”, IEEE Access, Vol. 7, pp. 9736 – 9744, Jan. 2019, DOI: 10.1109/ACCESS.2018.2890400
[26] J. M. O. Barth, et al., “Full Model-Free Control Architecture for Hybrid UAVs”, presented at the 2019 American Control Conference (ACC), Philadelphia, PA, USA, 29 August 2019. DOI: 10.23919/ACC.2019.8814993
[27] J. P. Kirk, “Evaluation of Unmanned Aircraft Flying Qualities Using JSBSim”, M.S. thesis, Department of Aeronautical Engineering, Air Force Institute of Technology (AFIT), Air University, Wright-Patterson AFB, Dayton, Ohio, U.S.A, 2016
[28] E. M. Cárdenas, P. J. Boschetti and A. Amerio, “Stability and Flying Qualities of an Unmanned Airplane Using the Vortex-Lattice Method”, in 47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exhibition, Jan. 2009, Orlando, Florida. DOI: 10.2514/6.2009-306

[29] MIL-F-8785C, Military Specification: Flying Qualities of Piloted Airplanes, United States Department of Defense, Washington, DC, USA, 1980.

[30] C. R. de Cos, et al., “High-Level Modular Autopilot Solution for Fast Prototyping of Unmanned Aerial Systems”, IEEE Access, Vol. 8, pp. 223827 – 223836, Dec. 2020, DOI: 10.1109/ACCESS.2020.3044098

[31] F. Kendoul, Y. Zhenyu, and K. Nonami, “Embedded Autopilot for Accurate Waypoint Navigation and Trajectory Tracking: Application to Miniature Rotorcraft UAVs", presented in 2009 IEEE International Conference on Robotics and Automation, Kobe International Conference Center, Kobe, Japan, May 12-17, 2009

[32] E. Capello, G. Guglieri and G. Ristorto, “Guidance and control algorithms for mini UAV autopilots”, Aircraft Engineering and Aerospace Technology, Emerald, Vol. 89, n. 1, p. 133–144, jan 2017. Available: https://doi.org/10.1108/aeat-10-2014-0161

[33] G. Cai, J. Dias and L. Seneviratne, “A Survey of Small-Scale Unmanned Aerial Vehicles: Recent Advances and Future Development Trends”, Unmanned Systems, Vol. 2, No. 2, pp. 175–199, (2014), DOI: 10.1114/S2301385014300017

[34] J. M. F. Fizá and L. B. Da Silva, “Piloto Autómático de uma Aeronave", Relatório de Trabalho Final de Curso, Licenciatura em Engenharia Eléctro e de Computadores, Instituto Superior Técnico, Universidade Técnica de Lisboa, Set. 2005.

[35] H. Y. Chao, Y. C. Cao, and Y. Q. Chen, “Autopilots for Small Unmanned Aerial Vehicles: A Survey", International Journal of Control, Automation, and Systems, Vol. 8 (1), pp. 36-44, 2010, DOI 10.1007/s12555-010-0105-z

[36] A. Sobron, D. Lundström, J. Staack and P. Krus, “Design and Testing of a Low-Cost Flight Control and Data Acquisition System for Unstable Subscale Aircraft", presented in 30th Congress of the International Council of the Aeronautical Sciences, DCC, Daejeon, Korea, Sep. 25-30, 2016.

[37] S. Choi, et al., “Development and Conversion Flight Test of a Small Tiltrotor Unmanned Aerial Vehicle”, Engineering Notes, Journal of Aircraft, Vol. 47, no. 2, pp. 730-732, March-April 2010.

[38] T. L. Jordan, J. V. Foster, R. M. Bailey and C. M. Belcastro, “AirSTAR: A UAV Platform for Flight Dynamics and Control System Testing”, presented at 25th AIAA Aerodynamic Measurement Technology and Ground Testing Conference, San Francisco, California, Jun. 5 – 8, 2006.

[39] R. Nelson, “Longitudinal Motion (Stick Fixed)” and “Lateral Motion (Stick Fixed)” in Flight Stability and Automatic Control, 2. ed., 1998. Ch 4 and 5, pp 164-167 and 203-205. ISBN 0-07-046273-9

[40] Paw, G. J. Balas, “Development and application of an integrated framework for small UAV flight control development”, Mechatronics, Vol. 21, n. 5, pp. 789-802, Aug 2011. Available: <https://doi.org/10.1016/j.mechatronics.2010.09.009>.

[41] D. Jensen, et al., “Development of polyuav: polytechnic university’s unmanned aerial vehicle”, AUVSI Undergrad Competition, Polytechnic University, Brooklyn, New York, USA, 2004.

[42] M. V. Cook, “Longitudinal Dynamics” and “Lateral-directional dynamics” in Flight Dynamics Principles: A Linear Systems Approach to Aircraft Stability and Control, 2. ed., Burlington, MA, USA Elsevier’s Science, 2007. Ch 6 and 7, pp. 138-169 and 174-205. ISBN 9780750669276

[43] B. Stevens, F. Lewis and E. Johnson, “Aircraft Dynamics and Classical Design Techniques”, in Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems. 2. ed. New Jersey, USA: John Wiley & Sons, Inc., 2003, ch 4, pp. 236-325. ISBN 9780471371458

[44] Stellaris® LM3S6965 Microcontroller: Data Sheet, TEXAS INSTRUMENTS, Austin, Texas, USA, 2014. Rev. 15852.2743. Available: https://www.ti.com/lit/ds/symmlink/lm3s6965.pdf.

MARCEO H. SANTOS was born in Patrocinio, MG, Brazil in 1990. He received the B.S. degree in Computer Engineering from University of Uberaba (UNIUBE) Brazil in 2013. He received the M.Sc. degree in applied electronic from Technological Institute of Aeronautics (ITA), Brazil in 2018.

NEUSA M. F. OLIVEIRA received the B.S. degree in Electrical Engineering from Federal University of Pernambuco (UFPE) Brazil in 1994. She received the M.Sc. and the Ph.D. degrees in control systems from Technological Institute of Aeronautics (ITA), Brazil in 1997 and 2004 respectively. She is an associate professor of applied electronics at ITA. Her research interests include autopilots for autonomous vehicle, sensor fusion and embedded systems.

ROBERTO D’AMORE received the B.S. degree in Electrical Engineering from Maua School of Engineering (IMT) Brazil in 1982. He received the M.Sc. and the Ph.D. degrees in applied electronic from Technological Institute of Aeronautics (ITA), Brazil in 1990 and 1998 respectively. He is an associate professor of applied electronics at ITA, and published the book “VHDL Description and Synthesis of Digital Circuits” (in Portuguese), LTC, 2005. His research interests include microelectronics, rapid prototyping, fuzzy processors, and communication protocols.