2014 International Conference on Unmanned Aircraft Systems (ICUAS)
May 27-30, 2014. Orlando, FL, USA

Guidance Software-in-The-Loop Simulation Using X-Plane and Simulink for UAVs

Adriano Bittar, Helosman V. Figuereido, Poliana Avelar Guimaraes and Alessandro Correa Mendes

Abstract—— **This paper introduces an new method to simulate a guidance algorithm running on Simulink that controls a fixed wing unmanned aircraft model running on the flight simulator X-Plane, which simulates the vehicle dynamics, sensors, and actuators. It presents the necessary settings for running simulations on small aircrafts in X-Plane and explains the UDP communication between the flight simulator and Simulink. The functions and codes to interpret X-Plane packets are explicated in details, and all codes are written in Matlab language in order to facilitate the development. As a final implementation we propose a guidance algorithm based on waypoints to validate the Software-In-the-Loop where the UAV needs to complete two missions that are also presented in this paper.**

I. INTRODUCTION

A couple of decades ago engineers and researchers would need to build a prototype of an airplane in order to investigate the effects of control laws or guidance algorithms that they have developed [1]. The prototype would be used as many times as possible, which eventually would cause an accelerated degradation, leading to manufacture another airplane. Besides the degradation, if the engineer decided to change a parameter in the wing, as an example, he would need to build another wing, consuming more time.

Nowadays Hardware-In-the-Loop (HIL) and Software-In-the-Loop (SIL) simulations are commonly used to evaluate controls and algorithms, because it easily allows to change the model (which would be the prototype), permits a fast development, increases the safety because you minimize the experimental flights, reduces the number of peripheral people involved, and consequently decreases the final cost of the project [2, 3].

A. Bittar was with the Instituto Tecnológico de Aeronáutica, São José dos Campos, Brazil when this work was perfomed and now is with Denver University, Denver, USA. (adrianobittar@gmail.com)

H.V. Figueiredo and A.C. Mendes are with the Instituto Tecnológico de Aeronáutica, São José dos Campos, Brazil.

This paper presents all the steps that are needed to implement a SIL simulation, where the model of the airplane runs on X-plane and the control and guidance algorithm runs on Simulink. It is described the configuration that is required to operate small unmanned aerial vehicles (SUAV) on X-Plane and Simulink functions developed to interpret X-Plane packets. Instead of using C functions as in [4, 5, 6] all the code is presented in Matlab language to facilitate the development.

The motivation of presenting this work is to help researchers that are starting working on simulations and wanted to go one step further on using a reliable mathematical model use a reliable flight simulator to visualize the behavior of the aircraft and test new control ideas.

Since there is a lack of detailed information on how to integrate Simulink with X-plane in literature, this paper can be used as a reference for engineers and researchers, who want to start developing Software or Hardware-in-The-Loop systems.

Neither the controls laws nor the aircraft's dynamic's equations are presented here, because the paper focuses on the integration of X-Plane and Simulink. The controls details for the architecture presented can be found at [7]. SUAV dynamic's equation can be found at [8].

The paper is organized as follows. The next section introduces the flight simulator X-Plane and the characteristics of the SUAV model used during the simulations. The two following sections detail the setting on X-Plane and Simulink, respectively. The fifth section details the proposed guidance algorithm. The overview implementation of Software-In-the-Loop is described in Section 6. Section 7 presents the results obtained. The conclusion discusses the validity of proposed modifications, based on the results obtained, as

978-1-4799-2376-2/14/$31.00 ©2014 IEEE

well as the research that still needs to be done in this project.

II. X-PLANE

The principle of operation of X-Plane is based on reading the geometric shape of any aircraft and then predicts how that aircraft will fly. The aircraft is broken down into small elements and then the forces on each element are calculated on an engineering process named “blade element theory”. Based on the mass of the aircraft and center of gravity the forces are converted into accelerations, which are integrated to generate velocities and positions.

X-Plane is certified by the U.S. Agency of Aviation (FAA -Federal Aviation Administration) to train pilots, because its method ensures a reliable system, since it is much more detailed, flexible, and advanced than the flight model based on stability derivatives that are used by most other flight simulators. The controllers developed and tested in X-Plane platforms have been successful when embedded in real aircraft [9] adding more credibility in the results that were obtained in this work.

The model airplane that was chosen for this paper is a Piper J-3 CUB 48S, manufactured by "The World Models." The main features of it, shown below, are found in the manufacturer's manual model.

1. Wingspan (Standard): 1.800mm [71.0 in]
2. Wing Area (Standard): 45.0 dm2 [698 sq in]
3. Weight: 2500g [5.5lbs]
4. Length: 1.200 mm [47 in]
5. Brushless Engine: 540W
6. Propeller: 12x7

The availability of a certified RC pilot that is trained on this model, the easy accessibility on Brazil’s market and the stability of the aircraft in flight are the main factors that this model was chosen for this research.

The model on flight simulator was built using the Plane-Maker (v 9.22), which is a program bundled with X-Plane. It is a specific tool that allows users to design any imaginable aircraft. Plane-Maker provides an interface to design a vehicle based on the physical specifications of the airplane (weight, wing span, control deflections, engine power, airfoil sections, etc.), whereas X-Plane simulator predicts how that built plane flies. Fig. 1(a) depicts the RC model. Fig. 1(b) shows the model built in Plane- Maker.

The surface’s positions commands on X-plane are normalized and can vary linearly from -1 to +1. The engine command is also normalized (0 and +1). The real deflection of the surface on the aircraft model is configured on the Plane-Maker. The equivalent ‘-1’ command deflection on X-Plane corresponds to a -20 degrees on the airplane model’s surface and the ‘+1’ to a deflection of +20 degrees. A deflection of a ‘+1’ in the engine command provides an output of 540 watts of power to the plane and a zero command turns off the engine.

III. X-PLANE SETTINGS

The adjustment that needs to be set to properly work with small aircraft on X-Plane, it is to configure the number of flight-model per frame, which establishes the amount of time that the flight simulator will calculate the forces in each frame. It can be adjusted on the Operations & Warnings panel (see Figure 2). A value between 6 and 10 is recommended. It will ensure better performance of the model during the simulations.

Figure 1(a). RC model Piper J-3

Figure 1(b). Piper J-3 on X-PLANE

Figure 2. Flight-Model per frame

An important feature of X-Plane is the ability to exchange information with the external environment through UDP (User Datagram Protocol). The UDP is a datagram oriented protocol without guarantees of packet delivery. This allows a high efficiency because there is no check as delivery or detection and error correction; it is possible to achieve high-speed data traffic compared with other protocols.

Simulink presents ready blocks for performing a UDP communication. And allows users to create functions where you can write code in the Matlab language for decoding the received UDP packet.

A. *Settling the network on X-Plane*

On this work the communication established between X-Plane and Matlab was performed on a single computer using the IP address "127.0.0.1", which is the network card's internal address . In this type of configuration is necessary to use four access points, two for Simulink (P1 and P3) and two for X-Plane (P2 and P4), and a single IP address, as shown in Figure 3.

The network configuration on X-Plane is shown in Figure 4 and Figure 5. In this work was used the version 10.25 (64 bits) of X-Plane released on December, 2013.

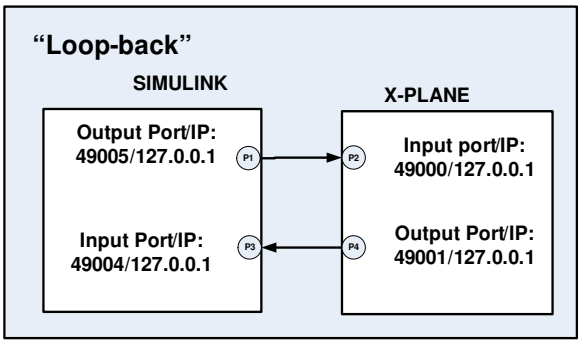

Figure 3. UDP Ports

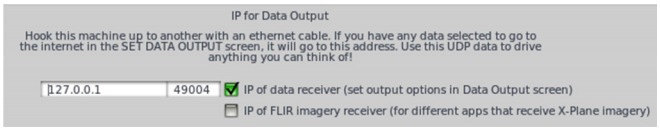

Figure 4. IP configuration

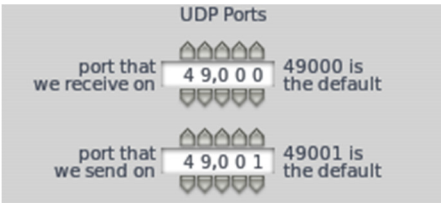

Figure 5. Ports configuration

B. *X-Plane Packet*

The data packet of X-Plane follows a standard format in which a header and a data set sequence are required. The header illustrated by Table 1 presents the following bytes:

1. Bytes 0-3 contain the "DATA" characters indicating that it is a data packet.

2. Byte 4 is an internal policy, and is not used for data transmission. It's normally defined as zero.

Table 1 – X-plane's Header

Position	**0**	**1**	**2**	**3**	**4**
Value	**"D"**	**"A"**	**"T"**	**"A"**	**I**

The following bytes are information about the state of the aircraft and can be selected according to Figure 6.

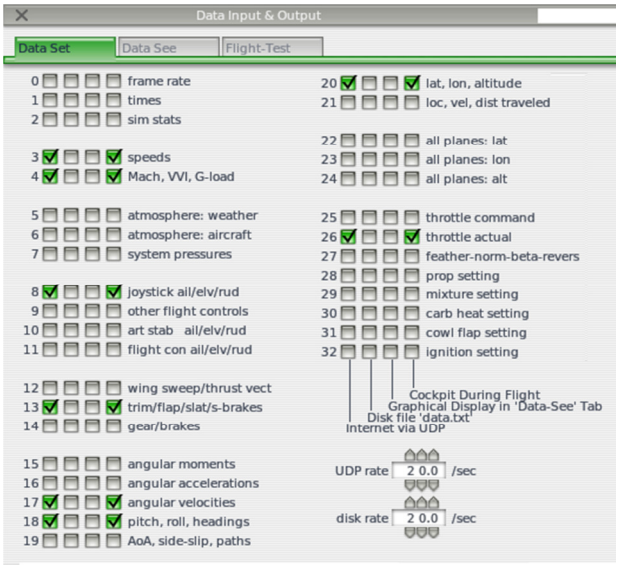

Figure 6. Data Input & Output

The data selected provide the navigation information obtained by the aircraft's sensors.

Each field selected in Fig. 6 generates a set of nine groups of 4 bytes each. So if the user selects 8 field on Fig. 6 the UDP packet will have 293

bytes (five bytes from the header, plus 8 x 9 x 4 bytes from the selected fields).

The first group of four bytes in a selected field presents a label to identify the set of data. The second group has the first instance of the data set; the third group has the second instance, etc.

The pattern of the X-Plane data follows the form illustrated by Table 2, which shows an example where only the Euler angles of the aircraft were selected. This set formed selecting label 18 (see Fig. 6), which shows the values of the angles of pitch, roll and heading (true and magnetic) .

The group of data is formatted as follows (remembering that indexes 0 to 4 form the header):

• GROUP 1: 5 to 8 bytes (L1, L2, L3, L4) which indicate the second selected parameter, in this case 18, because we want to Euler angles.

• GROUP 2: 9 to 12 bytes (P1, P2, P3, P4) are the pitch angle data.

• GROUP 3: 13 to 16 (R1, R2, R3, R4) presents the data of the angle of roll.

• GROUP 4: From 17 to 20 (D1, D2, D3, D4) we have information about the heading true.

• GROUP 5: From 21 to 24 (H1, H2, H3, H4) are the data of the magnetic heading.

• GROUP 6 to 9 (bytes 25 to 40) are not used in this case.

Each group of four bytes is formatted in simple floating point.

In Figure 6 are selected labels 03, 04, 08, 13, 17, 18, 20 and 26 that contains all the information needed by guidance and control blocks developed in this work.

Table 3 illustrates the information obtained with the selection of the labels.

Table 4 illustrates the commands that are sent by the control block to the X-Plane.

Table 3 – Parameter received from X-Plane

Number	Parameter	Label
1	Roll (ϕ)	18
2	Roll Rate (p)	17
3	Pitch (θ)	18
4	Pitch Rate (q)	17
5	Heading Ture (ψ)	18
6	Yaw rata	17
7	Altitude	20
8	Climb Rate	4
9	Airspeed	1
10	Latitude	20
11	Longitude	20
12	Profundor's Position	8
13	Aileron's Position	8
14	Engine's Power	26
15	Profundor's Trim tab Position	18
16	Aileron's Trim tab Position	18

Table 4 – Commands Sent

Commands	Label
Profundor's Position	8
Aileron's Position	8
Rudder's Position	8
Profundor's Trim tab Position	13
Aileron's Trim tab Position	13
Throttle	25

The engine power received by Simulink (Label 26) and the power (throttle) sent to the X-Plane (Label 25) are different. Label 26 provides the current engine power and label 25 supplies the power to be applied to the motor, this is due to the fact that the X-Plane simulates the delay time that the actuates consumes apply required power. This feature also allows the simulation of engine failures [10, 11].

Table 2 – Label Structure

Group 1 **Label 18**				**Group 2** **Pitch**				**Group 3** **Roll**				**Group 4** **Heading True**				**Group 4** **Heading Mag**			
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
L1	L2	L3	L4	P1	P2	P3	P4	R1	R2	R3	R4	D1	D2	D3	D4	H1	H2	H3	H4

IV. SIMULINK SETTINGS

The UDP communications blocks configuration in Simulink are illustrated in Figure 7.

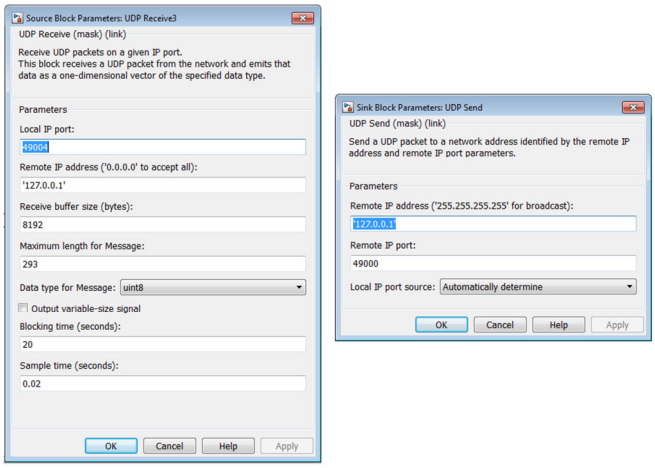

Figure 7. Ports configuration

A. *UDP Receiver Block on Simulink*

The packets received by X-Plane need to be broken in variables in order to be used by the control and guidance laws implemented in Simulink. For that we used two simulinks blocks: a byte unpack and an embedded Matlab Function. The unpack block is configured to output 5 bytes (the header), plus a matrix of 8 set of data with 9 fields, as shown in the left side of Figure 8.

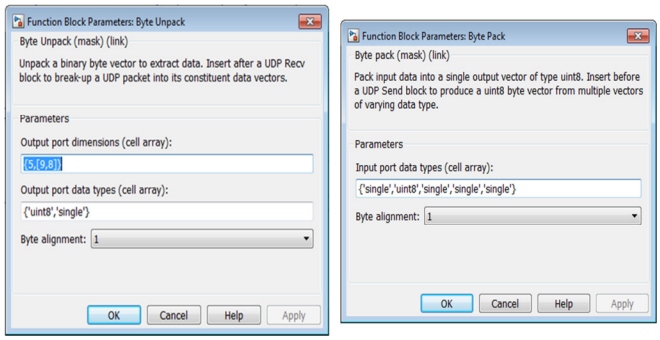

Figure 8. Byte Unpack Settings

The embedded function receives the matrix and converts the bytes in single format floating point, which are converted in double to be used by the following blocks.

```
1. function [velTrue_ktas,
   climb_rate,elevator, aileron,
   rudder, elevator_trim,
   aileron_trim, rudder_trim, q,p,r,
   pitch, roll, heading_true,
   latitutde, longitude, altitude,
   throttle_actual]    = fcn(data_in)
2. velTrue_ktas = data_in(4,1);
3. climb_rate = data_in(4,2);
4. elevator = data_in(2,3);
5. aileron = data_in(3,3);
6. rudder= data_in(4,3);
7. elevator_trim= data_in(2,4);
8. aileron_trim= data_in(3,4);
9. rudder_trim= data_in(4,4);
10. q= data_in(2,5);
11. p= data_in(3,5);
12. r= data_in(4,5);
13. pitch= data_in(2,6);
14. roll= data_in(3,6);
15. heading_true= data_in(4,6);
16. latitutde= data_in(2,7);
17. longitude= data_in(3,7);
18. altitude= data_in(4,7);
19. throttle_actual = data_in(2,8);
```

Figure 9 shows the final implementation with all blocks.

B. *UDP Sender Block on Simulink*

The inverse process is done in order to send commands to X-Plane, as seen in Figure 10. In this case commands from the controls blocks the inputs of an embedded Matlab function, and them are converted to the packed format expected by X-Plane. A "Byte Pack" block it's used to pack the data in an array of bytes to be send to X-Plane through the UDP Send Block.

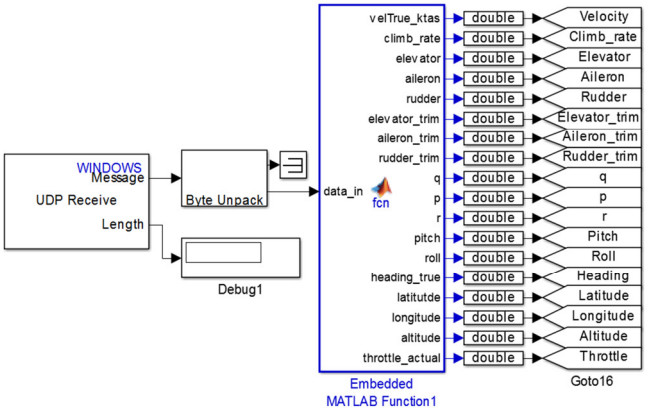

Figure 9. UDP receiver Block

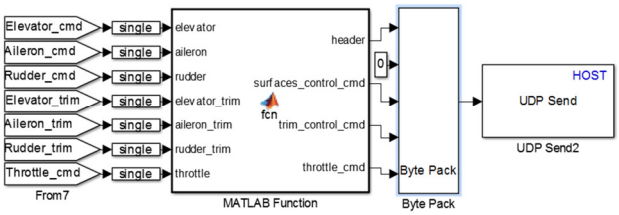

Figure 10. UDP Sender Block

The following code runs on the Matlab Function, and the Byte Pack block is configured as shown in the right side of Figure 8.

```
1. function [header,
   surfaces_control_cmd,trim_control_
   cmd, throttle_cmd]  =
   fcn(elevator, aileron, rudder,
   elevator_trim, aileron_trim,
   rudder_trim, throttle)

2. header = typecast(uint8([uint8(68)
   uint8(65) uint8(84) uint8(65)]),
   'single');
3. No_Comand = single(-999);
4. Surface_header =
   typecast(uint8([uint8(8) uint8(0)
   uint8(0) uint8(0)]), 'single');
5. surfaces_control_cmd =
   [Surface_header elevator aileron
   rudder No_Comand No_Comand
   No_Comand No_Comand No_Comand
   Surface_header No_Comand aileron
   rudder No_Comand No_Comand
   No_Comand No_Comand No_Comand];
6. Trim_header =
   typecast(uint8([uint8(13) uint8(0)
   uint8(0) uint8(0)]), 'single');
7. trim_control_cmd = [ Trim_header
   elevator_trim aileron_trim
   rudder_trim No_Comand No_Comand
   No_Comand No_Comand No_Comand
   Trim_header No_Comand aileron_trim
   rudder_trim No_Comand No_Comand
   No_Comand No_Comand No_Comand];
8. Throttle_header =
   typecast(uint8([uint8(25) uint8(0)
   uint8(0) uint8(0)]), 'single');
9. trhottle_cmd = [Throttle_header
   throttle No_Comand No_Comand
   No_Comand No_Comand No_Comand
   No_Comand No_Comand];
```

It's necessary to send the value "-999" parameters that the user don't intend to change, as in the *throttle_cmd*, where we filled the 7 other groups if the *No_comand* value.

V. Guidance Algorithm

The guidance block presented in this work generates reference signals to the control block that drives the vehicle to perform the desired mission [12]. The control block is not described in this paper, and can be found in [7]. There are numerous techniques that can be used for this purpose [13, 14, and 15]. Most algorithms used in unmanned aircraft focus on:

- Achieving the goal: the only concern in this type of algorithm is to reach a three-dimensional space point, called here waypoint, regardless of the path or efficiency.
- Optimization of the way: the UAV should find the smaller and more optimized path to achieve the goal or waypoint.

In order of simplicity and to easily demonstrate a guidance simulation using Simulink and X-Plane, in this paper we implement an algorithm that follows the first specification. The guidance block generates reference to the aircraft in order to make it reach the next waypoint. A waypoint is defined by latitude, longitude, and altitude, which are the parameters to be achieved by the UAV during the simulation.

The guidance is divided in two basic algorithms, one for switching waypoints and other to generate the reference heading based on latitude and longitude. Since the altitude comes directly from the waypoints, only the reference heading needs to be computed.

A. *Waypoint Switching*

In this kind of algorithm is necessary to control the waypoint order. The next waypoint is generated when the aircraft reaches the desired position, i.e., the current waypoint. This occurs when the aircraft enters a circle of radius (R) pre-settled, circling the current waypoint. If the last waypoint is reached the aircraft begins to circulate the last waypoint, or restarts the trajectory (this function can be selected). The trajectories were designed to have up to five waypoints.

In this Implementation, the waypoint is a central point, with the corresponding values of longitude and latitude, circled by three predefined radius, as

shown in Figure 11. If distance D between the UAV and central point of waypoint is less than or equal to the radius shown by the central circle, it is considered that the aircraft reached the desired waypoint. The others two radius has illustrative function only.

The distance D, given in degrees, is computed according to (1):

$$D_{graus} = \sqrt{(Lon_{wp} - Lon_{VANT})^2 + (Lat_{wp} - Lat_{VANT})^2} \quad (1)$$

It was also implemented a function called "home" that when triggered directs the UAV to the place where it took off. This is a way to be able to bring the aircraft back to the launch point at the end of a mission, or in the event of any failure or incident, is a safe way to abort the mission recovering the aircraft.

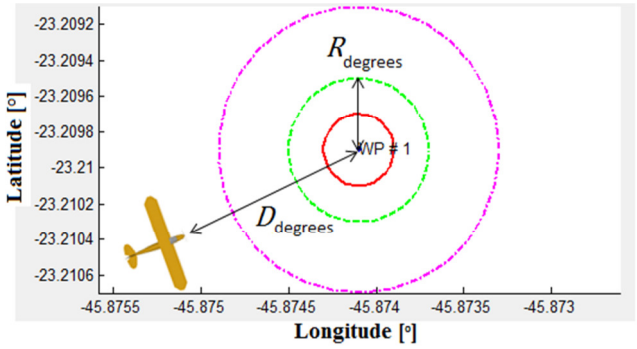

Figure 11. Waypoint

B. *Reference generation*

The heading angle to be followed by the aircraft is computed using LoS (Line of Sight), Figure 12. The computation of LoS depends on which quadrant the waypoint is in relation to the aircraft body system as [5], being necessary to use the function atan2 to gather information on the signs of the inputs in order to return the appropriate quadrant of the computed angle, which is not possible using the conventional arctangent function. . The equation used for computing the reference heading angle, given by (2), provides an angle from 0◦ to 360°, so it is necessary to transform the scale to -180° to +180° to force to the aircraft to follow the shortest possible path.

$$LoS = atan2\left[\frac{Lon_{wp} - Lon_{VANT}}{Lat_{wp} - Lon_{VANT}}\right] \quad (2)$$

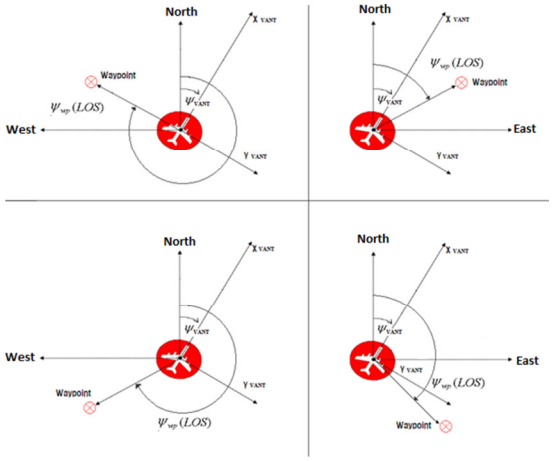

Figure 12. Line of Sight

VI. Implementation Overview

The final implementation of the work is according to Figure 13. The guidance block described get the latitude, longitude and altitude values from the "UDP receiver" block presented in Figure 8, and generates the reference signals for the control block. The control block analyzes the references and uses data from the aircraft to compute the commands that need to be applied on the aircraft's moving surfaces. These commands are sending to X-Plane trough the "UDP sender" block, Figure 9. Once X-Plane receives the commands it deflects the surfaces, and run the model physics, and generate navigation data that are sending to Simulink, restarting the process. In this work the update rate was 20 hertz, and can be adjusted by the "UDP rate", as shown in Figure 6.

All the important variables were logged in the workspace to be plotted after. So for each simulation it's possible to store all variables received from X-Plane.

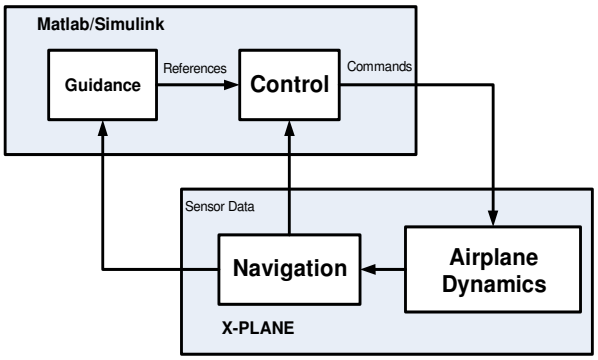

Figure 13. SIL Architecture

VII. RESULTS

The guidance simulation test is constituted of four waypoints; beside the initial point called "Home" according to Table 5. The trajectory selected has no big differences in altitude between the waypoint, although it has curves of almost 180 degrees.

Table 5 - Waypoints

	Latitude [°]	Longitude [°]	Altitude[1] [feet]
***Waypoint* Home**	-23.21976280	-45.8718492	2060
***Waypoint* 1**	-23.21910285	-45.8686291	2270
***Waypoint* 2**	-23.21726987	-45. 8645452	2340
***Waypoint* 3**	-23.21500788	-45. 8689079	2300
***Waypoint* 4**	-23.21783256	-45. 8710212	2250

[1] Altitude related to sea level.

In the first mission simulated, the UAV takes off from the waypoint "Home", go through the waypoint 1, waypoint 2, waypoint 3, and finally after reaching the waypoint 4, the UAV heads back to the Home waypoint, accomplishing the mission.

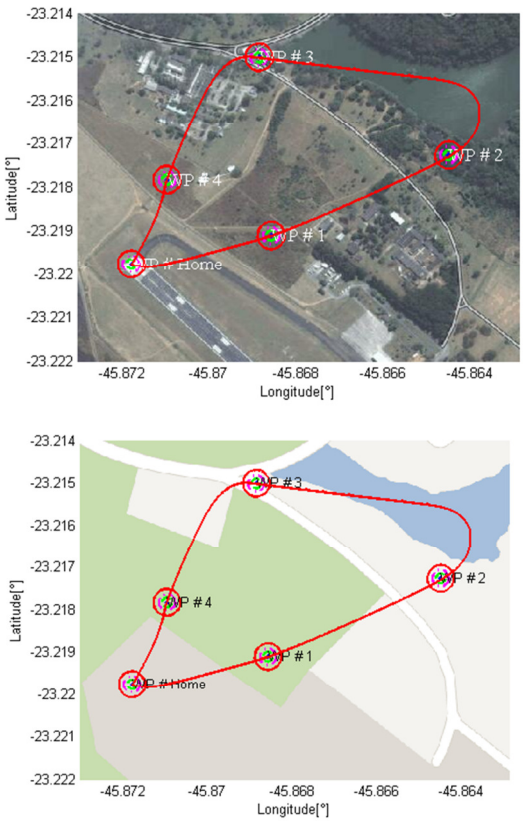

Figure 14. First Mission

Figure 14 illustrates the trajectory, which totalized 1.52 miles traveled by the UAV. The top side of the figure illustrates the satellite view of the area overflown; while on the bottom side is shown the map of the region. This pattern is repeated for the next simulation.

As defined, when the aircraft reaches the central circle of waypoint, which has radius equals to 0.0002°, we consider that the UAV reached the waypoint.

The second mission has the same waypoint, though the sequence is changed as follows:

***Waypoint* Home→ *Waypoint* 1→ *Waypoint* 2→ *Waypoint* 4→ *Waypoint* 3 → *Waypoint* 1→ *Waypoint* Home;**

This sequence was defined in order to force the UAV to performer curves to right and to left. Figure 15 shows the trajectory in 3D, whereas Figure 16 shows the figure the 2D plot.

Figure 17 is a screenshot taken from X-Plane, using the feature "Cycle 3-D Flight-Path" that traces the trajectory traveled by any aircraft. In this simulation the distance traveled was 2.84 km.

During this last mission the reference values of altitude and heading generated by the guidance algorithm, and the UAV response to these inputs, were monitored, Figure 18.

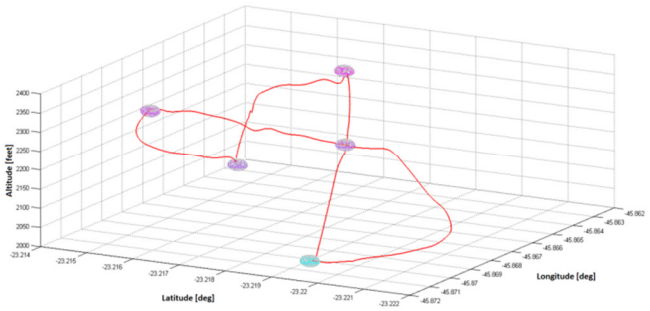

Figure 15. Second Mission: 3D trajectory

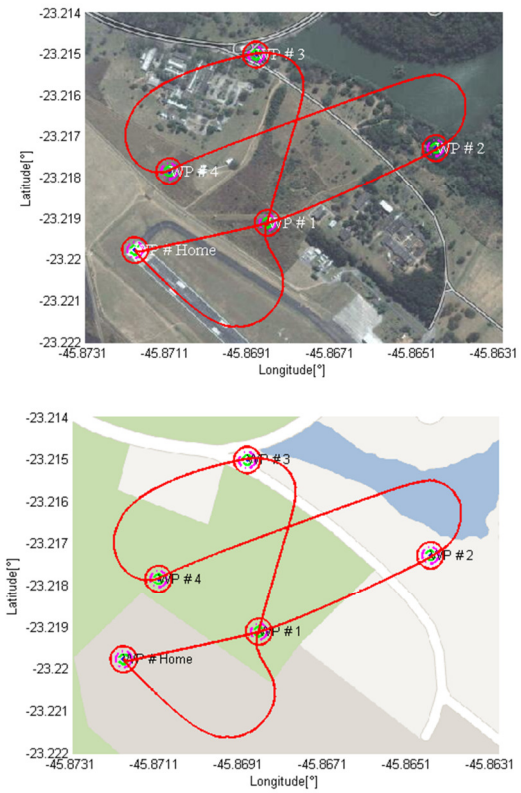

Figure 16. Second Mission: 2D trajectory

The results generated are represented by dashed lines, and the response of the aircraft by the solid line. The discontinuities presented in the graph direction, occurs due to the passage of 0 degrees to 360 degrees.

VIII. Conclusion and Future Work

With the presented architecture, it is possible to integrated X-Plane and Simulink in order to perform all kinds of simulation related to aircrafts.

Figure 17. X-Plane Screenshot: Superior View

The block presented in Simulink and the use of Matlab language only speeds the development and implementation of SIL platforms. To improve the time taken to process the packets in Simulink the user should run the program in the Rapid Accelerator mode, instead of normal mode, when using an infinite time. This will make the Simulink to compile the program generating MEX-FILE, which will be read by the Matlab engine.

The guidance application used to validate the SIL achieved expected results. Although not shown in this paper, the guidance and controls blocks in Simulink are not complex and can be substituted by other algorithms without modification on the UDP Sender and the UDP receiver. It makes the system modular and able to be reusable by other engineers and researches

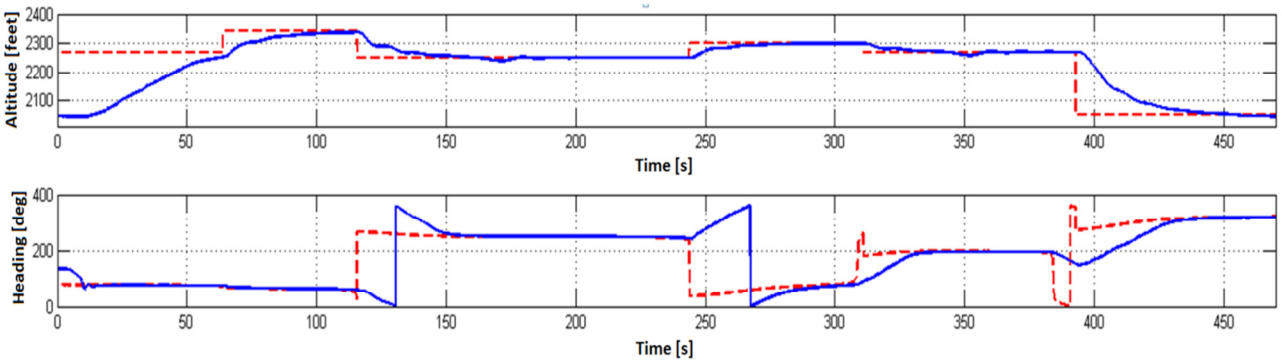

Figure 18. Altitude and Heading Response

The SIL simulation used to test the guidance algorithm proved to be an easy and fast way of verify new ideas without having to building a scale prototype, leading to minimize cost and time.

Although in this paper only a customized SUAV model was used, X-Plane's library has many other aircraft models that could have been used to verify the algorithm.

The future work will be based in to fronts: developing a method to integrate the flight simulator and Simulink through plug-ins and to use X-plane features to evaluate guidance algorithm using when the aircraft is exposed to disturbances like lateral wind or turbulences, as in [15].

The code and the Simulink file will be soon available for download on the Matlab Exchange Files.

REFERENCES

[1] J. S. Eterno, J. L. Weiss, D. P. Looze, and A. S. Willsky, "Design Issues for Fault Tolerant-Restructurable Aircraft Control", the 24th IEEE Conference on Decision and Control, pp. 900–905, December 1985.

[2] P. Chudy and P. Rzucidlo, "HIL Simulation of a Light Aircraft Flight Control System", the 31th IEEE Digital Avionics Systems Conference, pp. 6D1-1–6D1-12, October 2012.

[3] R. C. B. Sampaio, M. Becker, A. A. G. S., L. W. Freschi, M. P. Montanher, "Novel SIL Evaluation of an Optimal H1 Controller on the Stability of a MAV in Flight Simulator", the 2013 Aerospace Conference, pp. 1-8–6D1-12, March 2013.

[4] L. R. Ribeiro and N. M. F. Oliveira, "UAV autopilot controllers test platform using Matlab/Simulink and X-Plane," 2010 IEEE Frontiers in Education Conference (FIE), p. S2H-1-S2H-6, Oct. 2010.

[5] A. Bittar, N.M.F. Oliveira, D.A. Carvalho, "Control Architecture From Pegasus Autopilot: Design and Software-In-The-Loop Simulation," Congresso Brasileiro de Automática (CBA), pp. 4146-4153. October, 2012.

[6] Figueiredo, H.V.; Saotome, O. "Simulation Platform for Quadricopter: Using Matlab/Simulink and X-Plane", Robotics Symposium and Latin American Robotics Symposium (SBR-LARS),pp 51-55. October 2012.

[7] A. Bittar, N.M.F. de Oliveira, "Hardware-in-the-loop simulation of an attitude control with switching actuators for SUAV,". International Conference on Unmanned Aircraft Systems (ICUAS), pp. 134-142. May, 2013.

[8] R. W. Beard and T.W. McLain, *Small Unmanned Aircraft.* Princeton University, New York: First Edition, 2012.

[9] R. Garcia, and L. Barnes, "Multi-UAV Simulator Utilizing X-Plane", *IEEE J. Intelligent & Robotic Systems*, vol. 57, pp. 393-406, Oct. 2009.

[10] LAMINAR RESEARCH. X-Plane description. X-Plane Manual, 2013.

[11] X-Plane Wiki. Simulating Equipment Failures. Available in: <http://wiki.x-plane.com/Advanced_Simulation_in_X-Plane_10>. Last Access: 06/02/2012.

[12] G. Chowdhary, J. Ottander, and D. Guggenheim. "Low Cost Guidance, Navigation, and Control Solutions for a Miniature Air Vehicle in GPS Denied Environments". UAV Research Facility. School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332.

[13] Han, D., Kim, J., Lee, D., Cho, K. and Cho, S.,2008, "Autonomous Flight Test Using Angle Of UAV's Velocity Vector", Proceedings of the International Conference on Control, Automation and Systems, Seoul, Korea.

[14] S. R. B. Santos., C. L. Nascimento, S. N. Givigvi, et al., "Experimental Framework for Evaluation of Guidance and Control Algorithms for UAVs", Proceedings of Brazilian Congress of Mechanical Engineering, 2011.

[15] A. Bittar, N. M. F. Oliveira, H. V. de Figueiredo. "Hardware-In-the-Loop Simulation with X-Plane of Attitude Control of a SUAV Exploring Atmospheric Conditions". Journal of Intelligent & Robotic Systems. Pp. 1-17.