TESTBED FOR CONTROLLES OF LONGITUDINAL MOVEMENTS OF AIRCRAFT

Sérgio Ronaldo Barros dos Santos, sronaldo@ita.br.
Neusa Maria F. Oliveira, neusa@ita.br.
Instituto Tecnológico de Aeronáutica
Pça Mal. Eduardo Gomes, 50 - Vila das Acácias - 12228-900,
São José dos Campos - São Paulo - Brazil

Abstract. *The developed system described here is intended to be used as test platform to aircraft controllers. The system is made of two different blocks executing different tasks: one block implements the controllers necessary to the desired movement and the other block implements the model of the aircraft when performing the desired movement. Traditionally, the aircraft movements are classified as longitudinal and lateral movement and under specific flight conditions these movements are considered uncoupled, which makes possible to study them separately. The structure of the developed system should be used for both movements in an aircraft, changing the controllers and the aircraft model depending on the case in study. Here it will be described the development of a digital control system for the longitudinal movement of an aircraft. The dynamic model to represent the target aircraft, an Piper Dakota, is a two degrees of freedom model, describing the ascending and descending movement, the velocity variation in the vertical movement and the altitude change as a function of the aircraft climbing or descent. The longitudinal dynamic model is characterized by the pitch angle. The reference value of the pitch angle depend on the desired values of the aircraft altitude and velocity. The aircraft dynamic equations were implemented in MATLAB/SIMULINK. Longitudinal controllers were designed with the objective of maintaining the aircraft stability through the specified operations conditions. The designed controllers were discretized and implemented in a hardware, Rabbit 2000 TCP/IP development kit, with a processor dedicated to the control task. Data transmission between the hardware implementing the controllers and the PC with the MATLAB/SIMULINK implementing the aircraft dynamic model is done by the UDP (Uniform Datagram Protocol) communication protocol using the Ethernet module that exists in both system present in the system. The results show that the developed system structure is appopriate to test the controllers already implemented in computers dedicated to the controlling task. Also the graphics facilities of the MATLAB/SIMULINK environment represents an important characteristic of this structure. Finally, the designed controller implemented in the developed system reaches the specified requirements successfully.*
Keywords*: Longitudinal autopilot, PI and PID controller, UDP communication, Speed and altitude control.*

1. INTRODUCTION

The function of automatic control in an aircraft is performed by a control system known as Automatic Flight Control System (AFCS) or autopilot. Autopilot is a device used to control the pattern flight in an aircraft without human intervention (McLean, 1990). Furthermore autopilot has as principal function to stabilize the dynamic characteristic in an aircraft through the altitude, direction and speed control. The autopilot system acts directly in the mechanical system of an aircraft (Pallet, 1979) and (Roskam, 1995).

Nowadays, more and more automatic control systems are being studied and explored due to the possibility of this system to be used in an Unmanned Aerial Vehicles (UAV) (Sagahyroon, 2004) and (Ernst, 2006). This was the reason for the development of a test platform in a dedicated hardware.

The motion of an aircraft in free flight can be extremely complicated. The airplane has three translation motions (vertical, horizontal and transverse) and three rotational motions (pitch, yaw and roll). Traditionally, the aircraft movements are classified as longitudinal and lateral movements. The X - force, Z - force and pitching moment equation embody the longitudinal motion and the Y - force, rolling and yawing moment equation form the lateral motion. To separate the equation in this manner, the longitudinal and lateral motion must not be coupled. These are all reasonable assumptions provided the airplane is not in very rapid maneuver. Therefore, the system structure developed can be used to both aircraft motion, changing the controller and aircraft model depending on the case study (Nelson, 1998) and (Blakelock, 1991).

The proposed and developed test platform of airplane controllers is introduced herewith. Firstly, the problem that is being addressed – the structure of the loop to be implemented – is exposed. Then, we exposed how the problem was solved, the tools used and how to connect them to work together in order to implement aircraft control loops (Manseur, 2006). Finally, we show the results obtained using the platform with designed controllers to control the pitch angle of the target aircraft.

2. PROBLEM STATEMENT

The system shown in this work is inspired in the hardware in the loop approach. To develop this work, the chosen model was the dynamic model for the longitudinal motion representing the target aircraft, a Piper Dakota. It is a two degree of freedom model which describes the ascending and descending motion during flight condition, the velocity variation during vertical motion and the change of altitude realized due to climbed and descended aircraft (Bender, 2006).

The longitudinal control consists in attitude, velocity and altitude control in an aircraft. The speed control is essential to a longitudinal controller, due to existence of a large sensibility of speed in relationship change of pitch angle. Controlling the velocity will exist a direct relationship between the pitch angle and the altitude variation, becoming a important characteristic to a guidance system (Roskam, 1995) and (Huang, 2002). The longitudinal autopilot proposed is divided in three control loop, as show in Fig. 1.

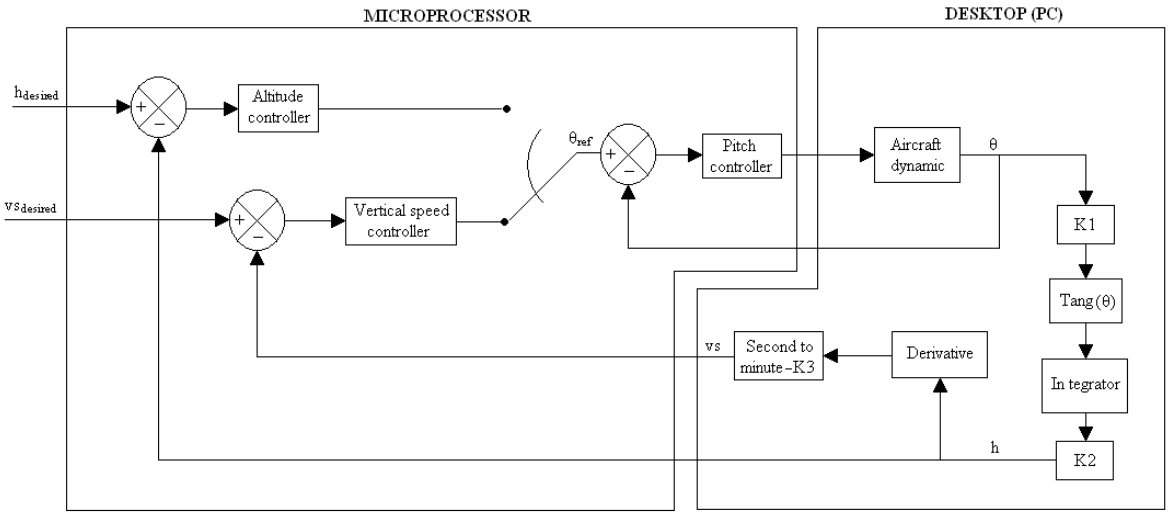

Figure 1. Block Diagram of the proposed longitudinal autopilot

Attitude loop: The pitch error signal generated by a comparison between the desired pitch angle and the pitch angle obtained of the aircraft dynamic model, is used as input to the compensator (pitch controller) and the compensator output is the command to elevator deflection. This output signal is responsible for command the pitch angle of the system to be equal the pitch angle desired.

Velocity loop: The vertical speed control of aircraft is done through the pitch angle controller. The vertical speed controller output is connected directly in the comparator input of pitch loop, being used to control the climbing and descending of an aircraft.

Altitude loop: The altitude error signal is used by altitude controller to generate a control signal to the pitch angle. The altitude controller is connected directly in the comparator input of pitch loop. Ideally, the altitude loop is used to control of altitude aircraft, when the altitude error signal is small. When the error signal is high, the speed controller should be used until the altitude error signal to reach a small value.

Figure 1 represents the general autopilot control loops and shows the tasks done by PC and the microprocessor dedicated to the test platform. The comparator and digital controllers are implemented in the microprocessor. Depending on the aircraft and the movement studied, different aircraft transfer function can be implemented. The longitudinal motion in an aircraft is controlled mainly by elevator servomotor (Albanes, 2001) and (Steer, 2004).

3. PROBLEM SOLUTION

The development of the platform proposed is discussed in this section. The system was divided in two distinct blocks, executing different tasks: one block implements the controllers necessary to the desired movement, and the other block implement the aircraft model in a Personal Computer (PC) executed in MATLAB/SIMULINK.

The microprocessor chosen to the controller's implementation was the Rabbit 2000 TCP/IP. The mainly components of the hardware Rabbit 2000 TCP/IP development kit are: Microprocessor Rabbit 2000 operating in 18.4 Mhz, flash memory of 256 kbytes, SRAM data memory of 128 kbytes, Ethernet port 10BaseT, serial port RS - 232 and RS - 485, real time clock, 4 digital input, 4 digital output, 7 timers of 8 bits, watchdog and size of 11 cm x 12 cm. The microprocessor Rabbit 2000 has two options of the Integrated Development Environment (IDE), the dynamic C tool distributed by Rabbit and the WinIDE tool of Softools. These tools use C language and a debugger to the program verification, however these tools does not emulate the programs (Rabbit Semiconductor, 2006a, 2006b).

Figure 1 shows the general test platform proposed in two distinct blocks. In PC was implemented the dynamic model of Piper Dakota, the transfer function used was θ(s)/δe(s). This model is the relationship between the pitch angle and the elevator deflection, the output value supplied corresponds to pitch angle, determined through the velocity and altitude value used as reference.

The pitch angle stabilized in an aircraft during flight condition produces the altitude alteration defined as ascending and descending rate of the aircraft. This way to control the altitude is the same that controls the ascending and descending rate in an aircraft (Roskam, 1995). Therefore, the altitude value can be obtained from the pitch angle given by the dynamic model, using the Eq. (1).

$$h(t) = \int \frac{d}{dt} h = k2 \int tan(k1.\theta) d\theta \tag{1}$$

The transfer function that represents the altimeter model to target aircraft is not used. Notice that the altitude value is obtained through the correct adjust of the parameters k1 and k2 in Eq. (1). These parameters will give a response very close to the real altitude value, replacing the altimeter model to the target aircraft. The k1 and k2 were adjusted to 2.98 and 23.45 respectively. The vertical speed is determined using the altitude variation response generated by pitch angle present in aircraft (Roskam, 1995). This parameter is given in feet/s and is converted to feet/min as k3 is adjusted to 60. The vertical speed approximate can be obtained using the Eq. (2).

$$Vs(t) = \frac{d}{dt} h(t) \tag{2}$$

Figure 2 shows the dynamic model implemented to the Piper Dakota in MATLAB/SIMULINK.

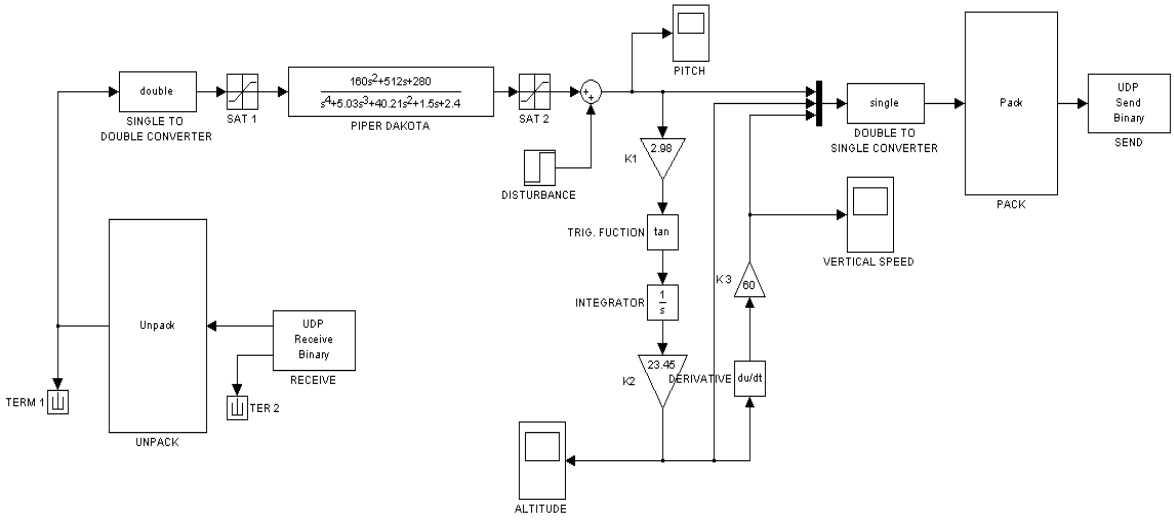

Figure 2. Dynamic model of the Piper Dakota implemented in MATLAB/SIMULINK

4. CONTROLLERS DESIGN

The linearized longitudinal transfer function between the pitch angle and the elevator deflection that represents the target aircraft model, a Piper Dakota, can be seen in Eq. (3).

$$G(s) = \frac{\Delta\theta}{\Delta\delta e} = \frac{\theta(s)}{\delta e(s)} = \frac{160s^2 + 512s + 280}{s^4 + 5.03s^3 + 40.21s^2 + 1.5s + 2.4} \tag{3}$$

Most literature that deals with aircraft control uses continuous control system as design technique. Actually, a lot of aircrafts still have continuous autopilots; however, they are based on analog circuit. As the controllers are implemented in digital microprocessor, the controllers are designed using the continuous control technique, and then discretized using bilinear transformation method (Choe, 2002). (Frutiger and Kim, 2003) has done a similar development for the Piper Dakota pitch control loop and it was an important reference to understand the system and it was also used to compare the results obtained in this paper.

The pitch controller was designed using the Bode method in the frequency domain (Turkoglu, 2008). The compensator was designed to increase the phase margin of the system and is represented as in Eq. (4).

$$D_p(s) = 1.5\frac{s+3}{s+20} \quad , p > z \tag{4}$$

The PI (Proportional and Integral) control was chosen to the vertical speed controller design. (A continuous controller properly designed guarantees perfect reference tracking, as well as zero steady state error though in the discrete controller approximate this is not exactly true, except when is used a high sampling frequency (Ogata, 2003)) The PI continuous controller was obtained as shows the Eq. (5).

$$D_v(s) = 0.002 + \frac{0.7}{s} \tag{5}$$

For the altitude controller, PID (Proportional, Integral and Derivative) control was chosen because the proportional action is essential to a good response of the control system, the integral action eliminates the off-set error, and the derivative action turns the controller response faster (Ogata, 2003). Equation (6) represents the PID controller designed.

$$D_a(s) = 0.3 + 0.008s + \frac{0.01}{s} \tag{6}$$

The bilinear transformation technique was used to obtain the digital controllers based on the emulation of the pitch, vertical speed and altitude continuous controllers (Hermerly, 1996). Bilinear transformation is such that the Eq. (7) is substituted into the continuous controllers to be discretized.

$$s = \frac{2}{T}\frac{z-1}{z+1} \tag{7}$$

The sample time used to determine the pitch, vertical speed and altitude discretized controllers was equal 100 μs. The discretized controllers are shown in Eq. (8), (9) and (10).

$$D_p(z) = \frac{0.862z - 0.6375}{z} \tag{8}$$

$$D_v(z) = 0.002 + 0.00004\frac{z+1}{z-1} \tag{9}$$

$$D_a(z) = 0.3 + 16\frac{z-1}{z+1} + 0.00005\frac{z+1}{z-1} \tag{10}$$

Considering the discretized transfer function Eq. (8), (9) and (10), the differential equations for the compensator, PI and PID to be implemented in the digital controller were obtained and they are written as in Eq. (11), (12) and (13) respectively.

$$pit_{cp} = 0.575e(k) + 0.425e(k-1) \tag{11}$$

$$vel_{pi} = 0.002e(k) + 0.00004[e(k) + e(k-1)] + vel_{pi}(k-1) \tag{12}$$

$$alt_{pi} = 0.3e(k) + \left\{16[e(k) - e(k-1)] - alt_{pid}(k-1)\right\} + \left\{0.00005[e(k) + e(k-1)] + alt_{pid}(k-1)\right\} \tag{13}$$

5. IMPLEMENTATION

The controllers were designed with the objective of maintaining the aircraft stability through the specified operations conditions. The designed controllers were implemented in hardware, Rabbit 2000 TCP/IP Development Kit, through a program written and debugged using Dynamic C tool and recorded in the flash memory. Figure 3 shows the the structure of the developed system.

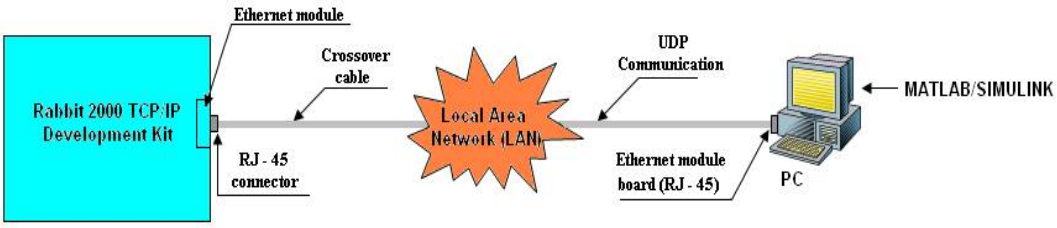

Figure 3. Diagram of the developed system

The data transmission between the controllers implemented in the hardware and the aircraft dynamic equation implemented in MATLAB/SIMULINK executed in a PC was done through UDP (Uniform Datagram Protocol) communication protocol using the Ethernet module existent in both systems. Afterwards, the data transmitted by MATLAB/SIMULINK is readily available in socket created in the Rabbit 2000. The *receive_data()* function used in the program will make the reading of 12 bytes received in data package. The data is stored in an input buffer using the *udp_recv(&udpSocket, in_udpBuffer, sizeof(in_udpBuffer))* instruction . The structure of the data package used in the comunication is shown in Fig. 4. It uses the IEEE standard for normalized single precision floating-point number, where "S" is the sign bit that determines the sign of the number – it assumes the value 0 or 1, representing a positive or negative number, respectively; "E" is the eight-bit biased exponent of the number, base 2 and "L" is the 23-bit fractional part of the number mantissa, base 2.

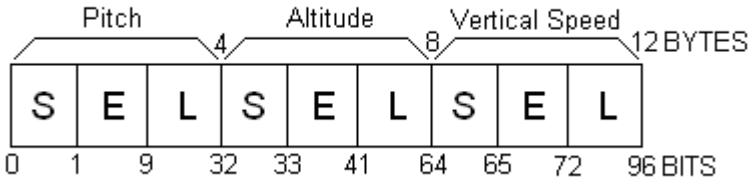

Figure 4. Structure of the used data package

A function named *compare ()* was declared in the program implemented in Rabbit 2000. In this function, the vertical speed reference value and the altitude reference value are declared and compared with the status of the aircraft to generate the error signal to be used by the controllers. The vertical speed reference value was defined as 500 feet/min and the altitude reference as 200 feet.

As we are dealing with two different control loops (besides the inner pitch loop), it is necessary to distinguish when each controller is actually in control. Consider Figure 1. Initially the vertical speed output (vs) is compared with the vertical speed reference ($vs_{desired}$) value generating an error signal, used to determine the vertical speed controller (PI) output. This output is used as pitch reference, necessary to keep vertical speed according to the vertical speed reference value, keeping the steady state error nearly zero.

The pitch reference is compared with the pitch obtained in feedback control system, generating another error signal. This new error is used by pitch controller (compensator), generating an output value used to produce the elevator deflection, consequently, a pitch angle in aircraft. When the aircraft reaches 90% of altitude reference value, the vertical speed controller (PI) will be disconnected and then the altitude controller (PID) will be connected, thus controlling the loop and keeping the pitch and vertical speed value constant. When the altitude controller is connected, the compare () function compares the altitude reference value to the aircraft altitude (h). The error obtained is the input of the PID controller which gives as its output the new pitch reference to be used in the inner loop. The pitch and vertical speed will have the zero value when the altitude reaches the desired value.

The data of the pitch controller output consists of 4 bytes. Theses bytes are stored in an output buffer using the function *transmit data ()* implemented in the microprocessor program. This data is transmitted to the aircraft dynamic model implemented in MATLAB/SIMULINK using the Ethernet module of Rabbit 2000. The transmission of data

packet is done using the *udp_sendto(&udpSocket, out_udpBuffer, sizeof(out_udpBuffer), remoteIP, remotePort)* instruction.

Figure 5 shows a program fragment in which the functions *receive_data()* and *transmit_data()* appear.

```
int transmit_data()
  {
       memcpy(&out_buffer[0],&out,4);   // Convert 32 bits floating point output data to a byte array output data with 4 bytes
       udp_sendto(&udpSocket, out_buffer, sizeof(out_buffer), remoteIP, remotePort);   // Send the output data to MATLAB
  }
                                                    .
                                                    .
                                                    .
int receive_data()
  {
       int pass;
       pass = udp_recv(&udpSocket,in_buffer, sizeof(in_buffer)); // Receive data package with 12 bytes of data
       if (pass >= 0)
           {
                memcpy(&in_pitch,&udpBuffer[0],4);   // Store the pitch angle obtained in feedback control system
                memcpy(&in_alt,&udpBuffer[4],4);      // Store the aircraft altitude obtained in feedback system
                memcpy(&in_vel,&udpBuffer[8],4);    // Store the V. Speed obtained  in feedback control system
                compare();          // Call the compare function
                controller();       // Call the controller function
                transmit_data();  // Call the transmit function
           }
  }
```

Figure 5. A program fragment with functions used in the communication between Rabbit 2000 and PC.

The MATLAB/SIMULINK uses the UDP block to send and receive data (see blocks UDP send Binary and UDP Receive Binary in Figure 2). The floating point data in single precision format (32 bits) with six decimal digits was used. This type of data was used in UDP communication between the MATLAB/SIMULINK and the Rabbit 2000 microprocessor for compatibility reasons, since it is understandable by both of them. However, the block that implements the aircraft dynamic transfer function uses floating point data in double precision format (64 bits) with ten decimal digits. Therefore, a conversion block was used to make the change from the single precision to double precision and vice-versa, during the microprocessor data transmission and reception.

The pitch, vertical speed, and altitude obtained from the dynamic model are transmitted using sample time. These parameters are multiplexed and converted from double precision to single precision format. So the 12 bytes are put into packages and then, these packages are transmitted by Ethernet module. The sample time used by the UDP block was adjusted to 100μs.

6. RESULTS

In this section, we present the results obtained using the structure proposed, Fig. 1. First of all, consider the situation: an aircraft in an equilibrium flight condition (at an original altitude, h_{orig}, and null vertical speed, vs=0, and pitch angle, θ=0) is commanded to change its altitude to a desired altitude ($h_{desired}$) (let´s suppose $h_{desired} > h_{orig}$). Fig. 6 represents how we intuitively expect the aircraft behaves in such situation:

- When the command is given, the aircraft changes its vertical speed and the aircraft becomes to increase its altitude. Related to this climbing movement it is expected the aircraft assumes a non null pitch angle, with a value related to the aircraft vertical speed. (During this movement, in the proposed autopilot, Fig. 1, the vertical speed controller is controlling the loop.)

- When the aircraft is close to the desired altitude, an transition stage is expected, when the vertical speed and the pitch angle should decrease to zero. (At this transition, there is a switch between the two controllers in the proposed autopilot and the altitude controller becomes to control the loop.)

- When the aircraft reaches the desired altitude, it reestablishes the trimmed flight, with θ=0 and vs=0. (at this stage, in the proposed autopilot, the altitude controller is controlling the loop.)

In an ideal world, with systems responding instantaneously and with perfect controllers, the expected response of the aircraft altitude (h), pitch angle (θ) and vertical speed (vs) would be like in Fig. 7. It represents graphically the

behavior of each parameter in the control system loop in the addressed problem and which behavior was intuitively presented in Fig. 6.

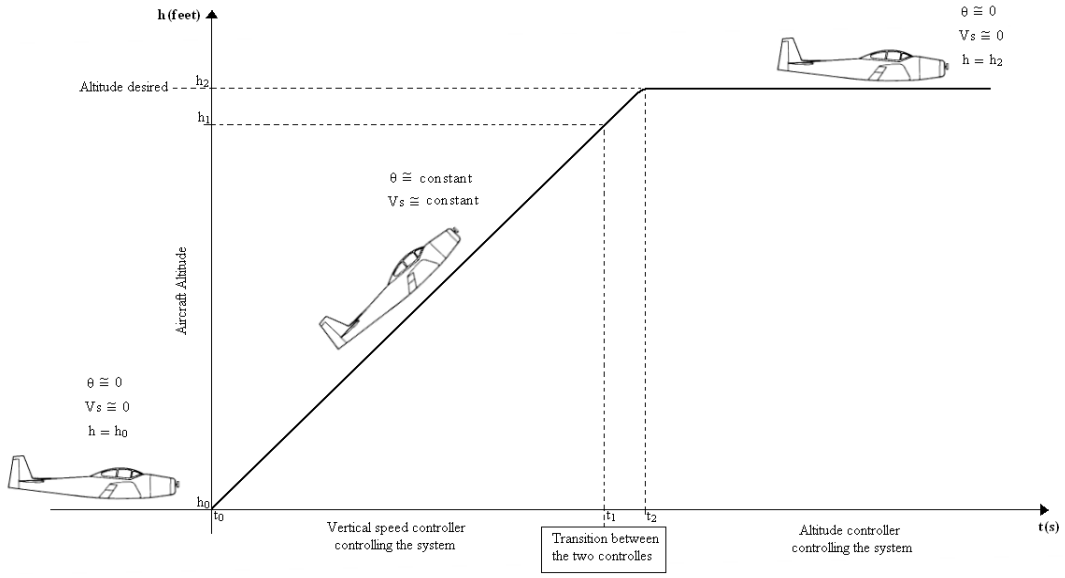

Figure 6. Aircraft response to a command for altitude change

Therefore, using the proposed system structure (Figure 1) and establishing the desired vertical velocity and altitude, the response of the cited parameters should be close to the graphics in Fig. 7. Of course, they would not be equal, since in the proposed control loop we have a physical system.

In order to have real system responses (and not ideal responses as in Figure 7) to compare with the responses of the proposed structure, the control loop using continuous controllers was implemented in MATLAB/SIMULINK.

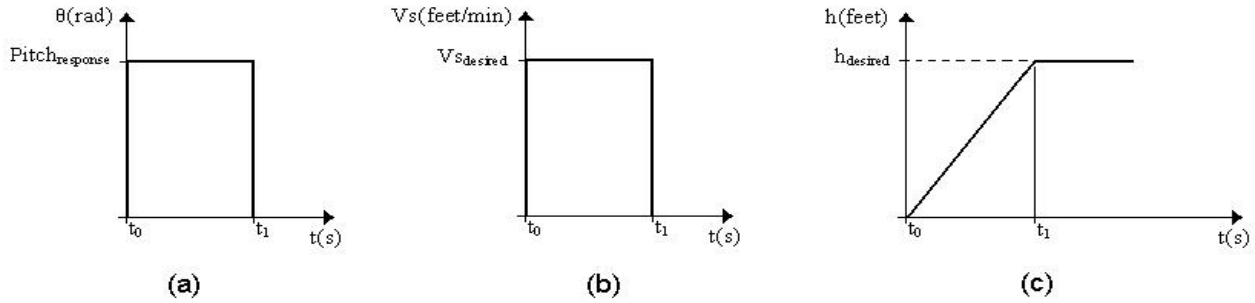

Figure 7. Pitch (a), vertical velocity (b) and altitude (h) ideal responses

Figures 3, 5 and 7 show the Piper Dakota response using continuous controllers implemented in MATLAB/SIMULINK. Figures 4, 6 and 8 show the Piper Dakota response using digital controllers implemented in the proposed system.

Consider 0 feet of altitude, 0 feet/sec of vertical velocity and 0 rad of pitch angle as the initial aircraft conditions. Then, the proposed control loop begins to control de aircraft using as reference values 500 feet/m in vertical speed and 200 feet in altitude. The response of the aircraft will be presented by the behavior of the three aircraft status, pitch angle, vertical velocity and altitude.

Figure 3 and 4 represents the pitch response to the given initial conditions and reference values. The system responds with a pitch angle of 0.09 rad which corresponds to 5.15^o. The response in the proposed system, Fig. 4, is similar to the response to the continuous system, Fig. 3, which points to a good performance of the digitalized controller implemented in the proposed system structure. When the altitude of the aircraft reaches the input value after 24s, the pitch angle becomes zero.

Comparing the ideal response with the MATLAB/SIMULINK continuous simulation response and the proposed structure response, we verify that they represent the same situation commented in Figure 6.

The MATLAB/SIMULINK response and the proposed structure response show problems intrinsic to physical systems, such as the delay in the actuator modeled in the aircraft dynamic, and perturbation in the response due to the switching between the controllers.

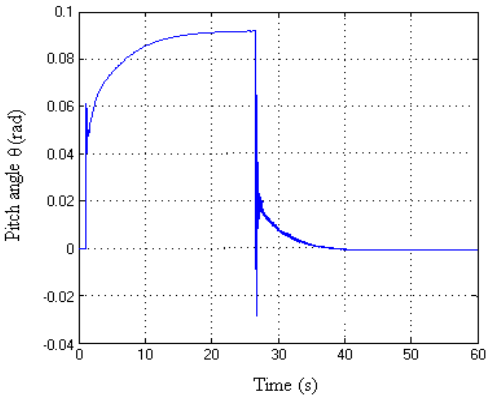

Figure 3. Response to pitch angle for continuous controller implemented in MATLAB/SIMULINK.

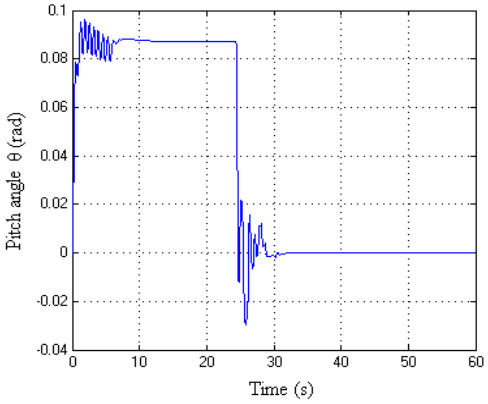

Figure 4. Response to pitch angle for digital controller implemented in the microprocessor in the proposed system.

Figure 5 and 6 shows the vertical speed response of the aircraft. The system presents the vertical speed of 500 feet/min in both graphics, corresponding to reference value defined in program. As before, when the altitude reaches the input value after 24s, the vertical speed will be equal to zero.

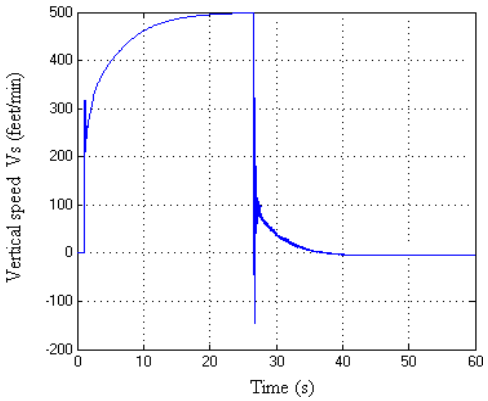

Figure 5. Response to vertical speed for continuous controller implemented in MATLAB/SIMULINK.

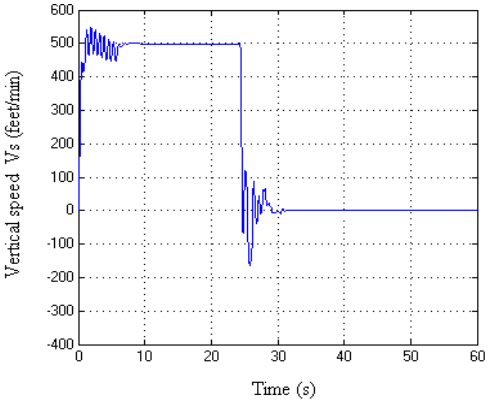

Figure 6. Response to vertical speed for digital controller implemented in the microprocessor in the proposed system.

Figure 7 and 8 shows the altitude response of the aircraft. The system presents the altitude of 200 feet in both graphics, corresponding to the reference value defined in program.

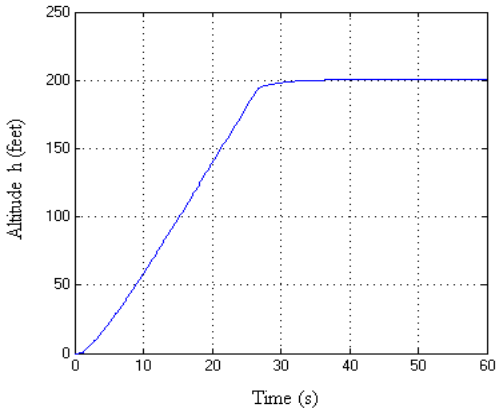

Figure 7. Response to altitude for continuos digital controller implemented in MATLAB/ SIMULINK.

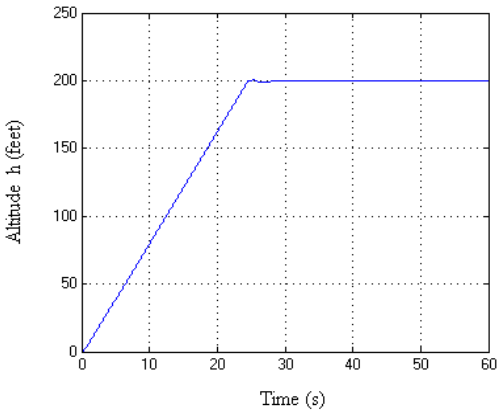

Figure 8. Response to altitude for controller implemented in the microprocessor in the proposed system.

The proposed system also presents more oscillatory response on the transition boundaries. Characteristic related to the discretization process.

Even though it is not presented here, the steady state error will never be equal zero. It happens because the discrete time process which takes place when the digital controllers are implemented in the microprocessor uses sampled values, obtained in periods of time pre-established by the communication between the UDP block implemented in MATLAB/SIMULINK and the Ethernet module of Rabbit 2000. However, it's important to notice that the problem happens as a consequence of the data sampling and it will be always present if a digital device is used to control the continuous system.

7. CONCLUSION

The development of this test platform for digital controllers was important for the familiarization with the problems which appears when the simulation approach is left and the implementation in a physical device takes place, even if it is a partial implementation as done in this case. The problem related to the communication between the controller and the aircraft should be solved by choosing an available communication interface in both modules and with a high data transmission rate in order to keep the controller actualized. In the presented system the UDP protocol was used to the communication between MATLAB (simulating the aircraft) and the Rabbit 2000. This protocol showed to be efficient because the possibility of transmit and receive a large quantity of bytes together, particularly being used in a 160 kbytes/s transmission rate. The problem related to the sampling process is inherent to discrete process. The problem was observed in the responses obtained working with the proposed platform, even thought it was not made efforts to minimize its effects.

Finally, the developed platform has shown to be a very useful intermediate stage in the complex problem of designing and implementing a controller in a real aircraft.

8. REFERENCES

Albanes, W., 2001, "Design of Guidance and Control Digital Autopilot", Computer Science corporation, Defense Systems Div, Journal of Guidance Control and dynamics, vol 4, page 126-133, Huntsville, Ala, USA.

Bender, J. G., Fenton, R. E., Olson, K. W., 2006, "An Experimental Study of Vehicle Automatic Longitudinal Control", IEEE Vehicular Technology Transactions, v20, p.114-123.

Blakelock, J. H., 1991, "Automatic Control of Aircraft and Missiles", John Wiley and Sons Inc, Second Edition, USA.

Choe, D.G., Kim J. H., 2002, "Pitch Autopilot Design Using Model-Following Adaptive Sliding Mode Control", AIAA Journal of Guidance, Control and Dynamics, v25, n4,p 826-829, Korea Advance Institute of Science and Techonology, Republic of Korea.

Ernst, D., Valavanis, K., Craighead, J., 2006, "Automated Process for Unmanned Aerial Systems Controller Implementation Using MATLAB", Control Automation 14th Mediterranean Conference, p 28-30.

Frutiger, M., Kim C., 2003, "Digital Autopilot Test Platform with Reference Design and Implementation of a 2-Axis Autopilot for Small Airplanes", Department of Electrical and Computer Engineering, University of Illinois at Urbana – Champaign, USA.

Hermerly, M.E., 1996, "Controle por Computador de Sistemas Dinâmicos", Ed Edgard Blucher, São Paulo, Brazil.

Huang, J., Lin, C.F.,Cloutier J.R., 2002, "Robust Feedback Linearization Approach to Autopilot Design", IEEE Control Applications Conference, v1, page 220 – 225, Dayton, OH, USA.

Manseur, R., 2006, "Implementation and Design of an Ummanned Aerial Vehicle", Spring, USA.

McLean, Donald., 1990, "Automatic Flight control Systems", Ed. Prentice Hall international, New York, USA.

Nelson, R.C., 1998, " Flight Stability and Automatic Control", McGraw-Hill, Second Editions, USA.

Ogata, K., 2003, "Engenharia de Controle Moderno", Ed. Prentice Hall do Brasil, Quarta Edição, São Paulo, Brazil.

Pallet, E.H.J., 1979, "Automatic Flight Control", Granada Publishing Limeted Technical Books, New York, USA.

Rabbit Semiconductor, 2006a "Rabbit 2000 TCP/IP Development Kit Getting Started Manual", Rabbit Inc, USA.

Rabbit Semiconductor, 2006b, "Rabbit 2000® Microprocessor Designer's Handbook", Rabbit Inc, USA.

Roskam, J., 1995,"Airplane Flight Dynamics and Automatic Flight Controls", Design Analysis and Research Coporation (DARcoporation), PART I and PART II, Lawrence, USA.

Sagahyroon, A., 2004, "Design and implementation of a low cost UAV controller", Industrial Technology, IEEE ICIT 2004 IEEE International Conference on Volume 3, Issue , 8-10 Dec. 2004 Page(s): 1394 - 1397 Vol. 3

Shamma, J. S., Cloutier, J. R., 1994, "Gain-Scheduled Missile Autopilot Design using Linear Parameter Varying Transformations", AIAA Journal of Guidance, Control and Dynamics, v16, n2, p 42-48, University of Texas at Austin, USA.

Steer, A.J., 2004, "Supersonic Transport Aircraft Longitudinal Flight Control Law Design", Aeronautical Journal ISSN 0001-9240 , v108, n1084, p 319-329, Department of Aerospace Sciences, Cranfield University, England.

Turkoglu, K., 2008, "PID Parameter Optimization of an UAV Longitudinal Flight Control System", Proceedings of World Academy of Science, Engineering and Technology, Volume 35, ISSN 2070-3740.