CHAPTER 8

HARDWARE IN THE LOOP SIMULATION

8.1 INTRODUCTION

Hardware-In-the-Loop is a form of real-time simulation. Hardware-In-the-Loop differs from pure real-time simulation by the addition of a “real” component in the loop. This component may be an Electronic Control Unit (ECU) for automotive, FADEC for Aerospace. The current industry definition of a Hardware-In-the-Loop system is shown in Figure 8.1. It shows that the controller is simulated and the ECU is real. The purpose of a Hardware-In-the-Loop system is to provide all of the electrical simulation needed to fully exercise the ECU.

![Diagram showing Hardware In The Loop definition: Dynamics is Simulated and Controller is Real](image)

In this chapter development of a platform for the implementation of the controller and verification of the results with MATLAB are dealt. It provides the experience to work in the real scenario with the microprocessor
and UDP communication. It helps to create a platform on which digital autopilots can be implemented and tested. The platform consists of a micro controller that hosts the autopilot code and interfaces with MATLAB. The aircraft landing control problem is designed in the MATLAB and their outputs are fed as inputs to the controller implemented in the hardware. The outputs from the hardware are again transmitted to the MATLAB and the results are verified. Thus the Hardware-in-the-Loop Simulation is developed.

8.2 BENEFIT OF HARDWARE IN THE LOOP SIMULATION

Benefit of Hardware-In-the-Loop is that testing can be done without damaging equipment or endangering lives. Aircraft autopilots are often used in university control system courses and textbooks as motivating examples. Unfortunately, most universities do not have laboratory settings suitable for research students to gain hands-on experience in the design, implementation and testing of such systems. While researchers could always design an autopilot in a simulation package such as MATLAB/SIMULINK, they generally lack the opportunity to see how their newly designed autopilot performs in real time environment. Flight-testing is generally used to test the newly designed autopilot system. Without testing it with a realistically simulated version such flight-tests often result in a crash and damage to the aircraft. In order to address the above issues, a platform for implementing and testing digital autopilots was designed and implemented.

8.3 SYSTEM OVERVIEW

The system consisting of three subsystems: The Autopilot Test Platform, the Hardware User Interface and the Autopilot. Figure 8.2 gives a high-level overview of the complete system.
Figure 8.2  System Overview

The platform consists of a microcontroller that hosts the autopilot code and interfaces with an advanced PC-based flight simulator. To facilitate enter settings into the autopilot, a hardware user interface is requires a keypad and LCD display. The design of autopilot and terminal flight phase for an unmanned air vehicle was described in previous chapters.

8.4  DESIGN DETAILS

8.4.1  Controller Design

The vertical speed to altitude controller transfer functions are given in the following form;

\[ D_p(s) = 1.5 \frac{s+3}{s+20}, \quad p > z \]  \hspace{1cm} (8.1)

\[ D_v(s) = 0.002 + 0.0008 \frac{1}{s} \]  \hspace{1cm} (8.2)

\[ D_a(s) = 0.3 - 0.80s + 0.01 \frac{1}{s} \]  \hspace{1cm} (8.3)

Vertical speed to altitude controller occurs 50 feet from the target altitude.
8.4.2 Digitizing the Transfer Function

The HILS software has to iterate faster than the system under test and the analog signals have to be digitized with a higher resolution than the system under test uses. So the transfer functions to be implemented in the hardware are converted from continuous to discrete form. The sample period was chosen as small as possible. The limiting factor was the time required for the Rabbit processor to receive and send a UDP packet. A sample period of 0.1 seconds was chosen (this is equivalent to a 10 Hz sample rate). This relatively low sample rate results in decreased damping of the discrete controller.

After making the substitution with $T = 0.1\text{sec}$ in pitch controller transfer function, and using MATLAB “c2d” command (which converts continuous-time models to discrete time models) the discrete controller transfer functions are obtained.

$$D_p(z) = \frac{0.8625z - 0.6375}{z}$$  \hspace{1cm} (8.4)

$$D_v(s) = 0.002 + 0.0004\frac{z + 1}{z - 1}$$  \hspace{1cm} (8.5)

$$D_a(s) = 0.3 + 16\frac{z - 1}{z + 1} + 0.005\frac{z + 1}{z - 1}$$  \hspace{1cm} (8.6)

The difference equations used to implement these controllers in software are easily obtained by inspection

$$e_{vvel}(n) = vvel_{\text{actual}}(n) - vvel_{\text{target}}(n)$$
$$vvel_p(n) = 0.002e_{vvel}(n)$$
$$vvel_i(n) = 0.0004\left[ e_{vvel}(n) + e_{vvel}(n - 1) + y_{vvel}(n - 1) \right]$$
$$y_{vvel}(n) = vvel_p(n) + vvel_i(n)$$  \hspace{1cm} (8.7)
\[
e_{alt}(n) = alt_{actual}(n) - alt_{target}(n)
\]
\[
alt_p(n) = 0.3e_{alt}(n)
\]
\[
alt_i(n) = 0.008\left[e_{alt}(n) + e_{alt}(n-1) + y_{alt}(n-1)\right]
\]
\[
alt_d(n) = 16\left[e_{alt}(n) - e_{alt}(n-1) - y_{alt}(n-1)\right]
\]
\[
y_{alt}(n) = alt_p(n) - alt_i(n) - alt_d(n)
\]
\[
e_{pitch}(n) = pitch_{actual}(n) - pitch_{target}(n)
\]
\[
y_{pitch}(n) = 0.575e_{pitch}(n) - 0.425e_{pitch}(n-1)
\]

8.4.3 User Datagram Protocol

User Datagram Protocol (UDP) is one of the core protocols of the Internet protocol suite. Using UDP, programs on networked computers can send short messages sometimes known as datagrams (using Datagram Sockets) to one another. UDP is sometimes called the Universal Datagram Protocol. UDP does not guarantee reliability or ordering in the way that TCP does. Datagrams may arrive out of order, appear duplicated, or go missing without notice. Time-sensitive applications often use UDP because dropped packets are preferable to delayed packets. UDP's stateless nature is also useful for servers that answer small queries from huge numbers of clients. Unlike TCP, UDP is compatible with forwarding the packet as broadcast and multicasting.

8.5 IMPLEMENTATION

The implementation of the discretized controller with the UDP communication program is done in the Dynamic C. The developed program is compiled and burned in the flash memory of the RCM 2200. In the
MATLAB, the aircraft dynamics and the sensors are designed and the outputs are converted to single value and all the input values are packed and they are sending through the UTP data cable to the hardware.

In the hardware the glide slope controller, blending function and the flare path controller algorithms are implemented. The packed output from the MATLAB is fed as input to the controller in the hardware and the values are calculated. The calculated data is again fed back to the MATLAB and the results are compared with the simulated results.

![Diagram of Hardware In The Loop Algorithm](image)

**Figure 8.3 Hardware In The Loop Algorithm**

Figure 8.3 describes the Hardware in the Loop Simulation. This provides the basic platform for testing various controllers and gives the idea in the Hardware Implementation. The working algorithm for the Hardware in the Loop Simulation is given as

**Step 1.** To send the data from MATLAB / SIMULINK Aircraft Model to the RCM 2200 through the UDP channel

**Step 2.** To receive the data from the aircraft model and store it in the buffer
Step 3. Manipulation of control law in the RCM 2200

Step 4. To send the data again to the aircraft model through UDP channel

Step 5. Aircraft model receives the required output and the landing simulation displayed in the X-plane simulator

Figures 8.4-8.6 show the experimental set up of the Hardware in the loop simulation setup.

**Figure 8.4 Aircraft Aligning to the Airport LLA**
Figure 8.5  Aircraft towards Runway Touch down Point

Figure 8.6  Rabbit Core Processor
8.6 COMPARISON OF RESULTS

The autopilot is tested by comparing the response from the controller by digital autopilot to that obtained from the simulation. While the degree to which the landing profile in response to the autopilot meets the specifications, other variables are compared to those from the simulation. These include vertical velocity and pitch parameters.

8.6.1 Response of a Linear Model

![Figure 8.7 Comparison of Pitch Angle Response](image1)

![Figure 8.8 Comparison of Velocity Response](image2)
From Figures 8.7-8.10 shows that the variations in the values are due to the discretization of the transfer function and the sampling rate. But the differences in characteristics are tolerable.
8.6.2 Response of a Non Linear Model

Figure 8.11 Comparison of Pitch Angle Response

Figure 8.12 Comparison of Velocity Response
Figure 8.13 Comparison of Angle of Attack Response

Figure 8.14 Comparison of Height Response
Figure 8.15 Comparison of Range Response

From Figures 8.11-8.15 show that the response of the Euler angles of hardware and MATLAB are exactly same. But the values of the velocity, theta, alpha and phi are having small differences, due to the discretization of transfer function and sampling time. Due to the sampling rate in the non linear model the variation in the values are created in the feed back.

8.7 CONCLUSION

In this section, the blending function control law designed in the MATLAB was implemented in the Hardware. The RCM2200 was chosen for its high compactability, high speed and low cost. The developed controller was implemented in the RCM 2200 processor. The codes were developed in the dynamic C software and burnt in the flash memory of the Rabbit Processor and tested using MATLAB simulation. The inputs from the MATLAB simulation were given to the RCM 2200 and computed data is again fed back to MATLAB simulation. The results from hardware were compared with MATLAB simulator results. Thus Hardware In the Loop
Simulation set up was developed. This provides the basic platform for testing various controllers and gives the idea in the Hardware Implementation.

There are still a couple of challenges and uncertainties remaining. It is necessary to increase the sample rate of the system. While it can be shown using simulations that merely doubling the sample rate results in greatly improved performance for the UAV flight control system, a much higher sampling rate needs to be supported for the platform to be of general use. The task of using Rabbit 3000 Microcontroller for future work is unknown at this point of time. Another option would be to replace the UDP link with a high-speed serial connection. This would eliminate the computational overhead for UDP communications on the Rabbit and may give higher sampling rates. The exact tradeoffs between the two solutions need to be further explored.