INDIAN INSTITUTE OF SCIENCE, BANGALORE

Summer Project Report

May-June 2010

Debarshi Patanjali Ghoshal

3rd Year (Undergraduate),
Electrical Engineering Department,
Jadavpur University, Kolkata.

25/6/2010

Report on summer project done during May-June (2010) under the guidance of Prof. M. Seetharama Bhat at Department of Aerospace Engineering, Indian Institute of Science, Bangalore.

Summary

I worked as a summer trainee for one month at the Department of Aerospace Engineering, Indian Institute of Science, Bangalore under the guidance of Prof M. Seetharama Bhat.

During this period (May-June 2010), I worked on waypoint navigation system of Micro Air Vehicles (MAV). For evaluation, a Matlab Simulink model was built consisting of navigation, guidance, and control loops about a dynamic model of a MAV. It yielded fairly good result.

I also set up the Hardware-in-the-Loop simulation system of Kestrel Autopilot System jointly with my senior co-workers Mr. Titas Bera and Mr. Chiranjit Shee. The simulation was implemented successfully.

Acknowledgements

I would like to thank Prof. M Seetharama Bhat for giving me this great opportunity to work under him as a summer trainee.

I would also like to thank my senior co-workers Mr. Titas Bera and Mr. Chiranjit Shee who have helped me in my project.

Waypoint Navigation system for Unmanned Aerial Vehicles (UAV)

Unmanned Aerial Vehicles have found diverse applications for both civil and military missions. To achieve the stated mission, the vehicle needs to have a certain level of autonomy to maintain its stability following a desired path under embedded guidance, navigation and control algorithm. To meet the increasingly more stringent operation requirements, the UAVs rely less and less on the skill of the ground pilot and progressively more on the autonomous capabilities.

This report describes a waypoint navigation system for UAVs.

State-variable model of Micro Air Vehicle (MAV) Sapthami-flyer was used for the modeling.[1] It is given below.

$$\begin{bmatrix} \dot{\beta} \\ \dot{\varphi} \\ \dot{p} \\ \dot{r} \\ \dot{\delta}_a \end{bmatrix} = \begin{bmatrix} -0.68 & 0.76 & 0.0076 & -0.99 & -0.12 \\ 0 & 0 & 1 & 0 & 0 \\ -902.59 & 0 & -11.025 & 10.98 & 1350.6 \\ 316.07 & 0 & 0.92 & -4.68 & -201.6 \\ 0 & 0 & 0 & 0 & -22 \end{bmatrix} \begin{bmatrix} \beta \\ \varphi \\ p \\ r \\ \delta_a \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 22 \end{bmatrix} u_a$$

$$\begin{bmatrix} \varphi \\ p \\ r \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \beta \\ \varphi \\ p \\ r \\ \delta_a \end{bmatrix}$$

It was assumed that the airspeed (speed with respect to air) of the MAV is constant, having a value of 12.7 m/s. It should be noted that this assumption is rather idealistic, as in actual flights it is very difficult to maintain constant speed during turning.

The motion of the MAV was constrained to constant altitude.

A bank angle controller was already designed by Mr. Chiranjit Shee.

I designed a heading-to-bank-angle controller. It consisted of a proportional feedback loop controller.

This, along with the previously developed controller formed the basis of the waypoint navigation system.

The response of this system to unit step heading command is shown in Fig. 1

The rise time of the response is found to be a little over 5 sec.

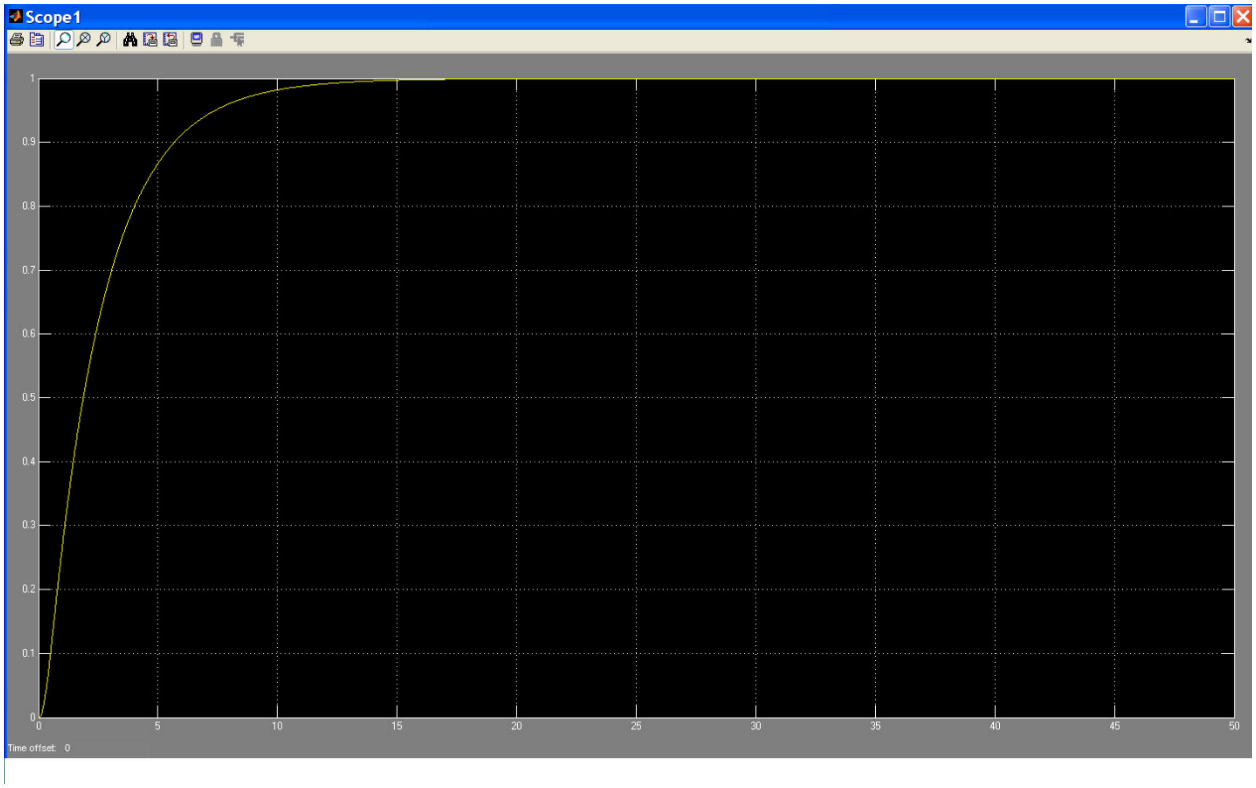


FIGURE 1

A database of waypoints is to be provided in x-y coordinate system (representing East-North direction).

In the beginning, the first waypoint in the database is considered. Suitable algorithm was designed to calculate the desired heading for the MAV by taking into account the current co-ordinate of the vehicle. Required heading command is fed to the controller system accordingly.

When the MAV is within a certain distance (proximity distance) of the waypoint in question, the algorithm considers that waypoint as reached and proceeds to the next waypoint in the database. The value of the proximity distance can be changed.

Setting a very low proximity distance ensures that the MAV flies over the exact waypoint co-ordinates but there is a significant amount of overshoot before the MAV turns towards the next waypoint. Increasing the proximity distance will ensure a smooth trajectory but the MAV may not fly over the waypoint precisely.

Fig. 2, 3 & 4 show three cases where the proximity distance is set as 1m, 10m & 30m respectively. In all the cases 4 waypoints were defined having co-ordinates (300, 0) , (300, 300) , (0, 300) & (0, 0).

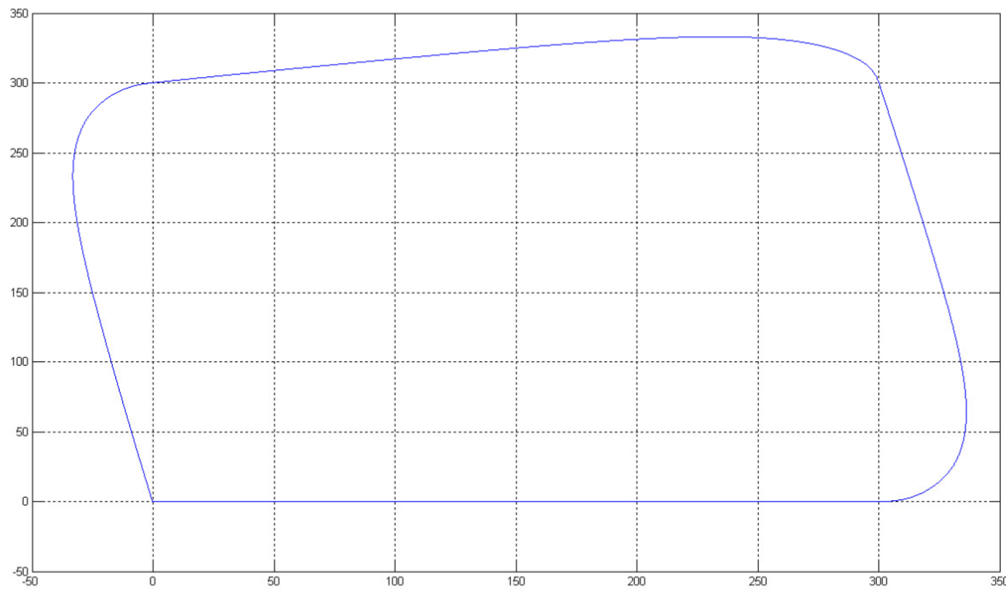


FIGURE 2

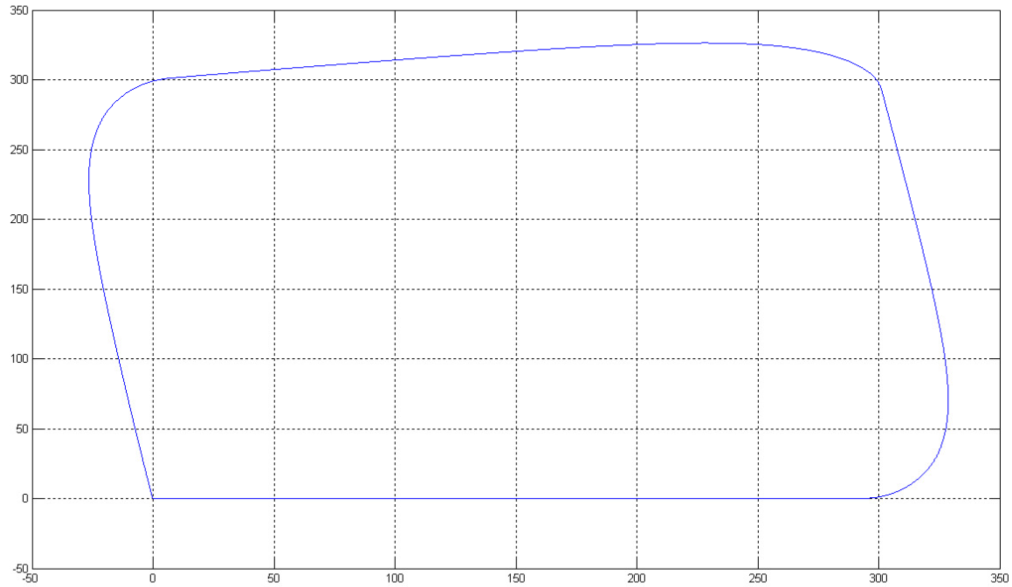


FIGURE 3

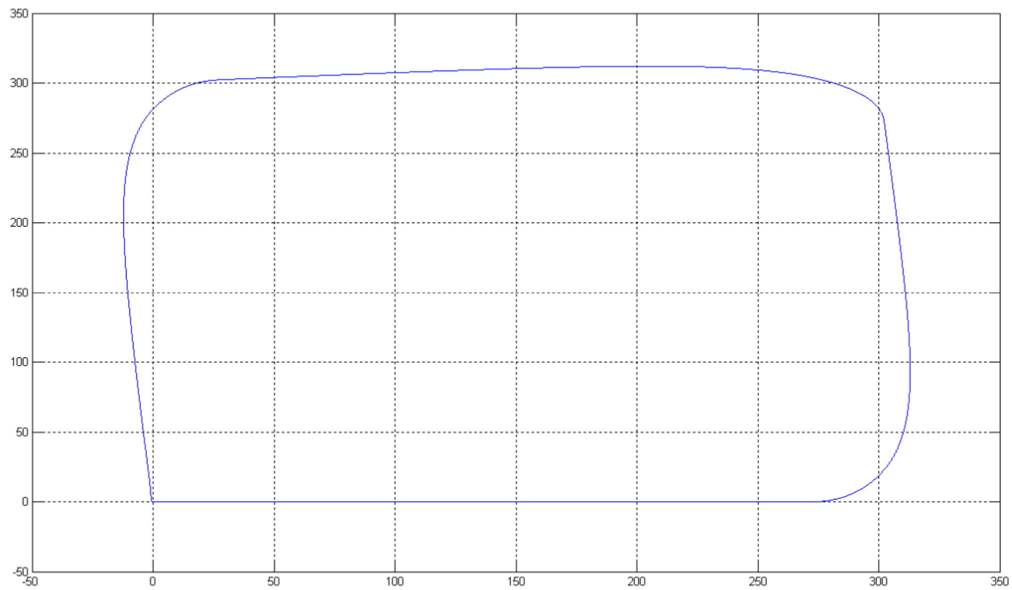


FIGURE 4

Effect of wind

As the MAVs are very lightweight vehicles, the effect of wind must be considered when evaluating the performance of the navigation system

So the system was also tested by adding wind to the simulation.

Fig. 5, 6 & 7 show the flight trajectories when the wind-speed is zero, 3m/s & 5m/s respectively. The wind direction is towards north i.e. towards positive y axis.

The waypoints are same as previously used. The proximity distance is set as 10m.

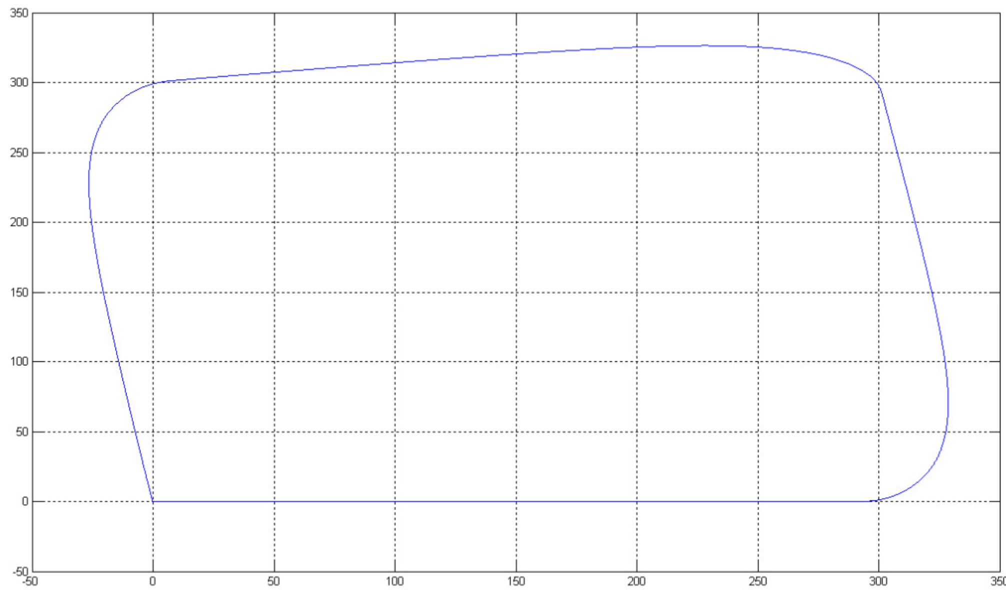


FIGURE 5

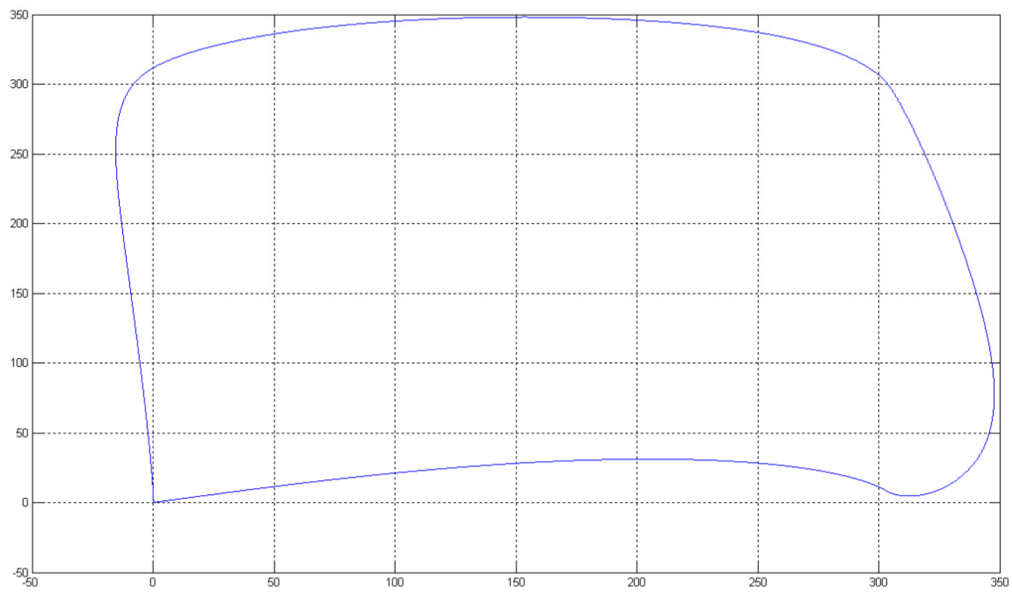


FIGURE 6

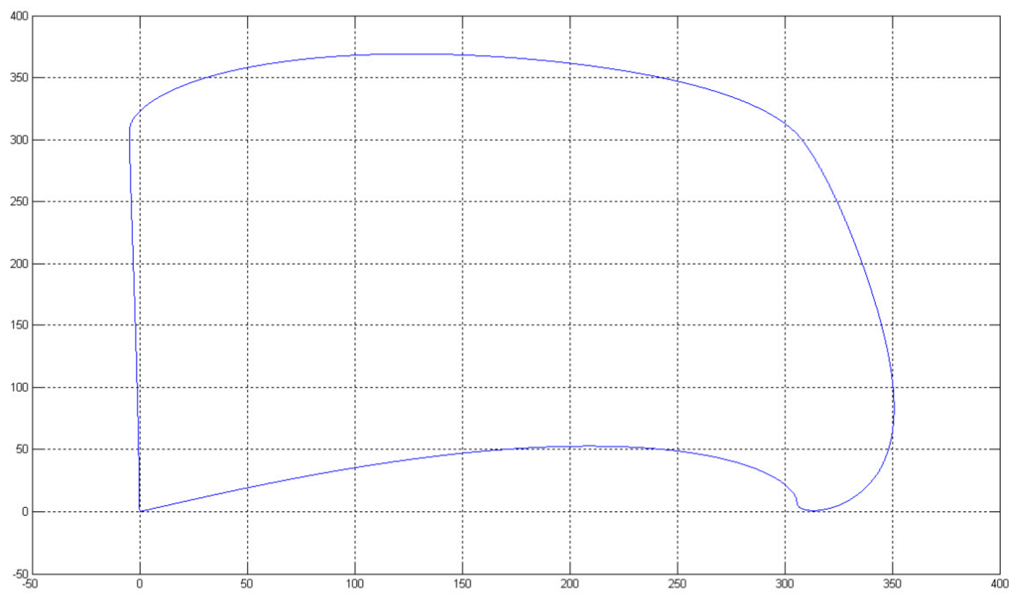


FIGURE 7

Kestrel Autopilot System

The Kestrel Autopilot (KAP) System provides intelligent, autonomous flight control, GPS waypoint navigation as well as autonomous takeoff, flight, and auto-landing routines of micro / mini unmanned aerial vehicles (UAV). Along with the ground station hardware and software control system, it offers a highly effective avionics package for a variety of military and commercial applications.

An external GPS unit is used for inertial navigation and wireless modems for ground station to autopilot communication. The autopilot can guide mini UAVs autonomously and/or receive dynamic user commands through the ground station or RC radio while providing live video feeds to the user. Its sensors are 3-axis rate gyros and accelerometers for attitude estimation, differential and absolute air pressure sensors for airspeed and altitude measurement. It is a small, low-cost, low-power solution that provides real-time trajectory generation and tracking and simple intuitive user interfaces.

The Virtual Cockpit ground control software makes “click ’n fly” operation easy while providing powerful mission planning, monitoring, and in-flight adjustment on a notebook computer, allowing the user to interact with the UAV in a variety of different modes that include stick & rudder, altitude-heading-velocity commands, and dynamic waypoint specification.

Hardware-In-The-Loop Simulator

The Kestrel Autopilot and the Virtual Cockpit have a built-in ability to simulate a 6 degree of freedom UAV through the use of a 3rd party, open source simulator called *Aviones*. Aviones displays the simulated flight in 3D allowing the user to quickly and easily test flight plans and new software development prior to full outdoor flight.

Aviones uses the idea of loadable code libraries for simulation. Aviones loads two libraries:

physics and autopilot.

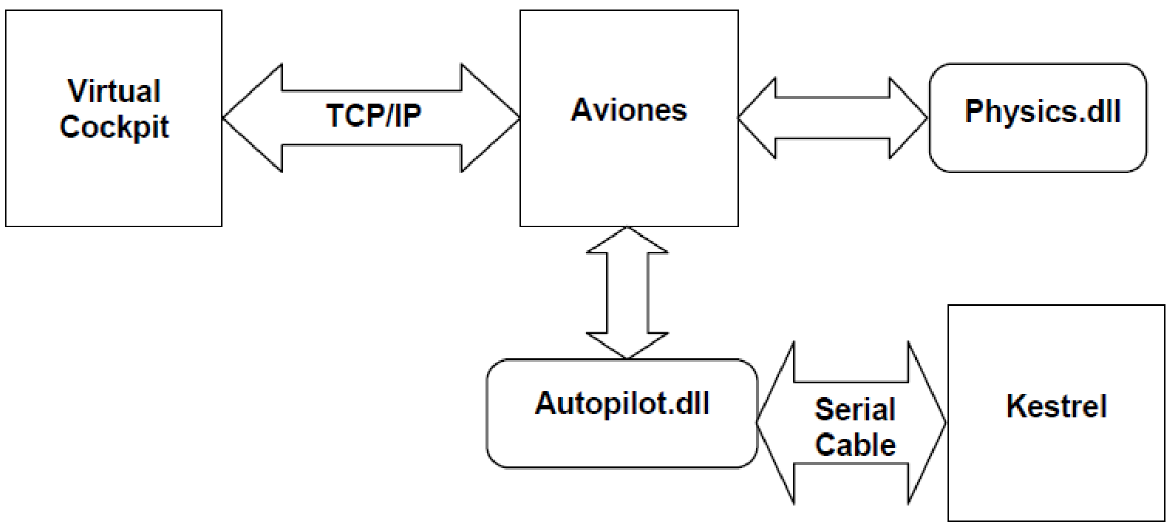
- The physics library computes how the airplane responds in its natural environment using user-definable airplane coefficients, wind, and 6 degree of freedom models.

- The autopilot library describes the autopilot and how it will respond to its orientation and position in the simulated world.

By using Procerus' proprietary autopilot DLL, Aviones can communicate with the Kestrel Autopilot and replace the autopilot's sensor information with that generated by the physics library. After the Kestrel receives the sensor information it can compute a controller output which is fed back into the physics library.

The Hardware-in-the-Loop (HIL) simulation can be set up two ways.

In our chosen method, the Virtual Cockpit communicates with the Aviones over a TCP/IP connection. Aviones then passes the data from the Virtual Cockpit to the autopilot over a wire serial connection. The serial cable is connected to the autopilot's modem port. The advantage of this method is that only one serial port is needed.



Hardware-in-the-Loop setup using TCP/IP connection

Setting up HIL system using TCP/IP connection

In this configuration, the autopilot communicates with Aviones through the Virtual Cockpit. Because high speed state information is passed between the autopilot and Aviones, the wireless link that is normally used to communicate between the Virtual Cockpit and the autopilot cannot be used. The modem must be unplugged and replaced with the serial programming cable provided by Procerus. The communications will pass between Aviones and the autopilot over this cable. The communications that normally go through the wireless link between the Virtual Cockpit and the autopilot will also go through this cable (after going through a TCP/IP link between Aviones and the Virtual Cockpit).

The following steps are used to setup the HIL simulation using the TCP/IP interface:

1. Remove the modem from the autopilot (either from the back of the autopilot or from the modem plug on the front).

2. The next step requires the use of the autopilot programming cable with the 5 wire programming connector. This is provided by Procerus. Plug the 5 pin header on the programming cable into the autopilot modem port.

3. Plug the other end of the programming cable into a free serial port on the computer that will be running the Virtual Cockpit. Make a note of the serial port number – it will be needed later.

4. Power on the autopilot

5. Load Aviones

6. Load the Virtual Cockpit

7. Open the Edit Agents window (This can be found in the Agent Menu -> Edit Agents).

8. The right-most column is labeled HIL for "Hardware-in-the-Loop." Check this box next to the agent number of the autopilot.

9. Click OK

10. A dialog box will pop up where the IP address of the computer running Aviones may be specified. If Aviones is running on the same computer running the Virtual Cockpit, click the Loopback button and press OK to connect to Aviones.

Virtual Cockpit will now attempt to connect to Aviones and add all the agents that were selected and had the HIL checkbox checked.

Note: If there is a communications port opened on the Virtual Cockpit, the Virtual Cockpit will automatically close this in the event that Aviones needs to open the same comm port.

If connected successfully, the Virtual Cockpit will indicate on its status bar and Aviones will have added the Agent to its list on the bottom of the window.

11. The next step is to make sure that Aviones is setup to talk to the autopilot through the correct comm port. Open up the HIL Simulation Control Dialog through the View Menu->HIL Sim Ctrl. When the dialog is opened it will fill in the current autopilot information that was originally contained in the autopilot1032_regs.txt file.

a. Place the cursor in the Serial Port # box.

b. Delete the value that is currently in the box using the backspace key (The box should turn red)

c. Enter the serial port number connected to the autopilot (From step three).

d. Press Enter – The box should turn white and the word "Open" should be displayed in the box to the right (see Figure 8.8).

12. With the correct serial port open, data should begin to flow between Aviones, the Virtual Cockpit, and the Autopilot. The Virtual Cockpit should show communications with the autopilot.

13. Start the simulation by placing the UAV in Take Off Mode using the Mode buttons.

14. Launch the UAV in Aviones: UAV pull down menu -> Launch if necessary.

Thus Hardware-in-the-loop simuation was successfully setup and executed.

References

[1]M Seetharama Bhat and Neha Satak. Development of Micro Air Vehicle – Zephyr and Sapthami.

Procerus Technologies. Kestrel User Guide. *3/20/07. Document Version 1.5*