Hardware Implementation of a Flight Control System for an Unmanned Aerial Vehicle

Student: Pankaj Maurya (Y2258)
B.Tech Project Supervised by: Dr. Rajat Moona
Computer Science and Engineering
IIT Kanpur

Abstract

The project aims to build up a computational platform for an autonomous unmanned aerial vehicle and implement the computational platform using off-the-shelf components for the hardware. We have chosen a miniature airplane for the project. The project has reached a stage where we have fabricated the airplane, designed control laws for it and implemented the major individual components separately, namely the control system, the actuation system and the data acquisition and filtering system.

The project was a joint effort of the departments of AE, CSE and EE. The work done for a CSE project has been highlighted in this report. The work involved implementation of control laws on a single board computer, the actuation system for deflecting the control surfaces using a servo controller card and servos and the communication protocol between the single board computer and the data acquisition and filtering system.

Table of Contents

1. Introduction

Research and development in the field of autonomous vehicles has always been attractive, for a variety of actual and potential applications in several different fields. Unmanned aerial, underwater and ground vehicles have been used in Earth and Space exploration, military reconnaissance and intelligence gathering, provision of data and services for commercial and scientific applications. Unmanned vehicles can perform critical tasks without endangering the life of human pilots. They can be designed and developed without having to account for the presence of a pilot and the associated life-support and safety systems, potentially resulting in cost and size savings and increased operational capabilities.

In this project we attempt to actually implement autonomous mode of operation on a self-fabricated miniature airplane. This is largely an inter-disciplinary project involving components from Electrical Engineering, Aeronautical Engineering, and Computer Science and Engineering.

2. Problem Statement

In this project we have to do a Hardware Implementation of a Flight Control System for an Unmanned Aerial Vehicle. The Hardware Implementation involves the On-Board Computer Setup, implementation of control laws, Sensing, Data Acquisition and Filtering and Actuation.

3. Project Objectives

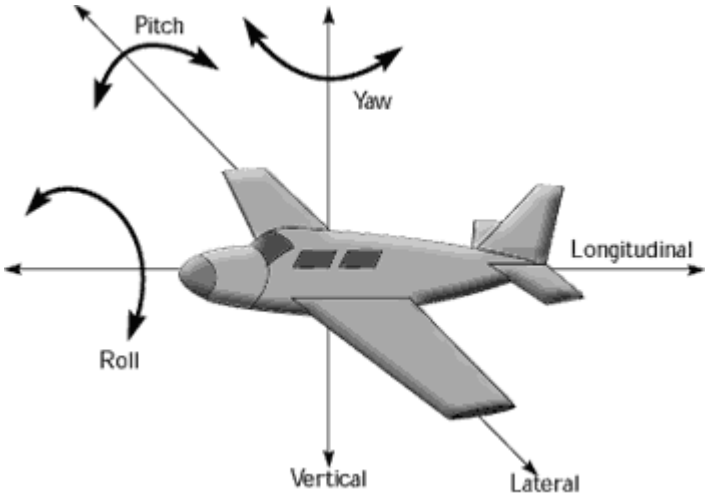
At the top level our project objectives can be stated as –

- Fabrication of airplane
- Design of control system
    - Simulations for testing stability and controllability.
- Hardware Implementation
    - On board Computer setup
    - Interfacing On board computer with actuators.
    - Data Acquisition from sensors, filtering of sensor data.
    - Communication system between On board computer and data acquisition and filtering system.
    - Implementation of control laws.
- Testing
    - Unit testing of control law implementation, filtering system
    - Hardware In Loop Simulation (HILS)

Amongst these, the topics related to CSE are described in detail in the next few sections.

4. Hardware Details

Here are some terms which are necessary to follow up this report:



4.1 Model Details

Our model is shown in Figure 1. The controllable parameters available for this miniature airplane are:

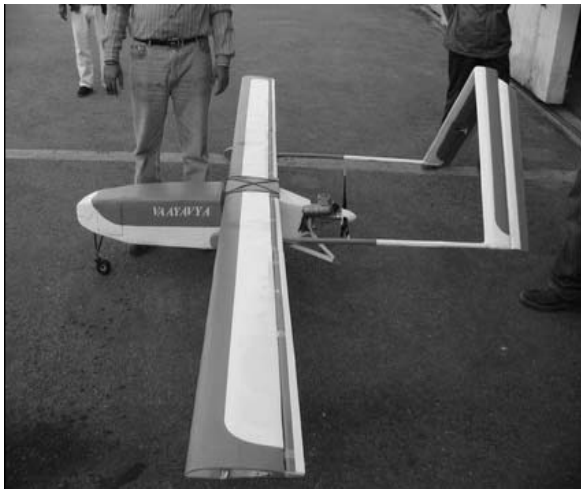


Figure 1

1. Engine throttle: more engine power can roughly translate into more airspeed.
2. Servo deflections in control surfaces – aileron, flaps, elevator, rudder.

4.2 Servo Motor

Servo motors are distinct from the general DC motors in the sense that they are position motors. One can precisely control the angular position (out of 180 degrees) of these motors using Pulse Width Position. Another key feature is that they are high torque motors, which is essential required to change the airplane's surface orientation against the drag. We are using Futaba S3003 Servo motors.

4.3 Servo Card

In order to communicate with the Servos, we need a hardware controller. We have used PicoPic from PicoBotics for this purpose. It is a versatile, light weight, robust and low on power, high performance Servo Controller Card. It can simultaneously control 20 servos and can be programmed over a serial port. It follows a very clean 5 character packet protocol for serial communication.

4.4 Sensors

We are using rate gyros, magnetometers, pressure sensors for data acquisition through an Analog to Digital Converter (ADC) and followed by filtering done using Kalman filtering implemented on an FPGA. Currently we have not managed to include these sensors in the simulation loop

4.5 Single Board Computer (SBC)

We have done our development on Soekris net4521. It is based on a 133 Mhz 486 class processor. It uses a Compact Flash module for program and data storage.

The Soekris net4521 is not really a general purpose SBC but more of a communication computer. Recently we have switched over to LIOD Evaluation Platform. The LIOD Evaluation Platform is a software development kit based on Intel PXA 270 Processor. The kit has peripheral and expansion capability for application software development.

We are running Linux on both these computers. I would like to emphasize here that it took us a lot of effort to configure and setup Linux on these computers. We had done an installation of Pebble on the flash card (256 MB) since it takes about 50MBs after installation and has all the packages we required at this initial stage. We had also done a smaller installation (about 10 MB) on the Soekris from scratch earlier but found Pebble to be easier to maintain and upgrade.

5. Problem Solving

Here we describe the approach we have followed. First task is to get an accurate estimate of the current orientation and position in space. This is followed by the execution of the control logic. The control logic determines the actuations based on factors such as desired parameters given to the autopilot, current state of the airplane and stability of the airplane. This is followed by sending the actuation commands to the servo controller card which takes care of moving the servos to the desired angles.

5.1 Obtaining accurate attitude of airplane

Our data acquisition and filtering is done on an FPGA. The sensor data is digitized using an Analog to Digital converter card and then fed to the FPGA. The FPGA runs Kalman filtering on the received data. It provides us with the filtered values of the physical quantities – Velocity, Angle, angular velocity and the navigation quantities in the 3 axis. We have designed and implemented an efficient communication protocol to obtain these values every cycle of control execution. Please refer Appendix A1 for a detailed functional specification.

5.2 Control logic execution

The control logic is the central component of the entire system. The control laws have been developed and tested (on MATLAB using Simulink) by our AE team. The control system is composed of a longitudinal control system (for controlling the pitch rate and airspeed) and a Lateral control system (for controlling the roll rate and yaw rate) along with a bank angle control system.

As shown in figure 2, the longitudinal autopilot takes in as inputs the commanded altitude, h^c and commanded velocity, V^c. The outputs are the elevator defection, δ_e, and the throttle command, δ_t. The Altitude Hold autopilot converts altitude error into a commanded pitch angle θ^c. Then, the pitch Attitude Hold block converts pitch attitude error into a commanded pitch rate q^c, which in turn is converted to the elevator command based on the error. The Velocity Hold autopilot converts velocity error to throttle command.

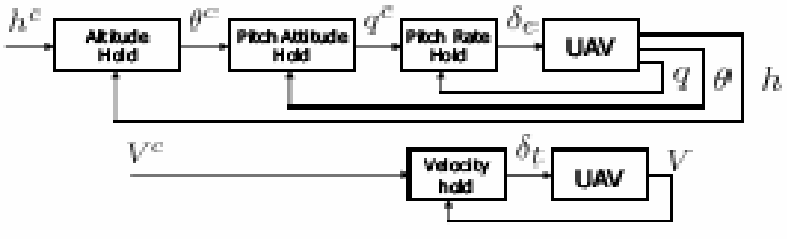


Figure 2

The lateral autopilot is shown in Figure 3. The input command to the lateral autopilot is the commanded heading, ψ^c. The output is the aileron command δ_a. The Heading Hold autopilot converts heading error to roll attitude command, φ^c. The Roll Attitude Hold autopilot converts roll angle error to roll rate command, p^c. The Roll Rate Hold autopilot converts the roll rate error to aileron command, δ_a.

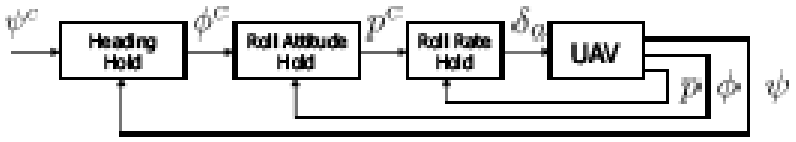


Figure 3

The bank angle control system is a simple PI (P – Proportional, I – Integral) controller which takes in the current bank angle and outputs a correction in the roll rate. Basically it tries to make the airplane fly at the desired bank angle (which is usually zero).

The control system which been designed and tested on MATLAB was a continuous time domain control system. See Appendix A2. The control system has been converted from continuous to discrete domain for implementation on a computer. After conversion, the control system essentially consists of equations in state space form:

$$X_{k+1} = A\,X_k + B\,U_k$$
$$Y_k = C\,X_k + D\,U_k$$

Where X_k is the state Matrix of the control system, U_k is the input matrix consisting of attitude of the airplane and Y_k is the output matrix containing the actuation values (angles of control surfaces).

Here A, B, C and D are matrices which are computed from a set of matrices KA, KB, KC and KD (different values for each airspeed range) based on the airspeed of the airplane. Basically the control system has been designed for a range of airspeeds (11 – 25 m/s) and simulations have been carried out for each airspeed in MATLAB giving us the corresponding KA, KB, KC and KD matrices for both the lateral and longitudinal control system. The computation of A, B, C and D from KA, KB, KC and KD also involves the parameter T, which is the cycle time of the control law's execution on the on board computer (T = 20 ms in our case).

Our control logic algorithm can thus be written as a simple sequence of the following steps:

1. Obtain the latest values of attitude and update the input matrices U_k for both the lateral and longitudinal control system.
2. Based on the current airspeed, calculate the matrices A, B, C and D for both the control systems.
3. Calculate the output for the aileron, flaps, rudder, throttle and elevator for this cycle.
4. Calculate the correction in the roll rate from the bank angle stability control system.
5. Calculate the next state (k+1) of the lateral and longitudinal control system.

6. Repeat Steps 1 to 5.

5.3 Actuation

Based on the actuation values generated from the control system we have to now set the control surfaces at the desired angles. This is done through the servo controller card (SCC). The SCC is can be interfaced to a serial port from which it can receive actuation commands and control multiple motors in parallel. We have calibrated the servos and obtained the ratio of angle deflection to PWM input. The setup had an accuracy of half-a-degree.
The readings were taken for four different speed values:

- speed 10 - a typical operational speed (Table II)
- speed 100 - a medium to fast operational speed (Table III)
- speed 1 - minimum possible operational speed (Table V).
- speed 0 - maximum operational speed supported by servo(Table IV).

The results are given below:

pwm in micorseconds	*angle in degrees*
800	0.0
1000	20.0
1200	39.5
1400	59.5
1600	78.5
1800	97.5
2000	117.0
2200	136.5
2400	157.0
2500	167.0

TABLE II
CALIBERATION READINGS OF AN S3003 SERVO MOTOR FOR SPEED = 10

pwm in micorseconds	*angle in degrees*
800	0.0
1000	20.0
1200	40.0
1400	60.0
1600	79.0
1800	97.5
2000	117.0
2200	136.5
2400	157.0
2500	167.0

TABLE III
CALIBERATION READINGS OF AN S3003 SERVO MOTOR FOR SPEED = 100

pwm in micorseconds	*angle in degrees*
800	0.0
1000	20.0
1200	40.5
1400	60.0
1600	79.0
1800	98.0
2000	117.0
2200	137.0
2400	157.0
2500	167.0

TABLE IV
CALIBERATION READINGS OF AN S3003 SERVO MOTOR FOR SPEED = 0

pwm in micorseconds	*angle in degrees*
800	0.0
1000	20.0
1200	40.0
1400	60.0
1600	79.0
1800	97.5
2000	117.0
2200	136.0
2400	156.5
2500	167.0

TABLE V
CALIBERATION READINGS OF AN S3003 SERVO MOTOR FOR SPEED = 1

The on-board computer creates a packet with the servo number and pwm value based on the control logic's output. This is sent to the SCC according to the communication protocol as described in A1.

6. Conclusion

We have done the hardware implementation of the individual components necessary for achieving the autonomous mode of operation in our airplane – namely the actuation system, the control laws implementation on a computer and the data acquisition and filtering. However we have still not done a Hardware In Loop Simulation of the entire system to test out the implementation in real time.

7. Future Work

We plan to test out our implementation using HILS to give us more confidence in the implementation before actually carrying out test flights. Our team of EE is already working on a vision based control system since 2 semesters which would add more control to the control system. Finally we will be putting in the avionics along with the control laws we have developed on the UAV model being developed by AE The first stage will be simple data acquisition in which we will try to plot the path traveled based on the data. The second stage will be to have autonomous control using one of the two – lateral and longitudinal control system and radio control the other. Finally we will try to have full autonomous control using the two control systems operating simultaneously.

I would like to mention that most of the components we have used are off the shelf components. We have had pretty bad experience working with on board computers with faulty serial ports which had taken us weeks to diagnose. We have had some minor and major crashes also in the last one year. Such an implementation has required us to work in domains of not only computer science but also understand concepts of aerospace engineering and electrical engineering.

8. Acknowledgement

We would like to express our sincere thanks to Prof. Rajat Moona, CSE who has supported us throughout the project; giving us ideas and encouragement. He has helped us in arranging for the much needed funds for the project. Every time we got stuck due to the lack of support from our working platform, Dr. Moona has helped us find a way out. We would also like to thank Media Lab Asia for providing us the equipments and technical support. Personally I would also like to thank all my team members who have supported me though out the project.

9. Bibliography

1. Aircraft Control and Simulation by Stevens, Lewis; Georgia Institute of Technology. (Chapters 4-7)
2. Autonomous Vehicle Technologies for small fixed wing UAVs by Derek Kingston, Randal Beard, Timothy McLain, Michael Larsen and Wei Ren, Brigham Young University.
3. BTP Report on ”A computational platform for an Autonomous Flying Robot” by Kumar Gaurav, Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur
4. Embedded Linux: Hardware, Software and Interfacing by Craig Hollabaugh (Chapters 1-4, 6)
5. Building A Platform For UAV Research by Paolo Messina, University of Pisa.
6. Development of a INS/GPS navigation loop for an UAV by Sven Ronnback, Lulea University of Technology.
7. Web Reference: Serial Port HOWTO http://www.tldp.org/HOWTO/Serial-HOWTO.html
8. Web Reference: Picobotics main website http://www.picobotics.com/PicoPicManual.html
9. Liod Evaluation Platform Manuals (www.51board.com)
10. Web Reference: Pebble Linux Distribution http://www.nycwireless.net/tiki-index.php?page=PebbleLinux

Appendix

A1 Functional Specification of the Communication Protocol

This functional specification is written for defining the communication protocol between the components: Single Board Computer (SBC), FPGA-B(connected to ADC-Analog To Digital Converter) and FPGA-A(connected to SCC - Servo Controller Card).

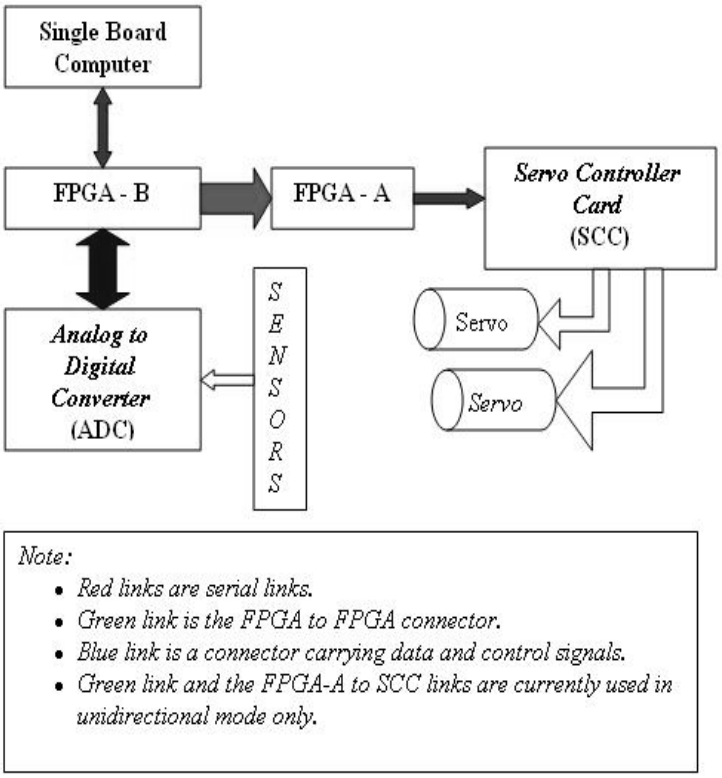


Fig 1(A) shows the block diagram of the system.

Version Change history

- Version 1 : Initial Protocol which had header, byte count, data, chksum->unnecessary complications and processing.
- Version 2 : Checksum, header removed; packet type and byte count merged in one byte; IP request and reply packets handling improved.
- Version 3: Bits dedicated to channel no, axis; no need of data bits size specification from SBC (earlier bit 5).
- Version 4: Calibrate command added (command is 0x40 = 01000000, 01 at 7:6 imply that its a packet from SBC to FPGA-B)

Description

The communication between the components is in form of packets. A packet is simply a stream of characters. There are 3 kinds of packets - Information Packets (IP: FPGA-B to SBC), Actuation packets (SBC to FPGA-A), Command/Request packets (SBC to FPGA-B). The packet structure is described below:

Protocol summary

Byte 1	**Src, Dest, length, request information encoded**
Bytes 2-n	**Data or command bytes need not be always present: many packets only contain 1 Byte**

Byte 1

Bit Number								Description
7	6	5	4	3	2	1	0	
0	X	X	X	X	X	X	X	A packet from SBC
0	0	X	X	X	X	X	X	Other End is FPGA-A: hence Bits 7:6 = 00 => Actuation packet
0	0	0	0	0	1	0	1	Bits 5:0 have the usual meaning of length For Actuation packet size is 5 bytes(details of following bytes in next table)
0	1	X	X	X	X	X	X	Other End is FPGA-B: Look at Bit 5
0	1	0	0	0	0	0	0	Calibrate command
0	1	0	X	X	X	X	X	Reserved for Future, Bits 5:0 with bit 5 hard coded to 0 => max 32 byte packet from SBC to FPGA-B Bytes 2-n in this case have not been decided presently.
0	1	1	X	X	X	X	X	IP(Information Packet) Request packet: no further bytes in packet Bits 4:2 encode the channel number and Bits 1:0 encode axis.
0	1	1	0	0	0	X	X	Channel number 0 requested: Channels correspond to the physical quantities: Velocity, angle, rate and Navigation (See Table 4)
0	1	1	0	0	0	0	0	Channel 0 ,Axis 1 component requested
0	1	1	0	0	0	0	1	Channel 0 ,Axis 2 component requested
0	1	1	0	0	0	1	0	Channel 0 ,Axis 3 component requested
0	1	1	0	0	0	1	1	Channel 0, All Axis components requested
0	1	1	0	0	1	X	X	Channel number 1 requested: Channels correspond to the physical

								quantities: Velocity, angle, rate and Navigation (See Table 4)
0	1	1	0	0	1	0	0	Channel 1 ,Axis 1 component requested
0	1	1	0	0	1	0	1	Channel 1 ,Axis 2 component requested
0	1	1	0	0	1	1	0	Channel 1 ,Axis 3 component requested
0	1	1	0	0	1	1	1	Channel 1, All Axis components requested
0	1	1	0	1	0	X	X	Channel number 2 requested: Channels correspond to the physical quantities: Velocity, angle, rate and Navigation (See Table 4)
0	1	1	0	1	0	0	0	Channel 2 ,Axis 1 component requested
0	1	1	0	1	0	0	1	Channel 2 ,Axis 2 component requested
0	1	1	0	1	0	1	0	Channel 2 ,Axis 3 component requested
0	1	1	0	1	0	1	1	Channel 2, All Axis components requested
0	1	1	0	1	1	X	X	Channel number 3 requested: Channels correspond to the physical quantities: Velocity, angle, rate and Navigation (See Table 4)
0	1	1	0	1	1	0	0	Channel 3 ,Axis 1 component requested
0	1	1	0	1	1	0	1	Channel 3 ,Axis 2 component requested
0	1	1	0	1	1	1	0	Channel 3 ,Axis 3 component requested
0	1	1	0	1	1	1	1	Channel 3, All Axis components requested
0	1	1	1	0	0	X	X	Unused currently
0	1	1	1	0	1	X	X	Unused currently
0	1	1	1	1	0	X	X	Unused currently
0	1	1	1	1	1	X	X	All Channels requested: Channels correspond to the physical quantities: Velocity, angle, rate and Navigation (See Table 4)
0	1	1	1	1	1	0	0	All Channels ,Axis 1 component requested
0	1	1	1	1	1	0	1	All Channels, Axis 2 component requested
0	1	1	1	1	1	1	0	All Channels, Axis 3 component requested
0	1	1	1	1	1	1	1	All Channels, All Axis components requested
1	X	X	X	X	X	X	X	SBC is destination.
1	0	X	X	X	X	X	X	Other end is FPGA-A: Reserved for Future.
1	1	X	X	X	X	X	X	Other end is FPGA-B. It is an Information Packet. Bytes 2-n represent the data bytes. If data bits = 16 then byte (i) is most significant byte and byte (i+1) is least significant byte. If data bits = 8 then byte alone represents the value, there is no padding. Bits 5:0 represent the length of the bytes to follow in the IP.

Bytes 2-n

Actuation Packet Bytes 2 - 6	
Byte 2	Address (120 = 0x78 = 0 1 1 1 1 0 0 0 by default)
Byte 3	Servo number to actuate (1 - 20)
Byte 4	Offset's hi; offset is in us of pwm
Byte 5	Offset's lo; offset is in us of pwm
Byte 6	Speed with which the actuation is to be executed (0 for max possible, 1 - 255 : 255 is max, 1 is slowest)
Information Packet Bytes 2 - n	
Byte (i)	If data bits = 16 then byte(i) is most significant byte If data bits = 8 then byte alone represents the value, there is no padding.
Byte (i+1)	If data bits = 16 then, byte(i+1) is least significant byte. If data bits = 8 then byte alone represents the value, there is no padding.
Note: In case multiple channels (and/or) multiple Axis components are requested, then the order of components in the data bytes is strictly according to lowest channel number first and then lowest Axis number first. Hence if all components are requested, then the components would be in order : u, v, w, phi, theta, psi, p, q, r, x, y and z.	

Physical Quantities, Channels and Axis explained

Channel Number	Axis Number	Physical Quantity
0		Velocity
	1	u
	2	v
	3	w
1		Angle
	1	phi
	2	theta
	3	psi
2		Rate of change of angle
	1	p
	2	q
	3	r

3		Navigation Quantities
	1	x=latitude
	2	y=longitude
	3	h=altitude

A2 Simulation and Control

The open loop (no control) simulink MATLAB simulation model is shown in the Figure 4 below.

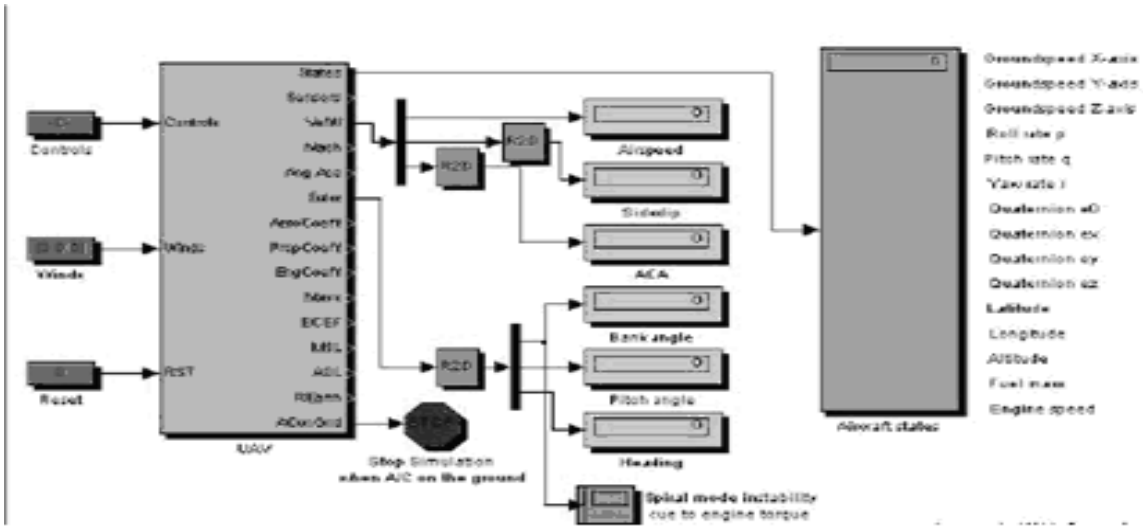


Figure 4

The Lateral control system developed is shown in Figure 5.

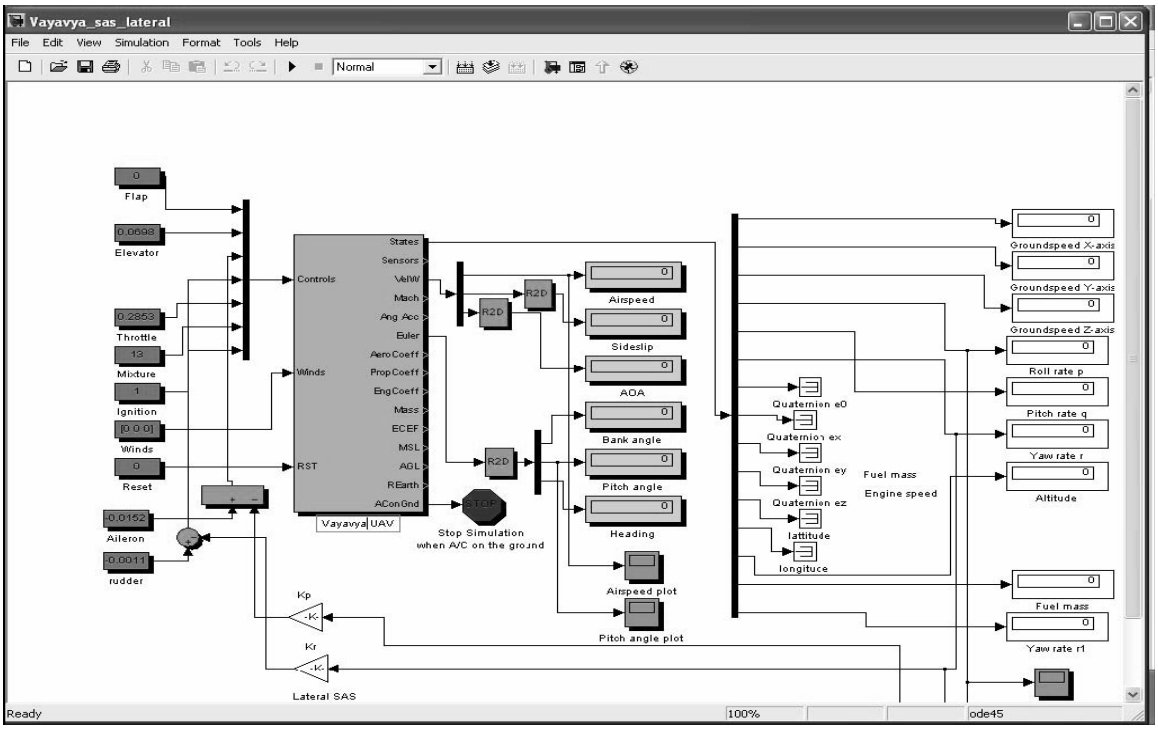


Figure 5

The Longitudinal control system is shown in the figure 6 below:

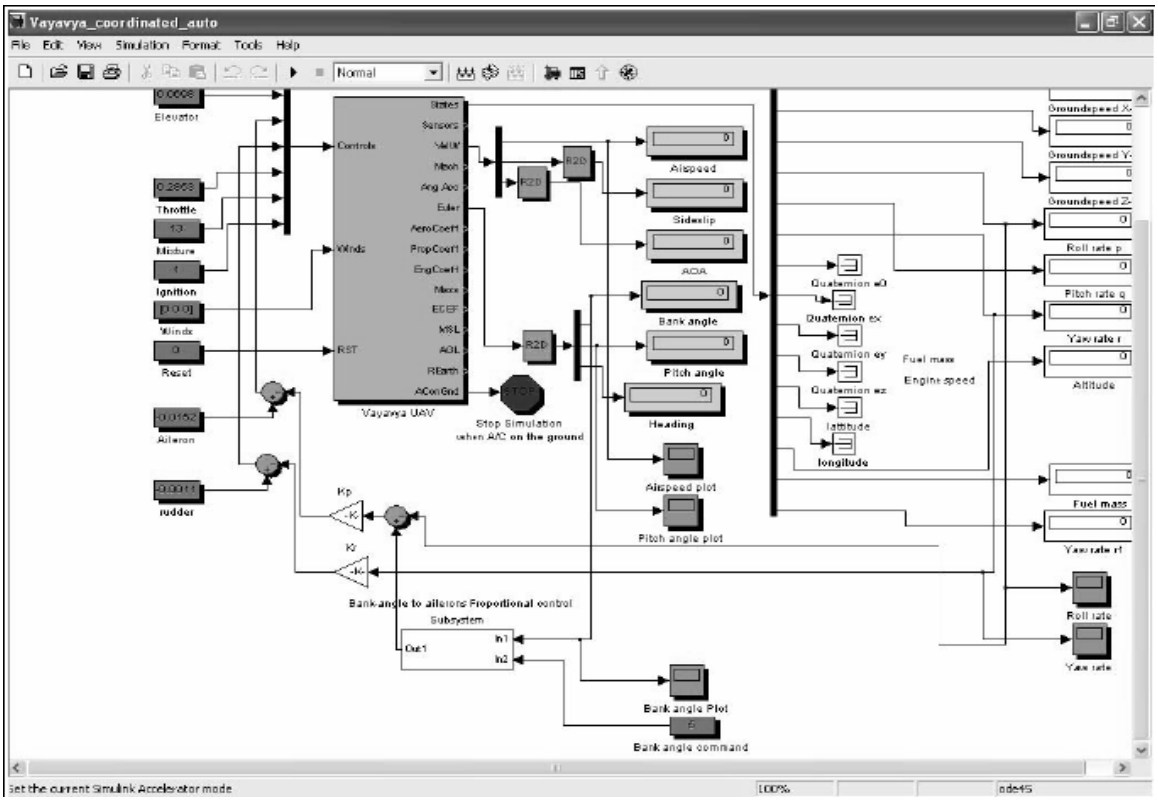


Figure 6