İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

DESIGN OF AN AUTOPILOT SYSTEM FOR A MICRO-AIR VEHICLE

M.Sc. Thesis by
Serdar ATEŞ, B.Sc.

Department : Mechatronics Engineering

Programme : Mechatronics Engineering

JANUARY 2009

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

DESIGN OF AN AUTOPILOT SYSTEM FOR A MICRO-AIR VEHICLE

M.Sc. Thesis by
Serdar ATEŞ, B.Sc.
(518061016)

Date of submission : 29 December 2008
Date of defence examination: 20 January 2009

Supervisor (Chairman) : Asst. Prof. Dr. Gökhan İNALHAN (ITU)
Members of the Examining Committee : Prof. Dr. Levent GÜVENÇ (ITU)
Asst. Prof. Dr. Tankut ACARMAN (GU)

JANUARY 2009

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

KÜÇÜK BİR HAVA ARACI İÇİN OTOPİLOT SİSTEMİ TASARIMI

YÜKSEK LİSANS TEZİ

Elektrik Müh. Serdar ATEŞ
(518061016)

Tezin Enstitüye Verildiği Tarih : 29 Aralık 2008
Tezin Savunulduğu Tarih : 20 Ocak 2008

Tez Danışmanı : Yrd. Doç. Dr. Gökhan İNALHAN (İTÜ)
Diğer Jüri Üyeleri : Prof. Dr. Levent GÜVENÇ (ITÜ)
Yrd. Doç. Dr. Tankut ACARMAN (GÜ)

OCAK 2009

ACKNOWLEDGEMENT

First, I would like to thank my advisor, Prof. Gökhan İnalhan, for giving me opportunity to be a member of his distinguished research team, widening my vision, providing world-class research environment, supporting me materially and morally, and introducing me UAV research area. He is not only a great mentor but also a wonderful leader. Working under his supervision in his lab was a marvelous experience to me.

I would also like to thank TÜBİTAK (Turkey Science and Technology Research Foundation) for supporting me financially during my graduate education.

I owe special thanks to İsmail Bayezit for being my projectmate and his contributions especially about MPC555 microcontroller.

This thesis study is a part of a common project which I and İsmail Bayezit handle together at graduate student level by focusing on the different aspects of it. Basically, I focus on the autopilot design and İsmail Bayezit focuses on the system identification.

I would like to thank Prof. Selim Çetinkaya for trusting and supporting me during my graduate education. I would like to thank Bahadır Armağan for his engineering support in every section of this study. I am deeply indebted to Fatih Erdem Gündüz for his infinite technical support and friendship.

I am very grateful to Miraç Kuddusi Aksugür for his effort during flight tests, sharing his knowledge about flight systems, and his deep friendship. I thank to Melih Fidanoğlu for kindly providing the logistic support and his friendship. Besides, I would like to thank Captain Selim Etger for being the test pilot.

In addition, my stay at Controls and Avionics Lab during my graduate education would have been harder without my labmates Emre Koyuncu, Mehmet Ertan Ümit, Nazım Kemal Üre, Ahmet Cemil Tural, Oktay Arslan, Taner Mutlu and people from Satellite Lab; Can Kurtuluş, Melahat Cihan, Taşkın Baltacı, İlke Akbulut.

I really thank to Tansu Gökçe whom I accept as my brother for his endless and invaluable support. I want to thank Eda Çetinkaya for her very precious support. I cannot describe her meaning to me by using any word in any language.

Finally, I really thank to my family. Their understanding and love mean a lot to me.

January 2009 **Serdar ATEŞ**

TABLE OF CONTENTS

Page

ABBREVIATIONS

CAN : Controller Area Network
ECEF : Earth-Centered-Earth-Fixed
ENU : East, North, Up
GPS : Global Positioning System
HIL : Hardware-in-the-Loop
IMU : Inertial Measurement Unit
LRF : Laser Range Finder
LLA : Latitude, Longitude, Altitude
NED : North, East, Down
UAV : Unmanned Aerial Vehicle
UGV : Unmanned Ground Vehicle
URF : Ultrasonic Range Finder
VTOL : Vertical Take-off and Landing

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF SYMBOLS

$\mathbf{A_{long}, B_{long}, C_{long}, D_{long}}$	: State-space matrices in longitudinal motion
$\mathbf{A_{lat}, B_{lat}, C_{lat}, D_{lat}}$	: State-space matrices in lateral motion
$\mathbf{{}_{NED}C_{ENU}}$	: Conversion Matrix from ENU-frame to NED frame
$\mathbf{{}_{ENU}C_{NED}}$	: Conversion Matrix from NED-frame to ENU-frame
L, M, N	: Moments acting about x-, y-, and z-direction
p, q, r	: Roll, Pitch, and Yaw Rates
U, V, W	: Linear Velocities in the x-, y-, and z-direction
X, Y, Z	: Forces acting about x-, y-, and z-direction
α, β	: Angle of Attack and Angle of Sideslip
Φ, θ, ψ	: Roll, Pitch, and Yaw Angles
Φ, λ, h	: Latitude, Longitude, Altitude

DESIGN OF AN AUTOPILOT SYSTEM FOR A MICRO-AIR VEHICLE

SUMMARY

This thesis study has two objectives. The main objective is designing of an autopilot system for a micro-air vehicle which enables the vehicle to navigate given waypoints autonomously. The other one is designing autonomous take-off and landing algorithms for small UAVs.

The microavionics system which is designed at Controls and Avionics Laboratory as previous research in recent years is used, improved and tested. The microavionics system is bus-backboned and cross-compatible across different experimental platforms such as ground vehicles, mini-helicopters and aircrafts. The bus-backboned architecture helps plug and play new hardware such as different sensors, different processors very easily.

The latest version of this microavionics system is tested on a ground vehicle. Ground vehicle tests consist of collecting sensory data in manual mode, analysing this collected data to design convenient controllers, testing these controllers with the help of HIL simulator, testing the tuned controllers in autonomous mode. At the end of these tests, the ground vehicle is able to navigate given waypoints autonomously.

After completing ground vehicle tests, the microavionics system is translated into a micro-air vehicle. The earlier flight tests are realized to collect sensory data as in ground vehicle tests phase. The convenient control loops are designed, tuned and implemented on the micro-air vehicle with the help of HIL simulator again.

In order to assist next researches on the UAVs, autonomous take-off and landing algorithms are designed and tested in simulation environment. The required sensors are added to the microavionics system and flight tests' data of these sensors are collected. The required procedures to parse the data collected from these sensors conveniently are mentioned and exemplified with the help of the sample sensor data.

KÜÇÜK BİR HAVA ARACI İÇİN OTOPİLOT SİSTEMİ TASARIMI

ÖZET

Bu tez çalışmasının iki amacı vardır. Ana amacı, küçük bir insansız hava aracının otonom olarak verilen koordinatlara gidebilmesini sağlayan bir otopilot sisteminin tasarlanmasıdır. Diğer amacı ise, küçük insansız hava araçları için otonom kalkış ve iniş algoritmalarının tasarlanmasıdır.

Son bir kaç yıl içerisinde önceki araştırma konularından biri olarak Kontrol ve Aviyonik Laboratuvarı'nda tasarlanmış olan mikroaviyonik sistem kullanılmış, geliştirilmiş ve test edilmiştir. Kullanılan bu mikroaviyonik sistem bir veri hattı etrafında çalışmakta ve yer araçları, mini helikopterler, uçaklar gibi değişik deneysel platformlarda çalışabilmesi için çapraz uyumlu bir yapıya sahiptir.

Bu mikroaviyonik sistemin en güncel hali bir yer aracı üzerinde test edilmiştir. Yer aracı testleri, otonom olmayan modda algılayıcı verilerinin toplanmasını, toplanan bu verilerin uygun kontrolcüler tasarlayabilmek için analiz edilmesini, tasarlanan kontrolcülerin HIL simülatörü (donanımın çevrimde olduğu simülatör) yardımıyla test edilmesini ve ayarlanmış bu kontrolcülerin otonom modda test edilmesini içermektedir. Bu testlerin sonunda, yer aracı otonom olarak verilen koordinatlara gidebilme yetisine kavuşturulmuştur.

Yer aracı testlerinin tamamlanmasından sonra mikroaviyonik sistem, küçük bir hava aracına aktarılmıştır. Bu hava aracıyla yapılan ilk uçuş testleri, tıpkı yer aracı test fazında olduğu gibi, algılayıcı verisi toplamak için gerçekleştirilmiştir. Yine HIL simülatörünün yardımıyla uygun kontrol döngüleri tasarlanmış, ayarlanmış ve küçük hava aracı üzerinde test edilmiştir.

İnsansız Hava Araçları üzerinde yapılacak olan gelecek çalışmalara yardımcı olabilmek amacıyla, otonom iniş ve kalkış algoritmaları tasarlanmış ve simülasyon ortamında test edilmiştir. Bu algoritmalar için gerekli algılayıcılar mikroaviyonik sisteme eklenmiş ve gerekli veriler test uçuşları ile toplanmıştır. Bu algılayıcılardan toplanan verilerin uygun şekilde işlenmesi için gerekli olan yordamlardan bahsedilmiştir ve örnek algılayıcı verisi üzerinde gösterilmiştir.

1. INTRODUCTION

The use of unmanned vehicles in civilian (metropolitan traffic monitoring, rapid assessment of disaster areas) and military (reconnaissance, target identification, tracking and engagement) domains has been increasing in recent years. The individually control of UAVs ranging from primitive inner loop control to fully autonomous waypoint navigation control is the fundamental research and implementation area in order to achieve the mentioned assignments. Autopilot term usually covers longitudinal and lateral control which enable the UAV to navigate given waypoint autonomously.

1.1 Thesis Objective and Related Work

This thesis study has two objectives. The main objective is designing of an autopilot system for a micro-air vehicle which enables the vehicle to navigate given waypoints autonomously. The other one is designing autonomous take-off and landing algorithms for small UAVs.

Related works about autopilot design and implementation are examined to design the convenient controllers.

In this study, the design and development of a cross-platform compatible and multiple bus backboned microavionics system that allows to test and develop such standardized hardware and software solutions in laboratory scale micro-air vehicles is provided. The system is designed to be cross-compatible across our experimental mini rotary-wing and fixed wing UAVs, and ground vehicles, and it is tailored to allow autonomous navigation and control for a variety of different research test cases.

Our microavionics system design is based on a research driven fundamental requirement to have a cross-compatible (i.e. the avionics can be used with minor modifications on different types of ground and air vehicles as depicted in Figure 3.1) architecture that can support coordinated autonomy experiments across a heterogenous fleet of vehicles. Another major driver for this platform was that it should allow a collaborative hardware and software development environment, in which researchers can work on different topics (such as flight controls, image-driven navigation or multiple vehicle coordination algorithms) and could later integrate their work for flight and ground experiments with minimal hardware and software redesign concerns. In comparison with elegant but monolithic microavionics architecture structured around different form factors such as single-board computers [7] or PC-104 stacks [8], we consider a scalable multi-processor architecture based on data bus backbones [9, 10]: Controller Area Network (CAN) control/mission bus and Ethernet payload bus.

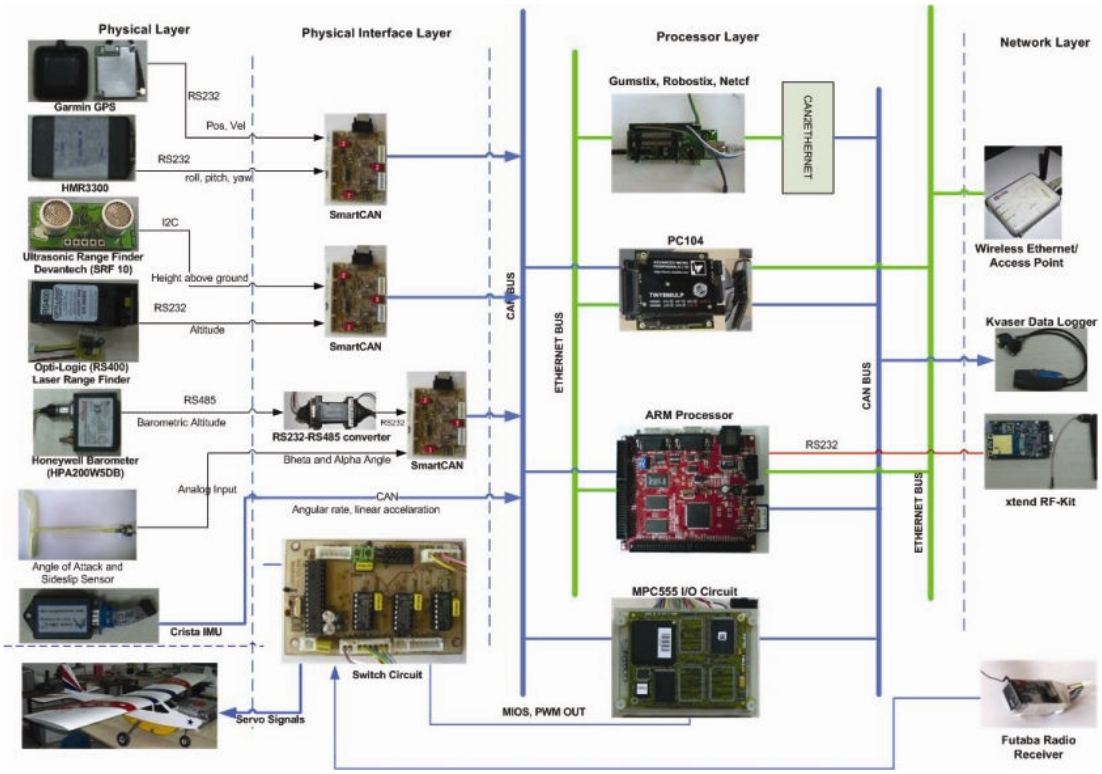

Figure 1.1 : Microavionics General Architecture Diagram

The standardized microavionics hardware includes sensors, a flight control computer (which is also denoted as autopilot), a flight management computer (which is also denoted as mission coordination computer) and a communication unit. The expandable architecture deploys a hybrid selection of COTS Motorola (MPC555), Arm processor boards (LPC2294), PC-104 processor stack and Gumstix modules, each with different operating systems and coding techniques (such as rapid algorithmic prototyping using automatic code generation via Matlab/Real Time Workshop Embedded Target). Specifically, MPC555 is used with Matlab/Real Time Workshop to rapidly design and prototype basic flight control algorithms using Simulink and the MPC555 embedded target automatic code generation feature. This fast-prototyping approach provides not only flexibility at coding level, but also provides seamless cross-platform compatibility. This is a major distinguishing factor of our architecture in comparison with general CAN Bus based architectures [10]. The microavionics system employs a complete sensor suite including Garmin GPS receiver, 3 axis accelerometer/gyro Crista IMU, Honeywell Altimeter and Digital Compass each used as the primary sensors. These units provide the real-time position, orientation and associated time-rate information. In addition, experiment specific units such as laser-range finder, ultrasonic sensors and wireless IP cameras are integrated as plug-and-play add-on modules through custom interface boards. The microavionics system includes a X-Tend wireless transceiver for communication with the ground control station. The cross-platform compatible microavionics design provides an enabling technology to be used for a range of activities including autonomous take-off and landing flight research. This technology has been translated to micro-helicopter [11] and ground vehicle operations, and currently it is being translated to tailsitter VTOL (Vertical Take-off and Landing) aircraft [12] operations within civilian environments that involves agile maneuvering in urban environments [13].

1.2 Thesis Organization

The organization of the thesis as follows: In Chapter 2, the design and the development of the cross-platform compatible microavionics hardware is given. Chapter 3 provides insight on the experimental mini flight and ground platforms. Chapter 4 will cover the mathematical model used. The software development, navigation and control loops, and autonomous take-off and landing algorithms will be discussed in Chapter 5, 6, and 7, respectively. In Chapter 8 HIL simulator will be discussed. Chapter 9 will conclude the thesis study.

2. DESIGN OF THE MICROAVIONICS SYSTEM

2.1 General Architecture

General design architecture of multiple bus backboned of microavionics system as shown in Figure 2.1 is composed of mainly four different layers. First layer: physical layer is composed of different kind of sensory devices, actuators and power circuits. This layer includes Crista IMU, Garmin GPS, Honeywell Altimeter, Honeywell Magnetometer, Laser Range Finder, Ultrasonic Range Finder, Angle of Attack and Sideslip sensors, are providing considerable flexibility in design and testing phase of autonomous and cooperative control experiments. The second layer is physical interface layer which is used for the coordination of sensory information to the CAN Bus with the help of RS485 to RS232 data type converter for Honeywell Altimeter (HPA) and in-house designed SmartCAN nodes, which enable sensory serial or analog input data type to CAN data type. In addition, switching operation between autopilot and remote control with the help of Switch Circuit as shown in Figure 2.6. Switching circuit has already tested our ground and flight platforms as in Figure 3.1 on going operations. Switch Circuit also enables 6 different PWM signals to the PWM Read (MDASM) channels of MPC555 which is important for system identification and model verification of the flight mission platforms. After capturing the PWM signals, MPC555 pack them as CAN messages with specific CAN IDs like sensory information.

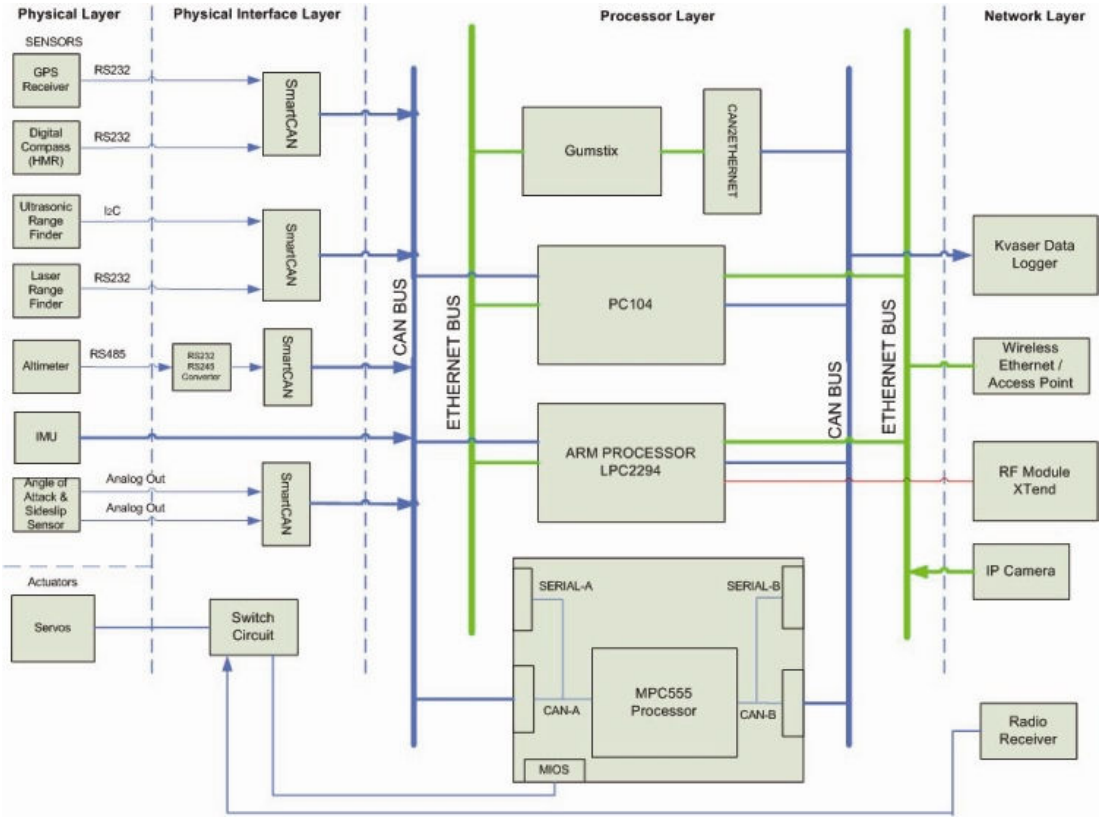

Figure 2.1 : Microavionics General Architecture Functional Diagram

Another layer is processor layer around Ethernet and CAN BUS as in Figure 2.1, which directs the control implementations with the help of Motorola MPC555 processor, high level concepts like task assignment, mission planning and collaborative control with the help of PC-104 processor and ARM processor for communication with Ground Station and coordination pattern computation. In addition, expendable Gumstix single board platform will recover the place of ARM processor depending on minimizing the dimension of microavionics design and decreasing the energy consume. A fourth layer is Network layer and consists of Kvaser data logger which logs information flow over CAN appending a time stamp to the message respectively. Second part of this layer is XTend RF module, IP camera and wireless Ethernet module. ARM Processor board (LPC2294) is provided the connection of RF Module with CAN BUS and all sensory information, and servo signals send to Ground Station with the help of XTend 900 MHz RF transceiver during the flight and ground tests. In addition, the IP Camera and wireless Ethernet module is used for real-time video transmission. Because of the security of the wireless communication and our system, separate battery is used for XTend RF Kit, and some other equipments.

2.2 Processors

Motorola MPC555 processor is a 40 MHz processor in Figure 2.1 and 2.6 having a PowerPC core with a floating point unit, accelerates the advanced algorithms. In addition, 26 kbytes of fast RAM is included. There is also a 448 kbytes of Flash EEPROM. A serial communication includes queued multi-channel module (QSMCM). MPC555 has two on-chip RS232 drivers and dual on-chip CAN 2.0B controller modules (TouCANs). The MPC555 core also includes two time processor units (TPU) and a modular I/O system (MIOS1). 32 analog inputs are included in the system with dual queued analog-to-digital converters (QADC64) [www.freescale.com]. In addition, integration of the phyCORE555 single board computer module on our new in-house designed MPC Carrier Board as in Figure 2.1 and 2.6 is achieved. As a consequence, our necessary general purpose IOs are available for our architecture and this board allows In-System Programming of MPC flash memory with the help of serial channel. As a result, our newly designed carrier board, covers 1/3 the area of the previous, phyCore555 and consume less energy.

Motorola MPC555 processor with our printed MPC555 Carrier Board, is the core processing unit in the microavionics architecture, which enables the programmer to design and prototype algorithms rapidly. One of the most critical utility is the support of Matlab/Simulink Real-Time Workshop. Programmer is able to use all capacity and devices of Motorola MPC555 with the support of Embedded Target Blockset and Automatic Embedded code generation feature. Embedded Target Blockset for MPC555 includes CAN Drivers, CAN Message Blocks, MPC555 Driver library and also versatile support for the user with some demos and example programs which illustrate effective using methodology of the library. In addition, all configurations of MPC555 resource are controlled directly from the MPC555 Resource Configuration Block of Simulink MPC555 specific library. As a result we are not related with low level programming of the processor so much, but even so we are strictly related with developing high level control and mission algorithms of the autonomous systems.

In our design, one of the mainly used functions of MPC555 hardware is the Modular Input Output System (MIOS) unit, which includes MIOS digital in, digital out, eight dedicated pins for MIOS Pulse Width Modulation Out and ten pins for MDASM Pulse Width or Pulse Period Measurement. The second of these function blocks is CAN Message Blocks which enables transmitting or receiving CAN data over CANBUS and converting data type during the communication. These blocks include not only CAN Message Packing, CAN Message Unpacking before and after transmission, but also features splitting the messages during the operation. The availability of simple and rapid shaping of message in Matlab/Simulink environment allows us to parse the sensory information with basic mathematical operations. Complex algorithms and mathematical actions are also accelerated with the floating point unit of the MPC555. Finally TouCAN transmit and TouCAN receive blocks directly use two on-chip CAN communication units and configure the CAN message format. These crucial advantages of the card, we met some problems on-going operations, which are sourced from the specifics of MPC555. For instance, it is strictly necessary to supply the MPC555 board 3.3V and 5V at the same time. Sometimes failures occur in voltage regulator IC of our MPC carrier board because of the cold weather. As a result, MPC555 caused problems during outside tests.[1]

The Arm Processor (LPC2294) [www.nxp.com] microcontroller is based on a 32-bit CPU in ARM7 family with its 256kbyte of embedded flash memory. 128-bit wide interface enables high speed 60 MHz operation. It is included four interconnected CAN interfaces with advanced filtering capability, serial interfaces include two UARTs (16C550), Fast I2C-bus (400kbit/s) and two SPIs. In addition, eight channel 10-bit ADC, two 32-bit timers (with four capture and four compare channels), PWM unit (six outputs), Real-Time Clock (RTC), and watchdog. TCP/IP communication is also supported by the Ethernet controller chip locating bottom side of the ARM. Real Time Operating Sytems such as QNXi, VxWorks are supported by ARM board and with this capability, extensive number of operations are applicable.

[1] In our earlier flight tests, we faced the failures of the MPC555 and its in-house designed carrier board. After an intensive review we could detect the low atmospheric temperature as the source of failures. We usually make test flights in the earlier hours of the day when the atmospheric temperature is about 0 Celcius degree. At this temperature level the voltage regulator IC in the in-house designed carrier board cannot work properly so that the MPC555 processor resets itself in infinite cycle. Furthermore we tested our microavionics system design at low level temperatures and fixed the carrier board. The test procedure involves keeping all microavionics hardware just like it is mounted on the aircraft (with batteries) in a refrigerator along 30 minutes about 0 Celcius degree and powering-up the hardware. We repeated this procedure in several times.

Arm processor board is mainly used in the communication of the ground station with the Avionics Box, computation of coordination patterns and this board is one of the main part of processor layer of our microavionics hardware. Various custom codes in C were written for this purpose such as CAN and Serial Communication, CAN to Serial conversion and vice versa, digital Kalman filtering and sensory message parsing.

Tiny886ULP - Ultra Low Power PC104+ Computer (http://amdltd.com) is a 100 % PC-AT compatible single board computer having PC104 bus connectivity. Tiny886ULP provides 1 GHz Crusoe processor (with FPU), 512Mb of SDRAM, IDE disk interface, 10/100 ethernet, 2xRS232, VGA output, 1xUSB, PC104+ bus and PS2 keyboard/mouse interface. With average power consumption of less than 5W provides easy integration, performance, low-power dissipation (fanless) for microavionic applications. Windows, Linux and QNX operating systems can be run on Tiny886ULP and various PC104 boards have driver support for these operating systems. Such as: CAN104 PC104 CAN controller board (provides 2 CAN 2.0b interfaces which gives the ability to communicate with our present avionic bus.) and Firespeed2000 board (provides three IEEE-1394 (firewire) serial ports. Hi-speed data transfers are enabled from such as video sources, state of the art avionic busses.)

The 100 % PC-AT compatibility makes it possible to develop code directly on Tiny886ULP and support for Windows, Linux and QNX operating systems make very easy to reusing and porting existing codes which are developed on regular desktop PCs.

Gumstix connex (http://www.gumstix.com/) single board computer measures just 80 mm x 20 mm x 6.3 mm but according to its small size it has considerable process power which makes gumstix ideal for microavionic systems. Gumstix connex board provides a 400 MHz Marvell XScale PXA255 processor (ARM V5 compatible), 64MB of SDRAM and 16MB on-board flash memory. Connectivity is achieved by 60-pin Hirose I/O header and a 92-pin bus header to additional expansion boards all powered from single 5V supply. We use "netCF" expansion board which provides compact flash card and 10/100 ethernet interface for network connectivity attached to our gumstix board. "Robostix" expansion card provides 6xPWM output, 2xUART,

1xI2C and 8xADC and very useful for robotic applications which can also be operated standalone.

Linux 2.6 operating system runs on gumstix and a free cross-compilation toolchain "buildroot" is provided by manufacturer for software development.

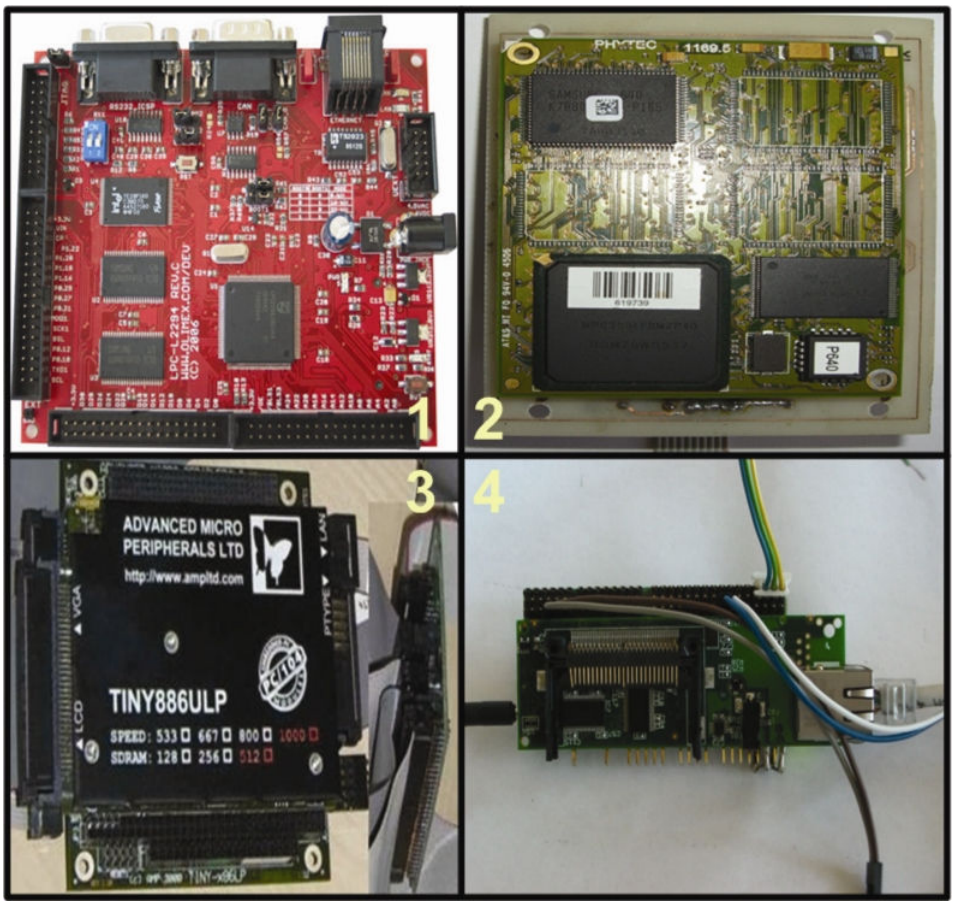

Figure 2.2 : Processors - The ARM Processors (LPC2294), MPC555 with In-house Designed Carrier Board, Tiny886ULP Ultra Low Power PC104+ Computer, Gumstix with Robostix and Netcf

2.3 Sensor Suite

Sensor suite in the microavioncs system design consists of an IMU, a GPS receiver unit, a magnetometer (digital compass), a barometric altimeter, a laser range finder, and an ultrasonic range finder.

As the GPS receiver unit, we use the readily available Garmin 15H. This device can track up to 12 satellites to determine the position and the velocity of the vehicle, and also outputs a PPS timing signal. This PPS signal is connected to the IMU for time stamping of IMU data. The NMEA sentences that will be transmitted by the GPS receiver unit are user selectable through its custom interface. The GPS data is

received approximately every one second. This unit is able to output the acquired data via a RS232 based serial interface. This unit provides precise and accurate position and velocity information of the vehicle which is notably vital for our design. Because, our navigation and control algorithms are very sensitive to position measurements.

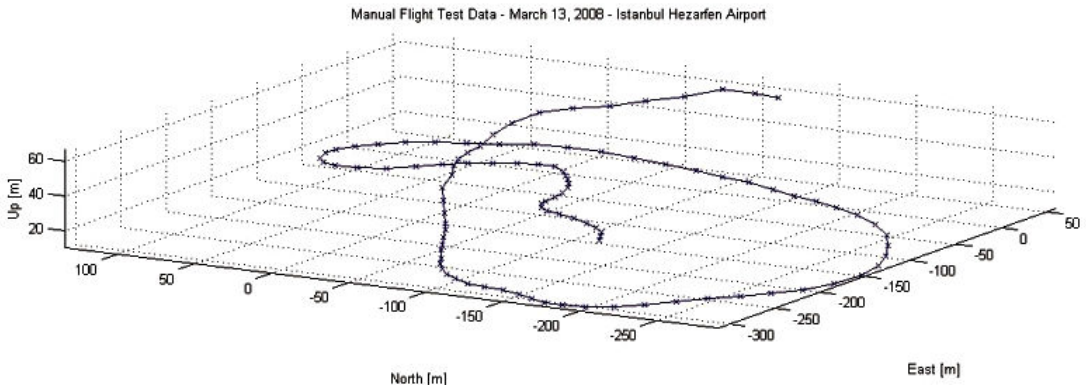

Figure 2.3 : Manual Flight Test GPS Data Recorded on March 13, 2008 at Istanbul Hezarfen Airport (Last Loop and Landing) for Autonomous Take-off and Landing Sensor Verification

For inertial acceleration and angular velocity measurements, we use Crista Inertial Measurement Unit (IMU) provided by Cloud Cap Technology. This unit is able to output the acquired data both via a RS232 based serial interface and a CAN interface. It also provides internal oversampling and averaging of the oversampled data automatically before transmission. The data transmission and oversampling rates are user selectable. The IMU is also able to time stamp the acquired signals via the PPS signal acquired from the GPS receiver unit. A data update rate of 50 Hz is used in this work with an oversampling rate of 20 Hz. Especially angular velocity measurements (p, q, r) are required by the autopilot control and navigation loops. Without these measurements, precise control of the vehicle is not possible.

The barometric altimeter unit used, Honeywell HPA (HPA200W5DB), is able to provide both temperature and pressure measurements. Thus the pressure measurements can be compensated according to the temperature. We use the factory setting date update rate of 10 Hz. The data update rate for this device is user selectable. This unit is able to output the acquired data via a RS232 based serial interface. This sensor gives us opportunity to calculate barometric altitude which must be measured for an air-vehicle navigation and control.

The magnetometer (digital compass) used for inertial orientation aiding is the Honeywell HMR 3300. This unit provides three axis measurement of orientation relative to the earth magnetic field. The orientation information of the heading angle is of 360 degrees and fully covers the range whereas the pitch and the roll are of 60 degrees. The data update for this device is 8 Hz. This unit is able to output the acquired data via a RS232 based serial interface. These Euler angles measurements are also vital and required by the autopilot control and navigation loops. Without these measurements, control of the vehicle is not possible at all.

In order to develop autonomous landing and take-off algorithms, the air-vehicle's actual altitude has to be measured very precisely and accurately to prevent failures. To achieve this, a laser range finder and an ultrasonic range finder are used. The laser range finder (LRF) unit used for the actual altitude which is strongly dependent on the terrain is the Opti-Logic laser range finder (RS400). The data update for this device is 10 Hz. This unit is able to output the acquired data via a RS232 based serial interface. This unit provides actual altitude of the vehicle relative to the terrain. The altitude information ranges up to 400 Yards.

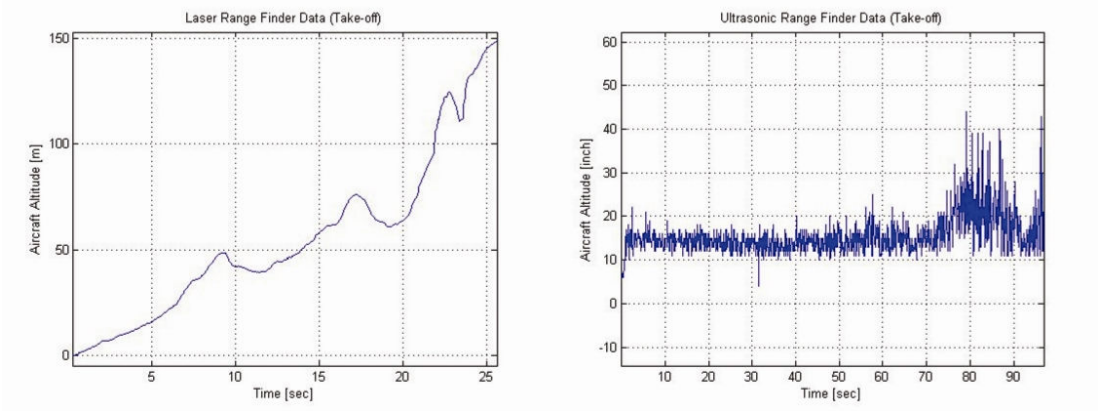

Figure 2.4 : Laser Range Finder and Ultrasonic Range Finder Data During Take-off (Manual Flight Test Data Recorded in April, 2008 at Istanbul Hezarfen Airport)

The ultrasonic range finder (URF) unit used for the actual altitude which is measured while take-off and landing is Devantech SRF10 ultrasonic range finder. This unit is able to output the acquired data via a I2C based interface. The data update rate for this device is 10 Hz. This unit provides actual altitude of the vehicle relative to terrain. The altitude information ranges up to 255 inches with one inch resolution.

An angle of attack (AoA) sensor and an angle of sideslip (AoS) sensor will be used in our microavionics system design to measure angle of attack and angle of sideslip which are two important states of the air-vehicle and must be used to design elegant control and navigation algorithms. These sensors resemble vertical stabilizers of air-vehicles and are coupled with infinite rotary potentiometers to give analog outputs related to the angular position.

An RF transceiver kit used for making a communication link between the air-vehicle, the ground station and the other vehicles is XTend 900 MHz RF transceiver.

The flight data recording unit is used for acquiring data from the CAN bus and record it to SD and MMC compatible cards. The memory can be up to 2 GBs which allow recording for hours for this work. The recorded data can easily be taken to a PC for analysis. This CAN based recording unit is also used as a CAN-USB converter, which can then be used for better debugging or system monitoring purposes. We use the Kvaser Memorator CAN data logger as the flight data data recording unit.

We made an execution profiling analysis (as shown in Figure 2.5) after connecting all sensor devices to the MPC555 processor to test the capability of the processor. With 0.019994 sec (about 20 msec which is our sampling time) the average sampling time maximum task turnaround time is 0.000559 sec in the worst case and the average turnaround time is 0.000542 sec. In this execution profiling analysis all sensory data are read and parsed to calculate the elapsed time for control and filtering. About 3 percent of processing capacity of the processor is used for parsing and the remaining source of processor is enough for control and filtering.

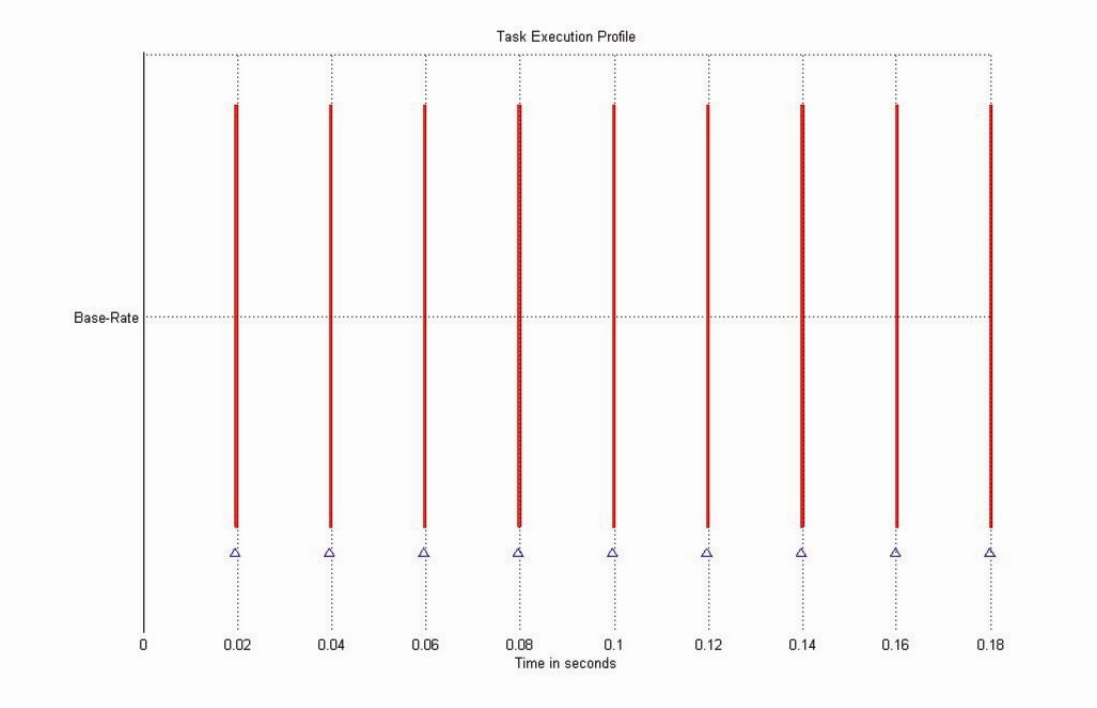

Figure 2.5 : Execution Profiling Analysis

2.4 Customized Boards: SmartCAN and Switch

SmartCAN: The distributed multi-processor design as shown in Figure 2.1 is structured around CAN Bus line (with additional flexibility to include an Ethernet bus line also) providing the ability to interoperate independent of processors with different functionalilites, native operating systems and thus coding standards. This also streamlines the microavionics allowing the reconfigurability for different sensor sets through sensor customization via SmartCAN modules (as shown in Figure 2.6).

Most of sensors used in the microavionics system design are not compatible with CAN Bus and have different output such as RS232, UART, I2C, analog, digital, etc... Our in-house developed SmartCAN modules have all sensors structured around CAN bus, read, parse, and send all sensory information with predefined CAN IDs to the CAN bus. These CAN IDs are selected in an importance order to give the critical sensors priority.

Every SmartCAN node design uses PIC18F458 to direct sensory information flow to and from CAN bus line. In case of using sensors which have RS232 or UART output, MAX232 IC is used to convert TTL signals. After reading sensory information, it is processed by the PIC microcontroller and transmitted to the CAN

bus via using PCA82C250 which is used for transition from the CAN protocol controller to the physical bus CAN High and CAN Low outputs.

Four different sensors can be connected to a SmartCAN node. They can be switched and monitored with on-board physical switches and system alive/data streaming led indicators. The SmartCAN customized boards can be programmed with in-circuit debugger programming ports which eliminate the handicap of unmounting the PIC ICs for programming.

PIC microcontrollers are programmed with the help of timer interrupt routines to reach the sampling rate of each sensor. In addition, watchdog timer software protection is activated to avoid sensor reading failures such as GPS receivers cause in many times while initializing.

Switch: Eventhough the microavionic system is designed for autonomous flight tasks, it is very probable that unpredictable situtations take form during flight tests, therefore a mechanism that will let the human factor intervene the platform during such conditions is a very critical functionality. For this purpose, a switch board as shown in Figure 2.6 is designed. An extended feature of the switch board is the capability of switching the PWM outputs between the manual flight (by RC) and autonomous flight (by MPC555).

The Switch Board takes 8 PWM inputs from the RC radio transceiver and 6 inputs from the MPC555 (or any other board) and directs the control inputs to the servo actuators. 2 of the 8 PWM inputs is for the channel selection and RC alive information purpose and the remaining 6 PWM inputs is for human control. The switching operation is established by a PIC18F873 and the 74HCT125 three-state octal buffer is used as the interface of channel switches. When one of the control sources (human pilot or MPC555) is not available, the Switch board assigns the other one as the actual controller automatically. In addition, for system identification and health-monitoring purposes, the signals going to the servo actuators are provided as another output which are read by the MPC555 in our architecture. These signals can be used by MPC555 to estimate the servo position at any desired time.

This board has an individual battery supply to avoid failures of the microavionics system design and its power bus, which utilizes the switch board to keep working properly even the microavionics system design is out of order.

This customized board also can be programmed with in-circuit debugger programming port which eliminates the handicap of unmounting the PIC for programming.

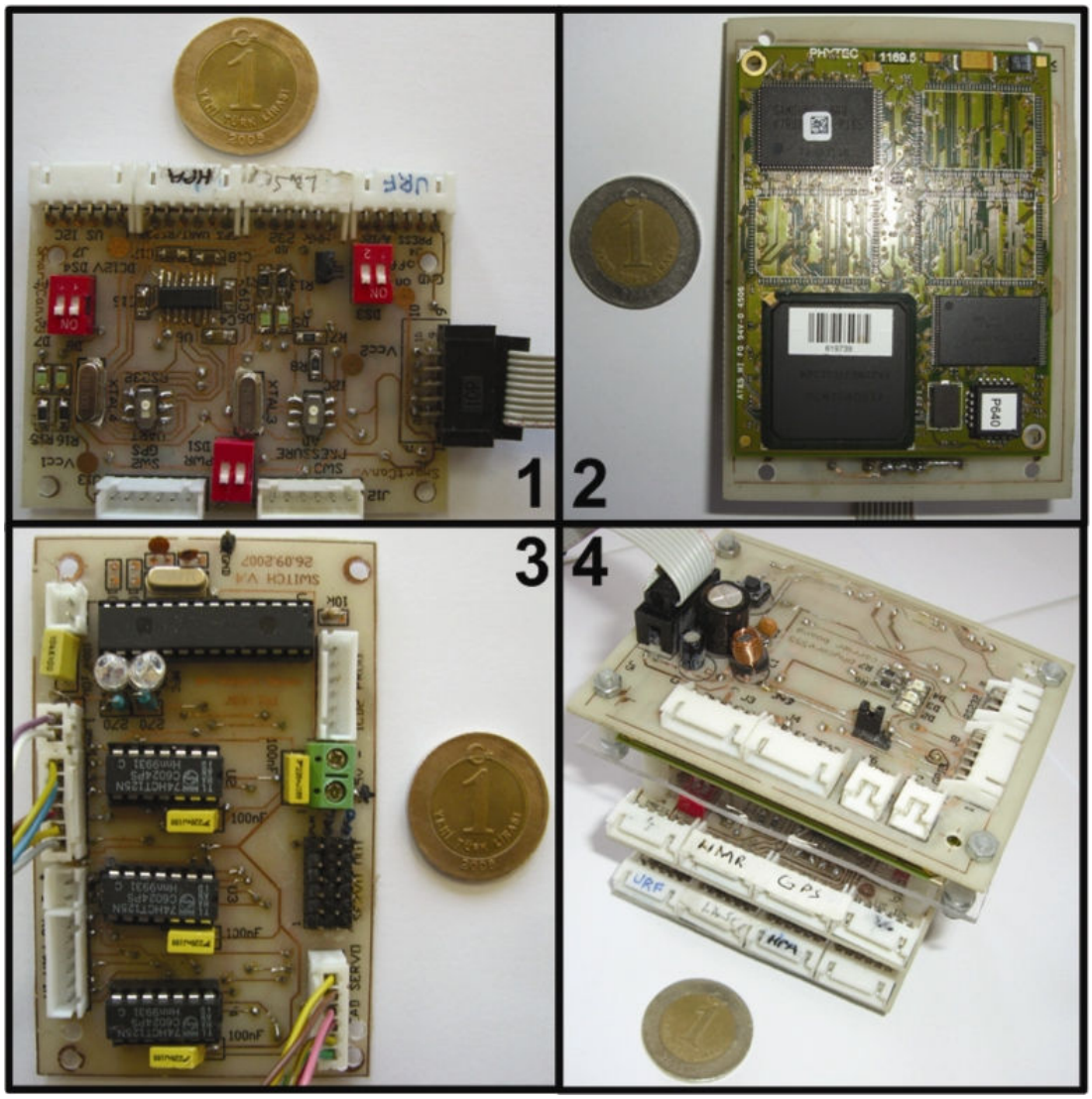

Figure 2.6 : Autopilot Hardware - SmartCAN Node, MPC555 with In-house Designed Carrier Board, Switch Board, Autopilot Deck

2.5 Ground Station

The ground station interface, which is shown in Figure 2.7, is used not only to visualize the air-vehicle sensor data, but also to send waypoint data to the air-vehicle's mission planning processor. The Ground Station graphical user interface is designed with Delphi 7.0 and has three tabs. At the first tab, pilot primary flight data, basic sensor reading with the sampling rates, CAN messages, MPC Alive state can be seen. At the second tab, the digital map is loaded and waypoint commands can be

sent to the air-vehicle. At the third tab, initial parameters related to serial communication protocol and digital map file can be changed. This GUI interface can record all CAN messages as Kvaser Memorator data logger does with the same data format, and can playback these log files simultaneously. It can also send a heathbeat signal to the air-vehicle to inform whether or not the communication link is still alive.

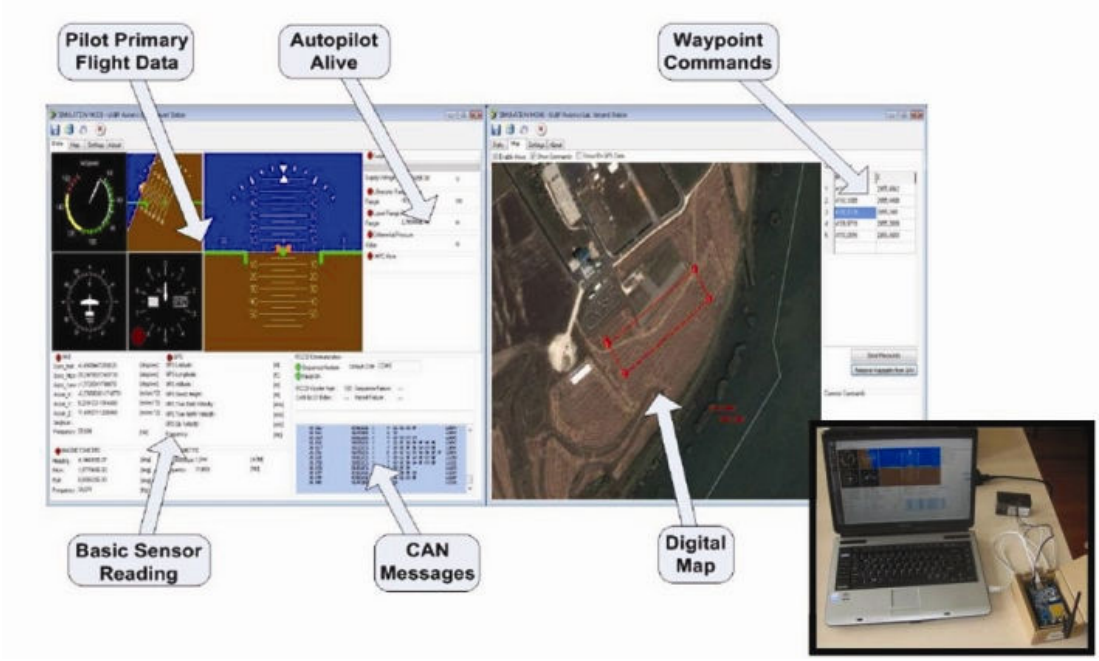

Figure 2.7 : Ground Station Graphical User Interface and Hardware Including a Laptop PC with RF Communication Module

Microavionics hardware setup with ground station on desktop configuration can be seen in Figure 2.8 and microavionics hardware setup on the aircraft configuration can be seen in Figure 2.9.

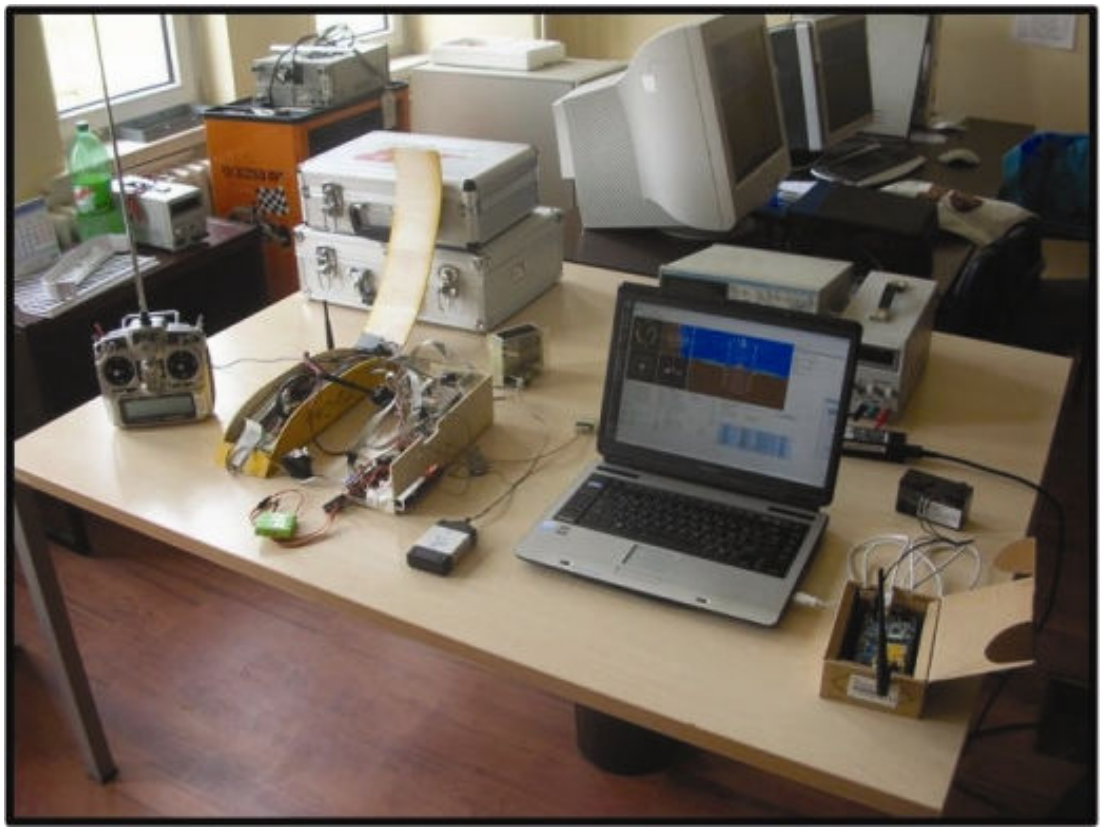

Figure 2.8 : Microavionics Hardware Setup with Ground Station on Desktop Configuration

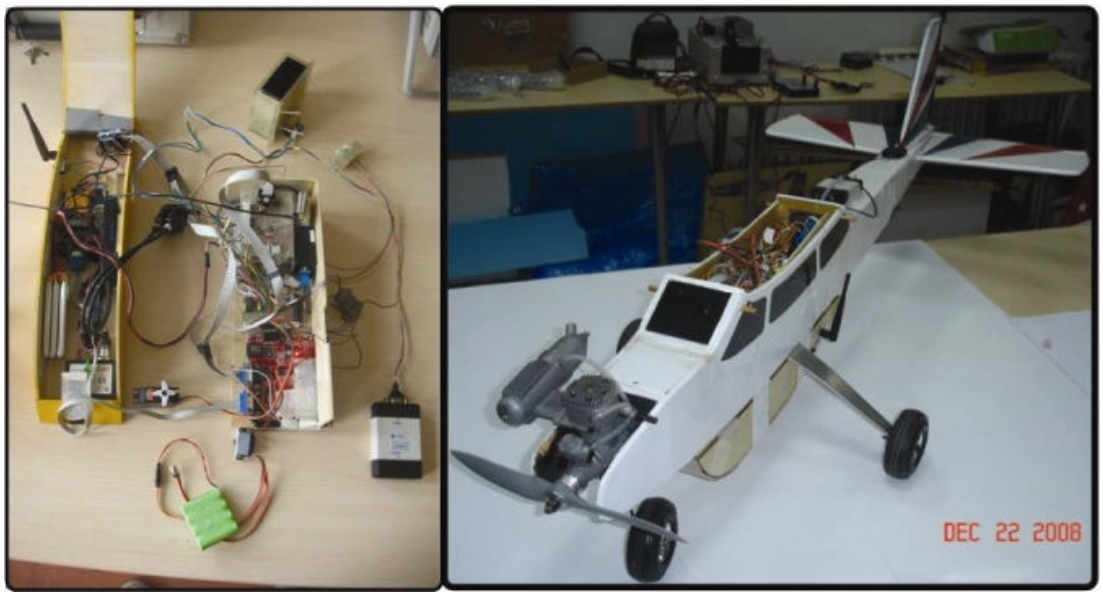

Figure 2.9 : Microavionics Hardware Setup on the Aircraft Configuration

3. TESTBEDS

A collection of in-house modified real UAVs/UGVs (including Vario Benzine and Acrobatic Helicopters, Trainer 60 fixed-wing platform and a 1/6 scale Humvee [11]) are integrated with our cross-platform compatible microavionics systems to support autonomous flight. The experimental micro-helicopter, fixed-wing and ground vehicles equipped with sensors and on-board computing devices, provide the necessary infrastructure to support advanced research on autonomous flight including vertical take-off and landing, vision based control and distributed autonomy.

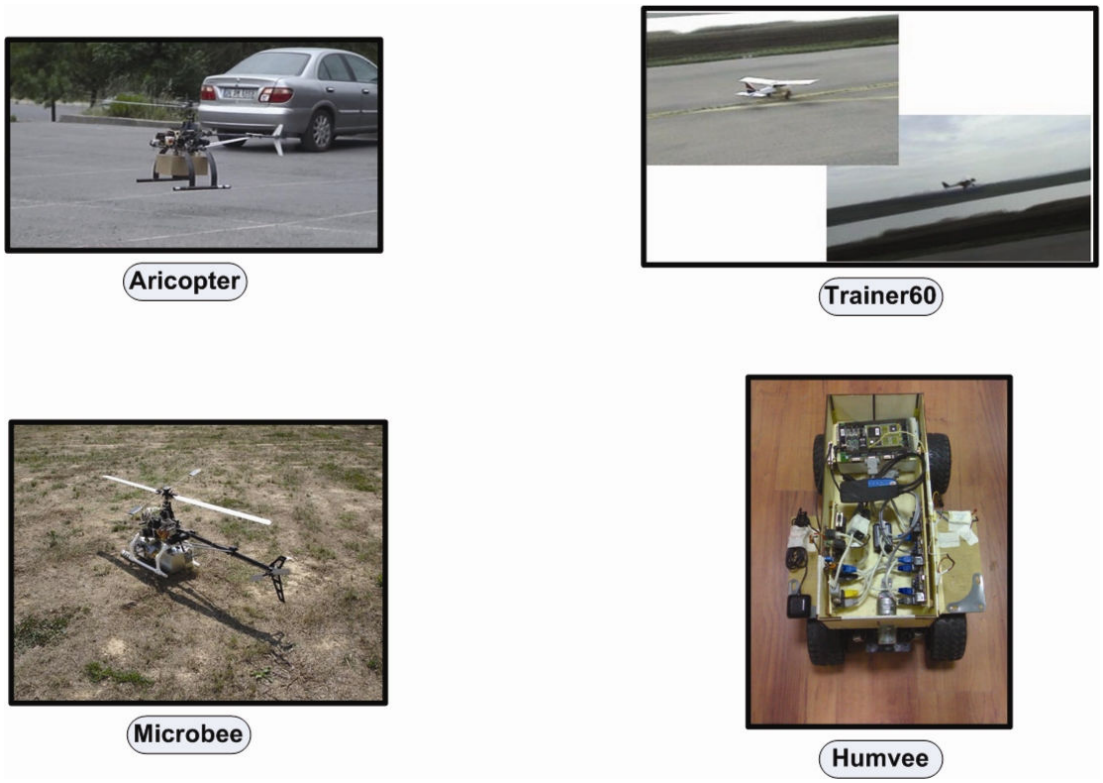

Figure 3.1 : Real Unmanned Aircraft and Ground Vehicles

"Aricopter's" base structure is a COTS benzine helicopter manufactured by German model helicopter manufacturer Vario. The Benzine trainer model of Vario was selected as the flying platform, because of Vario's advantages on price, structural strength, and lifting capacity, over the other options like Century Heli's "Predator Gasser" and Bergen R/C's "Industrial Twin". "Microbee" is an advanced flight platform that is based on Vario's Acrobatic helicopter. This helicopter has an

increased flight envelope and enables acrobatic maneuvers to be carried out. "Trainer 60" is a beginner level fixed wing platform with a wingspan of 160 cm and serves as the main aircraft platform for fixed wing tests. "Humvee" is a ground vehicle which is used for testing the basic functionality of the avionics system. It is a 1/6 scale model of Hummer and it is manufactured by Nikko.

These experimental flight and ground platforms provide a unique opportunity to prototype and demonstrate novel concepts in a wide range of topics covering agile maneuvering [7], advanced flight controls [9, 14], active vision sensing and fleet autonomy [15].

Humvee is the main platform as ground vehicle and Trainer 60 (shown in Figure 3.2) is the main platform as aerial vehicle in this thesis study.

Figure 3.2 : Trainer 60 Aircraft - The Main Aerial Vehicle at Controls and Avionics Lab

4. SIX DEGREE OF FREEDOM NONLINEAR AIRCRAFT MODEL AND ITS LINEARIZED FORM

Because of the fact that this thesis study focuses on controlling an aircraft as well as modelling an aircraft is out of the scope of this thesis study, generic equations of motion related to a fixed-wing aircraft are summerized in this chapter. The main references for this chapter are [1, 6, 16-20].

It is stated as "*The rigid body equations of motion are obtained from Newton's second law, which states that the summation of all external forces acting on a body is equal to the time rate of change of the momentum of the rigid body, and the summation of the external moments acting on the body is equal to the time rate of change of the moment of momentum (angular momentum).*" in [17].

$$\sum F = \frac{d}{dt}(mv) \tag{4.1a}$$

$$\sum M = \frac{d}{dt}H \tag{4.1b}$$

[X, Y, Z] are components of resultant aerodynamic force; [U, V, W] are components of velocity of the aircraft's center of mass; [L, M, N] are rolling, pitching, yawing moment terms, respectively; [p, q, r] are rolling, pitching, yawing velocity terms, respectively [1]. They can be seen in Figure 4.1.

4.1 Force Equations

$$X - mgs\theta = m(\dot{U} + qW - rV) \tag{4.2a}$$

$$Y + mgc\theta s\phi = m(\dot{V} + rU - pW) \tag{4.2b}$$

$$Z + mgc\theta c\phi = m(\dot{W} + pV - qU) \tag{4.2c}$$

4.2 Moment Equations

$$L = I_x \dot{p} - I_{xz} \dot{r} + qr(I_z - I_y) - I_{xz} pq \quad \textbf{(4.3a)}$$

$$M = I_y \dot{q} + rq(I_x - I_z) + I_{xz}(p^2 - r^2) \quad \textbf{(4.3b)}$$

$$N = -I_{xz} \dot{p} + I_z \dot{r} + pq(I_y - I_x) + I_{xz} qr \quad \textbf{(4.3c)}$$

where

$$I_x = \iiint_V (y^2 + z^2)\delta m \quad \textbf{(4.4a)}$$

$$I_y = \iiint_V (x^2 + z^2)\delta m \quad \textbf{(4.4b)}$$

$$I_z = \iiint_V (x^2 + y^2)\delta m \quad \textbf{(4.4c)}$$

$$I_{xy} = \iiint_V xy\,\delta m \quad \textbf{(4.4d)}$$

$$I_{xz} = \iiint_V xz\,\delta m \quad \textbf{(4.4e)}$$

$$I_{yz} = \iiint_V yz\,\delta m \quad \textbf{(4.4f)}$$

I_x, I_y, and I_z are moment of inertia terms about x, y, and z axis, respectively. I_{xy}, I_{xz}, and I_{yz} are called products of inertia.

4.3 Kinematics Equations

$$\dot{\theta} = qc\phi - rs\phi \quad \textbf{(4.5a)}$$

$$\dot{\phi} = p + qs\phi tg\theta + rc\phi tg\theta \quad \textbf{(4.5b)}$$

$$\dot{\psi} = \frac{qs\phi + rc\phi}{c\theta} \quad \textbf{(4.5c)}$$

4.4 Navigation Equations

$$\dot{p}_N = Uc\theta c\psi + V\left(-c\phi s\psi + s\phi s\theta c\psi\right) + W\left(s\phi s\psi + c\phi s\theta c\psi\right) \quad \textbf{(4.6a)}$$

$$\dot{p}_E = Uc\theta s\psi + V\left(c\phi c\psi + s\phi s\theta s\psi\right) + W\left(-s\phi c\psi + c\phi s\theta s\psi\right) \quad \textbf{(4.6b)}$$

$$\dot{h} = Us\theta - Vs\phi c\theta - Wc\phi c\theta \quad \textbf{(4.6c)}$$

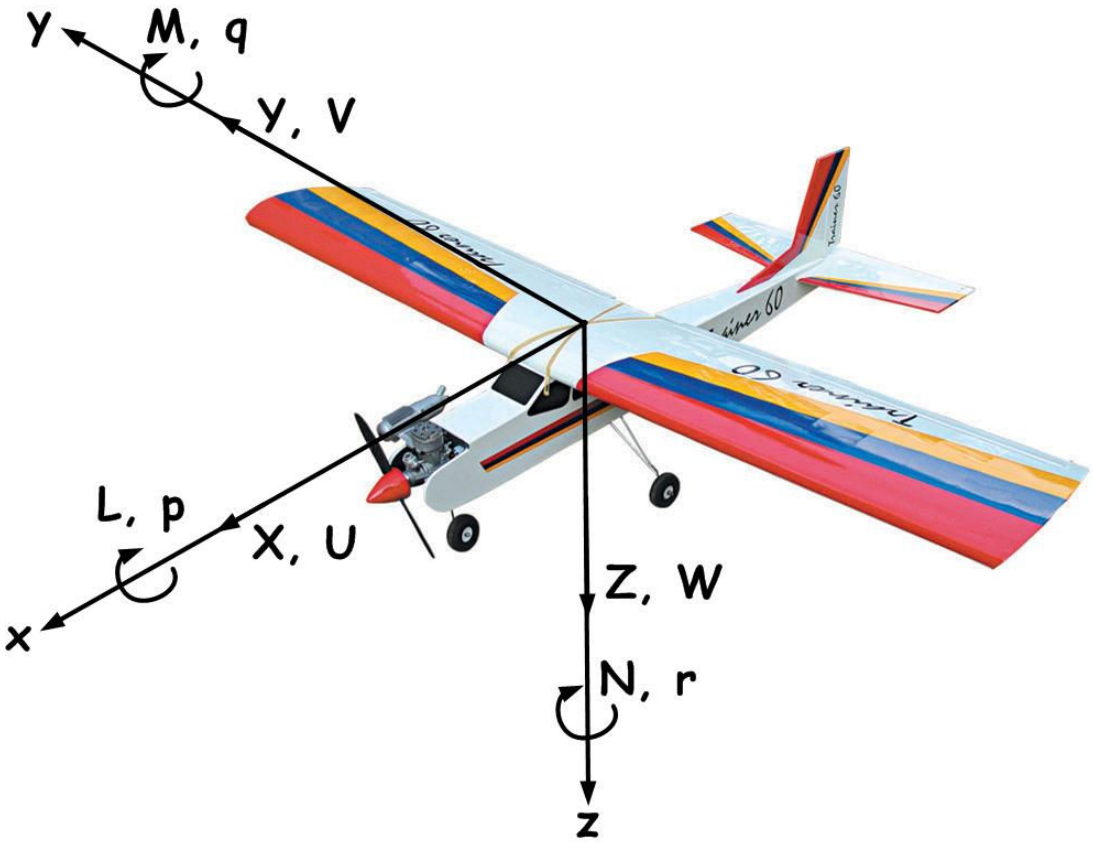

Figure 4.1 : Notation

4.5 Linearization

The aircraft has a nonlinear behavior. In order to describe the linear lateral and longitudinal systems some approximations which limit the flight envelope of the aircraft must be done. The linear model used in this thesis study is taken from [18] and modified by adding throttle input.

4.5.1 Small Angle Approximation

Small angle approximation is a widely-used linearization technique in the literature. The most important equations used in this technique are given in 4.7.

$$u = \frac{\Delta U}{U_0} \tag{4.7a}$$

$$\Delta\alpha = \frac{\Delta V}{U_0} \tag{4.7b}$$

$$\Delta\beta = \frac{\Delta W}{U_0} \tag{4.7c}$$

assuming that

$$\sin(\varepsilon) \approx \varepsilon \tag{4.8a}$$

$$\cos(\varepsilon) \approx 1 \tag{4.8b}$$

for small ε values.

where u is change in velocity, Δα is angle of attack deviation, and Δβ is angle of sideslip deviation.

4.5.2 Longitudinal Model

The state-space representation of longitudinal model used as mathematical model is given below.

$$\dot{x}_{long} = A_{long} x_{long} + B_{long} u_{long} \tag{4.9a}$$

$$y_{long} = C_{long} x_{long} \tag{4.9b}$$

$$\dot{x}_{long} = \begin{bmatrix} -0.106 & -0.265 & 0 & -0.354 \\ -0.354 & -12.1 & 1 & 0 \\ 1.36 & -29.8 & -3.85 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ \Delta\alpha \\ q \\ \Delta\theta \end{bmatrix} + \begin{bmatrix} 0 & 10 \\ -0.768 & 0 \\ -25.2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\delta_e \\ \Delta\delta_{th} \end{bmatrix} \tag{4.10}$$

$$y_{long} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ \Delta\alpha \\ q \\ \Delta\theta \end{bmatrix} \tag{4.11}$$

$$x_{long} = \begin{bmatrix} u \\ \Delta\alpha \\ q \\ \Delta\theta \end{bmatrix} \tag{4.12}$$

where u is change in velocity, Δα is angle of attack deviation, q is pitch rate, Δθ is pitch angle deviation, In addition, D matrix is a zero matrix because it does not have any direct effect on the output.

4.5.3 Lateral Model

The state-space representation of lateral model used as mathematical model is given below.

$$\dot{x}_{lat} = A_{lat}x_{lat} + B_{lat}u_{lat} \tag{4.13a}$$

$$y_{lat} = C_{lat}x_{lat} \tag{4.13b}$$

$$\dot{x}_{lat} = \begin{bmatrix} -0.101 & 0.0115 & -0.0056 & 0 & -0.997 \\ 0 & 0 & 1 & 0 & 0 \\ -200 & -0.500 & -19.4 & 0 & 5.74 \\ 0 & 0 & 0 & 0 & 1 \\ 19.37 & -0.0003 & -0.392 & 0 & -0.833 \end{bmatrix} \begin{bmatrix} \Delta\beta \\ \Delta\Phi \\ p \\ \Delta\Psi \\ r \end{bmatrix} + \begin{bmatrix} 0 & 0.274 \\ 0 & 0 \\ 209 & 18.4 \\ 0 & 0 \\ -2.47 & -51.1 \end{bmatrix} \begin{bmatrix} \Delta\delta_a \\ \Delta\delta_r \end{bmatrix} \tag{4.14}$$

$$y_{lat} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta\beta \\ \Delta\Phi \\ p \\ \Delta\Psi \\ r \end{bmatrix} \tag{4.15}$$

$$x_{lat} = \begin{bmatrix} \Delta\beta \\ \Delta\Phi \\ p \\ \Delta\Psi \\ r \end{bmatrix} \tag{4.16}$$

where Δβ is sideslip deviation, ΔΦ is roll angle deviation, p is roll rate, ΔΨ is yaw angle deviation, and r is yaw rate. In addition, D matrix is a zero matrix because it does not have any direct effect on the output.

5. SOFTWARE DEVELOPMENT

Algorithmic design and development of the multi-operational autonomous system is split into five different sections: flight/ground testing, system identification and model verification process, mathematical modeling of the flight/ground platform, controller design and simulation as depicted in Figure 5.1. First of all, our UAV/UGV platforms are realized around simple and fundamental generic mathematical models to simulate the system and design the appropriate controllers. These appropriate controllers are used in real flight and ground tests to develop the dynamic model of the flight or ground platforms. In addition, the verified dynamic models are tested with hardware and software simulation setups to improve controllers and autonomous algorithms. Then the conceptual design loop returns back to the outside flight and ground experiments to test the modified controllers and algorithms to improve the autonomy of the system. Our extra capabilities and abilities to realize the algorithmic design concept are described in the following sections.

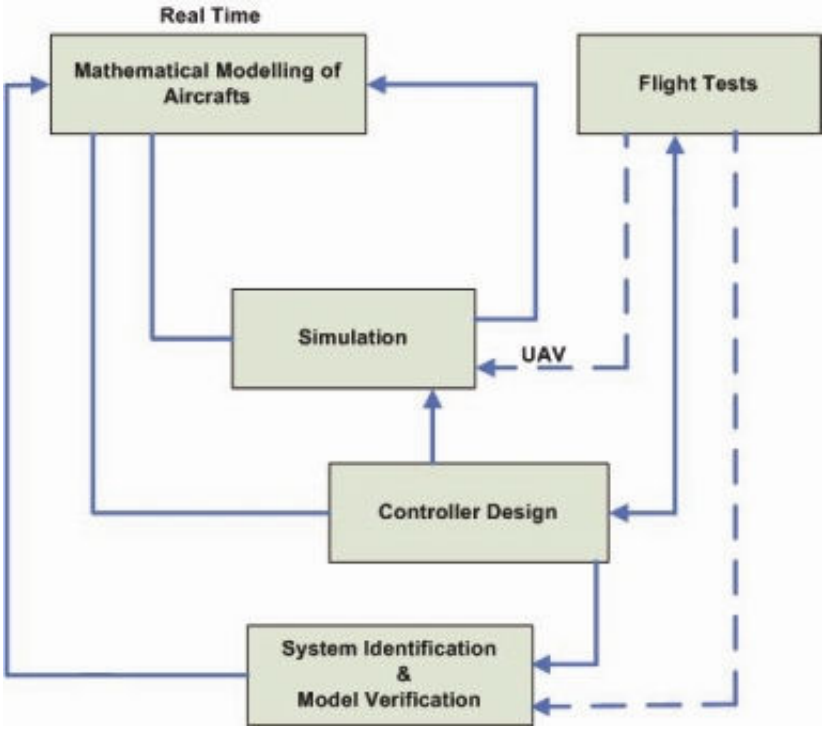

Figure 5.1: Design and Development Concept of Autonomous Flights

Matlab/Simulink is the main software development environment for designing the basic autonomous control algorithms (including landing and take-off) and waypoint navigation which can be rapidly embedded into the MPC555 microcontroller. All the inner and the outer control loops, and the navigation loops typical to mini UAVs are also realized on the MPC555 microcontroller. The microavionics system design is also capable for testing and implementation of autonomous control algorithms, agile maneuvers, and with minor software modifications it can also be used in formation flight experiments. The same microavionics design is currently being transformed for usage in a VTOL tailsitter [12] aircraft which has distinct and switching flight regimes such as hover flight, level flight, and hover-level flight transitions.

In the following sections, we will review two basic algorithmic development processes as examples. First, the autonomous control and a waypoint navigation experiment on "Humvee" is illustrated. Through this, we will provide insight on the extensive usage of Matlab/Simulink environment for simulation and actual code generation for the outside tests. Second example details the HIL testing of the Trainer 60 autopilot design before the actual flight tests. Both applications demonstrate the flexibility and the rapid-prototyping capability of the microavionics system.

5.1 Autonomous Control and Waypoint Navigation Experiment on Humvee

The process that we followed to develop the ground vehicle closed loop control and navigation algorithms is illustrated in Figure 5.2.

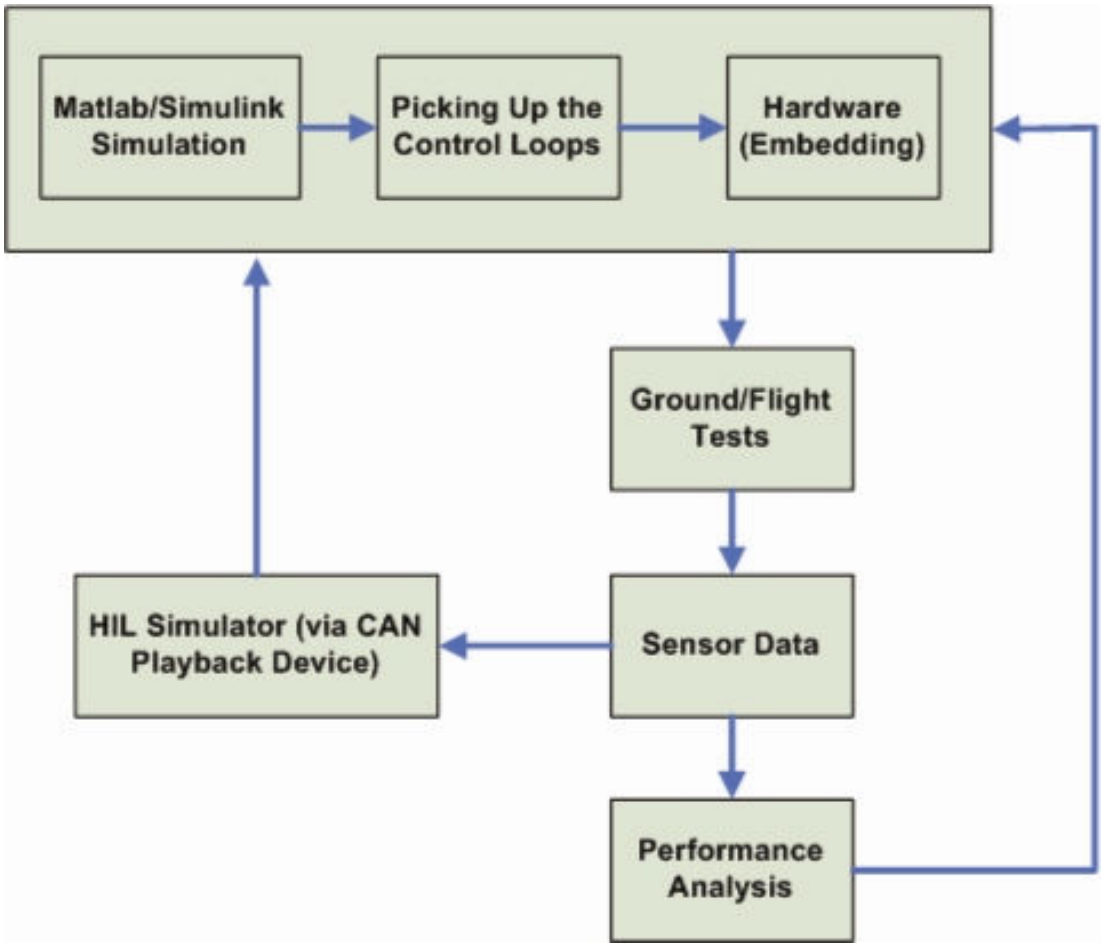

Figure 5.2: Microavionics Control Implementation Process Flow Diagram

At the first step, we made a Matlab/Simulink simulation which emulates all sensory data and control architecture via using a non-holonomic car-like ground vehicle's (as shown in Figure 5.3) dynamics.

$$\dot{x} = v\cos\psi \quad \textbf{(5.1a)}$$

$$\dot{y} = v\sin\psi \quad \textbf{(5.1b)}$$

$$\dot{\psi} = \frac{v}{l} tg\phi \quad \textbf{(5.1c)}$$

v is the linear velocity, ϕ is the steering angle, and ψ is the heading angle of the ground vehicle as depicted in Equation 5.1. The detailed information about the mathematical model mentioned in Equation 5.1 and in Figure 5.3 can be found in [2, 21].

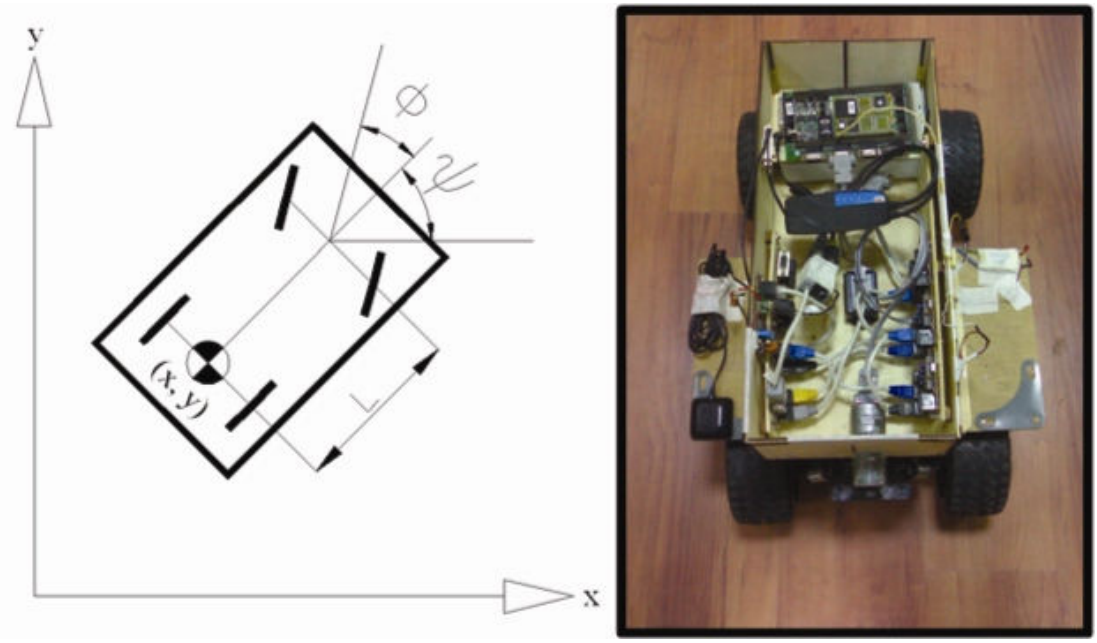

Figure 5.3: A Non-holonomic Mobile Robot Moving in a 2D Space [2] and Humvee Photo

The classical PD-controller design (as shown in Figure 5.4, 5.5) which controls the heading of the vehicle is tested with the help of this simulation. The principal aim of this simulation is to tune the two parameters which are proportional and derivative coefficients of the controller. Before adding our waypoint navigation algorithm to the simulation, the PD-controller is tuned.

We tuned the controller after outside testings. Initial values of the controller coefficients are selected arbitrary and changed according to outside testing performance. Percent overshoot, settling time and rise time are changed by tuning these parameters.[2] Our simple waypoint navigation algorithm is added to the simulation after obtaining an acceptable performance. In addition, the tuned parameters are cross-validated by Matlab/Simulink simulation, which is an exception to tune the controller in this order for our specific application. According to solid implementation process as depicted in Figure 5.2 outside testings are done after completing simulation phase.

[2] At the beginning phase of tuning controller, we faced with some mechanical problems. After a peer review we noticed that the source of these problems was servo actuator shifting. Hence, servo actuator of steering axe is trimmed.

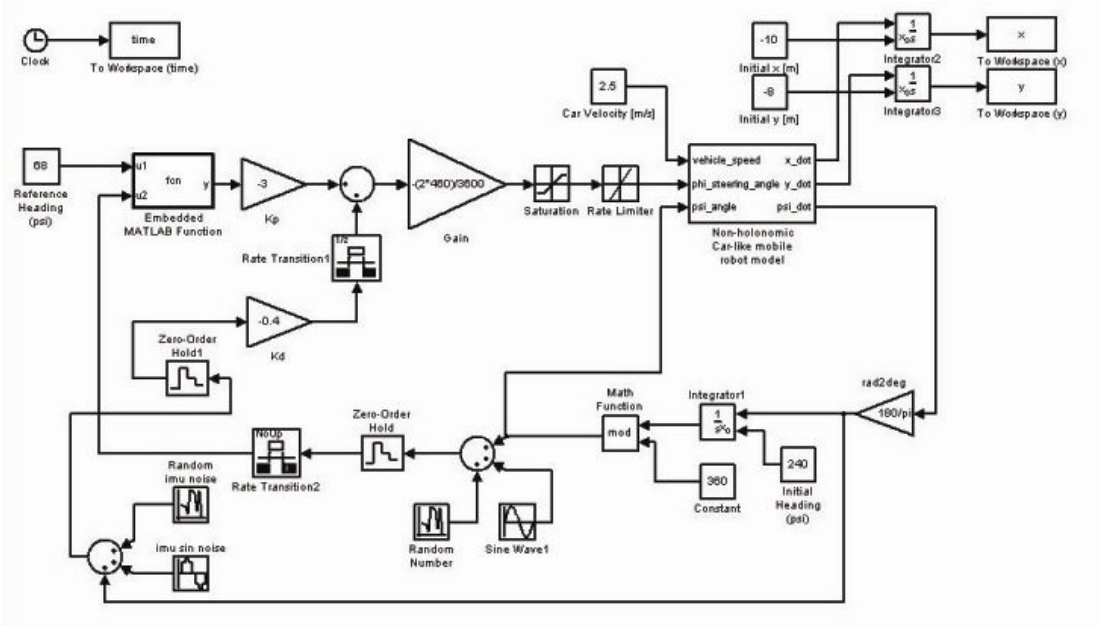

Figure 5.4: The Classical PD-controller Design Simulation which Controls the Heading of the Vehicle. In This Simulation Magnetometer and IMU Models which have Realistic Errors are Added.

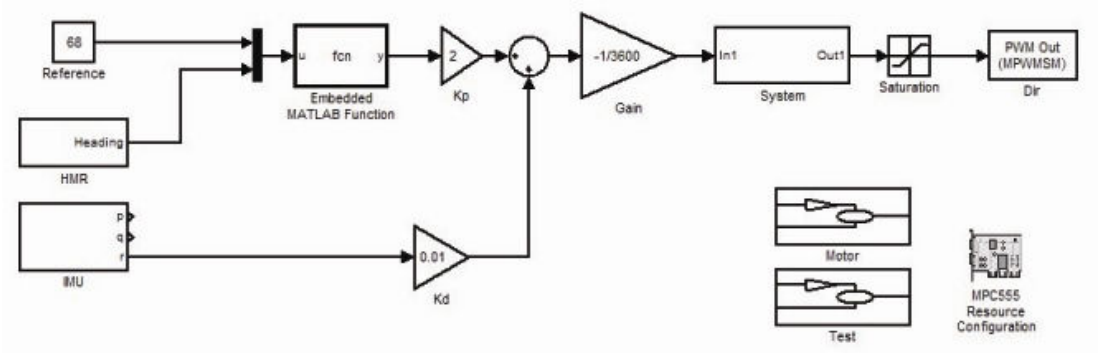

Figure 5.5: The Simulation which is Embedded into the MPC555 Microcontroller.A Magnetometer and an IMU are used to Update the Heading and Heading Rate Measurements, respectively.

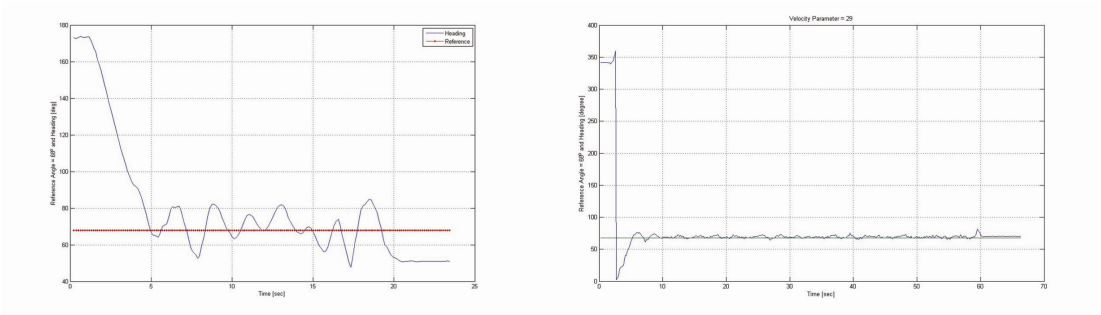

Figure 5.6: PD-controller Performance Results for Heading of the Ground Vehicle from the Real Outside Tests. The Reference Heading Angle is Taken as 68 [deg] in These Tests. The Left One is Untuned Controller Result, which has 66.03 [deg] Mean Value with the 10.71 [deg] Standard Deviation. The Right one is Tuned Controller Result, which has 69.27 [deg] Mean Value with the 1.74 [deg] Standard Deviation. These Outside Tests Took Place on October 19 - 2007 and October 20 - 2007, respectively.

Essentially, our simple waypoint navigation algorithm which is based on this heading controller tries to reach the given waypoints by changing the reference input angle between the vehicle heading and the target waypoint as in Figure 5.7. The waypoint navigation algorithm runs between two points with comparing the distance between the given waypoint and the ground vehicle. If the distance is equal to or less than a predefined limit (δ), the algorithm accepts the following point as the new target waypoint. All target waypoints are embedded in two vectors (latitude and longitude vectors in waypoint navigation algorithm Embedded Matlab Function Block) and waypoint navigation algorithm is fed by these latitude-longitude couples in a predefined order. Our microavionics system design is also capable of accepting target waypoints via Ground Station as depicted in Section 2.5. In real outside tests, a GPS receiver is used to update the position measurements and a digital compass is used to update the heading measurements. By the way, several GPS surveys have been achieved in order to verify and validate GPS sensor which is critical for the microavionics system as depicted in Appendix B. Linear velocity of the ground vehicle is taken as constant in order to simplify the controller structure. The simulation results, as seen in Figure 5.10, show that the performance of our controller and waypoint navigation algorithm is satisfactory.

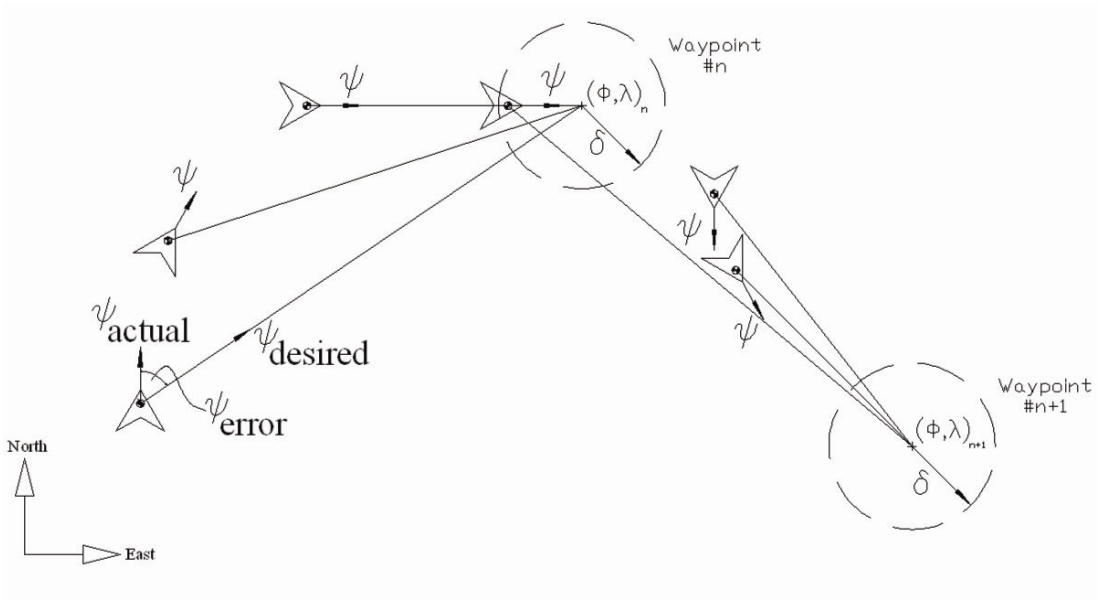

Figure 5.7: Primitive Heading Control Algorithm for Waypoint Navigation

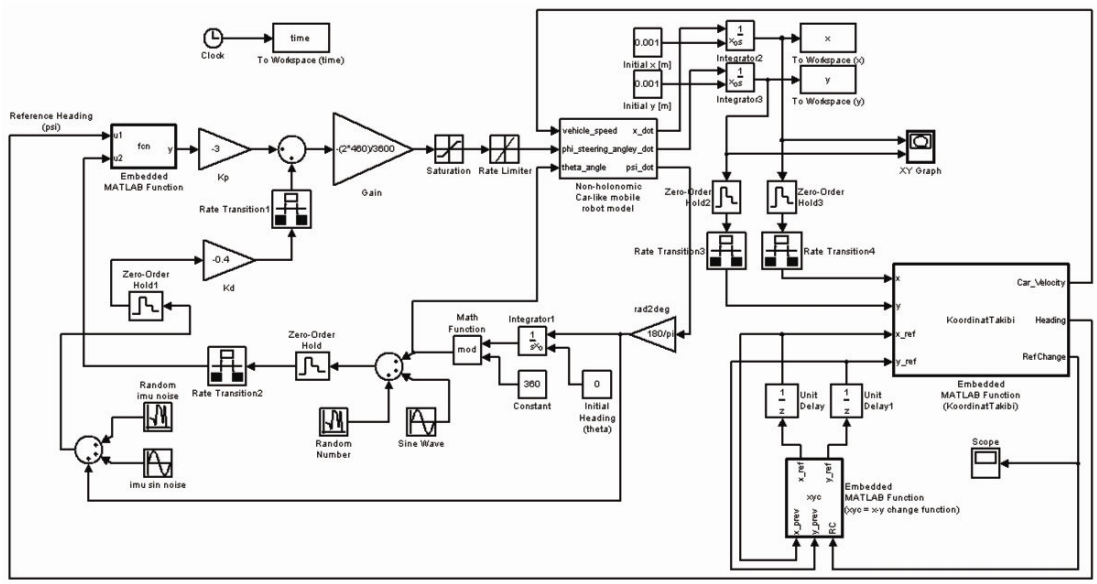

Figure 5.8: Matlab/Simulink Simulation of Heading Control Algorithm

After verification of the simulation, it is adapted to the MPC555 microcontroller. The new Matlab/Simulink simulation which is compatible with the Real-Time Workshop is given in Figure 5.5. Most of the blocks of the previous simulation must be replaced by the Matlab Real-Time Workshop compatible ones. For instance, ground vehicle's state blocks are replaced with TouCAN blocks (standard blocks of MPC555 Simulink library) to measure position information by GPS receiver and heading information by digital compass in adapted simulation. Moreover PWM Out blocks (standard blocks of MPC555 Simulink library) are added to drive servo actuators on the ground vehicle and mathematical model of the ground vehicle in the previous

simulation is removed because it is unnecessary to use in real outside tests. In addition, embedded codes needed for the design of the control and navigation algorithms when standard Simulink blocks are insufficient or exhaustive can be included in this simulation by using Embedded Matlab Function blocks in both standard C or m-file (Matlab Script Language) notation. After the adaptation procedure, the C code required by the MPC555 microcontroller is generated and embedded into the MPC555 microcontroller. The controller hardware (MPC555) becomes ready for outside testing. The outside test results given in Figure 5.10 can be used in two ways which are performance analysis and HIL simulator integration as depicted in Figure 5.2. The controller algorithm can be fine-tuned after performance analysis, system identification and model verification can be carried out, and mathematical model of the unmanned vehicle can be improved by using the sensor data obtained from outside tests. This sensor data can also be played back for visualization. In our application, the FlightGear open-source flight simulator is the main visualization environment.

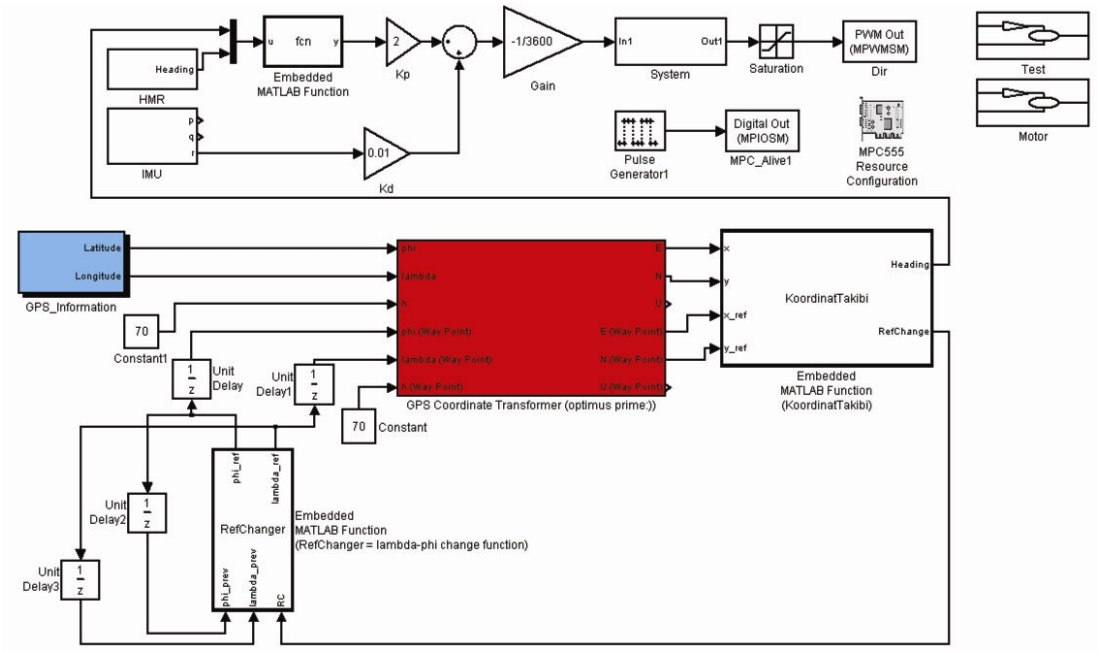

Figure 5.9: Matlab/Simulink Simulation of Heading Control Algorithm which is Embedded into the MPC555 Microcontroller

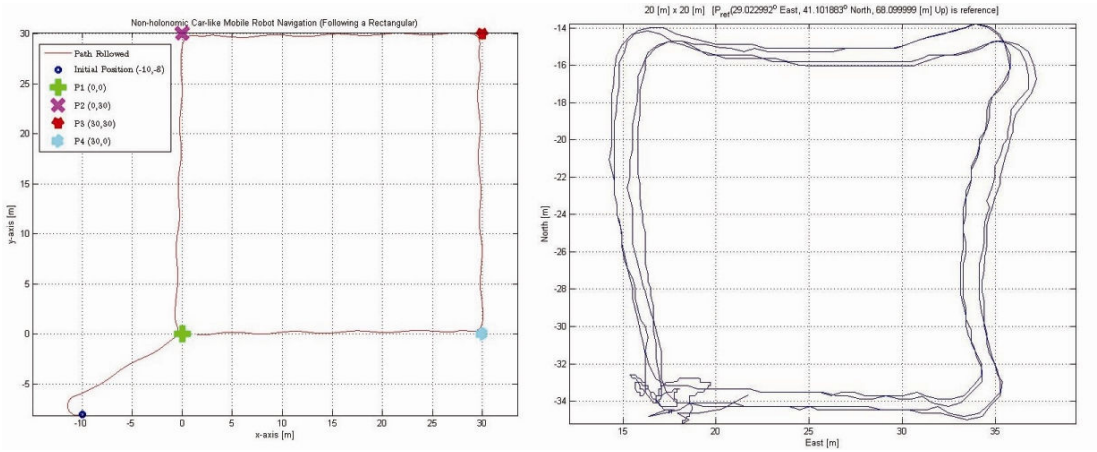

Figure 5.10: Matlab/Simulink Simulation and Outside Test Results for Closed-Loop Navigation and Control of Humvee (Waypoints [(0;0), (0;20), (20;20), (20;0)] meters in ENU frame for outside testing.

This microavionics control implementation process takes less time than conventional methods do and enables the researchers to focus on the design of control and navigation algorithms at a higher level and prevents the unnecessary effort of low-level programming.

6. NAVIGATION AND CONTROL LOOPS

All inner and outer control loops and control design structure discussed in this chapter were inspired and taken from [22]. In spite of the fact that some loops were modified so as to fit our implementation, the main control design structure was protected in this thesis study.

Basically heading hold, velocity hold, and altitude hold as depicted in Figure 61 must be accomplished in order to enable the air-vehicle to navigate given waypoints autonomously.

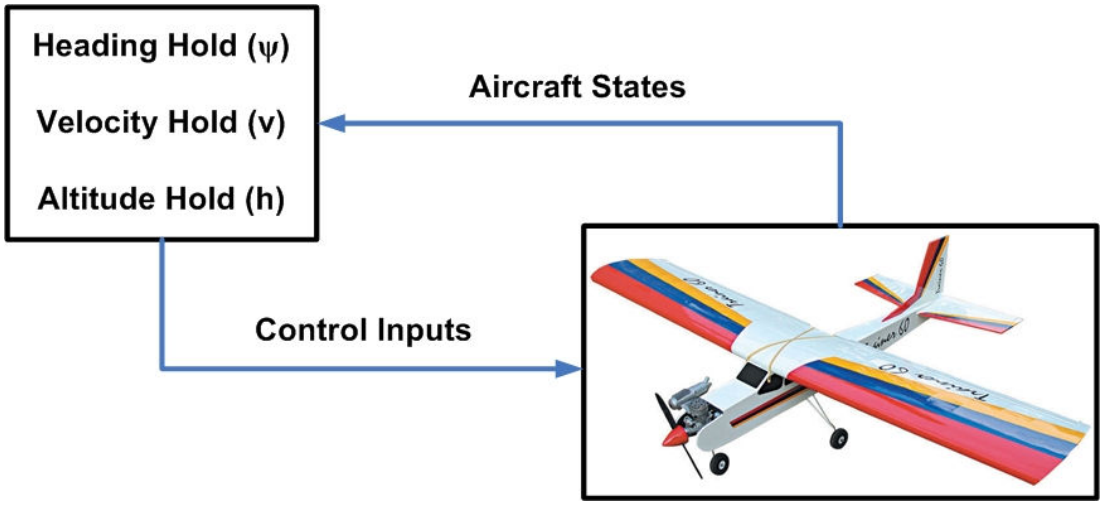

Figure 6.1: Heading, Velocity and Altitude Hold

The air-vehicle control design can be achieved in two divisions which are lateral and longitudinal divisions. Each division has inner control loops which control the related control surfaces, and outer control loops which control the related inner loops. All loops are PID-based. PID control gains are adjusted manually using the mathematical linear aircraft model described in Section 4.

6.1 Lateral Control

Yaw rate, roll angle, and heading must be controlled with the help of three inner loops and one outer loop for the air-vehicle lateral control. Inner loops drive the aileron and rudder servo actuators directly. These inner loops are driven by the outer loop. Outer loop basically generates the required reference values for the inner loops. This outer loop is driven by the primitive waypoint navigation algorithm which is described in Chapter 5.

6.1.1 Aileron from Roll

"Aileron from Roll" loop drives aileron servo actuator according to roll angle error so as to hold the roll attitude of the air-vehicle. This loop is shown in Figure 6.2.

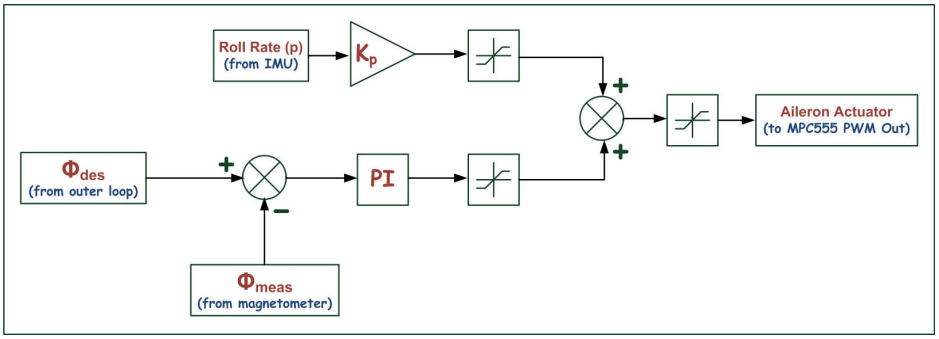

Figure 6.2: Inner Lateral Roll and Roll Rate Controller

The loop shown in Figure 6.2 is implemented in Matlab/Simulink environment as depicted in Figure 6.3.

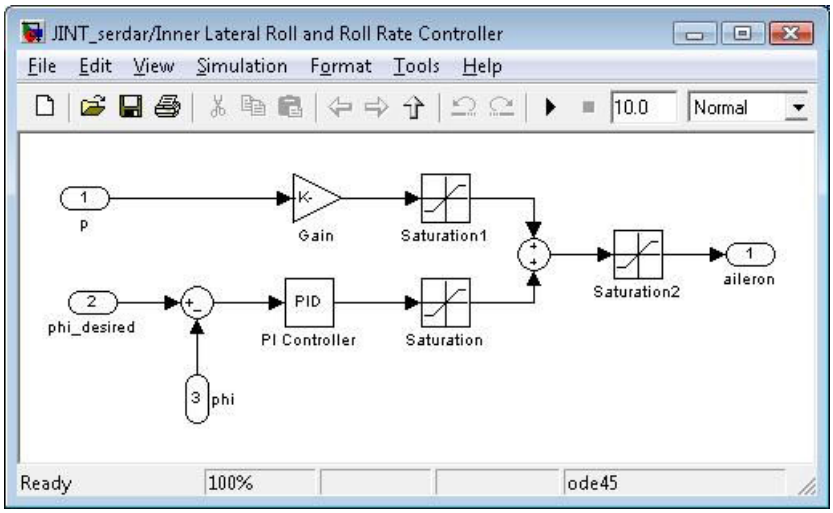

Figure 6.3: Inner Lateral Roll and Roll Rate Controller Simulink Implementation

6.1.2 Aileron from Roll Rate

"Aileron from Roll Rate" loop helps *"Aileron from Roll"* loop drive aileron servo actuator by damping the roll rate of the air-vehicle. A sum block is used to combine the control signals generated by these two inner control loops. This loop is shown in Figure 6.2.

The loop shown in Figure 6.2 is implemented in Matlab/Simulink environment as depicted in Figure 6.3.

6.1.3 Rudder from Yaw Rate

"Rudder from Yaw Rate" loop controls the yaw rate of the air-vehicle by driving the rudder servo actuator. This loop is shown in Figure 6.4.

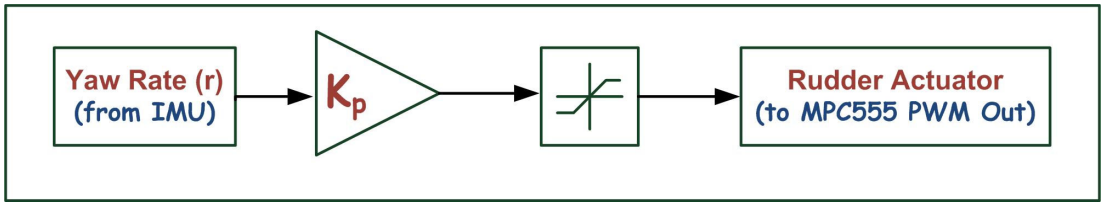

Figure 6.4: Inner Lateral Yaw Rate Controller

The loop shown in Figure 6.4 is implemented in Matlab/Simulink environment as depicted in Figure 6.5.

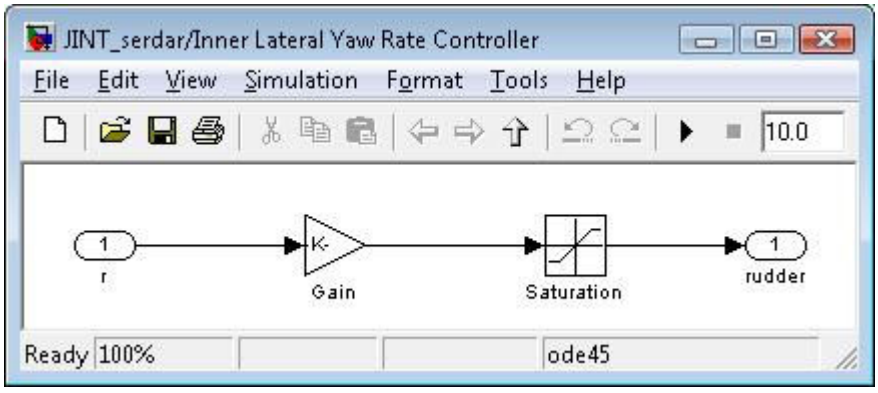

Figure 6.5: Inner Lateral Yaw Rate Controller Simulink Implementation

6.1.4 Roll from Heading

"Roll from Heading" loop controls the heading of the air-vehicle by driving *"Aileron from Roll"* inner control loop according to heading error. This loop is shown in Figure 6.6.

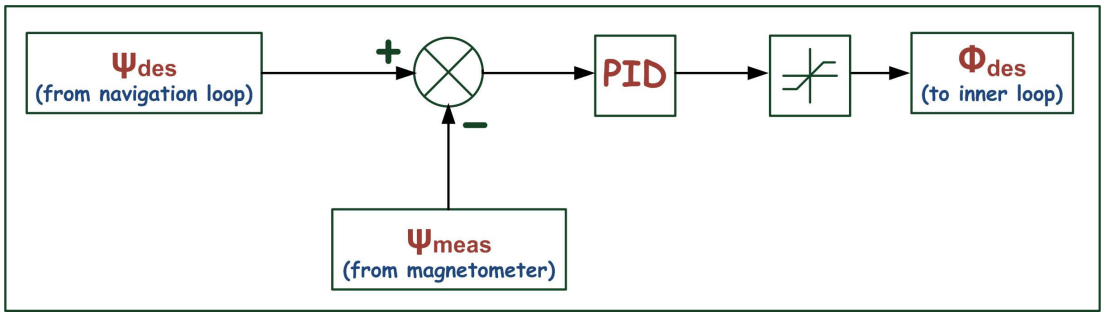

Figure 6.6: Outer Lateral Heading Angle Controller

The loop shown in Figure 6.6 is implemented in Matlab/Simulink environment as depicted in Figure 6.7.

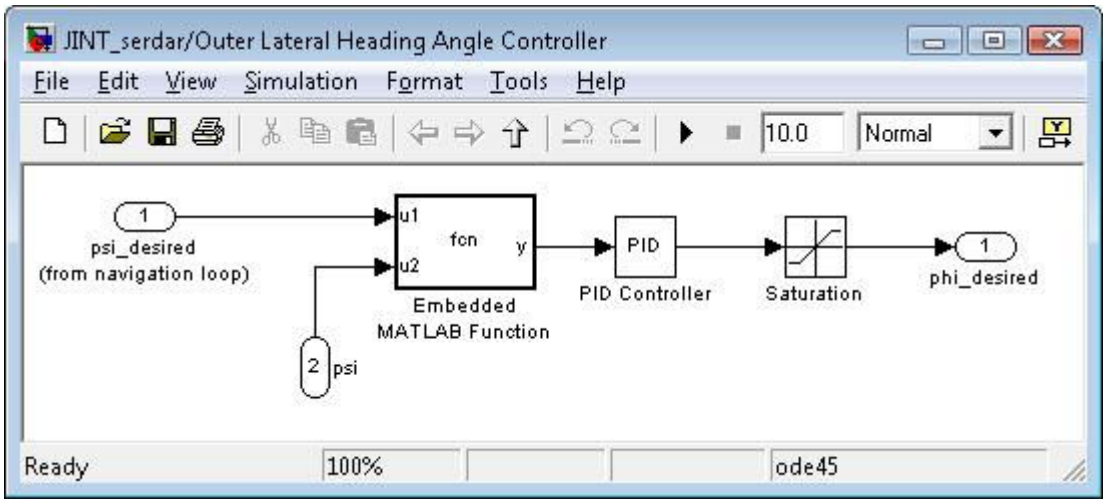

Figure 6.7: Outer Lateral Heading Angle Controller Simulink Implementation

6.2 Longitudinal Control

Velocity, pitch angle, and altitude must be controlled with the help of three inner loops and two outer loops for the air-vehicle lateral control. Inner loops drive the elevator and throttle servo actuators directly. These inner loops are driven by the outer loops. Outer loops basically generate the required reference values for the inner loops. These outer loops are driven by the primitive waypoint navigation algorithm which is described in Chapter 5.

6.2.1 Elevator from Pitch

"Elevator from Pitch" loop drives elevator servo actuator according to pitch angle error so as to hold the pitch attitude of the air-vehicle. This loop is shown in Figure 6.8.

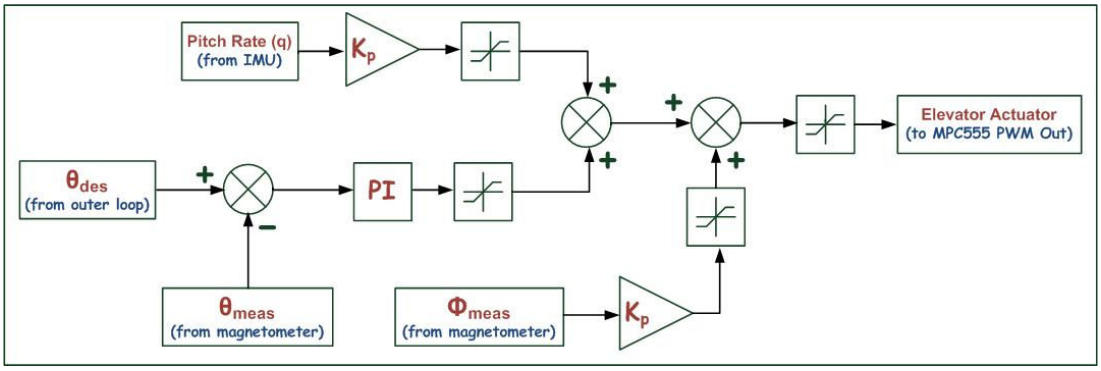

Figure 6.8: Inner Longitudinal Pitch and Pitch Rate Controller

The loop shown in Figure 6.8 is implemented in Matlab/Simulink environment as depicted in Figure 6.9.

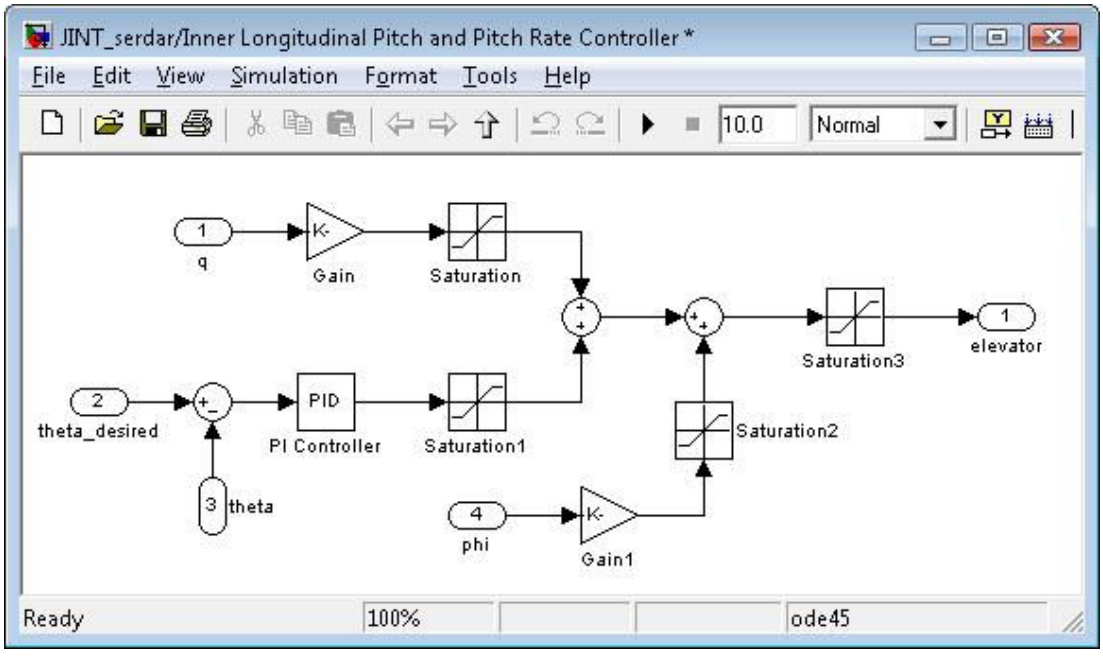

Figure 6.9: Inner Longitudinal Pitch and Pitch Rate Controller Simulink Implementation

6.2.2 Elevator from Pitch Rate

"Elevator from Pitch Rate" loop helps *"Elevator from Pitch"* loop drive elevator servo actuator by damping the pitch rate of the air-vehicle. A sum block is used to combine the control signals generated by these two inner control loops. This loop is shown in Figure 6.8.

The loop shown in Figure 6.8 is implemented in Matlab/Simulink environment as depicted in Figure 6.9.

6.2.3 Throttle from Airspeed

"Throttle from Airspeed" loop controls the airspeed of the air-vehicle by driving throttle servo actuator. This loop is shown in Figure 6.10.

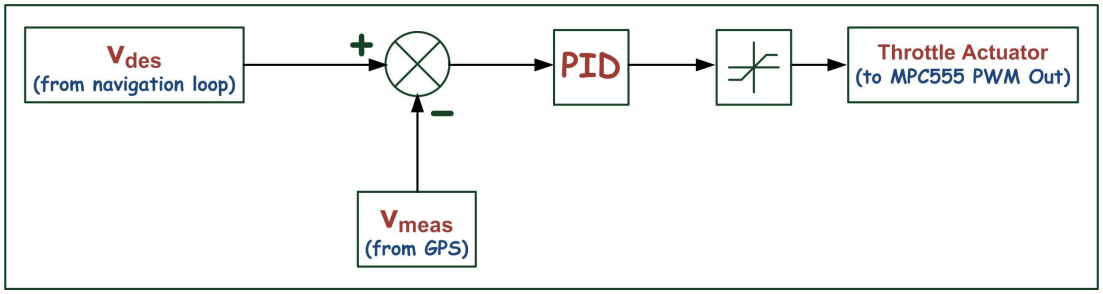

Figure 6.10: Inner Longitudinal Airspeed Controller

The loop shown in Figure 6.10 is implemented in Matlab/Simulink environment as depicted in Figure 6.11.

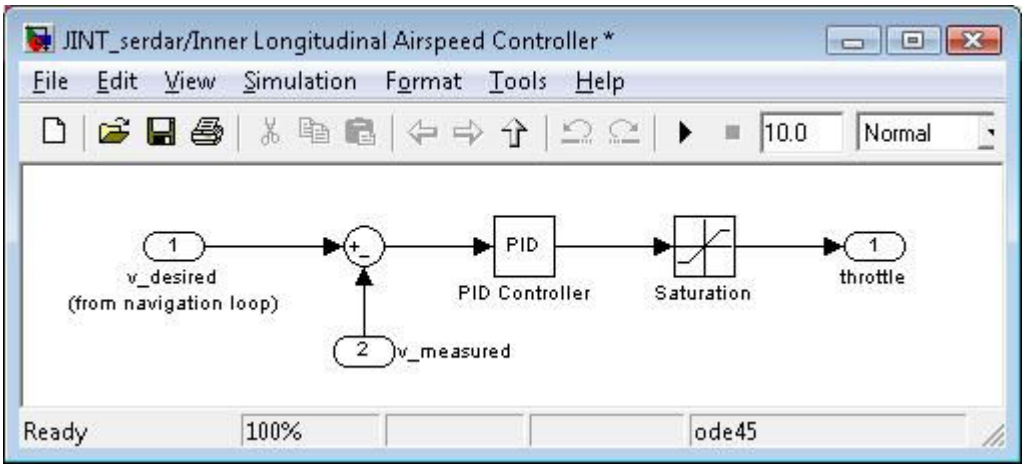

Figure 6.11: Inner Longitudinal Airspeed Controller Simulink Implementation

6.2.4 Pitch from Altitude

"Pitch from Altitude" loop drives *"Elevator from Pitch"* loop according to altitude error so as to hold the altitude of the air-vehicle. This loop should be used while the altitude error is small. Otherwise, *"Pitch from Airspeed"* loop should be used. This loop is shown in Figure 6.12.

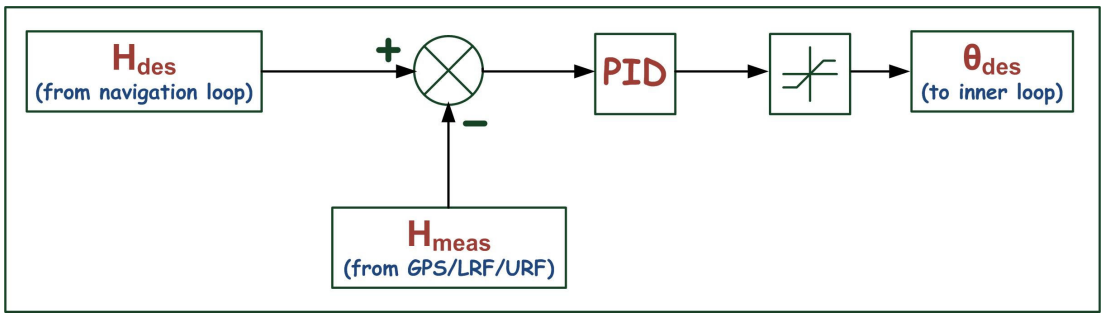

Figure 6.12: Outer Longitudinal Altitude Controller

The loop shown in Figure 6.12 is implemented in Matlab/Simulink environment as depicted in Figure 6.13.

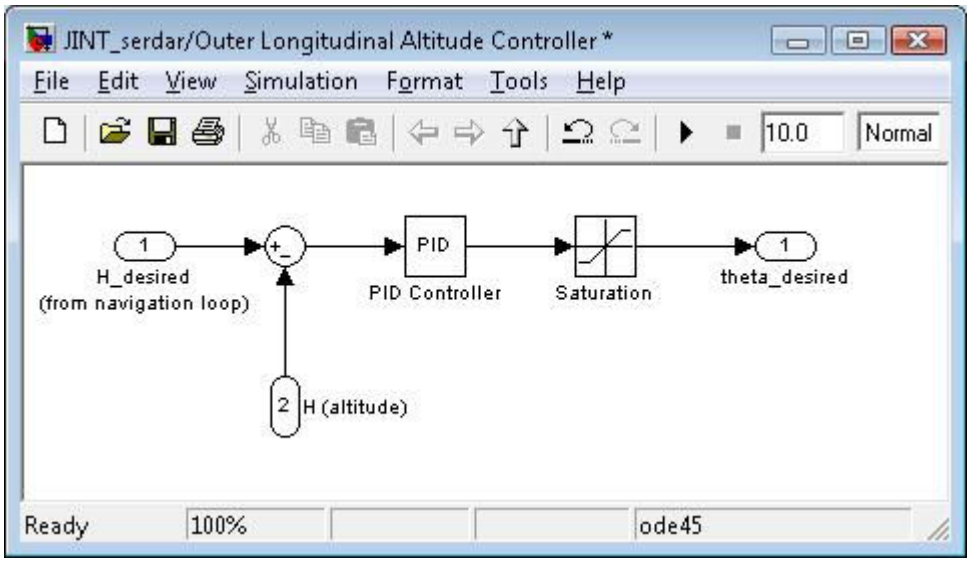

Figure 6.13: Outer Longitudinal Altitude Controller Simulink Implementation

6.2.5 Pitch from Airspeed

"Pitch from Airspeed" loop drives *"Elevator from Pitch"* loop according to airspeed error so as to hold airspeed of the air-vehicle. This loop should be used during climb and descent to hold airspeed of the air-vehicle. This loop is shown in Figure 6.14.

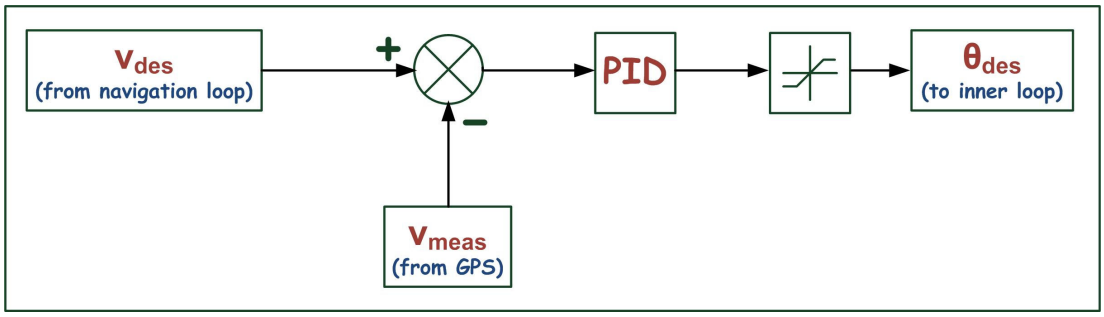

Figure 6.14: Outer Longitudinal Airspeed Controller

The loop shown in Figure 6.14 is implemented in Matlab/Simulink environment as depicted in Figure 6.15.

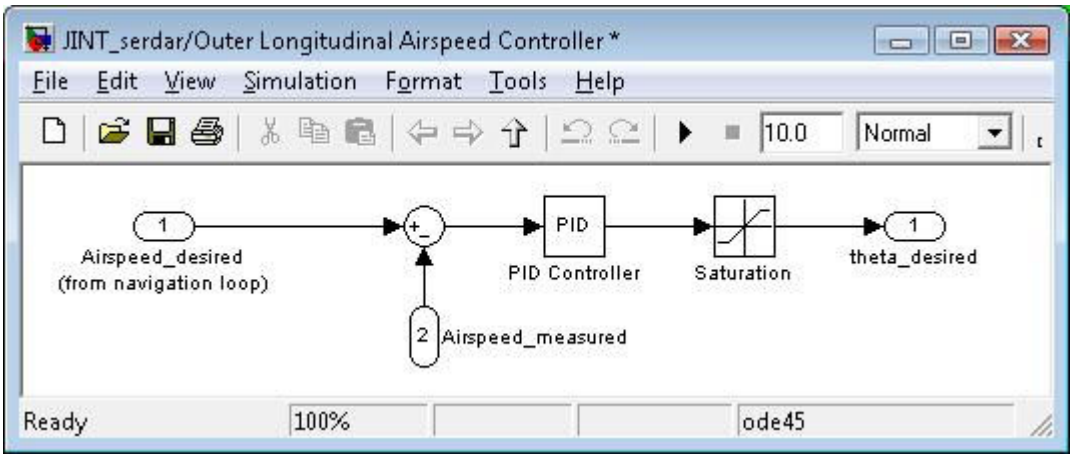

Figure 6.15: Outer Longitudinal Airspeed Controller Simulink Implementation

The loops mentioned in this chapter are realized in Matlab/Simulink environment together as in Figure 8.2.

7. AUTOMATIC TAKE-OFF AND LANDING ALGORITHMS

Autonomous take-off and landing algorithms designed for fixed-wing micro-air vehicles and required hardware including ultrasonic range finder, and laser range finder will be discussed in this chapter. The main references for this chapter are [3-5, 23-26].

7.1 Precise Measurement of Altitude

To enable the air-vehicle to take-off and land autonomously the air-vehicle's altitude which is strongly dependent on the terrain has to be measured very precisely and accurately [27-29]. Thus, an ultrasonic range finder and a laser range finder are being used (shown in Figure 7.1) to measure the altitude of the aircraft very precisely to design fully autonomous algorithms for take-off and landing within in civilian and urban environments. The same sensor set, the same laser range finder and the same ultrasonic range finder for landing are used, verified, and validated in [23, 26, 30].

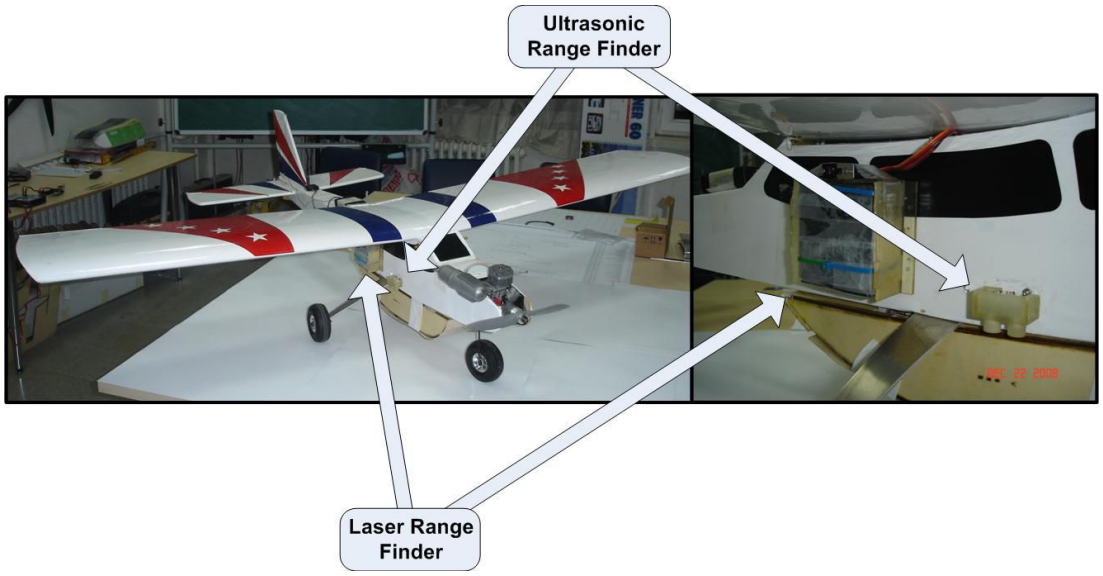

Figure 7.1: Laser Range Finder (Opti-Logic RS400) and Ultrasonic Range Finder (Devantech SRF10) Hardware Mounted on the Aircraft (Trainer60)

The altitude value must be calculated with the help of Equation 7.1 as depicted in [23].

$$h_{real} = h_{measured} \cos\theta \cos\phi \tag{7.1}$$

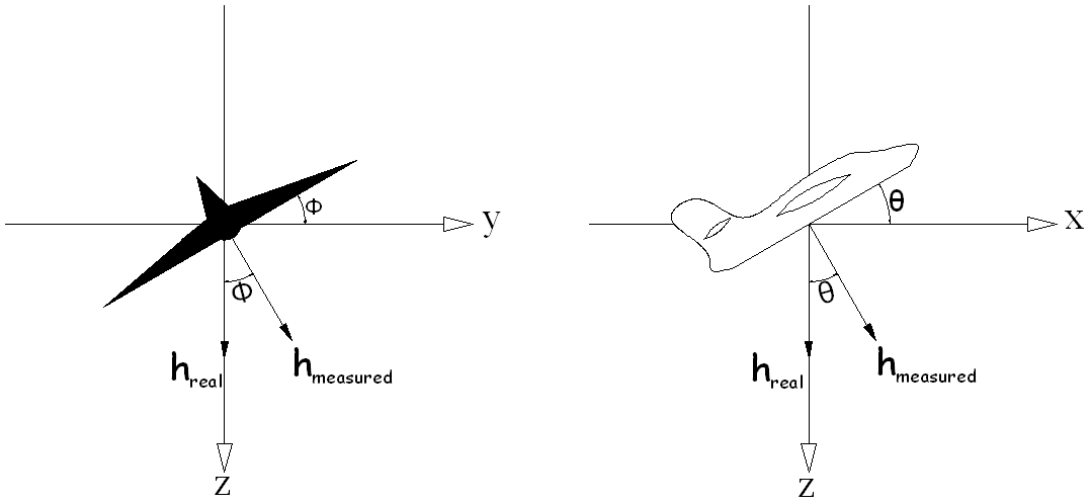

Figure 7.2: Relationship between the Real Altitude and the Measured Altitude

Because of the fact described in Equation 7.1, Figure 7.2, the laser range finder and the ultrasonic range finder may saturate, which causes the spikes on the altitude data. These spikes shown in Figure 7.3 must be filtered by considering the orientation of the air-vehicle.

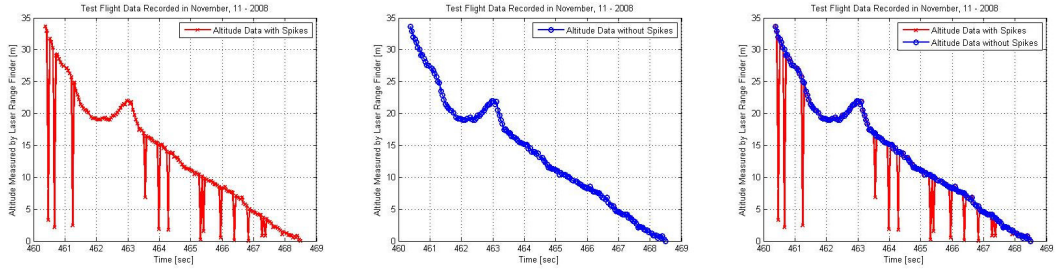

Figure 7.3: Manual Flight Data (Laser Range Finder) Recorded on November, 11 - 2008 During Landing and Spikes Occured

7.2 Autonomous Take-off Algorithm

Autonomous take-off procedure for micro aerial vehicles is defined as in Equation 7.2 in [3, 4].

$$\delta_t = 1 \tag{7.2a}$$

$$\delta_e = \left(k_{p_\theta} + \frac{k_{i_\theta}}{s} \right) \left(\theta^d{}_{take-off} - \theta \right) + k_{d_\theta} q \tag{7.2b}$$

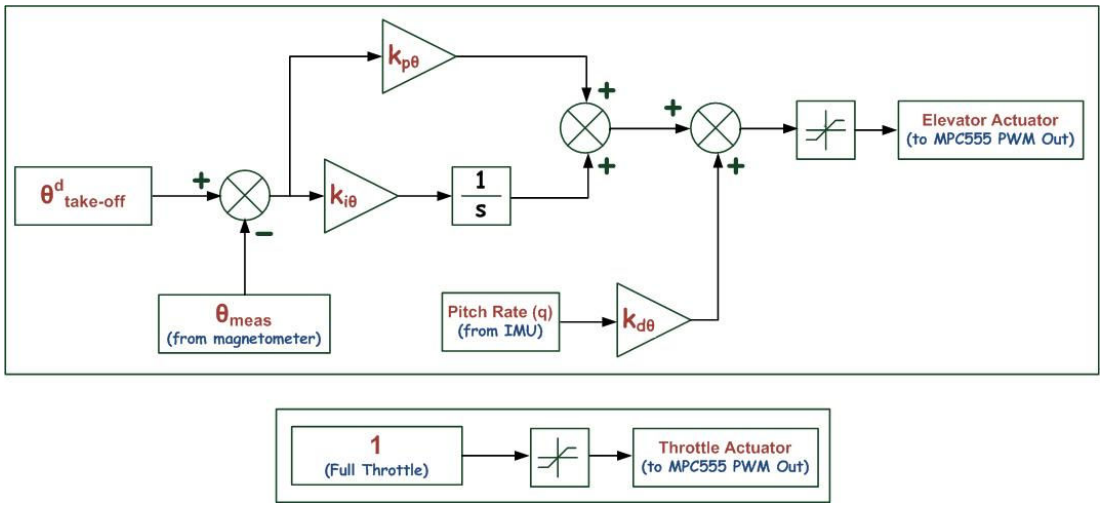

Figure 7.4: Autonomous Take-off Control Law for Micro Aerial Vehicles Defined in [3, 4]

Simulation results of autonomous take-off algorithm combined with autonomous primitive waypoint navigation algorithm which is discussed in Chapter 5 can be seen in Figure 7.5.

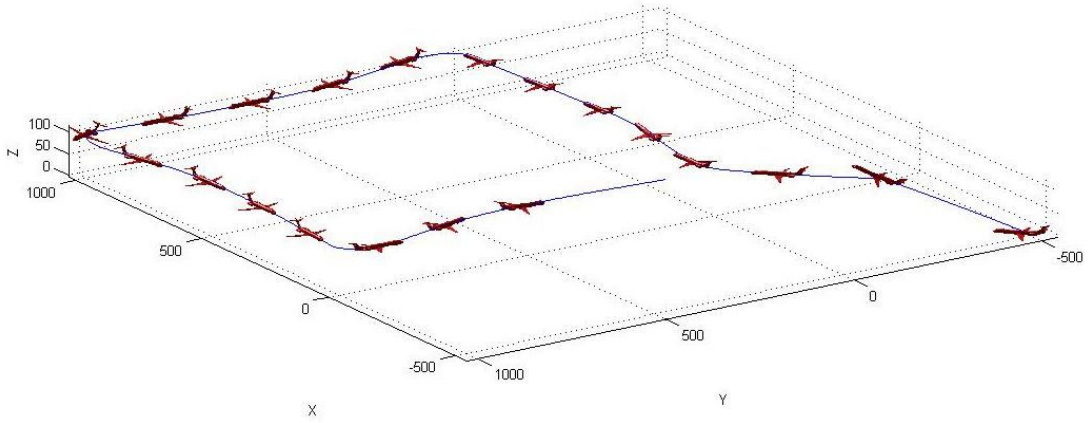

Figure 7.5: Autonomous Take-off Algorithm Simulation Results

Figures like Figure 7.5 are plotted with the help of [31].

7.3 Autonomous Landing Algorithm

Tracking a flare trajectory after gliding a path across the runway is the last phase of landing for fixed-wing aircrafts.

It is stated as *"The automatic flare control system is arranged to provide a flare trajectory. During the flare manoeuvre, the flight path angle of the aircraft has to be changed from -2.5 [degree] to the positive value which is recommended for*

touchdown; in other words, during the flare manoeuvre the control system must control the height of the aircraft's center of gravity and its rate of change such that the resulting trajectory corresponds as nearly as possible to the idealized exponential path." in [5].

This flare trajectory is shown in Figure 7.8. The equation which governs the idealized, exponential flare trajectory is Eqution 7.3 as depicted in [5].

$$h = h_0 e^{\frac{-t}{\tau}} \tag{7.3}$$

where h_0 is the flare entry height, U_0 is the approach speed of the aircraft.

After some arrangements as in [5], Equation 7.4 can be obtained.

$$\dot{h} = -K_a h \tag{7.4}$$

The block diagram of an automatic flare control system is given in Figure 7.6.

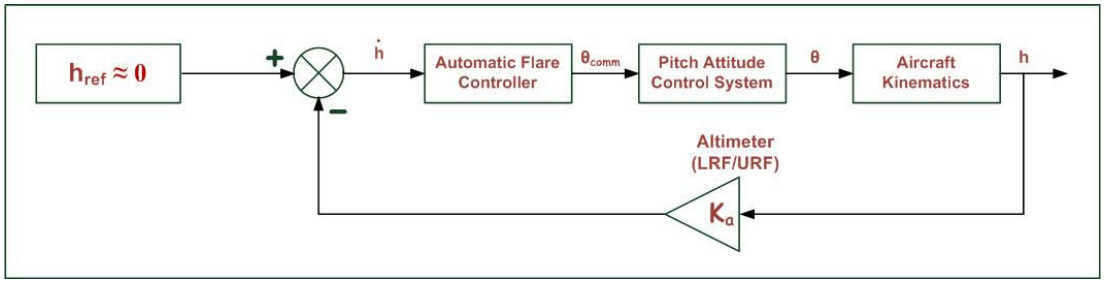

Figure 7.6: Block Diagram of Automatic Flare Control [5]

It can be seen that the pitch attitude control system is used in Figure 7.6. Flight path angle (γ) changes while pitch angle (θ) changes, which causes a change in height. Because of the fact that height measurements are very low, an accurate and precise height measurement is required for the control system. Thus, a laser range finder and an ultrasonic range finder are used.

The control law used is simply given with the help of Equation 7.5 in [5].

$$\theta_{comm} = -k_{c_1}\dot{h} - k_{c_2}\dot{h} \tag{7.5}$$

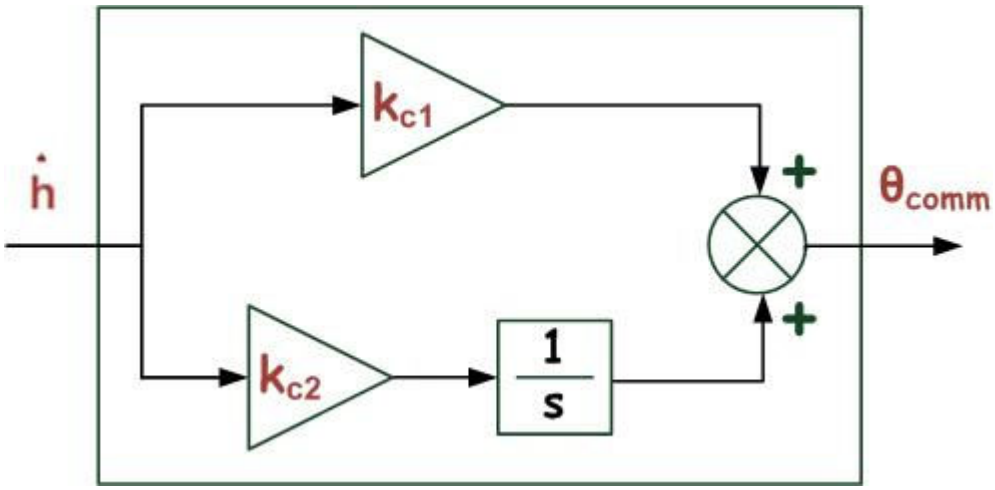

Figure 7.7: Automatic Flare Control Subsystem

Both in take-off and landing algorithms, aileron and rudder inner servo loops defined in Chapter 6 should be used.

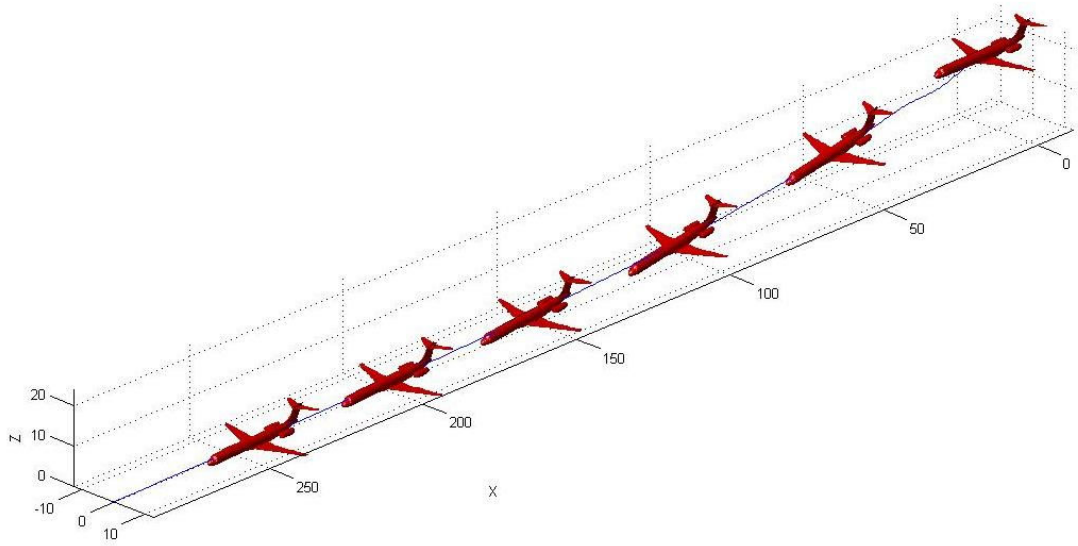

Figure 7.8: Autonomous Landing Algorithm (Automatic Flare Control) Simulation Results

8. HARDWARE-IN-THE-LOOP SIMULATOR

8.1 HIL Testing of the Autopilot System for Trainer 60

The HIL testing capability of an autopilot system for Trainer 60 is one of the most important step to design the multifunctional autopilot algorithm as it is described in generic conceptual design as in Figure 5.1.

HIL Setup in Figure 8.1 illustrates the general distribution structure of all hardware around CAN Bus and Ethernet BUS backbone. HIL platform can be split into four main layers. At the first layer, the sensory information is transmitted to MPC555 via CAN Playback Device or state information via feedback from xPC Target Box. The design of CAN Playback Device was one of the crucial point for realistic HIL testing and it is capable of transmitting sensory information (GPS, Altimeter, HMR, IMU, etc...) over CAN Bus via Softing PC To CAN Interface Card. It is important to note that this sensory information can also belong to real flight data recorded in the data logger (Kvaser Memorator). In addition, the wireless communication of HIL setup with Ground Station is available in this layer. At the second the processor layer, we have an MPC555 processor to different types of control algorithms, and ARM processor to compute coordination patterns, transmit or receive information via XTend RF kit, and PC104 for high level control implementations, mission planning, and task assignment. Third layer is named as dynamical model layer and includes xPC Target Box. Identified and verified aircraft or ground vehicle dynamic models is embedded in xPC Target Box, and receives vehicle control inputs, transmits vehicle states by means of CAN channel. Furthermore, position and attitude information are transmitted via UDP packets from xPC Target. In the fourth layer named as visualization layer, the information is converted to longitude, latitude and Euler data of the vehicle and sent to the network for visualization in FlightGear Flight Simulator. In addition, real vehicle HIL test is achieved as the control inputs can be sent to the aircraft servos as PWM signals.

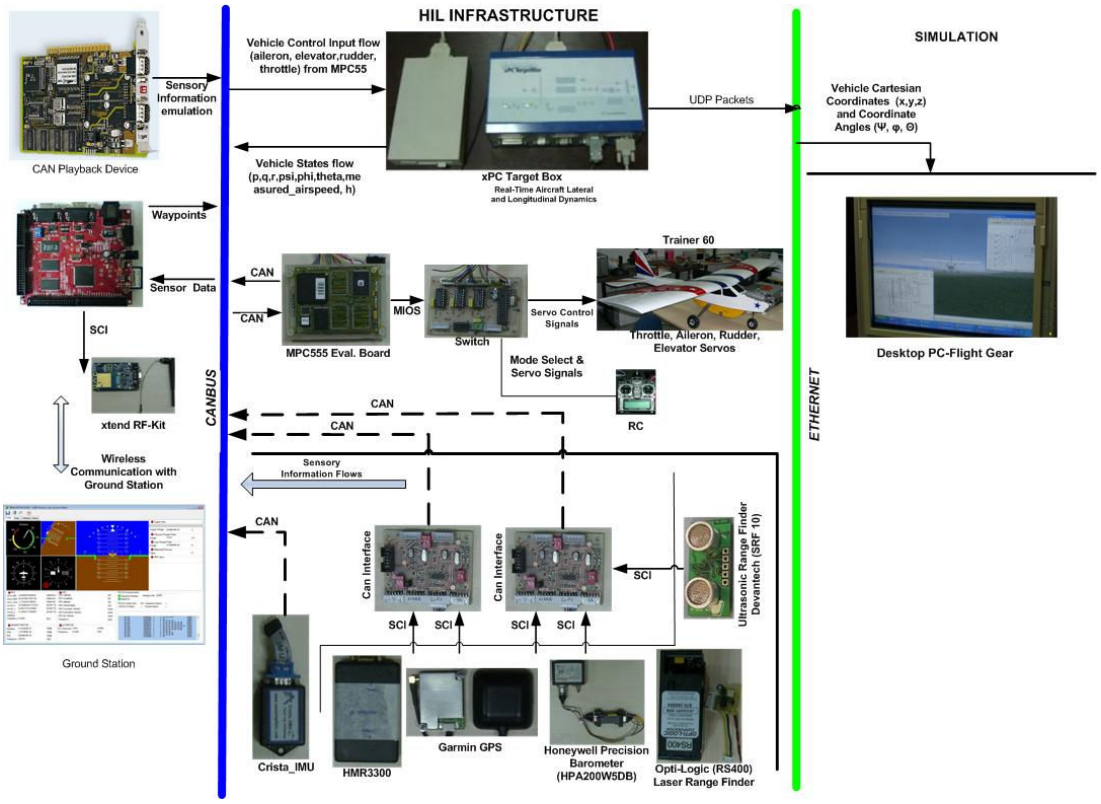

Figure 8.1: Hardware-in-the-Loop Simulator Design Structure

The Hardware-in-the-Loop test of the autopilot system is realized in two different ways. The first method is named as "Option-A:Playback Mode", which is illustrated in Figure 8.4. All the sensory information flow is from CAN Playback Device into the MPC555 over CAN Bus. MPC555 uses the data as the sensory information during real flights and generate PWM signals as the control signals of Trainer 60 model aircraft. As a result, the real Trainer 60 is animated our real ex-flights over the HIL testbed or xPC Target Box converts the sensory information to vehicle coordinates and coordinate angles to visualize the recorded flight via FlightGear. Second of all, "Option-B:Simulated Mode" is used for testing the autopilot algorithm and demonstrate its performance via FlightGear Flight Simulator as depicted in Figure 8.5. In the second mode, vehicle states flow over CAN Bus to the MPC555 as the sensory information and MPC555 generates control inputs. Then control signals are transmitted to the xPC Target via CAN messages. xPC Target sends UDP packets to visualize our autonomous way point algorithm afterwards. During the test of our Autopilot Algorithm, the verified dynamic model is embedded in xPC Target and control loops are embedded in MPC555 processor.

Autonomous waypoint navigation algorithm for Trainer 60 was tested via HIL system and split into three crucial layers as in Figure 8.2. In the core layer, there are inner loops to generate appropriate PWM signals for the maneuvers of Trainer 60. The core layer is directly connected to a second layer, which includes outer loops such as Altitude Hold, Airspeed Hold and Heading Hold (also defined as Coordinated Turn) [22] as shown in Figure 8.3 and generate the references for the inner loops. The third layer is a navigation layer and generate the altitude, velocity and heading of the aircraft according to our waypoint map as it is described in Chapter 5 for Humvee waypoint navigation algorithm in detail. In addition, navigation layer needs the state information of the aircraft which is provided from our Sensor Suite. Moreover, the piloted take off and the test of the autopilot algorithm during way point navigation is illustrated in Figure 8.2.

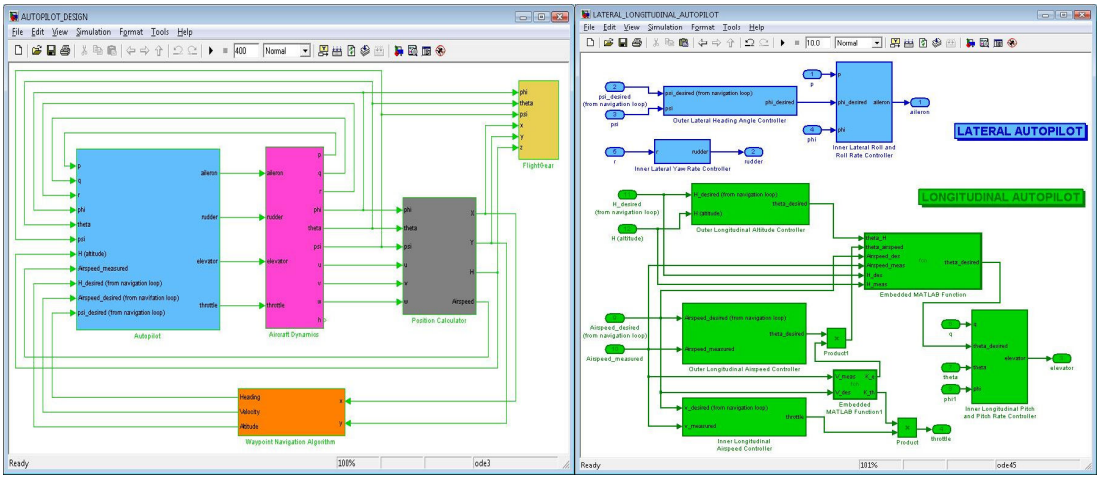

Figure 8.2: Matlab/Simulink Simulation of UAV Autopilot

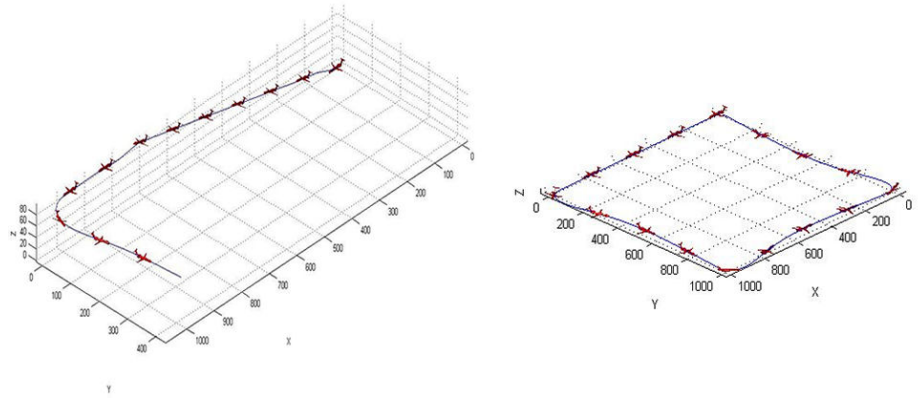

Figure 8.3: Matlab/Simulink Simulation Results of Micro-UAV Autopilot (Take-off by Human Pilot and Waypoint Navigation by Auto-pilot)

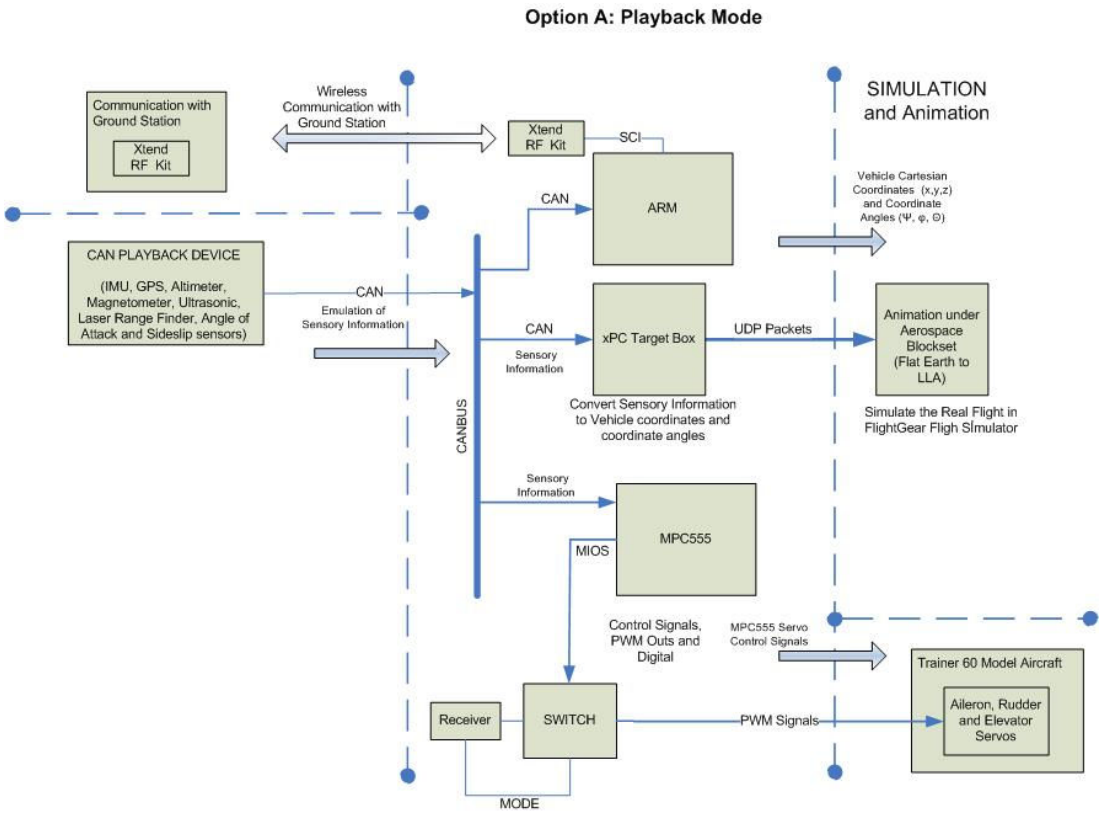

Figure 8.4: Hardware-in-the-Loop Simulator Design Structure - Option A

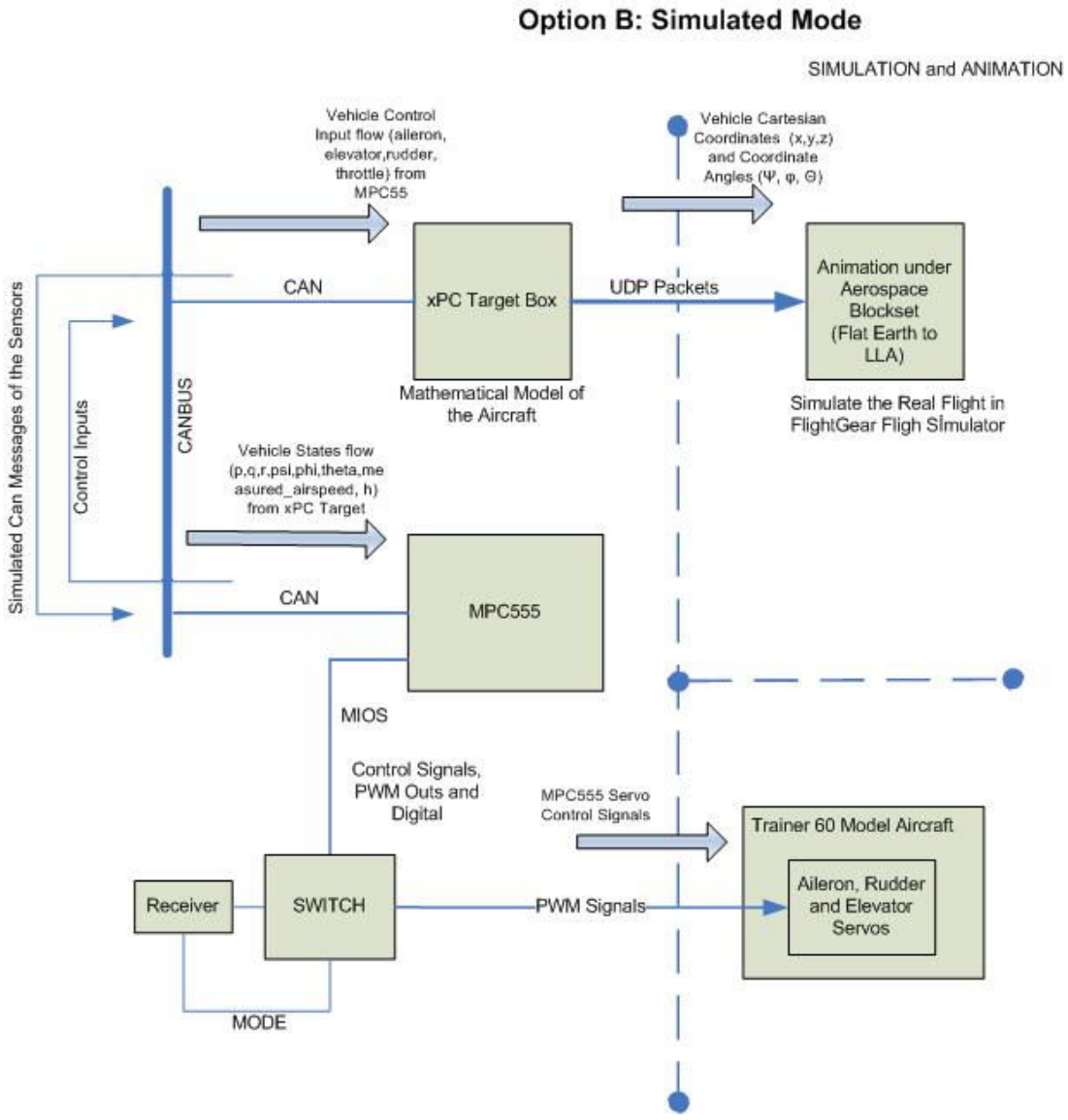

Figure 8.5: Hardware-in-the-Loop Simulator Design Structure - Option B

9. CONCLUSION

In this work, the design and the autonomous operation implementation details of a CAN and Ethernet bus back-boned microavionics system is presented. Integration tests of this microavionics hardware and software have been achieved.

In addition to mimicking the avionics structure of larger scale units used at high speed/high altitude UAVs, this system provides us with unique features such as rapidprototyping/automatic code generation capability and cross-platform-compatibility to ground, fixed-wing and rotary mini unmanned vehicles. Thus, ground vehicle simulations and outside tests have been done.

The bus-backbone also allows integration and testing of customized flight management computers and communication units that implement algorithms tailored towards fleet operations in joint manned-unmanned airspace. HIL integration of this unit provides an extensive testing capability of these units and algorithms before the risky and costly real flight testing that involves manned and unmanned vehicles.

Autopilot control and navigation algorithms as an enabling technology for a fixed-wing aircraft are designed and modified so as to fit our microavionics applications. Besides, automatic take-off and landing algorithms for fixed-wing aircrafts are examined, adapted and required hardware including a laser range finder and a ultrasonic range finder is discussed. HIL integration tests of autopilot control and navigation loops are achieved.

As future works, detailed system identification of the aerial platforms can be done to obtain more realistic mathematical model of them. Thus, controller design and implementation will be enhanced. Different filtering techniques such as Kalman Filter can be applied to the microavionics system to achieve better sensing and controlling performance. Different control algorithms such as LQR, fuzzy and other robust techniques can be applied to the microavionics system in an elegant way of rapid prototyping/automatic code generation capability of the microavionics system design to get more precise controlling results. Finally, considering the structure of

the microavionics system design, multi-agent applications where each node carries this microavionics system can be achieved.

REFERENCES

[1] **Etkin, B.**, 1990. Dynamics of Flight – Stability and Control, New York John Wiley and Sons, Inc., London Chapman and Hall Ltd.

[2] **Okatan, A., Dimirovski, G. M.**, July 9 – 12, 2002. Fuzzy Logic Navigation and Control of a Non – holonomic Vacuum Cleaner, Proceedings of the 10th Mediterranean Conference on Control and Automation – MED2002, Lisbon, Portugal.

[3] **Beard, R. W.**, Guidance and Control of Autonomous Miniature Air Vehicles.

[4] **Barber, D. B., Griffiths, S. R., McLain, T. W., Beard, R. W.** Autonomous Landing of Miniature Aerial Vehicles, *AIAA*.

[5] **McLean, D.**, 1990. Automatic Flight Control Systems, Prentice Hall.

[6] **Rönnback, S.**, 2000. Development of a INS/GPS Navigation Loop for an UAV, M.Sc. Thesis, Lulea Tekniska Universitet.

[7] **Gavrilets, V., Shterenberg, A., Martinosi I., Sprague, K., Dahleh, M. A., Feron, E.**, 2001. Avionics System for Aggressive Maneuvers, pp.38 – 43 vol.16.

[8] **Jang, J. S.**, 2003. Nonlinear Control Using Discrete – Time Dynamic Inversion Under Input Saturation: Theory and Experiment on the Stanford Dragonfly UAVs, Ph.D. Thesis, Stanford University.

[9] **Johnson, E. N., Schrage, D. P.**, 2003. The Georgia Tech Unmanned Aerial Research Vehicle: GTMax, Proceedings of the AIAA Guidance, Navigation, and Control Conference, Austin.

[10] **Elston, B. J., Frew, E.**, 2005. A Distributed Avionics Package for Small UAVs, AIAA Infotech at Aerospace, Arlington, VA.

[11] **Karaman, S., Aksugur, M., Baltaci, T., Bronz, M., Kurtulus, C., Inalhan, G., Altug, E. and Guvenc, L.**, 2007. Aricopter: Aerobotic Platform for Advances in Flight, Vision Controls and Distributed Autonomy, IEEE Intelligent Vehicles Symposium, Istanbul, Turkey.

[12] **Aksugur, M., Inalhan, G., Beard, R.**, 2008. Hybrid Propulsion System Design of a VTOL Tailsitter UAV, Wichita Aviation Technology Conference, Wichita, Kansas, USA.

[13] **Koyuncu, E., Ure, N. K., Inalhan, G.**, 2008. A Probabilistic Algorithm for Mode Based Motion Planning of Agile Air Vehicles in Complex Environments, International Federation of Automatic Control World Congress, Seul.

[14] **Shim, D. H., Sastry, S.**, 2006. A Situation – aware Flight Control System Design using Real – time Model Predictive Control for Unmanned Autonomous Helicopters, pp.38 – 43 vol.16

[15] **Evans, J., Inalhan, G., Jang, J. S., Teo, R., Tomlin C. J.**, 2001. Dragonfly: A Versatile UAV Platform for the Advancement of Aircraft Navigation and Control.

[16] **Inalhan, G.**, 2007. Flight Stability and Control Lecture Notes, Istanbul Technical University, Faculty of Aeronautics and Astronautics.

[17] **Nelson, R. C.**, 1998. Flight Stability and Automatic Control, McGraw – Hill International Editions, Aerospace Science and Technology Series.

[18] **Sorensen, B. V., Nielsen, C. K., Vilhelmsen, J. B., Persson, R. B., Hansen, S. H., Jensen, S. K.**, 2005. Autonomous Model Airplane, Technical Report 05gr830-student-report, Aalbord University, Institute of Electronic Systems, Department of Control Engineering.

[19] **Stevens, B. L., Lewis, F. L.**, 1992. Aircraft Control and Simulation, New York – Wiley.

[20] **Vural, S. Y.**, 2008. Küçük Bir İnsansız Hava Aracı için Otopilot Sistemi Tasarımı, M.Sc. Thesis, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü.

[21] **Luca, A. D., Oriolo, G., Samson, C.** Robot Motion Planning and Control – Chapter: Feedback Control of a Nonholonomic Car – like Robot, LAAS report 97438.

[22] **Christiansen, R. S.**, 2004. Design of an Autopilot for Small Unmanned Aerial Vehicles, M.Sc. Thesis, Brigham Young University.

[23] **Chan, W. L.**, 2006. Computer Aided Landing Control System Design for a Fixed Wing UAV, M.Sc. Thesis, National Cheng Kung University, Taiwan.

[24] **Riseborough, P.** Automatic Take – off and Landing Control for Small UAVs.

[25] **Littleton, C. A.**, October 2005, Autonomous Unmanned Aerial Surveillance Vehicle – Autonomous Control and Flight Dynamics.

[26] **Griffiths, S., Saunders, J., Curtis, A., McLain, T., Beard, R.** Obstacle and Terrain Avoidance for Miniature Aerial Vehicles.

[27] **Ateş, S., Bayezit, I., Gündüz, F. E., Aksugür, M. K., Armağan, B., İnalhan, G.**, 2008. Design and Hardware – in – the – Loop Integration of a UAV Microavionics System in a Manned – Unmanned Joint Airspace Flight Network Simulator.

[28] **Ateş, S., Bayezit, I., İnalhan, G.**, December, 2008. Design and Hardware – in – the – Loop Integration of a UAV Microavionics System in a Manned – Unmanned Joint Airspace Flight Network Simulator, *Journal of Intelligent and Robotic Systems (JINT)*, **53.**

[29] **Ateş, S., Bayezit, I., İnalhan, G.**, 2009. Unmanned Aircraft Systems – Book Chapter: Design and Hardware – in – the – Loop Integration of a UAV Microavionics System in a Manned – Unmanned Joint Airspace Flight Network Simulator, Springer.

[30] **Schafroth, D., Bouabdallah, S., Bermes, C., Siegwart, R.**, 2009. Unmanned Aircraft Systems – Book Chapter: From the Test Benches to the First Prototype of the muFly Micro Helicopter, Springer.

[31] **Scordamaglia, V.**, 09 Aug 2004 (Updated 17 Jan 2006), Trajectory and Attitude Plot Version 3.

APPENDICES

A. COORDINATE TRANSFORMATIONS

GPS receiver sensor gives the sensory information in geodetic frame using WGS-84 datum as latitude, longitude and altitude. This sensory information must be transformed into the local ENU/NED frame. Hence, the required formulations are taken from [6] and summarized.

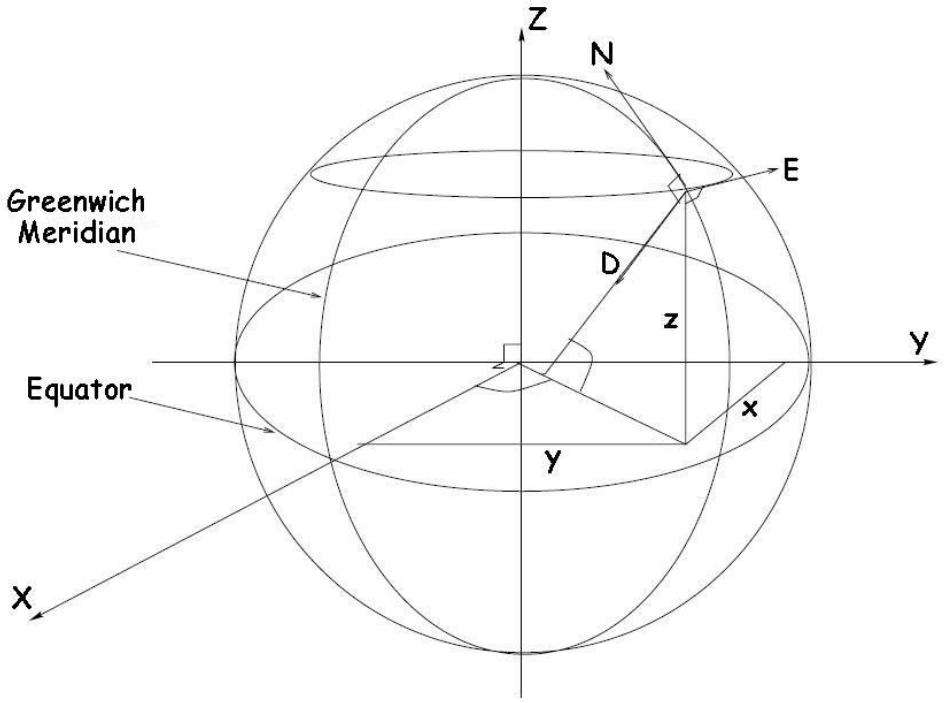

Figure A.1 : Earth NED Frame [6]

Earth NED Frame is shown in Figure A.1. [x, y, z] are ECEF coordinates, [X, Y, Z] are ECEF axes, [N, E, D] are NED frame axes, λ is longitude and Φ is latitude.

Table A.1 : WGS – 84 Coordinate Transformation Variables

Variable	Description
Λ	Longitude
Φ	Latitude
R_N	Local Earth radius
h	Height above sea level
a	Earth semi major axis, Earth radius at equator a=6378137
b	Earth semi minor axis, radius at poles b=a(1-f)
f	Earth flattering f=0.003352810664747

A.1 Geodetic to ECEF Transformation

The local Earth radius is a function of latitude because of the flattering of the Earth. It is assumed that the Earth is almost ellipse and the local Earth radius can be calculated with the help of Equation A.1.

$$R_N(\Phi) = \frac{a^2}{\sqrt{a^2 \cos^2 \Phi + b^2 \sin^2 \Phi}} \tag{A.1}$$

The cartesian coordinates can be calculated with the help of Equation A.2.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{ECEF} = \begin{bmatrix} (R_N + h)\cos\Phi\cos\lambda \\ (R_N + h)\cos\Phi\sin\lambda \\ (\frac{a^2}{b^2} R_N + h)\sin\Phi \end{bmatrix} \tag{A.2}$$

A.2 ECEF to NED Transformation

The calculated positions have to be transformed into the navigation frame. A local tangent plane is defined using Equation A.3, A.4 and A.5.

$$C(\lambda,\Phi) = \begin{bmatrix} -\sin\Phi\cos\lambda & -\sin\Phi\sin\lambda & \cos\Phi \\ -\sin\lambda & \cos\lambda & 0 \\ -\cos\Phi\cos\lambda & -\cos\Phi\sin\lambda & -\sin\Phi \end{bmatrix} \tag{A.3}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{NED} = C(\lambda,\Phi) \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{ECEF} \tag{A.4}$$

$${}_{ENU}C_{NED} = {}_{NED}C_{ENU} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{A.5}$$

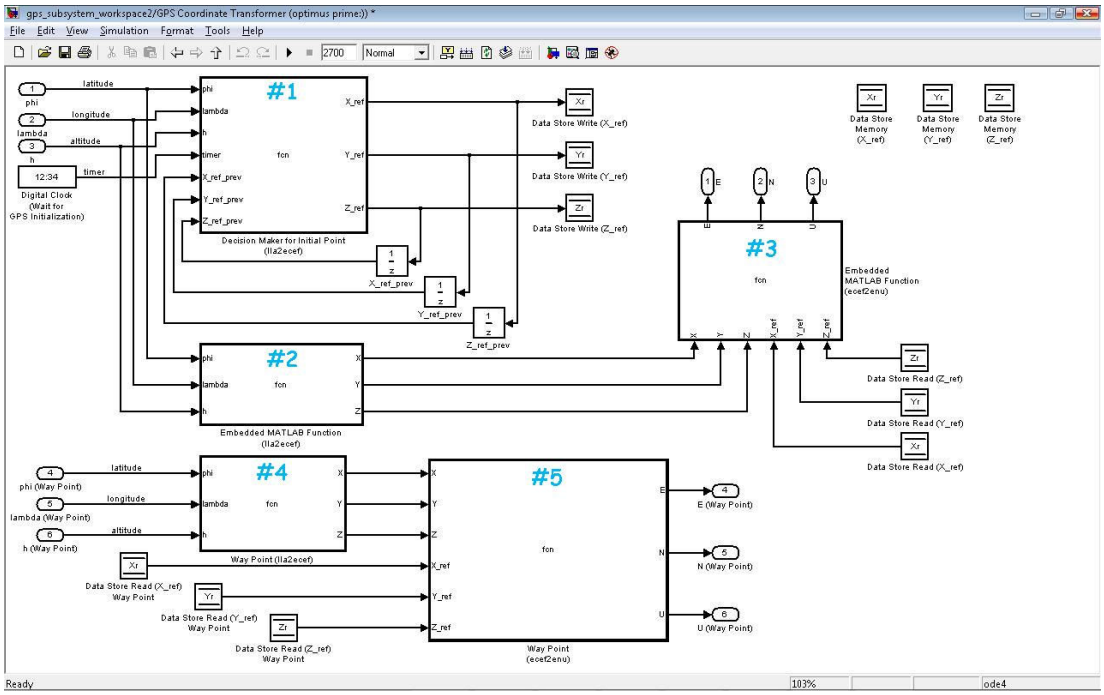

Figure A.2 : Matlab/Simulink Subsystem for Coordinate Transformation

The subcomponents of the red block in Figure 5.9 can be seen in Figure A.2. In Figure A.2, the block numbered #1 converts the geodetic coordinates into ECEF frame for reference point, the blocks numbered #2 and #3 convert the geodetic coordinates into ENU frame so as to use the GPS receiver sensor in the local tangent plane, the blocks numbered #4 and #5 convert the waypoints' geodetic coordinates into the same ENU frame.

B. GPS SURVEY

GPS receiver sensor was tested in several times with the help of GPS surveys before designing autonomous algorithms which are strongly dependent on this sensor. The football pitch of İstanbul Technical University was used as GPS survey area because of the fact it has a very geometric structure and landmarks such as penalty area spots, corner spots, center spot, etc... After a few GPS surveys, the coordinates of the penalty area spots are determined and these points are marked as waypoints to the navigation algorithm. The autonomous waypoint navigation algorithms' results are confirmed in Figure B.5.

GPS survey procedure involves selecting a point, powering-up GPS receiver sensor on that point, collecting sufficient number of data, and analyzing the collected data by using formulations mentioned in Appendix A with the method mentioned in Appendix C.

The following GPS survey was realized in December 1 - 2007 on Saturday.

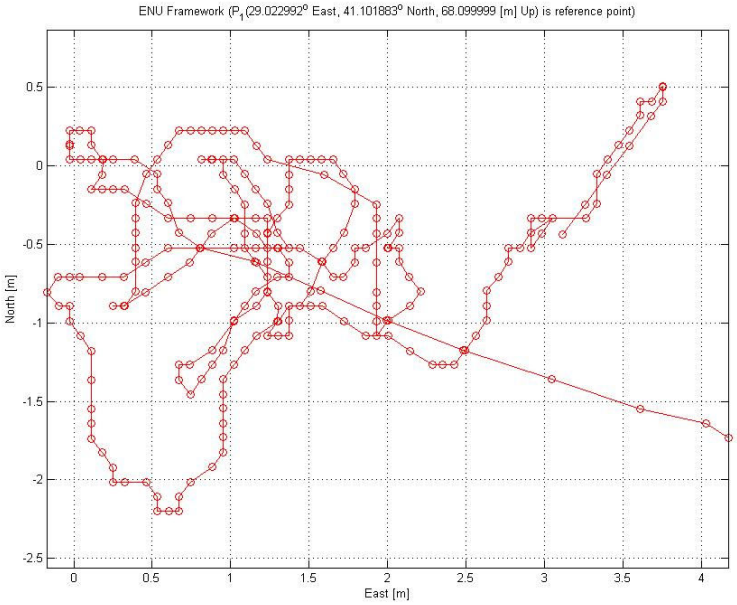

Figure B.1: ENU Coordinates of P_1

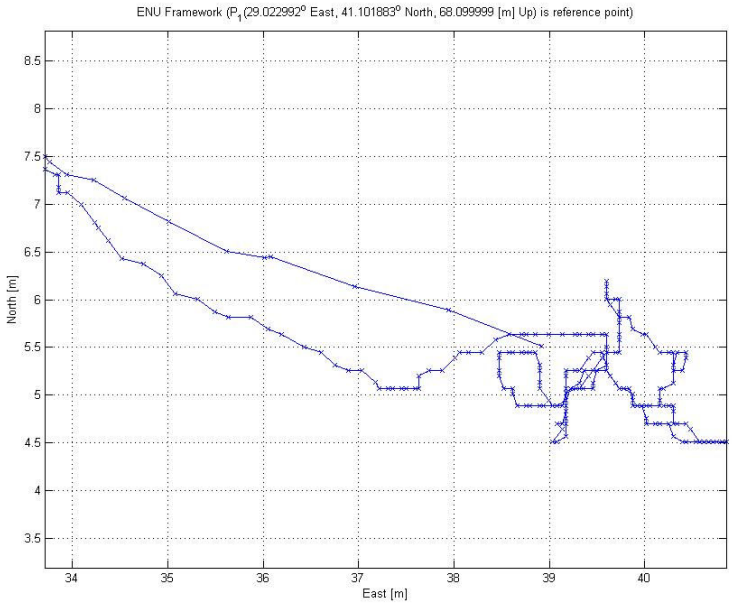

Figure B.2: ENU Coordinates of P_2

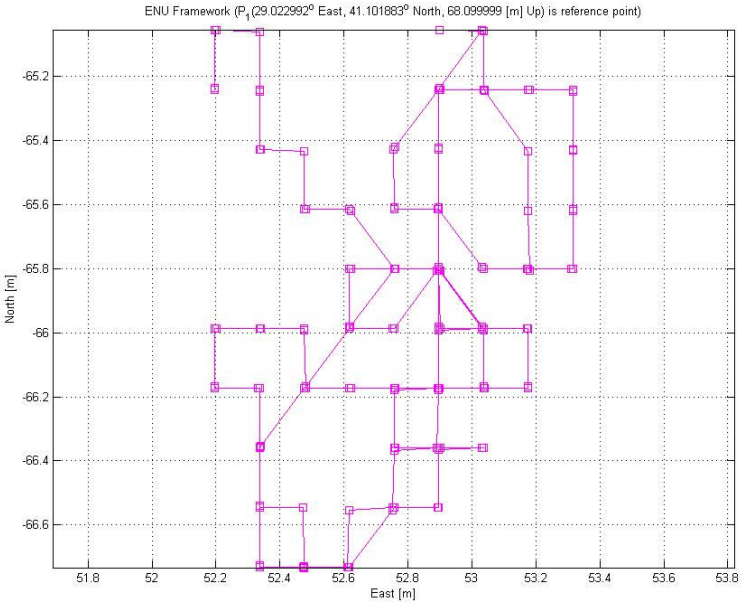

Figure B.3: ENU Coordinates of P_3

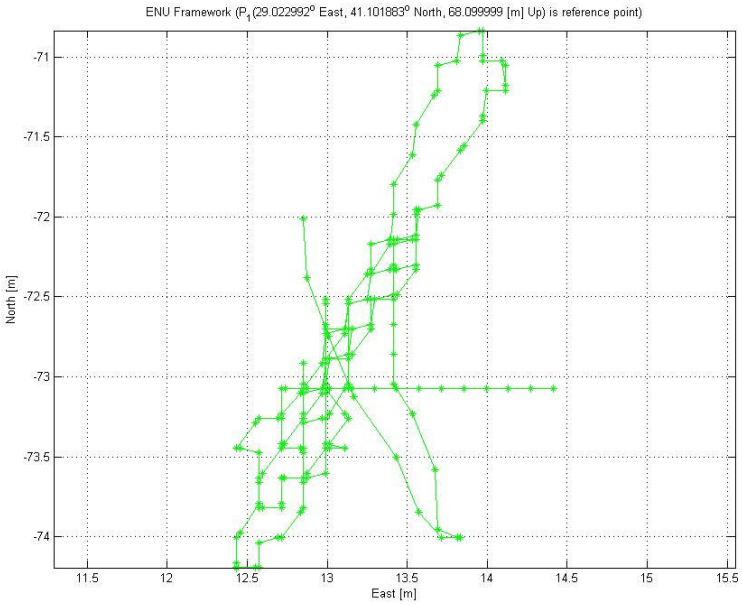

Figure B.4: ENU Coordinates of P_4

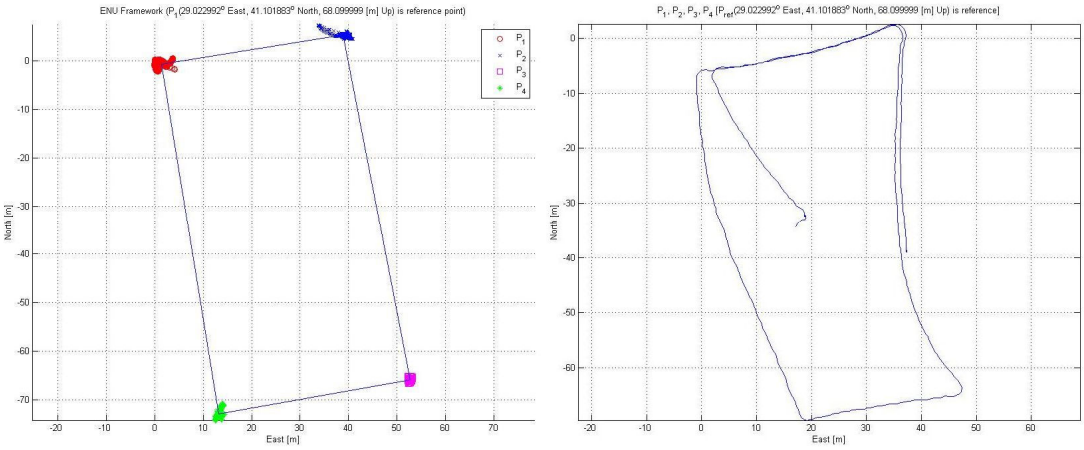

Figure B.5: ENU Coordinates of P_1, P_2, P_3, P_4, and Waypoint Navigation Algorithm Results Based on These Waypoints

Table B.1: Mean and Standard Deviation Values of P_1, P_2, P_3, P_4

Point	Latitude [deg]	Longitude [deg]
P_{1mean}	41.101877	29.023009
P_{1std}	5.794256e-006	1.28412e-005
P_{2mean}	41.101930	29.023457
P_{2std}	5.910631e-006	2.369873e-005
P_{3mean}	41.101293	29.023623
P_{3std}	4.362832e-006	2.087000e-005
P_{4mean}	41.101232	29.023148
P_{4std}	7.827470e-006	4.998457e-006

Another GPS survey realized at İstanbul Hezarfen Airport in March - 4, 2008 can be seen in Figure B.6. In this GPS survey, microavionics hardware including GPS receiver sensor is used to detect the edges of the RC pitch by walking on the edges with the hardware.

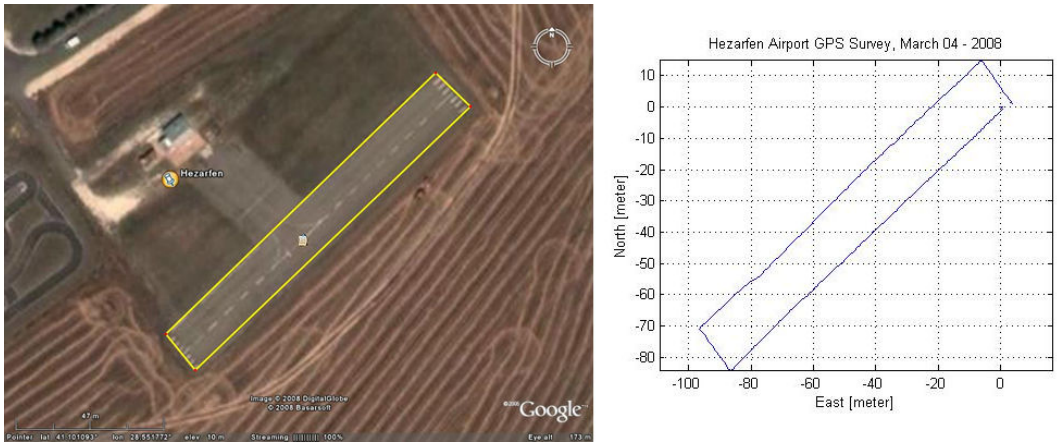

Figure B.6: Another GPS Survey Realized at İstanbul Hezarfen Airport on March - 1, 2007

C. DATA LOGGING AND ANALYSIS PROCESS

Data logging and analysis process can be seen in Figure C.1. There are two different methods to analyze the logged data.

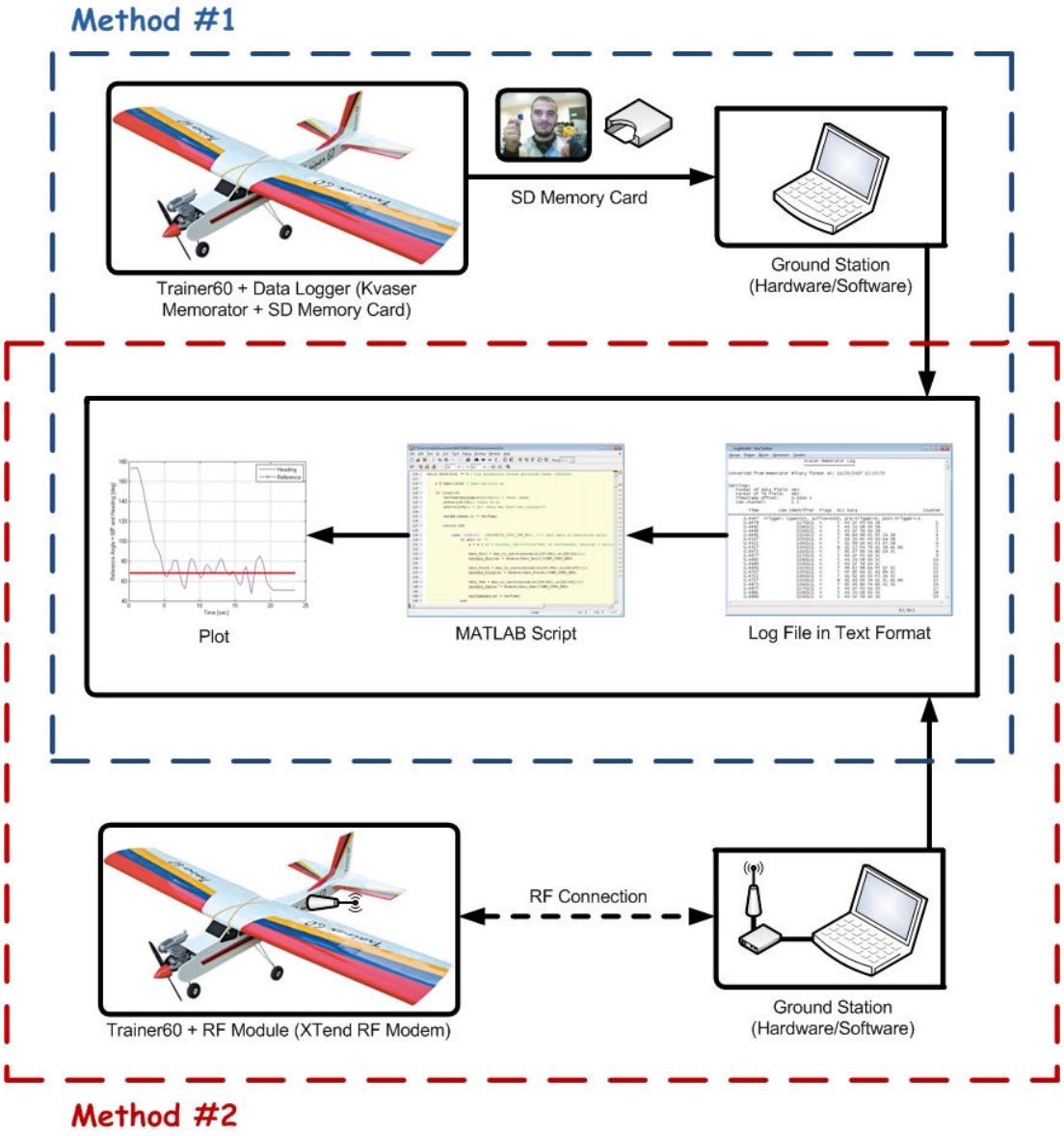

Figure C.1: Data Logging and Analysis Process

The first one, Method #1, records the flight data by a data logger (Kvaser Memorator) on a SD memory card. After finishing the flight test, SD memory card involving the flight data is unmounted from the data logger and mounted to the ground station hardware, a laptop PC, manually by a human operator. Then, recorded flight data is saved as a text file and this text file is processed to parse by a MATLAB script. After processing by the MATLAB script, each sensory data of

flight data is saved on MATLAB workspace as double vectors, one vector for data and one vector for time stamps. Finally, data recorded on MATLAB workspace is handled by using primitive MATLAB commands such as plot.

The second one, Method #2, sends the flight data in real time to the ground station via a RF module couple, XTend RF modems. The ground station GUI software records the flight data as the data logger on the Trainer60 does on the ground station hardware automatically. Then, recorded flight data is saved as a text file as in Method #1 and this text file is processed to parse by a MATLAB script. After processing by the MATLAB script, each sensory data of flight data is saved on MATLAB workspace as double vectors, one vector for data and one vector for time stamps. Finally, data recorded on MATLAB workspace is handled by using primitive MATLAB commands such as plot.

CIRRICULUM VITAE

Serdar Ateş was born in Edirne in 1982. He received his B.Sc. degree in Electrical Engineering from Yıldız Technical University with honor degree in 2006. He attended Mechatronics Engineering Master Programme at İstanbul Technical University in 2006 and he has been working as a research assistant at Controls and Avionics Laboratory at İstanbul Technical University since 2007. He was also a graduate scholar of TÜBİTAK (Turkey Science and Technology Research Foundation).