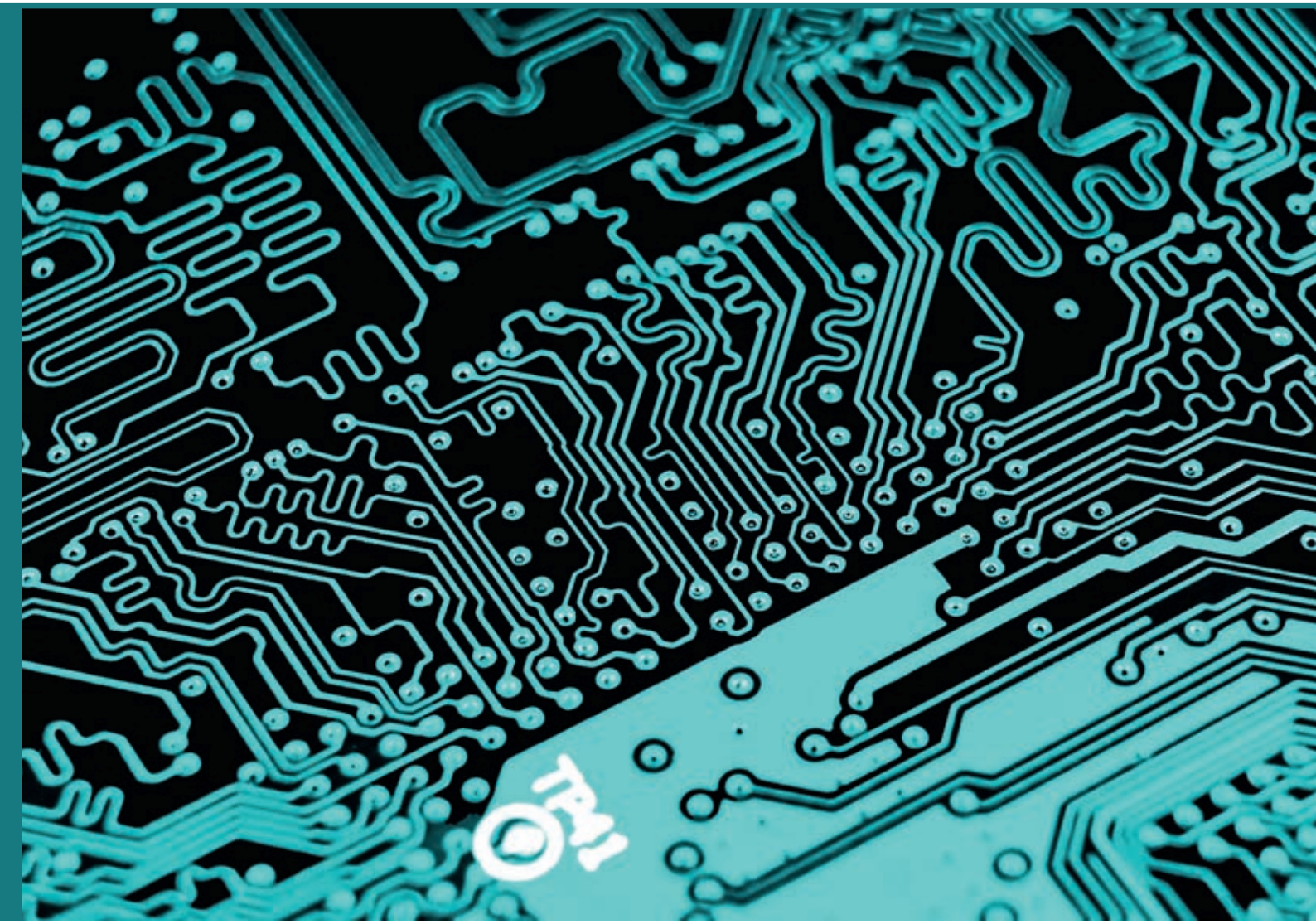
RIGA TECHNICAL UNIVERSITY

Donato Repole

RESEARCH OF PARALLEL COMPUTING NEURO-FUZZY NETWORKS FOR UNMANNED VEHICLES

Doctoral Thesis



RTU Press
Riga 2021

RIGA TECHNICAL UNIVERSITY

Faculty of Electrical and Environmental Engineering
Institute of Industrial Electronics and Electrical Engineering

Donato REPOLE

Doctoral Student of the Study Programme "Computerized Control of Electrical Technologies"

RESEARCH OF PARALLEL COMPUTING NEURO-FUZZY NETWORKS FOR UNMANNED VEHICLES

PhD Thesis

Scientific supervisor

Assistant Professor Dr. sc. ing.

Leslie Robert ADRIAN

RTU Press

Riga 2021

Repole D., Research of Parallel Computing Neuro-fuzzy Networks for Unmanned Vehicles. Doctoral Thesis. – Riga: RTU Press, 2021. – 249 p.

Published in accordance with the decision of the Promotion Council “RTU P-14” of 30 August 2021, Minutes No. 04030-9.12.1/9.

Parts of this work, including some travel costs and participation fees to conferences, have been supported by:

ROBOTRAX Limited (United Kingdom)

Purchase of materials and manufactured items have been supported by:

Lesla Latvia SIA

DOCTORAL THESIS PROPOSED TO RIGA TECHNICAL UNIVERSITY FOR THE PROMOTION TO THE SCIENTIFIC DEGREE OF DOCTOR OF SCIENCE

To be granted the scientific degree of Doctor of Science (Ph. D.), the present Doctoral Thesis has been submitted for the defence at the open meeting of RTU Promotion Council on 28 December 2021 at 12.00 at the Faculty of Electrical and Environmental Engineering of Riga Technical University.

OFFICIAL REVIEWERS

Dr. habil. sc. ing. Peteris Apse-Apsitis
Riga Technical University

Dr. habil. sc. ing. Raja Mazuir Raja Ahsan Shah
Coventry University, UK

Dr. habil. sc. ing. Andrés Gabriel García
Universidad Tecnológica Nacional, Bahía Blanca, Argentina

DECLARATION OF ACADEMIC INTEGRITY

I hereby declare that the Doctoral Thesis submitted for the review to Riga Technical University for the promotion to the scientific degree of Doctor of Science (Ph. D.) is my own. I confirm that this Doctoral Thesis had not been submitted to any other university for the promotion to a scientific degree.

Donato Repole ……. (signature)
Date: ……

The Doctoral Thesis has been written in English. It consists of 7 chapters, inclusive of the introduction; 111 figures, 19 tables; 88 equations; the total number of pages is 249. The Bibliography contains 66 titles.

ABSTRACT

The presented Doctoral Thesis illustrates the Author's researches in the field of VHDL based "neuro-fuzzy controllers". The Author's academic investigations involve numerous applications of "neuro-fuzzy controllers", and part of Doctoral researches focuses on evaluating different implementation methods. The decision of VHDL as "controller's hardware description language" is the outcome of the Author's academic researches, which are the core of the Author's international papers.

Presented work starts with an overview of autonomous mobile robotics applications, automotive applications and small Autonomous Unmanned Aerial Vehicles (derivative from RC planes), which describes the context where the Doctoral Thesis is implanted.

Then, the dissertation moves to the motivations behind the decision process of the selection of VHDL as "controller's hardware description language", strictly correlated to the flexibility and the advantages of using an FPGA instead of a multi-core MCU. A major focus is given to the FPGAs parallel processing functionality. Part of the Doctoral Thesis scrutinises methods to mitigate the complexity of a VHDL based description and the implementation of advanced learning processes.

Doctoral Thesis examines a novel software tool for the high-level "neuro-fuzzy controller" description capable of executing controller simulations, optimisation tasks, performing learning/training tasks, and exporting the controller in VHDL code.

The Thesis proposes an application case for the VHDL based "neuro-fuzzy controllers" researches, aiming the use of learning/training controller's capability to off-load the mechanical design. This approach targets the controller fine-tuning through a replicable process, which shall allow adapting the controller's parameters to the mechanical characteristics of the RC plane that shall be converted into a small Unmanned Aerial Vehicle. A series of mechanical and electrical/electronic hardware assumptions and definitions are made as pre-requisites for the controller conception. The proposal's focus is the controller's design strategy, scrutinising the design process, the description and the simulation of the "neuro-fuzzy controller".

Since the system's pre-requisites and boundary conditions are finalised to deliver a general aerial vehicle controller, the Thesis aims to deliver a "neuro-fuzzy controller" capable of replicating a human being pilot behaviour. Efforts are made to establish: fuzzy controller's simulation (fuzzy controller is the core of the "neuro-fuzzy controller" before the

learning/training process and the optimisation process), a learning/training process and, an optimisation process.

A learning capable controller design may result in a very sophisticated design, and the designer shall rely on robust software tools; the selection of the learning/training acceleration tool becomes a crucial step of the dissertation application case. Even more important for the dissertation is the definitions of the “Simulation Conditions” on which the “core fuzzy controller” shall be tested. In fact, a mandatory condition for an appropriate learning/training process is to use a “core fuzzy controller”, already capable of performing basic tasks, as the heart of the system.

What is drawn, between the lines, by the Doctoral Thesis is the introduction of a design strategy that is looking to develop solutions for complex controller architecture of mobile robotic vehicles (of any nature) or even for multiple industrial application. This work enables further investigative researches into autonomous robotics, particularly to the physical implementation of an Autonomous Aerial Unmanned Vehicle from an inexpensive RC plane.

A simplified RC plane design may be used, as a worst-case scenario for the controller design, where a 3D printed homebuilt aircraft may be turned into AUAV, through the process and the algorithms disserted. Replication of the learning/training process and their iteration on different mechanics and different RC planes to be adapted into AUAV may result in information gold mining for the researchers. Indeed, the determination of reliable processes allows researchers to reutilise the same principles for totally different applications, circumscribed only by the researcher’s imagination.

The Doctoral Thesis has been written in English. All summaries and conclusions and the results of the research relate to the hypothesis and the relationship between them. Researches outcome has the potential to evolve into other projects consisting of various methodologies extracted from the investigations.

The Thesis consists of 7 chapters, inclusive of the introduction and the subsequent conclusions.

The bibliography contains 66 reference sources and 12 appendices.

The volume of the present Doctoral Thesis is 249 pages.

It has been illustrated with 111 figures, 88 formulas and 19 tables.

Anotācija

Promocijas darbs ilustrē autora pētījumus saistībā ar "neironu faziloģikas kontrolleriem", kam pamatā ir VHDL. Autora akadēmiskie pētījumi ietver daudzus "neironu faziloģikas kontrolleru" izmēģinājumus, un daļa pētījumu ir vērsta uz dažādu to ieviešanas metožu novērtēšanu. Lēmums izmantot VHDL kā "kontrollera aparatūras aprakstīšanas valodu" ir autora akadēmisko pētījumu rezultāts, kas ir autora starptautisko rakstu pamatā.

Promocijas darba sākumā ir autonomas mobilās robotikas lietojuma, transportlīdzekļu lietojuma un mazu autonomu bezpilota lidaparātu (atvasinājums no radiovadāmām lidmašīnām) pārskats, kurā aprakstīts konteksts promocijas darba ietvaros.

Turpmāk disertācijā aprakstīts lēmumu pieņemšanas process, izvēloties VHDL kā "kontrollera aparatūras aprakstīšanas valodu", kas cieši saistās ar iespēju dažādību un priekšrocībām, izmantojot FPGA, nevis daudzkodolu MCU. Liela daļa uzmanības tiek pievērsta FPGA paralēlās apstrādes funkcionalitātei. Promocijas darbā tiek pārbaudītas metodes, lai mazinātu uz VHDL balstīta apraksta sarežģītību un progresīvu mācību procesu ieviešanu.

Promocijas darbā tiek pētīts jauns programmatūras rīks augsta līmeņa "neironu faziloģikas kontrollera" aprakstam, kas spēj izpildīt kontrollera simulācijas, optimizācijas uzdevumus, veikt mācīšanās/apmācības uzdevumus un spēj eksportēt kontrolleri VHDL kodā.

Disertācijā tiek piedāvāts uz VHDL balstītu "neironu faziloģikas kontrolleru" pētījumu izmantošanas gadījums ar mērķi izmantot mācīšanās/apmācības kontrollera spējas mehāniskās konstrukcijas noslogošanai. Šī pieeja ir vērsta uz kontrollera precīzu noregulēšanu ar atkārtojumu palīdzību, kas ļauj kontrollera parametrus pielāgot radiovadāmas lidmašīnas mehāniskajām īpašībām, kura tiks pārveidota par mazu bezpilota lidaparātu. Kā priekšnosacījums kontrollera koncepcijai tiek izveidota virkne mehānisku un elektrisku/elektronisku aparatūras pieņēmumu un definīciju. Šajā priekšlikumā galvenā uzmanība tiek pievērsta kontrollera projektēšanas stratēģijai, rūpīgi pārbaudot "neironu faziloģikas kontrolleru" veidošanas procesu, aprakstu un simulāciju.

Tā kā sistēmas priekšnosacījumi un robežnosacījumi ir pilnībā skaidri, tad, lai izveidotu universālu lidaparāta kontrolleri, disertācijas mērķis ir radīt "neironu faziloģikas kontrolleri", kas spētu imitēt cilvēka kā pilota rīcību. Ir mēģinājumi izveidot faziloģikas kontrollera simulāciju (faziloģikas kontrolleris ir "neironu faziloģikas kontrollera" pamats pirms

mācīšanās/apmācības un optimizācijas procesa), mācīšanās/apmācības procesu un optimizācijas procesu.

Tādu kontrolleri ar iemācīšanās opciju var būt ļoti sarežģīti izveidot un tā veidotājam jāpaļaujas uz izturīgiem programmatūras rīkiem – mācīšanās/apmācības paātrināšanas rīka izvēle kļūst par izšķirošu soli disertācijā aprakstītā izmantojuma gadījumā. Vēl nozīmīgāki ir "Simulācijas apstākļu" nosacījumi, kuros būtu jātestē "pamata faziloģikas kontrolleris". Patiesībā obligāts nosacījums atbilstošam mācīšanās/apmācības procesam ir izmantot "pamata faziloģikas kontrolleri" kā sistēmas centru, kas jau spēj veikt vienkāršus uzdevumus.

Disertācijā vispārējā ideja ir ieviest tādu projektēšanas stratēģiju, kuras mērķis ir izstrādāt risinājumus mobilo robotu transportlīdzekļu (jebkāda veida) sarežģītai kontrolleru arhitektūrai vai pat dažādām industriālām vajadzībām. Šis darbs dod iespēju turpināt pētījumus par autonomu robotiku, jeb radīt autonomu bezpilota lidaparātu no lētas RC (radiovadāmas) lidmašīnas.

Vienkāršotu RC lidmašīnas projektu var izmantot kā pēdējo variantu neveiksmīgāka rezultāta gadījumā, lai izveidotu kontrolleri, kur 3D formātā izdrukātu un mājās uzbūvētu lidaparātu procesa gaitā un ar pārrunāto algoritmu palīdzību varētu pārveidot par AUAV. Mācīšanās/apmācības procesu imitēšana un to atkārtojums dažādos mehānismos un dažādām RC lidmašīnām, ko pārveidotu par AUAV, pētniekiem varētu kļūt par informācijas zelta raktuvēm. Patiesi, uzticamu procesu noteikšana ļauj pētniekiem atkārtoti izmantot vienus un tos pašus principus pilnīgi atšķirīgiem pielietojumiem, ko ierobežo tikai pētnieka iztēle.

Promocijas darbs ir uzrakstīts angļu valodā. Visi kopsavilkumi un secinājumi, kā arī pētījumu rezultāti ir saistīti ar hipotēzi un mijiedarbību starp tiem. Pētījumu rezultātiem ir potenciāls attīstīties citos projektos, kas sastāvētu no dažādām metodoloģijām, kas rastos no pētījumiem.

Promocijas darbs sastāv no 7 nodaļām ieskaitot ievadu un sekojošos secinājumus.
Bibliogrāfija sastāv no 66 atsaucēm un 12 pielikumiem.
Promocijas darba apjoms ir 249 lapas.
Darbā ir 111 zīmējumi, 88 matemātiskās formulas un 19 tabulas.

CONTENTS

1 Introduction 13

1.1 Unmanned Vehicles Introduction 13

1.1.1 Unmanned Ground Vehicles 13

1.1.2 Unmanned Aerial Vehicles 16

1.1.3 Unmanned Underwater Vehicles 17

1.2 Topicality 18

1.3 Primary Hypothesis and Intentions 19

1.4 Methods of Research and Development 20

1.5 Scientific Novelty 21

1.6 Practical Application of Research Results 21

1.7 Dissemination of Research Results 22

2 Unmanned Vehicles Control Strategies Overview 24

2.1 Principles of flight dynamics 24

2.2 Linear Control 27

2.3 Non-Linear Control 31

2.4 Practical Non-Linear UAV control strategy, CPWL Mathematical model and controller approximation 34

2.5 Neuro-Fuzzy Logic for UV and Robotic applications 36

2.5.1 AI, Learning-Based Control 37

2.5.2 Data capture for the training process 39

3 Typical Architectures for Electric Vehicles 41

3.1 Automotive Regulations/Standards Overview 41

3.2 Low Voltage Electric Vehicles Architecture 43

3.3 High Voltage Electric Vehicles Architecture 44

3.4 Hybrid Architecture variants 46

3.4.1 Hybrid Electric Vehicle (HEV) Overview 46

3.4.2 Plug-In Hybrid Electric Vehicle (PHEV) Overview 50
3.4.3 Mild Hybrid Electric Vehicle (MHEV) Overview 50
3.5 Example of 48V REESS with Boost Voltage Converters 50
3.6 Electric Vehicles Power Converters 51
3.6.1 Discrete Power Elements design 51
3.6.2 Critical issues on Discrete Power Elements Power PCB 53
3.6.3 “Power Module”, Benefits for Electric Vehicle Power Converters 54
3.6.4 Wide bandgap (WBC) Semiconductors advantages for Electric Vehicles 57
3.7 Battery Technology Overview 57
3.7.1 PHEV, HEV and EV Battery Cell Chemistry Overview 58
3.7.2 Battery Management System (BMS) 60
3.7.3 Application of Fuzzy Logic for BMS 64
4 Theoretical Framework 65
4.1 Study Case Introduction 67
4.2 Controller’s Framework Definition 68
4.2.1 Controller’s Inputs Definition 68
4.2.2 Controller’s Outputs Definition 69
4.2.3 Rule Block and Defuzzification 70
4.2.4 VHDL implementation theory 70
4.2.5 VHDL Modelling theory 75
5 System’s Hardware Design Proposal 80
5.1 Core Hardware Definition 80
5.1.1 Control Unit, FPGA 81
5.1.2 Digital Motion Sensor 82
5.1.3 Gyroscope 84
5.1.4 Landing Proximity sensor 85
5.1.5 Navigation Monitor 87

5.1.6 Electronic Compass Unit 95
5.1.7 Motor Drive - Powertrain 95
5.1.8 Electro-Mechanical Actuators – SERVO 96
5.1.9 Data Storage 100
5.1.10 Battery management and Low Voltage power supply management 101
5.2 Human Remote Control 103
6 Study Case, Controller's Design Proposal 105
6.1 Controller's Inputs 105
6.1.1 VHDL Component A3G4250D 106
6.1.2 VHDL Component LIS3DSH 107
6.1.3 VHDL Component TESEO 108
6.1.4 VHDL Component, Safety Sensors 109
6.1.5 VHDL Component, Flight Parameters EEPROM 111
6.1.6 Controller's core inputs, summary 111
6.2 Controller's Outputs 111
6.2.1 VHDL Component SERVO 112
6.2.2 Powertrain's VHDL Components 112
6.2.3 VHDL Component, Flight Telemetry EEPROM 113
6.2.4 Controller's core outputs, summary 113
6.3 Fuzzy Logic Controller Design 114
6.3.1 Type "Rudder_SERVO", Membership Output Function 116
6.3.2 Type "Altitude_input", Membership Input Function 118
6.3.3 Type "Compass_input", Membership Input Function 120
6.3.4 Type "Energy_Status", Membership Input Function 123
6.3.5 Type "Speed_Input", Membership Input Function 124
6.3.6 Type "Pitch_angle_Input", Membership Input Function 125
6.3.7 Type "Yaw_angle_Input", Membership Input Function 127

6.3.8 Type “Rolling_angle_Input”, Membership Input Function 128
6.3.9 Type “Aileron_SERVOs”, Membership Output Function 129
6.3.10 Type “ELEV_SERVO”, Membership Output Function 131
6.3.11 Type “M1_THROTTLE”, Membership Output Function 133
6.3.12 Type “M2_THROTTLE”, Membership Output Function 135
6.4 Controller’s Rulebases 137
6.4.1 “ELEV_SERVO”, Rulebase 137
6.4.2 “Aileron_SERVO”, Rulebase 140
6.4.3 “RUDD_SERVO”, Rulebase 141
6.4.4 “M1”, Rulebase 143
6.4.5 “M2”, Rulebase 145
6.5 Fuzzy Controller System Structure 147
6.6 Fuzzy Controller Simulations and preliminary optimisation 149
6.6.1 Take-Off simulation 149
6.6.2 Route adjustment Simulation 160
6.6.3 Steady-state simulation 168
6.6.4 Adjustment due to gusty winds simulation 169
6.6.5 Landing Simulation 172
6.7 Controller 182
6.7.1 Digital_Processing VHDL component 186
6.7.2 “Fuzzy”, VHDL component 193
6.8 Data Capture for the learning Process 194
6.9 Learning/Training Process Description 199
6.9.1 Knowledge acquisition tool (“xfdm” tool) 200
6.9.2 The supervised learning 202
6.9.3 Error function 206
6.9.4 The Simplification tool - Xfsp 207

6.10 Proposed Learning Tool Configuration 209
7 Conclusions and Further Researches 212
REFERENCES 219
APPENDICES 225
A. Abbreviations 225
B. Non-Linear System Linearization Technique 230
C. Lyapunov Theorem 232
D. Lyapunov Stability Technique 233
E. CPWL Function introduction 235
F. Fuzzy logic Introduction 235
Fuzzy Logic Membership Input Functions 236
Fuzzy Hedges 237
Fuzzy Output Defuzzification 237
G. Fuzzy Logic applications in smart electrical systems 238
H. Neuro-Fuzzy introduction 239
Definition of Neuro-Fuzzy modules 240
Definition of Genetic Algorithm 241
I. TYPES OF NEURO-FUZZY SYSTEMS 242
Cooperative Neuro-Fuzzy Systems 242
Concurrent Neuro-Fuzzy Systems 243
Hybrid Neuro-Fuzzy Systems 243
J. ARTIFICIAL NEURAL SYSTEMS 244
K. Automotive trend interpretation for EV, PHEV and HEV 246
L. ANCILLARY RESULTS FROM THE THESIS WORK 247
UAV Motor Drive Doctoral Research 247
UAV Motor Drive Doctoral Research Conclusion 248

1 Introduction

1.1 Unmanned Vehicles Introduction

Nowadays, Unmanned Vehicles are getting more popular. Although during the last decades' Unmanned Vehicles mainly had a military application, today it is possible to observe Unmanned Vehicles in factories, streets, civil airfields and cities' parks. It is possible to divide Unmanned Vehicles into three groups:

a) Unmanned Ground Vehicles;
b) Unmanned Aerial Vehicles;
c) Unmanned Underwater Vehicles.

1.1.1 Unmanned Ground Vehicles

In the previous decades, "Unmanned Ground Vehicles" (UGV) were primarily associated with robots, especially indoor robots used for carrying goods from deposit to factory's working station. Over the last few years, an impressive technological acceleration brought to market many new UGV for both military and civil application. The most prevalent military applications are:

a) hazardous object manipulations;
b) explorer;
c) delivery of goods and supplies.

There are many attention-grabbing systems developed for military applications, such as the ANDROS (Northrop Grumman Unmanned Ground Systems - Figure 1.1) or the Lockheed Martin Squad Mission Support System (SMSS – Figure 1.2). Civil applications for UGV result in line with the Thesis researches and, it is possible to highlight a few exciting applications:

a) industrial applications (Autonomous Robots or AR);
b) Utility Unmanned Ground Vehicles (UUGV);
c) human transportation.

Figure 1.1: ANDROS - Northrop Grumman, Unmanned Ground Vehicles.

Figure 1.2: Lockheed Martin Squad Mission Support System (SMSS).

Industrial utilisation is the most common application of civil UGVs. The best example is a robotic unit that delivers parts from the stockroom to the single working station and withdraws from the working station to the refuse warehouse. Fundamentally this application has two advantages: increase the factory's efficiency/productivity and limit human handling of dangerous refuses. "Figure 1.3" represents a typical illustration of an industrial UGV operation environment.

Figure 1.3: example of Industrial Ground Vehicle.

"Utility Unmanned Ground Vehicle" is a general definition, which may be associated with a wide range of UGVs. Generally, it is associated with a vehicle that performs a specific task, which might be the delivery of a parcel or a farm field groundwork. A good example could be a driverless-capable truck able to assist the driver during his delivery route[1] (in order to increase the vehicle's productivity).

One more interesting example of a "Utility UGV" is represented by the autonomous tractor, as shown in "Figure 1.4".

[1] The amount of items that may be delivered during a shift by a driver, is generally affected by the availability of parking space near the delivery area. This particular kind of UGV may allow the driver to stop near the address where an item should be delivered, in meanwhile truck will autonomously move within a pre-defined area and will pick up driver after a specified amount of time. This strategy targets delivery time efficiency optimisation.

Figure 1.4: Prototype of Autonomous Tractor.

The last group of autonomous vehicles (human transportation UV topology) might be perceived in contrast with the UV's definition because the vehicle carries a "Human Load". In fact, by definition: UV may be interpreted as a vehicle without any humans or as a vehicle that is driverless and capable of performing complex task autonomously, independently by the load that they are carrying; in this case, passengers should be professed only like a "load" ("Human Load").

The best example of a driverless capable passenger vehicle is the "TESLA AUTOPILOT" (although the legislator, at the moment, does not allow the driver to take off both hands from the steering wheel while the vehicle is moving).

1.1.2 Unmanned Aerial Vehicles

Unmanned Aerial Vehicles (UAV) is the most famous category of Unmanned Vehicles, mainly due to the impact that UAV had on military combat strategies and techniques. Today, it is common to associate the perception of a UAV with the USAF models "PREDATOR" and "REAPER". It is also important to highlight that almost all developed countries are running programs for new UAVs and AUAVs combat systems that will progressively replace the human-piloted reconnaissance vehicles and, lastly, the combat aeroplanes.

Figure 1.5: General Atomics Predator B (or MQ-9 Reaper).

The civil market is also observing a broad interest in UAVs for multiple purposes. Increased availability of cheap and durable batteries mixed with the low cost and high-performance available electronics for controls and power electronics made it possible to diffuse small-sized hobby UAVs (widely accessible in a consumer electronics store; it is common to observe in our parks flying quad-copters or small model based aeroplanes controlled by a smartphone or just flying entirely autonomously).

1.1.3 Unmanned Underwater Vehicles

UUVs may be divided into the two categories of "Remotely Operated Underwater Vehicles" (ROUV) and "Autonomous Underwater Vehicles" (AUV). Previously, UUVs have been used for a limited number of tasks dictated by the technology available. Recently, with technological progress, UUVs and (particularly AUVs) are now being used for more and more challenging tasks. It is possible to highlight four main applications of UUVs.

- **Commercial:** UUVs are very popular in the oil and gas industry for a large variety of uses[2];

[2] Such as: the definition of detailed maps of the seafloor, pre-lay or post-lay subsea infrastructure survey, pipelines or any subsea infrastructure installation and maintenance.

- **Military**: the navies of multiple countries are currently producing UUVs to be used in oceanic warfare, with particular attention to the sea exploration to eradicate underwater mines threats or the sea exploration to detect unfriendly objects[3];
- **Research:** scientists rely on a heterogeneous[4] variety of UUVs to study lakes, seas, and the ocean floor;
- **Hobby:** many robotic enthusiasts enjoy constructing and operating UUVs as a hobby.

1.2 Topicality

Previously described autonomous vehicles applications (mobile robotics applications, driverless automotive applications, AUAV and UAV applications, etc.) define the context where the Doctoral Thesis is implanted. However, the *Doctoral Thesis objective is devoted to the research and development of VHDL based "neuro-fuzzy controllers" for small Autonomous Unmanned Aerial Vehicles* (derivative from RC planes).

For many years, fuzzy logic has been an attractive technology for designers of industrial, consumer and automotive products. Conversely, achieving the right balance between cost and performance results in a difficult task. In fact, fuzzy algorithms can be executed on low-cost MCUs, but as these have architectures that were not designed to handle fuzzy logic often their performance results being inadequate. Dedicated fuzzy processor microchips can meet the most demanding performance requirements, but it is mainly an expensive ASIC solution. Indeed, only a few full customs (or semi-custom) integrated fuzzy controllers exist and most of them are assembled from standard cells at the gate level. At the moment, an FPGA based solution is a valid option capable of delivering both: good system performances and high flexibility to the designer. Relevant scientific literature proves the strength of the use of FPGAs for the implementation of neuro-fuzzy controllers. FPGA's parallel processing capability results in a significant advantage over the use of conventional MCUs, which are operating serial data processing.

[3] There are a very large variety of objects that a navy may wish to detect. As simple example a navy may look for a missing airplane's wreckage or an illegal UUV used for drug smuggling.

[4] In function of the UUV's task its technical characteristics may significantly change. For instance, a variety of sensors can be affixed to AUVs to measure the concentration of various elements or compounds, the absorption or reflection of light, and the presence of microscopic life. Or operate conductivity-temperature-depth sensors (CTDs), fluorometers, and pH sensors.

The selection of VHDL as "controller's hardware description language" has its fundamentals on the VHDL efficiency and reliability for sophisticated hardware, verified in numerous scientific articles, such as an AUAV controller.

1.3 Primary Hypothesis and Intentions

As system pre-requisites and boundary conditions are finalised to deliver a general aerial vehicle controller, the Thesis aims to deliver a "neuro-fuzzy controller" capable of replicating a human being pilot behaviour. A series of mechanical and electrical/electronic hardware assumptions and definitions are made as pre-requisites for the controller's conception.

*Project's **Hypotheses** are:*

a) to use a fly-wing platform as a baseline for the small UAV mechanical design;
b) to use a not optimised mechanical design;
c) the controller shall compensate eventual mechanical misbalances;
d) the controller, by definition, has nine inputs (altitude, speed, pitch angle, rolling angle, yaw angle, estimated position, flight reference parameters, proximity sensor and, battery state of charge);
e) the controller, by definition, has five outputs (ailerons, elevator, rudder, left E-Motor and, right E-Motor)
f) to use a Lattice Semiconductors automotive-qualified FPGA;
g) to define a binding set of "simulation conditions" for the validation process of the controller;
h) to use two independent low-cost BLDC motors and two independent motor drivers;
i) the telemetry's data will be stored into a dedicated EEPROM.

The project's goals are to establish: fuzzy controller simulation, a learning/training process and, an optimisation process.

*Project's **Intentions** are:*

a) to move part of the hardware/mechanical design load, making it as thoroughly as possible, to the controller design;
b) overcome the design load of defining the UAV flight dynamics model;
c) to predispose the controller design to be easily adapted to different platforms with different flight dynamics models;
d) to define a control unit capable of parallel computation;

e) to design a fuzzy logic controller able to perform limited flying operations;
f) to validate the fuzzy logic controller through the use of simulations;
g) to define the fuzzy logic controller optimisation process;
h) to define an algorithm development strategy for the learning/training process.

1.4 Methods of Research and Development

Doctoral Thesis scrutinises methods to mitigate the complexity of a VHDL based controller's description and to implement advanced learning processes.

A learning capable controller design may result in being a very complex project, and the designer shall rely on robust software tools. The selection of the learning/training acceleration tool becomes a crucial step of the dissertation. However, paramount importance is assigned to the definitions of the "simulation conditions" on which the "core fuzzy controller" should be tested. In fact, a mandatory condition for an appropriate learning/training process is to use a "core fuzzy controller", already capable of performing basic tasks, as the heart of the system.

Many of the processes of theoretical calculations and graphical representation of the results have been obtained utilising a menagerie of software systems, including:

- Aforge.net (C# framework);
- ALDEC Active-HDL (VHDL compatible FPGA design creation and simulation environment);
- Altium Designer (hardware design environment);
- Cadence-OrCAD (hardware design environment);
- fuzzyTECH (Fuzzy/Neural GUI for fuzzy logic modelling and programming algorithms);
- Lattice Diamond (Lattice Semiconductors VHDL design environment);
- LT Spice (hardware design environment);
- MATLAB (multi-paradigm numerical computing environment);
- Maplesoft Maple (symbolic and numeric computing environment);
- Microsoft Excel (tables and spreadsheets);
- Microsoft Paint 3D (2D parts design);
- Microsoft PowerPoint (2D parts design);
- Microsoft Visio (2D parts design);
- Microsoft Word;

- Model-Sim (Mentor Graphics);
- Neural.NET (neural guided learning software);
- Pspice (Circuit modelling and analysis);
- Synopsys Synplify PRO (VHDL compatible FPGA synthesis software);
- Solidworks (3D parts design environment);
- XFL3 (Xfuzzy 3 GUI development environment for Neuro-Fuzzy system design, optimisation and simulations).

1.5 Scientific Novelty

The project's primary focus is a small AUAV (it is assumed to be an autonomous RC plane) "neuro-fuzzy controller" capable of replicating a human being pilot behaviour. Efforts are concentrated on the design, the description and the simulation of the "neuro-fuzzy controller". Doctoral Thesis introduces a design strategy proficient at supporting advanced controller's development for mobile robotic vehicles of any nature or even for multiple industrial application[5].

The dissertation examines a novel software's tool for the high-level "neuro-fuzzy controller description" capable of executing controller's simulations, optimising tasks, performing learning/training tasks, and exporting the controller in VHDL code.

The resulting outcome is a flexible and innovative system capable of being adapted to a different machine through a training-based process for the adjustment, weighting, and learning of the neuro-fuzzy control algorithm.

1.6 Practical Application of Research Results

The Thesis proposes an application case for the VHDL based "neuro-fuzzy controllers" researches, aiming the use of learning/training controller's capability to off-load the mechanical design. This approach targets the controller's fine-tuning through a replicable process, which shall allow adapting the controller's parameters to the mechanical characteristics of the RC plane that might be converted into a small AUAV.

A simplified 3D printed homebuilt RC plane is an extreme study case. Turning it into a basic AUAV, thanks to the use of core algorithms properly adapted and developed through a

[5] Few Author's international publications developed self-learning concepts capable of being applied to a wide range of difficult control networks design.

series of proposed optimisation and learning/training processes, it may represent a remarkable achievement.

Researches results are applicable to a wide range of autonomous robotics, not only to the physical implementation of an AUAV from an inexpensive RC plane. The determination of reliable processes may allow reutilising the same principles for different kind of applications.

1.7 Dissemination of Research Results

Author's academic investigations touched on several different applications of "neuro-fuzzy controllers", and part of Doctoral research focused on evaluating different implementation methods. The decision process that led to the selection of VHDL as "controller's hardware description language" is linked to the Author's academic researches.

The following ten publications are presented in the Doctoral Thesis:

1. L. R. Adrian, **D. Repole** and L. Ribickis, "Proposed neuro-guided learning for obstacle avoidance in AMBOA robotic device", 2015 56th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), Riga, 2015, pp. 1-5.
2. Janis Voitkans, Leslie R. Adrian, **Donato Repole**, "INVESTIGATION OF ELECTRICAL PARAMETERS FOR PCB TRANSFORMER" 15th International Scientific Conference: Engineering for Rural Development 25-27.05.2016 Jelgava, LATVIA, pp. 1445-1452.
3. L. R. Adrian, **D. Repole** and L. Ribickis, "High efficiency modular DC-DC power converter for adaption to industrial & hybrid robotics", 2016 57th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), Riga, 2016, pp. 1-5.
4. L. R. Adrian and **D. Repole**, "Intelligent autonomous environmental monitoring based on the AMBOA robot sensory system", 2017 IEEE 58th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), Riga, 2017, pp. 1-6.
5. **D. Repole** and L. R. Adrian, "Fuzzy nano piezo hybrid for fault detection in automotive power PCB", 2017 IEEE 37th International Conference on Electronics and Nanotechnology (ELNANO), Kiev, 2017, pp. 400-404.
6. **D. Repole** and L. R. Adrian, "Evaluation of GaN MOSFET for Unmanned Aerial Vehicles BLDC Motor Drive", 2018 IEEE 59th International Scientific Conference on

Power and Electrical Engineering of Riga Technical University (RTUCON), Riga, Latvia, 2018, pp. 1-4.

7. **D. Repole** and L. R. Adrian, “Introduction to Parallel MAS Control for MAS - Smart Sensor Networks”, 2019 IEEE 60th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), Riga, Latvia, 2019, pp. 1-5.
8. L. R. Adrian, **D. Repole** and A. Rubenis, “Comparative study of Lithium-Ion hybrid super capacitors”, 19th International Scientific Conference Engineering for Rural Development, 20-22.05.2020 Jelgava, LATVIA, pp. 906-912. DOI:10.22616/ERDev.2020.19.TF217.
9. **D. Repole** and L. R. Adrian, “VHDL based Neuro-Fuzzy Lithium-Ion Hybrid Super Capacitors management, Advantages of the high-level descriptions of neural fuzzy logic-based systems”, 2020 IEEE 61th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), Riga, Latvia, 5-6 Nov. 2020.
10. Krists Kviesis, Leslie Robert Adrian, Ansis Avotins, Olegs Tetervenoks and **D. Repole**, “MAS Concept for PIR Sensor-Based Lighting System Control Applications”, 8th IEEE Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE’2020), Vilnius (Lithuania) 2021, accepted for publication (ID: PID014).

2 Unmanned Vehicles Control Strategies Overview

Each kind of Unmanned Vehicles has a specific control strategy that allows the performing of particular tasks. In specific, it is taken as an example of the control strategy for a civil UAV. Generally, it might be applied to three different control strategies to govern a UAV, which are:

a) linear control;

b) non-linear control;

c) AI (learning-based control).

2.1 Principles of flight dynamics

Newton's laws of mechanics for a body frame, whose origin coincides with the aircraft's centre of mass, are defined by:

$$F_B = m \cdot \dot{v}_B + \omega_B \times m \cdot v_B$$

(Equation 1)

$$T_B = J_B \cdot \dot{\omega}_B + \omega_B \times J_B \cdot \omega_B$$

(Equation 2)

$$J_B = \int_{Aircraft} \rho(s)[(s \cdot s)I - s \otimes s]ds$$

(Equation 3)

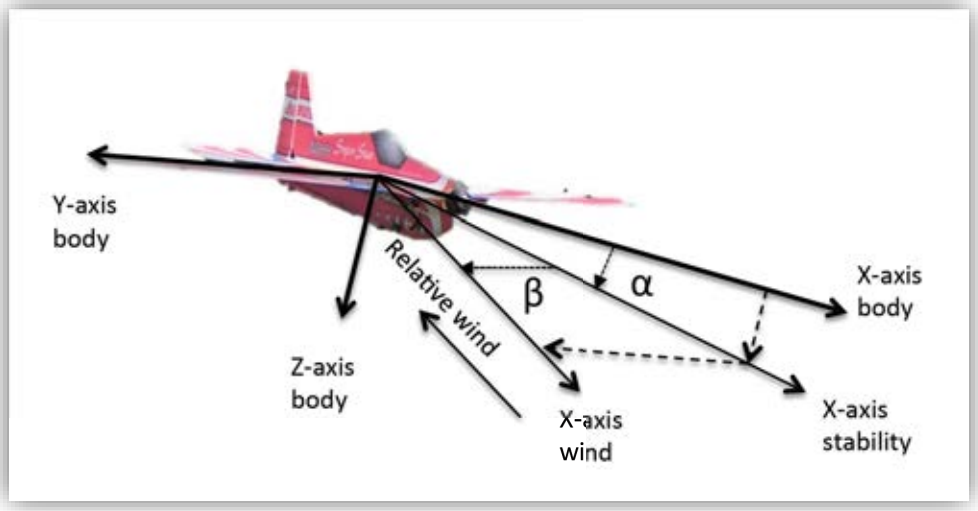


Figure 2.1: various body frames used for aircraft analysis.[1]

Where: the subscript “B” indicates vectors or tensors expressed in the body frame, the dot indicates differentiation with respect to time, “v_B” is the velocity of the aircraft’s centre of mass with respect to the inertial frame, “ω_B” is the angular velocity of the body frame with respect to the inertial frame, “m” is the aircraft's mass, and “J_B” is its inertia tensor, which is constant in the body frame. [1]

Equations mentioned above might be written as follows:

$$\frac{1}{m}F_B \equiv \frac{1}{m}\begin{bmatrix}X\\Y\\Z\end{bmatrix} = \begin{bmatrix}\dot{U} + QW - RW\\ \dot{V} + RU - PW\\ \dot{W} + PV - QU\end{bmatrix}$$

(Equation 4)

$$T_B \equiv \begin{bmatrix}\bar{L}\\M\\N\end{bmatrix} = \begin{bmatrix}J_{xx}\dot{P} - J_{xz}\dot{R} + QR\left(J_{zz} - J_{yy}\right) - PQJ_{xz}\\ J_{yy}\dot{Q} + PR\left(J_{xx} - J_{zz}\right) + \left(P^2 - R^2\right)J_{xz}\\ J_{zz}\dot{R} - J_{xz}\dot{P} + PQ\left(J_{yy} - J_{xx}\right) - QRJ_{xz}\end{bmatrix}$$

(Equation 5)

“F_B” denotes the sum of the forces acting on the vehicle (including aerodynamic, gravity, thrust, and buoyancy), and “T_B” denotes the sum of the moments of these forces about its centre of mass. [1]

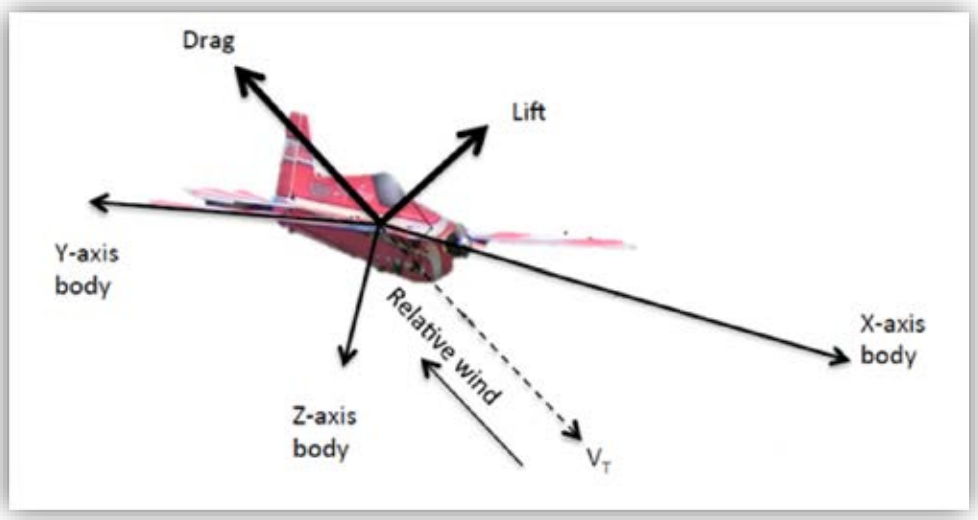


Figure 2.2: representation of the “lift” and the “drag” acting on the aircraft.[1]

Neglecting the aircraft's rotational dynamics and treating it as a point mass with no thrust, the twelve non-linear equations of motion (EOMs) used to represent 6-DOF aircraft motion (Honeywell, 1996:65-66) reduce to the following six non-linear differential equations. [2]

$$\dot{V}_t = \frac{1}{m}[-D - mg\,sin(\gamma)]$$

(Equation 6)

$$\dot{\psi} = \frac{1}{m \cdot V_t \cdot cos(\gamma)}[L\,sin(\phi)]$$

(Equation 7)

$$\dot{\gamma} = \frac{1}{m \cdot V_t}[L\,cos(\phi) - mg\,cos(\gamma)]$$

(Equation 8)

$$\dot{h} = V_t \cdot sin(\gamma)$$

(Equation 9)

$$\dot{E} = V_t \cdot cos(\gamma)\,sin(\psi)$$

(Equation 10)

$$\dot{N} = V_t \cdot cos(\gamma)\,cos(\psi)$$

(Equation 11)

Where:

ψ → Heading Angle (deg)

φ → Roll Angle (deg)

γ → Flight Path Angle (deg)

D → Drag (kgf)

E → Inertial Crossrange Distance (m)

g → Gravitational Acceleration (m/s^2)

h → Inertial Altitude (m)

L → Lift (kdf)

m → Mass (kg)

N → Inertial Downrange Distance (m)

T → Thrust

V_t → True airspeed (m/s)

In the case of a vehicle with a powertrain capable of generating a thrust, “Equation 6” is not valid and shall be considered “Equation 12”.

$$\dot{V}_t = \frac{1}{m}\left[T - D - mg\,sin(\gamma)\right]$$

(Equation 12)

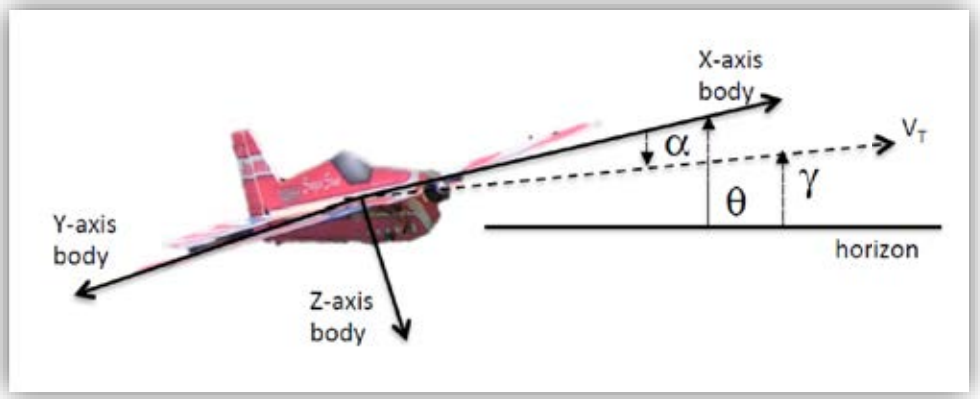


Figure 2.3: Dentition of the flight path angle.[1]

2.2 Linear Control

The UAV's control may be accomplished using estimates of its dynamic states, including position, velocity, angular rate, and attitude of the autonomous vehicle. Linear Control is the more straightforward approach for achieving a UAV autonomous control; a simplified set of aircraft motion equations can be derived for route planning purposes. In deriving these equations, the aircraft roll rate, pitch rate, and yaw rate dynamics are neglected and superseded using kinematic approximations. The resulting model describes the motion of a rigid point mass with kinematic path constraints. For this model, the position of the aircraft in an inertial frame whose x axis is parallel to the local horizon is denoted by: x_e, y_e, z_e. The flight "path angle γ" denotes the angle between the local horizon and the velocity vector of the aircraft "V_T". The "heading angle ψ" is the angle between: "V_T" and the z axis of the local inertial frame. The "bank angle ϕ" is the angle that the aircraft is banked about the velocity vector (V_T). The forces acting on the aircraft consist of the weight "$m \cdot g$", thrust "T", lift "L", and drag "D". The equations for a point mass model of a fixed-wing aircraft can then be formulated by assuming small "γ" and "ψ".

[1]

$$\dot{x}_e = V_T \cdot cos(\gamma) \cdot cos(\psi)$$

(Equation 13)

$$\dot{y}_e = V_T \cdot cos(\gamma) \cdot sin(\psi)$$

(Equation 14)

$$\dot{x}_e = -V_T \, sin(\gamma)$$

(Equation 15)

$$\dot{V}_t = \frac{1}{m}\left[T - D - mg \, sin(\gamma)\right]$$

(Equation 16)

$$\dot{\gamma} = \frac{1}{mV_T}[L \, cos(\phi) - mg \, cos(\gamma)]$$

(Equation 17)

$$\dot{\psi} = \frac{L \cdot sin(\phi)}{m \cdot V_T \, cos(\gamma)}$$

(Equation 18)

$$L \equiv \bar{Q} S C_l$$

(Equation 19)

$$D = \bar{Q} S (C_{D0} + K C_L^2)$$

(Equation 20)

Where:

C_{D0} → parasitic drag coefficient

C_L → lift coefficient

K → aircraft wing geometry constant

$\bar{Q}$ → dynamic pressure equal to $\frac{1}{2}\rho V_T^2$

ρ → air density

S → wing surface area (m^2)

A linear control tries to approximate a complex non-linear problem, although described by a simplified pattern of equations, into a linear system that may be reduced as a classical control feedback loop.

[3 and 4]

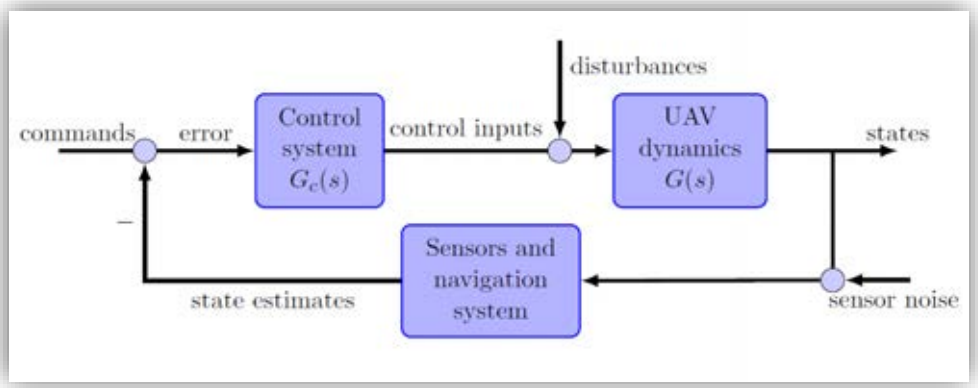


Figure 2.4: a classical control feedback loop representation. [1]

Due to the complexity of the dynamics, **there are two basic strategies for the control design.**

The first method is to continue the decomposition in the previous section to identify components of the dynamics that are well controlled by specific choices of the actuators, and then perform successive loop closure (Figure 2.5). In this case, the loops are nested by arranging that the outer-loop controller provides the reference commands for the inner loop. Figure 2.5 shows an example in which the outermost position control loop provides desired velocity commands using path following guidance previously discussed. The outer velocity control loop provides a reference (in this case, the desired quaternion value) for the inner attitude control loop. A key advantage of this approach is that it leads to a natural mechanism of handling limits on flights variables (e.g., bank or pitch angles) and actuator inputs because the reference commands can be saturated before being passed to the inner loop. Each step of the control design process may result simplistic, but the control loops' nesting leads to some challenges. A general design rule aims that the inner control loops result in "fast" dynamics, and then each successive loop added is "slower than the previous one". The primary challenges are the determination of what is fast or slow and the entity of the interaction between the inner/outer loops being closed[6]. "Figure 2.5" illustrates the previously described simplified approach (but extremely robust), which describes the "successive loop-closure control architecture". [1, 3 and 4]

[6] e.g., closing the outer loop might reduce the performance of the inner loop requiring a redesign.

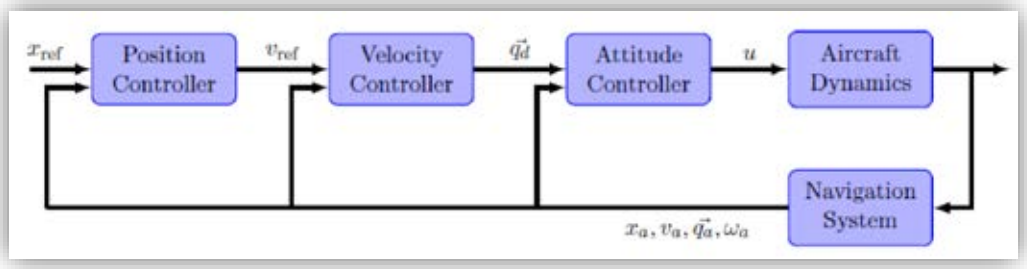


Figure 2.5: successive loop-closure control architecture.[1]

The second approach is to design a controller for the full dynamics, either linear or non-linear. The advantage of this approach is that it employs the state space control approaches to handle fully coupled dynamics. However, it is challenging to handle the actuator's saturation and very hard to include state constraints. Furthermore, unless done with extreme care, these controllers, especially in high-performance flight, can be very sensitive to modelling errors and omissions. [1 and 3]

As soon as is defined the architecture, the next step in any control design is to determine the dynamics of the system of interest (i.e., the full set of dynamics or the approximate inner loop dynamics). The subsequent action defines the requirements and the extent to which the dynamics will meet its goals. For example, could be requirements on a specific frequency (to ensure the dynamics are "fast") and damping (to ensure that the oscillations die out quickly) specifications on the pole locations. There may also be requirements on the maximum steady tracking error to a step command input. Since the vehicle's open-loop dynamics rarely satisfy these requirements, the typical approach uses linear feedback control to modify the pole locations and loop gains. [1 and 3]

A **full-state feedback controller** could implement this second strategy. Moving by a linearized and simplified state-space model, controller design is reduced to a classical exercise of well-known control techniques synthesis (controller could be designed using a specific technique such as: "Linear Quadratic Regulator", "Dynamic Output Feedback Controller", "Optimal Estimator", "Robust Controller", "Robust Control Synthesis" and etc.).

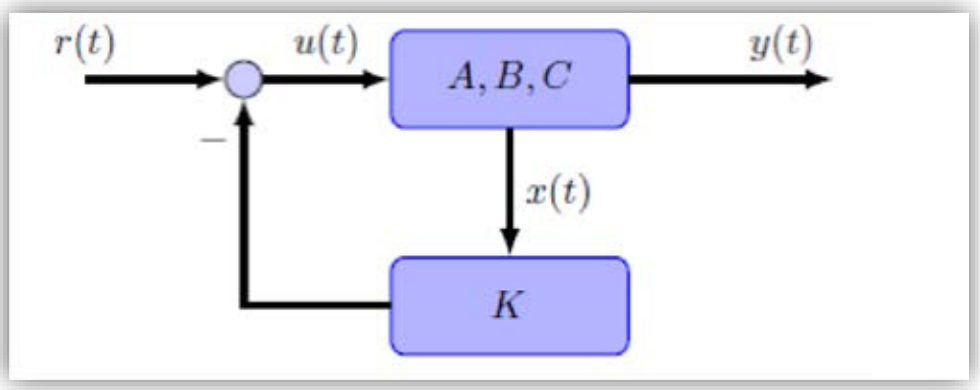


Figure 2.6: full-state feedback controller.[1]

2.3 Non-Linear Control

The scientific literature shows that for LTI systems, the controller's design may result in a simple task, but in front of a non-linear system, some other more complex techniques should be considered, with particular attention to the system's stability. The most common non-linear control techniques[7] are:

- Linearization (approximation);
- Feedback Linearisation;
- Lyapunov Stability;
- CPWL (Continuous Piecewise Linear Approximation).

It has been shown that aircraft dynamics can be linearized around equilibrium points (or trim conditions). A commonly used aircraft control methodology leverages this fact by designing a finite number of linear controllers, each corresponding to a linear model of the aircraft dynamics near a design trim condition. The key motivation in this approach is to leverage well-understood tools in linear systems design. [5]

Let $A_i, B_i, i \in \{1, \dots, N\}$ denote the matrices containing the aerodynamic and control effectiveness derivatives around the i^{th} trimmed condition $\bar{x}_i$. Let $X_1, \dots, X_N$ be a partition of the state space, i.e., $U_{i=1}^{N} X_i = \mathbb{R}^n, X_i \cap X_j = \emptyset$ for $\neq j$, into regions that are "near" the design trim conditions; in other words, whenever the state x is in the region X_i, the aircraft dynamics are approximated by the linearisation at $x_i \in X_i$. Then, the dynamics of the aircraft can be approximated as a state-dependent switching linear system as follows:

[7] Those techniques are briefly enunciated in the "Appendices section".

$$\dot{x} = A_i x + B_i u\,, \qquad when\ x \in X_i$$

(Equation 21)

The idea in a gain scheduling based control is to create a set of gains "K_i" corresponding to each of the switched models and apply the linear control $u = K_i x$. Contrary to intuition, however, merely ensuring that the i^{th} system is rendered stable (that is, the real parts of the eigenvalues $A_i + B_i K_i$ of are negative) is not sufficient to guarantee the closed-loop stability of "Equation 21". A Lyapunov based approach can be used to guarantee the stability of the closed-loop when using a gain scheduling controller. [5, 6 and 7]

Consider the following Lyapunov candidate:

$$V\big(x(t)\big) = x(t)^T P x(t)$$

(Equation 22)

where P is a positive definite matrix, that is, for all $x \neq 0,\ \ x^T P\, x > 0$. Therefore, $V(0) = 0$, and $V(x) > 0$ for all $x \neq 0$, making V a valid Lyapunov candidate. The derivative of the Lyapunov candidate is:

$$\dot{V}(x) = \dot{x}^{\,T} P x + x^{\,T} P \dot{x}$$

(Equation 23)

For the i^{th} system, "Equation 23" can be written as:

$$\dot{V}(x) = (A_i x - B_i K_i x)^T P x + x^T P (A_i x - B_i K_i x)$$

(Equation 24)

Let $\bar{A}_i = (A_i x - B_i K_i x)$; then from Lyapunov theory, it follows that for a positive definite matrix "Q" if for all i

$$\bar{A_i}^T P + P \bar{A_i} < -Q$$

(Equation 25)

then:

$$\frac{\partial V(x)}{\partial x} f(x) < -x^T Q x$$

(Equation 26)

In this case, "$V(x)$" is a standard Lyapunov function for the switched closed-loop system establishing the equilibrium's stability at the origin. Therefore, the control design task is to select the gains K_i such that "Equation 25" is satisfied. One way to tackle this problem is through the framework of "Linear Matrix Inequalities" (LMI) [5]. It should be noted that the

condition in “Equation 25” allows switching between the linear models to occur infinitely fast, this can be a reasonably conservative assumption for most UAV control applications. This condition can be relaxed to $\bar{A_i}^T P + P\bar{A_i} < -Q_i$ for $Q_i > 0$ to guarantee the stability of the system if the system does not switch arbitrarily fast. A rigorous condition for proving the asymptotic stability of a system of the form of “Equation 21” was introduced in [6].

Let $V_i, i \in \{1, \dots, N\}$be Lyapunov-like functions, i.e., positive definite functions such that $\dot{V}_i(x) < 0$ whenever $x \in X_i \backslash \{0\}$. Define $V_i[k]$ as the infimum of all the values taken by Viduring the $k-th$ time interval over which $x \in X_i$. Then, if the system satisfies the sequence non-increasing condition $V_i[k+1] < V_i[k]$, for all $k \in \mathbb{N}$, asymptotic stability is guaranteed. [5]

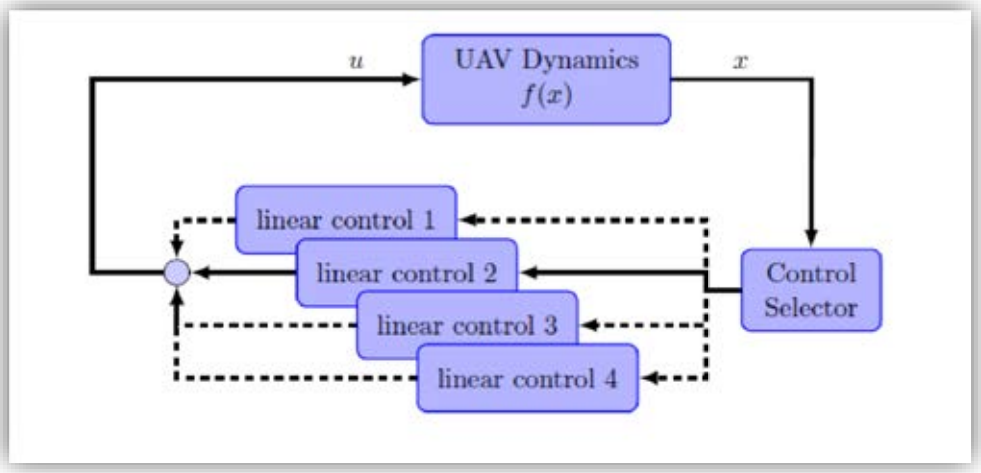


Figure 2.7: schematic of a gain scheduled scheme for UAV control[8]. [5]

The above example illustrates the use of Lyapunov techniques in synthesising controllers. In general, given the system $\dot{x}(t) = f\big(x(t)\big)$, and a positive definite Lyapunov candidate, the Lyapunov synthesis approach of creating robust exponentially stable controllers can be summarised as follows: let $\dot{V}(x) = g(x,u)$, find a control function $u(x)$ such that $\dot{V}(x) < -\epsilon V(x)$ for some positive constant ε.

Robust methods for Lyapunov based control synthesis are discussed in [8]. Furthermore, [9] provides an output feedback control algorithm for a system with switching dynamics. Linear Parameter Varying (LPV) systems’ framework leads naturally to the design and analysis of controllers based on a UAV dynamics representation via linearisation across

[8] The controller’s selector box decides which linear controller is to be used based on measured system states.

multiple equilibria. Gain scheduling based LPV control synthesis techniques have been studied for flight control, and conditions for stability have been established. [5]

2.4 Practical Non-Linear UAV control strategy, CPWL Mathematical model and controller approximation

Moving from the previous description of a few strategies for AUAV control design, in order to evaluate the complexity of a non-linear AUAV control, the “CPWL control strategy” is taken as an example.

Achieving a CPWL control algorithm may be a very intricate task, and it is the final result of many working steps. The first step is to create a mathematical model capable of describing the system’s dynamics during the flight operation. The second step is the test of those mathematical models through the use of powerful simulations software (e.g. “Matlab”). The use of comprehensive simulations increases the reliability of mathematical models and, supports the transition to the design of CPWL control algorithms. For a CPWL control algorithm, it is intended a control algorithm that is achieved according to the “continuous piecewise linear models” principles and implementation technique.

The equations previously shown are at the base of the mathematical models that describe the AUAV flight dynamics. Models have been developed by [2] for controlling a specific aircraft adapting the basic flight equations. These models are tailored to the physical characteristics of the vehicle. Indeed, this operation requires the investigation of all the unique coefficients associated with the vehicle, which are indispensable for the flight control equations’ solution.

For the mathematical modelling, the Author appreciates the use of “Maplesoft Maple” (a commercial computer algebra system developed and sold commercially by Waterloo Maple Inc.). This affirmation is motivated by the capability of “Maplesoft Maple” to manage “symbolic computations”. In fact, once that all the equations are written, it is possible to generate the “CompleteSystem”, as shown in the “Figure 2.8” example.

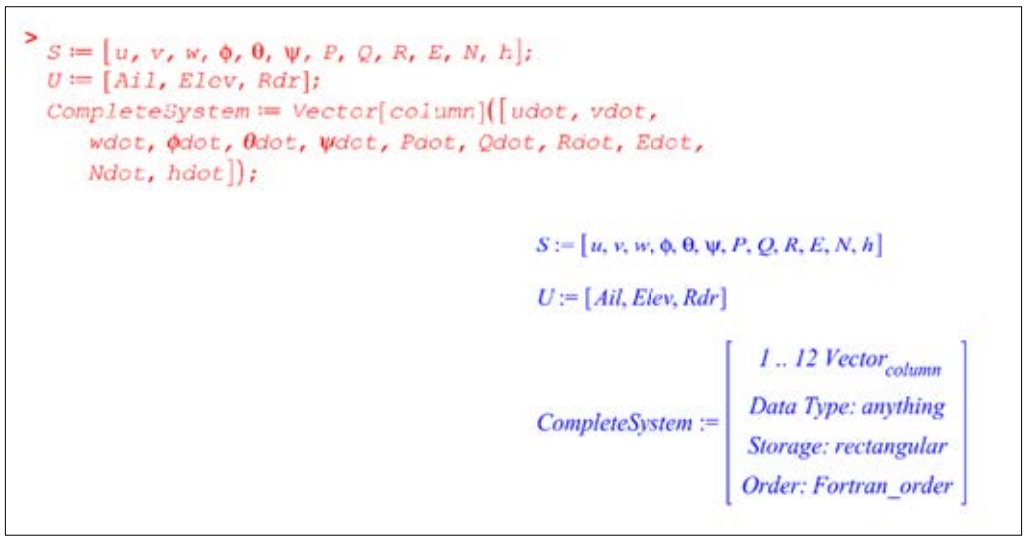


Figure 2.8: "Maplesoft Maple" code extract.

According to the theory of the non-linear differential system shown before, the CPWL controller design passes by the controller's "Affine Function" (called "F"). This function is determined through a series of iterations, as illustrated in Figure 2.9.

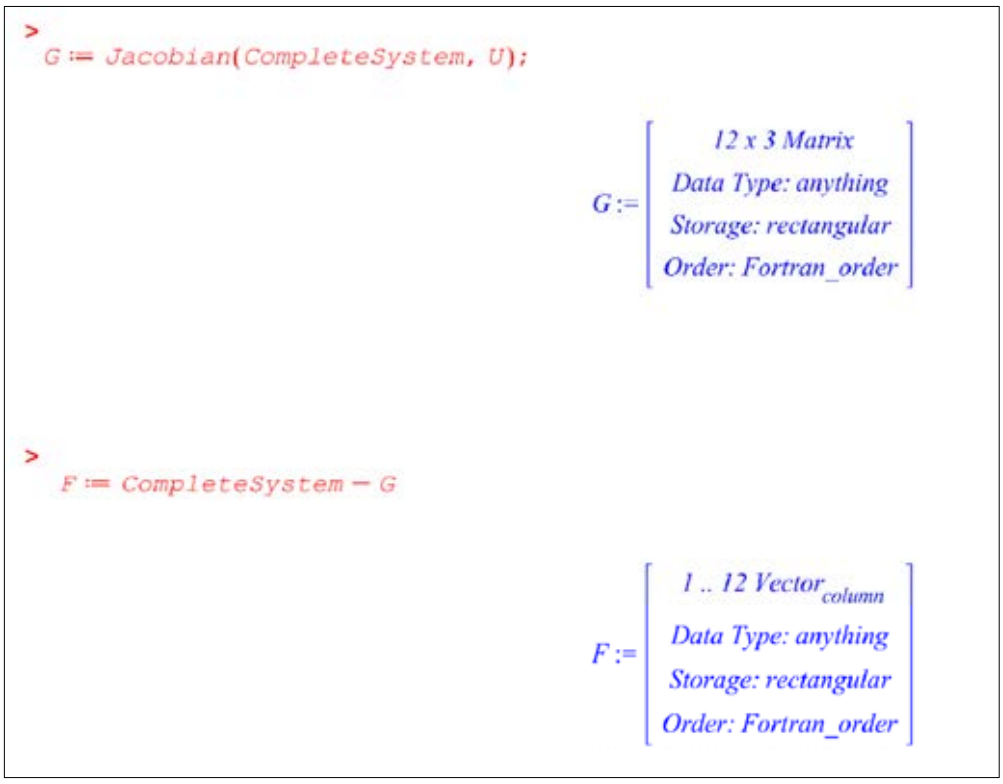


Figure 2.9: MAPLE code extract.

The function "F" is the output generated by "Maplesoft Maple" and should be exported to Matlab, which will use that function as input for the generation of the CPWL controller's parameter.

As the complete "Affine Function" could be very complicated, it may require a too-large computational power (Matlab may generate a control algorithm, which may result too heavy for a conventional MCU), and it is recommended to simplify the model that should produce a steepest simplification of the CPWL parameters.

The first simplification taken into consideration is to develop the "Affine Form" in the following equations.

$$C_D = (0.0027 \cdot C_l^2 + 0.017) + C_{D\delta} Elev$$

(Equation 27)

$$C_l = C_{L\beta} \cdot \beta + C_{L\delta A} Ail + C_{L\delta R} Rdr + \frac{b}{2V_t}\left[C_{Lp} P + C_{LR} R\right]$$

(Equation 28)

The purpose is to avoid quadratic or any other function different than linear (in the original system's controls, there is one control with a quadratic behaviour).

The next simplification operation may consist of the simplification of the winds reckonings and their coordinate transformation. The wind measures, taken in the coordinates $E\ N\ h$ and then transformed to $X\ Y\ Z$, present their influence on the relative velocity of the aircraft. The simplification consists of the assumption: the winds are not measured, and they appear in the form of white noise (this make it possible to delete the change of coordinates of the winds and work directly with winds in $X\ Y\ Z$). [2, 5, 10, and 11]

At this point, the simplified model can be exported[9] and may become Matlab's algorithm input. The "CPWL algorithm" is the result of a specific process, where a tailored "Matlab toolbox" (for the CPWL algorithm generation) computes the input mathematical model (Maple's exported model or "Matlab input"). The outcome, the "CPWL Algorithm", to be functional shall be translated into C-Code (or VHDL if it is used an FPGA), using the dedicated Matlab's tool. Generated C-Code is raw and needs a final configuration; this may vary in function of the MCU and its specific environment tool (such as MPLAB, IAR, Keil, etc.). The final step includes the MCU programming and debugging operation.

2.5 Neuro-Fuzzy Logic for UV and Robotic applications

When fuzzy systems become popular in industrial applications, many engineers perceived that developing a fuzzy system with excellent performance might not be a

[9] The model processed by "Maplesoft Maple", shall be exported in Matlab through a dedicated GUI's tool.

straightforward task. The problem of attaining membership functions and appropriate weighted rules is frequently a laborious and exhausting process of attempts and errors. This lead to the idea of employing learning algorithms in fuzzy systems. The neural networks that have robust learning algorithms were offered as an alternative to automate or assist the development of tuning fuzzy systems.

Methods for fine-tuning the fuzzy logic controllers are an evolution of fuzzy logic. A neuro-fuzzy controller uses the neural network learning techniques to tune the membership functions while preserving the semantics of the fuzzy logic controller intact. In conclusion, neural networks offer the possibility of solving the problem of tuning.

2.5.1 AI, Learning-Based Control

"Learning-Based Control" (LBC) is an alternative approach to control an Unmanned Vehicle and could be based on a hybrid neuro-fuzzy network tuned by a genetic algorithm. Practically the system uses the available parameters, or digitally processed differential parameters, as inputs of a specific fuzzy membership function.

The fuzzy system (MIFs, MOFs, FIS, etc.) could be processed in a dedicated neuro-fuzzy network. Subsequent interaction of neuro-fuzzy modules and genetic algorithms produce a neuro-fuzzy controller tuned by a genetic algorithm[10]. The training system produces a more elaborated and accurate "Fuzzy Inference System" (FIS).

Usually, a hybrid neuro-fuzzy controller uses a combination of several layers, and only two of them will be entirely readable because inner layers, as usual for a neuro-fuzzy system, are the neuro-fuzzy hidden layers.

The first layer contains all "Membership Input Functions" (MIFs), which for a UAV controller could be divided in:

- Flight dynamics membership functions;
- Trajectory membership functions;
- Energy estimation membership functions.

[10] Such iteration is the training process of the fuzzy logic control unit, by a genetic algorithm.

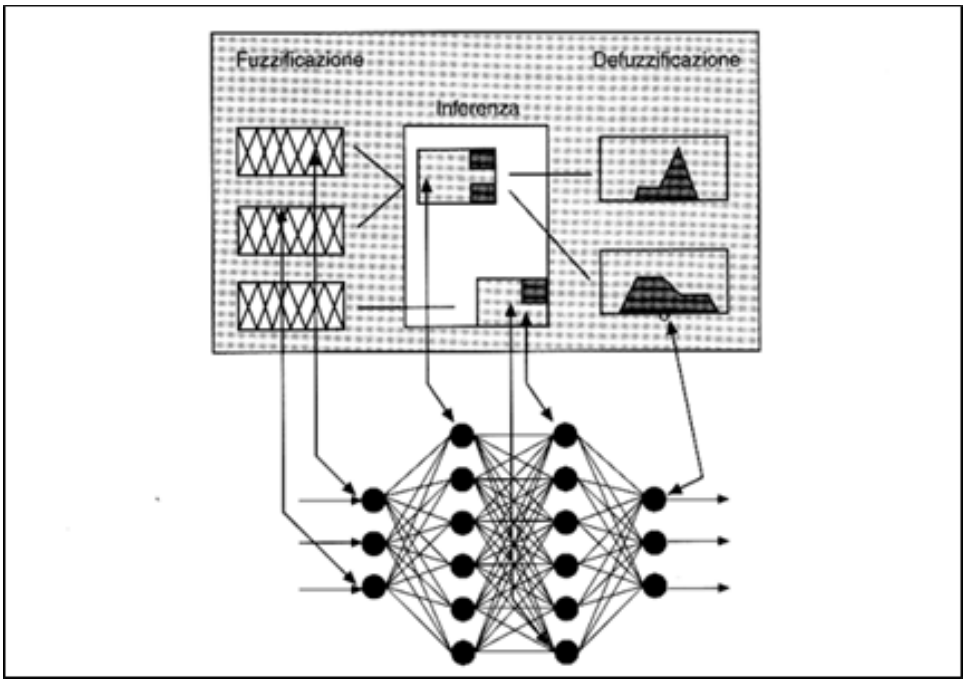


Figure 2.10: neural network blocks scheme representation.

Each MIF activates a specific group of neurones, and each membership function activates the "Status neurone"; neuro-fuzzy training (achieved into hidden layers) defines features and weights of the second layer neurones network.

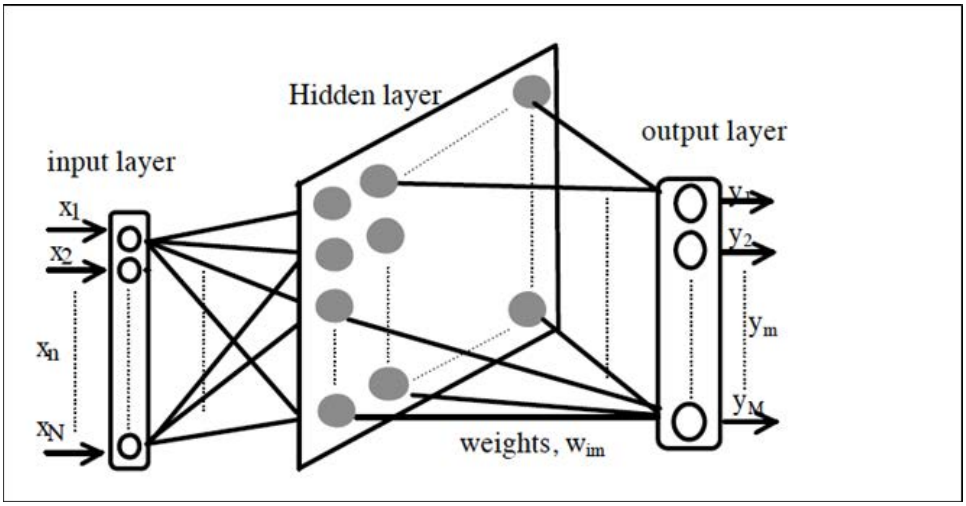


Figure 2.11: hybrid neuro-fuzzy network, hidden layer representation.

The second network (described in "Figure 2.11") activates the output layer, which may be associated with an array of actuators and with a powertrain. The successive step of the controller's design process is the definition of the "defuzzification method". The designer shall select the most appropriate between various methods and techniques; the most commonly used is the "Center of Area" method.

2.5.2 *Data capture for the training process*

There are many methods and techniques to achieve the learning system; for the study case, one robust strategy could be the collection of data during several human-piloted flights.

Data collection could be achieved as the UAV is guided via remote control (RC) through an area defined as the "selected environment", which is either the actual environment in which the UAV will operate or is a near facsimile of that environment. Therefore, appropriate hardware is required to provide the RC aspect of the learning process. The operator guides the UAV through a series of flight manoeuvres (such as the take-off, landing, and any other significant manoeuvre) to establish a base and bias pattern for the algorithm. The on-board data collection algorithm should be designed to capture both digital and analogue readings from the UAV sensor array during the allocated learning period "T_{learn}", during which time the received data is stored within the on-board memory chip. "T_{learn}" must not exceed the MOB maximum as in Equation 29. [12 and 13]

$$T_{learn} < \left(\frac{MOB \times 10^6}{\left(\left(Sen_d \times 1 \right) + \left(Sen_a \times 2 \right) \right) \times S_{ps} \times S} \right) Minutes$$

(Equation 29)

Where:

T_{learn} = Maximum run time.

SEN_d = Digital Sensors (1 or 2 bytes per sample).

SEN_a= Analog Sensors (generally 2 bytes per sample).

S_{ps} = Samples per second.

S = Number of seconds.

MOB = On-board memory Mbytes.

Collected data consists of raw sensor data from the UAV sensor array and should be processed through a selected genetic algorithm[11]. The process might be referred to a batch learning because it could be analysed using the "Delta Rule Method" after the data has been collected. The "Delta Rule" is illustrated by "Equation 30" and, in its purest form, as pronounced by [14].

[11] An example case may be the Aforge.net C# framework, which is purpose-designed for developers and researchers in the fields of Artificial Intelligence, neural networks, genetic algorithms, machine learning and robotics among other things)

$$\Delta W_{ij_x} = -\varepsilon \frac{\delta E}{\delta_{W_{ij}}} = \varepsilon \delta_{\alpha_{i_x}}$$

(Equation 30)

Scrutinising "Equation 30", it can be seen that the change in any particular weight is equal to the products of:

- the learning rate ε;
- the difference between the target and actual activation of the output node δ;
- the activation of the input node associated with the weight in question.

A higher value for ε will inevitably result in a greater magnitude of variation. Because each weight update has a physical limit for the error's reduction, many iterations are required in order to minimise error satisfactorily. In batch mode, the value of "Equation 31",

$$\frac{\delta E_p}{\delta_{W_{ij}}}$$

(Equation 31)

is calculated after each sample, and it is submitted to the network with the total derivative equation ("Equation 32") calculated at the end of an iteration by summing the individual pattern derivatives. When this value is calculated, it is possible to update the weights.

$$\frac{\delta E}{\delta_{W_{ij}}}$$

(Equation 32)

As long as the learning rate epsilon is small, batch mode approximates gradient descents. [12, 13 and 15]

3 Typical Architectures for Electric Vehicles

Autonomous Unmanned Vehicles (AUV) development (and evolution) is rigorously connected to the vehicles electrification process. Part of the Author's researches also analyses this process, using automotive applications as a significant study case.

The automotive market's electrification process is in constant growth, and the expectation is that the incoming regulations will strengthen the trend. In fact, across the globe, numerous cities and countries are opting for severe "Diesel Combustion Engines" limitation in the urban areas and, in a few cases, some substantial limitations as well as for "Petrol Combustion Engines". The E-Mobility process is going to affect the whole urban transport (private and public), this means that in the function of the vehicle's application a specific architecture will be used in order to satisfy the new regulations at the lowest production cost.

3.1 Automotive Regulations/Standards Overview

Scrutinising the EV/HEV electrical architectures and the annexe legislation, the most relevant regulation to highlight is the "UN/ECE-R100". This regulation defines the main requirements in terms of electrical safety and makes a clear architecture distinction in the function of the vehicle's battery configuration (topology, technology and voltage). It is imperative to highlight the following scopes of UN/ECE-R100:

- safety requirements with respect to the electric powertrain of road vehicles of categories M and N1, with a maximum design speed exceeding 25 km/h, equipped with one or more traction motor(s) operated by electric power and not permanently connected to the grid, as well as their high voltage components and systems which are galvanically connected to the high voltage bus of the electric power train;[12]
- safety requirements with respect to the "Rechargeable Energy Storage System" (REESS), of road vehicles of categories "M" and "N" equipped with one or more traction motors operated by electric power and not permanently connected to the grid.[13]

[16]

[12] UN/ECE-R100 Part I - this regulation does not cover post-crash safety requirements of road vehicles. [16]
[13] UN/ECE-R100 Part II - this regulation does not apply to REESS(s) whose primary use is to supply power for starting the engine and/or lighting and/or other vehicle auxiliary's systems. [16]

Automotive legislation, regulations and standards are in constant evolution in order to stay in line with the continuous technological progression of the automotive industries. Modern automotive standards compliance is becoming day by day more challenging due to the increasing complexity of new vehicles (especially for HEVs and EVs), due to the amount of "computational power"[14] installed on the new vehicles and due to the safety role associated with the functions performed by the vehicle's ECUs. Particularly relevant are the new "ADAS" functionalities installed on modern vehicles.

In this scenario, ISO 26262, which is an adaptation of IEC 61508, compliance represent a goal that every major automaker cannot miss, even more, for what is inherent to the security of new HEV/EV.

A wide range of variables influence the "SECURITY LEVEL", which "ISO 26262" defines as "ASIL LEVEL", which shall be associated with each ECU's and system's function. The target "ASIL LEVEL" imposes design rules, component level selections guidelines, software validation procedures, hardware tests and hardware validations protocols. The highest "SECURITY LEVEL" is defined as "ASIL LEVEL D" (represents the highest safety standards for automotive components/functions), generally associated with particular critical[15] functions.

It follows a short resume of a few prominent automotive standards.

[14] The amount of data exchanged across the vehicle between the vehicle's ECU is exponentially increasing. It is due to two factors: the increasing number of ECUs installed on the vehicle and the increasing amount of data broadcasted by the vehicle's ECUs. It is essential to annotate that each ECU is fitted with at least one modern automotive MCU (continuously growing in terms of computational power and functionalities) and, often with a combination of multiple MCUs and FPGA. It is obvious to highlight the safety challenge to secure such amount of data exchanged between within each ECU and across the vehicle.
[15] Breaking, Torque Control, Power Steering, etc.

Reference	Title	Published
ISO 26262	Road vehicles – Functional safety	2011
IEC 61508	Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES)	1994
ISO 16750	Road vehicles—Environmental conditions and electrical testing for electrical and electronic equipment	2006
IEC 60950	Information technology equipment - Safety	2005
ISO 12405	Electrically propelled road vehicles — Test specification for Lithium-Ion traction battery packs and systems	2011
IEC 62660	Secondary Lithium-Ion cells for the propulsion of electric road vehicles	2010
UL 2580	Batteries for Use In Electric Vehicles	2013
SAE J2464	Electric and Hybrid Electric Vehicle Rechargeable Energy Storage System (RESS) Safety and Abuse Testing	2009
SAE J2929	Electric and Hybrid Vehicle Propulsion Battery System Safety Standard - Lithium-based Rechargeable Cells	2011
IEC TS 61851	Electric vehicle conductive charging system	2017

Table 3.1: automotive standards summary.

3.2 Low Voltage Electric Vehicles Architecture

[16] clearly defines the "Low Voltage Architecture" as the system that utilises a REESS with a maximum operative voltage below 60 V_{DC}.

This architecture is predominant in the ultra-light, light and low power vehicle's segment. Its simplicity and its superior cost-efficiency make this architecture the most popular solution for vehicles having a powertrain's power rating below 15 kW.

This configuration's main advantage is: it is not required reinforced galvanic isolation between the REESS and the vehicle's chassis[16]. Below a "Low Voltage Architecture" representation[17].

[16] Since that vehicle's chassis is electrically connected to the negative pole of the secondary battery (12V or 24V), any electrical device, such as vehicle's ECUs, installed are referenced to the vehicle's chassis.
[17] The block diagram refers to a conventional system architecture applicable to light and low-power vehicles.

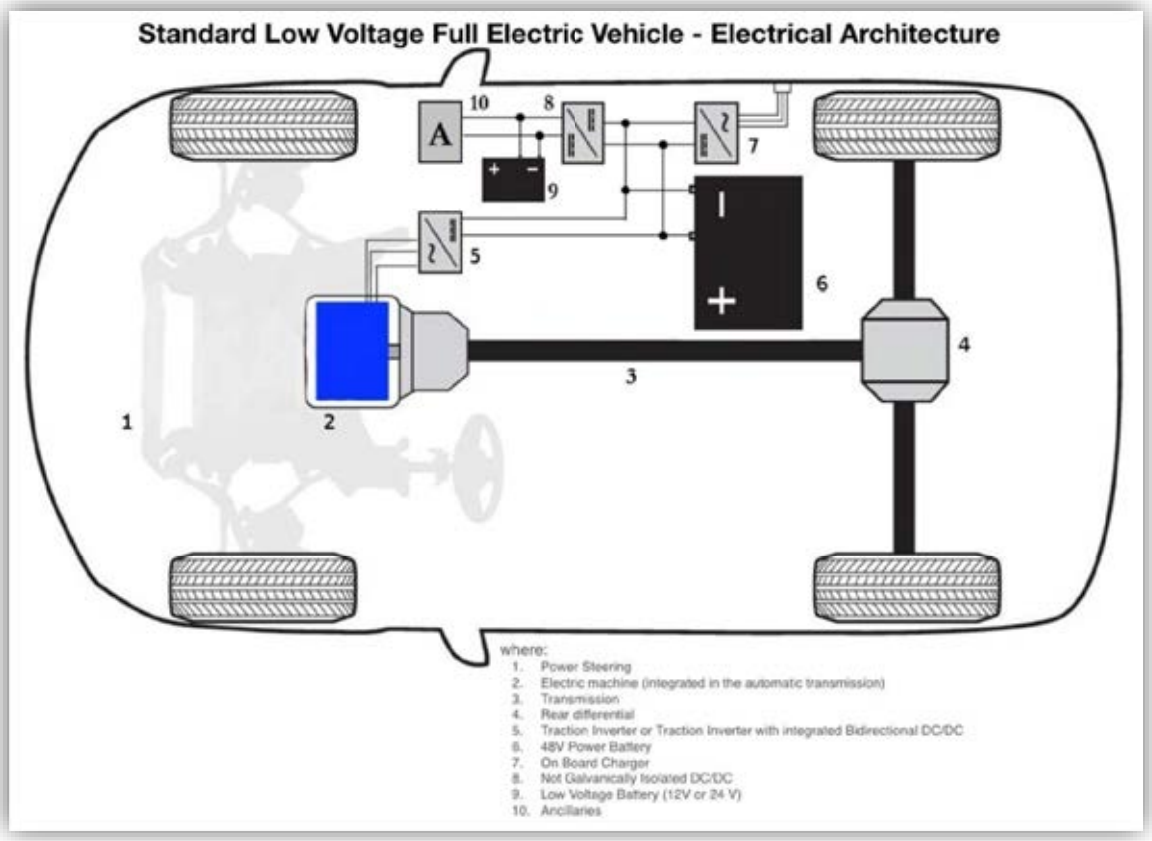


Figure 3.1: EV "Low Voltage Architecture" representation.

3.3 High Voltage Electric Vehicles Architecture

[16] allows the use of the "High Voltage Architecture" for passengers and commercial vehicles under strict conditions. Where "High Voltage" means the classification of an electric component or circuit, if its working voltage is greater than 60 V_{DC} ([16] does not allow the use of working voltage above 1500V_{DC}) or greater than 30V_{ac} ([16] does not allow the use of working voltage above 1000V_{ac}).

[16] makes mandatory the introduction of reinforced galvanic isolation[18] between the "High Voltage REESS" and the vehicle's chassis on which is connected the negative battery pole of the vehicle's "Low Voltage Battery" (thus each vehicle's "Low Voltage" device and circuit).

The main advantages of operating in "High Voltage" are:

[18] Please refer to UN/ECE-R100 (particularly to the section 5.1.3.2) and to the IEC60950.

- reduced current flow and the relative use of small cable cross-section[19];
- increased E-Motor performances (in particular power density and efficiency);
- E-Motor size/weight reduction.

What presented is the most popular architecture for EVs in the segment of passenger vehicles and heavy-duty with a powertrain power rating above 15kW. Currently, there are many REESS designs on the market; most commons are:

- approximately $200V_{DC}$ "NiMH Battery" with a bidirectional DCDC that generally fixes powertrain's operative voltage up to 600VDC;
- $200V_{DC}$ to $400V_{DC}$ "Lithium-Ion Battery" directly connected to the powertrain;
- $200V_{DC}$ to $400V_{DC}$ "Lithium-Ion Battery" with a bidirectional DCDC that generally fixes powertrain's operative voltage up to 800V;
- $600V_{DC}$ to $820V_{DC}$ "Lithium-Ion Battery" directly connected to the powertrain.

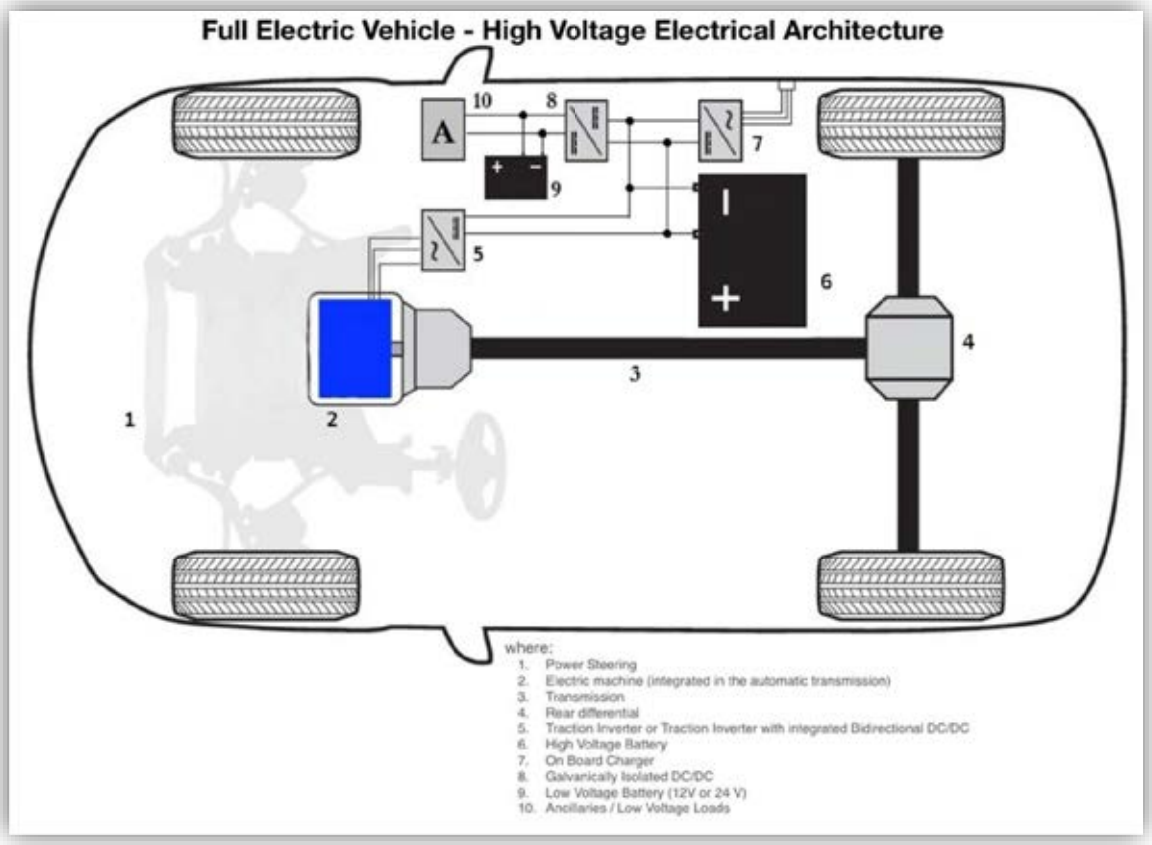


Figure 3.2: EV "High Voltage Architecture" representation.

[19] Considerable advantages in terms of weight reduction, power cables cost reduction and utilisation of power cables with considerable higher mechanical bending flexibility;

3.4 Hybrid Architecture variants

The HV evolution during the last decades brought to the automotive market a few architecture's variants. This process has its roots in the always popular objective for the automotive OEMs industries: vehicle cost optimisation. In truth, what may be effective in a particular vehicle's segment most likely will not give any economic advantage for a vehicle of a different segment. Typical HV topologies are:

- Full Hybrid (or Hybrid Electric Vehicle - HEV);
- Mild Hybrid (MHEV);
- Hybrid Plug-In (PHEV).

3.4.1 Hybrid Electric Vehicle (HEV) Overview

A Hybrid Electric Vehicle (HEV) uses the combined efforts of both a combustion engine and a battery-powered E-Motor to drive the vehicle. The work of driving the vehicle is distributed between the two propulsion sources in the best way possible at any given time.

For instance, the E-Motor can give the vehicle a boost of power, feasibly while climbing a hill, without burning additional fuel. The vehicle may also be able to drive for brief periods solely in EV mode, with ICE switched off.

The electric motor's power is produced by a built-in generator, or by the traction E-Motor, and then stored in a REESS. In an HEV, all power is generated on-board, and there is no plugging-in possible. The system charges the battery in two ways. Firstly, the combustion engine drives the electric motor (or a dedicated generator) to charge the battery. The second method is through regenerative braking, where the system utilises the E-Motor as a "power generator".

By definition, a Full Hybrid or HEV utilises a "High Voltage REESS" and complies with the safety standards and automotive regulation previously disserted (ISO26262, [16], IEC60950, etc.). At the moment, this is the most popular Hybrid topology; the best example for this topology is the TOYOTA Hybrid Synergy Drive (HSD)[20] architecture.

[20] Hybrid Synergy Drive (HSD) is the Toyota Motor Corporation hybrid car drivetrain technology brand name. It is the most popular hybrid system in the world and has sold more than 12 million units since the Prius was launched in Japan in August 1997. Currently, each year Toyota Group produces more than 1.5 million hybrid vehicles. HSD technology produces a full hybrid vehicle which allows the car to run on the electric motor only, as opposed to many other brand hybrids which cannot and are considered mild hybrids. The HSD combines an electric drive and a continuously variable transmission (CVT). The Synergy Drive is a drive-by-wire system with no direct mechanical connection between the engine and the engine controls: both the gas pedal/accelerator and the gearshift lever in an HSD car merely send electrical signals to a control computer. HSD is a refinement of the original Toyota Hybrid System (THS).

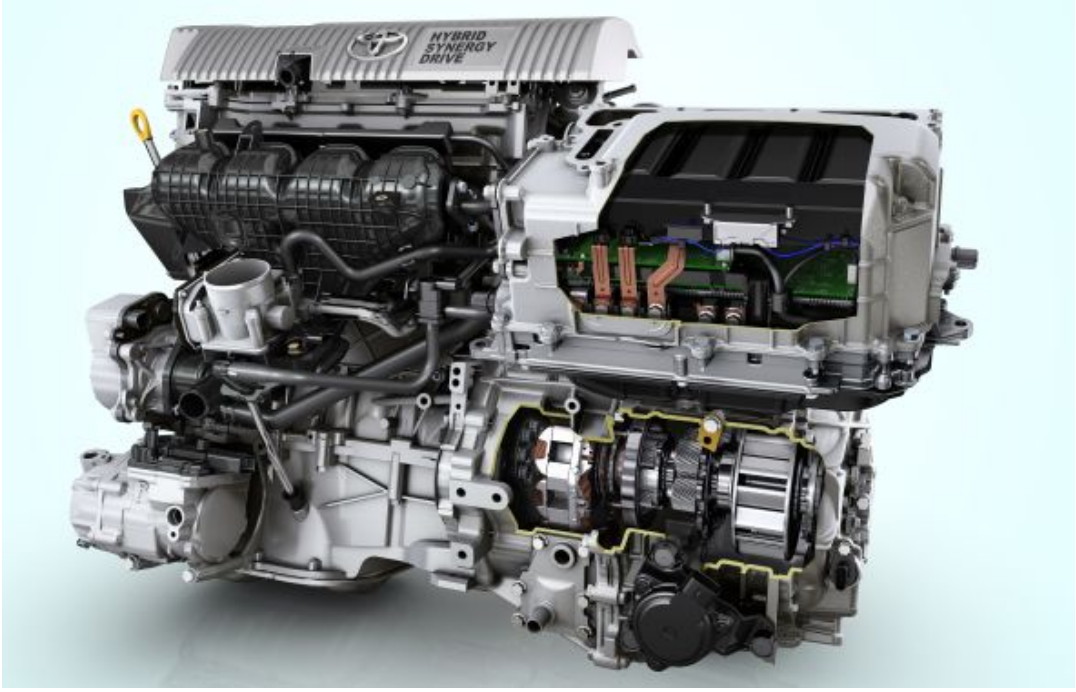


Figure 3.3 TOYOTA's Hybrid Synergy Drive.

Since 1997, Toyota pioneered a full hybrid system that consists of six primary components:

a) Internal Combustion Engine (ICE);
b) CVT Gearbox;
c) electric motor;
d) electric generator;
e) power control unit;
f) power split device that uses a particular type of gearbox to smoothly distribute power from the ICE, electric motor and electric generator;

As a complete system, HSD is a bright, reliable, and robust fuel-saving technology that seamlessly (and automatically) switches between electric power and conventional engine power. Proficient in adapting to different driving conditions, HSD effectively controls the power coming from both sources and tells the car how to combine them for the highest efficiency and performance.

As its name suggests, the system delivers tangible synergy between the two power sources. When the engine is running, it charges the battery via the generator; when driving conditions allow it, the generator can cut out the ICE and let the electric motor take over for zero-emissions travelling. The sophisticated engine management system can sense when the

car is stopped and will switch off the engine to conserve power and cut emissions, automatically starting up again when needed.

Traction control diverts energy back to the battery, where it is recycled, every time the user requests a vehicle deceleration. Instead of the energy being lost as heat or noise from the brakes, it is captured and used to power the electric motor later. It is incredibly efficient in stop-start traffic, where the system recovers and stores a considerable amount of energy, increasing the vehicle's efficiency. This approach, as well, reduces the emissions of microparticles generated by the mechanical interaction between brake pads and brake disks.

The HSD's peculiarity is the particular ICE with a slightly different engine cycle than the conventional Otto-type four-stroke cycle. Called the Atkinson cycle[21], this modified four-stroke cycle produces less heat, and it is, consequently, more efficient.

A general-purpose HEV system that may be a great study case is the ZF System developed for various car manufacturers. As may be observed in Figure 3.5, this solution is very flexible and aims to efficiently adapt the conventional FR-ICE vehicle structure (front engine-rear traction vehicle) to an HEV. In contrast with HSD, this solution uses only one electric motor embedded in an integrated transmission. This integrated transmission, which includes an E-Motor and an automatic gearbox, is installed between the ICE and the drive shaft. This solution allows to mechanically disconnect the engine (and switch it off) from the transmission while the electric motor may, for example, perform a regenerative braking operation or, most simply, spin the drive shaft.

[21] The Atkinson-cycle engine is a type of ICE invented by J. Atkinson (1882) to increase fuel efficiency, although if compared to the conventional Otto cycle engine, it results capable of lower power density. Toyota's variable valve timing solution is a very reliable, fuel-efficient oriented design.

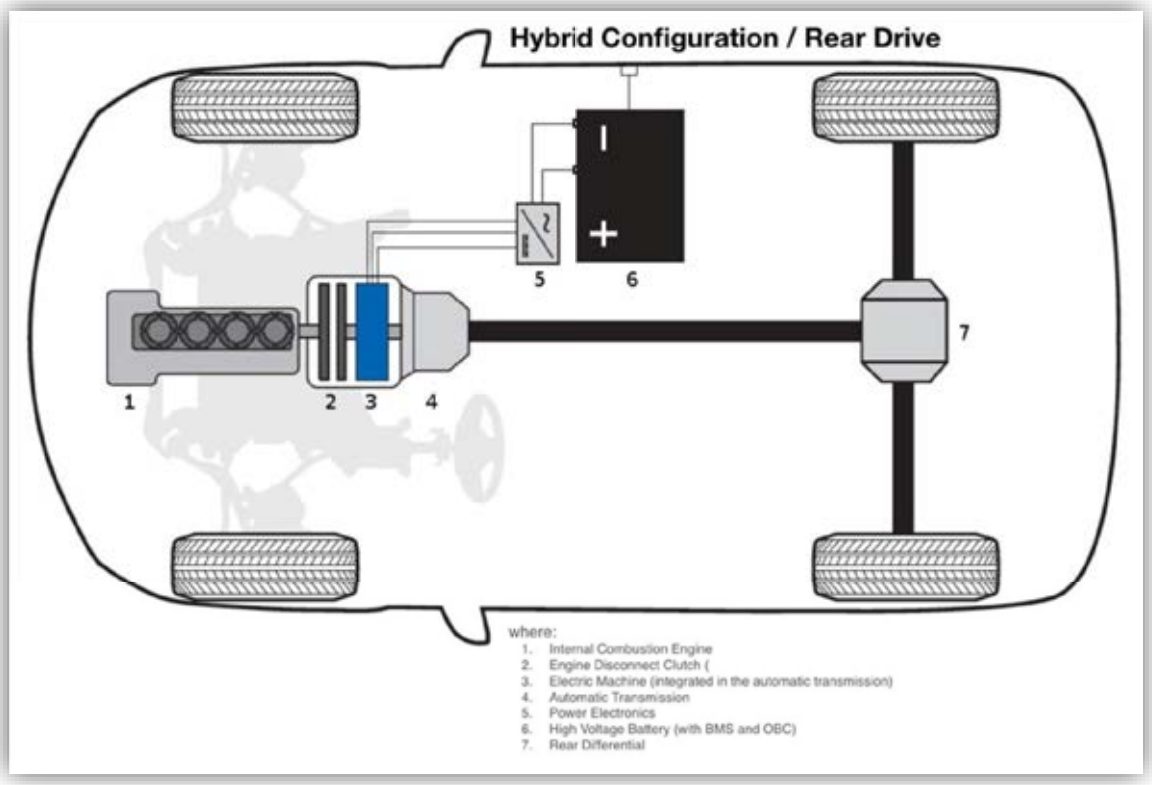


Figure 3.4: FR HEV configuration.

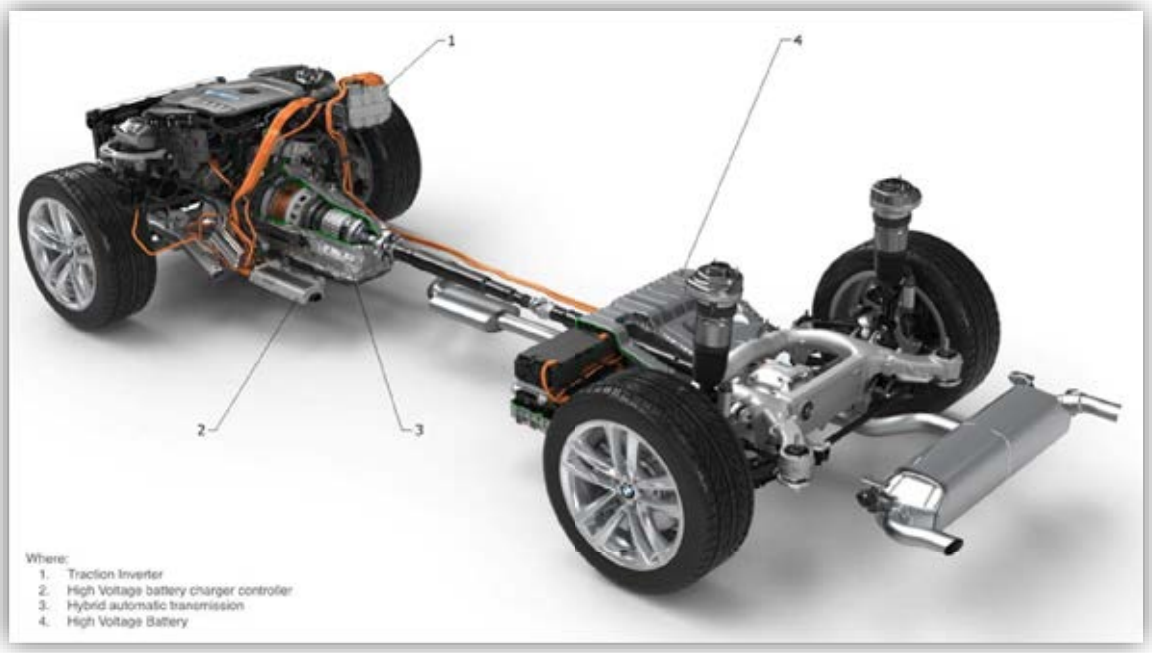


Figure 3.5: BMW HEV architecture.

3.4.2 Plug-In Hybrid Electric Vehicle (PHEV) Overview

Except for the fact of having a few core enhancements, PHEV architecture is very similar to the HEV architecture. In contrast with an HEV, the PHEV has an on-board battery charger (OBC), which allows the vehicle of being charged from the grid.

Usually, the REESS capacity of a PHEV is between the HEV and the EV REESS capacity, and this allows PHEV to drive in "zero-emission mode" for a considerable distance (depending on the vehicle may be above 40 km).

3.4.3 Mild Hybrid Electric Vehicle (MHEV) Overview

The operational principles of MHEVs are very similar to the operational principles of HEVs. The main difference between MHEV and HEV is the vehicle's REESS and the E-Motor's design strategy. By definition, an MHEV utilises a small "Low Voltage" REESS, while the HEV utilises a "High Voltage" REESS. The lower REESS capacity and relatively low operating voltage limit the power rating of the E-Motor, which is generally rated below the 15kW.

As for the previous example of "Low Voltage EV", the "MHEV" is usually a low-cost hybrid solution that generally targets the compact size vehicles market.

3.5 Example of 48V REESS with Boost Voltage Converters

Although not popular as the previous ones, a particular EV architecture is characterised by a "Low Voltage" REESS and a "High Voltage" E-Motor. It is gaining the attention of a few debates in the automotive industries due to the availability on the market of always more performing "WBG Power Semiconductors". New WBG devices, which can work with very high switching frequencies compatible with the planar Transformers/Inductors technology, make it possible to build very efficient small form-factors bidirectional DC/DC. This architecture's main advantage is to reduce the vehicle's hazards restricting the "High Voltage" operations only when the vehicle is in a "DRIVE MODE". As soon as the ignition key is "OFF", all DC/DC converters will be disengaged, and as soon as the "High Voltage DC-link capacitor" will be discharged, no "High Voltage" will be detected on the vehicle. What described represents a significant advantage for the vehicle service procedures and the passenger's safety in case of a vehicle's "fault" or "crash event".

As for the previous examples of "Low Voltage EV" and "MHEV", this specific study case may target the compact size vehicles market.

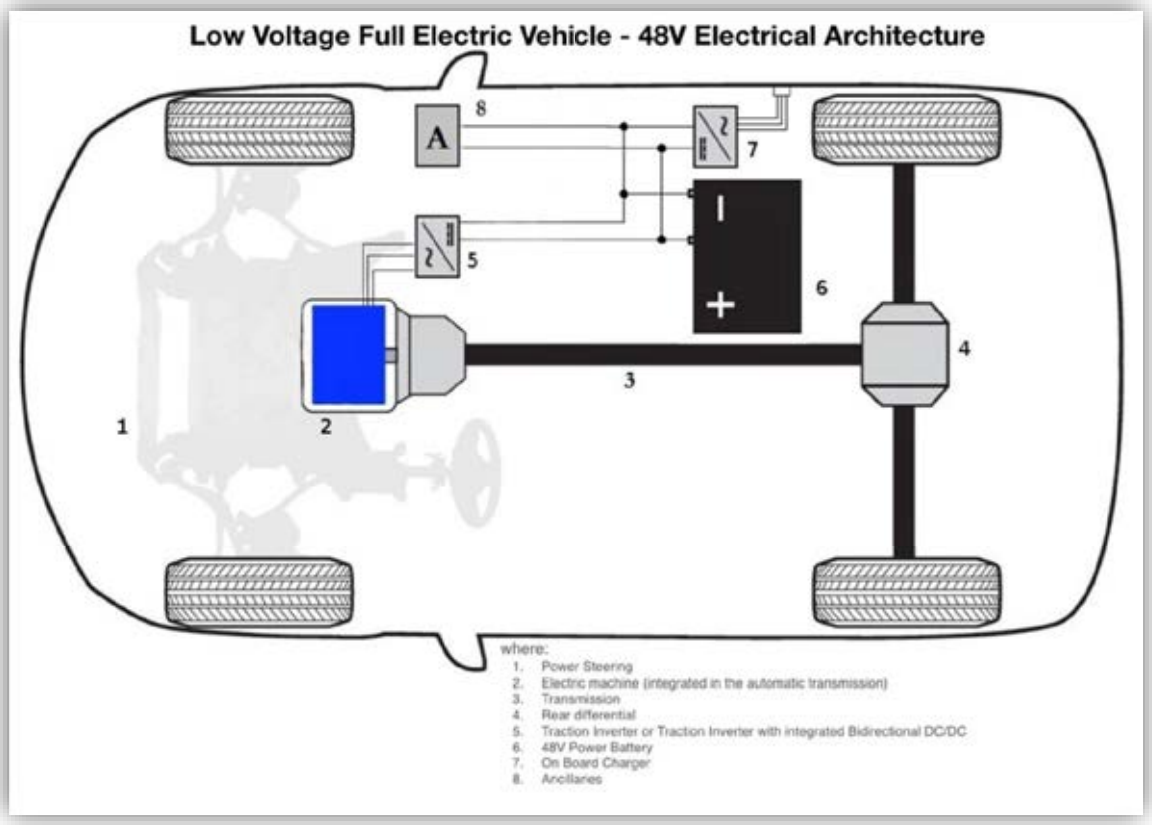


Figure 3.6: EV with a "Low Voltage" REESS and a "High Voltage" E-Motor.

3.6 Electric Vehicles Power Converters

Currently, e-Mobility (Vehicles) and Industrial Service Hybrid Robots (Factory ISH) applications are becoming progressively more accessible, and a significant evolution is occurring for the power electronics parts of these systems. Constant growth in terms of quantities of vehicles produced and power ratings of vehicles, emphasises the development of new, more efficient, more cost-effective and with higher power density traction inverters. It is possible to identify two primary design strategy: discrete power elements design and power module design.

3.6.1 Discrete Power Elements design

A low-cost oriented design would most likely look to a traction inverter design realised with discrete power elements. Indeed, this strategy will sacrifice the power density, life expectancy and the power electronics elements switching frequency favouring a cost-saving. Although, the installation on a PCB (as for Figure 3.7 and Figure 3.8) of many discrete components per switch is generally necessary to match the E-Motor power ratings [17].

A remarkable study case is the first generation of "TESLA" EV powertrain shown in Fig. 3.7, which uses 14 x IGBT discrete elements per every single inverter's switch in order to drive the required current to the mated E-Motor. To achieve such high current ratings, it is usually necessary to connect additional laminated bus bars to the PCB, which results in increased weight and higher vulnerability to vibration. To ensure the appropriate electrical and thermal connection between the PCB and bus bars, bespoke connections (joints) are usually indispensable[22]. [17]

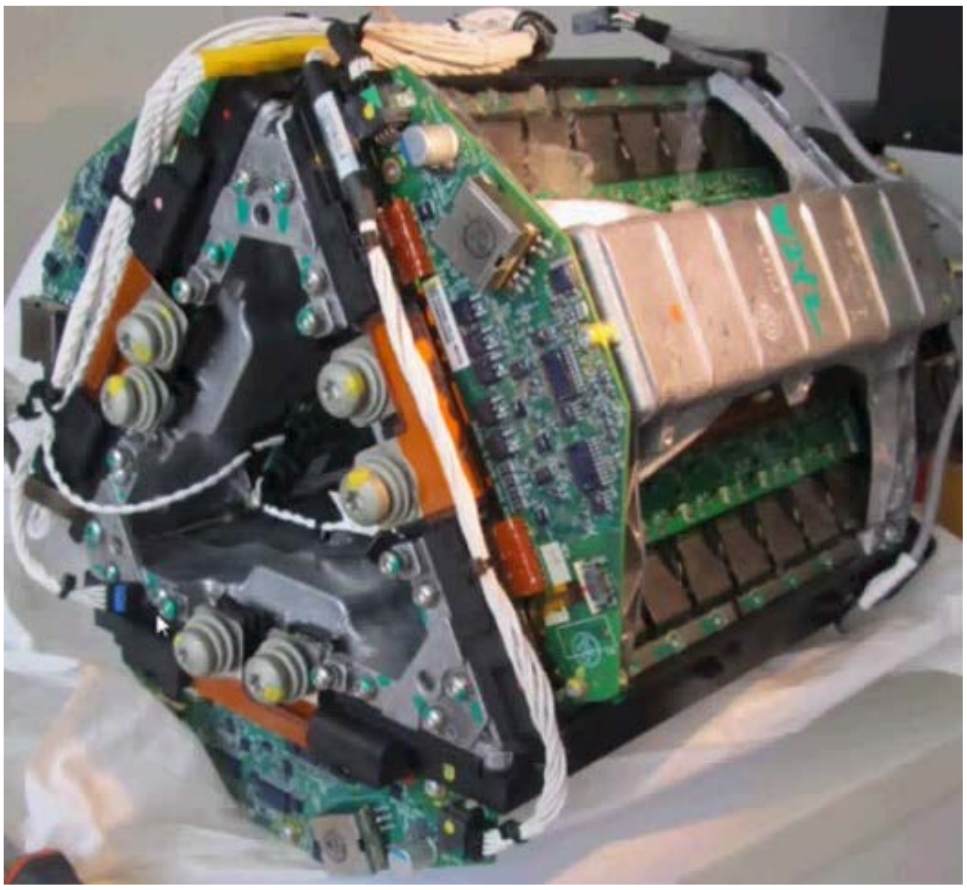


Figure 3.7: TESLA integrated powertrain.

The TESLA's powertrain displays a fascinating mechanical design. It utilises three "Power PCBs", one for each motor's phase (shown in Figure 3.8), with the motor positioned within the cavity achieved by the triangular formation of the "Power PCBs" (shown in Figure 3.7). On every "Power PCB", there are 28 IGBTs installed and, the electrical connection of these power element forms a half-bridge inverter. Each IGBT is thermally connected to its

[22] At each point, the pressure is achieved by a fixing element that ensures the contact between the surfaces of two conductive parts.

cold plate (heatsink), and the PCB applies a force to the cold plate in order to ensure the thermal connection between each IGBT and the cold plate.[23] [17]

Figure 3.8: TESLA Model S half-bridge inverter power PCB.

3.6.2 Critical issues on Discrete Power Elements Power PCB

E-Mobility vehicles must comply with a wide range of several safety requirements, ensure a minimum product lifetime, and provide fail-safe mechanisms in case of a fault. In the majority of cases, the vehicle's DCU and traction inverters can guarantee a high level of safety as well as being able to prevent catastrophic faults[24]. [17]

Studies to address the issues related to the forces that the Power PCB applies on the cold plate have been made by the Author. Typically, there is a "Thermal Material"[25] between the component and the cold plate, which needs to be compressed accordingly. A lack of compression will increase the thermal resistance between the component and cold plate, or additionally, excessive compression may trigger an insulation fault, as shown in Figure 3.9. [17]

[23] The action is performed by an optimised array of mechanical screws, accordingly selected. This ensures a uniform force applied from the top side of each IGBT to the cold plate.

[24] A fault on the Power PCB may occur in many circumstances, more likely due to a collision, lack of insulation or of some other external event.

[25] This material is generally a "Gap Pad" or "Thermal Paste", the advantage of using the "Gap Pad" lies on its reinforced galvanic isolation capability.

Nano-piezo electric sensors might act as safety feedback and monitor the existence of a correct thermal connection because it may detect if an appropriate compression is applied (power electronics devices to the cold plate[26]). [17]

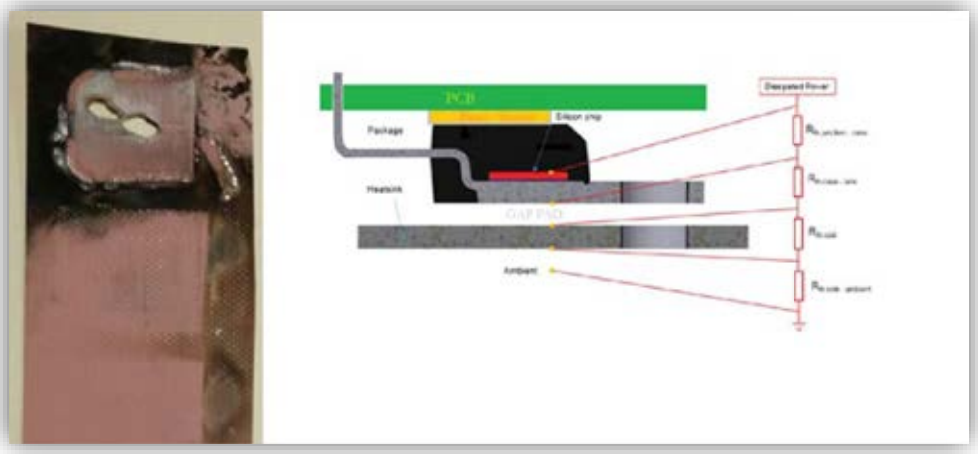


Figure 3.9: example of excessive compression fault [left] and vertical section of a study case "Power PCB" [right].

A Common "Power PCB" is manufactured in layers, constructed mainly by copper films and FR4's variants, and it is possible to bend the PCB at specific points when not symmetrical and not uniform forces are applied. Uniformity of force spread is necessary to prevent inappropriate mechanical stress on the PCB and achieve an excellent thermal connection between the cold plate and power components.

In this kind of PCB, a single element that does not establish a good enough thermal connection will most likely trigger a system fault. Consequently, a power stage fault may have catastrophic consequences on other systems directly connected with the traction inverter, such as the vehicle's REESS. [17]

3.6.3 "Power Module", Benefits for Electric Vehicle Power Converters

"Power Modules" based power electronics systems are the most popular design choice for modern EV, HEV, and PHEV. Its popularity has its roots in the following factors:

- limited space available;
- power requirements;
- traction inverter life expectancy (warranty).

[26] Thus, a contact IGBT's heat-sinks to the cold plate.

In order to reach a high switching frequency with low switching losses, a package design targets the lowest stray inductance for both the module and the system-level bus-bar design. A common approach is to use a low-inductance, overlapping planar structure.

“Direct Bonded Copper” (DBC) substrates have become an essential electronic circuit board for multichip power semiconductor modules. They replace complicated assemblies based on lead frames and refractory metallised substrates due to ease of assembly and the low-temperature coefficient of expansion of DBC, which matches silicon despite thick copper metallisation. The DBC technology allows the bonding of copper to alumina and aluminium nitride, fusing copper to copper has been developed to establish efficient water cooling devices with sophisticated internal microchannel structures for cooling power laser diodes and other high power density electronics. The continuous demand to satisfy more stringent requirements for temperature cycling reliability and mechanical stability in automotive, avionics and space applications result in more investments in new, more reliable and more performing DBC solutions. [18]

There are four different kinds of ceramic materials that can be bonded with copper foils by either the DBC or AMB (Active Metal Brazing) process. Each combination of these insulating materials and joining technologies meets specific demands and is suitable for different applications. Furthermore, thick printed copper may be an alternative solution for special applications. According to their physical properties, the costs of metallised substrates vary widely.

Because of the outstanding performance ratio against cost-efficiency, DBC aluminium oxide ceramic is the most commonly used substrate. For many industrial applications, the performance is sufficient to meet the lifetime and the thermal dissipation requirements. If higher mechanical performance is required, “ZTA” DCB offers an even higher bending strength, whereas thermal conductivity is comparable to “Al_2O_3” DCB (due to the direct bonding process, costs for production are in an acceptable range).

Joining copper foils on highly thermal conductive aluminium nitride can be done utilising AMB and DCB. However, DCB is a more cost-effective method, whereas “AlN” AMB shows enhanced thermal cycling performance. Because of the inherent low mechanical strength of “AlN”, the insulating layer needs to be thick compared to “Al_2O_3”, ZTA (Zirconia Toughened Alumina) and “Si_3N_4”. Furthermore, aluminium nitride DCB (especially AMB) is quite expensive and mainly used for very high voltage applications. Silicon nitride combines both superb mechanical properties and enhanced thermal conductivity. However,

prices for raw ceramic substrates are still high and, until now, AMB is the only joining technology applicable for metallisation.

The die improvement imposes a further irreversible trend, and the power losses are reduced by each new IGBT and new Power MOSFET technology (device's die). Therefore, power density increases and die size shrinks over the development time. Recently the maximum junction temperature increases from 125°C from the past to 200°C. In the future are expected new material combinations and joining technologies, primarily when a wide bandgap (WBG) material like silicon carbide (SiC) or gallium nitride (GaN) is used.

This dissertation highlight that in the power semiconductor package technology, there is still room for improvements. The most commons stacks layer combination, from the die to case bottom, are already used for several decades. It shows considerable potential for improvement from the material science perspective. Indeed, it impacts the cost, the performance, the thermal dissipation and the system's reliability.

For the best reliability: the substrate, its functional surfaces, the die-attach, and other packaging materials must be perfectly matched together. Therefore, the fine-tuning of the material set is mandatory, including a wide range of qualification and intensive FMEAs. [19]

What disserted displays the automotive "Power Module" mechanical design complexity, which targets to:

- maximise the thermal exchange between each power electronics element (die form) and the package heat-sink;
- minimise the stray inductance;
- maximise space consumption;
- maximise the system's reliability;
- minimise the stress on key "Power Module" components (die, wire bonding and, etc.);

Those efforts are rewarded by the opportunity to obtain the best of performance and reliability from any power electronics element (Diode, IGBT or Power MOSFET). Significant limitations[27] occurring on traction inverters are associated with the DC-link capacitor (usually a PP film capacitor).

[27] In terms of mechanical design, volume consumption and converter's forecasted life-time.

3.6.4 Wide bandgap (WBC) Semiconductors advantages for Electric Vehicles

WBC Power Elements are more expensive than the standard "Si" technology, especially if compared to the IGBT alternatives. The notable advantages of WBC power elements, such as SiC Technology, are:

- lower losses (higher efficiency);
- higher switching frequency;
- higher junction temperature.

The selection of the correct power element technology for an automotive engineer is the outcome of a complex process driven by the end application technical requirements. It is possible to affirm that the SiC technology main advantage is the possibility of using a higher switching frequency, which enables the design of smaller E-Motors because it is rated to a much higher frequency. Therefore, the use of a higher switching frequency implies the size reduction of the DC-Link Capacitor[28].

The possibility of being more efficient and working at junction temperature above 200 degrees Celsius is a significant mechanical advantage on a system vehicle level. There are gains in terms of components packaging, not only because the traction inverter may be smaller and more convenient to install within the vehicle, but mainly for the more relaxed cooling requirements. Since a significant vehicle's design challenge is to ensure the appropriate cooling to each component, the combination of lower losses and higher acceptable coolant temperature produces a substantial cost reduction. Those facts explain why SiC technology is gaining popularity in every power converter for automotive application.

3.7 Battery Technology Overview

At the moment, automotive REESS technology represents the main limitation for the EV breakthrough and the full EVs market quota ramps up. Comparing to conventional combustion engine vehicles, EVs suffer a technological gap in three key factors, which are:

- REESS cost;
- range and charging time;
- lifetime.

[28] Many solutions based on PP film Capacitor and CERA-LINK (TDK) capacitors, demonstrate this opportunity.

Observing the technological evolution of Lithium-Ion technology, there is a shred of evidence that during the last 24 months, ample technological signs of progress have been achieved in the following areas:

a) cost per KWh;
b) charging cycles;
c) charging capability;
d) discharging capability.

At the moment, OEMs are trying to prioritise the investments in technologies, potentially, capable of performing super-fast charging of the vehicle's "REESS" and the relative infrastructures for performing this operation on a large scale. Automotive OEMs are targeting to transfer enough energy[29] in 15-20 minutes to allow the vehicle to drive for approximately 200 km.

3.7.1 PHEV, HEV and EV Battery Cell Chemistry Overview

At the moment, the most popular technologies used by automotive industries to implement REESS for HEV, PHEV and EV applications are:

a) NiMH;
b) Lithium-Ion (Li-Ion).

The most self-evident intrinsic difference between Li-Ion and NiMH batteries is the material used. Lithium-Ion batteries are made of carbon and highly reactive Lithium, which allows high-density energy storage. Nickel-metal Hydride batteries use Hydrogen to store energy, with Nickel and another metal (such as Titanium) to secure the Hydrogen-Ions (keeping a lid on the Hydrogen-Ions). Such different chemistry structures implicate substantial practical differences. A rough comparative parametric summary of both technologies may be the following.

- **Cost:** Nickel-metal Hydride batteries, at the moment[30], is the less expensive technology.

[29] It means to convoy up to 250KW from the charging station to the EV's REESS; this requires a cell's chemistry capable of allowing charging profile up to 5C.

[30] As production of Lithium-Ion cells is currently ramping up, it is expected that the economies of scale will play a role and price of Li-Ion cells will drop.

- **Weight:** NiMH batteries are larger and heavier than Li-Ion batteries (the density energy stored is critical in EV and PHEV applications, slightly less for HEV application, limiting the NiMH technology applications[31]).
- **Power:** current NiMH batteries can handle sudden power demands just as quickly as Lithium-Ion batteries, but the strength of Lithium-Ion cells is their ability to be charged and discharged more rapidly.
- **Energy:** Li-Ion technology has higher energy storage density and is less affected by the memory effect than NiMH technology (this affects the battery's capacity).
- **Durability:** both types of batteries are durable, and both have been in use for years in various applications; this is the one area where NiMH technology has an advantage[32].

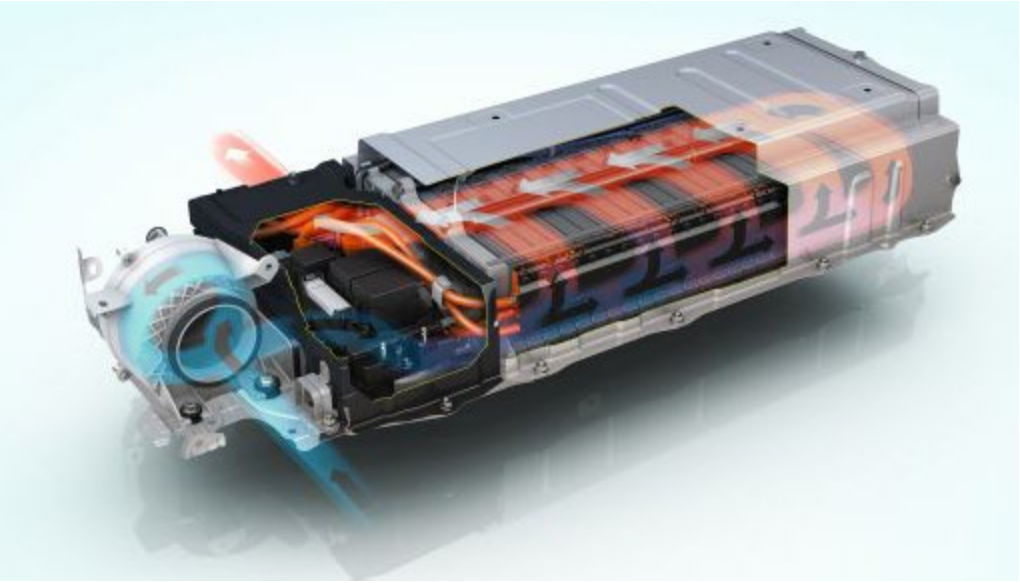


Figure 3.10: Toyota Prius HEV Nickel-Metal Hydride Battery (NiMH).

The scrutiny of these two technologies allows the Author to articulate a set of **conclusions.** The NiMH battery is currently an integral part of a large portion of HEVs on the market, while the new EVs and PHEVs use mostly Lithium-Ion batteries. The Li-Ion battery has the potential of eclipsing the NiMH battery, but it will require a few more years.

[31] Obviously, lighter battery packs with higher energy density makes it possible to extend the vehicle range.
[32] NiMH batteries are more predictable when it comes to performances.

The study case of the standard Toyota HEV NiMH REESS[33] appears very intriguing. It proved to be remarkably reliable in extreme environments during the last decades, especially during the vehicle cold start. Nowadays, Toyota is offering up to 14 years of warranty on their HEV NiMH battery pack, and it is common to obtain a prolonged warranty on a similar kind of REESS from other car manufactures. Li-Ion batteries do not last as long in extreme temperatures, particularly in hot climates. Presently, there are massive investments efforts to improve the Li-Ion batteries chemistry to improve performances (especially under thermal stress) and last as long as the vehicles they power.

3.7.2 Battery Management System (BMS)

The automotive EV market continues to be constrained by demand for longer range vehicles, improved functional safety, and decreased charge times and cost. Unlike a single energy storage element, such as a fuel tank, an EV's battery pack (or REESS) consists of hundreds or thousands of individual battery cells working together. BMS is interfaced with each cell, providing accurate cell measurement from the time the pack is manufactured to its retirement. Then, reading the remaining energy in a battery is more complicated than dispensing liquid fuel. While a fuel tank has a fixed dimension and carries fuel which amount can be estimated with excellent accuracy, an electrochemical storage system reduces its size and the "in and outflowing Coulombs" cannot be assessed with great accuracy as the battery ages.

Today's automotive battery market continues to be not merely cost-driven, but the demand for longer range vehicles, decreased charge times, and functional safety has paramount importance. Although, with up to 40% of the sticker price of an EV attributed to the REESS, performance and lifetime become the crucial factor in an EV's brand success. Truthfully, the EV battery design is a very intricate task, where shall be considered a range of priorities, including price, reliability and, safety. Market burdens are giving challenging battery management system requirements, demanding the adherence to the highest of standards with the narrowest of tolerances. Since a battery system is expected to deliver more than a hundred Amperes with a pedal's push (it is beneficial to operate at the highest voltage to be efficient).

[33] The batteries in Toyota's hybrid vehicles are efficient, corrosion-resistant units designed to last, which is why Toyota's standard battery warranty is five years or 100,000 miles and can be extended up to 15 years with no limit on total mileage. The batteries are actual units that have to store sufficient voltage to power the car with no assistance from the petrol engine.

However, Lithium-Ion battery cells can deliver only a few Volts and, to extract enough power, a large number of battery cells shall be connected together in series as one long stack.

The apparent result is that the BMS is responsible for monitoring the EV's REESS and managing **critical processes such as ensure battery safety, productivity, and longevity**. In fact, **the purposes of a BMS are**:

- to provide battery safety and longevity;
- to reveal the "State of Function" in the form of the "State of Charge" (SoC) and the "State of Health" (SoH, capacity);
- thermal monitoring and calibration;
- to indicate the "End of Life" when the capacity falls below the user-set target threshold;
- to provide cell balancing in multi-cell battery chains[34] (the most common automotive BMS architecture assigns this function to EBMs);
- to provide authentication and identification[35];
- to provide communications and diagnostics, BMSs incorporate some form of communication between the REESS, other vehicle's ECUs and the charger or the test equipment[36] (communication interfaces to allow the user to access to the battery for modifying the BMS control parameters or for diagnostics/test purpose).

In modular design, an EV's REESS is built with several battery modules connected in series and parallel accordingly. Electronics are attached directly to each cell in the stack, broadcasting back voltage and temperature, coordinated with output current. BMS requires a robust communication interface between the central unit (the BMS ECU) and the peripherals electronics units (electronic battery monitors, EBMs). It allows a modular design (architecture), fully extensible for a variety of different customer end applications.

BMS is continuously monitoring the cells, delivering reliable measurement accuracy over time, temperature, and operating environment; to do that, BMS relies entirely on the

---

[34] Small differences between cells due to production tolerances or operating conditions tend to be amplified during the charging/discharging cycles and, weaker cells became overstress during the charging causing them to became even weaker until they eventually fail (it will, most likely, cause premature failure of the whole battery).

[35] The BMS also allows the possibility to record information about the cell such as the manufacturers, type designation and the cell chemistry (which can facilitate automatic testing and the batch or serial numbers or date of a manufacturer which enable traceability in case of cell failure).

[36] A BMS might have a link to another system interfacing with the battery for monitoring its conditions history.

information broadcasted by EBMs[37], in order to estimate the REESS's SoC and SoH. Every cell's current and temperature must be controlled through a multi-layered algorithm at the central node (BMS processor)[38].

The BMS carefully monitors, controls, and distributes the reliable charge and discharge of the entire battery system during its lifetime. Precise monitoring of current and voltage profiles is critical, as overcharging a battery can cause an endothermic reaction (even an explosion), and undercharging (or a full discharge) magnifies the battery ageing. The quality and the reliability of the BMS directly impacts the miles per charge that an EV can deliver, maximizes the REESS lifetime, and, as a result, lowers the cost of ownership. Considering the investment for the EV's REESS, the value of BMS performance is clear, and it becomes even more evident as automotive designers consider warranty (and lifetime) costs. If one cell dies in a long stack of battery cells, eventually, the whole system may be lost. So, shall be adequately monitored and managed each cell, every day for the vehicle's life. Li-Ion cells cannot be operated to the full extent of their charge and discharge range. They must be kept in a particular range[39], as recommended by the manufactures (could be a range between 20% and 90% or slightly different depending on chemistry), or the cells are weakened. While SoC is helpful, the readout is incomplete without also tracking the capacity as the battery fades. Capacity is the primary indicator of battery SoH and should be part of the BMS functionalities. Knowing SoC and SoH, it is possible to estimate the "State of Function" (SoF).

By definition, the battery consists of:

- stored energy;
- the empty portion that can be recharged ;
- the inactive portion that is permanently lost due to ageing.

Rated capacity refers to the manufacturer's quantified capacity in Ah (Ampere-hours) that is only valid when the battery is new; available capacity indicates the actual energy storage capability derived by deducting the inactive part. "SoC" refers to the stored energy, which also includes the inactive part.

---

[37] The EBM has limitations; it cannot estimate the cell's capacity effectively. This can be mitigated by adding capacity estimations on the EBM's software.

[38] Key Values: Accuracy, Reliability, and Stability

[39] BMS devices address safety and reliability concerns by providing measurements that ensure each cell is functioning within a constrained operating range.

A BMS is programmed to rated capacity, and it measures the in and outflowing Coulombs that relate to the available capacity. As the capacity falls, the Coulomb count decreases, and this divergence enables capacity estimation. The most accurate readings are possible when computing the Coulombs from a fully discharged battery during a complete charge or discharging a fully charged battery to the cut-off point. Such clean starts are occasionally possible, and real-life capacity estimations get entangled over time.

A BMS sets flags when experiencing a full discharge and charge. During a rest period, an upper-level BMS might calculate SoC on hand of the stable open-circuit voltage and begin counting the Coulombs during charge and discharge from that vantage point. A few BMS also look at voltage recovery after removing a load to estimate SoC and (or) SoH.

Although the BMS is very effective in detecting anomalies, it is not that easy to elaborate even the most predictable health indicator. “Capacity fade” is challenging to estimate because voltage and internal resistance are commonly not affected. A typical BMS usually responds to anomalies that lie outside capacity estimation, such as voltage differences among cells caused by cell imbalances and a change in internal resistance.

A BMS might take the imprint of the “chemical battery” during charging and discharge. As well, the BMS establishes the “digital battery” that communicates with the user.

Nowadays, the leading automotive industry is targeting superior accuracy, stability and safety ratings. It is common to observe new EV, PHEV and HEV vehicles BMS targeting full compliant “ASIL D” systems.

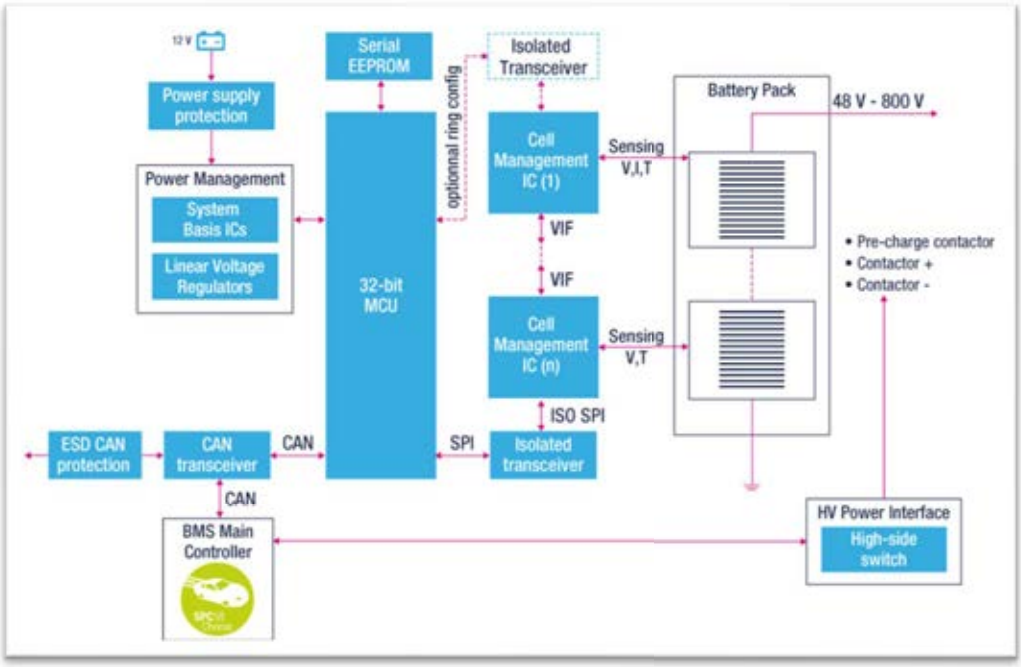


Figure 3.11: ST BMS reference design block diagram.

3.7.3 Application of Fuzzy Logic for BMS

Although there is interest in the topic and fuzzy logic may be a capable control strategy for the BMS ECUs, at this time, it is not a popular choice for automotive manufacturers.

However, **in the future, BMS might combine the information of the "digital battery" with that information of the "chemical battery" to provide reliable "SoF" data through advanced learning algorithms, which may allow a BMS to predict an eventual replacement.**

4 Theoretical Framework

A learning capable AUAV controller design is the core of the Thesis project; the Author's preferences lie in the "fuzzy logic" and "neural networks" arena. However, the system's software may lead itself to a whole array of possibilities regarding the choice of protocol and subsequent proprietary software, the software architecture of the system becomes the main focus of the proposal.

For many years, fuzzy logic has been a fascinating technology for designers of industrial, consumer and automotive products. However, achieving the right balance between cost and performance has not always been easy. Fuzzy algorithms can be executed on low-cost conventional microcontrollers, but as these have architectures that were not designed to handle fuzzy logic, the software overhead often makes the performance inadequate. Dedicated fuzzy processor chips can meet the most demanding performance needs. Today, only a few full custom (or semi-custom) integrated fuzzy controllers exist, and most of them are assembled from standard cells at the gate level. [20]

The dissertation starts with the scrutiny of a high-level design approach. The usage of high-level description methodologies for modelling fuzzy controllers reduce development time significantly, making a rapid design of custom fuzzy hardware possible. VHDL [21] for design capture and VHDL based logic synthesis are an efficient method for designing complex hardware.

However, for describing regular structures like finite state-machines, a different approach could be more appropriate. For describing such structures, could be used state-charts. Besides, a commercial tool based on state-charts incorporates a VHDL generation facility for generating synthesizable code. The employment of state-charts formalism for capturing a fuzzy control system's rule base is a widely used approach in the scientific literature.

Fuzzy controller relies on conventional principles for the interface and the information exchange. In the controller, an external device's information (such as a sensor) is converted into an output control signal to drive a device[40] (or multiple devices) via the process of fuzzification, rule evaluation and defuzzification. These processes are all based on a set of

[40] Such as motors, actuators etc.

membership functions and the FIS; it is available a vast scientific literature, such as [20, 21, 22, 23, 24 and 25], to support those design processes.

The VHDL assembly of the fuzzy control system and the synthesis to a gate-level description for Field Programmable Gate Array (FPGA) technology is usually performed using dedicated tools provided by the FPGA manufacturer (often are available dedicated open-source compilers).

Although the motivation behind the implementation of a fuzzy controller in VHDL is driven by the need for an inexpensive hardware implementation of a generic fuzzy controller for use in industrial and commercial applications. There are several advantages more for this approach. Field Programmable Gate Array (FPGA) is used as a hardware platform because FPGA allows very high logic capacity (the amount of digital logic that can be mapped into a single FPD) [26 and 27]. FPGAs offer more flexibility than ASICs because the chip can be reprogrammed[41], allowing to redesign portions of the system's circuits for optimisation [28]. With the use of cost-effective FPGA for the implementation of the fuzzy logic controller, it is possible to fully benefit from the parallel computational capabilities of the fuzzy logic (and neural networks). [29]

A significant advantage of using an FPGA, after the parallel computational process, is the possibility of having interchangeable blocks-based software where the objects associated with each "Rulebase" represent an independent "VHDL component". For each "VHDL component", if a standard porting layout is used, it could be possible to tune the controller within the main state machine algorithms performing only two operations:

- adapt the algorithm's objects (adapt or replace at the "Component Level" the rules, the membership functions, etc.);
- define the new weights of each "Rulebase" (algorithms can be trained[42] to achieve a perfectly tuned system). [29]

This solution enables a dynamic[43] FIS tuning via the cloud or a dedicated learning process typical of the neural network design. Enabling this functionality, it is possible then to influence the behaviour of a specific "device", adapting the "controller's behaviour" to particular tasks or environmental conditions.

[41] FPGA based systems can be reprogrammed several times.
[42] It might be used as a pre-defined algorithms setup.
[43] It might be seen as a partially DES function.

4.1 Study Case Introduction

Neural networks, given their learning ability and adaptability, are applied in areas such as robotics (Bekey and Goldberg, 1993; Rao, 1995; Zouetal.,2006), image processing (Carpenter and Grossberg, 1992; Egmont-Petersenetal., 2002; Hongetal., 2009), and speech recognition (Othman and Riadh, 2008; Lippman, 1988). Within the hybrid systems, the neuro-fuzzy systems combine both paradigms; on the one hand, the system of linguistic rules generated by an expert, on the other hand, the learning ability of neural networks applied to this system. The applications include pattern recognition (Ray and Ghoshal, 1997; Pal and Mitra,1999), robotics (Rusuetal., 2003; Wongsuwarn and Laowattana, 2006), non-linear system identification (Babuska and Verbruggen, 2003; Panchariyaetal., 2004), adaptive signal processing (Li and Tsai, 2006; Chabaaetal., 2009) and, etc. [30]

The study case focuses on the learning process of neuro-fuzzy networks for the control of a small electric UAV. The project's idea is to move part of the hardware/mechanical design load, making it as simple as possible, to the controller's design. The main goal is to study a technique in order to make it possible to build a controller able to learn and tune itself in order to control its own flight properly. By assumption, the design employs, as a baseline, a fly-wing platform for the small UAV mechanical design due to the packaging constraints for batteries and all electronics. For packaging constraints, it is meant a not optimised design in terms of size/volume, the outcome of a low-cost hardware/electronics solution, which is divided into three groups: the central control unit, the actuators' (E-Motors and SERVO-Motors) drivers and the battery monitoring unit.

As the main project's ambition is to implement a parallel computation unit, the "Control Unit" is assumed to be built around an automotive-qualified FPGA, while the motor drivers used are low-cost motor driver available on the market. The algorithm's development strategy moves from the definition of a few essential fuzzy logic functions used as fundamentals for the learning process.

For essential fuzzy logic functions, it is meant a set of fundamental "Membership Input Functions" (MIF), a set of fundamental "Membership Outputs Functions" (MOF) and a detailed set of "Rulebases" (FIS). Such a primary fuzzy logic controller may perform a limited flying operation, but to achieve a proper vehicle control, it is indispensable to perfectly tailor a set of weights for each function of each "Rulebase". Calling back what described in "Chapter 2", in order to define such weights, it is required the full knowledge of the "UAV Dynamics". The identification of the weight's values is associable with the

identification of the unique physical parameters of the vehicle; required information (as described in "Chapter 2") to solve the "Vehicle's Dynamic Equations". This proposed work aims to overcome the design's load of defining the "UAV Flight Dynamics model"; **by assumption, the controller's design is not based on the solution of the "UAV Dynamics Equations".**

In front of the absence of this essential information, the controller's key point is the learning process, which is the outcome of two critical processes: data capture from a human-controlled flying operation and by the consequent learning/training process, which is a software-based data computation of the information previously captured.

4.2 Controller's Framework Definition

As previously introduced, the "Control Unit" receives external devices' information and generates the output control signals to drive a device (or multiple devices) via the process of fuzzification, rule evaluation and defuzzification. Such kind of processes are based on a set of membership functions and FIS; numerous publications, such as [20, 21, 22, 23, 24 and 25], illustrate the processes' details. The first step of the "Control Unit" design is the definition of the system's peripherals: the sensors, the actuators and the UAV's powertrain.

4.2.1 Controller's Inputs Definition

By definition, sensors are peripherals capable of detecting pre-defined information (generally defined as the "controller's inputs"). Each input is associated with a function and, if required, a functional safety rating; the application case associates a specific MIF for each sensor data.

As previously declared, the systems controller's goal is to replicate the behaviour[44] of a "Human Pilot"; to achieve that, the controller requires the information provided by a set of heterogeneous sensors. It is reasonable that a "Pilot" might control the vehicle giving more attention to particular parameters (or sensors) and less attention to others. In the proposed "controller's framework", this "behaviour" is implemented using the primary fuzzy logic controller's optimisation process first and then a learning/training process.

It is assumed that the study case system requires the following mandatory parameters:

a) altitude;

[44] For a Pilot's behaviour, it is intended the Pilot's driving style.

b) speed;
c) pitch angle;
d) rolling angle;
e) yaw angle;
f) estimated position;
g) REESS SoC[45].

As previously introduced, the strategy to be pursued is to utilise a simplified mechanical design of a small UAV and to focus on the "neuro-fuzzy learning process". The assumptions made on the selection of the controller's mandatory inputs are coherent with a mechanical design based on a flying-wing concept[46].

4.2.2 Controller's Outputs Definition

By definition, an "actuator" is a component of a machine that is moving and controlling a specific mechanism. An "electro-mechanical actuator" utilises a relatively low power electrical signal to control the mechanical load. In the proposed study case, each "electro-mechanical actuator control signal" represents a controller's output.

In regards to the study case, the controller's outputs are the following:

- ailerons SERVO-Motors (two units, complementary control);
- Elevator SERVO-Motor;
- Rudder SERVO-Motor;
- Propulsion E-Motors (two independent units).

Indeed, it is assumed that the "controller" has six outputs, although it is more appropriate to declare that there are five independent outputs since that the two ailerons are controlled by a single control signal (the ailerons control signal splits into two complementary signals). This signal conditioning is a digital operation, and it is implemented within the FPGA by a digital signal processing sub-system.

[45] Mostly for safety purposes.
[46] This mechanical concept overcomes the REESS and hardware/electronics packaging issues typical of small UAV based on small RC planes architecture.

4.2.3 *Rule Block and Defuzzification*

FIS (the fuzzy logic "Rule Block"[47]) is the controller's core, which in several specialised fuzzy logic GUI environments is addressed as a single "Rulebase" or a set of multiple independent "Rulebases". Each rule has a unique weight, which defines the importance of function to function link for the system's decision-making process. At the beginning of the "Rulebase" design process, unregulated weights are assigned, according to assumptions made on the available data. Later the weights can be tuned by a "learning/training process" as a result of field tests.

4.2.4 *VHDL implementation theory*

It can briefly be mentioned that one of the main reasons that influenced the success of fuzzy systems, neural networks and neuro-fuzzy systems is their ability to approximate continuous non-linear functions. In this area within the fuzzy systems, the following works can be cited: Wang (1992), Kosko (1994), Zeng and Singh (1996), Rovatti (1998), Kreinovichetal (2000), Caoetal (2001), and Landajo et al. (2001). Concerning neural networks, the following contributions should be highlighted: Stinchcombe and White (1989), Cotter (1990), Hornik (1991), Attali and Pagès (1997) and, Castro et al. (2000). On neuro-fuzzy networks, the following references are emphasised: Buckley (1993), Castro (1995), Jang et al. (1997), Nauck and Kruse (1999), Wang and Wei (2000), and Wu et al. (2010). [30]

The different applications of soft computing algorithms have been realised on different supports overtime, depending on the technologies of the moment. The following have been used: general-purpose processors, dedicated processors, dedicated coprocessors, specific designs with Very Large Scale Integration (VLSI) integration scales, either analogue or digital or mixed, up to the current reconfigurable hardware (HW) devices. Use has also been made up of multi-processor platforms for systems that require high-speed computation and supporting parallel processing, as in the case of neural networks. The use of one or other support has been conditioned by different requirements, among others, power consumption, processing speed, size, portability and cost. [30]

[47] The "Rule Block" links accordingly MIFs to the MOFs. It may be seen as a simplified definition of the block.

Digital hardware achieved the most crucial development due to the consolidation of programmable or reconfigurable devices, mainly in the FPGAs. The high integration density and the power introduced by the parallel structures achieved by this technology have enabled implementations of fuzzy inference systems with a high number of fuzzy rules, neural networks with a large number of layers and neurons, including learning algorithms, and finally, neuro-fuzzy systems based on fuzzy rules and endowed with learning mechanisms of the same type as those used in neural networks. [30]

In this proposed structure, it is considered that a programmable integrated circuit is a hardware that a user can reconfigure by means of any specific technique. These are commercial devices such as FPGAs, where the applications, as in the dedicated ones, are entirely HW based. One aspect that makes these devices particularly attractive is their ability to re-programme and the existence of EDA tools that facilitate their design. A recent type of programmable device called FPAA (Field Programmable Analog Array) can also be mentioned, which emerged as a commercial option in the 2000s, but in a very low integration density and with few applications made in the area at hand. [30]

Digital implementations have superior immunity against factors such as noise, temperature or voltage variation, among others. In contrast, the processing speed tends to be lower than in ASIC analogue devices, although evolution in integration technologies has changed this scenario.

In this classification, FPGAs are emphasised because they can be programmed through circuit design using graphic or, preferably, HW description languages like VHDL (Very High-Speed Hardware Design Language) or Verilog. Further development of FIS privileges the FPGAs because of their ability to reconfigure and the low “time to market”. For example, consulting the “Web of Knowledge - Web of Science”, there are approximately 340 works on FPGAs only between 1990 and 2012. [30]

A few literature examples of fuzzy and neuro-fuzzy systems build on digital programmable integrated circuits are:

- Manzoul and Jayabharathi (1992) presented the work “Fuzzy controller on FPGA chip” (a fuzzy controller expressed as Boolean equations on an FPGA).
- Hung and Zajak (1995) presented the implementation of a fuzzy inference system on an FPGA in the article “Design and implementation of a hardware fuzzy inference system”.

- Hollstein et al. (1996), in the article "Computer-aided design of fuzzy systems based on generic VHDL specifications", presented a development tool for performing parallel processing architectures[48] or sequential rules.
- Blake et al. (1998), in the article "The implementation of fuzzy systems, neural networks and fuzzy neural networks using FPGAs", presented three approaches to "Soft Computing", but the article will focus on the FIS[49].
- D'Amore et al. (2001), in the article "A two-input, one-output bit-scalable architecture for fuzzy processors", presented an automatic synthesis of a fuzzy system with scalability, with either the bits of the I/O variables or the bits of the membership functions of the I/O (the synthesis process is performed using the VHDL code).
- Raychev et al. (2005) presented the work "VHDL modelling of a fuzzy co-processor architecture" (presented a hardware accelerator for fuzzy calculations).
- Hung (2007), in the article "Using FPGA technique for design and implementation of a fuzzy inference system", implemented a fuzzy inference system with the max-min compositional rule with the COG being the defuzzification method applied.
- Lizárraga et al. (2008), in the article "Modeling and simulation of the defuzzification stage using Xilinx system generator and Simulink", illustrated a defuzzification stage, using the "Height method" (Driankovetal.,1996).
- Fung et al. (2009), in the article "FPGA-based adaptive fuzzy back-stepping control for a micro-positioning Scott–Russell mechanism", presented a fuzzy controller with an error feedback mechanism applied to a micro-positioning Scott–Russell type (The actuator was piezoelectric).
- Hsu et al. (2010), in the article "Chip-implementation of a self-tuning non-linear function control for DC-DC converters", proposed a model-free STNFC design method suitable for real-time practical applications[50].

---

[48] Each FIS consists of three distinct modules: fuzzification, rules of inference and composition/defuzzification. Modules can be described in C or VHDL. The defuzzification follows the MOA (Midpoint of Area) method.

[49] The interest of the article is, taking a non-linear function of three variables, to compare the approximation capacity between the architecture e on an FPGA and the architecture on Matlab.

[50] The system is applied to a DC-DC converter based on an FPGA controlling the duty-ratio of PWM modulator in the DC-DC converter. The article highlights the following points: STNFC is a system without heavy computational loading, the parameter-learning algorithm is designed based on the Lyapunov stability theorem to guarantee the system stability, there are successful

- Kung et al. (2011) in "Simulink/Modelsim co-simulation and FPGA realisation of speed control IC for PMSM drive", implemented a fuzzy-control based speed control IC for a Permanent Magnetic Synchronous Motor (PMSM).
- Abramson et al. (1998), in the paper "FPGA based implementation of a Hopfield neural network for solving constraint satisfaction problems"[51], described and solved the N-Queen problem using a Hopfield neural network (used to solve complex optimisation problems) to demonstrate and solve the potential of a custom computer-based on FPGA technology.
- Omondi and Rajapakse (2002) published the work, "Neural networks in FPGAs" in which an approach is made to parallelism and arithmetic, the HW or SW implementation, and which finally examines a case of Independent Component Analysis (ICA) (Comon, 1994), implementing an independent component neural network (ICNN) over the Xilinx XCV812C.
- Kim et al. (2003) presented the article "FPGA implementation of ICA algorithm for blind signal separation and adaptive noise cancelling" (applied to speech recognition in noise environments and echo[52]).
- Ide and Saito (2006) presented the article "FPGA implementations of neocognitrons" applied to character recognition and biometric measures.[53]
- Bastos et al. (2006) presented the work "FPGA implementation of neural network-based controllers for power electronics applications", where an ANN[54] governs a buck converter (step-down DC/DC) based on the behaviour of a SACT controller (synergetic approach to control theory).
- Hu et al. (2008), in the article "Key issues of FPGA implementation of neural networks", give an overview of the different parts and methods involved in the

applications of the STNFC system to control the forward DC-DC converter, and finally, the proposed STNFC methodology can be easily extended to other DC-DC converters.

[51] The paper highlights that in this architecture, first, the weights are small land can be represented using small integers. They also reduce the carry propagation delays. It means that the hardware responsible for the accumulation can be optimised for small integer values. Second, the vector product becomes a set of conditional additions without the need to perform any multiplication operations. Third, the interconnection between neurons is fixed and dictated by the nature of the constraints in the problem.

[52] Algorithms of signal separation (blind signal separation) and algorithms of adaptive noise cancellation (adaptive noise cancelling) were implemented.

[53] It describes the implementation of a reconfigurable ANN on a parallel computer architecture based on FPGAs, called REOMP (Reconfigurable Orthogonal Multi-Processor Memory). On this architecture, Neocognitrons are implemented. A Neocognitron ANN model is a feed-forward topology proposed by Fukushima (1982), based on the model of Hubel and Wiesel (1968) concerning the research of the vision from a biological point of view.

[54] The ANN chosen had the structure 4–4–1 and was trained to have the high-performance characteristics of the SACT controller.

design of the ANNs such as data representation, inner-products computation, implementation of activation function, storage and update of weights, nature of learning algorithm and design constraints.

- Shoushan et al. (2010), in the article "A single layer architecture to FPGA implementation of BP artificial neural network", presented a back-propagation ANN design and constructed an application for classifying the defects of the carbon-fibre reinforced plastic.[55]
- Mekki et al. (2010), in the article "FPGA-based implementation of a real-time photovoltaic module simulator", proposed a multilayer perceptron (MLP) for simulation and implementation of a real-time PV-module[56] on FPGA.
- Hasanien (2011), in the article "FPGA implementation of adaptive ANN controller for speed regulation of permanent magnet stepper motor drives", studied the dynamic response of a PMSM under full load torque and underload disturbance.[57]
- Cárdenas et al. (2012), in the article "Development of an FPGA based real-time power analysis and control for distributed generation interface", presented the development and the experimental evaluation of a power control system for a single-phase grid-connected which has several energy sources connected.[58]
- Soleimani et al. (2012), in the article "Biologically inspired spiking neurons: piece-wise linear models and digital implementation", proposed PWL models with a fewer number of multipliers for implementations of spiking neural networks on FPGAs.[59]

---

[55] A one-dimensional systolic array of the finite impulse response (FIR) filter for the back-propagation algorithm is introduced. All calculated parameters are stored on a RAM, and the implementation of excitation function (sigmoid) is performed on Look-up tables. The design is implemented on two FPGAs, and a comparison between the resources used is performed.

[56] The evaluation of the performance of a PV-module is based only on meteorological data such as air temperature and total solar radiation and can be used for prediction of the PV electrical energy output under actual climatic conditions.

[57] The evaluation of the performance of a PV-module is based only on meteorological data. The model has been developed and Simulated under Matlab/Simulink and the optimal configuration has been written in VHDL on ModelSim and then implemented on an FPGA.

[58] An ADALINE network is used for control and synchronisation of the power of the electric network. The learning is performed by means of the Widrow-Hoff algorithm. The article performs a comparison between the Adaline network and the FFT implemented off-line on Matlab. The results are similar.

[59] The models replaced the operation "square" by comparison or "absolute value"; this means that in digital implementations the multipliers are replaced by comparators which implies that they can implement a large number of neurons. The network is trained with a supervised and unsupervised learning algorithm. The results show that: the 91.7% accuracy in the recognition and the implemented PWL models are significantly faster than the Izhikevich (2003) model with a simple combinational multiplier.

- Saadi and Bettayeb (2013) in the article "ABC optimised neural network model for image deblurring with its FPGA implementation", try to improve radiological images degraded during acquisition and processing.[60]

[30]

Concerning the design flow of a fuzzy system, two different levels may be considered. **The algorithmic level** specifies the functional behaviour of the system. The objective within this level is to define the shape of the membership functions, the implication mechanism, the "Rulebase", and the defuzzification strategy that better achieve the proposed system task. At **the circuit level**, the designer has to select an efficient system architecture, design the required building blocks, and verify the temporal behaviour of the system. Therefore, a design methodology for a fuzzy system has to cover the different design stages, from the system's specification up to the system's prototyping and testing. To accomplish this task, some authors proposed the use of VHDL as a language to describe and model the high-level system [31, 32, 33 and 34] and the employment of specific architectures of fuzzy processor [32, 35 and 36].

4.2.5 VHDL Modelling theory

VHDL language imposes some limitations if confronted with the versatility and expressiveness of other fuzzy logic oriented languages (such as XFL3 [37]). On the other hand, it is crucial to adapt the system's characteristics (types of membership functions, inference algorithms, defuzzification mechanisms) to its hardware implementation. [32]

The premise, VHDL will be used as the working platform for the system implies that the fuzzy system description must be synthesisable (hardware realisations on FPGA). A synthesisable VHDL algorithm requires to adapt and tune the characteristics of the controller (types of membership functions, inference algorithms, defuzzification mechanisms, etc.) to the physical hardware implementation (FPGA printing). [32]

The [32] and the [38] describe the advantages of the high-level descriptions of neuro-fuzzy systems in an easy way[61]. To achieve behavioural modelling might be used a VHDL description style where the system's definition structure (fuzzy sets, rule base, etc.) and the operator's description (connectives, fuzzy operations) are defined separately, making it

[60] An autoregressive moving average (ARMA) model is identified using an ANN. The network training is improved using a novel swarm optimisation algorithm called Artificial Bees Colony (ABC), inspired by the foraging intelligence of honey bees.

[61] Linguistic variables, rule base, fuzzy operators.

possible to describe both the fuzzy system structure and the processing algorithm independently. The description format makes it possible to use linguistic hedges in order to compact the rules defining the system's behaviour. High-level descriptions' main advantage is the availability of tools capable of translating a fuzzy logic oriented language with a GUI interface into a VHDL code[62]. [32 and 38]

Proposed work utilises the "XFUZZY XFL version 3.5" (or XFL3) GUI developed by "Instituto de Microelectrónica de Sevilla (IMSE-CNM)" [38] to describe the fuzzy logic controller (or the neuro-fuzzy logic controller after the learning/training process) and then translate this description to a valid VHDL code.

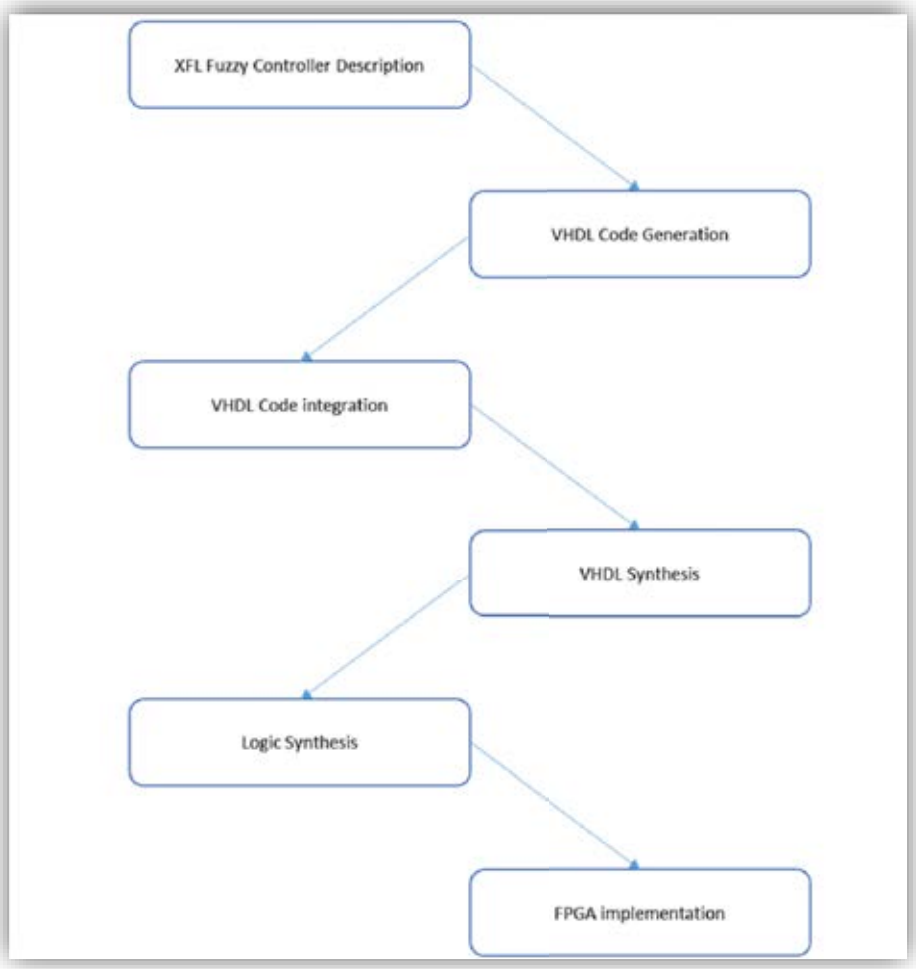


Figure 4.1: proposed design flow.

[62] VHDL code could be automatically integrated on specific supported FPGAs or shall be manually integrated by the designer into the "VHDL Top Level Architecture" that the user uses to describe all other systems' blocks and interfaces.

"Xfuzzy" is a development environment that eases the specification, verification and synthesis of fuzzy inference systems. XFL3 specification language is the core of the tools integrated into such described environment [37]. A set of standard functions encased in a module called the "XFL library" performs the parsing and semantic analysis of XFL specifications and stores them using an abstract syntax tree. This format is used inside the environment when handling system descriptions. "Figure 4.1" illustrates the design flow for the hardware implementation of fuzzy systems.

The starting point of the process is a behavioural description of the system using the specification language XFL. The verification process is carried out with the help of the simulation and learning tools provided by the environment.

Once the system specification is validated, the subsequent step is the generation of the VHDL code from the XFL description. The synthesis tool, called "xfvhdl", translates the XFL specification into a VHDL description according to the realisation strategy (behavioural or structural). In the case of behavioural strategy, "xfvhdl" gives a system description according to the description style[63]. It includes a package containing the type and function definitions. In specific architecture, "xfvhdl" employs a cell library containing the parameterised VHDL description for the basic building blocks. There are two kinds of blocks: data path building blocks (implementing the inference algorithm) and control blocks (controlling the memory write/read operations and the signals which control the operation scheduling). The code used in the cell library description is compatible with the restricted VHDL implementations of most synthesis tools. It is essential to highlight that the output VHDL code is "raw"[64], and it is necessary to adapt the code to the end FPGA and the system's "state-machine".

[32] validates the VHDL description using a simulation process (simulator Model-Sim of Mentor Graphics), then performs the logic synthesis stage[65] and then creates[66] the circuit description of the fuzzy system. [32]

---

[63] The behavioural modelling uses a VHDL description style where the system structure description and the operator description are defined separately, making it possible to describe both the fuzzy system structure and the processing algorithm independently.

[64] For "Raw" code it is intended that the generated VHDL describes only the "Architectures" which contains the "neuro-fuzzy block" without any configuration of the FPGA, this because the XFUZZY XFL3.5 GUI is able to configure only two specific FPGA; proposed work focuses on a not supported brand and P/N.

[65] Such as XST of Xilinx or FPGA Express of Synopsys. [32 and 38]

[66] In order to accelerate the development of those two design stages, "xfvhdl" provides two additional outputs: a "testbench file" to ease the simulation of the fuzzy system and a command script file to drive the synthesis and Xilinx FPGA implementation. The following architectural options are defined by the user when "xfvhdl" is run: architectural options (memory-based MFCs or arithmetic MFCs), knowledge-base (predefined using ROM or programmable using RAM), and memory implementation (using

The study case, due to the complexity and by the necessities to generate a general-purpose algorithm compatible with a wide range of potential FPGAs (rather than focusing on the specific P/Ns supported by the XFL GUI), requires few more intermediate steps. It is indispensable a manual integration of the generated VHDL algorithm within the whole system (the main VHDL algorithm, which includes: sensors interface, actuators interface, powertrain interface, system's clock, etc.). This operation shall be performed on the adopted VHDL environment; it cannot be performed on XFUZZY GUI.

The tuning stage is usually a very intricated task when designing fuzzy systems. The system behaviour depends on the logic structure of its "Rulebases" and the membership functions of its linguistic variables. The tuning process often exploits the adjustments of the different membership function parameters that appear in the system definition. Since the number of parameters to modify (simultaneously) is high, manual tuning is cumbersome, and automatic techniques are required. The two learning mechanisms most widely used are supervised and reinforcement learning. In **supervised learning techniques,** the desired system behaviour is given by a set of training (and test) input/output data, while **in reinforcement learning,** what is known is not the exact output data but the effect that the system has to produce on its environment, thus making necessary the monitoring of its on-line behaviour. [38]

[38] shows that the Xfuzzy 3 environment includes four tools for this design stage: "xfdm" and "xftsp" are knowledge acquisition tools. The first one allows obtaining the structure of inference systems used as fuzzy approximators or classifiers. In contrast, the second one primarily focuses on time series prediction applications. "xfsl" is a parameter adjustment tool based on the use of supervised learning algorithms. In supervised learning techniques, the system's desired behaviour is described by a set of training (and test) patterns. **Supervised learning attempts to minimise an error function that evaluates the**

---

distributed or block-type RAM, distributed ROM or combinational logic). The storage strategy for the knowledge base conditions the VHDL description style of the system modules. The choice between the two alternatives (ROM or combinational logic) depends on the synthesis tool. Synopsys tools (as an example) will implement the MFC using combinational logic. Nevertheless, the Xilinx synthesis tool (XST) can detect the ROM structure, and the implementation tool can configure properly the XC4000 and Spartan2E basic building blocks (CLBs and Slices, respectively). Synopsys tools cannot identify a memory from that VHDL code, but the Xilinx tool can make two kinds of implementations: using distributed memory, or using block memory (Spartan2, Spartan3 and Virtex families).

The selection of the synthesis tool and the implementation options are performed as "xfvhdl" command parameters. Option-C allows the choice between Xilinx-XST (x), Synopsys-FPGA Express (e) or Synopsys-FPGA Compiler. Parameter-M allows a selection between the RAM memory implementation using distributed RAM (d), block RAM (b), ROM (o) or combinational block. The "xfvhdl" command admits additional parameters to determine the FPGA device, the synthesis effort level, and the synthesis optimisation objective (area or speed). [32 and 38]

difference between the actual system behaviour and its desired behaviour defined by the set of input/output patterns. Finally, "xfsp" is a simplification tool that allows reducing the number of membership functions and compacting the rules bases of a fuzzy system to facilitate its software or hardware implementation and to increase its linguistic interpretability. [38]

The tool "xfdm" facilitates the identification of fuzzy systems from numerical data using different algorithms based on matrix partitioning (Grid Partitioning) or data grouping (Cluster Partitioning) techniques. "xfdm" can be executed from the command line or through its graphical user interface using the "Data Mining" option of the "Tuning" or the corresponding icon in the main window of the environment. [38]

The main window of "xfdm" is divided into two parts. The upper part configures the identification process, defining: the selection of the algorithm, input/output data file, number of inputs and outputs, inputs style and fuzzy system style. [38]

The system's learning and training activities represent the last design process before generating the final VHDL code[67].

[67] Which shall be integrated into the main VHDL algorithm before to be synthesised and then printed on the target FPGA.

5 System's Hardware Design Proposal

As previously introduced, the assumption is to use a simplified RC plane mechanical design as a worst-case scenario for the controller design, where a 3D printed homebuilt aircraft may be turned into AUAV through the process and the algorithms disserted. The design uses a twin-motor fly-wing platform as a mechanical baseline for the controller's design dissertation. For the proposed work, the electronic hardware configuration proposal endeavours to perform a few functions, which are:

- to define the core electronics/hardware required by the "Autonomous Flight Mode Controller";
- to define the hardware for the neuro-fuzzy controller;
- to define the hardware for the learning/training process.

5.1 Core Hardware Definition

Budget is the main project's limitation, and the electronic design and prototyping converge on cost optimisation. This concept is reinforced by the study case assumptions made previously; as mentioned before, most of the design load moves to the controller's design. In order to reduce the cost has been privileged the usage of free sample parts and functional development boards.

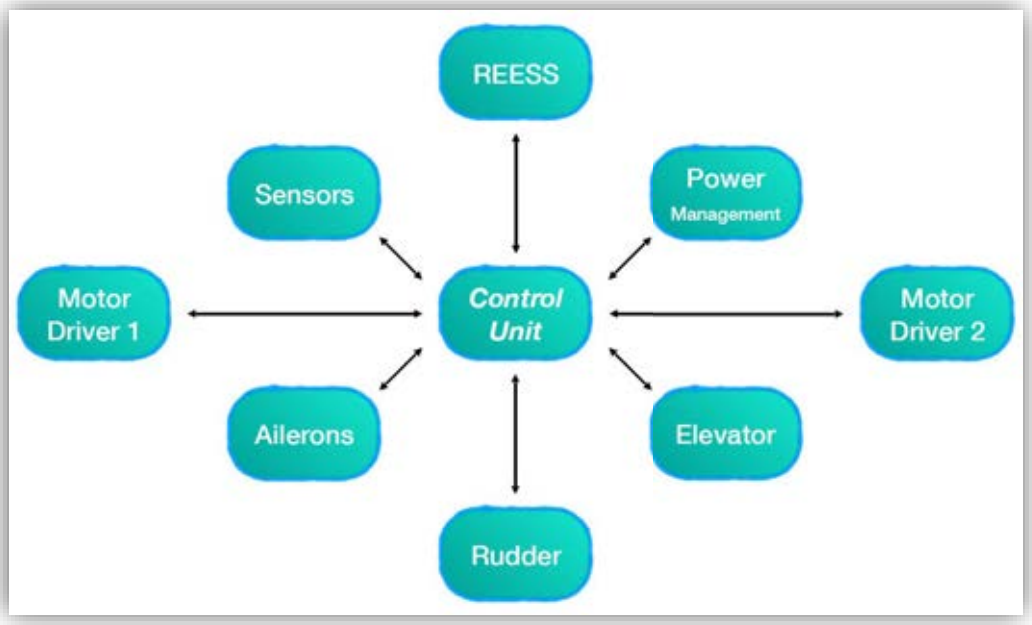


Figure 5.1: HW high-level block diagram.

Figure 5.1 describes the block diagram of the employed electronic hardware; the system's core unit is the FPGA, which acts before as the system's gateway (collects and digital processes all peripherals information) and then acts as the system's controller performing a parallel computation of the collected information.

5.1.1 Control Unit, FPGA

FPGA is the core component of the AUAV control unit, which collect the information from all the sensors and generates commands for the actuators and the motor drivers. By assumption, the proposal's control strategy targets to implement a neuro-fuzzy network, which, by definition, is a parallel controller. The decision of using an FPGA comes from the fact that an FPGA has a parallel computation capability while any standard MCU sequentially executes[68] operations. For FPGA's parallel computation capability, it is meant that the FPGA allows designing the controller in blocks that will work in parallel and in a symbiosis between each other. The practical result will be that each MIF will be processed simultaneously and independently, as well as for each MOF[69].

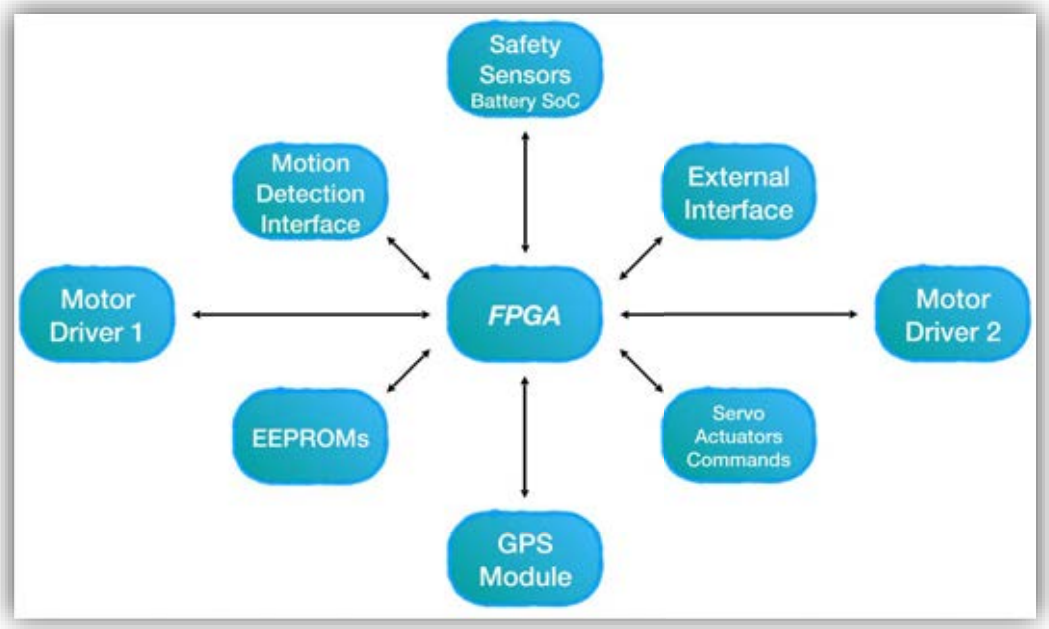


Figure 5.2: FPGA's peripherals block diagram.

[68] It computes the data in a serial loop.
[69] Both MIF and MOF use similar integration structures within the controller.

For this specific project, the Author advocates the use of a “Lattice Semiconductor” automotive-qualified FPGA; few factors drive the decision:

a) low unit cost;
b) high reliability;
c) production longevity;
d) free compiler;
e) low-cost debugger/programmer device.

5.1.2 Digital Motion Sensor

For the implementation of autonomous applications has paramount importance, the estimation of the vehicle’s orientation. To implement such peripheral, it is available for the project a “LIS3DSH”. It is a very low-power device with a high-performance three-axis linear accelerometer belonging to the “nano” family with an embedded state machine that can be programmed to implement autonomous applications.

The “LIS3DSH” has dynamically selectable full-scales of ±2g/±4g/±6g/±8g/±16g and is capable of measuring accelerations with output data rates from 3.125 Hz to 1.6 kHz. The unit configuration happens via six control registers, a FIFO control register and several registers for the calibration (device’s fine-tuning). The system’s outputs are accessible on seven registers: one temperature register and two registers for each axis output.

“Control Register 1”, State Machine 1 (SM1) control register, defines the SM1 Interrupt Enable / port selection.

0	0	0	0	0	0	0	0

Table 5.1: proposed configuration for the LIS3DSH “Control Register 1”.

“Control Register 2”, State Machine 2 (SM2) control register, defines the SM2 Interrupt Enable / port selection.

0	0	0	0	0	0	0	0

Table 5.2: proposed configuration for the LIS3DSH “Control Register 2”.

“Control Register 3” defines the “Interrupt” configuration (the assumption is to do not use Interrupts on the state machine).

0	0	0	0	0	0	0	0

Table 5.3: proposed configuration for the LIS3DSH “Control Register 3”.

"Control Register 4" enables the sensor and defines its working frequency (in this case set to 400Hz) and empowers the "Block Data Update" (BDU) to avoid the reading of values (most significant and least significant parts of the acceleration data) related to different samples. Expressly, when the BDU is activated, the data registers related to each channel hold the device's most recent acceleration data produced. However, if the reading of a given pair[70] starts, the refresh for that pair results blocked until both MSB and LSB data sections will be read.

0	1	1	1	1	1	1	1

Table 5.4: proposed configuration for the LIS3DSH "Control Register 4".

"Control Register 5" controls four functions: the unit Self-Test, the output scale (by default, it is 2g, for the application it is more appropriate to select 4g), the anti-aliasing bandwidth filters (proposed configuration sets to 800Hz) and the SPI interface mode selection.

0	0	0	0	1	0	0	0

Table 5.5: proposed configuration for the LIS3DSH "Control Register 5".

"Control Register 6" is an advanced functionality register that enables BOOT, FIFO, "Stop on watermark", and the functionality of register address automatically increased during multiple byte access with a serial interface. For the application, the proposed configuration accepts the default configuration.

0	0	0	0	0	0	0	0

Table 5.6: proposed configuration for the LIS3DSH "Control Register 6".

In conclusion:

- FIFO Control Registers are set to default because the FIFO advance operations are not required;
- device tuning and system calibration are achieved on all 3-axis via a dedicated "Offset Register "and by a constant shift register;
- all other registers are kept as default;

[70] i.e. OUT_X_H and OUT_X_L, or OUT_Y_H and OUT_Y_L, or OUT_Z_H and OUT_Z_L.

- all “Output registers” contains data in TWO’s complement format (signed integer); this means that the data should be converted before getting meaningful information.

5.1.3 Gyroscope

The correct estimation of the vehicle’s motion angles requires, in addition to the 3-axis accelerometer, the use of a gyroscope. The “A3G4250D” is a low power 3-axis angular rate sensor able to provide high stability at zero rate level and sensitivity over temperature and time. It includes a sensing element and an IC interface capable of providing the measured angular rate through a standard SPI protocol. The “A3G4250D” is configured via 15 Registers, has three read-only operational/status registers and has seven output registers, one for the temperature and six for the axis acceleration computed in “Degree Per Second” (DPS).

“Control Register 1” defines: the “Output Data Rate”, the “Bandwidth Cut-Off Frequency”, “The Power Mode” and, the 3 Axis enables. In the application case, ODR is set (Output Data Rate) to 200Hz, and the Cut-Off Frequency is set to 50Hz.

0	1	1	0	1	1	1	1

Table 5.7: proposed configuration for the A3G4250D “Control Register 1”.

“Control Register 2” defines the digital filters and, in particular, the High Pass Filters. Default mode configuration is the selection for this register (“Normal Mode” - 2Hz High pass filter).

0	0	1	0	0	0	1	1

Table 5.8: proposed configuration for the A3G4250D “Control Register 2”.

“Control Register 3” manages Interrupts. It is selected as a standard default mode operations.

0	0	0	0	0	0	0	0

Table 5.9: proposed configuration for the A3G4250D “Control Register 3”.

“***Control Register 4”*** manages the SPI Communication, default[71] setup is selected.

0	0	0	0	0	0	0	0

Table 5.10: proposed configuration for the A3G4250D “Control Register 4”.

[71] BLE sets the LSB Register and MSB registers addresses (pag.14 of [39]).

"Control Register 5", configures the memory management. It is kept the default mode.

0	0	0	0	0	0	0	0

Table 5.11: proposed configuration for the A3G4250D "Control Register 5".

All other registers are considered to be as default. All Output registers contain data in TWO's complement format (signed integer); this means that the data should be converted before to get meaningful information. Temperature sensor's output data consists of only one 8-bit TWO's complement format signed integer register, while each axis 16-bit gyroscope's output data is stored in 2 x 8-bit registers with the address defined by "Control Register 4". First, a conversion to "Decimal base" and then a multiplication by the device's sensitivity of 8.75m DPS per integer will produce the Gyroscope output, expressed by the equation:

$$R_t = SC \cdot (R_m - R_0)$$

(Equation 33)

where,

R_t → the actual angular rate, given in DPS

R_m → the MEMS gyroscope measurement, given in signed integer LSBs

R_0 → the zero-rate level[72] given in signed integer LSBs

SC → the scale factor (or sensitivity) given in DPS/LSB

5.1.4 Landing Proximity sensor

Landing manoeuvre is a critical operation for a controller to perform, and the likelihood of an approach error is not negligible. It suggests that shall be considered the risk associated with the landing manoeuvre.

In order to help the controller to perform the last phase of the landing manoeuvre safely, it is necessary to know when the vehicle is close to the ground; it means that the vehicle is going to arrive shortly to the touch-down. The Author's proposed solution to the problem is a ventral installation of obstacle detection sensors, which will allow detecting the soil, as "Figure 5.3" illustrates.

A design assumption is: as soon as the system detects the ground, the controller will disable the powertrain (propellers will not produce anymore trust and, in case of an impact

[72] The gyroscope output when no angular rate is applied.

of the propeller on the ground, the damages to the motor will be limited). Due to the importance of the information, The Author's proposal privileges an independent two-stage proximity detection. First stage detection operated by an obstacle sensor tuned for detection in the range of 1 to 1.5 meters, and a second stage operated by a sensor capable of higher resolution (and accuracy) but capable of detecting targets only at a much shorter distance.

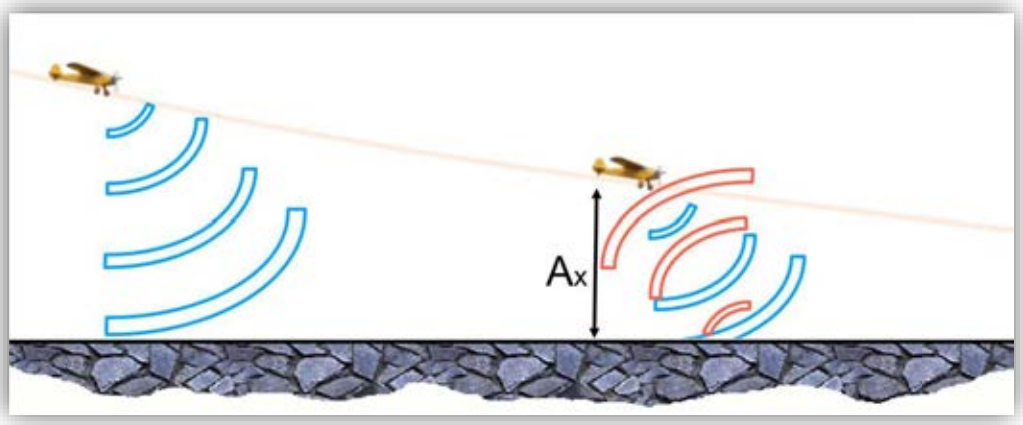


Figure 5.3: representation of landing sensor operation during the landing manoeuvre.

There are several devices capable of performing the first stage detection for the hardware proposal; the Author's selection is the "ST VL53L1X" plug-in board. It is a "Time of Flight" (ToF), laser-ranging sensor, enhancing the ST "FlightSense™" product family. It is also possible to program the size of the Region of interest (ROI) on the receiving array, allowing a reduced sensor's "Field of View" (FoV).

The VL53L1X has three distance modes (DM): short, medium, and long. Long-distance mode allows reaching the longest possible ranging distance of 4 m. However, this maximum ranging distance is impacted by ambient light. Short distance mode is more immune to ambient light, but its maximum ranging distance is typically limited to 1.3 m and ranging frequency up to 50 Hz. This solution results in the optimal solution for the application, and it is assumed that the system will detect the ground when the vehicle elevation is in the range between 120cm and 140 cm.

VL53L1X is designed to operate with the VL6180X; it is the latest product based on ST's patented "FlightSense™" technology. It is a ground-breaking technology allowing absolute distance to be measured independent of target reflectance. Instead of estimating the distance by measuring the amount of light reflected back from the object (which is significantly influenced by colour and surface), the VL6180X precisely measures the time the light takes to travel to the nearest object and reflect back to the "ToF" sensor. The

detection distance is in the range of 10 cm. Although the detection distance may look very limited, it is crucial to highlight that this information not only has a safety purpose (redundant sensor for safety-critical function) but allows future controller improvements on the landing manoeuvre.

It possible to conclude that this approach allows the designer to be confident that at the touch-down, the powertrain will be disabled and gives enough flexibility for future controller quality improvements.

5.1.5 Navigation Monitor

A human being pilot to monitor its route relies on a multitude of information; it is possible to highlight a few significant ones:

a) the estimated geophysical position or the "Global Positioning System" localization (GPS coordinates);
b) coordinates of the final destination;
c) the vehicle's altitude;
d) the vehicle's direction;
e) the vehicle's speed.

Given the destination coordinates and altitude, a standard GPS Module can supply all the requested relevant information. Acknowledging the vehicle's flight dynamics, the Author advises having a redundant altimeter in order to increase the quality of the data and reduce the error typical of the GPS altitude measurements. A cost-effective solution is the combination of a GPS module with a piezoresistive absolute pressure sensor which functions as a digital output barometer. The Author's opinion is that optimal utilisation of these sensors requires a reading of the digital output barometer value and the GPS values before starting the take-off manoeuvre; then use these data as calibration information to identify the "0m" altitude reference value. It is likely to observe discordant values, in order to increase the quality of the information, Author privileges to broadcast to the controller a weighted average altitude value to the next stage.

In front of a predefined flight route, the expectation is that the vehicle will accordingly reach a flying altitude and keep the target altitude until that the vehicle is close enough to the final destination to approach the landing manoeuvre. Landing manoeuvre will utilize a reference altitude reference value, a function of the final target destination's distance.

It is assumed that the core controller will use an altitude “approximation error” (or “relative error”), a function of the reference altitude value and the perceived vehicle altitude as for “Equation 34”.

$$Altitude_{error} = \frac{Altitude_{reference} - Altitude_{read}}{\left(\frac{Altitude_{reference} + Altitude_{read}}{2}\right)}$$

(Equation 34)

5.1.5.1 GPS Module, Teseo-LIV3F

There is a wide range of available plug-in GPS modules for the “hardware proposal”, the Author’s benchmark is the ST Teseo-LIV3F module. It is an easy to use “Global Navigation Satellite System” (GNSS) standalone module, connectable with I^2C or UART communication port.

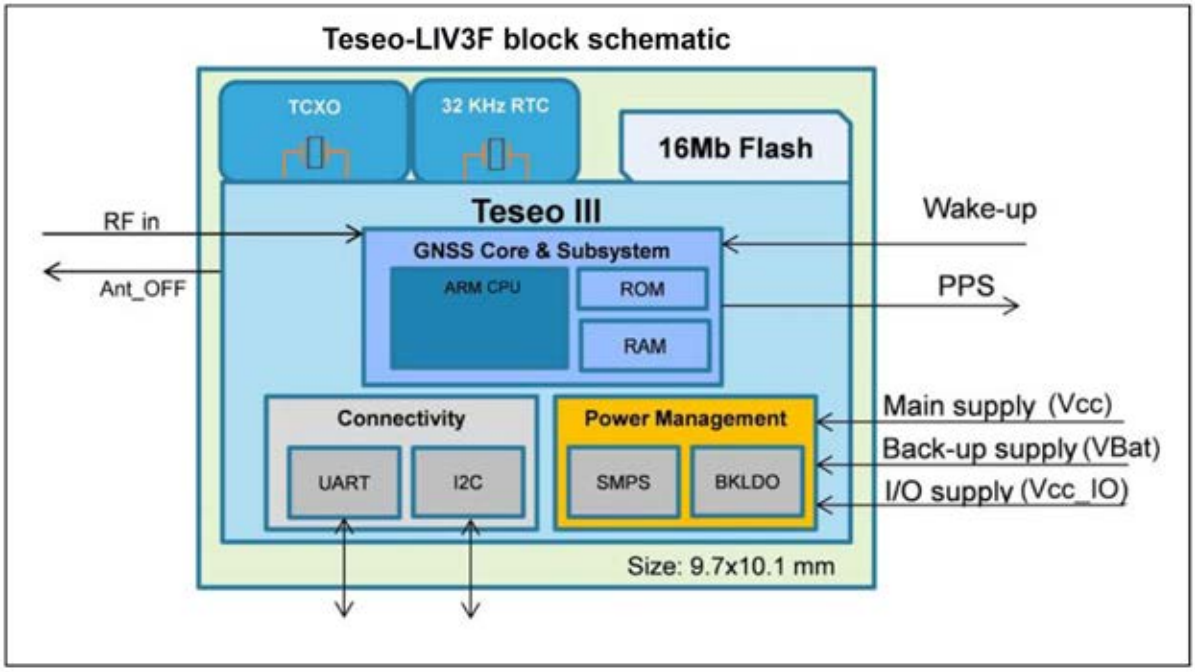


Figure 5.4: ST Teseo-LIV3F module, block scheme.[40]

This device results in a cost-effective plug-in solution for any robotic applications; it is capable of delivering, with enough accuracy, the essential information to establish a navigation monitoring interface, such as:

- altitude;
- geographical position (Geographic Coordinates);
- heading angle;
- speed.

Parameter	Specification	GPS & GLONASS	GPS & BeiDou	GPS & Galileo	Unit
Time To First Fix[73]	Cold start Warm start	<32	<36	<30	s
		<25	<29	<26	
	Hot start	<1.5	<2.5	<2	
Sensitivity [74] [75] [76] [77]	Tracking	-163	-163	-163	dBm
	Navigation[78]	-158	-158	-158	
	Reacquisition[79] [80]	-156	-156	-156	
	Cold start	-147	-147	-147	
	Warm start	-148	-148	-148	
	Hot start	-154	-151	-154	
Max fix rate	—	10	10	10	Hz
Velocity accuracy[81]	—	0.01	—	0.01	m/s
Velocity accuracy[82]	—	0.1	—	0.1	m/s
Heading accuracy[67]	—	0.01	—	0.01	°
Heading accuracy[68]	—	2.3	—	2.4	°
Horizontal position accuracy[83]	Autonomous	<1.8[69]	<1.5[69]	—	m
	SBAS	<1.5[69]	—	—	
Accuracy of time pulse	RMS				
	99%	±12.4	±29.0	±21.8	ns
Frequency of time pulse	—	1	1	1	Hz
Operational limits[84]	Dynamic[85]	<4.5g	<4g	<4.5g	—
	Altitude[86]	18000	18000	18000	m
	Velocity[72]	515	515	515	m/s

Table 5.12: ST Teseo-LIV3F module datasheet summary.

The previous table is extracted from the ST Teseo-LIV3F module datasheet and summarizes the most relevant device characteristics information. It is imperative to highlight that the vehicle initialisation shall include a time of at least 36s in order to allow the module to lock enough satellites signals[87]. Only after this waiting time, the controller can begin the

[73] All satellites at -130 dBm - TTFF@50%.
[74] Demonstrated with a good external LNA.
[75] For hot start, all sats have the same signal level except one (pilot sat @-145 dBm).
[76] For BEIDOU tracking sensitivity refers to MEO sats. For GEO the tracking sensitivity is -151 dBm. For GALILEO the signal level refers to both pilot and data components.
[77] For GALILEO the signal level refers to both pilot and data components.
[78] Configurable Value.
[79] All satellites at same signal level.
[80] Minimum level to get valid fix after reacquisition.
[81] 50% @ 30 m/s - linear path.
[82] 50% @0.5 g - shape path.
[83] CEP 50%, 24h static, Roof Antenna.
[84] Verified the limit checking the fix availability.
[85] Special configuration for high dynamic scenario.
[86] ITAR limits.
[87] Teseo Module cold start may require up to 36 s.

initialisation and perform the altitude calibration task. It is unavoidable for the controller to set the "0m" altitude reference value every time before starting the take-off manoeuvre. It is meaningful to report the absence of an altitude accuracy indication; this information limitation (lack of accuracy) reinforces the assumption made to use a redundant altimeter in order to mitigate the data inaccuracy.

A valuable device feature is the capability of data-logging[88]. Teseo-LIV3F receiver[89] can, locally, save the resolved GNSS position on the internal flash memory in order to be retrieved on demand from the host. The recorded data is configurable; data-logging supports three types of data logged, and each type has a different size and different data-logged[90]. [40]

All the data logged types have: timestamp, latitude and longitude, while other fields depend on the type. "Table 5.13" enlightens the details.

Type	Size	Altitude	Odometer	Geo	Quality	Qual_idx	Fix	Speed
1	12	—	—	X	—	X	X	—
2	16	X	—	X	X	—	X	X
3	20	X	X	X	X	—	X	X

Table 5.13: ST Teseo-LIV3F module data logging types.[40]

5.1.5.2 Redundant Altimeter

The Author's preference is to use a cost-effective device capable of estimating accurate enough the vehicle flying altitude; LPS25HB[91] results in a suitable choice since it is widely available on target application PCB (similar alternative hardware solutions[92] are available on the market). It is a piezoresistive absolute pressure sensor that functions as a digital output barometer. The device incorporates a sensing element and an Integrated Circuit (IC) interface, which communicates through I^2C or SPI from the sensing element to the application. The sensing element, which detects absolute pressure, consists of a suspended membrane manufactured using a dedicated process developed by ST.

[88] Datalogging can be enabled, disabled and erased using proprietary NMEA runtime commands. Datalogging subsystem supports both: Circular buffer and Standard buffer.
[89] Teseo-LIV3F supports only one datalog at a time.
[90] Teseo-LIV3F can support until 12 hours logging using "log-type 1" and fix-rate at 1 Hz.
[91] The LPS25HB is available in a full-mould, holed LGA package (HLGA). It is guaranteed to operate over a temperature range extending from -30 to +105 °C. The holed package allows external pressure to reach the sensing element. The LPS25HB is a high resolution, digital output pressure sensor packaged in an HLGA full-mould package.
[92] For example, SparkFun Electronics MPL3115A2 PRESSURE/ALTIMETER P/N SEN-11084 or Parallax ALTIMETER MODULE MS5607 P/N 29124.

The comprehensive device includes a sensing element based on a piezoresistive "Wheatstone Bridge" approach and a sensor's interface, which communicates a digital signal from the sensing element to the application. "Figure 5.5" clarifies the IC block scheme.

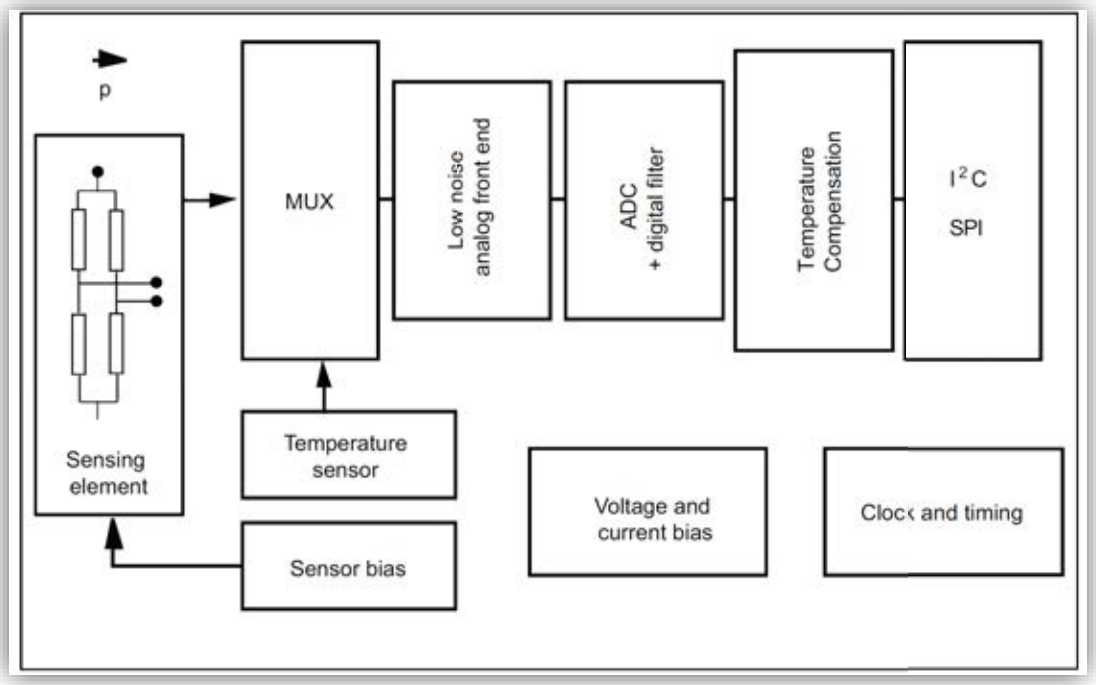


Figure 5.5: LPS25HB block scheme. [41]

The device has a large number of read-only registers and read/write registers, and the most significant ones are: "Control Register 1" (20h), "Control Register 2" (21h), PRESS_OUT_H (2Ah), PRESS_OUT_L (29h) and PRESS_OUT_XL (28h).

"Control Register 1" (address: 20h) defines the "Power-down control", the "Output data rate selection" (ODR [2:0]), the "Interrupt generation enable", the "Block data update" (BDU), the "Reset Auto-zero function" and the "Serial Interface Mode selection" (SIM - SPI). The register is set in order to ensure the device "active mode" (bit 7, PD = 1), with a 12.5Hz frequency (ODR [2:0] set to "011") continuous update (BDU, "0" default value); other parameters are kept as default, ensuring the four-wire SPI operation mode (SIM, "0" default value). "Table 5.14" clarifies the register's setting.

1	0	1	1	0	0	0	0

Table 5.14: proposed configuration for the LPS25HB "Control Register 1".

"Control Register 2" (address: 21h) defines: the "Reboot memory content", the "FIFO enable", the "STOP_ON_FTH93", the "Enable to decimate the output pressure to 1Hz with FIFO Mean mode", the "I^2C interface enable", the "Software reset", the "Auto-zero enable", and the "One shot mode enable". Except for the I^2C interface control forced to "Disabled", all bits will be kept as default (set to "0") in order to keep the "normal mode operation". Table 5.15 elucidates the register's setting.

0	0	0	0	1	0	0	0

Table 5.15: proposed configuration for the LPS25HB "Control Register 2".

The pressure data are stored in three registers: PRESS_OUT_H (2Ah), PRESS_OUT_L (29h) and PRESS_OUT_XL (28h). The value is expressed as "TWO's complement". The information translation in hPa (pressure) requires that the algorithm shall take the "TWO's complement" of the complete word and then divide it by 4096 hPa. As described in the below picture. [41]

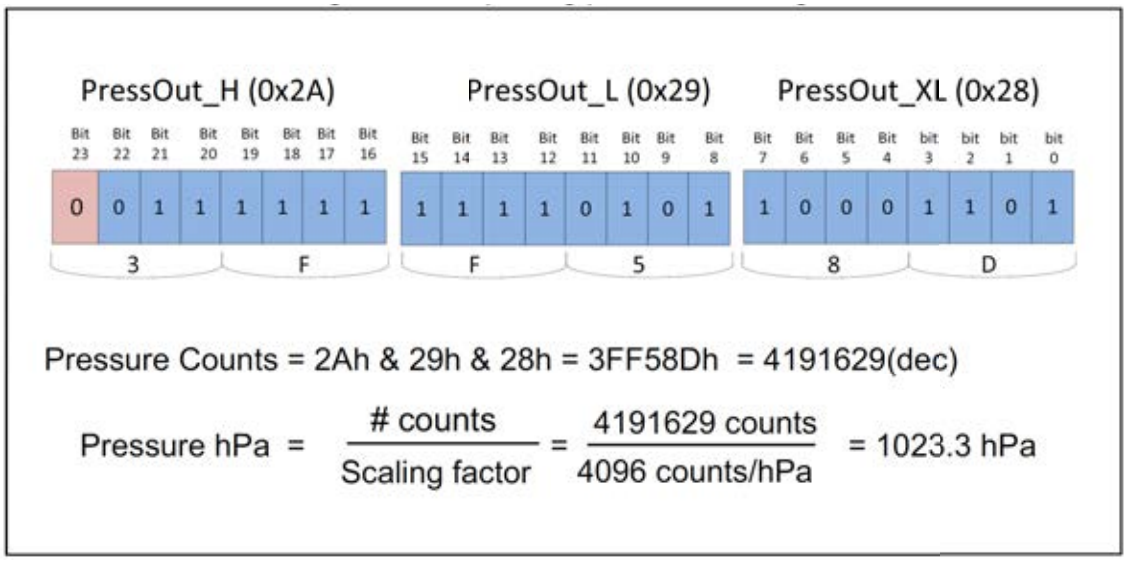


Figure 5.6: LPS25HB output pressure algorithm example [41].

Moving by the consideration that the following table and graph illustrate the relationship between altitude and pressure using the "default values"[94] for pressure and temperature at sea level. Then considering that by definition, the altitude at a given air pressure can be calculated using "Equation 35" for an altitude up to 11 km (36,090 feet), the Author proposes the use of

[93] "Enable the FTH_FIFO bit in FIFO_STATUS (2Fh) for monitoring of FIFO level".
[94] Using ISA standards, the defaults for pressure and temperature at sea level are 1013.25 hPa and 288 K.

such information as a baseline and to operate a linearization in the limited RC plane operational altitude.

$$h = h_b + \frac{T_b}{L_b} \cdot \left[\left(\frac{P}{P_b} \right)^{-\left(\frac{R \cdot L_b}{g_0 \cdot M} \right)} - 1 \right]$$

(Equation 35)

Where:

P → dynamic pressure [Pa]

P_b → static pressure (pressure at sea level) [Pa]

T_b → standard temperature (the temperature at sea level) [K]

L_b → standard temperature lapse rate [K/m] = -0.0065 [K/m]

h → height about sea level [m]

h_b → height at the bottom of the atmospheric layer [m]

R → universal gas constant = 8.31432 [(N·m)/(mol·K)]

g_0 → gravitational acceleration constant = 9.80665$\left[\frac{m}{s^2}\right]$

M → molar mass of Earth's air = 0.0289644 [kg/mol]

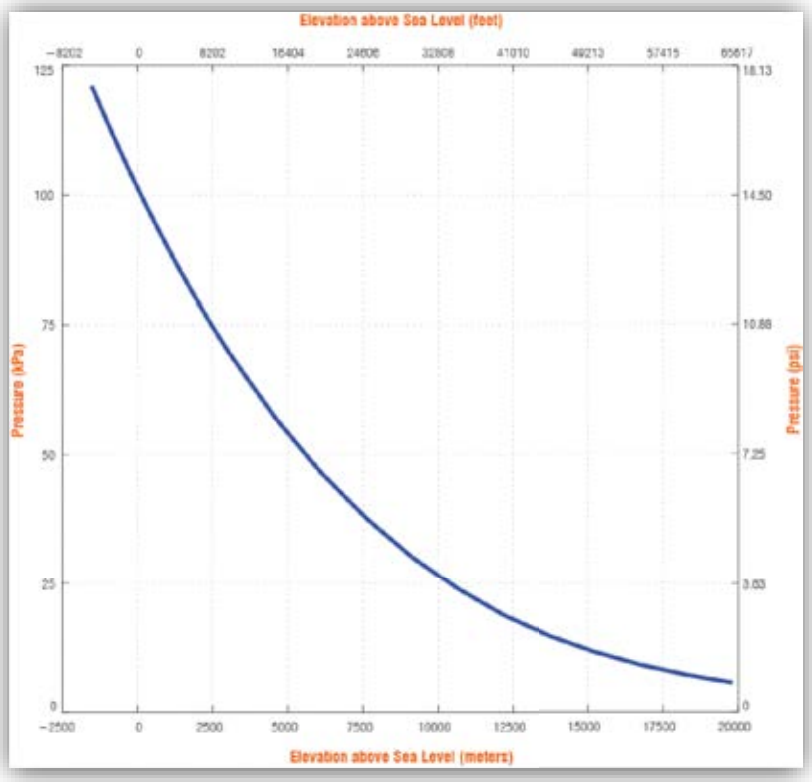


Figure 5.7: graphical representation of "Equation 35".

Altitude above Sea Level			Absolute Atmospheric Pressure		
feet	miles	meters	kPa	atm	psia
-5000	-0.95	-1524	121.0	1.19	17.55
-4000	-0.76	-1219	116.9	1.15	16.95
-3000	-0.57	-914	112.8	1.11	16.36
-2000	-0.38	-610	108.9	1.07	15.79
-1000	-0.19	-305	105.0	1.04	15.24
-500	-0.09	-152	103.2	1.02	14.96
0	0.00	0	101.3	1.00	14.70
500	0.09	152	99.5	0.98	14.43
1000	0.19	305	97.7	0.96	14.17
1500	0.28	457	96.0	0.95	13.92
2000	0.38	610	94.2	0.93	13.66
2500	0.47	762	92.5	0.91	13.42
3000	0.57	914	90.8	0.90	13.17
3500	0.66	1067	89.1	0.88	12.93
4000	0.76	1219	87.5	0.86	12.69
4500	0.85	1372	85.9	0.85	12.46
5000	0.95	1524	84.3	0.83	12.23
6000	1.14	1829	81.2	0.80	11.78
7000	1.33	2134	78.2	0.77	11.34
8000	1.52	2438	75.3	0.74	10.92
9000	1.70	2743	72.4	0.71	10.51
10000	1.89	3048	69.7	0.69	10.11
15000	2.84	4572	57.2	0.56	8.29

Table 5.16: parametric solution of "Equation 35".

Assuming that the RC plane will operate in an altitude range between "0 m" and "305m", the Author assumes a linearization accurate enough for the system. The linearization process upshot asserts that the pressure will decrease by about 11.9hPa (or 1.19kPa) every 100m (or will decrease by 0.119hPa each meter).

To make the conversion assumption robust, before the take-off, it is necessary to correlate the pressure value read before the manoeuvre start to the "0 m" altitude parameter. It means that at the system turn-on shall follow the barometric altimeter initialisation, which performs the altitude calibration task. During the calibration, the first reading of the first pressure output, expressed in "hPa", shall be stored in the FPGA's internal RAM (or in any other functional memory)[95]. Author preference is to make the altimeter's SPI interface externally accessible to the user. This feature will allow the user to calibrate the device and store the calibration data in a dedicated memory unit, together with all other flight planning parameters. The algorithm's output value is the solution of the following equation.

[95] The reference pressure value in "hPa" linked to the "0m" altitude reference value.

$$Altitude_{LPS25HB} = \frac{P_{0m,hPa} - P_{LPS25HB,hPa}}{0.119\ \left(\frac{hPa}{m}\right)}$$

(Equation 36)

Where:

$P_{0m,hPa}$ → 0 m pressure reference value [hPa]

$P_{LPS25HB,hPa}$ → dynamic pressure read [hPa]

5.1.6 Electronic Compass Unit

The implementation of the Electronic Compass peripheral is software achieved using the information of the 3-axis accelerometer, the gyroscope and the GPS Module.

5.1.7 Motor Drive - Powertrain

The Author's academic researches [17 and 42] focus on new technologies of "Motor Drives" and the state of the art of "WBG" technologies for powertrain applications. Notably, research works focused on studying new and more efficient technology for "AUAV Motor Drive"[96] and a cost/performance/reliability comparison with conventional "Si" Technology conventional alternative.

For the proposed "AUAV Motor Drive", priority is given to a reliable oriented solution, in line with [42] conclusions. A set of stand-alone "ST Microelectronics Eval BRD"[97], based on "Si" technology power elements, is chosen. It is a low-cost solution that does not require developments or ASIC implementations.

The peripheral is fully capable of controlling the "E-Motor's Torque", and it is interfaced with the FPGA via an RS-232 port (communication standard). FPGA unit (thus the proposed

---

[96] The chosen parameters are in line with the current requirements for a small UAV or Small Hybrid Robots that operate on DC battery systems, maintaining the Integrity of the Specifications [42]. Research work moved by the prerequisites of driving compatible 3-ph brushless motor in line with what required by the AUAV project in the object of the doctoral work.

Main parameters summary:

- Form-factor: Smaller than (W127mm x H80mm x L127mm)
- Converter topology: 3 Phase Inverter
- Input voltage: 20 volts to 48 volts
- Nominal input: 36Volts
- Battery Configuration: 10 x 4.2 VDC Li-Ion 5Ah Batteries, series configuration
- Maximum Output Current: 15Arms.

[97] The "Motor Drive Unit" includes a control unit and a power unit. The control unit can perform a full E-Motor "Speed Control" or "Torque control". User is allowed to set the control strategy together with the E-Motor settings. The power unit is capable of accepting a variety of "Power MOSFETs", but it is capable of operating only with a "Low Voltage" [16] electrical power source. It allows the user to select the MOSFETs P/N in the function of the power ratings of the E-Motor.

controller) acts as “Master” and each “AUAV Motor Drive” acts as “Slave”.It means that the central controller will enable/disable the peripherals and will dynamically set the E-Motor’s “Torque Demand”.

5.1.8 Electro-Mechanical Actuators – SERVO

Previous paragraph 4.2.2 defines, for what regards the study case, the controller’s outputs. This paragraph goal is to dissert the “electro-mechanical actuators” hardware design. Although there is not a pre-defined set of requirements, the selection process of each actuator is influenced by four key variables:

a) the actuator’s “accuracy”;
b) the actuator’s “form-factor”;
c) the actuator’s “reliability”;
d) the actuator’s “output torque”.

The necessity to utilize a small “form-factor” electro-mechanical actuator with a relatively large “torque density” suggests using the SERVO-Motor topology. A device with a “metal gear” is preferred to ensure the system’s reliability.

By definition, the **ailerons control** requires two complementary actuators; the selection of the SERVO-Motor P/N is restricted by the space available for the installation inside the “wing’s frame”. “Form-factor” results being the primary driver and a particular “slim SERVO-Motor” with metal gear, specifically designed to fit inside tiny UAV wings, is desirable. **Hitec HS-125MG** slim metal gear wing SERVO-Motor is selected for the **ailerons control** because it represents a good trade-off between costs, performances, reliability and size.

Elevator control may use a single or a dual actuator in the function of the UAV mechanical design. There are similarities with the aileron’s actuators technical requirements and physicals constraints (such as space available and packaging issues). In order to simplify the algorithms and reduce the vehicle’s BOM, the elevator will be actioned by a single actuator: a single “Hitec HS-125MG” slim metal gear wing SERVO-Motor.

Although the same SERVO-Motor model controls ailerons and elevator, each function requires a specific gear that will act as a torque converter and actuator’s operating angle adapter. The following figure summarizes the “Hitec HS-125MG” technical specifications.

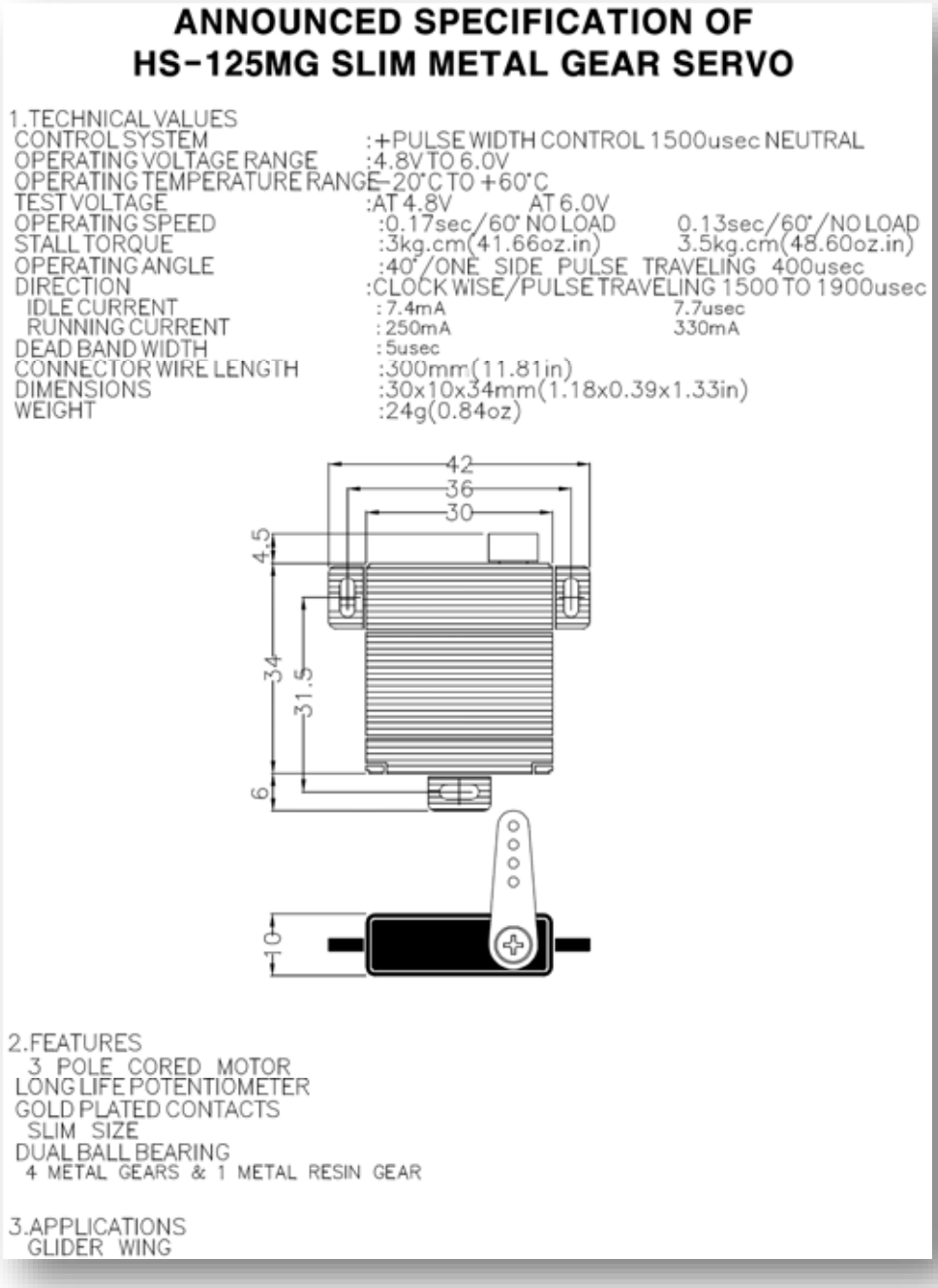
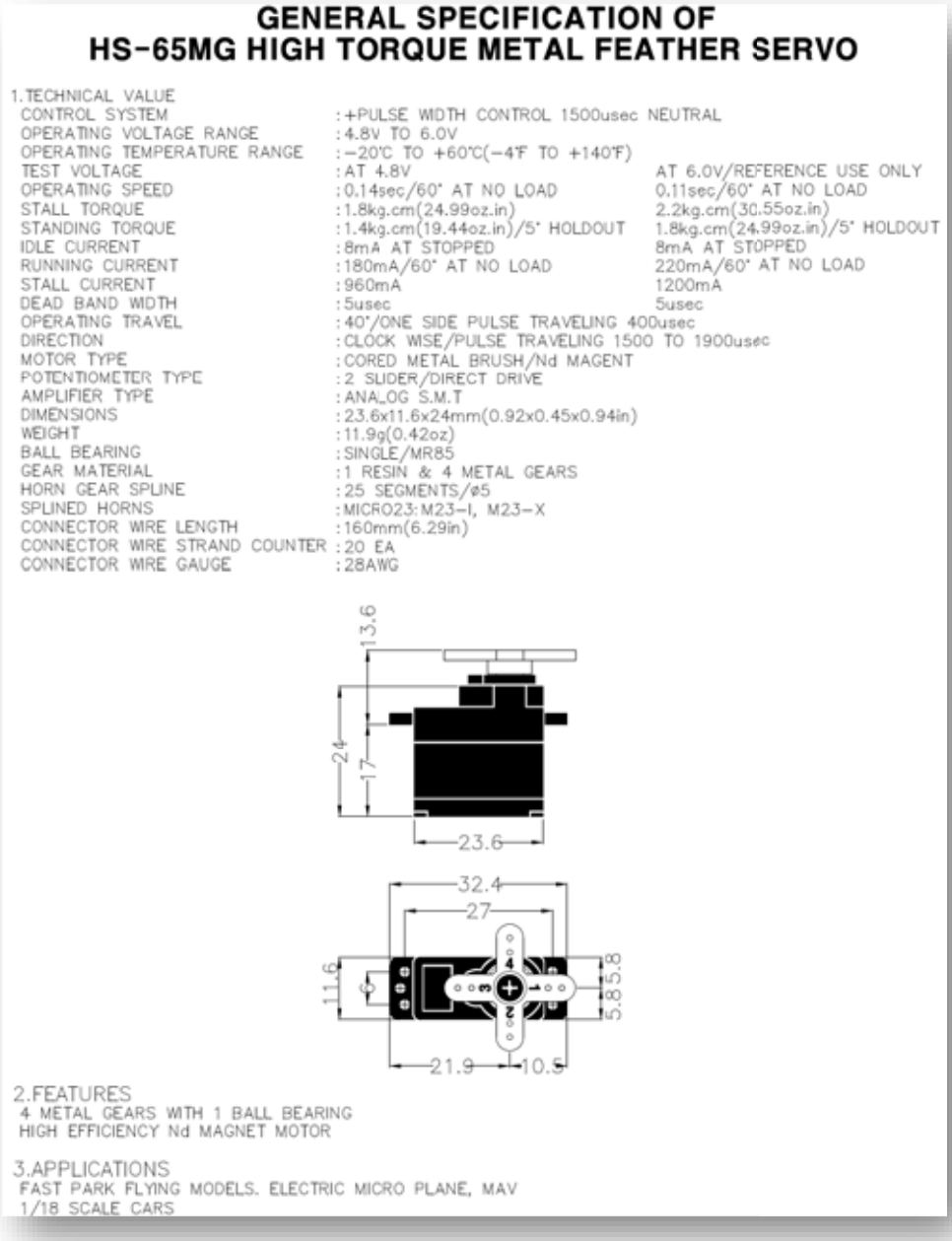


Figure 5.8: "Hitec HS-125MG" technical specifications summary.

Rudder controller may use a single or a dual actuator, depending on the UAV mechanical design. The study case uses a dual rudder design with a single control signal (parallel configuration; the same control signal controls both rudders). Space constraints are

not so restrictive[98] as for the ailerons because a high output torque is not mandatory for this application; a low cost oriented solution is desired. **HS-65MG** metal gear servo is chosen for the **rudder control** because it represents a good trade-off between costs, performances and reliability.



GENERAL SPECIFICATION OF
HS-65MG HIGH TORQUE METAL FEATHER SERVO

1.TECHNICAL VALUE

CONTROL SYSTEM	:+PULSE WIDTH CONTROL 1500usec NEUTRAL	
OPERATING VOLTAGE RANGE	:4.8V TO 6.0V	
OPERATING TEMPERATURE RANGE	:-20°C TO +60°C(-4°F TO +140°F)	
TEST VOLTAGE	:AT 4.8V	AT 6.0V/REFERENCE USE ONLY
OPERATING SPEED	:0.14sec/60° AT NO LOAD	0.11sec/60° AT NO LOAD
STALL TORQUE	:1.8kg.cm(24.99oz.in)	2.2kg.cm(30.55oz.in)
STANDING TORQUE	:1.4kg.cm(19.44oz.in)/5° HOLDOUT	1.8kg.cm(24.99oz.in)/5° HOLDOUT
IDLE CURRENT	:8mA AT STOPPED	8mA AT STOPPED
RUNNING CURRENT	:180mA/60° AT NO LOAD	220mA/60° AT NO LOAD
STALL CURRENT	:960mA	1200mA
DEAD BAND WIDTH	:5usec	5usec
OPERATING TRAVEL	:40°/ONE SIDE PULSE TRAVELING 400usec	
DIRECTION	:CLOCK WISE/PULSE TRAVELING 1500 TO 1900usec	
MOTOR TYPE	:CORED METAL BRUSH/Nd MAGENT	
POTENTIOMETER TYPE	:2 SLIDER/DIRECT DRIVE	
AMPLIFIER TYPE	:ANALOG S.M.T	
DIMENSIONS	:23.6x11.6x24mm(0.92x0.45x0.94in)	
WEIGHT	:11.9g(0.42oz)	
BALL BEARING	:SINGLE/MR85	
GEAR MATERIAL	:1 RESIN & 4 METAL GEARS	
HORN GEAR SPLINE	:25 SEGMENTS/ø5	
SPLINED HORNS	:MICRO23:M23-I, M23-X	
CONNECTOR WIRE LENGTH	:160mm(6.29in)	
CONNECTOR WIRE STRAND COUNTER	:20 EA	
CONNECTOR WIRE GAUGE	:28AWG	

2.FEATURES
4 METAL GEARS WITH 1 BALL BEARING
HIGH EFFICIENCY Nd MAGNET MOTOR

3.APPLICATIONS
FAST PARK FLYING MODELS. ELECTRIC MICRO PLANE, MAV
1/18 SCALE CARS

Figure 5.9: "Hitec HS-65MG" technical specifications summary.

[98] Technical requirements for the SERVO-Motor selection are less stringent if compared to the ailerons and the technical rudder requirements.

Both, the **HS-65MG** metal gear SERVO-Motor and the **HS-125MG** slim metal gear wing SERVO-Motor, although they are different SERVO-Motors, could be controlled by a similar PWM signal. It means that it is possible to define only one generic VHDL component (algorithm) replicable for each SERVO-Motor interface. The postulation is that this particular VHDL algorithm will be named as "SERVO Control Signal Generator VHDL component".

The VHDL algorithm generates an output SERVO control signal made out of two parameters:

- refresh frequency of 20ms;
- pulse width range goes from 1.5ms to 1.9ms (provided by the manufacturer).

"Figure 5.8" and "Figure 5.9" SERVO-Motors datasheet's extracts show that both, the "HS-65MG" metal gear SERVO-Motor and the HS-125MG slim metal gear wing SERVO-Motor, have an operative angle of 40 degrees and a pulse control range of 400µs ("Figure 5.10" presents the acceptable control waveforms; pulse fluctuating from 1500µs to 1900µs).

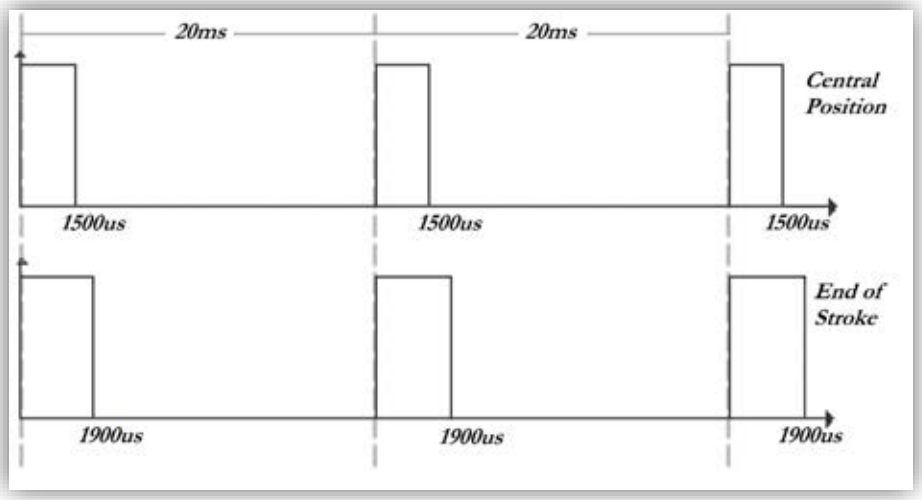


Figure 5.10: HS-65MG and HS-125MG control waveforms.

The acceptable resolution for the SERVO control signal (which results in the quantity of position SERVO-Motor can take) could be 7-bit or 8-bit, resulting in 128 or 256 positions.

Therefore, the input clock frequency for the VHDL component "SERVO control signal generation" should be:

$$f_{servo,7bits} = \left(\frac{resolution}{pulse\ control\ range}\right) = \left(\frac{128}{400\mu s}\right) = 320kHz$$

(Equation 37)

or

$$f_{servo,8bits} = \left(\frac{resolution}{pulse\ control\ range}\right) = \left(\frac{256}{400\mu s}\right) = 640kHz$$

(Equation 38)

A 7-bit control resolution will correspond to a theoretical angle step of 0.3125 degrees (control pulse step width equal to 3.125µs), while the 8-bit control resolution will correspond to a theoretical angle step of 0.15325 degrees (control pulse step width equal to 1.5625µs).

By definition, the "Dead Band Width" is used by the manufacturers to avoid servo dancing at its centre position by telling it to stay in position until the difference between new command and old command is greater than the "Dead Band Width". Figure 5.8 and Figure 5.9 datasheet's extracts define to 5µ the SERVO's "Dead Band Width", increasing the SERVO control signals resolution from 7-bit to 8-bit most likely will not produce any practical improvement. It inspires the use of a VHDL component, "SERVO Control Signal Generator", with a digital signal resolution of 7-bit and a clock of 320kHz (Equation 37).

It is imperative to highlight that the proposed FPGA uses a 3V3 I/O interface, and the proposed SERVO-Motors logical circuits require a 5V logical level interface. It means that a "Not Inverting Logic Level Translator" or a "Not Inverting Gate Driver" should be installed between the FPGA and the "SERVO-Motor". Therefore, the circuit design needs to protect the "Control Unit" (particularly the FPGA) from external events[99].

5.1.9 Data Storage

A complex system like the technical proposal of the "Thesis" requires dedicated memory, where to store a wide range of data. Indeed, it is indispensable a data storage element, accessible to the user, where to upload the flight parameters or operate the sensors' calibration[100]. This operation results indispensable to program, time by time, the flight route. It is also necessary to allocate a second memory unit where to store all the flight telemetry to use for the controller learning/training purpose or just for a flight parametric analysis.

The Author's preference is to use a standard SPI interfaced memory. A wide range of compatible devices is capable of fulfilling the task's requirements, although each technology available results optimised for a specific set of applications. The Author considers the EEPROM memory technology an excellent trade-off for the technical proposal, privileging

[99] Such as ESD events, short to battery, short to ground, etc...
[100] For example, a place where to store the "0m" value of the redundant altimeter calibration, etc...

the reliability instead of larger memory sizes. The assumptions made is to use a set of two EEPROM devices with a memory size of 2Mbit, associable to the “ST M95M02-A125”. In the case will be required a larger memory size for the data storage element, the Author acknowledged in a standard SPI interface “NOR Flash Memory” the technological boost for this application.

5.1.10 Battery management and Low Voltage power supply management

As the main goal is to create a controller in a VHDL environment able to replicate human decisions and behaviours in the control of a small UAV, a limited, but not negligible, consideration should be given to the “System Power Management”.

At first look, “System Power Management” should be focusing primarily on the system protection in case of “Battery Overcharged” or “Battery Discharged”, nevertheless it is not limited only to these operations. Indeed, a human pilot may change the way of driving the vehicle by external variables in order, for instance, to save energy. The estimated SoC value is the crisp value of an input membership function which will be used as a secondary variable in the rule block for the powertrain motor throttle control.

For the study case, it is assumed that the only vehicle source of power is the “Main Battery”, which is defined as “Low Voltage REESS” because it has a maximum output voltage below 60V [16]. From the last statement, it is possible to adapt the architecture described in “paragraph 3.2” for the proposal. The safety legislation and the international standards do not require a “galvanic isolation” between the main source of power (REESS, primary PSU) and the controller electronics (secondary PSU) when the primary source of energy is defined as “Low Voltage” (according to “Chapter 3” analysis). Dedicated power electronics circuitries, or “secondary PSU block”, are required to convert the “REESS Voltage” to several secondary low voltage power rails in order to power-up all the vehicle’s electronic components.

It assumed that the “Secondary PSU Block” shall operate with an input voltage in the range between 24VDC and 48VDC and generate the following secondary power rails:

- 12V ± 5%, rated to 2A;
- 5V ± 2%, rated to 2.2A;
- 3V3 ± 2%, rated to 2.2A;
- 1V2± 2%, rated to 4A.

As well, it is assumed a "Secondary PSU Block" design compliant to the following basic requirements:

- ESD (IEC 61000-4-2, level 4);
- load dump protection (ISO 16750-2, Test A and B);
- pulse (ISO 7637-3, Pulse 3a and 3b).

In the automotive industries, it is common to have similar requirements for the low voltage power management of the vehicle's ECUs (Electronic Control Unit). A widely used design strategy is shown in Figure 5.11. For the HW proposal, the Author advocates using slightly over-rated power supplies to increase the system's efficiency and robustness. The implementation of the block diagram drawn in Figure 5.11 represents a feasible option, compatible with the power budget typical of a system like a vehicle's control unit.

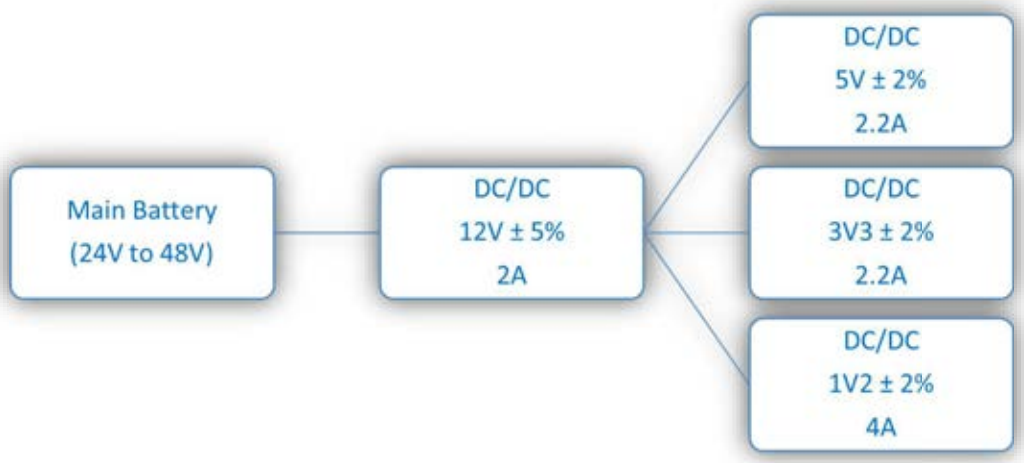


Figure 5.11: proposed HW PSU block diagram for the control unit.

"Paragraph 3.7" described the structure of automotive REESS ("Automotive Powertrain Battery Pack") and of "Battery Management Systems". The most common architecture described is based on multiple "Battery Modules" connected in parallel and in series accordingly to the target application. In this architecture, each "Battery Module" has a dedicated "EBM module" responsible for cell monitoring and balancing.

The vehicle's "REESS", which is the unique system's energy source, due to the limited complexity[101] and due to the low voltage nature of the system, is reduced to a single battery module (thus a single EBM).

[101] In the specific, the total number of cells in series per EBM less than 16.

For the proposal hardware setup, the assumptions are:

- the use of a single EBM module (electronic battery monitor) instead of a more complex BMS;
- that the EBM uses an ASIC chip (there is a wide range of devices available in the market);
- the EBM is capable of enforcing several protection features and is capable of broadcasting the battery module parameters via an SPI interface.

These assumptions require a dedicated "VHDL Block" to interface the FPGA's controller to the EBM through an SPI protocol. As well will be necessary to pre-process the EBM's information before to be transferred to the neuro-fuzzy controller.

In this configuration, the controller entirely relies on the EBM to enforce protection and safety mechanisms to the REESS. Indeed, the EBM will operate the battery cell balancing, the battery monitoring and the battery protection functions autonomously. The practical result is that the vehicle's control unit can only observe the REESS's SoC.

5.2 Human Remote Control

The principles behind utilising a human remote control have their roots in the necessity of gathering the conditions for deep controller learning through a training process. Moving from the drawn hypothesis and from the strategy described in "Chapter 4", a learning/training prerequisite is that an RC plane controlled by a human will generate and record the "raw data". Telemetry, stored into a dedicated memory, at the end of the flight shall be opportunely recovered (extracted from the external physical memory to the designer PC), verified and then loaded to into the "GUI" that, as an ultimate outcome, will build (and export in VHDL language) the final "neuro-fuzzy controller algorithm".

This approach entails that the RC plane hardware needs to fulfil a set of technical requirements, such as:

- a human pilot shall control the RC plane via a remote radio-controller;
- the whole hardware (control unit) described in paragraph 5.1 shall be integrated with the remote control unit hardware;
- the fuzzy logic controller shall be flexible enough to be opportunely modified to read, process and store all vehicle's actuators and flights parameters.

The strategy behind these requirements is based on the compulsion to elaborate a manoeuvre with the environmental parameters (system input parameters), with the human

pilot behaviours (pilot commands), the navigation monitoring parameters and, the flight's parameters.

The Author proposal solution relies on the FPGA's flexibility and the controller's hardware description language (VHDL). For these reasons, the VHDL controller's algorithm shall include the following features:

- a dedicated "Flight Parameters Memory Interface Block", in charge of the SPI interface that connects the FPGA to the memory storage device (an independent 2Mbit EEPROM);
- a dedicated "Telemetry Memory Interface Block", in charge of the SPI interface that connects the FPGA to the memory storage device (an independent 2Mbit EEPROM);
- "Telemetry Memory Interface Block" receives as input the "Fuzzy Controller Inputs", the "Fuzzy Controller Output", the status of the physical actuators and powertrain torque demands.

Specialised RC plane component stores offer a wide range of affordable remote control units, complete of actuators and actuators drivers, capable of plug and play solutions. This detail, in conjunction with the fact that the "PhD Thesis focus" is the research on "Neuro-Fuzzy Controllers" to be designed using VHDL as "controller's hardware description language", make marginal a detailed description of the plug-in remote control units and their actuator drivers.

Such kind of plug-in systems are easily exchangeable, and usually, users privileges customised solutions in order to maximise their comfort[102] or their entertainment. The Author decision is: to do not restrict the selection process to a particular remote control system for the RC plane. The efforts will focus merely on the VHDL controller algorithm functionality since it is the only relevant research.

[102] It may be useful to remind that the human pilot of the RC plane, perhaps is not trained to use a particular system, or he will be more effective with a solution on which he matured most of his expertise.

6 Study Case, Controller's Design Proposal

As previously described, the proposed work scope is to minimise the hardware design giving the most of the design load to the VHDL-Neuro Fuzzy controller. This design strategy moves by the minimisation of the sensors installed on the vehicle, seeking for a cost-effective compromise between mechanical constrains, electronics hardware/sensors available and flight control principles.

"Chapter 2" introduced the flight's control principles, while "Chapter 5" articulated the hardware proposal. "Chapter 4" described the academic and theoretical basis of the decision-making process that leads to the adoption of basic fuzzy logic principles and neural networks (as a potential output of the learning process) for the "study case".

As previously described, the first step of the controller's design is the identification of the "System's Inputs", as "System's Environment Variable"[103] or as "Shell Variable"[104], the "Actuators" or "System's Outputs" and then link them with a "Transfer Function". As the proposed work targets a neuro-fuzzy controller, the "System Transfer Function" is the set of all MIFs, all MOFs and all Rulebases[105] (FIS). [43]

6.1 Controller's Inputs

Paragraph 4.2 explores the controller's technical requirements, defining a set of "controller's inputs" and "controller's outputs". Each controller's input represents a "Physical Quantity"[106], and to be utilisable by an FPGA/VHDL system shall be adequately expressed in a digital form. It means that for each system's input, or output, will correspond a sensor, or an actuator, described in "paragraph 5.1". The interface between the system's peripheral and the neuro-fuzzy controller is implanted on a dedicated FPGA's section by a "VHDL component's algorithm"[107].

[103] An environment variable is a dynamic-named value that can affect the way running processes will behave on a controller.
[104] A shell variable is a variable that affects only to the current "Shell" or "Function". In contrast, an environment variable is available system-wide and can interact with other functions of the controller.
[105] Each "Rulebase" rule's weights obtained as a result of the "learning/training process".
[106] A "Physical Quantity" is a property of a material or system that can be quantified by measurements. A "Physical Quantity" can be expressed as a combination of a magnitude and a unit.
[107] An appropriate algorithm capable of configuring the sensor and manipulating the sensor's output data accordingly.

6.1.1 VHDL Component A3G4250D

Paragraph 5.1.3 described the hardware (ST A3G4250G) characteristics of the vehicle's gyroscope and then defined the operational setting on theoretical assumptions.

In order to implement a "Parallel Computing Controller" on an FPGA using VHDL algorithms, a dedicated "VHDL component" named "A3G4250D" is created. It means that a dedicated set of FPGA logic gates are programmed to interact only with the physical gyroscope, and one other set of FPGA logic gates are programmed to process and broadcast the gyroscope's information.

The hardware architecture defines the SPI protocol as a communication method and defines the "FPGA" as "SPI Master" and the peripheral as "SPI Slave". VHDL component "A3G4250D" includes a sub-component that expresses the SPI interface and physically interacts with the peripheral ("SPI_Master" VHDL component). The "A3G4250D" component includes a second sub-component (it acts as the A3G4250D's state machine), which manages the data stream[108] (interfaced to the "SPI_Master Component"), the peripheral's configuration and, the data digital-processing.

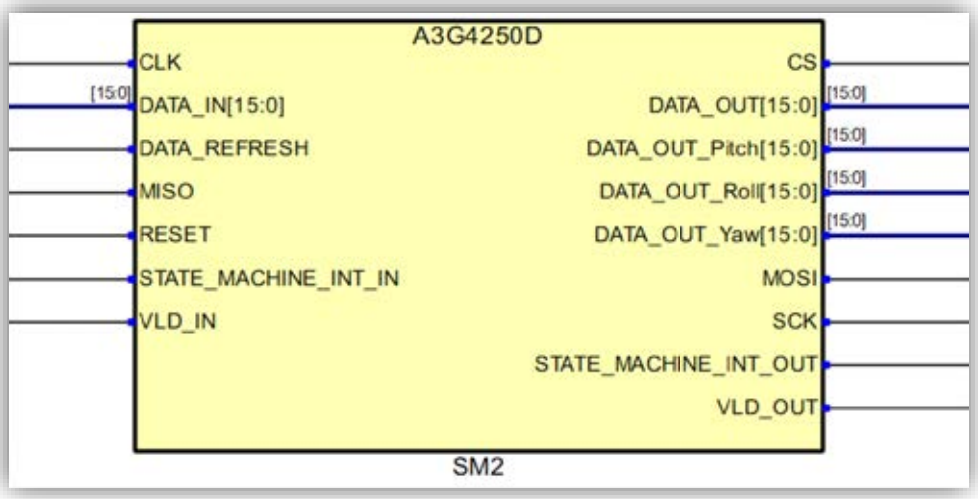


Figure 6.1: RTL view of the "VHDL component" named as "A3G4250D".

For data processing of the "A3G4250D" gyroscope's output registers, it is intended that the algorithm converts each 16-bit value expressed in TWO's complement format (signed integer) into a 16-bit STD_LOGIC_VECTOR. The outcomes are a 16-bit yaw angle, a 16-

[108] At the power-up "A3G4250D's state machine" configures the peripheral (according to paragraph 5.1.3 configuration) and then cyclically (cycles starts with a "Data Refresh" command) reads and processes the peripheral's output data registers.

bit pitch angle and a 16-bit roll angle coming out from the “VHDL component” (as illustrated in Figure 6.1). It is important to remark that the input clock and the data refresh clock for the “A3G4250D” come from the highest hierarchal VHDL block.

6.1.2 VHDL Component LIS3DSH

Paragraph 5.1.2 described the hardware (ST LIS3DSH) characteristics of the vehicle’s 3-axis linear accelerometer and then defined the operational setting on theoretical assumptions. In order to implement a “Parallel Computing Controller” on an FPGA using VHDL algorithms, a dedicated “VHDL component” named as “LIS3DSH” is created. It means that a dedicated set of FPGA logic gates are programmed to interact only with the peripheral, and one other set of FPGA logic gates are programmed to process and broadcast the sensor’s information.

Most of paragraph 6.1.1 analysis is valid also for the “LIS3DSH” VDHL component, and it is possible to reutilize the sub-component which expresses the SPI interface and physically interacts with the peripheral (“SPI_Master” VHDL component). The “LIS3DSH” component includes as well a second sub-component (it acts as the LIS3DSH’s state machine), which manages the data stream (interfaced to the “SPI_Master Component”), the peripheral’s configuration and, the data digital-processing.

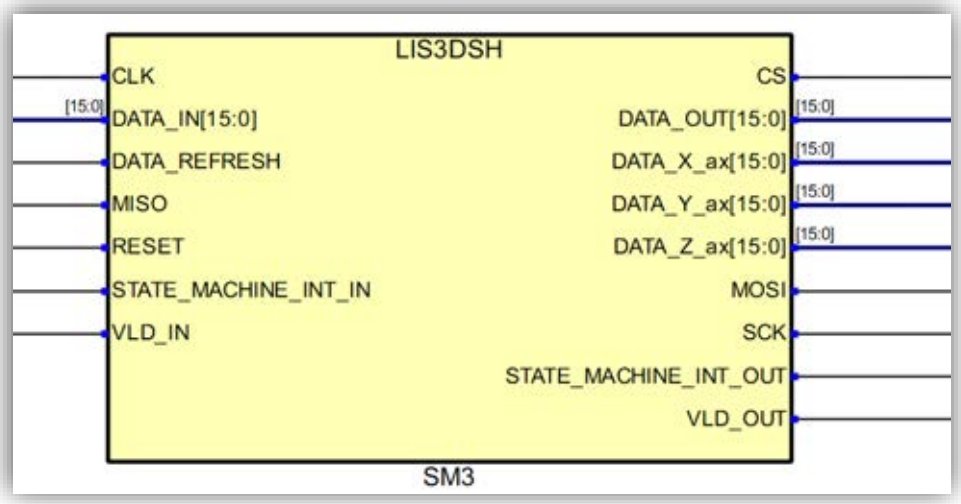


Figure 6.2: RTL view of the “VHDL component” named as “LIS3DSH”.

For data processing of the “LIS3DSH” Electronic compass output registers, it is intended that the sub-component converts each 16-bit value expressed in TWO’s complement format (signed integer) into a 16-bit STD_LOGIC_VECTOR. These operations result in a set of outputs from the “VHDL component”: a 16-bit “DATA_X_ax”, a 16-bit “DATA_Y_ax” and

a 16-bit “DATA_Z_ax” (explicated by “Figure 6.2”). The input clock and the data refresh clock for the “LIS3DSH” come from the highest hierarchal VHDL block.

6.1.3 VHDL Component TESEO

Paragraph 5.1.5 described the vehicle’s navigation monitor hardware characteristics (ST TESEO module and the redundant altimeter) and defined the operational setting on theoretical assumptions. In order to implement a “Parallel Computing Controller” on an FPGA using VHDL algorithms, a dedicated “VHDL component” named as “TESEO” is created. It means that a dedicated set of FPGA logic gates are programmed to interact only with the navigation monitor peripheral, and one other set of FPGA logic gates are programmed to process and broadcast the peripheral’s information.

“VHDL component TESEO” includes three sub-components: “SPI_Master”, “LPS25HB” and “Process_Teseo”. The first one expresses the SPI interface and physically interacts with the GNSS peripheral. The second component manages the redundant altimeter (it acts as the LPS25HB’s state machine and peripheral SPI interface). The last sub-component manages the GNSS data stream (interfaced to the “SPI_Master Component”), the peripheral’s configuration and, the data digital-processing.

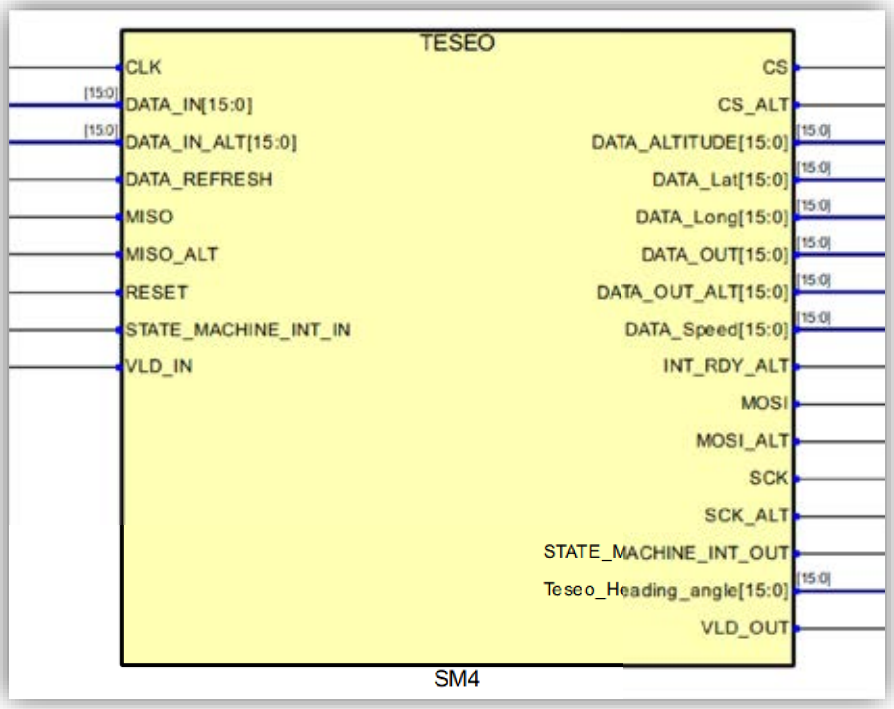


Figure 6.3: external RTL view of the “VHDL component” named as “Teseo”.

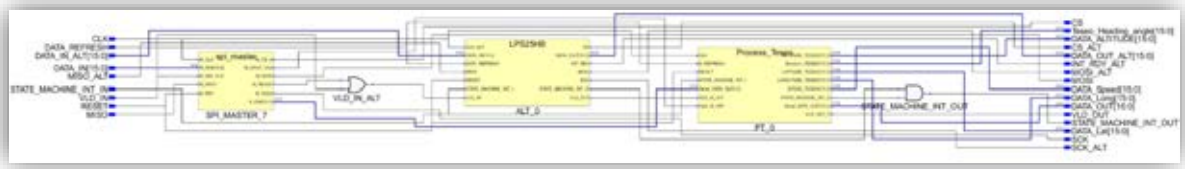


Figure 6.4: internal RTL view of the "VHDL component" named as "Teseo".

The input clock and the data refresh clock for the "VHDL component TESEO" come from the highest hierarchal VHDL block.

6.1.4 VHDL Component, Safety Sensors

The VHDL component "Safety_Sensors" interfaces a set of vehicle's peripherals with the controller's core. The peripherals are:

- BMS;
- short-distance proximity sensor;
- long-distance proximity sensor.

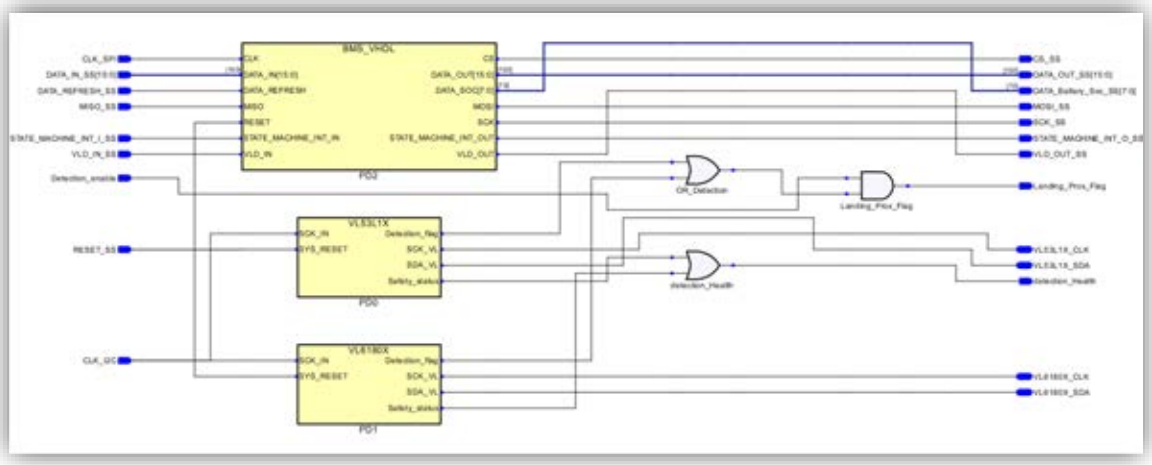


Figure 6.5: internal RTL view of the "VHDL component" named as "Safety_Sensors".

6.1.4.1 BMS_VHDL Component

The "BMS_VHDL component" directly interfaces the controller with the "EBM" (or BMS). The design assumption is that the "BMS_VHDL component" acts like the "SPI_Master" and that the "EBM" acts like the "SPI_Slave". It allows the controller to interrogate the "EBM" and obtain the meaningful data associated with the battery module. The obtained information is digital processed and then broadcast to the controller's core.

6.1.4.2 Proximity Sensor Components

The design assumption made is: as soon as the ground will be detected, the controller will disable the powertrain. Due to the importance of the information, the Author's proposal perseveres a redundant solution.

Both ST VL53L1X plug-in board and VL6180X plug-in board are interfaced with an I^2C protocol. VHDL Proposed design targets the implementation of two independent "I^2C Data Bus", each one dedicated to a single plug-in board (it is mandatory for a full redundant scheme). Since that the goal is to implement a parallel and redundant functionality, the VHDL component "Safety Sensor" code population highlight three different VHDL blocks allocated for the ground detection:

- VHDL component "VL53L1X";
- VHDL component "VL6180X";
- proximity sensor's logic gates.

"VL53L1X" (as well for the "VL6180X") VHDL sub-component firstly configures the peripheral and then reads the output register of the device (according to the peripheral's supplier application note). If one detection is observed, the sub-component flags the detection through an active-high digital signal: "Detection_Flag".

The "proximity sensor's logic gates" are associated with logic gates digital processing structure, which receives three input signals ("Enable_detection", "VL53L1X_detection" and "VL6180X_detection") and generates one output signal, the "Landing_Prox_Flag" (which will allow disabling the propellers). Follows the truth table of the "proximity sensor's logic gates".

Enable_detection	VL53L1X_detection	VL6180X_detection	Landing_Prox_Flag
0	**0**	**0**	**0**
0	**0**	**1**	**0**
0	**1**	**0**	**0**
0	**1**	**1**	**0**
1	**0**	**0**	**0**
1	**0**	**1**	**1**
1	**1**	**0**	**1**
1	**1**	**1**	**1**

Table 6.1:"Prox_Sensor_logic_gates" truth table.

6.1.5 VHDL Component, Flight Parameters EEPROM

A system's operation precondition is the definition of the flight route and the sensor's[109] calibration. This procedure results in a user uploading the flight parameters into the memory storage before the vehicle's take-off manoeuvre.

"Flight Parameters EEPROM VHDL component", managing a dedicated SPI interface, reads the parameters stored and accordingly broadcast the valuable information to the controller. The consequent digital acceleration process achieved by the "Flight Controller" will translate these parameters into a flight route. The mandatory parameters to upload are:

- final destination geographical coordinates;
- sensor's calibration values;
- target flight altitude.

6.1.6 Controller's core inputs, summary

According to the assumptions made, the controller's core requires the following input parameters:

a) altitude;
b) speed;
c) pitch angle;
d) rolling angle;
e) yaw angle;
f) estimated position;
g) flight reference parameters;
h) proximity sensor;
i) battery's SoC.

6.2 Controller's Outputs

Paragraph 4.2 study explores the technical requirements for the "System's Controller", defining a set of "controller's inputs" and "controller's outputs". Each controller's output is expressed in a digital form and requires the conversion into a specific actuator's control signal.

[109] Such as the calibration of the redundant altimeter.

For each “actuator” or “motor” described in “Chapter 5” shall be defined a specific “VHDL component” with an appropriate algorithm able to configure and adequately control the electromechanical output device.

6.2.1 VHDL Component SERVO

Paragraph 5.1.8 described the hardware characteristics of the “SERVO Motors” (Hitec HS-65MG and HS-125MG) and defined the operational setting on theoretical assumptions. In order to control such servo motors using FPGA and VHDL algorithms, a dedicated “VHDL component” named “PWM_SERVO” is created. It means that a dedicated set of FPGA logic gates are programmed to convert a 7-bit digital value into a PWM signal with a frequency of 50Hz (refresh time of 20ms) and a T_{ON} time between 1500µs and 1900µs as previously described in paragraph 5.1.8.

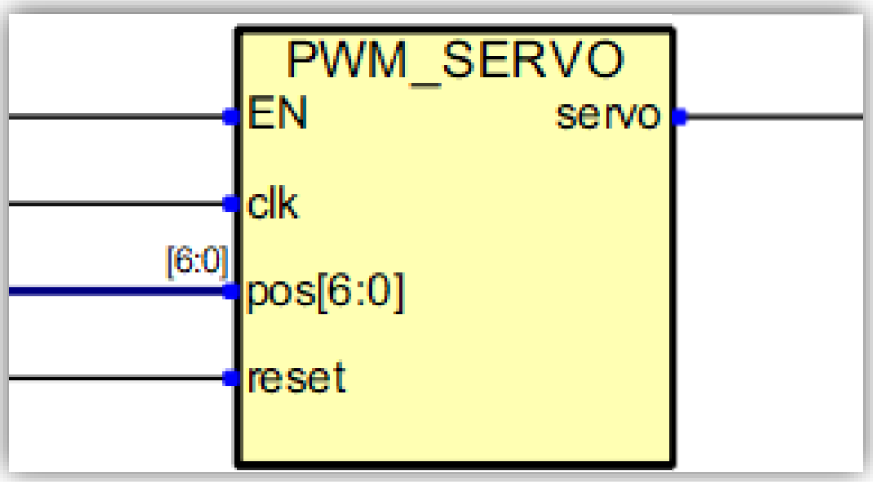


Figure 6.6: external RTL view of the “VHDL component” named as “PWM_SERVO”.

6.2.2 Powertrain’s VHDL Components

Paragraph 5.1.10 described the E-Motor drivers’ hardware characteristics (ST EVAL BRD) and defined the operational setting on theoretical assumptions. In order to control such motors using FPGA and VHDL algorithms, a dedicated “VHDL component” named “MOTOR_INT” is created. Performing this operation a dedicated set of FPGA logic gates are programmed to convert 8-bit digital value information into an RS232 command which defines the motor’s torque demand[110], as previously described in paragraph 5.1.10.

[110] It is a static parameter not available to the vehicle’s controller. It is set by the user together with all the E-Motor configuration parameters on the motor driver.

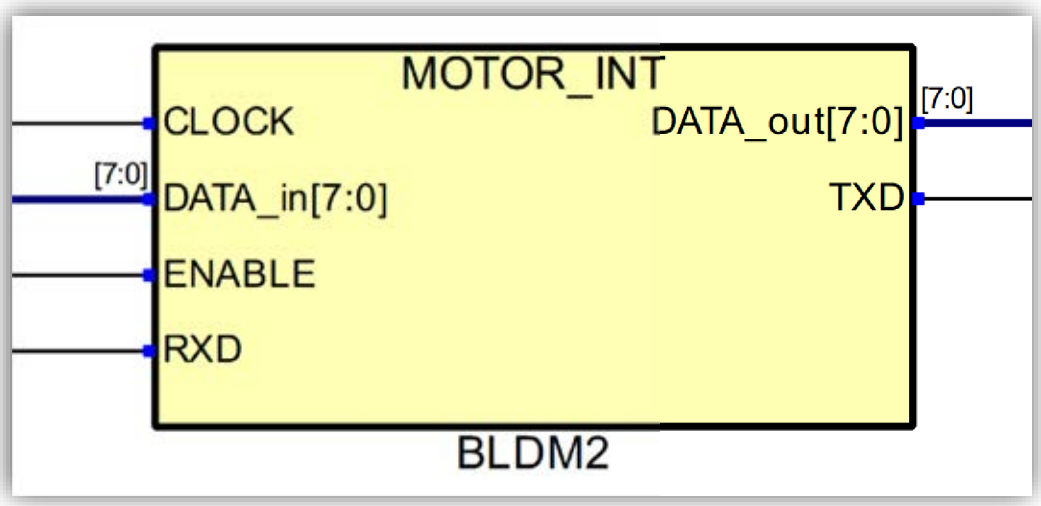


Figure 6.7: external RTL view of the "VHDL component" named as "MOTOR_INT".

The VHDL component "MOTOR_INT" is used for both E-Motors. The vehicle's left E-Motor is associated with the VHDL component instance "BLDM1", and the vehicle's right E-Motor is associated with the VHDL component instance "BLDM2" (as for "Figure 6.7").

6.2.3 *VHDL Component, Flight Telemetry EEPROM*

A learning/training process assumes paramount importance for the technical proposal. This procedure requires the capture of the vehicle's flight parameters while a human-being pilot controls the vehicle; the consequent action is the memorisation of the captured information into data memory storage. As well as, this operation may be taken while the vehicle is flying in "full autonomous mode"; although the data may not be optimal for learning/training purposes, it may be beneficial for data analysis aims.

"Flight Telemetry EEPROM" VHDL component, managing a dedicated SPI interface, reads a wide range of system's parameters and then accordingly writes the physical memory's registers. At the flight's end, as soon as the system is disengaged, the user can access the EEPROM via the dedicated port. The presented feature allows the user to download the captured data.

6.2.4 *Controller's core outputs, summary*

By assumptions, the neuro-fuzzy flight controller generates the following output signals:

- left E-Motor torque demand, (RS232 communication interface);
- right E-Motor torque demand, (RS232 communication interface);
- ailerons SERVO-Motor, PWM control signal;

- elevator SERVO-Motor, PWM control signal;
- rudders SERVO-Motor, PWM control signal.

6.3 Fuzzy Logic Controller Design

From “paragraph 4.2” theoretical prospects, the controller’s heart is the “FIS” (or Rules Block) which utilizes a set of 5 “Rulebases” functions. Each “Rulebase” accordingly uses sets of MIFs crisps values to activate sets of weighted MOFs. [43]

XFUZZY GUI encases a set of membership functions into a “Type”, which is usually associated with a sensor (MIF association) or with an actuator (MOF association). GUI’s definition of the controller physical “Input Variables” and “Outputs Variables” requires an existing environment “Type” to be linked with it.

Patently, **the creation of a fuzzy system in the “XFUZZY” (or “XFL3 GUI”) environment usually starts with the definition of the “Operator Set”**. An “Operator Set” in “XFL3 GUI” is an object containing the mathematical functions assigned to each fuzzy operator. Fuzzy operators can be binary (like the T-norms and S-norms employed to represent linguistic variable connections, implication, or rule aggregations), unary (like the C-norms or the operators related with linguistic hedges), or can be associated with “defuzzification methods”. [38]

The second step, in the description of a fuzzy system, is the creation of the linguistic variable types, using the “Type Creation Interface”. A new “Type” necessitates the introduction of its identifier and universe of discourse (minimum, maximum and cardinality). The interface includes several predefined types corresponding to the most usual partitions of the universes. These predefined types contain homogeneous triangular, trapezoidal, bell-shaped and singleton partitions, shouldered-triangular and shouldered-bell partitions. Other predefined types are equal bells and singletons, which may be the first option for output variable types. When one of the aforementioned predefined types is selected, the number of membership function of the partition must be introduced. [38]

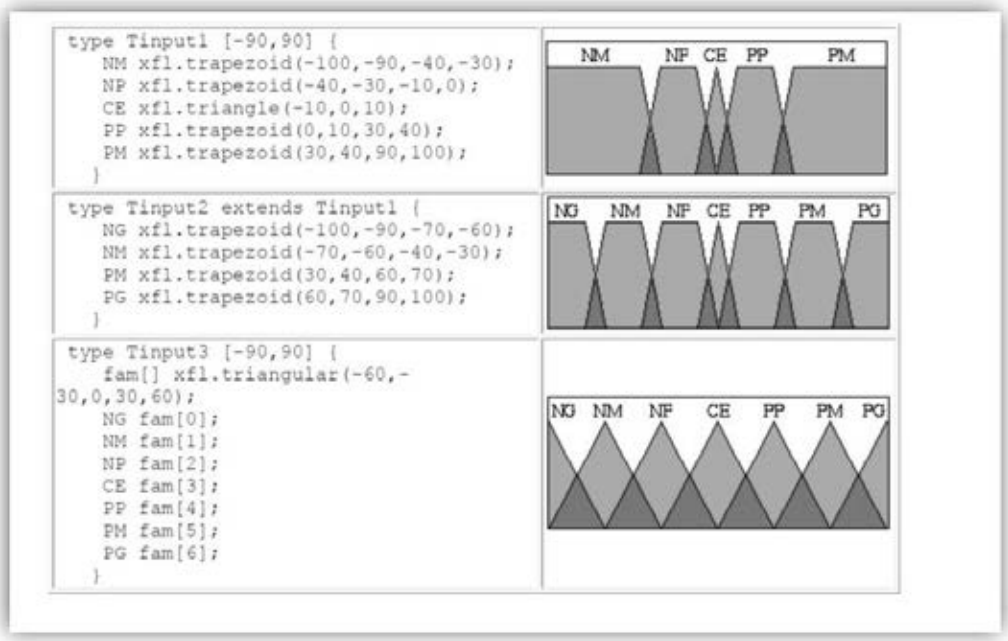


Figure 6.8: XFUZZY environment examples of types. [38]

An XFL3 type is an object that describes a type of linguistic variable. It means to define its universe of discourse, to name the linguistic labels covering that universe, and to specify the membership function associated with each label. Linguistic labels can be defined in two modes: free membership functions or members of a family of membership functions. In the last one, the family of membership functions must be defined in advance. A free membership function uses its own set of parameters while the members of a family share the list of parameters of that family. It results helpful to reduce the number of parameters and representing constraints between the linguistic labels (such as the order or a fixed overlapping degree). [38]

The types so defined inherit the universe of discourse and the labels of their parents automatically. The labels defined in the type's body are either added to the parent labels or overwrote if they have the same name.

The third step in defining a fuzzy system is the description of each "Rulebase", expressing the relationship among the system variables. A dedicated GUI's interface[111]

[111] The contents of the rules can be displayed in three formats: free, tabular, and matricial. The free format uses three fields for each rule. The first one contains the confidence weight. The second field shows the antecedent of the rule. It is an auto-editable field, where changes can be made by selecting the term to modify (a "?" symbol means a blank term) and by using the buttons of the window. The third field of each rule contains the consequent description. It is also an auto-editable field that can be modified by clicking the "->" button. New rules can be generated by introducing values on the last row (marked with the "*" symbol).
The button bar at the bottom of the free form allows to create conjunction terms ("&" button), disjunction terms ("|" button), modified terms with the linguistic hedges not ("!" button), more or less ("~" button), slightly ("%" button), and strongly ("+"

could be used to define the “Rulebase” (the lists of input and output variables parameters), and to structure the functions with the appropriate operator sets and the confidence weight to be used.

The accurate definition of the “Operator Sets”, “Variable Types”, and “Rulebases” is propaedeutic for the fuzzy system design progress. The definition of the global input and output variables, using the “**Variable Properties**”[112] interface (GUI’s window), is the following design step.

The concluding operation in a fuzzy system definition is the description of its hierarchical structure.

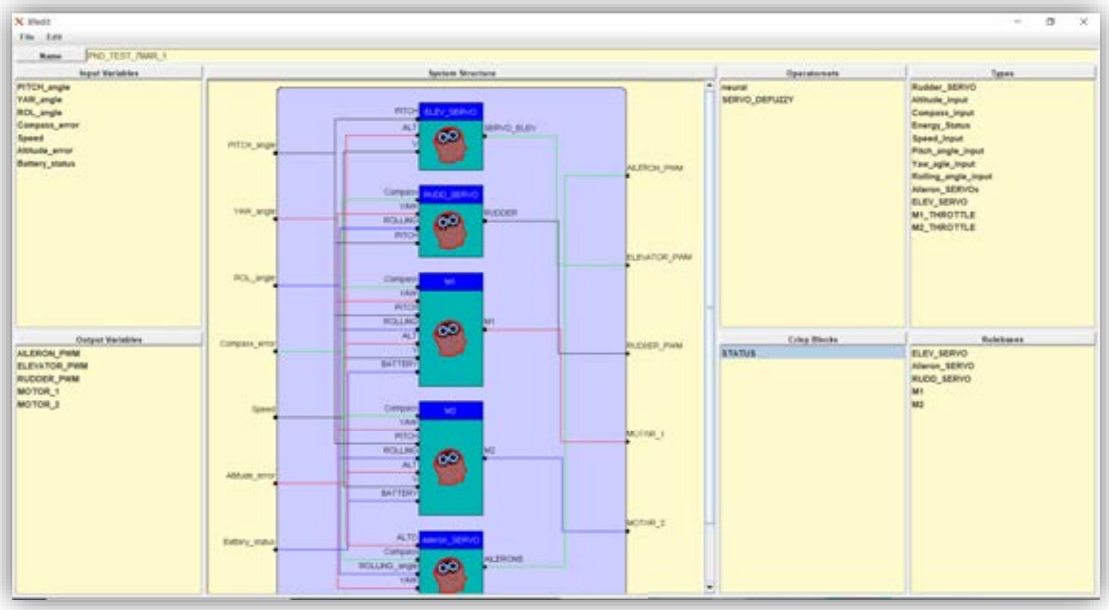


Figure 6.9: flight controller’s description in XFL3 GUI.

6.3.1 Type “Rudder_SERVO”, Membership Output Function

The Type “Rudder_SERVO” contains five MOFs, each of them associated with a specific rudder manoeuvre that a “Human being Pilot” may perform in front of external conditions while a predefined route is followed. Membership functions are:

a) Left_turn;

button), and single terms relating a variable and a label with the clauses equal to (“==”), not equal to (“!=”), greater than (“>”), smaller than (“<”), greater or equal to (“>=”), smaller or equal to (“<=”), approximately equal to (“~=”), strongly equal to (“+=”), and slightly equal to (“%=”). The “->” button is used to add a rule conclusion. The ">..<" button is used to remove a conjunction or disjunction term (e.g. a term “v == l & ?” is transformed into “v == l”). The free form allows the user to describe more complex relationships among the variables than the other forms. [32, 37 and 38]

[112] The information required to create a variable is its name and type.

b) Left_wind_comp;

c) Cruise_rudder;

d) Right_wind_comp;

e) Right_turn.

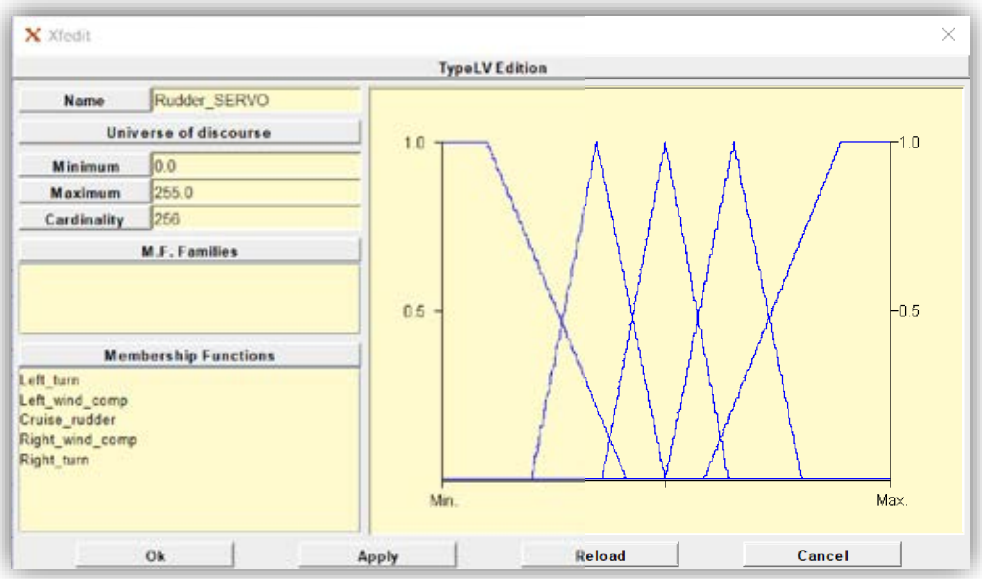


Figure 6.10: Type "Rudder_SERVO", graphical representation.

The MOFs design strategy is based on the rudder's mechanical behaviour steered by the electromechanical actuator, which is the SERVO-Motor described in paragraph 5.1.8. Moving from the disserted hardware description of the "Rudder Servo", the membership function "Cruise_rudder" is a conventional triangular membership function centred to the mechanical rudder angle equal to 0°, which corresponds to the +20° of the rudder's SERVO-Motor electrical angle. Depending on the SERVO-Motor control signal resolution utilised, it could be 7-bit or 8-bit (0 to 127 steps of 0 to 255 steps), the 0° rudder mechanical angle corresponds to an output value of 127 for the case of 8-bit resolution (or 63 for the case of 7-bit resolution).

The proposed neuro-fuzzy controller utilises 8-bit resolution data, and then a VHDL component will perform the information's digital acceleration, which will be converter in a PWM signal with a 7-bit resolution.

"Right_turn" is a conventional trapezoidal membership function associated with a heavy rudder manoeuvre which results in a significant positive "Yaw angle" ("Left_turn" is the mirror/symmetrical function, associated with a heavy rudder manoeuvre which results into a significant negative "Yaw angle"). The resulting output control signal for the rudder's

SERVO-Motor theoretically may set the electrical angle in a range between 24° and 40° (between 0° and 16° for the “Left_turn” function), which corresponds to a mechanical angle between +4° and +20° (between -4° and -20° for the “Left_turn” function).

“Right_wind_comp” is a conventional triangular membership function associated with a slight rudder manoeuvre which results in a moderate positive “Yaw angle” (“Left_wind_comp” is the mirror/symmetrical function, associated with a slight rudder manoeuvre which results in a moderate negative Yaw angle). Resulting output control signal for the rudder’s SERVO-Motor, theoretically may set the SERVO-Motor electrical angle in a range between 20° and 28° (between 12° and 20° for the “Left_wind_comp” function), which corresponds to a mechanical angle between +0° and +8° (between 0° and -8° for the “Left_wind_comp” function). This function results particularly beneficial when the vehicle is flying through moderate turbulence or gusty winds. These conditions, both, can cause yaw through weather vaning and result in an undesirable vehicle rolling. “Left_wind_comp” function is intended to compensate undesirable yaw to the right caused by weather conditions, while “Right_wind_comp” function is intended to compensate undesirable yaw to the left caused by weather conditions.

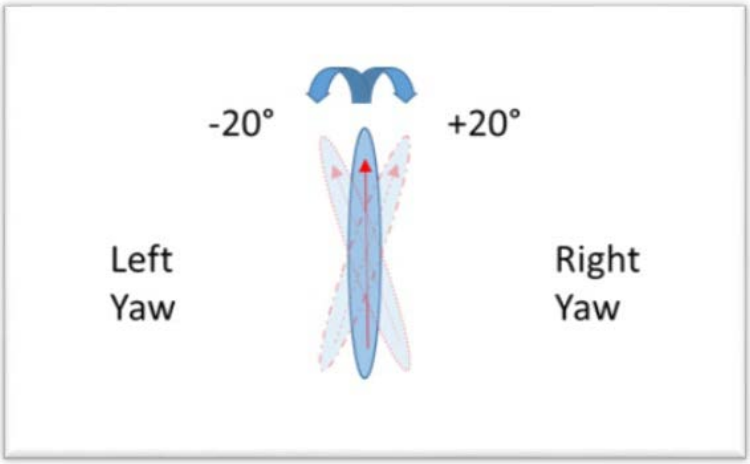


Figure 6.11: representation of the rudder working principle.

6.3.2 Type “Altitude_input”, Membership Input Function

The Type “Altitude_input” contains four MIFs, each of them associated with a specific “Human being Pilot” interpretation of the altitude indicator while a predefined route is followed. Membership functions are:

a) Low_ALTD;
b) Cruise_ALTD;
c) Soft_high;

d) OVER_ALTD.

The MIFs design strategy is based on the interpretation of the "Altitude_Error" relative error input value[113], defined by:

$$Altitude\_Error = \frac{(Reference\ Altitude\ value) - (Actual\ Altitude\ value)}{\left(\frac{(Reference\ Altitude\ value) + (Actual\ Altitude\ value)}{2}\right)}$$

(Equation 39)

By assuming the input signal resolution is set to 8-bit (0 to 255 steps), the centre value is associated with an Altitude_error = 0 corresponds to an input value of 127.

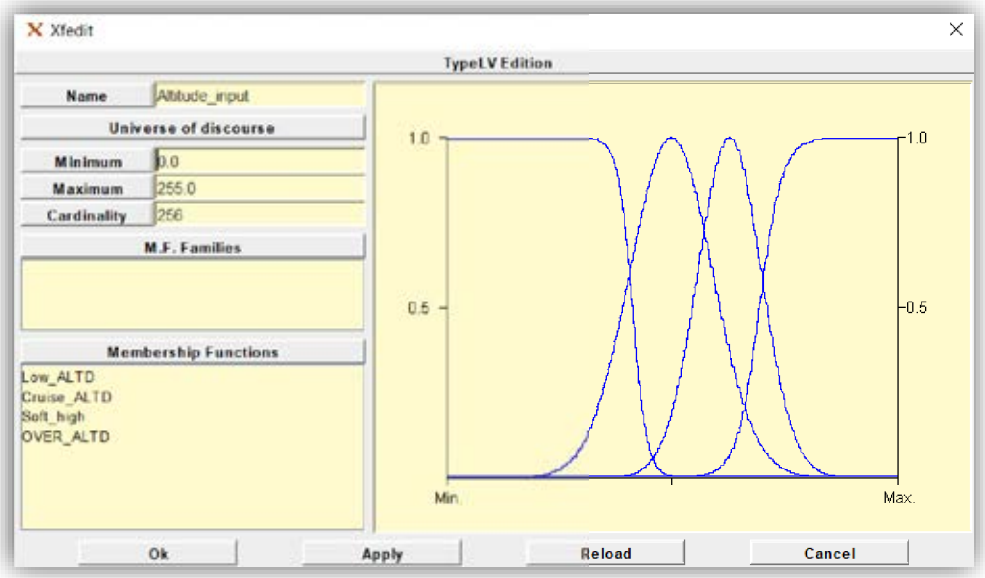


Figure 6.12: Type "Altitude_input", graphical representation.

"Low_ALTD" membership function is associated with a "Human Pilot" perception of an altitude below the reference route altitude value while he is interpreting the sensor's feedback and the other environment variables.

"Cruise_ALTD" membership function is associated with a "Human Pilot" perception of flying at the correct altitude, while the "Soft_high" membership function describes a "Human Pilot" behavioural, which uses the altitude as energy storage. Seeking the altitude as energy storage may be a common manoeuvre when a tail-wind increases the vehicle speed or even favours the vehicle to act as a glider.

[113] "Actual Altitude Value" is the value read by the sensor while the "Estimated Altitude Value" is the value required by the programmed cruise (value read from EEPROM).

"OVER_ALTD" membership function is associated with a "Human Pilot" perception of an undesirable altitude above the reference route altitude value while he is interpreting the sensor's feedback and the other environment variables.

6.3.3 Type "Compass_input", Membership Input Function

The Type "Compass_input" contains three MIFs, each of them associated with a specific "Human Pilot" interpretation of the electronic compass while a route shall be followed. Membership functions are:

a) Negatitive_angle_err;

b) Cruise;

c) Positive_agnle_err.

The MIFs design strategy is based on the interpretation of the "Compass_Error" differential input value, which is:

$$Compass\_Error = (requested\ Electr.Compass\ Value) - (Actual\ Electr.Compass\ Value)$$

(Equation 40)

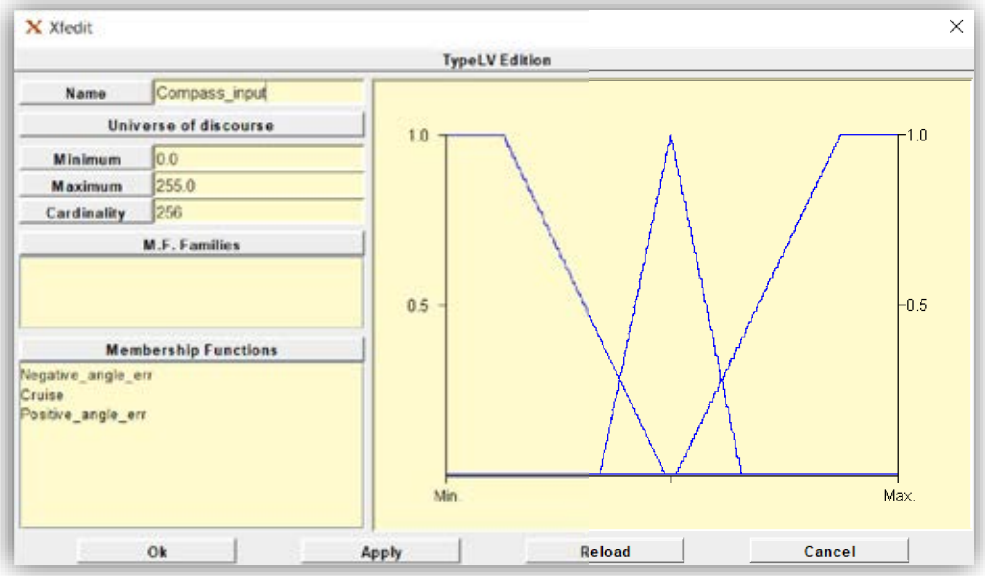


Figure 6.13: Type "Compass_input", graphical representation.

The information "Compass_error" is the outcome, according to "Equation 40", of a digital process implemented by a bespoke VHDL algorithm, which utilizes few sensors[114] to

[114] Accelerometer output data, Gyroscope output data and the most critical TESEO GPS module parameters.

obtain the optimal vehicle's direction ($requested\ Electr.Compass\ Value$), and the current vehicle's heading angle ($Actual\ Electr.Compass\ Value$).

"Negatitive_angle_err" membership function is associated with the "Human being Pilot" perception of a plane drifting to the left from the correct direction with the result of pointing off-target (pointing to the left of the waypoint or final destination). As for the case of "Figure 6.14".

Figure 6.14: negative navigation angle error, graphical representation.

"Cruise" membership function is associated with a "Human Pilot" perception of a vehicle that follows the correct route and flies towards the waypoint (or the final destination). As for the case of "Figure 6.15".

Figure 6.15: cruise navigation, graphical representation.

"Positive_agnle_err" membership function is associated with the "Human Pilot" perception of a plane drifting to the right from the correct direction with the result of pointing off-target (pointing to the right of the waypoint or final destination, according to the illustration of "Figure 6.16").

Figure 6.16: positive navigation angle error, graphical representation.

6.3.4 Type "Energy_Status", Membership Input Function

The Type "Energy_Status" contains four MIFs, designed to influence[115] the throttle management of the vehicle's powertrain. The outcome is that manoeuvres may be affected by the Battery SoC to a certain degree[116], depending on the environmental conditions. Membership functions are:

a) Derating;

b) Low_SOC;

c) Healthy_SOC;

d) Overcharged.

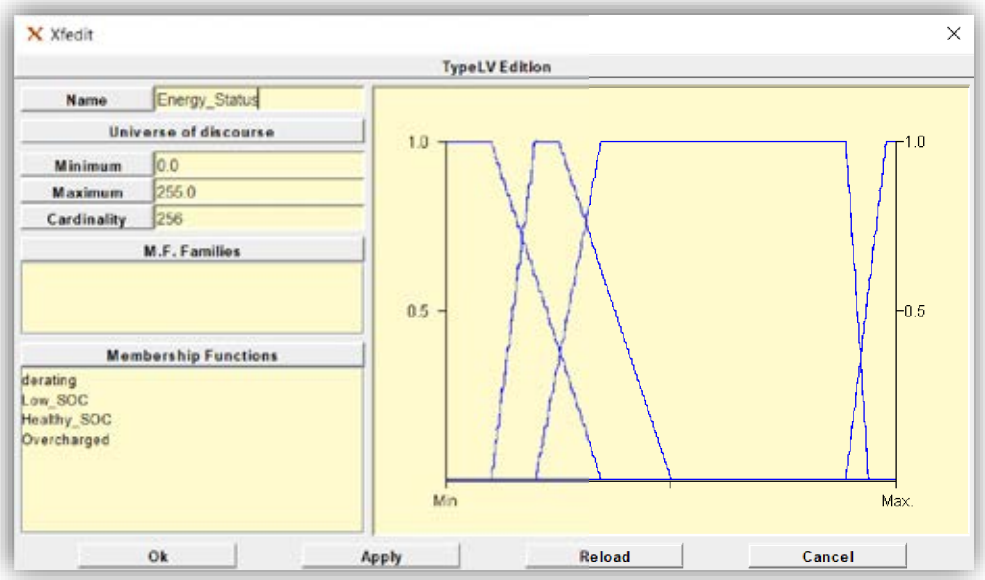


Figure 6.17: Type "Energy_Status", graphical representation.

Assuming that the input signal represents the "Battery State of Charge" (SoC) is an 8-bit signal, where the value "0" (or "00" in hex format) represents a full discharged battery (SoC equal to 0%), and the value of "255" (or "FF" in hex format) represents a full charged battery (SoC equal to 100%).

The membership function "derating" wants to influence the throttle control in order to prevent the deterioration of the battery (battery protections) during shallow SoC operations.

---

[115] Concur to the neuro-fuzzy logic controller energy-saving future innovations.

[116] The environmental variables and the navigation (such as altitude and direction) have a primary influence for the throttle control. The influence of the "Battery SoC" has a secondary influence for the throttle control.

The membership function “Low_SOC” is associated with the “Human being Pilot” conservative throttle control that prioritizes the power economy on the system performances during low SoC operations. It may result easily associable to a “Human being Pilot” that in front of a low fuel level indication changes his driving style to reach his destination.

The membership function “Healthy_SOC” has a negligible influence on the throttle control because this function is not associated with any safety limitation. It means that the weight of this membership function is overwhelmed by the weight of the other variables.

The membership function “Overcharged” is associated with the case of a battery fully charged[117]. It wants to influence the throttle control, boosting the power output. It results valuable during the “Take OFF” manoeuvre.

6.3.5 Type “Speed_Input”, Membership Input Function

The Type “Speed_Input” contains five MIFs, each of them associated with a specific “Human being Pilot” interpretation of the speed indicator while a predefined route is followed.

Membership functions are:

a) Stall_speed;
b) Low_speed;
c) Cruise_Speed;
d) High_Speed;
e) Excessive_speed.

The membership function “Stall_speed” is associated with the “Human being Pilot” interpretation of the speed sensor, resulting in a perception of a too low speed that may result in a stall.

The membership function “Low_speed” is associated with the “Human being Pilot” perception of a low speed that shall be addressed before it may degenerate into a vehicle’s hazardous condition.

The membership function “Cruise_Speed” is associated with the “Human being Pilot” perception of correct vehicle speed.

[117] Depending on the technology (Chemistry) of the battery (cells) may be deteriorative to keep the battery charged above 95%.

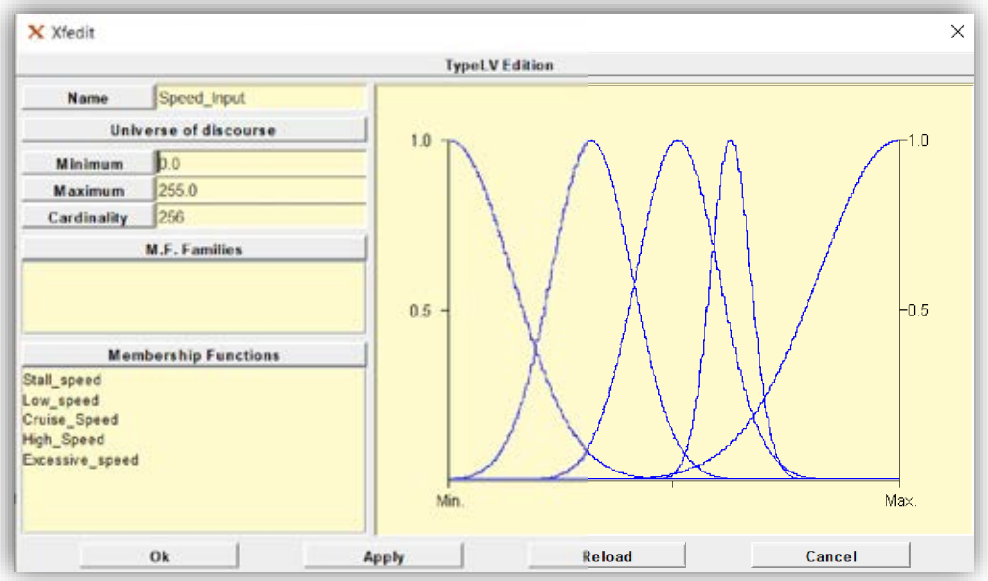


Figure 6.18: Type "Speed_Input", graphical representation.

The membership function "High_Speed" is associated with the "Human being Pilot" perception of a higher speed that may be caused by tail-winds or an excessive throttle. The function's goal is to increase the vehicle's safety (minimising the risks of reaching excessive speeds) and reduce energy consumptions (extend the range, using, for example, the altitude as energy storage).

The membership function "Excessive_Speed" is associated with the "Human being Pilot" perception of a very high speed that may be dangerous for the vehicle's safety. The membership function's goal is to allow a quick correction of the speed parameter, relying not merely on the throttle control but also on the elevator for a variation of the pitch angle.

6.3.6 *Type "Pitch_angle_Input", Membership Input Function*

The Type "Pitch_angle_input" contains three MIFs, each of them associated with a specific "Human Pilot" interpretation of the vehicle's gyroscope (and of the 3-axis vehicle's accelerometer) while he is following a pre-defined route. Membership functions are:

a) descending_pitch;

b) cruise_pitch;

c) ascending_pitch.

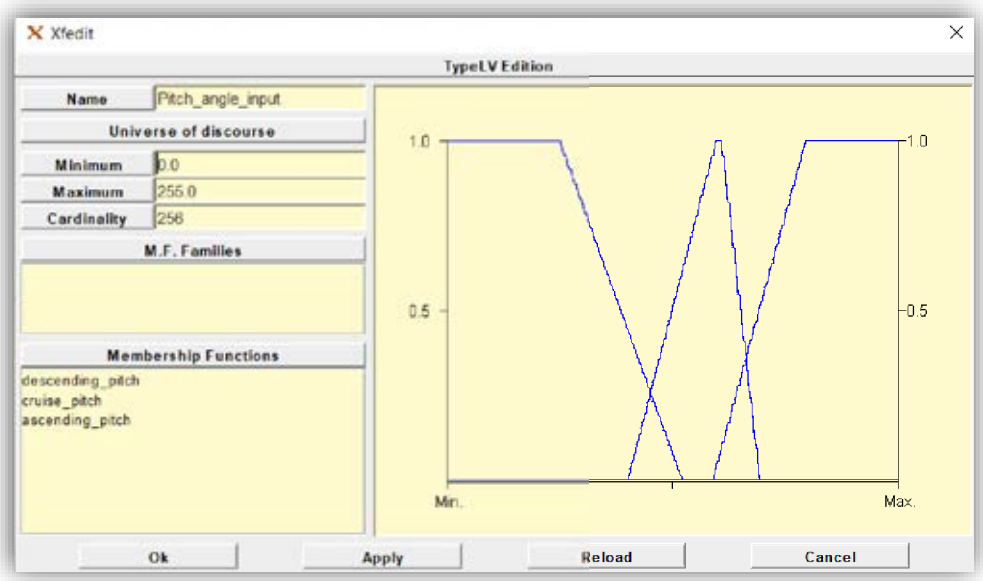


Figure 6.19: Type "Pitch_angle_input", graphical representation.

Figure 6.20: example of an aerial vehicle's pitch indicator instrument.

The membership function “descending_pitch” is associated with the “Human being Pilot” interpretation of the vehicle’s pitch attack angle, which results in the perception of a manoeuvre causing an altitude loss.

The membership function “cruise_pitch” is associated with the “Human being Pilot” perception of a correct pitch attack angle and that no corrective manoeuvre shall be implemented.

The membership function “ascending_pitch” is associated with the “Human being Pilot” interpretation of the vehicle’s pitch attack angle, which results in the perception of a manoeuvre causing an altitude increase.

6.3.7 Type “Yaw_angle_Input”, Membership Input Function

The Type “Yaw_angle_input” contains three MIFs, each of them associated with a specific “Human Pilot” interpretation of the vehicle’s instrumentation (in this case, the 3-axis vehicle’s accelerometer) while he is seeking for an identified route. Membership functions are:

a) Yaw_left;

b) Yaw_stable;

c) Yaw_right.

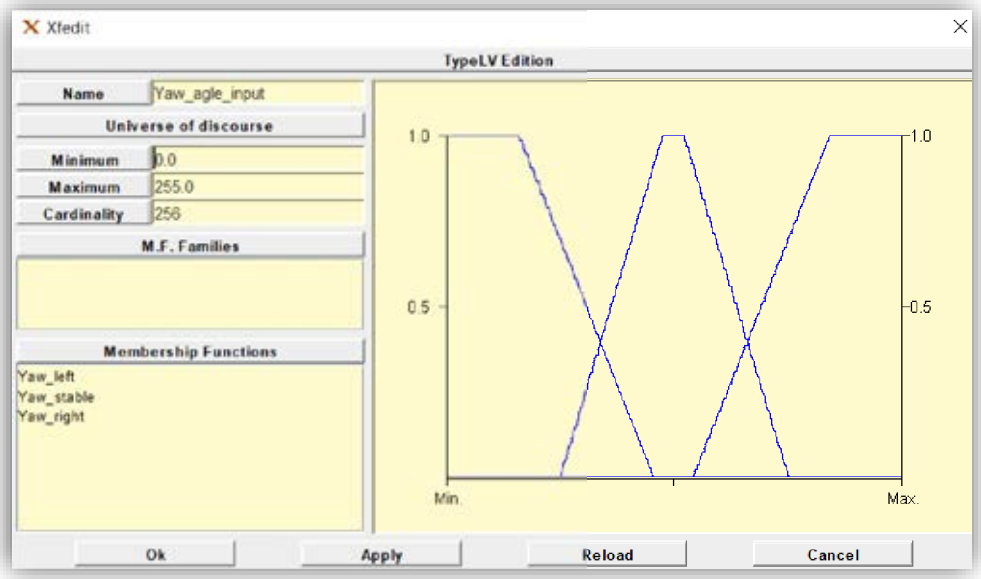


Figure 6.21: Type “Yaw_angle_input”, graphical representation.

The membership function “Yaw_left” is associated with the “Human being Pilot” perception of a vehicle’s yaw to the left.

The membership function “Yaw_stable” is associated with the “Human Pilot” awareness of negligible yaw influence on the vehicle’s flight (no additional corrective manoeuvre required to compensate yaw forces acting on the vehicle). The membership function “Yaw_right” is associated with the “Human being Pilot” perception of a vehicle’s yaw to the right.

6.3.8 Type “Rolling_angle_Input”, Membership Input Function

The Type “Rolling_angle_input” contains five MIFs, each of them associated with a specific “Human Pilot” interpretation of the gyroscope while a route shall be followed.

Membership input functions are:

a) Hard_Rolling_left_angle;

b) Soft_Rolling_left_angle;

c) No_Rolling;

d) Soft_Rolling_right_angle;

e) Hard_Rolling_right_angle.

The membership function “Hard_Rolling_left_angle” is associated with the “Human being Pilot” interpretation of the vehicle’s rolling angle, which results in the perception of a manoeuvre causing a heavy rolling to the left.

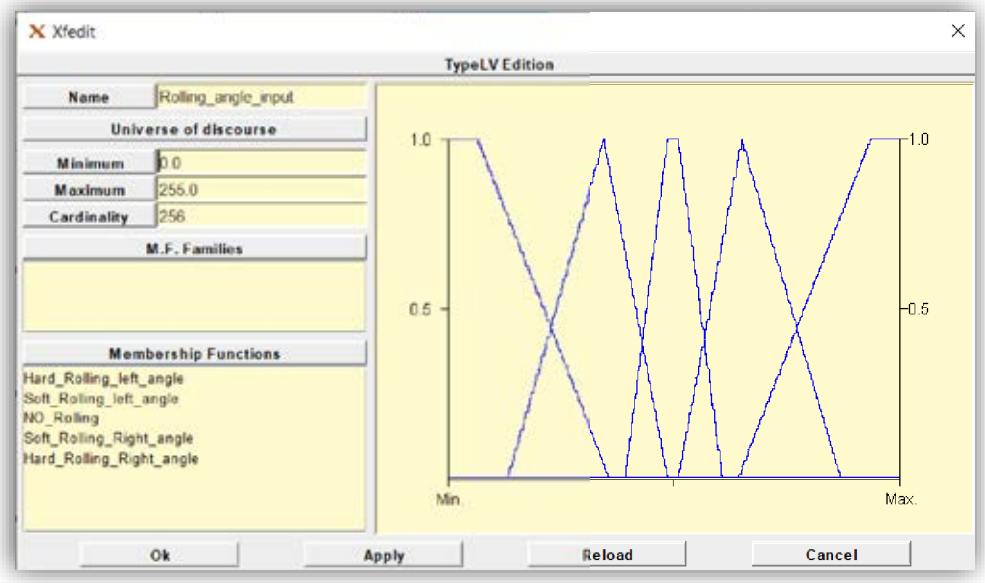


Figure 6.22: : Type “Rolling_angle_input”, graphical representation.

The membership function “Soft_Rolling_left_angle” is associated with the “Human being Pilot” interpretation of the vehicle’s rolling angle, which results in the perception of a manoeuvre causing a light rolling to the left.

The membership function “No_Rolling” is associated with the “Human being Pilot” perception of a not rolling vehicle.

The membership function “Soft_Rolling_right_angle” is associated with the “Human being Pilot” interpretation of the vehicle’s rolling angle, which results in the perception of a manoeuvre causing a light rolling to the right.

The membership function “Hard_Rolling_right_angle” is associated with the “Human being Pilot” interpretation of the vehicle’s rolling angle, which results in the perception of a manoeuvre causing a heavy rolling to the right.

6.3.9 Type “Aileron_SERVOs”, Membership Output Function

The Type “Aileron_SERVOs” contains five MOFs, each of them associated with a specific aileron based manoeuvre that a “Human being Pilot” may perform in front of external conditions while a predefined route is followed. Membership functions are:

a) Turn_left_rol;
b) Left_wind_Rol_comp;
c) Cruise_Rolling;
d) right_wind_Rol_comp;
e) Turn_right_rol.

The MOFs design strategy is based on the mechanical behaviour of the “ailerons” steered by the electromechanical actuators, which are the SERVO-Motors described in paragraph 5.1.8. Moving from the disserted hardware description of the ailerons SERVO-Motors, the membership function “Cruise_rolling” is a conventional trapezoidal membership function centred to the ailerons mechanical angle equal to 0°, which corresponds to the +20° of the servo electrical angle. Depending by actuator’s control signal resolution utilised could be of 7-bit or 8-bit (0 to 127 steps or 0 to 255 steps), the 0° ailerons mechanical angle corresponds to an output value of 127 for the case of 8-bit resolution (or 63 for the case of 7-bit resolution). The proposed neuro-fuzzy controller utilises 8-bit resolution data, and then a VHDL component will perform a digital acceleration of the information, which will be converter in a PWM signal with a 7-bit resolution.

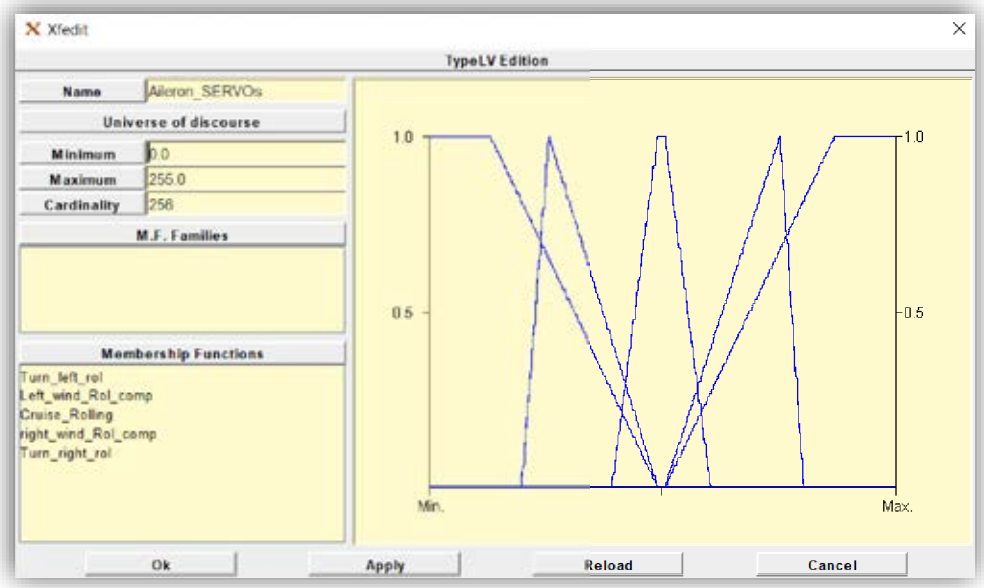


Figure 6.23: Type "Aileron_SERVOs", graphical representation.

"TURN_left_rol" is a conventional trapezoidal membership function associated with a heavy ailerons manoeuvre which results in a significant positive rolling angle ("TURN_right_rol" is the mirror/symmetrical function; it is associated to a heavy ailerons manoeuvre which results into a significant negative rolling angle). The resulting output control signals for the two ailerons' actuators are complementary. Theoretically, it may set the left aileron SERVO-Motor electrical angle in a range between 0° and 11°, while the right aileron SERVO-Motor electrical angle will be set to the complementary value between 40° and 29° (for the TURN_right_Rudder function, the left ailerons SERVO-Motor electrical angle will be set between 40° and 29°, while the right SERVO-Motor angle will be set to the complementary value between 0° and 11°). The corresponding mechanical angle will be between +20° and +9° for the left aileron or between -20° and -9° for the right aileron mechanical angle (for the TURN_right_Rudder function, the corresponding mechanical angle will be between -20° and -9° for the left aileron or between +20° and +9° for the right aileron mechanical angle).

"Left_wind_Rol_comp" is a conventional triangular Membership Function associated with a light ailerons manoeuvre which results in a moderate positive rolling angle ("right_wind_Rol_comp" is the mirror/symmetrical function, associated with a light Ailerons manoeuvre which results in a moderate negative rolling angle). The resulting complementary output control signals for the two Ailerons actuators theoretically may set the left aileron SERVO-Motor electrical angle in a range between 11° and 20°. In contrast, the right aileron

SERVO-Motor electrical angle will be set to the complementary value between 20° and 29° (for the right_wind_Rol_comp function, the left ailerons servo electrical angle will be set between 20° and 29°, while the right SERVO-Motor electrical angle will be set to the complementary value between 11° and 20°). The corresponding mechanical angle will be between +0° and +9° for the left aileron or between -0° and -9° for the right aileron mechanical angle (for the right_wind_Rol_comp function, the corresponding mechanical angle will be between -0° and -9° for the left aileron or between +0° and +9° for the right aileron mechanical angle).

6.3.10 Type "ELEV_SERVO", Membership Output Function

The Type "ELEV_SERVO" contains four MOFs, each of them associated with a specific elevator manoeuvre that a "Human being Pilot" may perform in front of external conditions while a predefined route is followed. Membership functions are:

a) STALL
b) Descending
c) Cruise
d) Climb

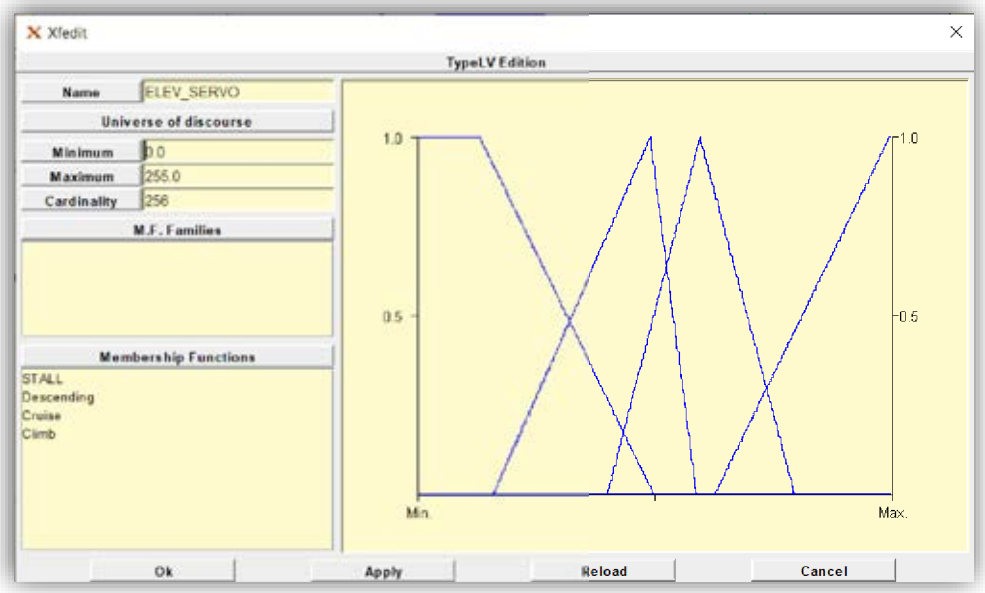


Figure 6.24: Type "ELEV_SERVO", graphical representation.

The MOFs design strategy is based on the mechanical behaviour of the "elevator" steered by an electromechanical actuator, which is the SERVO-Motor described in paragraph 5.1.8. The proposed neuro-fuzzy controller utilizes 8-bit resolution data, and then a VHDL

component will perform a digital acceleration of the information, which will be converter in a PWM signal with a 7-bit resolution.

Moving from the disserted hardware description of the elevator's SERVO-Motor, the membership function "STALL" is a conventional trapezoidal membership function associated with a large elevator's manoeuvre, which may cause an outsized negative pitch attack angle. The resulting output control signal for the elevator's actuator theoretically may set the SERVO-Motor electrical angle in a range between 0° and 16°, which corresponds to a mechanical angle between -20° and -4°.

"Descending" is a conventional triangular membership function associated with a moderate elevator's manoeuvre, which may cause a moderate negative or neutral pitch attack angle. The resulting output control signal for the elevator's actuator theoretically may set the SERVO-Motor electrical angle in a range between +8° and +24°, which corresponds to a mechanical angle between -12° and +4°.

Figure 6.25: graphical representation of the effect of the elevator action on the vehicle's pitch angle.

"Cruise" is a conventional triangular membership function associated with a moderate elevator's manoeuvre, which may establish a moderate positive pitch attack angle. The resulting output control signal for the elevator's actuator theoretically may set the SERVO-Motor electrical angle in a range between +20° and +30°, which corresponds to a mechanical angle between 0° and +10°.

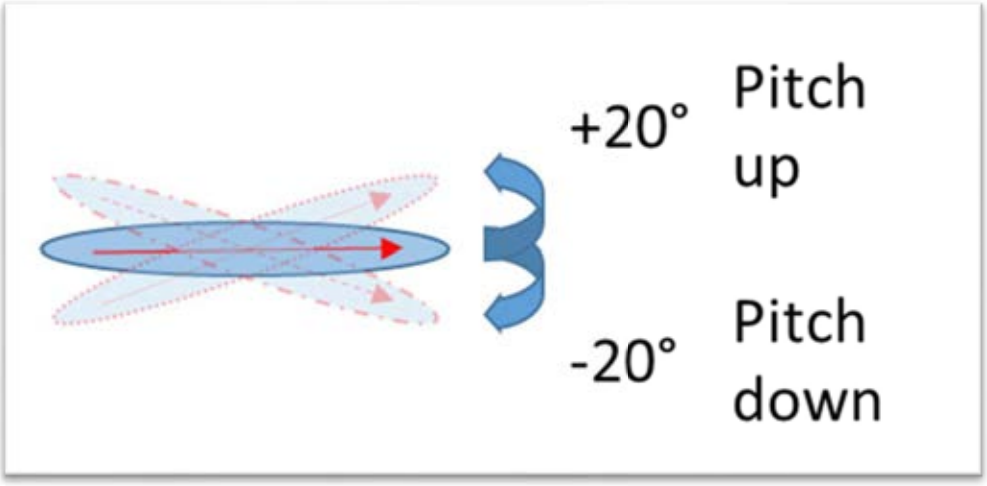


Figure 6.26: graphical elevator operation description.

"Climb" is a conventional triangular membership function associated with a substantial elevator's manoeuvre, which may cause a vehicle's large positive pitch angle. The resulting output control signal for the elevator's actuator theoretically may set the Servo electrical angle in a range between 24° and 40°, which corresponds to a mechanical angle between +4° and +20°.

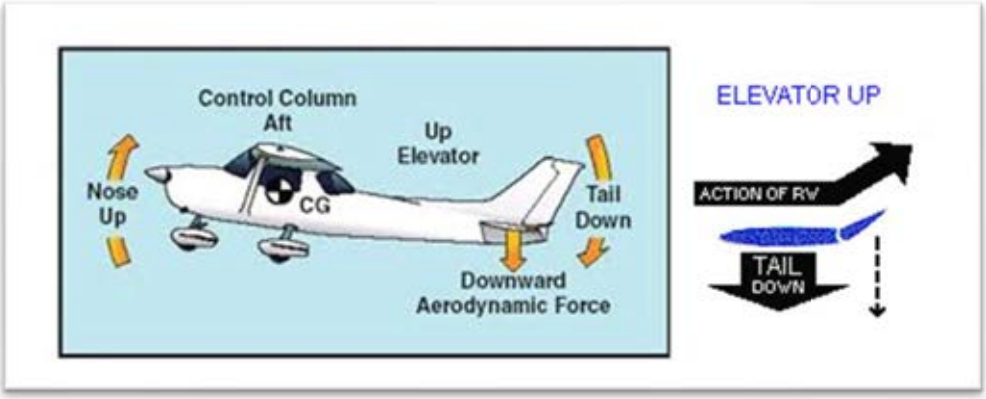


Figure 6.27: graphical representation of the effect of the elevator action on the vehicle's pitch angle.

6.3.11 Type "M1_THROTTLE", Membership Output Function

The Type "M1_THROTTLE" contains five MOFs, each of them associated with a specific throttle regulation of the left-wing powertrain that a "Human being Pilot" may execute in front of external conditions while a predefined route is followed. Membership Functions are:

a) glider_1;
b) Descending1;
c) ECOCRUISE1;

d) SUPERCRUISE1;

e) TAKE_OFF1.

By assumption, the “Torque Demand command” resolution chosen is 8-bit (0 to 255 steps).

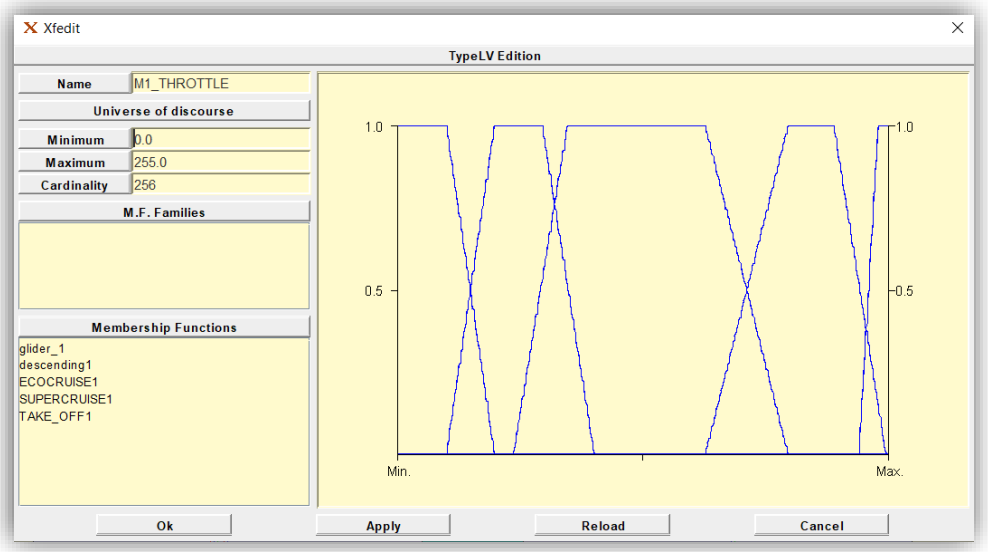


Figure 6.28: Type “M1_THROTTLE”, graphical representation.

The membership function “glider_1” is a conventional trapezoidal membership function associated with a very low torque demand generated by the “flight controller” to the “3ph Motor Driver” that manages the left-wing E-Motor. This membership function aims to allow the vehicle to perform a flight with a minimal thrust. Theoretically, the resulting torque demand may be set in a range between 0% and +19.5% (0 to 50).

“Descending1” is a conventional trapezoidal membership function associated with a low torque demand generated by the “flight controller” to the “3ph Motor Drive” that manages the left-wing E-Motor. The goal of this membership function is to help a safe descending manoeuvre. Theoretically, the resulting torque demand may set in a range between +9.8% and +39.8% (25 to 102).

“ECOCRUISE1” is a conventional trapezoidal membership function associated with a steady torque demand generated by the “flight controller” to the “3ph Motor Drive” that manages the left-wing E-Motor. This membership function aims to define the most effective (encouraging the low consumption operation) torque demand for a stable cruise flight. Theoretically, the resulting torque demand may set in a range between 23.4% and +79.2% (60 to 203).

“SUPERCRUISE1” is a conventional trapezoidal membership function associated with a consistent torque demand generated by the “flight controller” to the “3ph Motor Drive” that manages the left-wing E-Motor. This membership function aims to achieve a “fast cruise” flight or support manoeuvres that require broad thrust. Theoretically, the resulting torque demand may set in a range between 62.5% and +99.2% (160 to 254).

“TAKE_OFF1” is a conventional trapezoidal membership function associated with a full-throttle torque demand generated by the “flight controller” to the “3ph Motor Drive” that manages the left-wing E-Motor. The membership function is associated with particular manoeuvres such as the take-off manoeuvre or the stall avoiding manoeuvre, where the E-Motor shall operate at the highest power ratings. Theoretically, the resulting torque demand may set in a range between 93.8% and +100 % (240 to 255).

6.3.12 Type “M2_THROTTLE”, Membership Output Function

The Type “M2_THROTTLE” contains five MOFs, each of them associated with a specific throttle regulation of the right-wing powertrain that a “Human being Pilot” may execute in front of external conditions while a predefined route is followed. Membership Functions are:

a) glider_2;
b) Descending2;
c) ECOCRUISE2;
d) SUPERCRUISE2;
e) TAKE_OFF2.

By assumption, the “Torque Demand command” resolution chosen is 8-bit (0 to 255 steps).

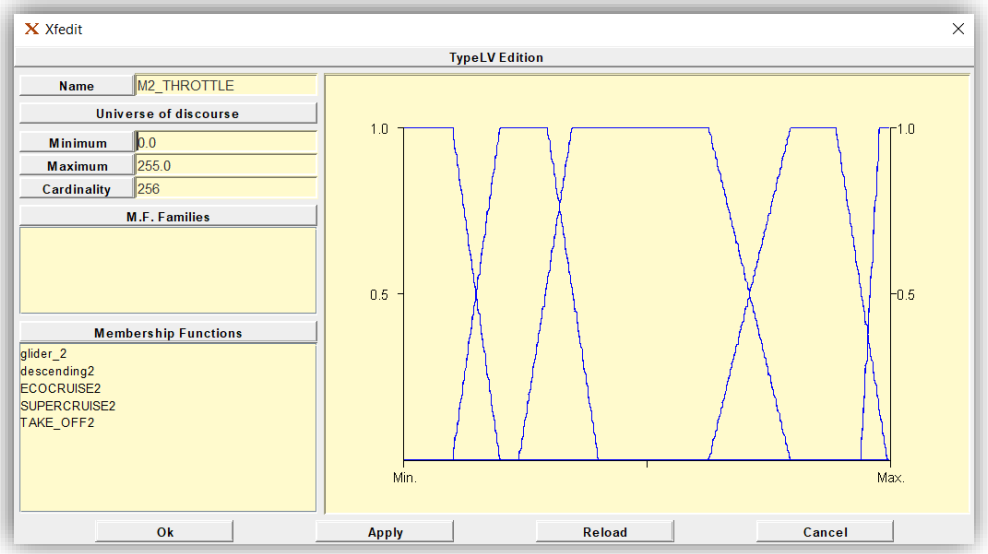


Figure 6.29: Type "M2_THROTTLE", graphical representation.

The membership function "glider_2" is a conventional trapezoidal membership function associated with a very low torque demand generated by the "flight controller" to the "3ph Motor Driver" that manages the right-wing E-Motor. This membership function aims to allow the vehicle to perform a flight with a minimal thrust. Theoretically, the resulting torque demand may be set in a range between 0% and +19.5% (0 to 50).

"Descending2" is a conventional trapezoidal membership function associated with a low torque demand generated by the "flight controller" to the "3ph Motor Drive" that manages the right-wing E-Motor. The goal of this membership function is to help a safe descending manoeuvre. Theoretically, the resulting torque demand may set in a range between +9.8% and +39.8% (25 to 102).

"ECOCRUISE2" is a conventional trapezoidal membership function associated with a steady torque demand generated by the "flight controller" to the "3ph Motor Drive" that manages the right-wing E-Motor. This membership function aims to define the most effective (encouraging the low consumption operation) torque demand for a stable cruise flight. Theoretically, the resulting torque demand may set in a range between 23.4% and +79.2% (60 to 203).

"SUPERCRUISE2" is a conventional trapezoidal membership function associated with a consistent torque demand generated by the "flight controller" to the "3ph Motor Drive" that manages the right-wing E-Motor. This membership function aims to achieve a "fast cruise" flight or support manoeuvres that require broad thrust. Theoretically, the resulting torque demand may set in a range between 62.5% and +99.2% (160 to 254).

"TAKE_OFF2" is a conventional trapezoidal membership function associated with a full-throttle torque demand generated by the "flight controller" to the "3ph Motor Drive" that manages the right-wing E-Motor. The membership function is associated with particular manoeuvres such as the take-off manoeuvre or the stall avoiding manoeuvre, where the E-Motor shall operate at the highest power ratings. Theoretically, the resulting torque demand may set in a range between 93.8% and +100 % (240 to 255).

6.4 Controller's Rulebases

By definition, to implement a parallel computational capable neuro-fuzzy controller, it is necessary to create a set of independent "Rulebase" for each output. As described in "Chapter 4", the "controller" shall generate five independent outputs; thus, five bespoke "Rulebases" shall be designed. For a bespoke "Rulebase" is intended the entity which encloses multiple combinations of rules. Each rule combines and weights sets of independent membership crisp values and associates a specific membership output function to be activated with the corresponding elaborated value. [32, 38 and 43]

6.4.1 "ELEV_SERVO", Rulebase

The generation of the control signal "SERVO_ELEV" (for the "elevator's SERVO-Motor") passes by the activation of the Type "ELEV_SERVO" MOFs. The rules and the weights for the membership functions activation are listed in the "Rulebase": "ELEV_SERVO". The activated membership function shall be "de-fuzzified" accordingly. The definition of the "defuzzification method" is set selecting the "Operatorset" in the "XFL3 GUI". [38]

For the study case, it is used the "SERVO_DEFUZZY" Operatorset, which uses the "xfl.CenterOfArea()" defuzzification function.

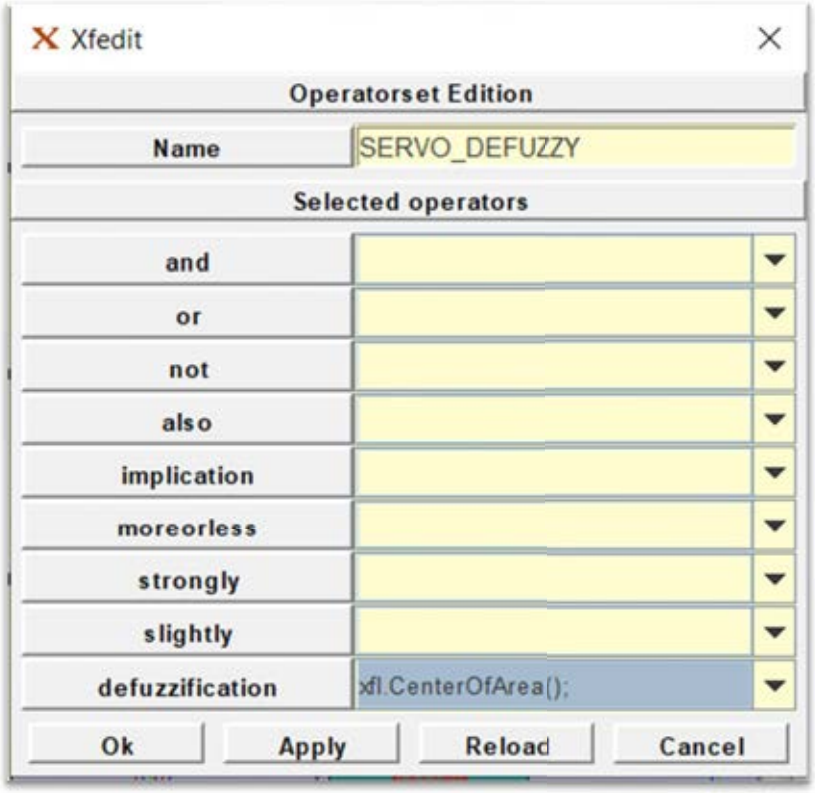


Figure 6.30: "Operatorset" GUI's interface.

The first step for the "Rulebase's design" is the definition of the "Rulebase's inputs" and the association of each input to a dedicated "Type". For the "ELEV_SERVO Rulebase", the inputs are:

a) PITCH (associated with the Type "Pitch_angle_input" and its 3 MIFs - pitch angle feedback);
b) ALT (associated with the Type "Altitude_input" and its 4 MIFs - altitude feedback);
c) V (associated with the Type "Speed_Input" and its 5 MIFs - speed information).

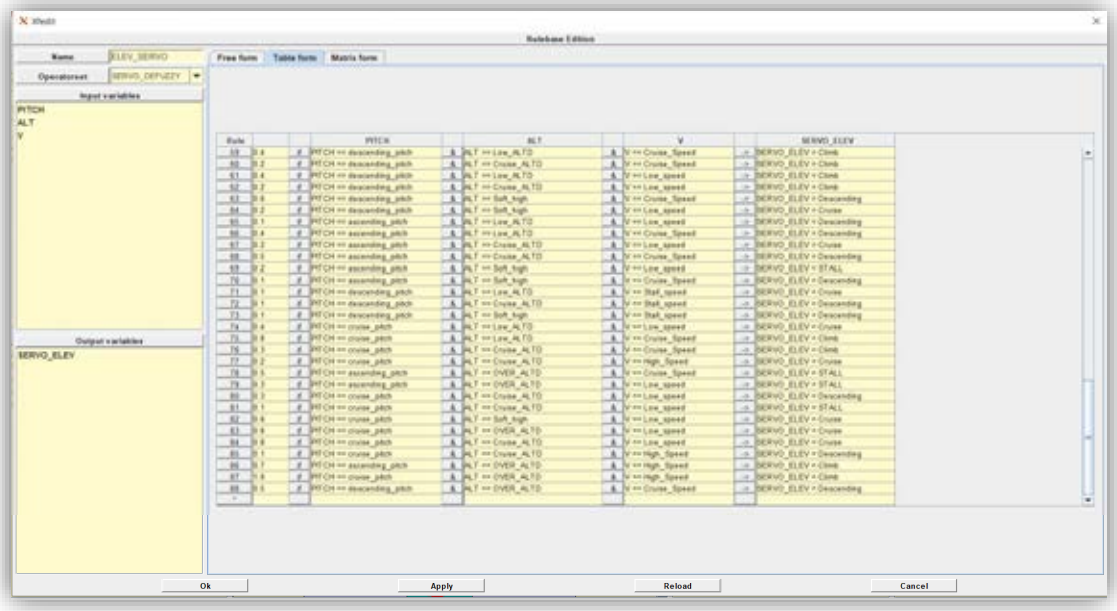


Figure 6.31: Table form of the "ELEV_SERVO Rulebase" (GUI's interface).

The Rulebase's design targets the minimisation of the number of rules (or combinations) in order to make more approachable the test of the controller and, more critical, the reduction of the FPGA logic gates requirements.

The design process is articulated in two steps. The first step uses an implementation that targets the activation of only one membership function for each determined combination of MIFs. Indeed, it is acceptable that a specific combination of MIFs shall activate, most likely with different weights, different MOFs. It represents the second step of the design process, which focuses on the "fine-tuning"[118] of the "Rulebase".

Current "Rulebase" incorporates "89" rules listed as a matrix of cases of "Human being Pilot" action-reaction behaviours, where for "action" it is intended the monitoring of the flight dynamic, and for "reaction" it is intended the elevator correction manoeuvres that a "Human Being Pilot" would most likely perform[119]. The level of the reaction severity is determined by the weight value associated with each rule.

[118] It is subject to the simulation outcomes. In fact, it necessary to work on fine-tuning to achieve enough robust controller before to move to the learning-training process.
[119] Assumptions made by the controller's designer.

6.4.2 "Aileron_SERVO", Rulebase

The generation of the control signal "AILERONS" (for the "aileron's SERVO-Motors") passes by the activation of the Type "Aileron_SERVO" MOFs. The rules and the weights for the membership functions activation are listed in the Rulebase: "Aileron_SERVO".

As previously described, the "defuzzification method" selected in the Rulebase's "Operatorset" is the "SERVO_DEFUZZY", which uses the "xfl.CenterOfArea()" defuzzification function.

For the "Aileron_SERVO Rulebase", the inputs are:

a) ALTD (associated with the Type "Altitude_input" and its 4 MIFs – the vehicle's altitude feedback);
b) Compass (associated with the Type "Compass_input" and its 3 MIFs - the vehicle's heading angle error);
c) ROLLING_angle (associated with the Type "Rolling_angle_input" and its 5 MIFs - rolling angle feedback);
d) YAW (associated with the Type "Yaw_angle_input" and its 3 MIFs - yaw angle feedback).

The Rulebase's design targets the minimisation of the number of rules (or combinations) in order to make more approachable the test of the controller and, more crucial, the reduction of the FPGA logic gates requirements.

The design process is articulated in two steps. The first step uses an implementation that targets the activation of only one membership function for each determined combination of MIFs. As previously described (paragraph 6.4.1), the second step of the design process focuses on the "fine-tuning" of the "Rulebase", adding, if necessary, a set of new weighted rules.

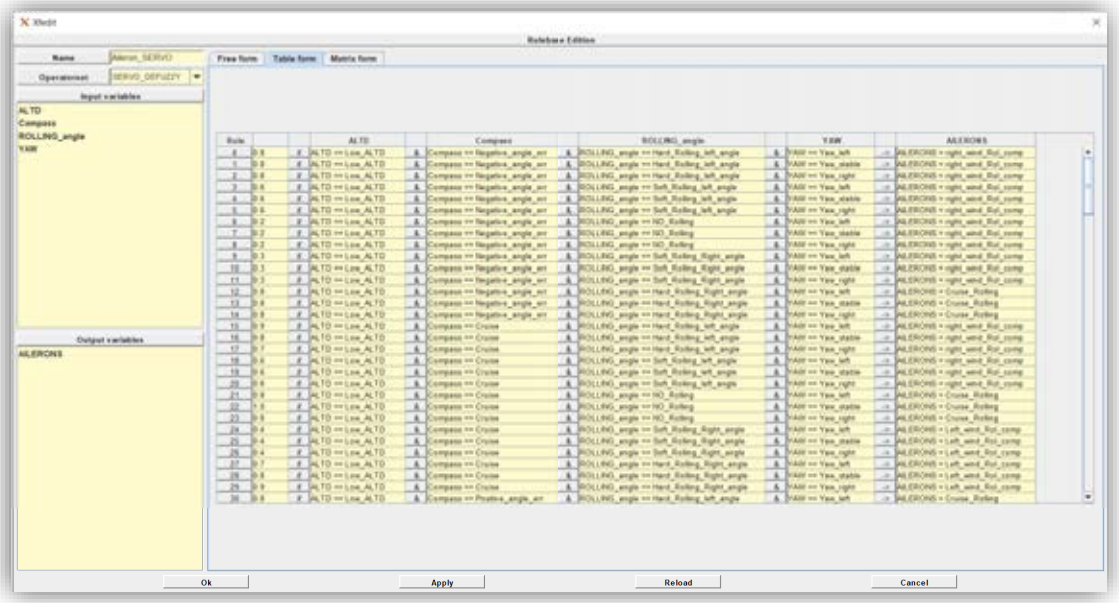


Figure 6.32: Table form of the "Aileron_SERVO Rulebase" (GUI's interface).

"Aileron_SERVO" Rulebase is built on "180" rules, listed as a matrix of cases of "Human Being Pilot" action-reaction behaviours, where for "action" it is intended the monitoring of the flight dynamic, and for "reaction" it is intended the rolling correction manoeuvres that a "Human Being Pilot" would most likely perform[120]. The level of the reaction severity is described by the weight value associated with each rule.

6.4.3 "RUDD_SERVO", Rulebase

The generation of the control signal "RUDDER" (for the "Rudder SERVO-Motor") passes by the activation of Type "Rudder_SERVO" MOFs. The rules and the weights for the membership functions activation are listed in the Rulebase: "RUDD_SERVO".

As previously described, the "defuzzification method" selected in the Rulebase's "Operatorset" is the "SERVO_DEFUZZY", which uses the "xfl.CenterOfArea()" defuzzification function.

For the "RUDD_SERVO Rulebase", the inputs are:

a) Compass (associated with the Type "Compass_input" and its 3 MIFs - the vehicle's heading angle error);

[120] Assumptions made by the controller's designer.

b) YAW (associated with the Type “Yaw_angle_input” and its 3 MIFs - yaw angle feedback);

c) ROLLING_angle (associated with the Type “Rolling_angle_input” and its 5 MIFs - rolling angle feedback);

d) PITCH (associated with the Type “Pitch_angle_input” and its 3 MIFs - pitch angle feedback).

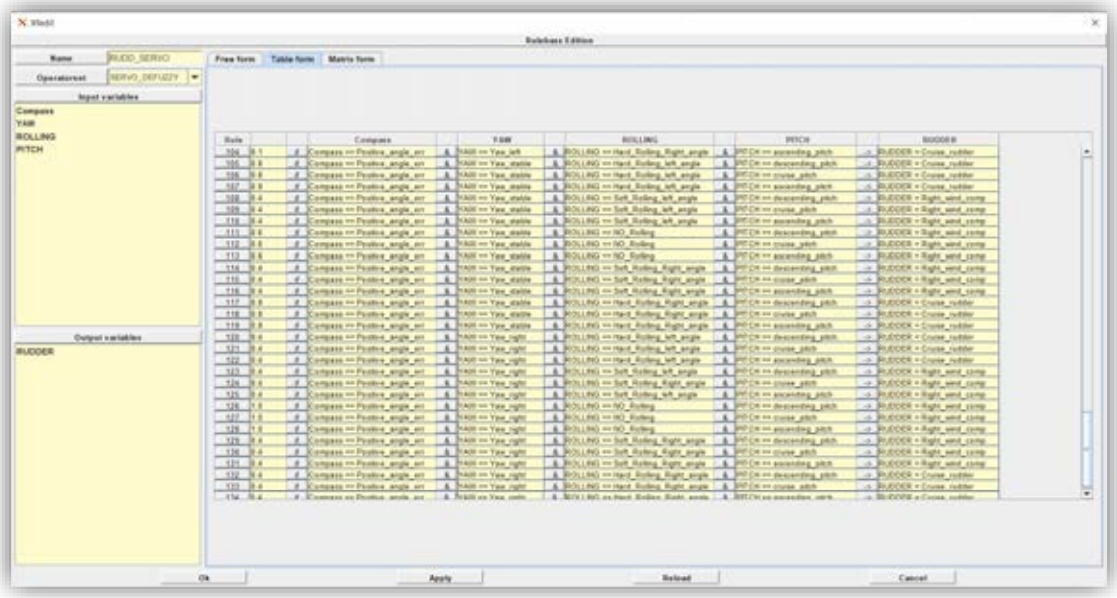


Figure 6.33: Table form of the “Rudder_SERVO Rulebase” (GUI’s interface).

The Rulebase’s design targets the minimisation of the number of rules (or combinations) in order to make more approachable the test of the controller and, more critical, the reduction of the FPGA logic gates requirements.

The design process is articulated in two steps. The first step uses an implementation that targets the activation of only one membership function for each determined combination of MIFs. As previously described (paragraph 6.4.1), the second step of the design process focuses on the “fine-tuning” of the “Rulebase”, adding, if necessary, a set of new weighted rules.

“RUDD_SERVO” Rulebase is built on “135” rules, listed as a matrix of cases of “Human Being Pilot” action-reaction behaviours, where for “action” it is intended the monitoring of the flight dynamic, and for “reaction” it is intended the yaw correction

manoeuvres that a "Human being pilot" would most likely perform[121]. The level of the reaction severity is described by the weight value associated with each rule.

6.4.4 "M1", Rulebase

The generation of the control signal for the "M1" (the "left-wing E-Motor" control signal) passes by the activation of Type "M1_THROTTLE" MOFs. The rules and the weights for the membership functions activation are listed in the Rulebase: "M1".

For the current "Rulebase", the "defuzzification method" selected in the Rulebase's "Operatorset" is defined as "neural", which uses the "xfl.CenterOfArea()" defuzzification function.

For the "M1 Rulebase", the inputs are:

a) Compass (associated with the Type "Compass_input" and its 3 MIFs - the vehicle's heading angle error);
b) YAW (associated with the Type "Yaw_angle_input" and its 3 MIFs - yaw angle feedback);
c) PITCH (associated with the Type "Pitch_angle_input" and its 3 MIFs - pitch angle feedback);
d) ROLLING (associated with the Type "Rolling_angle_input" and its 5 MIFs - rolling angle feedback);
e) ALT (associated with the Type "Altitude_input" and its 4 MIFs - altitude feedback);
f) V (associated with the Type "Speed_Input" and its 5 MIFs - speed sensor);
g) BATTERY (associated with the Type "Energy_Status" and its 4 MIFs - Battery SoC information).

[121] Assumptions made by the controller's designer.

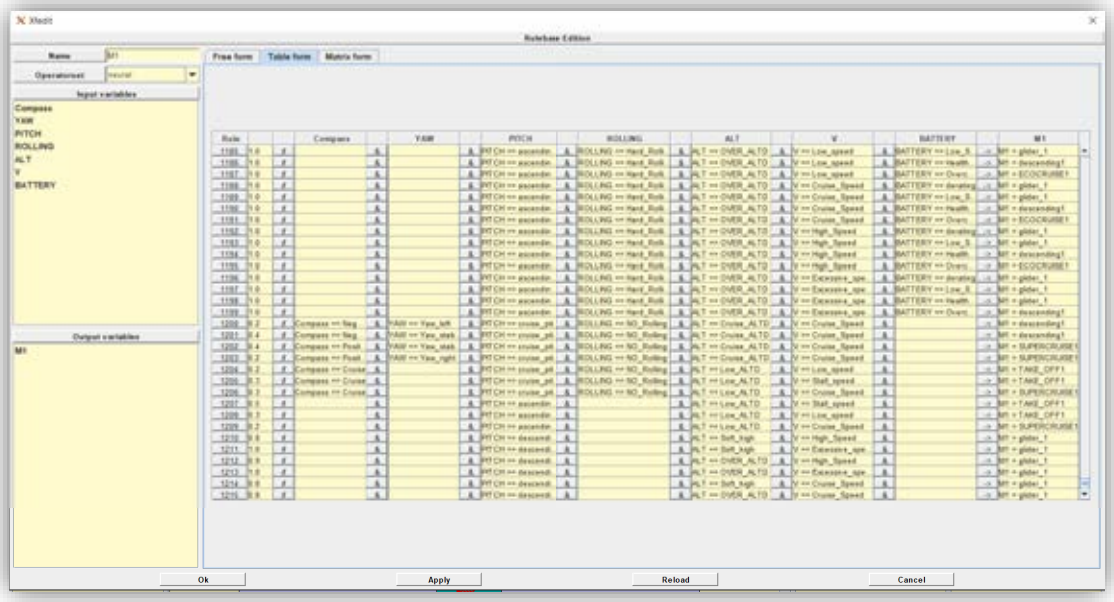


Figure 6.34: Table form of the "M1 Rulebase" (GUI's interface).

The Rulebase's design targets the minimisation of the number of rules (or combinations) in order to make more approachable the test of the controller and, more critical, the reduction of the FPGA logic gates requirements.

The presence of 7 inputs for the "Rulebase" makes it unrealistic[122] to cover each possible combination of MIFs. The identification of a method that allows a substantial reduction of the "Rulebase's" rules will require a few more design considerations.

The strategy used for the "Rulebase" design process is articulated in two steps. The first design step utilizes only five of the seven inputs and targets the activation of only one MOF for each determined combination of MIFs.

The second design step focuses on the definition of a set of combinations and rules from the most influential conditions that involve the whole set of seven inputs. This operation may be seen as the "fine-tuning" of the "Rulebase", adding, where necessary, a set of new weighted rules.

The generated "M1" Rulebase contains "1216" rules, listed as a matrix of cases of "Human Being Pilot" action-reaction behaviours, where for "action" it is intended the monitoring of the flight dynamic and for "reaction" it is intended the "left-wing E-Motor

[122] A large number of the input combination and the potential rules will produce a negligible contribution for the definition of the control quality and performances, but the controller cost (the computational power required; thus FPGA logic gates, will limit the selection of the physical device to the higher performance family) would increase significantly.

Throttle" control that a "Human being pilot" would most likely perform. The level of the reaction severity is described by the weight value associated with each rule. As previously described, a specific combination of MIFs may activate, most likely with different weights, different MOFs.

It is of paramount importance to highlight that the definitions of the rules and the associated weights does not only aim to perform a correct flight and ensure system protection and reliability but also targets the minimisation of the energy used. The desired outcome is the life extension of the system most vulnerable components: the REESS.

6.4.5 "M2", Rulebase

The generation of the control signal for the "M2" (the "right-wing E-Motor" control signal) passes by the activation of Type "M2_THROTTLE" MOFs. The rules and the weights for the membership functions activation are listed in the Rulebase: "M2".

For the current "Rulebase", the "defuzzification method" selected in the Rulebase's is the "neural" "Operatorset", which uses the "xfl.CenterOfArea()" defuzzification function.

For the "M2 Rulebase", the inputs are:

a) Compass (associated with the Type "Compass_input" and its 3 MIFs - the vehicle's heading angle error);
b) YAW (associated with the Type "Yaw_angle_input" and its 3 MIFs - yaw angle feedback);
c) PITCH (associated with the Type "Pitch_angle_input" and its 3 MIFs - pitch angle feedback);
d) ROLLING (associated with the Type "Rolling_angle_input" and its 5 MIFs - rolling angle feedback);
e) ALT (associated with the Type "Altitude_input" and its 4 MIFs - altitude feedback);
f) V (associated with the Type "Speed_Input" and its 5 MIFs - speed sensor);
g) BATTERY (associated with the Type "Energy_Status" and its 4 MIFs - Battery SoC information).

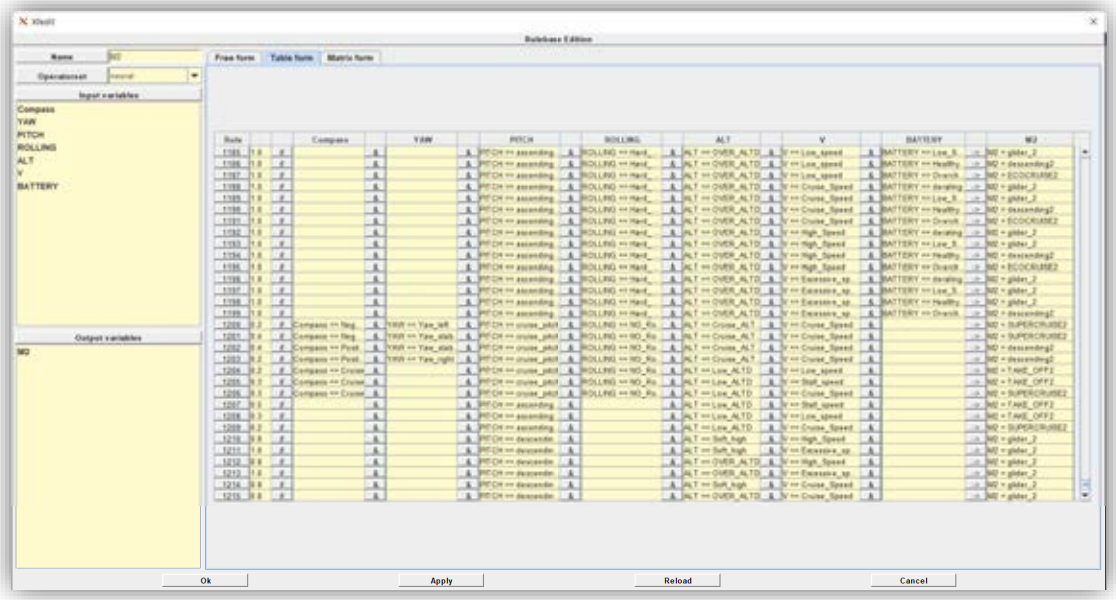


Figure 6.35: Table form of the "M2 Rulebase" (GUI's interface).

The Rulebase's design targets the minimisation of the number of rules (or combinations) in order to make more approachable the test of the controller and, more critical, the reduction of the FPGA logic gates requirements.

The presence of seven inputs for the "Rulebase" makes it unrealistic[123] to cover each possible combination of MIFs. The identification of a method that allows a substantial reduction of the "Rulebase's" rules will require a few more design considerations.

The strategy used for the "Rulebase" design process is articulated in two steps. The first design step utilizes only five of the seven inputs and targets the activation of only one MOF for each determined combination of MIFs.

The second design step focuses on the definition of a set of combinations and rules from the most influential conditions that involve the whole set of 7 inputs. This operation may be seen as the "fine-tuning" of the "Rulebase", adding, where necessary, a set of new weighted rules.

The generated "M2" Rulebase contains "1216" rules, listed as a matrix of cases of "Human Being Pilot" action-reaction behaviours, where for "action" it is intended the monitoring of the flight dynamic and for "reaction" it is intended the "right-wing E-Motor

[123] A large number of the input combination and the potential rules will produce a negligible contribution for the definition of the control quality and performances, but the controller cost (the computational power required; thus FPGA logic gates, will limit the selection of the physical device to the higher performance family) would increase significantly.

Throttle” control that a “Human being pilot” would most likely perform. The level of the reaction severity is described by the weight value associated with each rule. As previously described, a specific combination of MIFs may activate, most likely with different weights, different MOFs.

It is of paramount importance to highlight that the definitions of the rules and the associated weights does not only aim to perform a correct flight and ensure system protection and reliability but also targets the minimisation of the energy used. The desired outcome is the life extension of the system most vulnerable components: the REESS.

6.5 Fuzzy Controller System Structure

“Figure 6.36” shows the high level “fuzzy controller structure” described in the previous paragraphs, using a hardware description layout. In the picture, it is possible to observe on the left-side the controller’s inputs and on the right side the controller’s output. The internal connections between the controller’s blocks, where each block represents a controller’s “Rulebase”, demonstrate the controller’s parallel computation capability; each “Rulebase” operates independently and in parallel with all other “Rulebases”.

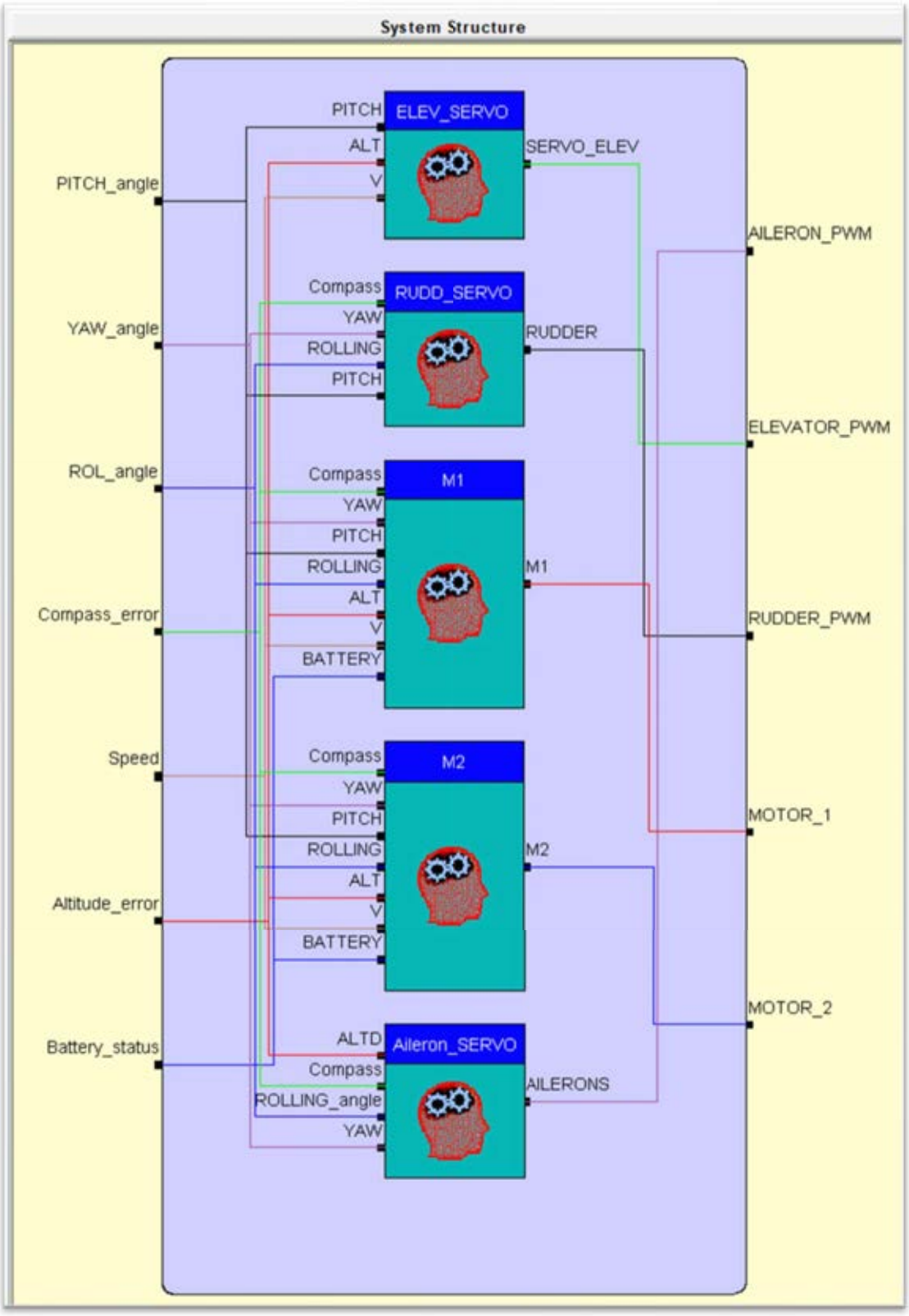


Figure 6.36: "Fuzzy Controller Structure" designed in the XFUZZY environment.

6.6 Fuzzy Controller Simulations and preliminary optimisation

Before exporting the "fuzzy controller" in the "VHDL" language, it is recommendable to implement a series of simulation and, if possible, optimise the "fuzzy controller". There are several options available able to perform a fuzzy controller simulation. The Author's appreciates the functionalities available in the XFUZZY environment and its GUI[124]. The GUI's functionalities used for the proposed work are:

a) controller's graphical representation;
b) Type's membership functions optimisation;
c) Rulebase's optimisation;
d) fuzzy controller behaviour "Monitorization" (defines the controller's exact outputs in front of a predefined set of inputs, it may be associated with a mathematical solution of the physical controller's equations).

From the listed functionalities, the Author decides to emphasise the utilisation of the "Inference Monitor" functionality to perform a controller's raw simulation. The "goal" of the raw simulation is to check the "controller response" in certain conditions, such as take-off, landing, route adjustments, steady-state flight (stable flight), and gusty winds compensation manoeuvre.

The proposal relies on the outcome of this set of simulations to implement an iteration based adjustment of the Rulebase's "weights" and "functions" (or weighted rules) before exporting the "VDLH" algorithms and perform any learning processes.

6.6.1 Take-Off simulation

Take-off and landing are the most complex tasks to be performed. Take-off simulations result in being very complex to simulate without a very detailed and comprehensive model of the vehicle dynamics behavioural[125]. The assumption is to identify several manoeuvre points and check the controller behaviour in those specific moments. Of course, as many points may be identified as more reliable, it will be the raw simulation outcome.

The procedure for take-off starts with the vehicle accelerating until it reaches enough speed. The pilot can then rotate the vehicle, and it will start ascending. The determination of

[124] The "Digital" description of the "Fuzzy Controller" built on XFUZZY3.5 (version 3.5).
[125] Parameters like aero-dynamical coefficients, vehicle acceleration behaviour, stall speed, take-off speed and etc., have paramount importance for very detailed simulation.

this particular speed called rotation speed (V_R), is a critical factor in determining take-off performance[126]. For safety reasons, V_R is usually determined as being:

$$V_R = 1.1 \cdot V_{stall}$$

(Equation 41)[127]

Alternatively (whichever is greater[128]):

$$V_R = 1.05 \cdot V_{Min,CONTROL}$$

(Equation 42)

It can be calculated based on knowledge of the aircraft take-off configuration and hence the maximum achievable lift coefficient $C_{L(\max)}$. To maintain level flight, the lift produced must equal the weight; hence the stall speed can be calculated as:

$$V_{stall} = \sqrt{\left(\frac{2 \cdot W}{C_{L(max)} \cdot \rho \cdot S}\right)}$$

(Equation 43)

The raw simulation aims to observe the "Controller" behaviour with particular attention in five critical moments of the take-off manoeuvre (the manoeuvre start in $t=t_0$ may also not be considered a pivotal moment). Looking at the following pictures, Figure 6.37 and Figure 6.38, it is possible to identify such moments as:

a) manoeuvre start, the vehicle is going to accelerate (speed $v(t) = 0$, at $t=t_0=0$);
b) the vehicle is at the acceleration's peak, and it is moving with speed $v(t) = V_a$, ($t = t_1$, Figure 6.38);
c) the vehicle is moving at speed $v(t) = \frac{V_R}{\sqrt{2}} = V_1$, ($t = t_2$, Figure 6.38);
d) the vehicle reaches the "rotation speed", it is moving with speed $v(t) = V_R$, ($t=t_3$, Figure 6.37);
e) the vehicle is moving at speed $v(t) = V_2$, where $V_{stall} \cdot \sqrt{2} \leq V_2 \leq 2 \cdot V_{stall}$ ($t = t_4$, Figure 6.37);
f) the vehicle moving with speed $v(t) = V_c;\ V_2 < V_R \cdot \sqrt{2} \leq V_c \leq 2 \cdot V_R$, ($t = t_5$).

[126] Take-off rules vary slightly depending on the aircraft category. Small commuter aircraft should be considered as meeting "FAR 23" rules, and transport category aircraft should comply with "FAR 25" rules.
[127] Stall speed, V_{stall}, is the lowest speed that the aircraft can be flown before the airflow starts to separate from wings as the angle of attack becomes too great. The wing is assumed in this case to be in take-off configuration or "clean".
[128] For a conventional aircraft, there is only a small difference between V_R calculations based on stall speed or minimum control speed. Minimum control speed, $V_{Min,Control}$ is a more complex calculation and requires knowledge of the stall characteristics, of the tailplane and of the elevator.

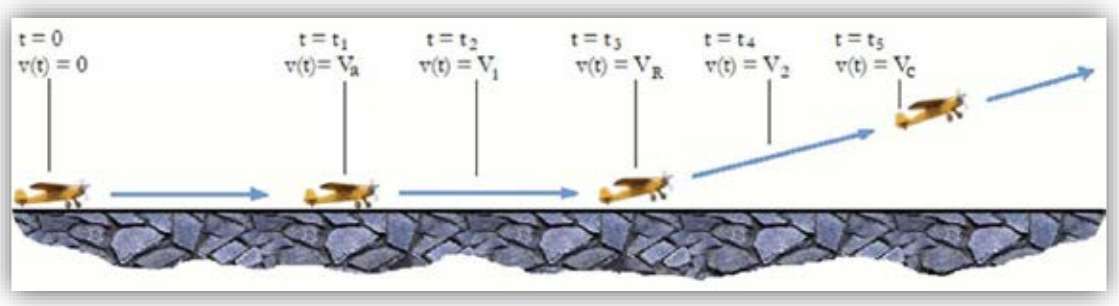


Figure 6.37: take-off manoeuvre, graphical animation.

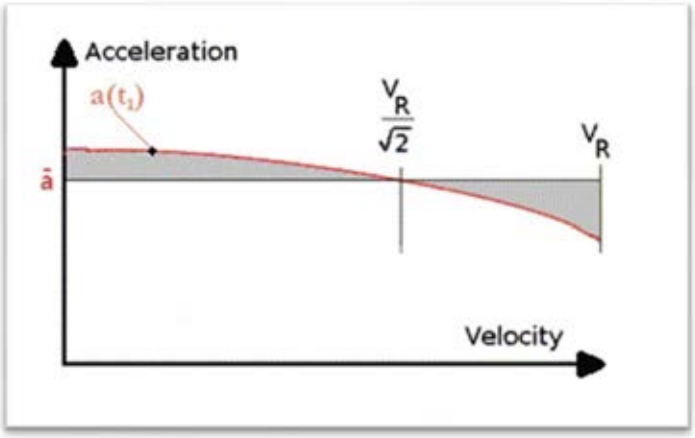


Figure 6.38: take-off manoeuvre, acceleration vs velocity graph.

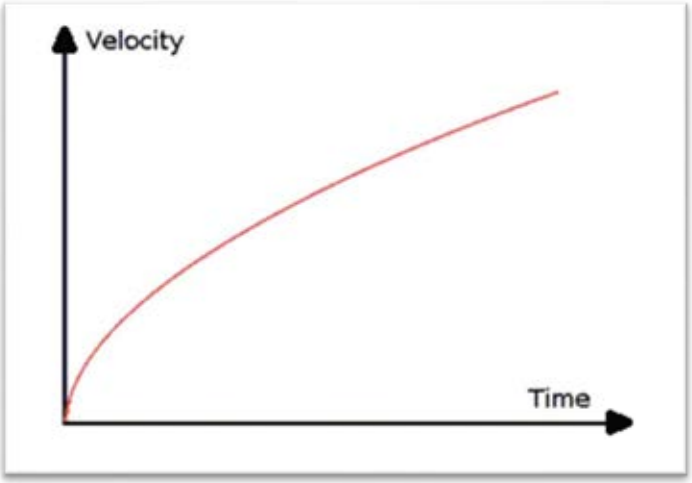


Figure 6.39: Take-off manoeuvre, velocity vs time indicative graph.

6.6.1.1 Simulation at t=t0

For the circumstance "t = t_0", by assumption, it is used a set of controller's input parameters compatible with the physical environmental parameters that a human pilot may observe at the moment of the take-off manoeuvre start. Those parameters are:

- "Pitch attack angle" is approximately 0.8 degrees (PITCH_angle);
- "Yaw angle" is approximately 0 degrees (YAW_angle);
- "Rolling angle" is approximately 0 degrees (ROL_angle);
- "Navigation Heading angle error" is 0 degrees (Compass_Error)[129];
- "Vehicle's speed" read is 0%[130](Speed);
- "Altitude relative error"[131] read is approximately -67% (Altitude error);
- "Battery's SoC" read is 95.3% (Battery_status).

Those parameters shall be translated into an 8-bit digital form accordingly. A human pilot looks at the sensors that may appear as values in the previous format. The controller reads 8-bit resolution digital processed input values, where for digital processing it is meant the data elaboration disserted in "Chapter 5". It means that the set of input parameters, delivered into the "XFUZZY Inference Monitor" tool, are chosen according to the "Chapter 5" definition of the digital processes.

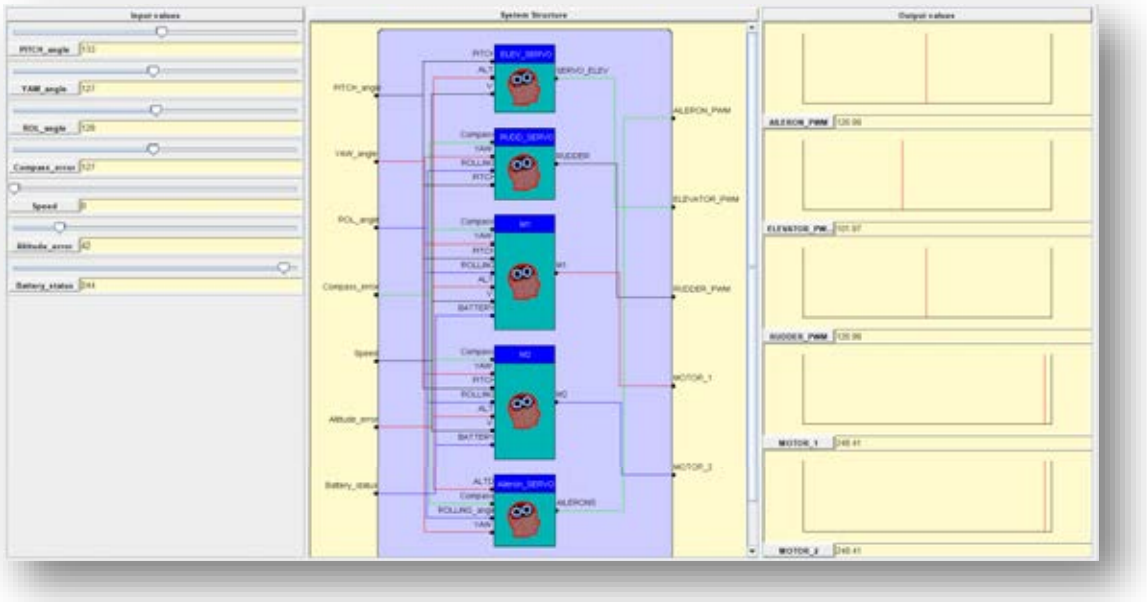


Figure 6.40: simulation's outputs at t=t₀.

"Figure 6.40" represents the result of the simulation taken into a defined instant, where the input parameters are the previous mechanical parameters listed, opportunely translated in

[129] Heading angle error, it is set "null" during the take-off manoeuvre by the "VHDL compass function".
[130] It is used the absolute maximum rated speed of the vehicle as to the 100% reference.
[131] It may be addressed as well as "Approximation error". Reference formulas are : $E_r = \frac{E_A}{y_m}$, where E_r is the relative error, E_A is the absolute error and y_m is the average value (or reference value). The controller uses the percent error: $E_{r\%} = 100 \cdot E_r$.

digital form, and the output values are the "fuzzy controller's output values". As for the input values, output values shall be digitally processed before becoming the physical control signals that could go out of the FPGA, according to the definitions of "Chapter 5". Fundamentally, the "fuzzy controller" generates an 8-bit resolution digital information format for each output, which shall be translated into a PWM signal for the SERVO Motor or into a torque demand request via RS232 to the powertrain's driver interface. The mechanical outcome opportunely translated from "Figure 6.40" are:

- the mechanical angle of the "Ailerons" set to an approximately 0-degree position;
- the mechanical angle of the "Elevator" set to approximately -4 degrees position;
- the mechanical angle of the "Rudder" set to an approximately 0-degree position;
- left motor torque demand set to 96.9%[132];
- right motor torque demand set to 96.9%.

6.6.1.2 Simulation at $t=t_1$

The assumptions made for the controller's behaviour simulation at the moment $t=t_1$ is that the vehicle is in "full acceleration"[133] (graph of Figure 6.38) and that a set of controller's input parameters, compatible with the physical environmental parameters that a human pilot may observe at that moment, are used. Those parameters are:

- "Pitch attack angle" is approximately +0.5 degrees (PITCH_angle);
- "Yaw angle" is approximately 0 degree (YAW_angle);
- "Rolling angle" is approximately 0 degree (ROL_angle);
- "Navigation Heading angle error" is 0 degrees (Compass_Error)[134];
- "Vehicle's speed" read is 7.8% (Speed);
- "Altitude relative error" read is approximately -67% (Altitude error);
- "Battery's SoC" read is 94.9% (Battery_status).

As previously described, the set of input parameters, delivered into the "XFUZZY Inference Monitor" tool, are chosen according to "Chapter 5" definition of the digital

[132] It is the percentage of the absolute maximum torque achievable by the powertrain motor (it is a safety value set by the user on the E-Motor driver parametrisation).
[133] For "full acceleration" it is intended that the vehicle's speed is significantly below the parameter "V_{stall}", its acceleration reached the absolute "acceleration peak", and the vehicle's acceleration is going to decrease according to the graphs of Figure 6.38 and Figure 6.39.
[134] The vehicle's heading angle error, it is set to "null" during the take-off manoeuvre by the VHDL SW compass function.

processes and according to “paragraph 6.6.1.1” consideration regarding the vehicle’s speed and altitude relative error.

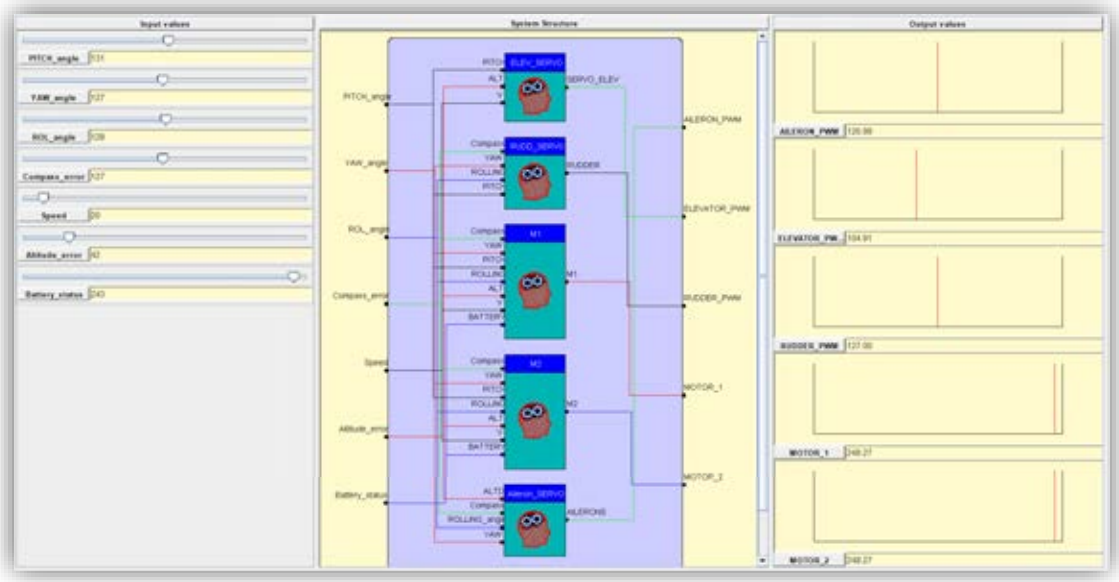


Figure 6.41: simulation’s outputs at t=t_1.

Figure 6.41 represents the result of the simulation taken under the conditions associated with the instant t=t_1. Each value present in the simulation requires a digital process to be associable with a physical value (as previously described). The mechanical outcomes opportunely translated from “Figure 6.41” are:

- the mechanical angle of “Ailerons” set to an approximately 0-degree position;
- the mechanical angle of “Elevator” set to approximately -3.6 degrees position;
- the mechanical angle of “Rudder” set to an approximately 0-degree position;
- left motor torque demand set to 96.9%;
- right motor torque demand set to 96.9%.

Previously described, “paragraph 6.6.1.1”, considerations regarding the vehicle’s torque demand are valid for this simulation and will be valid for all following simulations.

6.6.1.3 Simulation at t=t_2

The assumptions made for the controller’s behaviour simulation at the moment t=t_2 is that the vehicle speed is equal to $v(t) = \frac{V_R}{\sqrt{2}} = V_1$ (graph of “Figure 6.38”). Furthermore, a set of controller’s input parameters, compatible with the physical environmental parameters that a human pilot may observe at that moment, are used. Those parameters are:

- “Pitch attack angle” is approximately 0.3 degrees (PITCH_angle);

- “Yaw angle” is approximately 0 degree (YAW_angle);
- “Rolling angle” is approximately 0 degree (ROL_angle);
- “Navigation Heading angle error” is 0 degree (Compass_Error)[135];
- “Vehicle’s speed” read is 16.4% (Speed);
- “Altitude relative error” read is approximately -67% (Altitude error);
- “Battery’s SoC” read is 94.5% (Battery_status).

As previously described, the set of input parameters, delivered into the “XFUZZY Inference Monitor” tool, are chosen according to “Chapter 5” definition of the digital processes.

Previous, “paragraph 6.6.1.1”, consideration regarding the vehicle’s speed and altitude relative error are valid and will be valid for all following simulations.

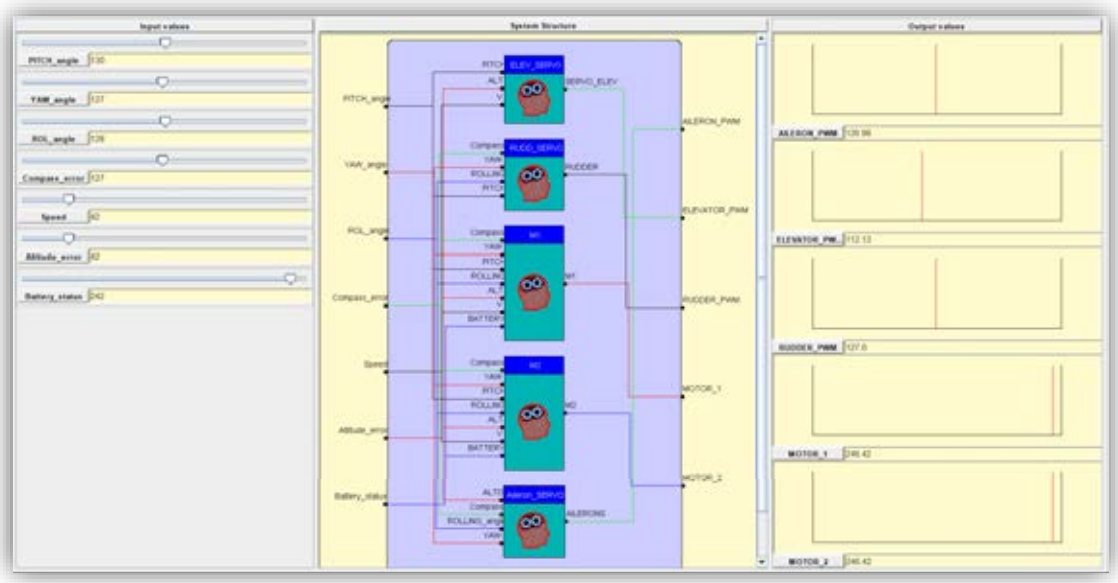


Figure 6.42: simulation’s outputs at $t=t_2$.

Figure 6.42 represents the results of the simulation taken under the conditions associated with the instant $t=t_2$. Each value present in the simulation requires a digital process to be associable to a physical value (as previously described). The mechanical outcomes opportunely translated from “Figure 6.42” are:

- the mechanical angle of “Ailerons” set to an approximately 0-degree position;
- the mechanical angle of “Elevator” set to approximately -2.5 degrees position;

[135] The vehicle’s heading angle error, it is set to “null” during the take-off manoeuvre by the VHDL SW compass function.

- the mechanical angle of “Rudder” set to an approximately 0-degree position;
- left motor torque demand set to 96.1%;
- right motor torque demand set to 96.1%.

6.6.1.4 Simulation at t=t_3

The assumptions made for the controller’s behaviour simulation at the moment t=t_3 is that the vehicle speed is equal to $v(t) = V_R$ (graphs of “Figure 6.37” and “Figure 6.38”). Moreover, a set of controller’s input parameters, compatible with the physical environmental parameters that a human pilot may observe when the vehicle reaches the “Rotational Speed”, are used. Those parameters are:

- “Pitch attack angle” is approximately 0 degree (PITCH_angle);
- “Yaw angle” is approximately 0 degree (YAW_angle);
- “Rolling angle” is approximately 0 degree (ROL_angle);
- “Navigation Heading angle error” is 0 degree (Compass_Error)[136];
- “Vehicle’s speed” read is 23.4% (Speed);
- “Altitude relative error” read is -67% (Altitude error);
- “Battery’s SoC” read is 94.1% (Battery_status).

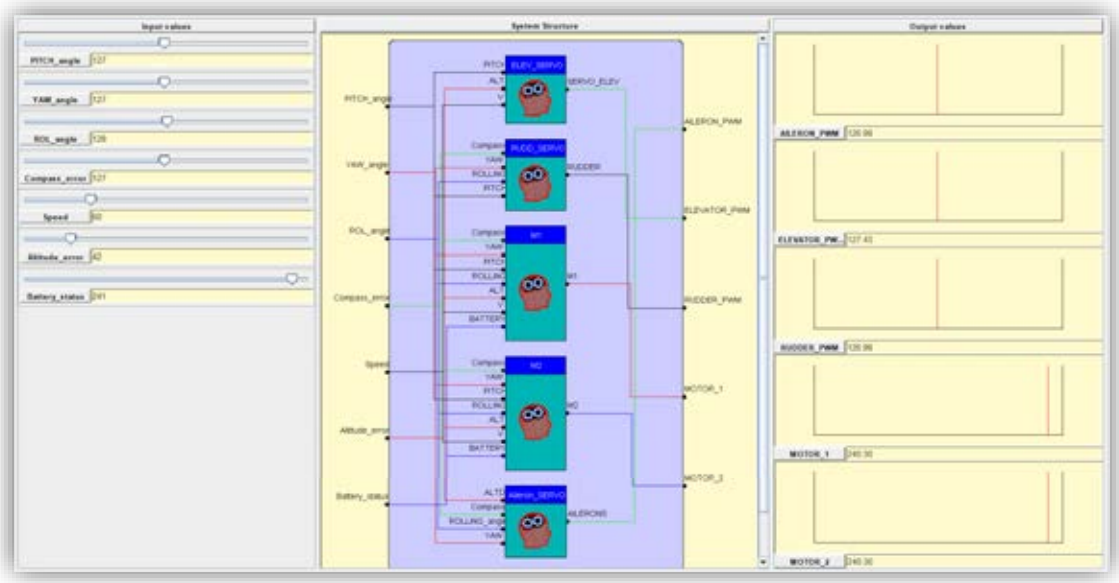


Figure 6.43: simulation’s outputs at t=t_3.

[136] The vehicle’s heading angle error, it is set to “null” during the take-off manoeuvre by the VHDL SW compass function.

As previously described, the set of input parameters delivered into the "XFUZZY Inference Monitor" tool are chosen according to the "Chapter 5" definition of the digital processes.

Figure 6.43 represents the results of the simulation taken under the conditions associated with the instant $t=t_3$. Each value present in the simulation requires a digital process to be associable with a physical value (as previously described). The mechanical outcomes opportunely translated from Figure 6.43 are:

- the mechanical angle of "Ailerons" set to an approximately 0-degree position;
- the mechanical angle of "Elevator" set to an approximately 0-degrees position;
- The mechanical angle of "Rudder" set to an approximately 0-degree position;
- left motor torque demand set to 93.8%;
- right motor torque demand set to 93.8%.

6.6.1.5 Simulation at $t=t_4$

The assumptions made for the controller's behaviour simulation at the moment $t=t_4$ is that the vehicle speed is equal to $v(t) = V_2$, where $V_{stall} \cdot \sqrt{2} \leq V_2 \leq 2 \cdot V_{stall}$ (graphs of "Figure 6.37" and "Figure 6.38"). Then a set of controller's input parameters, compatible with the physical environmental parameters that a human pilot may observe when the vehicle starts the ascending manoeuvre at the beginning of vehicle rotation, are used. Those parameters are:

- "Pitch attack angle" is approximately +2.8 degrees (PITCH_angle);
- "Yaw angle" is approximately 0 degree (YAW_angle);
- "Rolling angle" is approximately 0 degree (ROL_angle);
- "Navigation Heading angle error" is 0 degree (Compass_Error)[137];
- "Vehicle's speed" read is 34.4% (Speed);
- "Altitude relative error" read is -50.8% (Altitude error);
- "Battery's SoC" read is 93.8%.

As previously described, the set of input parameters, delivered into the "XFUZZY Inference Monitor" tool, are chosen according to "Chapter 5" definition of the digital processes.

[137] The vehicle's heading angle error, it is set to "null" during the take-off manoeuvre by the VHDL SW compass function.

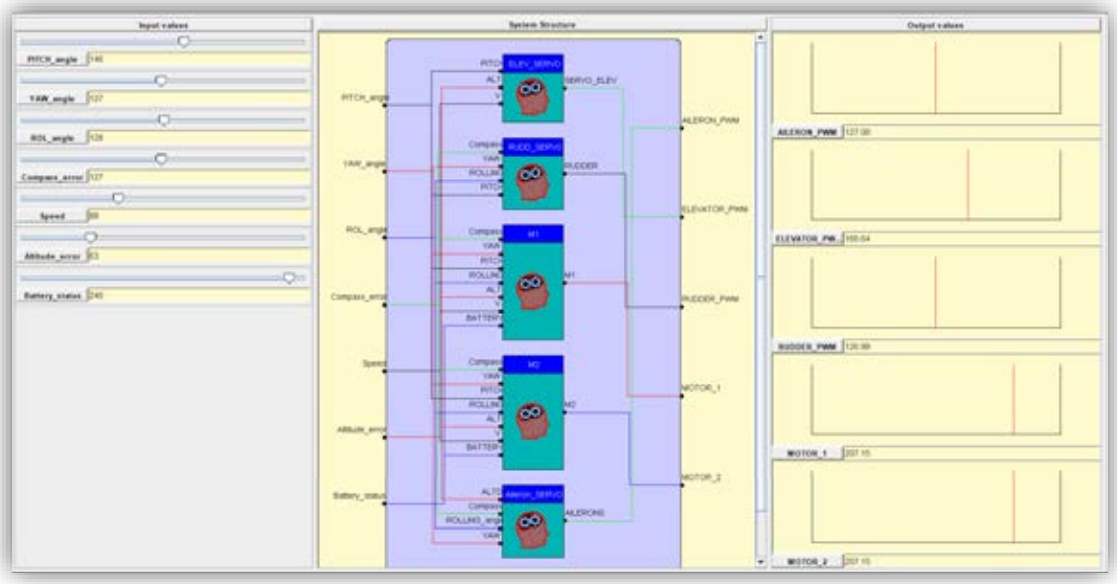


Figure 6.44: simulation's outputs at t=t_4.

"Figure 6.44" represents the results of the simulation taken under the conditions associated with the instant t=t_4. Each value present in the simulation requires a digital process to be associable with a physical value (as previously described). The mechanical outcomes opportunely translated from "Figure 6.44" are:

- the mechanical angle of "Ailerons" set to an approximately 0-degree position;
- the mechanical angle of "Elevator" set to approximately +5.2 degrees position;
- the mechanical angle of "Rudder" set to an approximately 0-degree position;
- left motor torque demand set to 80.9%;
- right motor torque demand set to 80.9%.

6.6.1.6 Full Climbing manoeuvre at t = t_5

The assumptions made for the controller's behaviour simulation at the moment t=t_5 is that the vehicle speed is equal to $v(t) = V_c$, where $V_2 < V_R \cdot \sqrt{2} \leq V_c \leq 2 \cdot V_R$ (graphs of "Figure 6.37" and "Figure 6.38"). Then a set of controller's input parameters, compatible with the physical environmental parameters that a human pilot may observe when the vehicle is in the middle of the ascending manoeuvre, are used. Those parameters are:

- "Pitch attack angle" is approximately +5 degrees (PITCH_angle);
- "Yaw angle" is approximately 0 degree (YAW_angle);
- "Rolling angle" is approximately 0 degree (ROL_angle);

- "Navigation Heading angle error" is 0 degree (Compass_Error)[138];
- "Vehicle's speed" read is 41% (Speed);
- "Altitude relative error" read is -43% (Altitude error);
- "Battery's SoC" read is 92.6% (Battery_status).

As previously described, the set of input parameters, delivered into the "XFUZZY Inference Monitor" tool, are chosen according to "Chapter 5" definition of the digital processes.

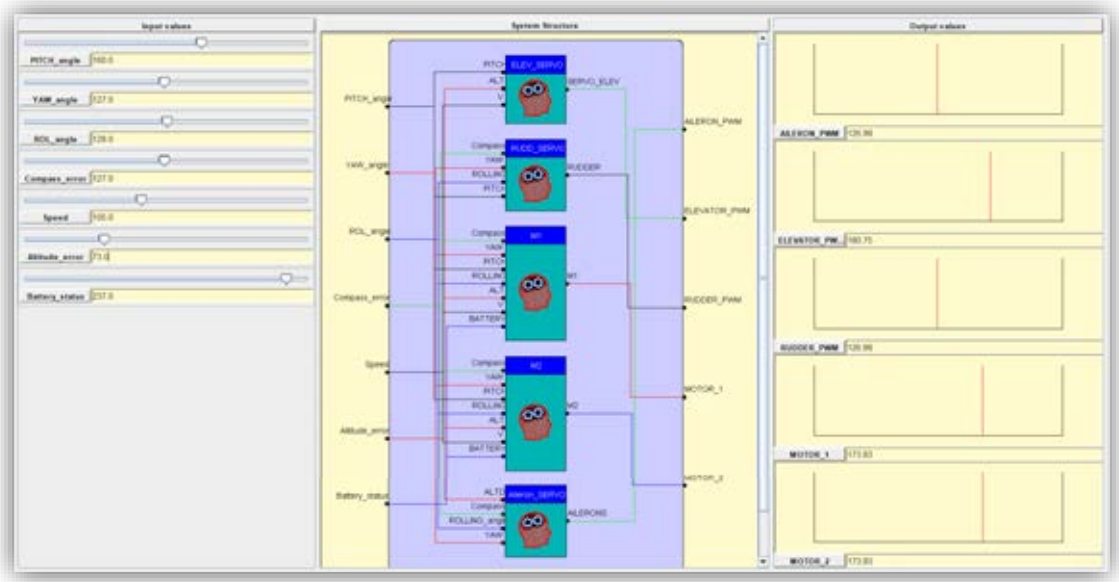


Figure 6.45: simulation's outputs at t=t_5.

Figure 6.45 represents the result of the simulation taken under the conditions associated with the instant t=t_5. Each value present in the simulation requires a digital process to be associable with a physical value (as previously described). The mechanical outcome opportunely translated from "Figure 6.45" are:

- the mechanical angle of "Ailerons" set to an approximately 0-degree position;
- the mechanical angle of "Elevator" set to approximately +8.3 degrees position;
- the mechanical angle of "Rudder" set to an approximately 0-degree position;
- left motor torque demand set to 68%;
- right motor torque demand set to 68%.

[138] The vehicle's heading angle error, it is set to "null" during the take-off manoeuvre by the VHDL SW compass function.

6.6.1.7 Take-off simulation Conclusion

The simulations outcome shows that the controller's decisions are compatible with the decisions that a human pilot may take in similar environmental conditions.

6.6.2 Route adjustment Simulation

In order to allow a smooth take-off, a high-level system assumption values the "enable" of the vehicle's "heading angle correction. This design assumption complies with all other design requirements for the VHDL block that generates the heading angle error 8-bit data, which generates the "fuzzy controller" input "Compass_error".

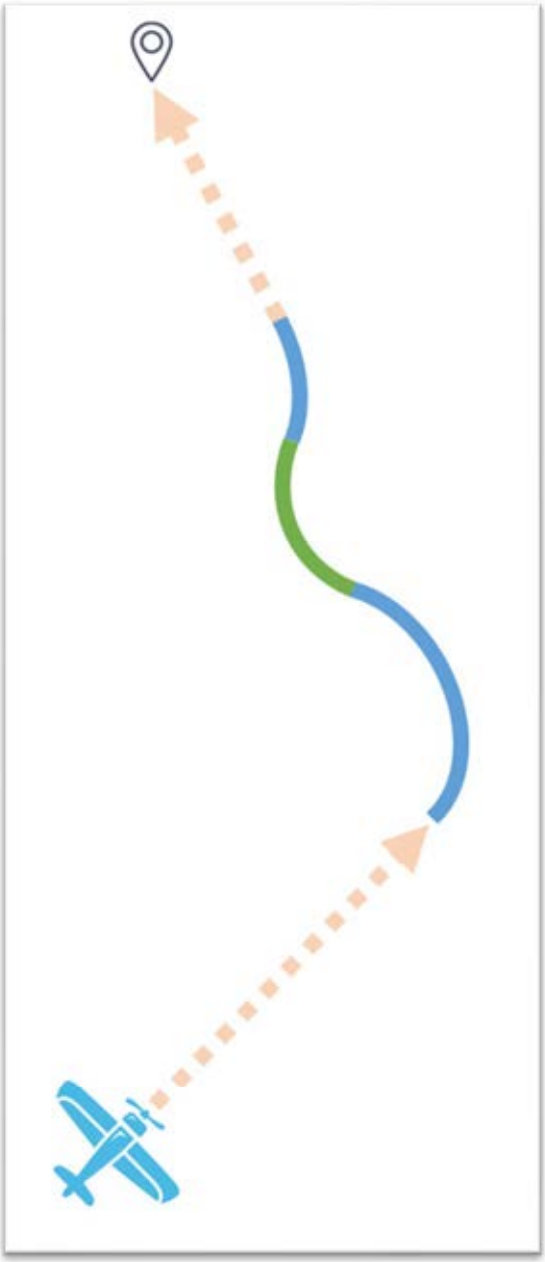


Figure 6.46: example of the vehicle's flight route adjustment.

When the vehicle wakes up and starts the take-off preparation, the "heading angle error" is forced to the fixed value "127" (in binary "0111 1111"), which results in a mechanical 0 degree heading angle error. As soon as the vehicle reaches a speed close enough to the cruise speed, the VHDL block that implements the electronics compass, and its correlated functionalities, is enabled to broadcasts the actual "heading angle error" (it is not anymore forced the fixed value corresponding to a mechanical 0-degree error). From this point, the vehicle will be enabled to adjust its route and take the correct direction.

Design assumption is to use, as trigger parameter, the speed value $v(t) = V_x = 112$. The value "112" is an 8-bit parameter (value in a range between 0 and 255), corresponding to 44% of the absolute maximum vehicle's speed.[139]

What "Figure 6.46" illustrates is a simple vehicle's flight operation, which may be used as a simplified baseline for the controller behaviour simulation and, in particular, for the route adjustment manoeuvre simulation. It is possible to observe the first take-off operation (previously described, "paragraph 6.6.1"), a series of route adjustments (it is taken merely as a simplified benchmark three route adjustments) and landing operation.

6.6.2.1 Heavy negative Heading angle error adjustment, at t=t_6

The assumptions made for the controller's behaviour simulation at the moment t=t_6 is that the vehicle's speed is close enough to the cruise speed (according to what stated before, the vehicle's speed is: $v(t) \geq V_x$) and the heading angle error is significant (grave) as for the below picture. The manoeuvre aims to change the vehicle heading angle, assuming that the vehicle direction is off course by -63.5 degrees.

The controller's input parameters chosen are compatible with the physical environmental parameters that a human pilot may observe when, just after the take-off manoeuvre, has to change the vehicle direction[140]. Those parameters are:

- "Pitch attack angle" is approximately +4.2 degrees (PITCH_angle);
- "Yaw angle" is approximately 0 degrees (YAW_angle);
- "Rolling angle" is approximately 0 degrees (ROL_angle);
- "Navigation Heading angle error" is -63.5 degrees (Compass_Error)[141];

[139] The assumption that $V_x = 112$ is subject to variations after the learning/training process.

[140] In this case, changing the vehicle's direction means a turn to the left, as for "Figure 6.47".

[141] The "Compass_error" value "0" corresponds to a heading angle error of -180°, a "Compass_error" value "127" corresponds to a heading angle error of 0-degree, a "Compass_error" value "255" corresponds to a heading angle error of +180°. A

- “Vehicle’s speed” read is 45.3% (Speed);
- “Altitude relative error” read is -15.6% (Altitude error);
- “Battery’s SoC” read is 88% (Battery_status).

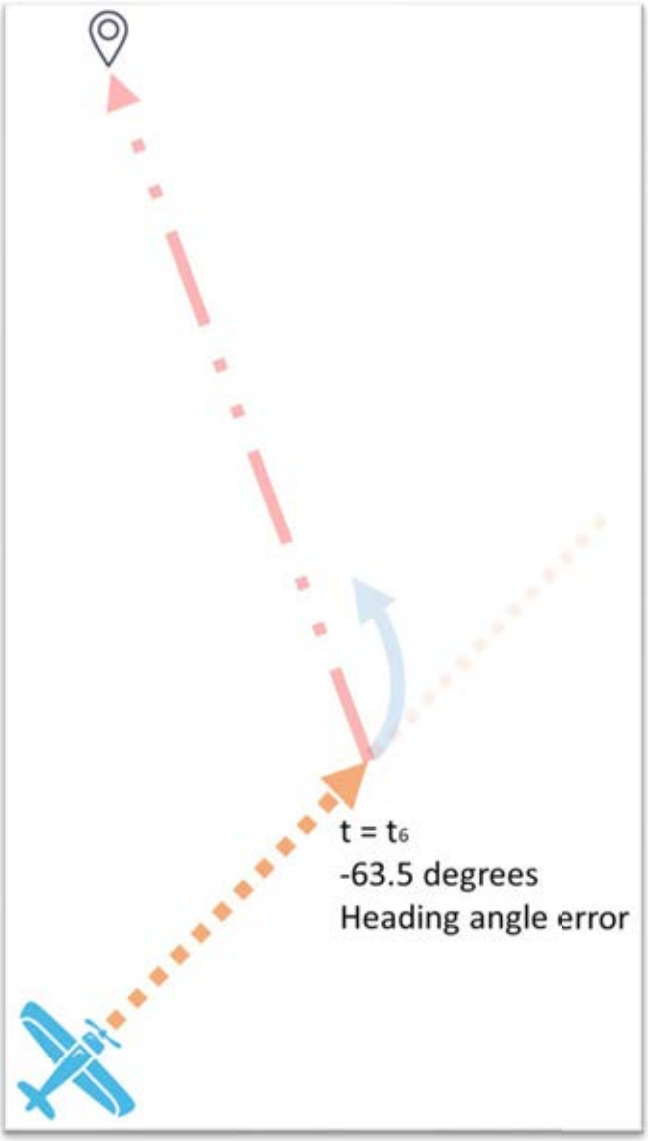


Figure 6.47: graphical representation of the flight route adjustment in $t=t_6$.

As previously described, the set of input parameters, delivered into the “XFUZZY Inference Monitor” tool, are chosen according to “Chapter 5” definition of the digital processes.

“Compass_error” input value “82” corresponds to a heading angle error in the range between -63.28° and -63.69°. By assumption, it is considered a value o -63.5°.

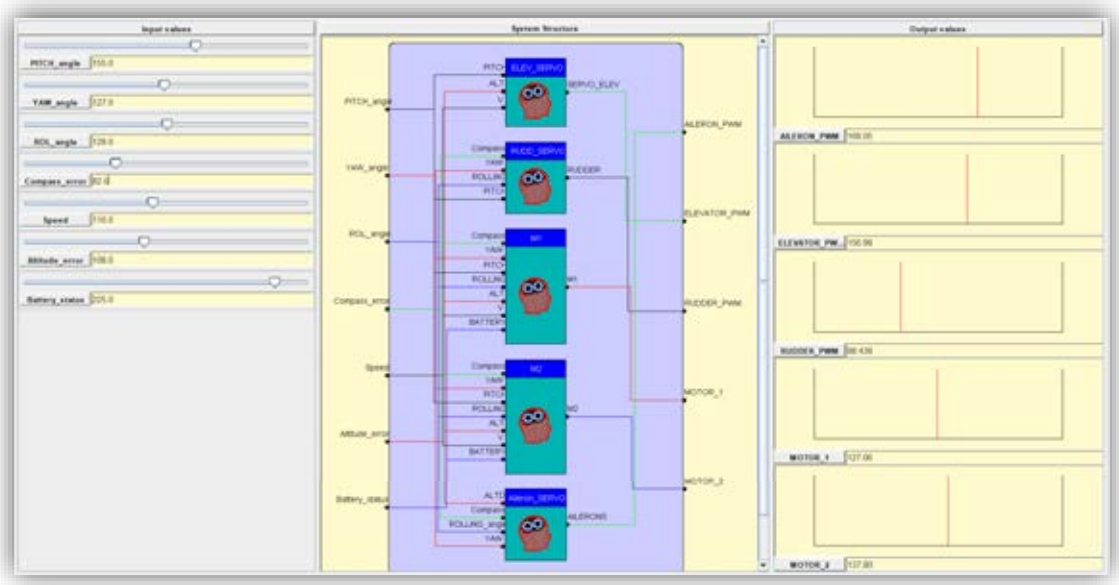


Figure 6.48: simulation's outputs at $t=t_6$.

"Figure 6.48" represents the results of the simulation taken under the conditions associated with the instant $t=t_6$. Each value present in the simulation requires a digital process to be associable with a physical value (as previously described). The mechanical outcome opportunely translated from "Figure 6.48" are:

- the mechanical angle of "Ailerons" set to approximately +6.3 degree position;
- the mechanical angle of "Elevator" set to approximately +4.5 degrees position;
- the mechanical angle of "Rudder" set to approximately -6.2 degrees position;
- left motor torque demand set to 49.6%;
- right motor torque demand set to 53.9%.

6.6.2.2 Mild Positive Heading angle error adjustment, at $t=t_7$

The assumptions made for the controller behaviour simulation at the moment $t=t_7$ is that the vehicle speed is close to the cruise speed, and a heading angle error is present as illustrated by "Figure 6.49". The manoeuvre aims to change the vehicle heading angle, assuming that the vehicle's direction is off course by +37 degrees.

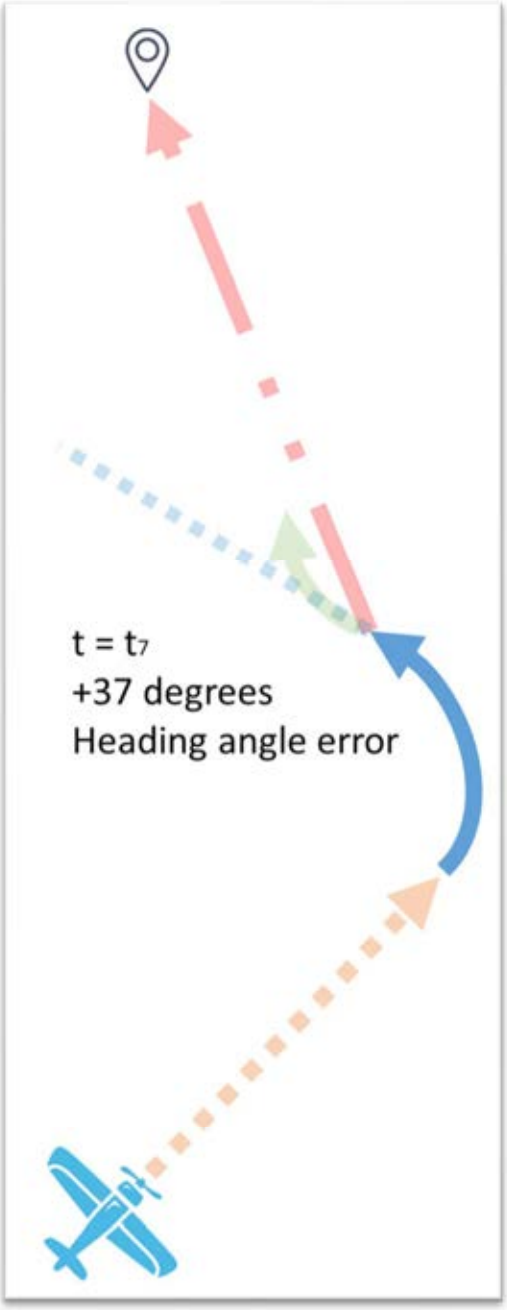


Figure 6.49: graphical representation of the flight route adjustment in $t=t_7$.

The controller's input parameters chosen are compatible with the physical environmental parameters that a human pilot may observe when changing the vehicle direction[142]. Those parameters are:

- "Pitch attack angle" is approximately +4.7 degrees (PITCH_angle);
- "Yaw angle" is approximately -2.5 degrees (YAW_angle);

[142] In this case, changing the vehicle's direction means a turn to the right, as for Figure 6.49.

- "Rolling angle" is approximately +0.8 degrees (ROL_angle);
- "Navigation Heading angle error" is +37 degrees (Compass_Error)[143];
- "Vehicle's speed" read is 46.1% (Speed);
- "Altitude relative error" read is -7% (Altitude error);
- "Battery's SoC" of 80.9% (Battery_status).

As previously described, the set of input parameters, delivered into the "XFUZZY Inference Monitor" tool, are chosen according to "Chapter 5" definition of the digital processes.

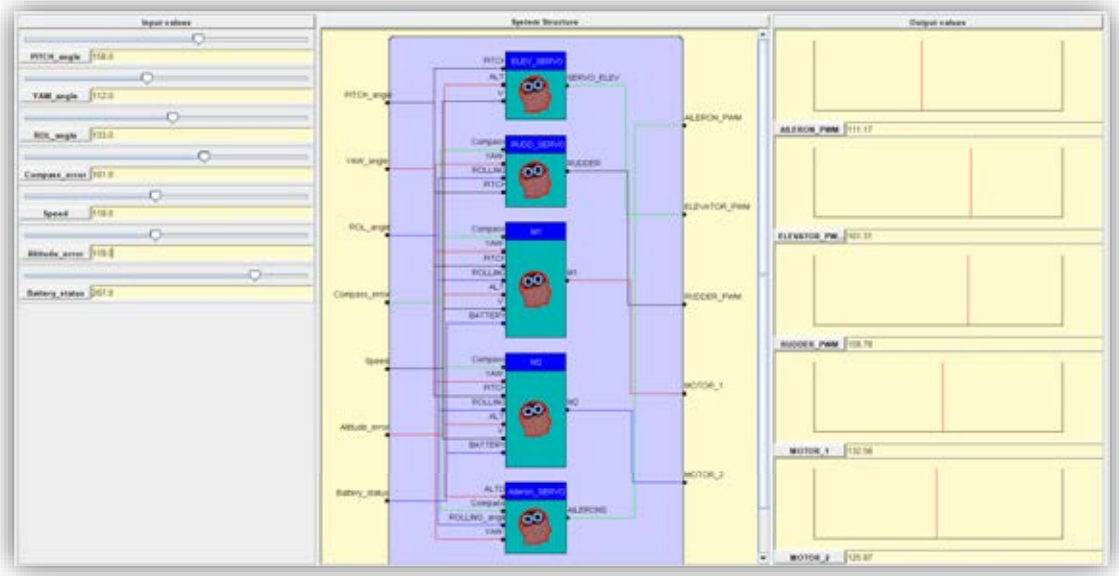


Figure 6.50: simulation's outputs at t=t_7.

"Figure 6.50" represents the results of the simulation taken under the conditions associated with the instant t=t_7. Each value present in the simulation requires a digital process to be associable with a physical value (as previously described). The mechanical outcomes opportunely translated from "Figure 6.50" are:

- the mechanical angle of "Ailerons" set to approximately -2.7 degree position;
- the mechanical angle of "Elevator" set to approximately +5.2 degrees position;
- the mechanical angle of "Rudder" set to approximately +4.8 degrees position;
- left motor torque demand set to 52%;

[143] The "Compass_error" value "0" corresponds to a heading angle error of -180°, a "Compass_error" value "127" corresponds to a heading angle error of 0-degree, a Compass_error value "255" corresponds to a heading angle error of +180°. A heading angle error of +37° corresponds to a Compass_error input value of "161".

- right motor torque demand set to 49.4%.

6.6.2.3 Moderate Negative Heading angle error adjustment, at t=t_8

The assumptions made for the controller's behaviour simulation at the moment t=t_8 is that the vehicle speed is close to the cruise speed, and a moderate heading angle error is present, as for the below picture. The manoeuvre aims to change the vehicle heading angle, assuming that the vehicle direction is off course by -20 degrees.

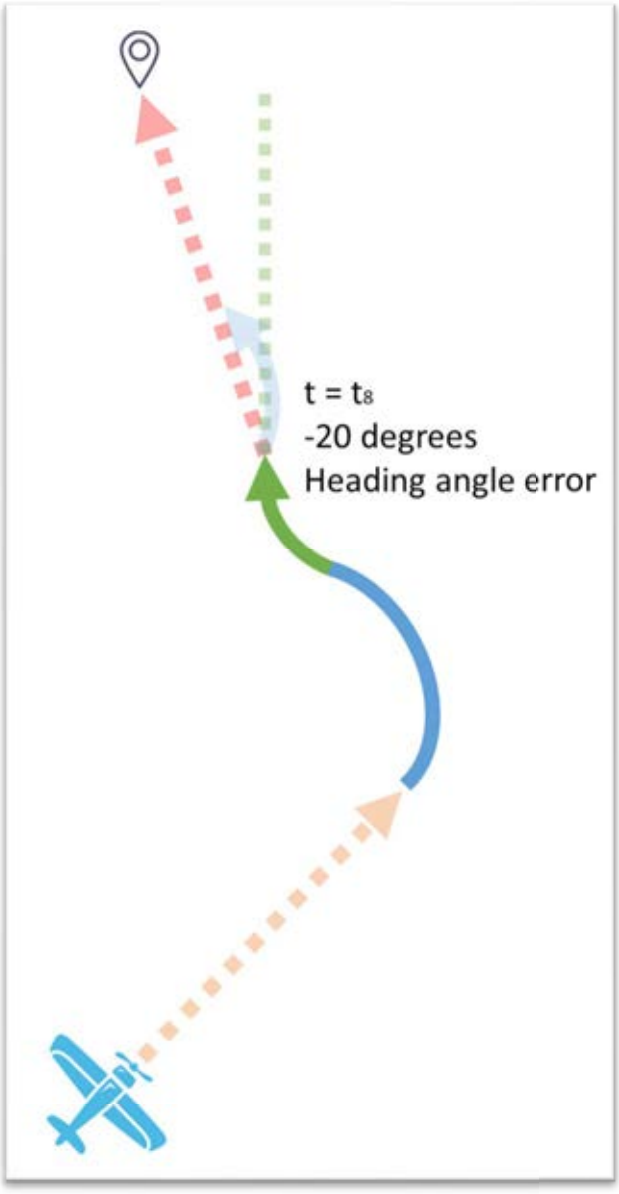


Figure 6.51: graphical representation of the flight route adjustment in t=t_8.

Input controller parameters chosen shall be compatible with the physical environmental parameters that a human pilot may observe when, just after the take-off manoeuvre, has to change the vehicle direction[144]. Those parameters are:

- "Pitch attack angle" is approximately +5 degrees (PITCH_angle);
- "Yaw angle" is approximately +1.9 degrees (YAW_angle);
- "Rolling angle" is approximately -1.6 degrees (ROL_angle);
- "Navigation Heading angle error" is -20 degrees (Compass_Error)[145];
- "Vehicle's speed" read is 46.5% (Speed);
- "Altitude relative error" read is -12.5% (Altitude error);
- "Battery's SoC" read is 75% (Battery_status).

As previously described, the set of input parameters, delivered into the "XFUZZY Inference Monitor" tool, are chosen according to "Chapter 5" definition of the digital processes.

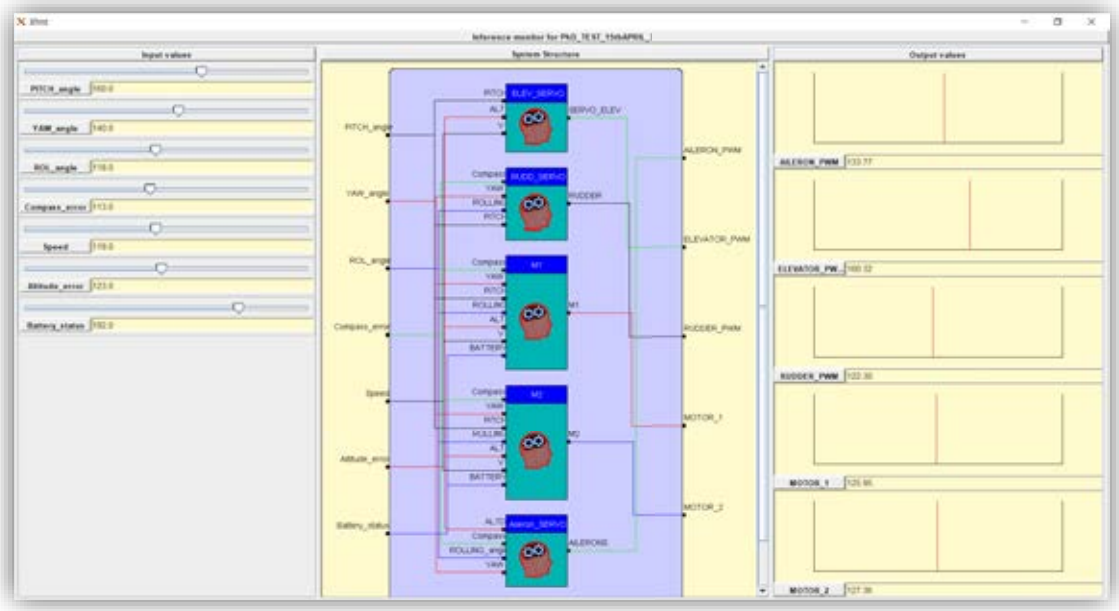


Figure 6.52: simulation's outputs at t=t_8.

"Figure 6.52" represents the results of the simulation taken under the conditions associated with the instant t=t_8. Each value present in the simulation requires a digital process

[144] In this case, changing the vehicle direction means a turn to the left.

[145] The "Compass_error" value "0" corresponds to a heading angle error of -180°, a "Compass_error" value "127" corresponds to a heading angle error of 0-degree, a "Compass_error" value "255" corresponds to a heading angle error of +180°. A heading angle error of -20° corresponds to a Compass_error input value "113".

to be associable with a physical value (as previously described). The mechanical outcomes opportunely translated from “Figure 6.52” are:

- the mechanical angle of “Ailerons” set to approximately +1.6 degree position;
- the mechanical angle of “Elevator” set to approximately +5 degrees position;
- the mechanical angle of “Rudder” set to approximately -0.9 degrees position;
- left motor torque demand set to 49.2%;
- right motor torque demand set to 49.6%.

6.6.2.4 Conclusion, route adjustment simulation

Simulations outcomes show that the controller’s decisions are compatible with the decisions that a human pilot may take in front of similar environmental conditions.

6.6.3 Steady-state simulation

The assumptions made for the controller’s behaviour simulation at the moment $t=t_9$ is that the vehicle moves at the cruise speed and the cruise altitude (or close enough). An insignificant heading angle error is present. The manoeuvre’s goal is to assure that the controller will keep the vehicle in a stable flight path under such conditions.

The simulation takes as an assumption a set of controller’s input parameters associable with a steady-state flight. The physical environmental parameters taken as baseline, which a human pilot may use as a reference, are the following:

- “Pitch attack angle” is approximately +5 degrees (PITCH_angle);
- “Yaw angle” is approximately +0.2 degrees (YAW_angle);
- “Rolling angle” is approximately -0.3 degrees (ROL_angle);
- “Navigation Heading angle error” is approximately -3.9 degrees (Compass_Error);
- “Vehicle’s speed” set to 46.9% (Speed);
- “Altitude relative error” read is +13.3% (Altitude error);
- “Battery’s SoC” read is 62.5% (Battery_status).

Such parameters are translated into an 8-bit digital form accordingly. A human pilot looks at the sensors that may appear as values in the previous format. The controller reads an 8-bit resolution digital processed input values, where for digital processing it is intended the data elaboration disserted in “Chapter 5”.

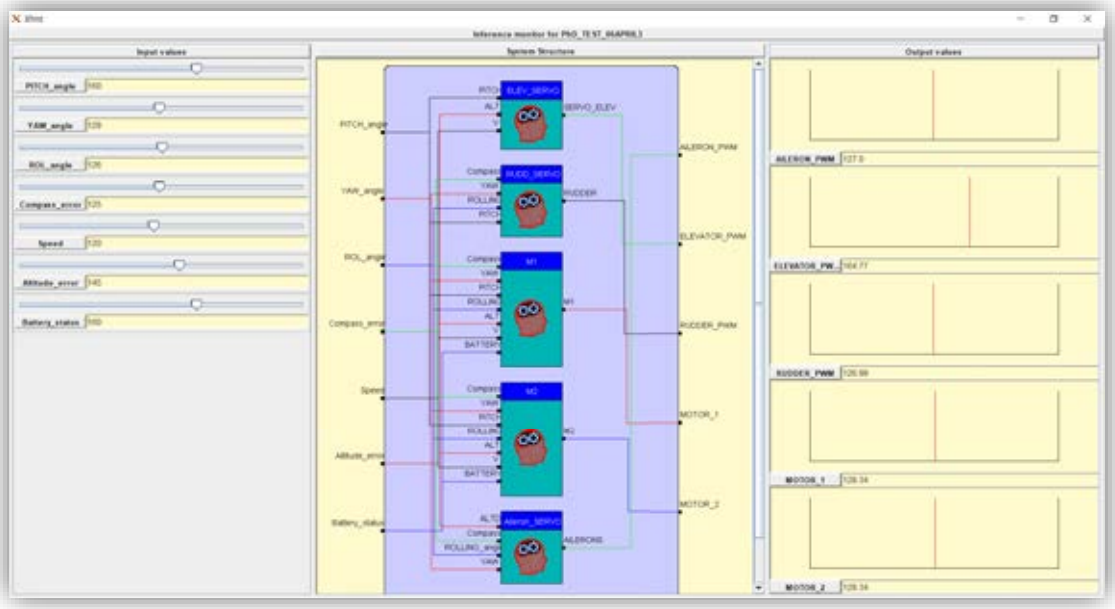


Figure 6.53: simulation's outputs, at t=t_9.

"Figure 6.53" represents the results of the simulation taken into a defined instant, where the input parameters are the previous mechanical parameters listed opportunely translated in digital form, and the output values are the fuzzy controller's output values. The "fuzzy controller" generates an 8-bit resolution digital information for each output, which shall be then translated into a PWM signal for the SERVO-Motor or into a torque demand request via an RS232 interface to the powertrain's driver interface.

The mechanical outcomes opportunely translated from "Figure 6.53" are:

- the mechanical angle of "Ailerons" set to a 0-degree position;
- the mechanical angle of "Elevator" set to approximately +5.8 degrees position;
- the mechanical angle of "Rudder" set to a 0-degree position;
- left motor torque demand set to 50.2%;
- right motor torque demand set to 50.2%.

Simulations outcomes show that the controller decisions are compatible with the decisions that a human pilot may take in similar environmental conditions.

6.6.4 Adjustment due to gusty winds simulation

The assumptions made for the controller's behaviour simulation at the moment t=t_{10} is that the vehicle is approaching the descending manoeuvre and that a gusty wind suddenly hits the vehicle. The gusty wind may introduce a consequent heading angle error. The

manoeuvre aims to observe how the controller will react to keep the vehicle in a stable flight path under such conditions.

The physical environmental parameters taken as study cases are associable with a gusty wind that impacts the vehicle from left to right. Alternatively, a gusty wind may impact the vehicle from right to left.

6.6.4.1 Case of a gusty wind that impacts on the vehicle from the right to the left

The simulation assumes a set of controller's input parameters associated with a steady-state flight perturbed by gusty winds from the right that causes a vehicle to drift to the left. This results in environmental parameters like the following:

- "Pitch attack angle" is approximately 4.1 degrees (PITCH_angle);
- "Yaw angle" is approximately -2.8 degrees (YAW_angle);
- "Rolling angle" is approximately +2.1 degrees (ROL_angle);
- "Navigation Heading angle error" is approximately +11.3 degrees (Compass_Error);
- "Vehicle's speed" read is 44.1% (Speed);
- "Altitude relative error" read is -4.7% (Altitude error);
- "Battery's SoC" read is 46.1% (Battery_status).

As previously described, the set of input parameters, delivered into the "XFUZZY Inference Monitor" tool, are chosen according to "Chapter 5" definition of the digital processes.

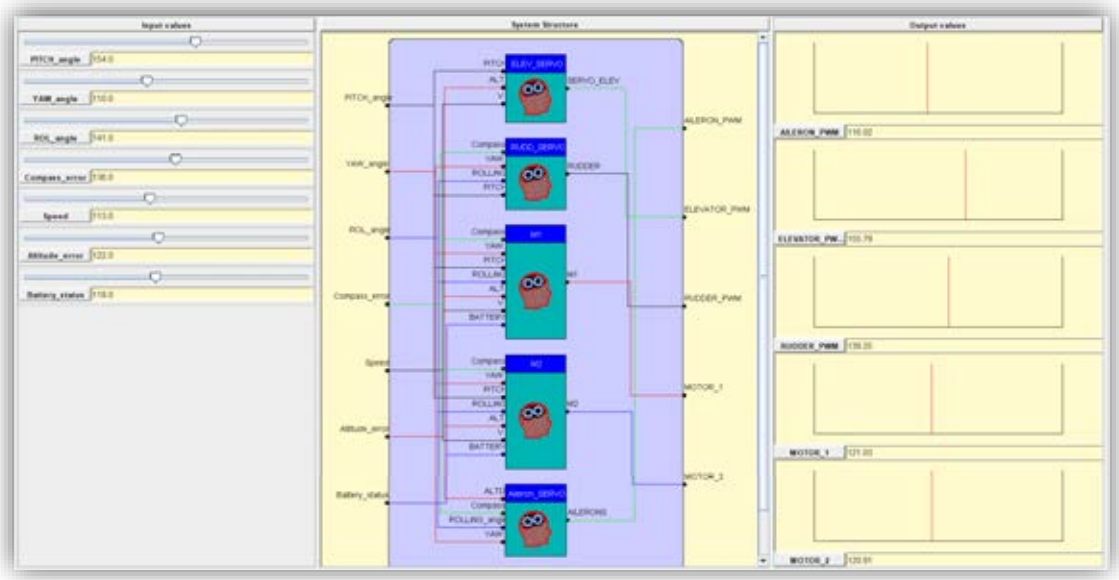


Figure 6.54: simulation's outputs at t=t_{10}.

“Figure 6.54” represents the results of the simulation taken under the conditions associated with the instant $t=t_{10}$. Every value present in the simulation requires a digital process to be associable with a physical value (as previously described). The mechanical outcomes opportunely translated from “Figure 6.54” are:

- the mechanical angle of “Ailerons” set to approximately -1.9 degree position;
- the mechanical angle of “Elevator” set to approximately +4.4 degrees position;
- the mechanical angle of “Rudder” set to approximately +1.7 degrees position;
- left motor torque demand set to 47.3%;
- right motor torque demand set to 47.3%.

6.6.4.2 Alternative case, gusty wind from the left to the right that influences the flight

Alternatively, to the previous case, the simulation takes as assumption a set of controller’s input parameters associable to a steady-state flight perturbed by gusty winds from the left that causes a vehicle drift to the right. This results in environmental parameters like the following:

- “Pitch attack angle” is approximately 4.1 degrees (PITCH_angle);
- “Yaw angle” is approximately +6 degrees (YAW_angle);
- “Rolling angle” is approximately -3.5 degrees (ROL_angle);
- “Navigation Heading angle error” is approximately -25.3 degrees (Compass_Error);
- “Vehicle’s speed” read is 45.3% (Speed);
- “Altitude relative error” read is -4.7% (Altitude error);
- Battery SoC read is 44.2% (Battery_status).

As previously described, the set of input parameters, delivered into the “XFUZZY Inference Monitor” tool, are chosen according to “Chapter 5” definition of the digital processes.

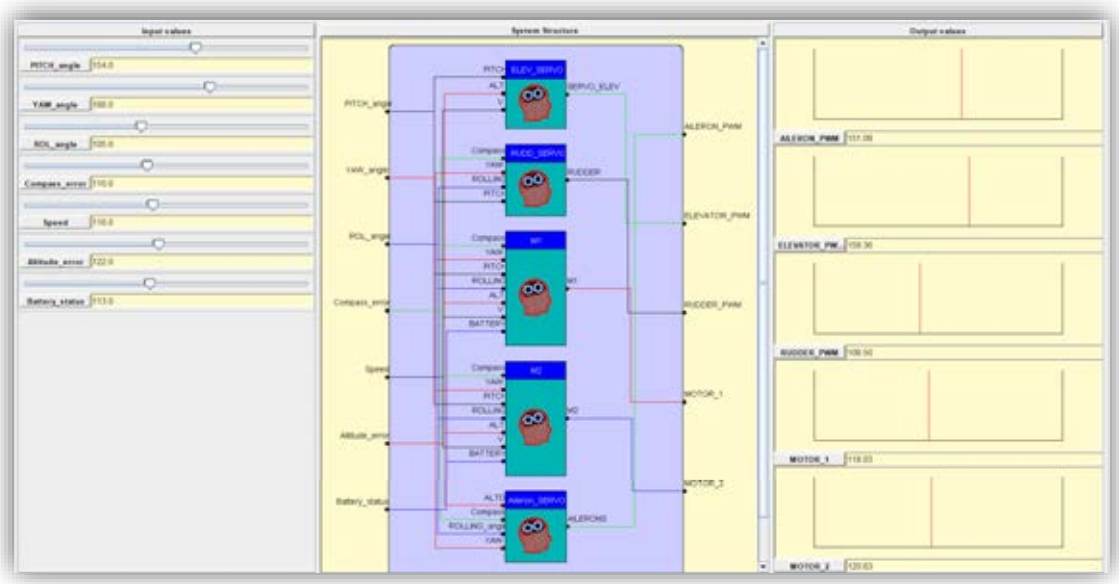


Figure 6.55: alternative simulation's outputs at $t=t_{10}$.

"Figure 6.55" represents the results of the simulation taken under the alternative conditions associated with the instant $t=t_{10}$. Each value present in the simulation requires a digital process to be associable with a physical value (as previously described). The mechanical outcomes opportunely translated from "Figure 6.55" are:

- the mechanical angle of "Ailerons" set to approximately a +3.6 degree position;
- the mechanical angle of "Elevator" set to approximately a +4.8 degrees position;
- the mechanical angle of "Rudder" set to approximately a -3 degrees position;
- left motor torque demand set to 46.1%;
- right motor torque demand set to 47.3%.

6.6.4.3 "Controller Behavior" under gusty wind conclusions

Simulations outcomes show that the controller decisions are compatible with the decisions that a human pilot may take in similar environmental conditions.

6.6.5 Landing Simulation

Vehicle's landing simulation results being a challenging manoeuvre to simulate without a very detailed and comprehensive model of the vehicle's dynamic behavioural[146]; it results

[146] Parameters like the aero-dynamical coefficient, vehicle acceleration behaviour, stall speed, take-off speed and etc., have paramount importance for very detailed simulation.

evident after the study of "Chapter 2". The assumption made is to identify several manoeuvre points[147] and check the controller behaviour in those specific moments (similar to the strategy used to simulate the take-off manoeuvre).

For the proposed work, the landing simulation targets the controller's behaviour in six key points of the manoeuvre. The first step is the landing approach[148] and, then the descending manoeuvre will begin. The conclusion of the descending manoeuvre is associated with the vehicle touch-down with the ground. Between the descending manoeuvre start and its conclusion (in this case is the so-called touch-down), there are two more crucial points. The first one is associated with the start of the vehicle's deceleration, where a pitch manoeuvre increases drag and decelerate the aircraft (usually achieves this during the flare portion of the approach, ideally to a minimum flying speed). The other crucial point is when the powertrain motors are disabled, it happens before the touch-down (according to the theory elaborated in "paragraph 5.1.4).

The "touch-down velocity" (V_{TD}), ideally, is as close as possible to the stall speed (slightly superior) of the aircraft in landing configuration. The deceleration on the landing roll from V_{TD} to V_0 could be accomplished by braking and reverse thrust, but the system assumption made is to leave the vehicle passively decelerate, keeping the powertrain disabled. Looking at "Figure 6.56", it is possible to identify such moments as:

a) the vehicle is approaching the landing manoeuvre, and it is moving with speed $v(t) = V_{FC}$, (t=t_{11}, "Figure 6.56");
b) the vehicle begins the descending manoeuvre (t=t_{12}, "Figure 6.56"), and it is moving with speed $v(t) = V_{SD}$;
c) the vehicle is executing the descending manoeuvre with speed $v(t) = V_D$, (t=t_{13}, "Figure 6.56");
d) the vehicle is executing a deceleration[149] before disengaging the powertrain, and it is moving with speed $v(t) = V_{FD}$, (t = t_{14}, "Figure 6.56");
e) the vehicle is proxy to the touch-down, it is moving with speed $v(t) = V_{TD}$, where $(1.1 \cdot V_{stall}) \leq V_{TD} \leq (V_{stall} \cdot \sqrt{2})$, (t = t_{15}, "Figure 6.56").

[147] Obviously, as many points may be identified as more reliable will be the outcome of the raw simulation.
[148] Vehicle's alignment to the target at a sustainable speed. At this point, the assumption is that the vehicle is moving with speed close enough to the vehicle's optimal cruise speed.
[149] It is expected a pitch manoeuvre during the flare portion of the approach which will increase drag and decelerate the aircraft to minimum flying speed.

f) End of manoeuvre, $v(t) = 0$, (t = t_{16}, Figure 6.56).

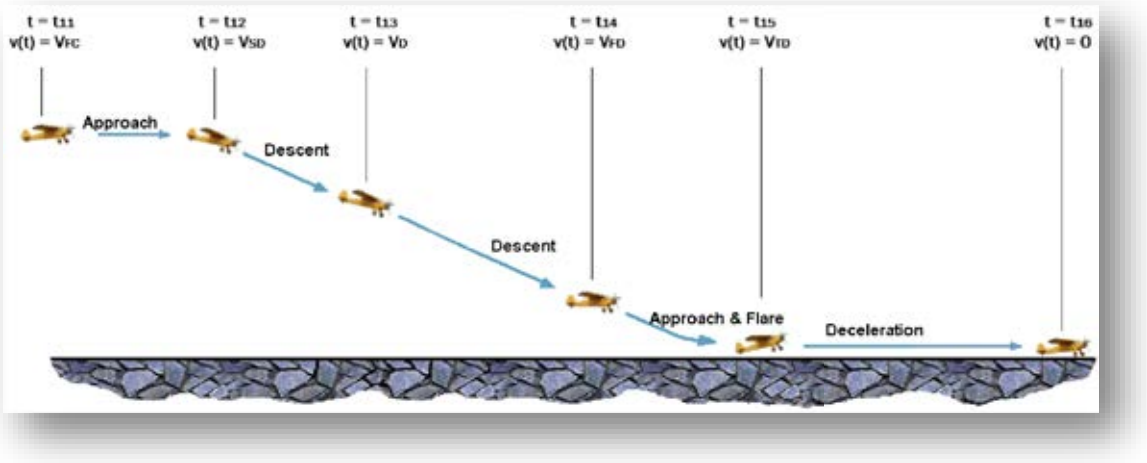


Figure 6.56: landing manoeuvre, graphical animation.

6.6.5.1 Preliminary Landing Approach, simulation at t=t_{11}

The landing manoeuvre starts at "t = t_{11}" when the vehicle performs the final alignment and begins to engage the descending approach. By assumption, are used a set of controller's input parameters compatible with the physical environmental parameters that a human pilot may observe just before starting the landing manoeuvre. Those parameters are:

- "Pitch attack angle" is approximately +3.8 degrees (PITCH_angle);
- "Yaw angle" is approximately -0.9 degrees (YAW_angle);
- "Rolling angle" is approximately +0.6 degrees (ROL_angle);
- "Navigation Heading angle error" is approximately +12.7 degrees (Compass_Error);
- "Vehicle's speed" read is 43.4% (Speed);
- "Altitude relative error" read is +39.1% (Altitude error);
- "Battery's SoC" read is 41.1% (Battery_status).

Those parameters are translated into an 8-bit digital form accordingly. A human pilot looks at the sensors that may appear as values in the previous format. The controller reads 8-bit resolution digital processed input values, where for digital processing it is meant the data elaboration disserted in "Chapter 5". It means that the set of input parameters, delivered into the "XFUZZY Inference Monitor" tool, are chosen according to the "Chapter 5" definition of the digital processes.

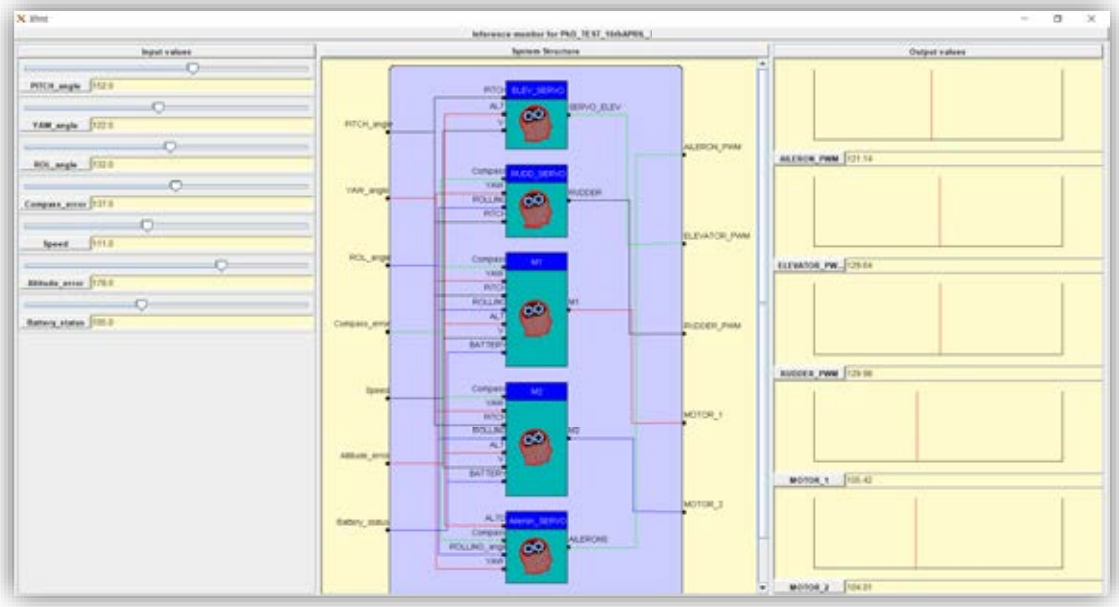


Figure 6.57: simulation's outputs at t=t_{11}.

"Figure 6.57" represents the simulation results taken into a defined instant, where the input parameters are the previous mechanical parameters listed opportunely translated in digital form, and the output values are the fuzzy controller output values. According to the definitions of "Chapter 5", those output values are digitally processed before becoming the physical control signals broadcasted by the FPGA. The "fuzzy controller" generates an 8-bit resolution digital information format for each output, which is translated into a PWM signal for the SERVO-Motor or into a torque demand request via RS232 to the powertrain's E-Motor driver interface.

The mechanical outcome opportunely translated from "Figure 6.57" are:

- the mechanical angle of "Ailerons" set to approximately -1.1 degree position;
- the mechanical angle of "Elevator" set to approximately +0.3 degrees position;
- the mechanical angle of "Rudder" set to approximately +0.3 degree position.
- left motor torque demand set to 41%;
- right motor torque demand set to 40.6%.

6.6.5.2 Landing Descending Approach, simulation at t=t_{12}

When the landing manoeuvre begins, the vehicle shall start to change the pitch attack angle; it is assumed to observe a null pitch angle at "t = t_{12}", and the vehicle may be still performing the final alignment to the target. By assumption, are used a set of controller input

parameters compatible with the physical environmental parameters that a human pilot may observe at the beginning of the landing manoeuvre. Those parameters are:

- “Pitch attack angle” is approximately 0-degree (PITCH_angle);
- “Yaw angle” is approximately +0.3 degrees (YAW_angle);
- “Rolling angle” is approximately -0.9 degrees (ROL_angle);
- “Navigation Heading angle error” is approximately +3 degrees (Compass_Error);
- “Vehicle’s speed” read is 47.3% (Speed);
- “Altitude relative error” read is +43.8% (Altitude error);
- “Battery’s SoC” read is of 39.9% (Battery_status).

As previously described, the set of input parameters, delivered into the “XFUZZY Inference Monitor” tool, are chosen according to “Chapter 5” definition of the digital processes.

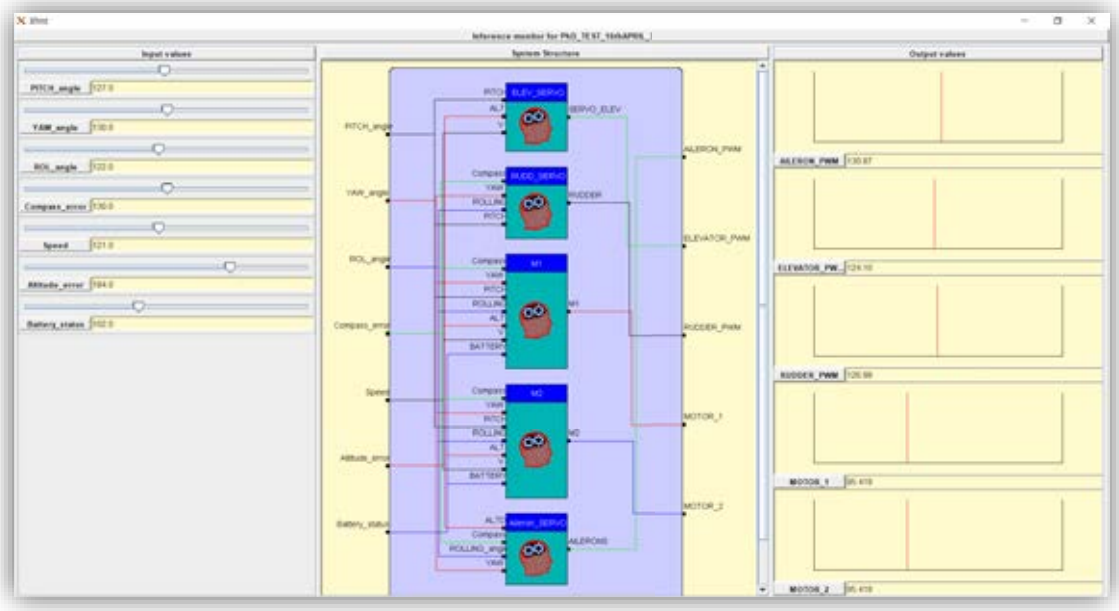


Figure 6.58: simulation’s outputs at $t=t_{12}$.

“Figure 6.58” represents the result of the simulation taken under the alternative conditions associated with the instant $t=t_{12}$. Each value present in the simulation requires a digital process to be associable with a physical value (as previously described).

The mechanical outcome opportunely translated from “Figure 6.58” are:

- the mechanical angle of “Ailerons” set to approximately +0.5 degree position;
- the mechanical angle of “Elevator” set to approximately -0.6 degrees position;

- the mechanical angle of “Rudder” set to a 0-degree position;
- left motor torque demand set to 37.1%;
- right motor torque demand set to 37.1%.

6.6.5.3 Descending simulation at $t=t_{13}$

In the middle of the landing manoeuvre, the vehicle is full descending and most likely with a negative pitch attack angle, which might accentuate a vehicle speed increase. For the study case “$t = t_{13}$”, the vehicle is most likely aligned to the target, under acceleration (this behaviour may be typical for RC plane manoeuvres due to their extreme lightweight), and the vehicle might start manoeuvres[150] to reduce the speed. By assumption, are used a set of controller’s input parameters compatible with the physical environmental parameters that a human pilot may observe when during the vehicle descent has to reduce the vehicle’s speed. Those parameters are:

- “Pitch attack angle” is approximately -2.6 degrees (PITCH_angle);
- “Yaw angle” is approximately -0.7 degrees (YAW_angle);
- “Rolling angle” is approximately +0.4 degrees (ROL_angle);
- “Navigation Heading angle error” is approximately -2.8 degrees (Compass_Error);
- “Vehicle’s speed” read is +53.9% (Speed);
- “Altitude relative error” read is +39.1% (Altitude error);
- “Battery’s SoC” read is 36.8% (Battery_status).

As previously described, the set of input parameters delivered into the XFUZZY Inference Monitor tool are chosen according to “Chapter 5” definition of the digital processes.

“Figure 6.59” represents the results of the simulation taken under the alternative conditions associated with the instant $t=t_{13}$. Each value present in the simulation requires a digital process to be associable with a physical value (as previously described).

The mechanical outcome opportunely translated from “Figure 6.59” are:

- the mechanical angle of “Ailerons” set to a 0-degree position;
- the mechanical angle of “Elevator” set approximately to +2.8 degrees position;

[150] By presumption, it is the point where the vehicle needs to change the pitch angle sign, from negative to positive.

- the mechanical angle of “Rudder” set to a 0-degree position;
- left motor torque demand set to 34.4%;
- right motor torque demand set to 34.4%.

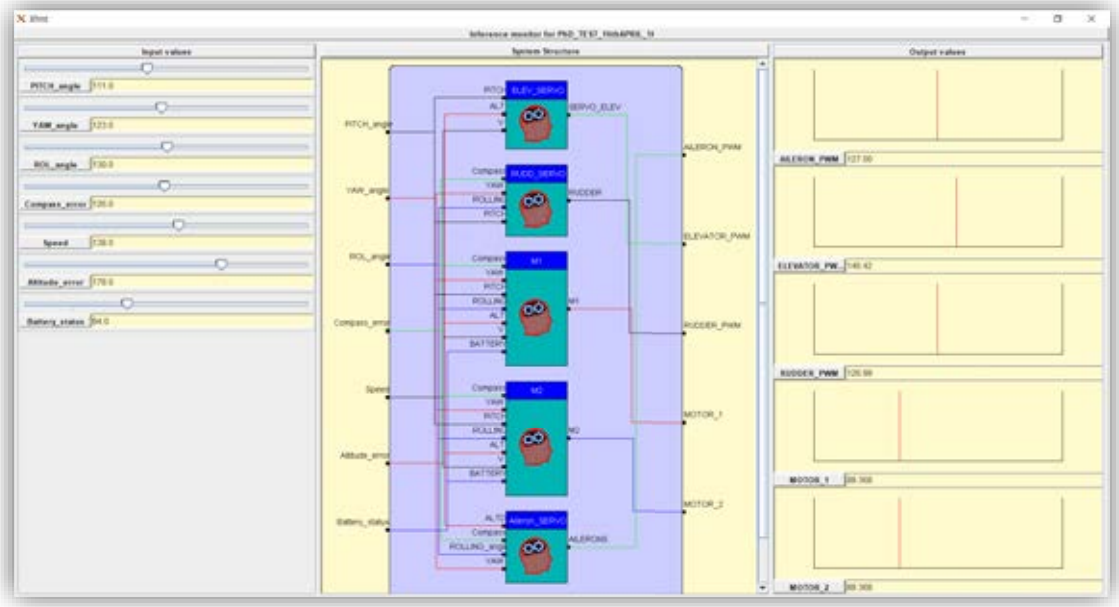


Figure 6.59: simulation’s outputs at t=t_{13}.

6.6.5.4 Descending simulation at t=t_{14}

When the vehicle is approaching the end of its descent, it most likely has a positive pitch attach angle, and it is decelerating. A pitch manoeuvre is expected during the flare portion of the approach, which increases the vehicle’s drag and decelerates the aircraft (up to the touch-down velocity).

For the study case “t = t_{14}”, the vehicle is aligned to the target within an acceptable error, and it is close to detect the “ground” with the proximity sensor. According to the “Chapter 5” avowals, after that the take-off is performed, as soon as the proximity sensor detects the ground, both E-Motors (left and right) will be disabled. The vehicle’s altitude is slightly above the detection altitude (proximity sensors did not detect the ground yet), and E-Motors are still active.

By assumption, are used a set of controller’s input parameters compatible with the physical environmental parameters that a human pilot may observe when the vehicle approaches the end of the descent. Those parameters are:

- “Pitch attack angle” is approximately +1.3 degrees (PITCH_angle);
- “Yaw angle” is approximately -0.5 degrees (YAW_angle);

- “Rolling angle” is approximately +0.2 degrees (ROL_angle);
- “Navigation Heading angle error” is approximately -2.8 degrees (Compass_Error);
- “Vehicle’s speed” read is 43.4% (Speed);
- “Altitude relative error” read is -16.5% (Altitude error);
- “Battery’s SoC” read is 35.5%.

As previously described, the set of input parameters delivered into the XFUZZY Inference Monitor tool are chosen according to “Chapter 5” definition of the digital processes.

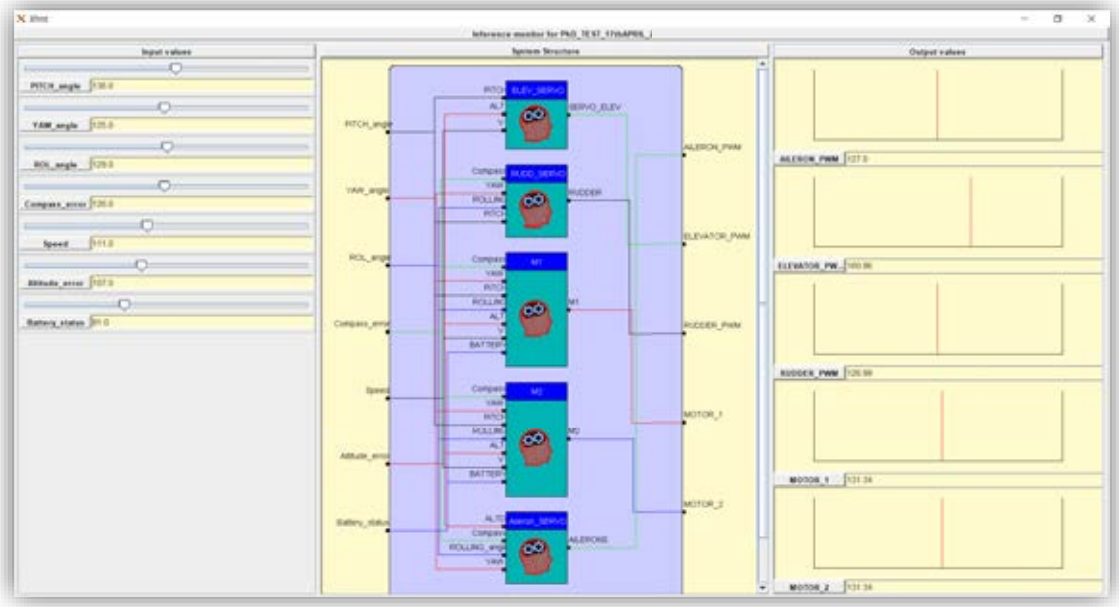


Figure 6.60: simulation’s outputs at t=t_{14}.

“Figure 6.60” represents the results of the simulation taken under the alternative conditions associated with the instant t=t_{14}. Each value present in the simulation requires a digital process to be associable to a physical value (as previously described).

The mechanical outcome opportunely translated from “Figure 6.60” are:

- the mechanical angle of “Ailerons” set to a 0-degree position;
- the mechanical angle of “Elevator” set approximately to +5.2 degrees position;
- the mechanical angle of “Rudder” set to a 0-degree position.
- left motor torque demand set to 51.2%;
- right motor torque demand set to 51.2%.

6.6.5.5 Descending simulation at $t=t_{15}$

As soon as the vehicle detects the proximity of the ground, both motors are disabled. A pitch manoeuvre is still expected during the flare portion of the final approach to the ground, which increases drag and decelerate the aircraft to minimum flying speed.

For the study case "$t = t_{15}$", the vehicle is aligned to the target within an acceptable error, and it is in the "touch-down" proximity. The vehicle's altitude is below the ground detection altitude (as "Figure 6.56" illustrates) but higher than "0m".

By assumption, are used a set of controller's input parameters compatible with the physical environmental parameters that a human pilot may observe a moment before touching the ground. Those parameters are:

- "Pitch attack angle" is approximately +4.3 degrees (PITCH_angle);
- "Yaw angle" is approximately -0.5 degrees (YAW_angle);
- "Rolling angle" is approximately +0.2 degrees (ROL_angle);
- "Navigation Heading angle error" is approximately -2.8 degrees (Compass_Error);
- "Vehicle"s speed" read is +36.8% (Speed);
- "Altitude relative error" read is -28.9% (Altitude error);
- "Battery's SoC" read is 34.4%.

As previously described, the set of input parameters delivered into the XFUZZY Inference Monitor tool are chosen according to "Chapter 5" definition of the digital processes.

"Figure 6.61" represents the results of the simulation taken under the alternative conditions associated with the instant $t=t_{15}$. Each value present in the simulation requires a digital process to be associable to a physical value (as previously described).

The mechanical outcome opportunely translated from "Figure 6.61" are:

- the mechanical angle of "Ailerons" set to a 0-degree position;
- the mechanical angle of "Elevator" set approximately to +8.2 degrees position;
- the mechanical angle of "Rudder" set to a 0-degree position.
- although left motor torque demand set to 51.6%, **Motor is disabled**;
- although right motor torque demand set to 51.6%, **Motor is disabled**.

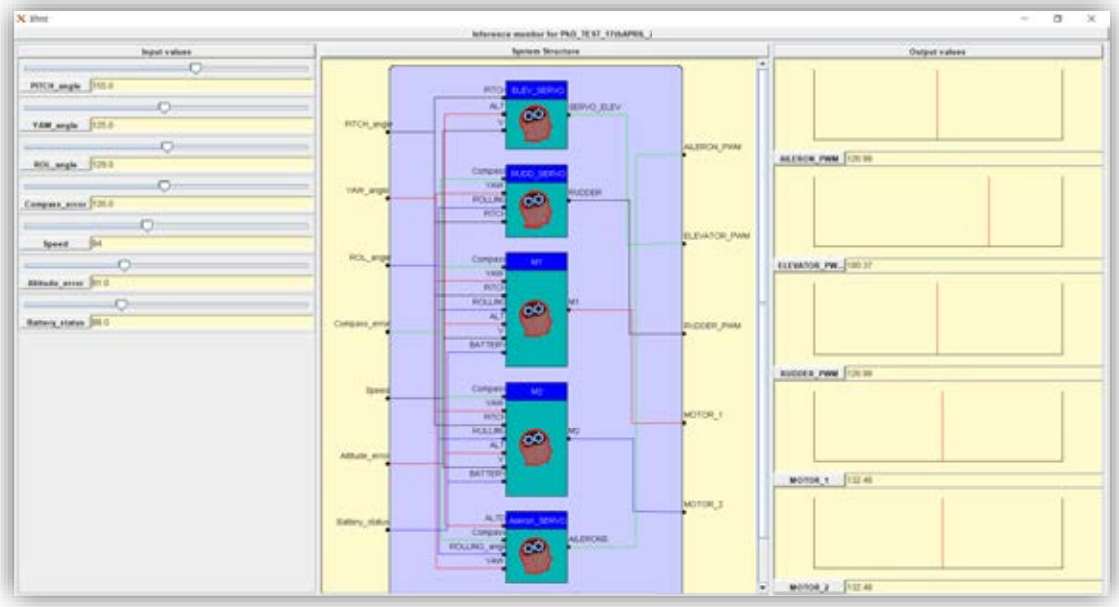


Figure 6.61: simulation's outputs at t=t_{15}.

6.6.5.6 Descending simulation at t=t_{16}

After that, the vehicle touches the ground, is achievable a very limited controllability of the vehicle since it is left passively decelerating until the "End of Manoeuvre", which is associated with $v(t) = 0$, (t = t_{16}).

6.6.5.7 Landing simulation, Conclusions

Moving by the outcome of the previous simulation and using as reference the landing paths of below "Figure 6.62", it is feasible to state that the controller most likely will replay a human pilot behaviour while performing the landing manoeuvre[151].

In fact, at t=t_{11} controller begins to slowly move the vehicle's nose down, decreasing the powertrain power demand and then slowly decreasing the elevator's attack angle. At t=t_{12}, the vehicle reaches a null pitch attach angle, and the controller continues to decrease the pitch attach angle and the power demand for the motors. At t=t_{13}, the vehicle has a negative pitch angle and is accelerating, and the controller reacts, continuing to decrease the motors' power and moving up the elevator. At t=t_{14}, the vehicle's pitch is going up, and the speed is decreasing; the controller's reaction is to continue increasing the elevator attack angle and

[151] It is essential to highlight that the environmental conditions of the simulations are not taking into account heavy meteorological interference.

increase the motor power demand because the vehicle crossed the reference altitude. As soon as the proximity sensor detects the ground, according to "Chapter 5" design system requirements, both motors are disabled, and the vehicle is moving passively. The controllability of the vehicle after t=t_{14} is limited to the SERVO-Motors. Between t_{14} and $t_{15,}$ the controller performs a pitch manoeuvre during the flare portion of the approach, which increases drag and decelerates the aircraft. As soon as the vehicle hits the ground in $t_{15,}$ the vehicle passively decelerates until it reaches the terminal speed $v(t) = 0$ in t = t_{16}.

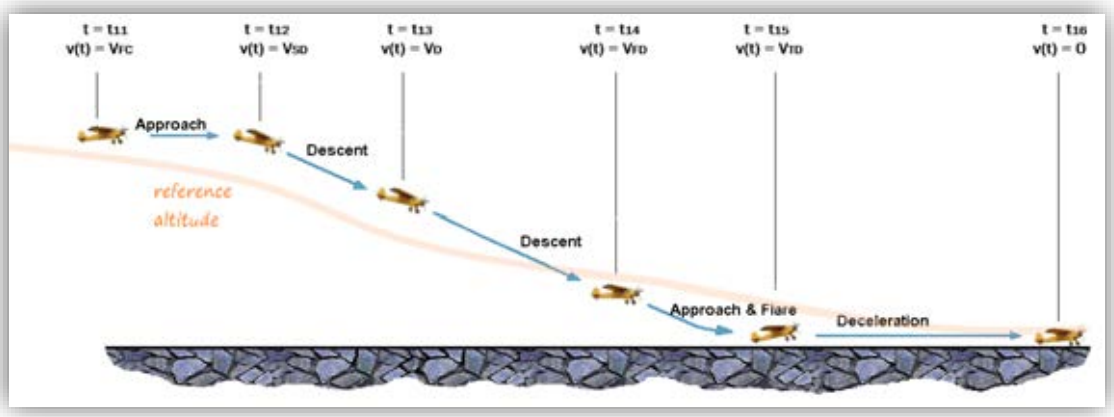


Figure 6.62: reference landing manoeuvre altitude against a vehicle's manoeuvre position, graphical animation.

6.7 Controller

Previous paragraphs have paramount importance for the physical preliminary controller design (the controller's algorithm core of the learning/training process[152]). The physical VHDL controller design has a multi-layer structure, associable to any hierarchical hardware schematics design.

The "Top Layer" describes the peripherals interfaces and the interaction between the peripherals with the flight controller's core. A typical VHDL schematics RTL view, generated after the algorism's synthesis, allows the user to observe the whole system as an array of blocks connected each other with data bus and independent digital signals. "Figure 6.63" exposes the RTL view of the complete system's VHDL algorithm, generated after the main algorithm's synthesis (through the SW "Synplify Pro").

[152] The definition of a correct set of preliminary parameters capable of performing the essential tasks is beneficial for a reliable learning/training process.

The consequent step is the VHDL code population for each block. Since the previous paragraphs describe the peripherals interface blocks, the focus moves on the core of the flight controller, which is the VHDL component called by the Author “NEURAL” (VHDL component instance: “N0”). The VHDL component “NEURAL” code represents a second hierarchical layer of the algorithm’s structure and, as “Figure 6.64” illustrates, is divided into a few sub-components:

a) “Digital_Processing” (it accordingly computes the input data in order to establish the correct digital connection between the heart of the controller and its peripherals);
b) “Fuzzy” (it contains the neuro-fuzzy flight controller);
c) “Neural_Block_StateMachine” (it accordingly coordinates the system’s peripherals with the heart of the controller);
d) “Neural_enable” (it manages the system’s safety routines and peripheral’s enables).

The first two components merit an exhaustive analysis due to their algorithms’ complexity, while the last two components scrutiny results irrelevant for the dissertation.

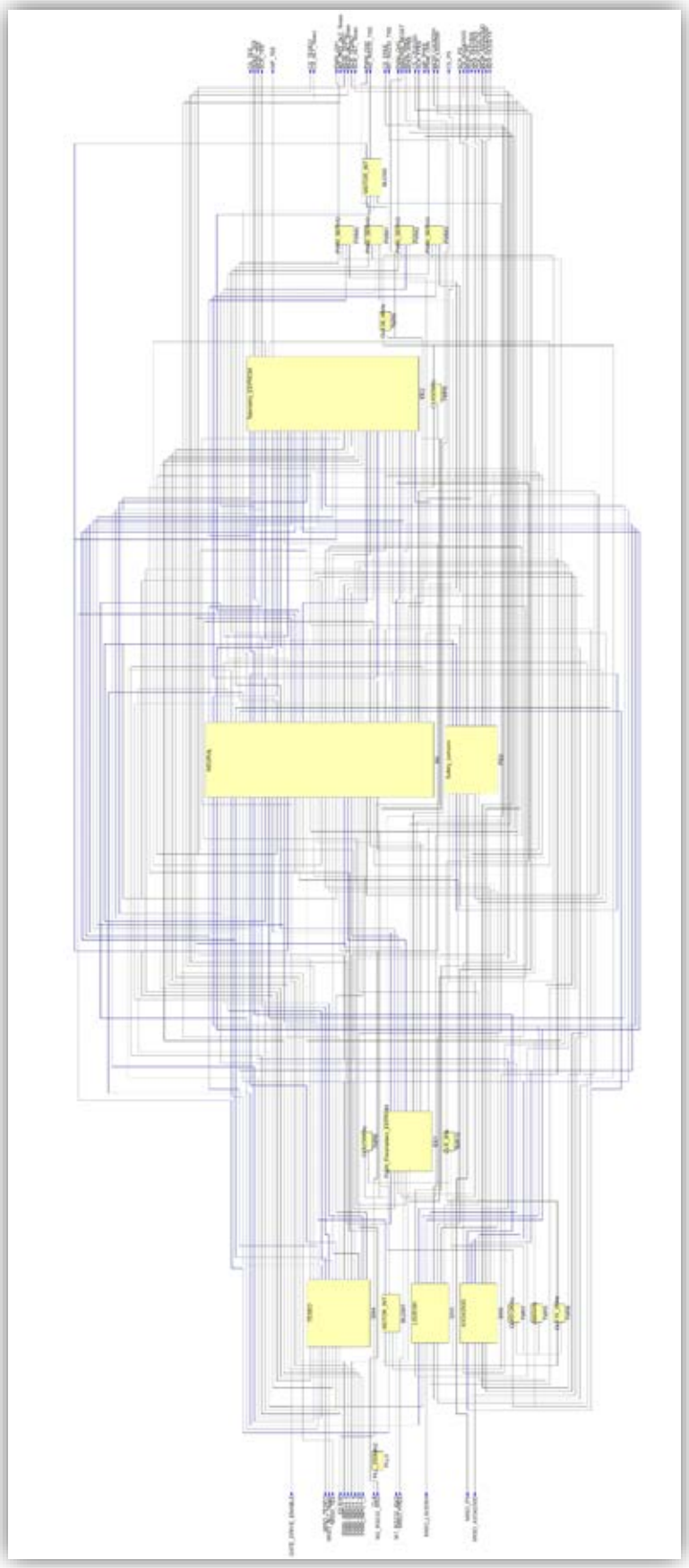


Figure 6.63: RTL view, the outcome of the main VHDL algorithm's synthesis.

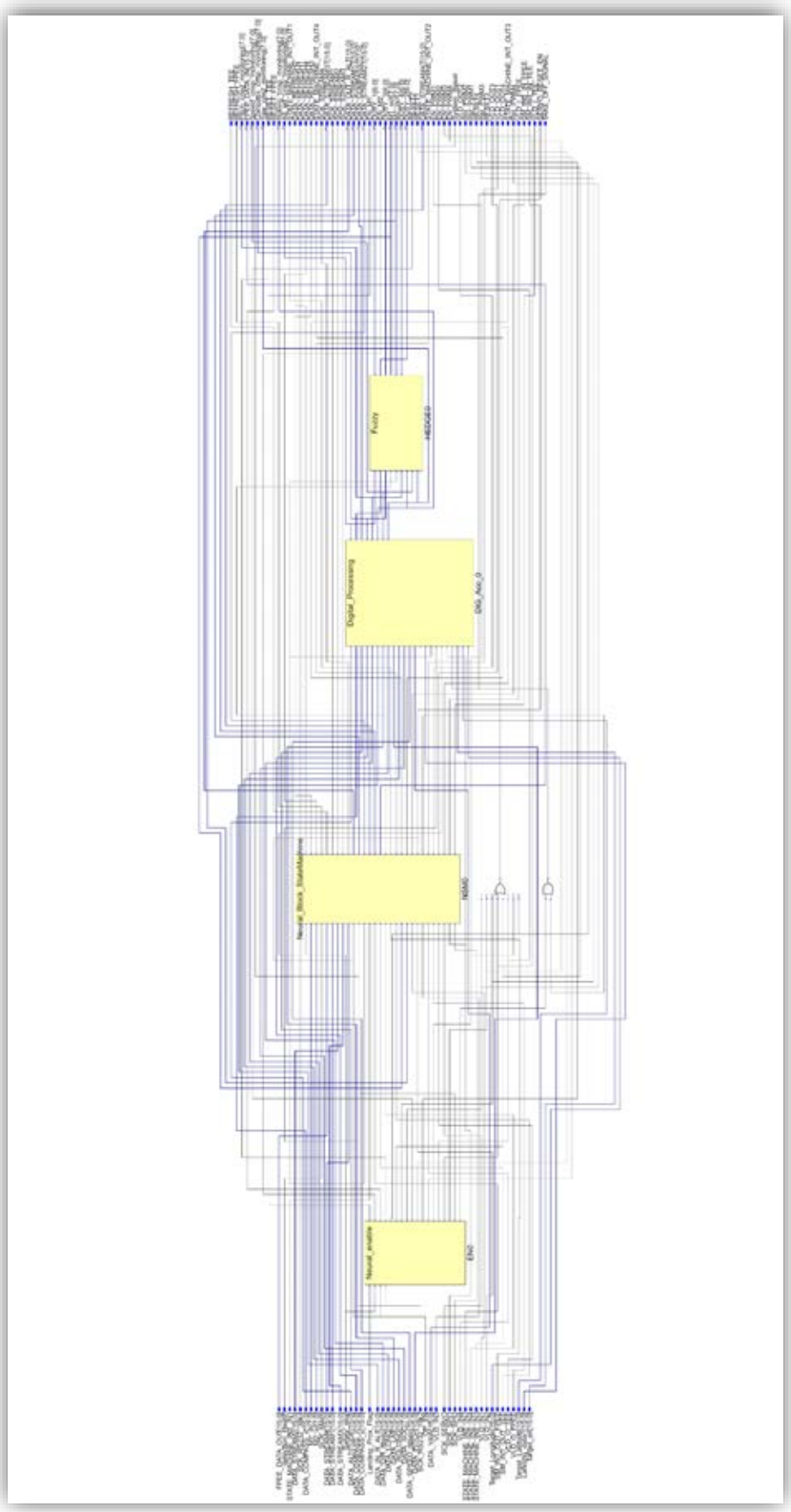


Figure 6.64: RTL view, the outcome of the "NEURAL" VHDL component algorithm's synthesis.

6.7.1 Digital_Processing VHDL component

The VHDL component "Digital_Processing" receives a large amount of raw data from the vehicle's sensors (from the VHDL components defined in "paragraph 6.1"). The raw data to be utilizable require an appropriate manipulation of the information. "Figure 6.65" expresses the RTL view of the VHDL component "Digital_Processing"[153]. Eight bespoke VHDL components, listed below, implement this operation.

a) ANGLE_input;
b) SPEED_input;
c) ALTITUDE_input;
d) ENERGY_input;
e) ROLLING_input;
f) YAW_input;
g) PITCH_input;
h) COMPASS_input.

The VHDL code population of these blocks embodies a third hierarchical layer of the physical controller structure.

[153] Obviously, the RTL view is an outcome of the "Digital_Processing" VHDL component algorithm's synthesis, which is encased within the main algorithm. Thus it is an operation subsequent to the synthesis of the system's entire algorithm.

Figure 6.65: RTL view, the outcome of the "Digital_Processing" VHDL component algorithm's synthesis.

6.7.1.1 "ANGLE_input", VHDL component

The VHDL component "ANGLE_input" receives an array of multiple 16-bit independent information from the VHDL components: "A3G4250D", "LIS3DSH", and "TESEO". It also receives an array of logic signals: enable, primary clock, secondary clock and take-off enable. "Compass_angle" is the component's 8-bit output, which will be used by "Telemetry EEPROM" and by a few more sub-components within the "Digital_Processing" VHDL component. The output expresses the vehicle's heading angle.

Component's safety mechanisms analysis are irrelevant for the dissertation.

6.7.1.2 "SPEED_input", VDHL component

The VHDL component "SPEED_input" receives a 16-bit "DATA_IN" input information from the VHDL component "TESEO" and an array of logic signals (primary clock, secondary clock, operation enable, and take-off enable). Component's algorithm generates an 8-bit "Speed" output value in a form utilizable by the neuro-fuzzy controller.

In fact, the 16-bit input information expresses an absolute measure of the vehicle's speed in meter per second, while the neuro-fuzzy controller needs 8-bit information that expresses a relative vehicle's speed. For relative vehicle speed, it is intended that given a scale between 0 and 255 (8-bit resolution), the vehicle's maximum allowed speed would correspond to "255" while the vehicle's null speed will correspond to "0". It allows expressing the vehicle's speed in percentage in order to adapt the control algorithm to other vehicles efficiently.

Component's safety mechanisms analysis are irrelevant for the dissertation.

6.7.1.3 "ALTITUDE_input", VHDL component

The VHDL component "ALTITUDE_input" receives an array of multiple 16-bit independent information from the VHDL components: "COMPASS_input", "Flight_Parameters_EEPROM" and "TESEO". It also receives an array of logic signals: enable, primary clock, secondary clock and take-off enable. Component's algorithm generates an 8-bit "Altitude_error" output value in a form utilizable by the neuro-fuzzy controller.

The first component's algorithm operation is the calculation of a weighted average between the 16-bit information obtained by the GPS module (from VHDL component "Process_Teseo") with the 16-bit information obtained by the redundant altimeter (from VHDL component "LPS25HB"), according to the following equation:

$$Altitude_{read} = \frac{35 \cdot Altitude_{Process\_Teseo} + 65 \cdot Altitude_{LPS25HBd}}{100}$$

(Equation 44)

The second operation performed is to check the "take-off enable" logical input, then in parallel read the "landing information" broadcasted by the VHDL component "COMPASS_input" (DATA_OUT_Landing) and the reference altitude value broadcasted by the VHDL component "Flight_Parameters_EEPROM" ("TARGET_ALT"). In the case of active "take-off enable" (it acts like an automotive KL30 logic signal more than an automotive KL15 logic signal), a bespoke multiplier coefficient will accordingly modify the "DATA_OUT_Landing" value[154], else the "DATA_OUT_Landing" value will be untouched.

Consequently, the resulting altitude reference value is calculated according to:

$$Altitude_{reference} = \frac{TARGET\_ALT \cdot DATA\_OUT\_Landing}{65535}$$

(Equation 45)

Then, obtained values $Altitude_{reference}$ and $Altitude_{read}$ are used to solve the "Equation 34" (paragraph 5.1.5).

[154] The value of the bespoke multiplier coefficient depends on the vehicle's flights dynamic, its mechanical design and its typical take-off requirements.

The final operation is to digital process the calculated solution of the "Equation 34" in order to deliver to the neuro-fuzzy controller ("Altitude_error" information) 8-bit information that expresses a compatible relative vehicle's altitude error. For compatible relative altitude error, it is intended that given a scale between 0 and 255 (8-bit resolution), the vehicle's relative altitude error will correspond to a spectrum of values within the following boundaries:

- "0" in case of a relative error $\leq -100\%$;
- between "1" and "126" in case of $-100\% < Altitude_{error} < 0$;
- "127" in case of null relative error;
- between "128" and "254" in case of $0 < Altitude_{error} < +100\%$;
- "255" in case of a relative error $\geq +100\%$.

It allows expressing the vehicle's altitude relative error in percentage in order to comfortably adapt the control algorithm to other vehicles and various flight plans.

Component's safety mechanisms analysis are irrelevant for the dissertation.

6.7.1.4 "ENERGY_input", VHDL component

The VHDL component "ENERGY_input" receives an 8-bit "DATA_IN" input information from the component "BMS_VHDL" and an array of logic signals (primary clock, secondary clock, operation enable, and take-off enable). Component's algorithm generates an 8-bit "Battery_status" output value in a form utilizable by the neuro-fuzzy controller.

Given a scale between 0 and 255 (8-bit resolution), for compatible 8-bit "Battery_status" output, it is intended that the vehicle's REESS SoC will correspond to a spectrum of values within the following boundaries:

- "0" in case of $SoC = 0\%$;
- between "1" and "254" in case of $0\% < SoC < +100\%$;
- "255" in case of $SoC = 100\%$.

Component's safety mechanisms analysis are irrelevant for the dissertation.

6.7.1.5 ROLLING_input, VHDL component

The VHDL component "ROLLING_input" receives an array of multiple 16-bit independent information from the VHDL components: "A3G4250D" and "LIS3DSH". Therefore, it receives 8-bit information from the VHDL component "ANGLE_input", representing the vehicle's heading angle. It also receives an array of logic signals: enable, primary clock, secondary clock and take-off enable.

"ROL_angle" is the component's 8-bit output information generated by the "ROLLING_input" VHDL component algorithm. This output information is used by the "neuro-fuzzy controller" and the "Flight Telemetry EEPROM" VHDL component. The 8-bit resolution output signal expresses the vehicle's rolling angle in a bespoke arrangement, according to the following guidelines:

- "0" in case of $Rolling_{angle} \leq -20°$;
- between "1" and "126" in case of $20° < Rolling_{angle} < 0°$;
- "127" in case of a null rolling angle;
- between "128" and "254" in case of $0° < Rolling_{angle} < +20°$;
- "255" in case of $Rolling_{angle} \geq 20°$.

Component's safety mechanisms analysis are irrelevant for the dissertation.

6.7.1.6 YAW_input, VHDL component

The VHDL component "YAW_input" receives an array of multiple 16-bit independent information from the VHDL components: "A3G4250D" and "LIS3DSH". Therefore, it receives 8-bit information from the VHDL component "ANGLE_input", representing the vehicle's heading angle. It also receives an array of logic signals: enable, primary clock, secondary clock and take-off enable.

"YAW_angle" is the component's 8-bit output information generated by the "YAW_input" VHDL component algorithm. This output information is used by the "neuro-fuzzy controller" and the "Flight Telemetry EEPROM" VHDL component. The 8-bit resolution output signal expresses the vehicle's YAW angle in a bespoke arrangement, according to the following guidelines:

- "0" in case of $Yaw_{angle} \leq -20°$;
- between "1" and "126" in case of $20° < Yaw_{angle} < 0°$;
- "127" in case of a null yaw angle;
- between "128" and "254" in case of $0° < Yaw_{angle} < +20°$;
- "255" in case of $Yaw_{angle} \geq 20°$.

Component's safety mechanisms analysis are irrelevant for the dissertation.

6.7.1.7 PITCH_input, VHDL component

The VHDL component "PITCH_input" receives an array of multiple 16-bit independent information from the VHDL components: "A3G4250D" and "LIS3DSH". Therefore, it

receives 8-bit information from the VHDL component “ANGLE_input”, representing the vehicle’s heading angle. It also receives an array of logic signals: enable, primary clock, secondary clock and take-off enable.

“PITCH_angle” is the component’s 8-bit output information generated by the “PITCH_input” VHDL component algorithm. This output information is used by the “neuro-fuzzy controller” and the “Flight Telemetry EEPROM” VHDL component. The 8-bit resolution output signal expresses the vehicle’s PITCH angle in a bespoke arrangement, according to the following guidelines:

- “0” in case of $Pitch_{angle} \leq -20°$;
- between “1” and “126” in case of $20° < Pitch_{angle} < 0°$;
- “127” in case of a null pitch angle;
- between “128” and “254” in case of $0° < Pitch_{angle} < +20°$;
- “255” in case of $Pitch_{angle} \geq 20°$.

Component’s safety mechanisms analysis are irrelevant for the dissertation.

6.7.1.8 COMPASS_input, VHDL component

The VHDL component “Compass_input” is the most complex sub-component within the VHDL component “Digital_Processing”.

The VHDL component “Compass_input” receives an array of multiple 16-bit independent information from the VHDL components: “A3G4250D”, “LIS3DSH”, “Flight_Parameters_EEPROM” and “TESEO”. Therefore, it receives 8-bit information from the VHDL component “ANGLE_input”, representing the vehicle’s heading angle. It also receives an array of logic signals: enable, primary clock, secondary clock and take-off enable.

The VHDL component “Compass_input” reads the vehicle’s current position (from the VHDL component “TESEO”) and reads the vehicle’s target position (from the VHDL component “Flight_Parameters_EEPROM”), then computes the captured data. This data computation produces the ideal vehicle’s heading angle and the distance between the vehicle and the target position. Obtained data is the input of a new data manipulation process. The ideal vehicle’s heading angle is compared with the actual vehicle’s heading angle (current vehicle’s heading angle) read from the VHDL component “ANGLE_input”. The outcome is a simple differential value, which allows the neuro-fuzzy controller to adjust its routes accordingly.

$$Compass_{error} = Vehicle's\_Heading\_angle_{ideal} - Vehicle's\_Heading\_angle_{actual}$$

(Equation 46)

"Compass_error" is an 8-bit information output signal that the VHDL components "Compass_input" broadcasts to the neuro-fuzzy controller. The resulting 8-bit information expresses the vehicle's heading angle error in a bespoke arrangement, according to the following guidelines:

- "0" in case of $Compass_{error} = -180°$;
- between "1" and "126" in case of $-180° < Compass_{error} < 0°$;
- "127" in case of a null heading angle error;
- between "128" and "254" in case of $0° < Compass_{error} < +180°$;
- "255" in case of $Compass_{error} = 180°$.

The second output generated by the VHDL component "Compass_input" is the "DATA_OUT_Landing", which is a piece of 16-bit resolution information transmitted to the VHDL component "ALTITUDE_input". This information has paramount importance during the landing operation because it defines the reference flight altitude's correction factor that the VHDL component "ALTITUDE_input" shall use to calculate the "Altitude_error".

This parallel computation performed by the VHDL component "Compass_input" starts with the reading of the input information "DATA_LANDING", received from the VHDL component "Flight_Parameters_EEPROM, which sets the distance[155] to the final position that the controller will use to start the landing manoeuvre. As shorter the distance will be as more severe reference altitude correction will be requested to the VHDL component "ALTITUDE_input". When the $Distance_{reference}$ (input value "DATA_LANDING" read) is larger than the $Distance_{read}$ (actual vehicle's distance to the target), it is valid the "Equation 47".

$$DATA\_OUT\_Landing = \frac{Distance_{read}}{Distance_{reference}} \cdot 65535$$

(Equation 47)

The resulting 16-bit "DATA_OUT_Landing" information generated by VHDL component "Compass_input" provides a piece of landing information to the VHDL component "ALTITUDE_input" in a bespoke arrangement, as following:

- "0" in case of $Distance_{read} = 0$;
- between "1" and "65534" according to "Equation 47";

[155] The distance between the target landing position and the instantaneous vehicle's position expressed in meters.

- "65535" in case of $Distance_{reference} \leq Distance_{read}$.

6.7.2 *"Fuzzy", VHDL component*

"Xfuzzy environment" contains the tool "xfvhdl", which uses the high-level hardware description language VHDL to facilitate the hardware implementation, through FPGAs or ASICs, of inference systems. The "tool"[156] allows the direct synthesis of complex fuzzy systems composed of different inference modules and crisp blocks. [44]

The GUI's "FPGA implementation section" collects information regarding the definition options[157] for FPGAs. For the rule memory, it can be chosen to implement them with ROM, RAM or logical blocks. Once all "Rulebases" and crisp blocks of the system are defined, it is possible to generate the VHDL code of the components associated with the fuzzy system. The generation of a "testbench" file, also described in VHDL, follows the fuzzy system VHDL code generation and allows verifying its functionality.

For hierarchical systems, each "Rulebase" requires a VHDL description, as well as a "testbench" that allows obtaining the control surface corresponding to each of them. In this case, a VHDL file corresponding to the upper level of the hierarchy is also generated, which describes the interconnection of the different "Rulebases" and crisp blocks that make up the system, as well as a "testbench" that allows simulating the whole system.[38]

What described impacts on the final VHDL assembly directly. By definition, proposal work uses a hierarchical design ("Figure 6.63" and "Figure 6.64") and the final VHDL code constructions strategy targets a parallel computation for each independent "Rulebase".

Using the "xfvhdl" tool of the "XFuzzy" environment, a set of five independent, code populated VHDL components are exported. Those components are encased in the central "flight controller", and the interconnections between the different "Rulebases" and the corresponding upper hierarchical level input/output ports are defined and then refined.

The "Rulebases" parallel computational design relies on the allocation of the hardware described by the VHDL code will be implemented in a sheathed area of the physical FPGA. "Figure 6.66" demonstrates such concepts.

---

[156] The tool allows the generation of standardised membership functions of triangular, sh_triangular, and trapezoid types by means of arithmetic techniques. If input membership functions are not normalised, the arithmetic calculation option for antecedents is disabled.

[157] Among them, the type of RAM and ROM to be used.

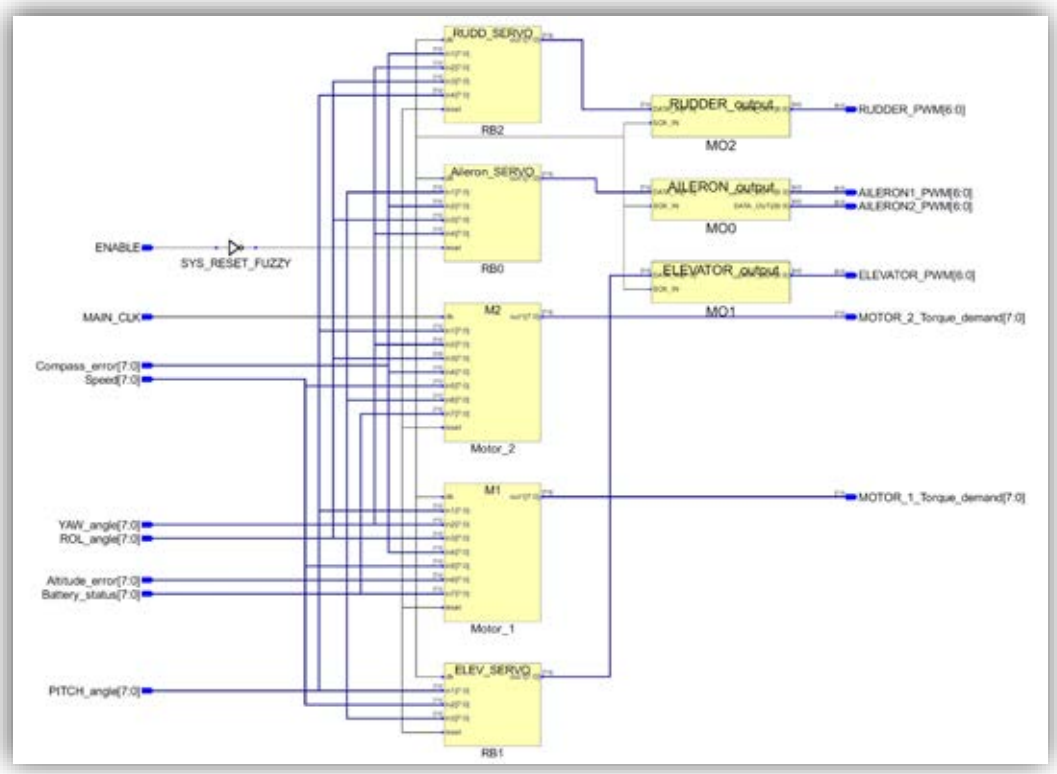


Figure 6.66: RTL view, the outcome of the "Fuzzy" VHDL component algorithm's synthesis.

The study case application VHDL algorithm encases the VHDL components associable to the five "Rulebases" into the VHDL component "Fuzzy", as it is possible to observe in the RTL view of "Figure 6.66"; which is the outcome of the "Fuzzy" VHDL component algorithm's synthesis. An array of three more sub-components are created and encased within the VHDL component "Fuzzy", those are:

a) AILERON_output
b) ELEVATOR_output
c) RUDDER_output.

The first subcomponent converts a single 8-bit resolution signal into two complementary 7-bit resolution signals, according to what previously described ("paragraph 4.2.2", "paragraph 5.1.8" and "paragraph 6.3.9"). The second and third subcomponents merely convert 8-bit resolution information into 7-bit resolution information.

6.8 Data Capture for the learning Process

Data collection is generally achieved as the vehicle is guided via remote control (RC) through an area defined as the "selected environment", which simulates the environment and the flight conditions in which the vehicle will most likely operate. Therefore, appropriate

hardware is required to provide the RC features of the learning process, according to the assumptions and the choices described in "paragraph 5.2".

The operator guides the vehicle through a series of flight pre-defined route, simulating as many as possible manoeuvres to establish a base and bias pattern for the algorithm. A dedicated on-board data collection algorithm is compulsory, and it results in a spin-off functionality of the controller described in the previous paragraph. As a benchmark, the VHDL algorithm shall capture the flight sensors data, the flight actuators output and the torque demand for the E-Motors, during the allocated learning period T_{learn} (or time available for the flight telemetry monitor), during which time the on-board memory chip retains the received data. By definition, T_{learn} must not exceed MOB maximum as in "Equation 48" [158] (which is a bespoke adaptation of "Equation 29" of "paragraph 2.5.1").

$$T_{learn} < \left(\frac{MOB}{NRR \cdot S_{ps}} \right)$$

(Equation 48)

Where:

T_{learn} → maximum run time expressed in seconds

MOB → on-board memory expressed in bytes

NRR → Number of Registers Read for each sample (1 Register = 1 byte)

S_{ps} → samples per second

For the proposed configuration, a vehicle diagnostics and monitor VHDL bespoke algorithm is designed to read all the vehicle's parameters and store the data on external memory, with a sampling frequency of 8Hz. It is assumed that the external memory is a standard 2Mbit EEPROM (as for "paragraph 5.1.9", an alternative[159] may be a NOR FLASH memory with a standard SPI interface) connected with the FPGA using an SPI BUS[160]. The integration of the external memory, and its related functionalities, within the main VHDL algorithm is achieved via the VHDL component "Telemetry_EEPROM" (as shown in the RTL view of Figure 6.63). Selected SPI BUS clock[161] shall compile with the equation:

---

[158] Because it will result in corrupted data, due to the undesirable registers overwriting.

[159] Especially in the case of a larger data storage shall be guaranteed.

[160] It is assumed that the FPGA is the "SPI MASTER" and the memory is the "SPI SLAVE".

[161] It is assumed the value of 1 MHz as an upper limit because the maximum SPI clock allowed may change with the memory P/N (it is recommended to use a value that may be accepted by a large number of devices) and, the potential distance between the FPGA and the memory may affect the robustness of the data stream when a higher clock frequency is used.

$$f_{SPI} > \left(8 \cdot NRR\right) \cdot \left(8 \cdot S_{ps}\right) \leq 1MHz$$

(Equation 49)

According to “Equation 49”, the VHDL component “Telemetry_EEPROM” utilizes an SPI BUS communication clock of 512KHz.

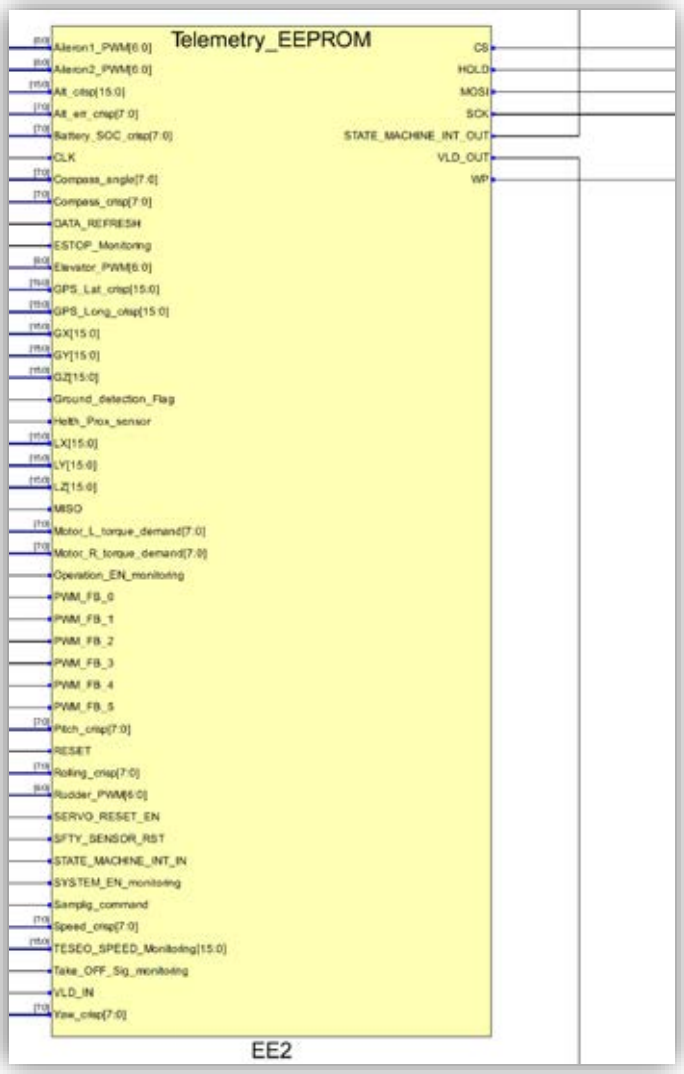


Figure 6.67: macro of the RTL view outcome of the main VHDL algorithm’s synthesis (macro of Figure 6.63).

Reference	Description	Num. of Registers
AILERON SERVO	Input and Output Value – SERVO angle	4
ELEVATOR SERVO	Input and Output Value – SERVO angle	2
RUDDER SERVO	Input and Output Value – SERVO angle	2
M1	Input and Output Value – L. motor torque	2
M2	Input and Output Value – R. motor torque	2
Gx	Input – X axle angular acceleration sensor	2
Gy	Input – Y axle angular acceleration sensor	2
Gz	Input – Z axle angular acceleration sensor	2
Lx	Input – X axle linear acceleration sensor	2
Ly	Input – Y axle linear acceleration sensor	2
Lz	Input – Z axle linear acceleration sensor	2
YAW	Input – digitally processed YAW feedback	1
PITCH	Input – digitally processed Pitch feedback	1
ROLLING	Input – digitally processed Rolling feedback	1
BATTERY	Input – digitally processed Battery SoC	1
COMPASS	Input – digitally processed compass error	1
HEADING ANGLE	Input – digitally processed compass error	1
ALTITUDE	Input – absolute altitude value	2
ALTITUDE_ERR	Input – digitally processed altitude error	2
GPS Module SPEED	Input – absolute speed value	2
SPEED	Input – digitally processed speed value	1
GPS Position	Input – digitally processed position value	4
STATUS_Register	Input – System monitoring register	1
Timing	Output Value – Sampling Timing value	2

Table 6.2: VHDL component "Telemetry_EEPROM" registers map.

According to "Table 6.2", VHDL algorithm requires to write 44 memory registers for each sampling (NRR=44), making it possible to estimate the maximum data memorisation

time (or maximum run time) according to the “Equation 48”, since the assumed EEPROM (or Flash) memory size is of 2Mbit (or 256kB as described in the “paragraph 5.1.9”). The result is a maximum run time of 727.27 seconds (approximately 12 minutes). It means a second training flight time limitation after the limitation of the battery’s energy. The Author’s recommendation is to operate training data mining (RC driving) for not more than 10 minutes.

Particular attention requires the data capture algorithm of the VHDL component “Telemetry_EEPROM”. By definition, the algorithm is a bespoke added functionality to the primary “Flight Controller”. The “Telemetry_EEPROM” VHDL component is designed to operate in both circumstances: in case the system is working in “autonomous mode” or in “training mode” (with a “Human Pilot”).

The Author’s technical requirements definition of the data mining operation implies ensuring that the following rules are respected:

- The “Telemetry_EEPROM” VHDL component encase a set of new VHDL sub-components capable of reading a PWM control signal (the SERVO-Motors PWM control signal generated directly by the RC equipment and the powertrain’s feedback PWM signals generated by each E-Motor Driver);
- for the “training mode” operation, it is necessary to ensure the hardware separation of the SERVO-Motors FPGAs outputs with the actuators control signal feedback (it may happen within the FPGA with an algorithm variant or externally with the depopulation of the resistor between the FPGA’s output pin and the SERVO-Motor driver circuit);
- The “Telemetry_EEPROM” VHDL requires a sub-component that will read an array of eight logic signals and will associate its value to a particular bit of the “STATUS Register”[162].

Implementing these operations allows obtaining all the information needed since all the data described in “Table 6.2” will be associated with a defined time.

[162] “STATUS_Register” map: Helth_Prox_sensor [BIT_0], Ground_detection_Flag [BIT_1], ESTOP_Monitoring [BIT_2], SFTY_SENSOR_RST [BIT_3], Operation_EN_monitoring [BIT_4], Take_OFF_Sig_monitoring [BIT_5], SERVO_RESET_EN [BIT_6], SYSTEM_EN_monitoring [BIT_7].

6.9 Learning/Training Process Description

The tuning stage is usually one of the most complex tasks when designing fuzzy systems. The system behaviour depends on the logic structure of its "Rulebases" and the membership functions of its linguistic variables. The tuning process focuses on adjusting the different membership function parameters that appear in the system definition. The simulations' outcome shows that the preliminary systems can perform basic tasks and behave according to what is expected from a "human being pilot" facing the same environmental conditions. In conclusion, **the preliminary tuning is successful.**

Since the number of parameters to simultaneously modify is high, advanced manual tuning is undoubtedly a cumbersome, and automatic techniques may be required. The two learning mechanisms most widely used are "supervised learning" and the "reinforcement learning".

In **supervised learning techniques,** the desired system behaviour is given by a set of training (and test) input/output data, while in **reinforcement learning,** what is known is not the exact output data but the effect that the system has to produce on its environment, thus making necessary the monitoring of its on-line behaviour. [38]

Both techniques are appropriate to the Thesis principles, although the Author privileges the **"supervised learning techniques"** because more congenial to the work mindset. "Xfuzzy 3" GUI is used as a baseline; the software environment includes four tools for this design stage:

- **"xfdm"** allows obtaining the structure of inference systems used as fuzzy approximators or classifiers;
- **"xftsp"** focuses on time series prediction applications[163] [45];

[163] The tool xftsp generates fuzzy inference systems that implement autoregressive models for the short- and long-term prediction of time series. To do this, it applies a methodology based on the use of non-parametric noise or residual variance estimates (to select the optimal number of input variables) in combination with Xfuzzy supervised learning and identification tools (to determine the structure of the systems). This methodology responds to a direct prediction strategy, which implies the construction of an autoregressive model for each of the terms of the desired prediction horizon. In each case, the optimal subset of inputs is selected a priori by a non-parametric noise estimate (for example, the Delta Test). The specification of the fuzzy system corresponding to each prediction horizon is then obtained through an iterative process in which successive identification and adjustment phases are carried out, increasing the number of linguistic labels of the inputs, until the system error enters the previously estimated range. xftsp can be executed in graphic mode, using the option "Time Series Prediction" of the "Tuning Menu" or the corresponding icon in the main window of the environment, or from the command line using a configuration file. [38]

- **"xfsl"** is a parameter adjustment tool based on the use of supervised learning algorithms[164];
- **"xfsp"** is a simplification tool that allows reducing the number of membership functions and compacting the rules bases of a fuzzy system to facilitate its software or hardware implementation and to increase its linguistic interpretability.

[38]

6.9.1 Knowledge acquisition tool ("xfdm" tool)

It is noteworthy to highlight that the tool "xfdm" facilitates the identification of fuzzy systems from numerical data through the use of different algorithms grouped into two categories.

Structure-oriented algorithms[165] represent the first group. These algorithms perform a fixed or variable partition of the input variables' universes discourse and analyse the numerical data that describe the system's behaviour to assign a rule for each line of the input file. Subsequently, they resolve the conflicts that may have occurred while selecting the fuzzy system rules based on their activation degrees and the configuration parameters defined by the user. "xfdm" includes three identification algorithms that work with fixed partitions (Wang & Mendel, Nauck and Senhadji) and one that includes a variable number of partitions (Incremental Grid). Additionally, the "Flat System" option allows the generation of fuzzy system specifications with a flat I/O behaviour that can be useful as input to the training tool or other Xfuzzy facilities.

The specific options and parameters of these algorithms are:

a) Nauck:
   - Number of rules: number of rules to identify
   - Type of selection: "Best rules" or "Best per class
b) Sendhadji:
   - Number of rules: number of rules to identify
c) Incremental Grid:

[164] In supervised learning techniques, the desired behaviour of the system is described by a set of training (and test) patterns. Supervised learning attempts to minimise an error function that evaluates the difference between the actual system behaviour and its desired behaviour defined by the set of input/output patterns. [38]

[165] Matrix partitioning (Grid Partitioning).

- Limit of MFCs, Limit of Rules, Limit of RMSE (the execution of the algorithm ends when one of these limits is reached)
- Learning option, activated/not activated

[38]

Cluster-oriented algorithms[166] represent the second classification group. "**xfdm**" tool includes other algorithms to generate a fuzzy system from a series of data using clustering techniques. By grouping sets of points in clusters represented by prototype points, it is possible to reduce the algorithm's information load and allowing fuzzy systems with fewer rules. The tool includes four algorithms that use a fixed number of clusters (Hard C-Means, Fuzzy C-Means, Gustafson-Kessel and Gath-Geva) and two algorithms that allow iteratively varying the number of clusters until the limit defined by the user is reached (Incremental Clustering and ICFA).

The specific options and parameters of these algorithms are:

a) Incremental Clustering:
   - Neighbourhood radius
   - Maximum number of clusters

b) Fixed Clustering:
   - Clustering algorithm[167]
   - Number of clusters
   - Limit on iterations
   - Fuzziness index
   - Limit on cluster variation
   - Learning option, activated/not activated

c) ICFA (Incremental Clustering for Function Approximation):
   - Number of clusters
   - Maximum Iterations
   - Fuzziness index
   - Limit on cluster variation
   - Activate migration, activated/not activated

[38]

[166] Data grouping (Cluster Partitioning) techniques.
[167] Hard C-Means, Fuzzy C-Means, Gustafson-Kessel and, Gath-Geva.

6.9.2 *The supervised learning*

Supervised learning algorithms aim to reduce the error function that defines the deviation between the actual and the desired system behaviour; they can be considered optimisation algorithms.

The supervised learning tool "xfsl"[168] allows applying supervised learning algorithms to adjust (training process) fuzzy systems into the design flow of "Xfuzzy 3.5". The learning process configuration **starts with the (it is the first step) selection of a training file that contains the input/output data of the desired behaviour** (a test file, whose data check the generalisation of the learning, is discretionary to the end-user[169]). [46]

The second step in the tuning process configuration is the selection of the learning algorithm ("**xfsl**" admits many learning algorithms[170]). **Once the algorithm is selected, an error function must be chosen**. The tool offers several error functions[171] capable of expressing the deviation between the actual and the desired behaviour.

"**xfsl**" contains two processing algorithms to simplify the designed fuzzy system. **The first algorithm prunes the rules and reduces the membership functions** that do not reach a significant activation or membership degree. There are three versions of the algorithm:

a) pruning all rules that are never activated over a certain threshold;
b) pruning the worst N-rules;
c) pruning all rules except the best N-ones.

The second algorithm clusters the membership functions of the output variables. The number of clusters can be fixed to a certain quantity or computed automatically. These two processing algorithms are applicable to the **system before the tuning process (pre-processing option) or after it (post-processing option).** An end condition[172] has to be specified to finish the learning process.

---

[168] "xfsl" contains many different supervised learning algorithms.

[169] The format of these two patterns files is just an enumeration of numeric values that are assigned to the input and output variables in the same order that they appear in the definition of the system module in the "XFL3 description".

[170] Regarding gradient descent algorithms, it admits Steepest Descent, Backpropagation, Backpropagation with Momentum, Adaptive Learning Rate, Adaptive Step Size, Manhattan, QuickProp and RProp. Among conjugate gradient algorithms, the following are included: Polak-Ribiere, Fletcher-Reeves, Hestenes-Stiefel, One-step Secant and Scaled Conjugate Gradient. The second-order algorithms included are: Broyden-Fletcher-Goldarfb-Shanno, Davidon-Fletcher-Powell, Gauss-Newton and Mardquardt-Levenberg. Regarding algorithms without derivatives, the Downhill Simplex and Powell's method can be applied. Finally, the statistical algorithms included are Blind Search and simulated annealing (with linear, exponential, classic, fast, and adaptive annealing schemes).

[171] By default, the Mean Square Error is selected.

[172] This condition is a limit imposed over the number of iterations, the maximum error goal, or the maximum absolute or relative deviation (considering both the training and the test error).

"**xfsl**" can be applied to any fuzzy system described by the XFL3 language, even to systems that employ particular functions defined by the user. Vital is the scrutiny of the system's feature that may impose limitations over the learning algorithms to apply (for instance, a non-derivative system cannot be tuned by a gradient-descent algorithm). [38]

6.9.2.1 Gradient Descent Algorithms

The similarity between fuzzy systems and neural networks led to apply the neural learning processes to fuzzy inference systems. Under these circumstances, a well-known algorithm employed in fuzzy systems is the "Back Propagation algorithm", which adjusts the parameter values proportionally to the gradient of the error function in order to reach a local minimum. Since this algorithm's convergence speed is slow, several modifications could be beneficial, like using a different learning rate for each parameter or adapting the control variables of the algorithm heuristically. An exciting modification that improves the convergence speed takes into account the gradient value of two successive iterations. It makes available information about the curvature of the error function. The algorithms QuickProp and RProp follow this idea. "**xfsl**" admits:

a) Backpropagation;
b) Backpropagation with Momentum;
c) Adaptive Learning Rate;
d) Adaptive Step Size;
e) Manhattan;
f) QuickProp;
g) RProp.

[38]

6.9.2.2 Conjugate Gradient Algorithms

The gradient-descent algorithms generate a change step in the parameter values that is a function of the gradient value at each iteration (and possibly at previous iterations). Since the gradient indicates the direction of maximum function variation, it may be convenient to generate not only one step but several steps, which minimise the function error in that direction. The illustrated strategy, which is the basis of the steepest-descent algorithm, has the detriment of producing a zig-zag advancing because the optimisation in one direction may deteriorate previous optimisations. The solution is to advance by conjugate directions that do

not interfere with each other. The several conjugate gradient algorithms reported in the literature differ in the equations used to generate the conjugate directions.

The main drawback of the conjugate gradient algorithms is the implementation of a linear search in each direction, which may be costly in terms of function evaluations. It is possible to avoid the linear search by using second-order information, that is, by approximating the second derivative with two close first derivatives. Illustrated idea defines the basis of the scaled conjugate gradient algorithm. Among conjugate gradient algorithms, "**xfsl**" includes the following:

a) Steepest Descent;
b) Polak-Ribiere;
c) Fletcher-Reeves;
d) Hestenes-Stiefel;
e) one-step Secant;
f) Scaled Conjugate Gradient.

[38]

6.9.2.3 Second-Order Algorithms

A valid approach capable of speeding up the convergence of learning algorithms considers the second-order information of the error function, that is, of its second derivatives or, in matricial form, of its Hessian. Considering complicated the computation of the second derivatives may be beneficial to approximate the Hessian employing the gradient values of successive iterations. It is the idea of Broyden-Fletcher-Goldarfb-Shanno and Davidon-Fletcher-Powell algorithms.

A meaningful case is when the function to minimise is a quadratic error because the Hessian can be approximated by only the first derivatives of the system outputs, as done by the Gauss-Newton algorithm. Since this algorithm can lead to instability when the approximated Hessian is not-positive defined, the Marquardt-Levenberg algorithm solves this problem by introducing an adaptive term. The second-order algorithms included in the tool are:

a) Broyden-Fletcher-Goldarfb-Shanno;
b) Davidon-Fletcher-Powell;
c) Gauss-Newton;
d) Mardquardt-Levenberg.

[38]

6.9.2.4 Algorithms Without Derivatives

The gradient of the error function is not always obtainable (calculated) because it can be too costly or not defined. In these cases, it is possible to employ optimisation algorithms without derivatives. An example is the "Downhill Simplex algorithm", which considers a set of function evaluations to decide a parameter change. Another example is Powell's method, which implements linear searches by a set of directions that evolve to be conjugate. The algorithms of this kind are too much slower than the previous ones. An optimal solution could be to estimate the derivatives from the secants or to employ not the derivative value but its sign (as RProp does), which allows the estimations from small perturbations of the parameters. [38]

All the above-commented algorithms do not reach the global but a local minimum of the error function. The statistical algorithms can discover the global minimum because it is possible to generate different system configurations that spread the search space. One way of broadening the space explored is to generate random configurations and choose the best one. It applies the "Blind Search algorithm", whose convergence speed is plodding. Another way is to perform small perturbations in the parameters to find a better configuration, such as the iterative improvements algorithm. A better solution is to employ "simulated annealing algorithms". The strategy exploits the analogy between the "learning process"[173] and the evolution of a "physical system"[174]. Simulated annealing provides good results when the number of parameters to adjust is low. When it is high, the convergence speed can be so slow; beneficial could be the generation of random configurations, apply gradient descent algorithms and select the best solution.

There are applicable algorithms without derivatives: the Downhill Simplex and Powell's method. As well are available statistical algorithms: "blind search" and "simulated annealing" (with linear, exponential, classic, fast, and adaptive annealing schemes).

When optimising a differentiable system, Broyden-Fletcher-Goldarfb-Shanno and Mardquardt-Levenberg algorithms are the most adequate. If it is not possible to compute the system derivatives, as in hierarchical fuzzy systems, the best choice is to use these algorithms with the option of estimating the derivative. Simulated annealing is recommendable when

[173] The "learning process" is intended to minimise the error function.
[174] The "physical system" tends to lower its energy as its temperature decreases.

there are a few parameters to tune, and the second-order algorithms drive the system to a non-optimal minimum.

[38]

6.9.3 Error function

The error function expresses the deviation between the actual behavior of the fuzzy system and the desired one by comparing the input/output patterns with the output of the system for those input values. "**xfsl**" defines seven error functions:

a) mean_square_error (MSE);
b) weighted_mean_square_error (WMSE);
c) mean_absolute_error (MAE);
d) weighted_mean_absolute_error (WMAE);
e) classification_error (CE);
f) advanced_classification_error (ACE);
g) classification_square_error (CSE).

Listed functions are normalised by the number of patterns, the number of output variables, and the range of each output variable so that the error function's range is between 0 and 1. MSE, WMSE, MAE and WMAE are eligible for systems with continuous output variables. While CE, ACE and CSE are the best fit for classification systems. These are the equation for the first functions:

$$MSE < SUM\left(\frac{\left(\frac{Y-y}{range}\right) \star\star 2}{(num\_pattern \star num\_output)}\right)$$

(Equation 50)

$$WMSE < SUM\left(\frac{w \star \left(\frac{Y-y}{range}\right) \star\star 2}{(num\_pattern \star SUM(w))}\right)$$

(Equation 51)

$$MAE < SUM\left(\frac{\left|\left(\frac{Y-y}{range}\right)\right|}{(num\_pattern \star num\_output)}\right)$$

(Equation 52)

$$WMAE < SUM\left(\frac{w \star \left|\left(\frac{Y-y}{range}\right)\right|}{(num\_pattern \star SUM(w))}\right)$$

(Equation 53)

[38]

The output of a fuzzy classification system is the linguistic label that has the most significant activation degree. A common way of expressing these systems' deviation is the number of classification failures (classification_error, CE). It is not an optimal choice for tuning because many system configurations produce the same number of failures. A valid improvement adds a term that measures the selected label's distance to the desired one (advanced_classification_error, ACE). These two error functions are not differentiable, so they cannot be used with derivative-based learning algorithms (which are the fastest). A better choice is to consider each linguistic label's activation degree as the actual output and the desired output as "1" for the correct label and "0" for the others. The error function estimation may use the system's square error (classification_square_error, CSE), which is differentiable and functional with derivative-based learning algorithms. [38]

6.9.4 The Simplification tool - Xfsp

The tool "**xfsp**" allows applying simplification algorithms, both to the membership functions and to the neuro-fuzzy system "Rulebases", to obtain a more forthright description or one that is easier to comprehend from the linguistic point of view. [47]

6.9.4.1 Membership functions simplification

There are several Membership functions simplification methods available in the scientific literature; the proposed work uses as baseline three simplification processes:

a) "Purge";
b) "Clustering";
c) "Similarity".

The "**Purge Mechanism**" looks for those membership functions which are not used in any "Rulebase" and eliminates them. It may happen not only as a consequence of previous simplification processes but also with a fuzzy system definition based on translating heuristic knowledge.

The "**Clustering Method**" uses the "Hard C-Means algorithm" to search for a small number of clusters (prototype membership functions) that allow grouping several of the

original functions. The clusters' evaluation occurs in the space formed by the different parameters that define the membership functions, being possible to apply weights to each one of them. The user can define the final number of prototypes, or in the alternative automatically calculated by applying different validity indices: Dunn separation index, Davies-Bouldin index and Dunn generalised indexes.

The third technique examines a **merging process** based on the **similarity between the different functions**. This process iteratively searches for the pair of most similar functions and replaces them with a single function if the degree of similarity exceeds a threshold defined by the user. The process ends when it is not possible to merge more functions. [38]

6.9.4.2 Rulebases simplification

There are several "Rulebases" functions simplification methods accessible in the scientific literature; there are four applicable processes to the "Ruleset":

a) compress;
b) pruning;
c) expand;
d) tabular simplification.

The "**compression method"** combines all the rules that share the same consequent, connecting their antecedents by disjunctions ("or" connective).

The "**expansion method"** implements the process complementary to compression. Both methods can help the user to visualize better and understand the "Rulebase", but in reality, they do not perform an adequate simplification. The simplification operation can perform the pruning method and (or) the tabular simplification.

The "**pruning process"**[175] is usually a pre-processing method applied ere any simplification. This process evaluates the degree of activation of the rules to eliminate:

a) the "n" worst rules;
b) all rules except the "n" best rules;
c) all rules whose degree of activation is below a threshold (the user sets both the number "n" and the threshold).

The last of the simplification mechanisms examined is the **"tabular simplification method"** of the "Rulebases" on an extension of the Quine-McCluskey algorithm. This

[175] Pruning allows reducing the number of rules by selecting the most important in the context of a particular application.

method performs an ordered linear search[176] to find all combinations of logically adjacent minterms of the n-variable function to be simplified. [38]

6.10 Proposed Learning Tool Configuration

Moving from the hardware assumptions made in "paragraph 5.2" and using as baseline the theoretical studies of "paragraph 6.9", it is possible to operate the proposed "learning tool" configuration.

A preliminary condition is the utilisation of a training file compatible with the "learning tool". It requires that the raw data stored in the external memory shall be transferred on the user PC and then processed in order to ensure the conformation of the input/output information pattern with the "learning tool" standards (as well shall be given to the file the extension ".trn").

Configuration starts with the selection of the learning technique. The Author justifies its predilection for **supervised learning techniques** because this technique focuses on the system behaviour given by a set of training input/output data (physical information captured). It matches the training file goals and its generation's modality because the "data collection[177]" outcome allows replicating a human being pilot flight behaviours under certain flight conditions.

The successive configuration step is the selection of the learning algorithm. The Author proposes to opt for "**Gradient Descent algorithms**", which are well-known algorithms employed in fuzzy systems learning processes. Although the most common variant is the "Back Propagation algorithm", it is preferred to use the "**Manhattan algorithm**". The root of the decision is in the principles behind the algorithms. "Back Propagation algorithm" modifies the parameter values proportionally to the gradient of the error function in order to reach a local minimum. Since this algorithm's convergence speed is slow, modifications such as a different learning rate for each parameter or adapting, heuristically, the control variables of the algorithm are beneficial. "**RProp algorithm**" follows this strategy and improves significantly the convergence speed taking into account the gradient value of two successive iterations providing information about the curvature of the error function. The "**Manhattan**

[176] It begins with a list of all the minterms of the function to later obtain successively lists with (n-1)-, (n-2)-, ..., variable implicants until no more implicants can be formed, thus obtaining the so-named "prime implicants" of the function. The last step is to select the minimum number of prime implicants that cover all the minterms. [38]

[177] It is achieved as the vehicle is guided via remote control (RC) through an area defined as the "selected environment".

algorithm" is chosen because it represents a good trade-off between the algorithm's accuracy and the computational power required.

The subsequent configuration step is the selection of the error function. The Author assumes the default function, "**Mean Square Error**" function, as an appropriate option. The decision's aim attempts to avoid the unnecessary heavy computational process.

Following configuration tool selection is the identification of the best processing algorithm to use for the designed fuzzy system simplification. Since the processing algorithms can be applied to the system before the tuning process (pre-processing option) or after it (post-processing option).

The Author's choice is to use the "**post-processing option**". In order to complete the process, it is necessary to select the pruning algorithm that prunes the rules and reduces the membership functions. Author preference is for the method of **pruning the worst "N"[178] rules**.

The last configuration step is the definition of the end condition[179]; it is mandatory to specify how shall conclude the learning process. This condition limits the number of iterations, the maximum error goal, or the maximum absolute or relative deviation (considering both the training and the test error). The end condition can be one of the following:

a) Limit on Iterations;
b) Limit on Training Error;
c) Limit on Training RMSE;
d) Limit on Training MXAE;
e) Limit on Training Rel. Variation;
f) Limit on Test Error;
g) Limit on Test RMSE;
h) Limit on Test MXAE;
i) Limit on Test Rel. Variation.

[178] Each "Rulebase" has a different number of functions; Author targets a reduction in the range between 10% and 20% of the "Rulebase" number of functions.

[179] This condition is a limit imposed over the number of iterations, the maximum error goal, or the maximum absolute or relative deviation (considering both the training and the test error).

The Author preliminary “end condition” setup imposes a limit of 25 iterations. Further Author researches may focus on the enhancements quantification of more complex and application customised application’s setups.

Not mandatory tool configurations are not considered in this dissertation, although they may potentially be used in further investigation.

7 Conclusions and Further Researches

The "Thesis" consists of three parts; the first part, which covers the first three chapters, introduces the work and describes a comprehensive picture that frames the Thesis (defines the technical proposal's background) and Author's researches. The second part includes "Chapter 4" and defines the theoretical framework of the technical proposal strategy. The third part consists of the "Chapter 5" and "Chapter 6" and debates the technical proposal.

Work's introduction starts with a quick overview of Unmanned Vehicles currently available technologies and future technological evolutions. The first introductive Chapter concludes with the analysis of the Author's investigations, where are exposed:

- Dissertation's Topicality;
- Dissertation's primary "hypothesis" and "intentions";
- Methods of Research and Development;
- Dissertation's scientific novelty;
- Dissertation's practical application of research results;
- Dissemination of research results.

The second Chapter gives a short exposition of conventional UVs control techniques with a focus on UAVs. Besides, the final introduction's stage scrutinises the current integration of autonomous technologies on next-generation automotive vehicles. Particular attention is given to the electrification process of automotive technologies and the contextual evolution of the functional safety requirements with the relative international legislation. The "Thesis" introduction produces the following avowals:

- AV's (autonomous vehicle) technology is developing extremely fast;
- AV's technology is the centre of enormous investments;
- a large number of big corporations are allocating a consistent part of their budget for AV's R&D;
- currently, a critical industrial effort is the validation[180] of the autonomous vehicle technology and the cost reduction of the system built with a large variety of "ECUs" (electronic control units);
- software development represents the highest R&D cost for AV's architecture.

[180] In fact, considerable risks are associated with future autonomous vehicles technology, in particular, the incomplete international legislation and international standards currently available;

The second part of the work builds up a theoretical framework for the application proposal associated with the Thesis. This section clearly defines the goals of the work and draws the strategy to achieve them. Significant attention is given to the academic research analysis of "fuzzy logic controllers" and "neuro-fuzzy controllers" applications and implementations. Efforts aim to construct the fundaments on which to build the controller's design strategy proposal. The outcomes of this preliminary study are:

- the final goal of the proposed work is to implement a controller capable of replaying the behaviour of a human pilot while he is driving a small RC aeroplane;
- a second target of the proposed work is to create a controller capable of being tuned in the function of the hardware (the small physical UAV, or RC aeroplane capable of autonomous operations);
- a "neuro-fuzzy controller"[181] is the control strategy aimed for the small UAV;
- the assumptions made is that the "neuro-fuzzy controller" has seven inputs and five outputs;
- a detailed study of the academic literature suggests the use of an FPGA to process the controller;
- the choice of FPGA is driven by its flexibility and by its capability to process multiple functions in parallel (parallel computation capability);
- VHDL will be used as the working platform for the systems[182], although the VHDL language imposes some limitations if compared with the flexibility and expressiveness of other fuzzy logic oriented languages;
- in order to achieve behavioural modelling, the Author's suggestion is to use a VHDL description style where the system's structure description (fuzzy sets, rule base) and the operator's description (connectivity, fuzzy operations) are defined separately (This makes it possible to describe both the fuzzy system structure and the processing algorithm independently);
- the description format makes it possible the use of linguistic hedges in order to compact the rules defining the system behaviour (A significant advantage of

[181] The ideal "neuro-fuzzy controller" shall be capable of parallel computation.

[182] It implies that the fuzzy system description must be synthesisable (a synthesisable VHDL algorithm requires to adapt and tune the characteristics of the controller) to the physical hardware implementation (FPGA printing).

using this approach is the availability of a tool able to translate a fuzzy logic oriented language with a GUI interface into a VHDL code);

- proposed work, to describe the fuzzy logic controller and then translate this description into a valid VHDL code, utilises the "XFUZZY XFL3.5" GUI (or XFL3) developed by Instituto de Microelectrónica de Sevilla (IMSE-CNM); [38]
- XFL3 description language is a development environment that eases the specification, verification and synthesis of fuzzy inference systems;
- a set of essential functions, called the XFL library, performs the parsing and semantic analysis of XFL specifications and stores them using an abstract syntax tree[183].

The third Thesis section moves from the hardware description of a small RC plane transformable into a small AUAV. The Hardware description covers the electronics hardware description and marginally the mechanical hardware description. The mechanical description includes the description of a simple RC plane powered by a low voltage REESS (according to "Reg.100" definition of low voltage REESS) and a set of two independent low voltage BLCD E-Motors. The description of the electronic systems results comprehensive due to the efforts made for the definition. The proposal's primary goal, which targets to move the design's load from the mechanical design to the controller's design, is the use of the neuro-fuzzy controller to adapt itself to the vehicle's characteristics, allowing the simplified mechanical design of the drone (or RC plane).

The core of the research work and of the technical proposal is the controller's design based on a multi-layer structure. The design starts with the identification of the "system's inputs", the "actuators", or "system outputs", and then links them with a "Transfer Function". Since that the proposed work is oriented on a neuro-fuzzy controller, the "System Transfer Function" is implemented by a specific set of MIFs, MOFs, FIS ("Rulebases") and a learning process from a training file[184].

The VHDL controller's multi-layer structure is associable with any hierarchical hardware schematics design. The "Top Layer" describes the peripherals interfaces and the interaction between the peripherals with the core of the flight controller.

[183] This format is used inside the environment when handling system descriptions.

[184] A simplified "Hedge Block" with rules and weights obtainable following a "Learning Process" and "Optimisation Process".

The first VHDL "Top Layer" section is populated with a set of independent blocks specifically designed to process the "System's Inputs". Each block, independently (in parallel), manages a determined sensor and then digital process the relative information before broadcasting the data to the "neuro-fuzzy controller" (the core of the controller).

The second section of the VHDL "Top Layer" algorithm incorporates the VHDL algorithms exported from the "XFUZZY GUI". The neuro-fuzzy flight controller is encased in a bespoke VHDL component called "NEURAL" (from Figure 6.63, VHDL component instance: "N0"), and its VHDL code represents a second hierarchical layer of the algorithm's structure. The "NEURAL" VHDL component" utilises a set of five sub-components, each of them built on a specific "Rulebase". For hierarchical systems, a VHDL description[185] is generated for each "Rulebase", which acts independently, and populates the linked sub-component. The VHDL code population of these sub-components embodies a third hierarchical layer of the physical controller structure.

The third VHDL "Top Layer" algorithm's section is populated with a set of independent components specifically designed to process the "system's outputs". Each VHDL component is associated with a single neuro-fuzzy controller's output. Each block, independently (in parallel), is interfaced with the neuro-fuzzy controller and digital processes the relative information before broadcasting to the electro-mechanical actuators the control signals.

The primary focus of the controller design is the "neuro-fuzzy unit". For this design, the XFUZZY GUI results exceptionally effective for the controller description, design, simulation, optimisation and the learning/training process. The technical proposal pre-requisites are:

- by assumption, a significant hardware design simplification (mechanical and electronic) is pursued;
- the final goal is to compensate the hardware simplification with a controller capable of being easily tuned and capable of learning;
- it is defined the mechanical hardware architecture as a baseline for the controller design;
- it is defined the electronic hardware architecture as a baseline for the controller design.

[185] It is the outcome of the translation in VHDL of the XFUZZY GUI neuro-fuzzy controller description.

"Chapter 6" investigations deliver the following significant results:

- the RTL views of the synthesisable system's VHDL code (Figure 6.63);
- a full description of the "neuro-fuzzy" controller;
- an optimisation strategy for the "Controller";
- a controller's learning/training execution strategy;
- a detailed simulation analysis of the raw, fuzzy flight controller.

With the presented researches and the derivative "neuro-fuzzy" controller's design, the Author aims a precise final delivery: demonstrate the feasibility of a flexible and cost-effective controller able to mimic the driving behaviour of a human pilot and also be capable of behaviours corrections with learning/training processes.

The Author relies on the simulation analysis presented in "Chapter 6" to demonstrate the controller's basic behaviours, flexibility, robustness, and potential future developments. Although the simulation analysis covers a wide range of cases, a particular emphasis is given to the controller behaviours during complex manoeuvres such as the take-off and the landing.

Take-off and landing due to the manoeuvres' complexity require a set of assumptions due to the absence of the physical model of the RC plane used. These assumptions are associated with a set of unique physical parameters that may be obtained from a learning/training process or from the mechanical hardware manufacturers datasheet.

For each analysed manoeuvre, the analysis describes a controller that takes time by time the expected decision, the expected decision that a human being pilot may most likely take if facing similar environmental conditions. **It results remarkable that a controller not yet optimised and not yet trained can produce such results.** Although the outcome is in line with the theoretical research done, it may be expectable that a series of learning/training operations may be necessary before establishing a system capable of performing a fully autonomous flight.

The description made gives all the information necessary to progress with the project and obtain the necessary economic resources required for the high development cost of such systems. The sourcing of all components required for the complete system's physical realisation is not the only cause of the delay, ensuring the proper testing environment and facilities allocation results being significant braking elements for physical tests implementation. Physical installation of the system in a real-world environment is currently underway and is a consequence of the Thesis work.

An Analysis and Conclusions of the proposed Control Strategy Quality:

the appraisal of the proposed "controller's quality" in the field of AUAV constructed on an adapted small RC plane is both varied and subjective. Many claims are made regarding the "controller's quality" to perform individual manoeuvres outside sets of pre-defined environmental conditions. The learning/training process limitation is that in front of unpredictable conditions, a learning/training process, by definition, is not applicable (by definition, it is not possible to define training for unpredictable conditions). It means that the Author cannot rely entirely on the learning process's contribution, although it is an important strategic asset.

So far as relates to the controller ability to perform manoeuvres under pre-defined environmental conditions where the landscape is known, where the vehicle position in the space is under control, and the target manoeuvre is known (it is intended a manoeuvre that a human being is capable of defining and then mimicking), it is possible to expect that the controller will, with a reasonably good margin of error, react accordingly.

The controller efficiencies may be affected by the accuracy/reliability of the particular type of sensors, the quality of the information associate with the "system's environmental variables" (or "global environmental information"). The learning/training process quality targets to mitigate such risk. With future researches is expected a "characterisation" of this mitigation factor.

Future research will involve the addition of a fully neural network fruit of the availability of a "physical vehicle"[186] and the results of a learning/training process described in "Chapter 6". Supplementary plans are to adapt different models of RC planes to carry the electronic hardware described in "Chapter 5" and then verify the flexibility of the controller and its capability to adapt to new mechanical characteristics using the learning/training process.

Future research including but not limited to:

- industrial applications of the "Neuro-Fuzzy Controller" built on FPGA;
- Cloud-based neural networks for industrial applications;
- Cloud-based neural networks for environment safety-critical monitoring;
- automotive application of "neuro-fuzzy controller" built on FPGA;
- FPGA automotive applications for ADAS;
- FPGA automotive applications for powertrain;

[186] It was not possible to implement due to the absence of an RC place capable of being modified to act as AUAV, and simultaneously capable of being driven by remote and store the flight diagnostics.

- automotive applications for Artificial Intelligence;
- self-driving system for automotive applications;
- data collection for pseudo-memory applications;
- practical applications for swarm robotics manipulation through memory harvesting;
- long-range exploration technologies for fully autonomous vehicles;
- safety modelling for closed environment robotics;
- an investigation into appropriate control methods for data access, including MOB, cloud or other access methods for single robots, swarm robots or remote exploration robots.

REFERENCES

[1] How J.P., Frazzoli E., Chowdhary G.V. (2015) Linear Flight Control Techniques for Unmanned Aerial Vehicles. In: Valavanis K., Vachtsevanos G. (eds) Handbook of Unmanned Aerial Vehicles. Springer, Dordrecht.

[2] Randel J. Gordon, USAF, "OPTIMAL DYNAMIC SOARING FOR FULL SIZE SAILPLANES", AIR FORCE INSTITUTE OF TECHNOLOGY, Wright-Patterson Air Force Base, Ohio (USA), September 2006.

[3] Saeed, Adnan & Bani Younes, Ahmad & Islam, Shafiqul & Dias, Jorge & Seneviratne, Lakmal & Cai, Guowei. (2015). A Review on the Platform Design, Dynamic Modeling and Control of Hybrid UAVs. 2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015.

[4] R. C. Nelson, Flight Stability and Automatic Control, 2nd ed., McGraw Hill, ch. 7-10, pp. 236-394.

[5] Girish C.V., Emilio F., Jonathan H.P., Hugh L. (2015) Nonlinear Flight Control Techniques for Unmanned Aerial Vehicles. In: Valavanis K., Vachtsevanos G. (eds) Handbook of Unmanned Aerial Vehicles. Springer, Dordrecht.

[6] Michael S. Branicky. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. IEEE Transactions on Automatic Control, 43(4):475{482, April 1998.

[7] Daniel Liberzon. Handbook of Networked and Embedded Control Systems, chapter Switched Systems, pages 559-574. Birkhauser, Boston, 2005.

[8] Wassim M. Haddad and Vijay Sekhar Chellaboina. Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach. Princeton University Press, Princeton, 2008.

[9] L. Rodrigues and J. P. How. Observer-based control of piecewise-affine systems. International Journal of Control, 76(5):459-477, 2003.

[10] García, Andrés Gabriel & Agamennoni, Osvaldo & Figueroa, Jose. (2009). Applying continuous piecewise linear approximations to affine non-linear control systems. IFAC Proceedings Volumes (IFAC-PapersOnline). 6. 114-119. 10.3182/20090616-3-IL-2002.0028.

[11] Andrés G. García and Osvaldo E. Agamennoni, (2008). Continuous Piecewise Linear Control for Nonlinear Systems: The Parallel Model Technique. WSEAS Conferences in Istanbul, Turkey, May 27-30, 2008, pages 39-44.

[12] L. R. Adrian, **D. Repole** and L. Rbickis, “Proposed neuro-guided learning for obstacle avoidance in AMBO a robotic device”, 2015 56th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), Riga, 2015, pp. 1-5.

[13] L. R. Adrian and **D. Repole**, “Intelligent autonomous environmental monitoring based on the AMBOA robot sensory system”, 2017 IEEE 58th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), Riga, 2017, pp. 1-6.

[14] McClelland, J. L. & Rumelhart, D. E. (1988). A simulation-based tutorial system for exploring parallel distributed processing. Behaviour Research Methods, Instruments & Computers, 2, 263-275.

[15] Reed, R. D., & Marks, R. J. (1999). Neural Smithing: supervised learning in feed forward artificial neural networks. Cambridge, MA: MIT Press.

[16] UN/ECE-R100/Rev.2, Regulation No. 100, Revision 2.

[17] **D. Repole** and L. R. Adrian, “Fuzzy nano piezo hybrid for fault detection in automotive power PCB”, 2017 IEEE 37th International Conference on Electronics and Nanotechnology (ELNANO), Kiev, 2017, pp. 400-404.

[18] Schulz-Harder, Jürgen. (2003). Advantages and new development of direct bonded copper substrates. Microelectronics Reliability. 43. 359-365. 10.1016/S0026-2714(02)00343-8.

[19] Inorganic Substrates for Power Electronics Applications, Anton Miric, M. Sc., Peter Dietrich, M Sc., M.A., Heraeus Deutschland GmbH and Co. KG, 63450 Hanau Germany.

[20] K. Tanaka (Translated by T. Niimura), 1997, “An Introduction to Fuzzy Logic for Practical Applications”, Springer- Verlag, New York, Ch. 4, 5, pp. 86-136.

[21] IEEE. IEEE Standard VHDL Language Reference Manual. IEEE, NY, 1988. IEEE Standard 1076-1987.5.

[22] Zadeh, L.A., "Fuzzy Sets," Information and Control, 8, 338-353, 1965.

[23] Brubaker, David I., "Fuzzy Logic Basics: Intuitive Rules Replace Complex Math," EDN, June 18, 1992, pp.111.

[24] Glenn A, “Fundamentals of Fuzzy Logic Part I & II”, SENSORS, April 1993.

[25] Earl Cox, The Fuzzy Systems Handbook (1994), ISBN 0-12-194270-8.

[26] SHABIUL ISLAM, NOWSHAD AMIN, M.S.BHUYAN, MUKTER ZAMAN “FPGA Realization of Fuzzy Temperature controller for industrial application” WSEAS

TRANSACTIONS on SYSTEMS and CONTROL Manuscript received June 16, 2007; revised Sep. 17, 2007.

[27] Brown, S.D., Francis, R.J., Rose, J., and Vranesic, Z.G., Field-Programmable Gate Arrays, Kluwer Academic Publishers,1996.

[28] K.T. Tho, K.H. Yeow, F. Mohd-Yasin, M.S. Sulaiman, and M.I. Reaz, VHDL Modeling of Boolean Function Classification Schemes for Lossless Data Compression, WSEAS Transactions on Computers, Vol.3, No.2, 2004, pp. 365-368.

[29] **D. Repole** and L. R. Adrian, "Introduction to Parallel MAS Control for MAS - Smart Sensor Networks", 2019 IEEE 60th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), Riga, Latvia, 2019, pp. 1-5.

[30] Bosque, Guillermo & del Campo, Inés & Echanobe, Javier. (2014). Fuzzy systems, neural networks and neuro-fuzzy systems: A vision on their hardware implementation and platforms over two decades. Engineering Applications of Artificial Intelligence. 32. 10.1016/j.engappai.2014.02.008.

[31] de Oliveira, José. (1995). A Design Methodology for Fuzzy System Interfaces. Fuzzy Systems, IEEE Transactions on. 3. 404 - 414. 10.1109/91.481949.

[32] A. Barriga, S. Sánchez-Solano, P. Brox, A. Cabrera, I. Baturone, "Modelling and implementation of fuzzy systems based on VHDL", International Journal of Approximate Reasoning, Volume 41, Issue 2, 2006, Pages 164-178, ISSN 0888-613X, https://doi.org/10.1016/j.ijar.2005.06.018.

[33] A. Zamfirescu, C. Ussery, VHDL and fuzzy logic if-then rules, in: Proceedings of the Euro-VHDL, Hamburg, 1992, pp. 636–641.

[34] D. Galan, C.J. Jimenez, A. Barriga, S. Sanchez-Solano, VHDL package for description of fuzzy logic controllers, European Design Automation Conference, Brighton, 1995, pp. 528–533.

[35] T. Hollstein, S.K. Halgamuge, M. Glesner, Computer-aided design of fuzzy systems based on generic VHDL specifications, IEEE Transactions on Fuzzy Systems 4 (4) (1996).

[36] E. Lago, C.J. Jimenez, D.R. Lopez, S. Sanchez-Solano, A. Barriga, Xfvhdl: A tool for the synthesis of fuzzy logic controllers, Design Automation and Test in Europe, DATE'98, Paris, 1998, pp. 102–107.

[37] F.J. Moreno Velo, S. Sanchez-Solano, A. Barriga, I. Baturone, D.R. Lopez, XFL3: A new fuzzy system specification language, WSES/IEEE Multiconference on Circuits,

Systems, Communications and Computers (CSCC'2001), Creta, July 2001, pp. 361–366.

[38] FUZZY LOGIC DESIGN TOOLS, Xfuzzy 25th, V. 3.5, March 2018, Instituto de Microelectrónica de Sevilla (IMSE-CNM), http://www2.imse-cnm.csic.es/Xfuzzy/Xfuzzy_3.5/download.html#DISTRIBUTION.

[39] ST Microelectronics, "A3G4250D :3-axis digital output gyroscope", Application Note: AN5148 (Rev.1 - May 2018).

[40] ST Microelectronics, Teseo-LIV3 GNSS module User Manual, UM2229 (Rev.4 - 17.12.2019).

[41] ST Microelectronics, LPS25HB MEMS pressure sensor: 260-1260 hPa absolute digital output barometer, datasheet – production data, (Rev.4 – 16.08.2016).

[42] **D. Repole** and L. R. Adrian, "Evaluation of GaN MOSFET for Unmanned Aerial Vehicles BLDC Motor Drive", 2018 IEEE 59th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), Riga, Latvia, 2018, pp. 1-4.

[43] Silvio Cammarata, "Reti Neuronali, Dal Perceptron alle reti caotiche e neuro-fuzzy", seconda edizione, ETASLIBRI, 1997.

[44] M. Brox, S. Sánchez-Solano, E. del Toro, P. Brox, F. J. Moreno-Velo CAD Tools for Hardware Implementation of Embedded Fuzzy Systems on FPGAs IEEE Transactions on Industrial Informatics 2012 DOI: 10.1109/TII.2012.2228871.

[45] F. Montesino, A. Lendasse, A. Barriga, Autoregressive time series prediction by means of fuzzy inference systems using nonparametric residual variance estimation Fuzzy Sets and Systems 2010, DOI: 10.1016/j.fss.2009.10.018.

[46] F. J. Moreno-Velo, I. Baturone, A. Barriga, S. Sánchez-Solano Automatic Tuning of Complex Fuzzy Systems with Xfuzzy Fuzzy Sets and Systems 2007 DOI: 10.1016/j.fss.2007.03.006.

[47] I. Baturone, F. J. Moreno-Velo, A. Gersnoviez, "A CAD Approach to Simplify Fuzzy System Descriptions 2006 IEEE International Conference on Fuzzy Systems", DOI: 10.1109/FUZZY.2006.1682033.

[48] Fullér R. (2000), "Artificial neural networks". Introduction to Neuro-Fuzzy Systems. Advances in Soft Computing, vol 2. Physica, Heidelberg. DOI: https://doi.org/10.1007/978-3-7908-1852-9_2.

[49] Vieira, José & Morgado-Dias, F. & Mota, Alexandre. (2004). Neuro-Fuzzy Systems: A Survey.

[50] Syed, M & Bensenouci, Ahmed & Alghamdi, Saleh & Ghany, A.M.. (2001). SHORT-TERM LOAD FORECASTING USING ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM (ANFIS) APPLICATION TO ALEPPO LOAD DEMAND.

[51] D. Nauck, F. Klawon; R. Kruse, “Foundations of Neuro-Fuzzy Systems”, J. Wiley & Sons, 1997.

[52] B. Kosko, “Neural Networks and Fuzzy Systems: A Dynamical System Approach to Machine Intelligence”, Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[53] E. Czogala and J. Leski, “Neuro-Fuzzy Intelligent Systems, Studies in Fuzziness and Soft Computing”, Springer Verlag, Germany, 2000.

[54] Szandała, Tomasz. (2015). Classification Based on Lingual Variables Using Expert Matrix Obtained with Genetic Algorithm. Procedia Computer Science. 71. 143-149. 10.1016/j.procs.2015.12.180.

[55] J.M.Zurada, Introduction to Artificial Neural Systems (West Publishing Company, New York, 1992).

[56] Robert Fullér, Neural Fuzzy Systems, Donner Visiting professor Abo Akademi University, ISBN 951-650-624-0 ISSN 0358-5654, 1995, pages. 157-160.

[57] D. Nauck, F. Klawon; R. Kruse, “Foundations of Neuro-Fuzzy Systems”, J.Wiley & Sons, 1997.

[58] T. C. Lin, C. S. Lee, “Neural Network Based Fuzzy Logic Control and Decision System”, IEEE Transactions on Computers, 1991, Vol.40, no. 12, pp. 1320-1336.

[59] R. Jang, “Neuro-Fuzzy Modelling: Architectures, Analysis and Applications”, PhD Thesis, University of California, Berkley, July 1992.

[60] H. R. Berenji and P. Khedkar, “Learning and Tuning Fuzzy Logic Controllers through Reinforcements”, IEEE Transactions on Neural Networks, 1992, Vol. 3, pp. 724-740.

[61] D. Nauck, R, Kurse, “Neuro-FuzzySystems for Function Approximation”, 4th International Workshop Fuzzy-Neuro Systems,1997.

[62] S. Tano, T. Oyama, T. Arnould, “ Deep Combination of Fuzzy Inference and Neural Network in Fuzzy Inference”, Fuzzy Sets and Systems, 1996, Vol. 82(2), pp. 151- 160.

[63] S. Sulzberger, N. Tschichold e S. Vestli, “FUN: Optimization of Fuzzy Rule Based Systems Using Neural Networks”, Proceedings of IEEE Conference on Neural Networks, San Francisco, March 1993, pp. 312-316.

[64] F. C. Juang, T. Chin Lin, “An On-Line Self Constructing Neural Fuzzy Inference Network and its applications”, IEEE Transactions on Fuzzy Systems, 1998, Vol. 6, pp. 12-32.

[65] M. Figueiredo and F. Gomide; "Design of Fuzzy Systems Using Neuro-Fuzzy Networks", IEEE Transactions on Neural Networks, 1999, Vol. 10, no. 4, pp.815- 827.

[66] N. Kasabov e Qun Song, “Dynamic Evolving Fuzzy Neural Networks with ‘m- out-of-n’ Activation Nodes for On-Line Adaptive Systems”, Technical Report TR99/04, Department of Information Science, University of Otago, 1999.

APPENDICES

A. Abbreviations

ABC	→ Artificial Bees Colony
ACE	→ Advanced Classification Error
ADAS	→ Advanced Driver Assistant
Ah	→ Ampere-hours
AI	→ Artificial Intelligence
Al_2O_3	→ Aluminium Oxide
AlN	→ Aluminium Nitride
AMB	→ Active Metal Brazing
ANN	→ Artificial Neural Network
ARB	→ Assessment Rules Block
ARMA	→ Autoregressive Moving Average
ASIC	→ Application-Specific Integrated Circuit
AR	→ Autonomous Robots
AUAV	→ Autonomous Unmanned Arial Vehicle
AUV	→ Autonomous Underwater Vehicles
AV	→ Autonomous Vehicles
BDU	→ Block Data Update
BLDC	→ Brushless DC Electric Motor
BMS	→ Battery Management System
CE	→ Classification Error
CEP	→ Circular error probable
CLB	→ Configurable Logic Block
CPWL	→ Continuous Piecewise Linear Approximation
COG	→ Centroid Method (Fuzzy Logic)
CSE	→ Classification Square Error
CTDs	→ Conductivity-Temperature-Depth sensors
CVT	→ Continuously Variable Transmission
DBC	→ Direct bonded copper

DC	→ Direct Current
DCU	→ Drive Control Unit
DES	→ Decentralized Control System
DM	→ Distance Modes
DPS	→ Degree Per Second
DSP	→ Digital Signal Processor
EBM	→ Electronic Battery Monitor
ECU	→ Electronic Control Unit
EDA	→ Electronic Design Automation tools
e.g.	→ exempli grata - the abbreviation of a Latin phrase meaning: “for example”
E-Motor	→ Electric Powertrain’s Motor
ESD	→ Electro-Static Discharge
EV	→ Electric Vehicle
FAR	→ Federal Aviation Regulations
FFT	→ Fast Fourier Transform
FIR	→ Finite Impulse Response
FIS	→ Fuzzy Inference System
FMEA	→ Failure Modes and Effects Analysis
FOB	→ Fuzzy Output Block
FoV	→ Field of View
FPD	→ Field Programmable Device
FPAA	→ Field Programmable Analog Array
FPGA	→ Field Programmable Gate Array
FR	→ Front Engine – Rear-Wheel Drive
FR4	→ Flame Retardant, glass-reinforced epoxy laminate material
GA	→ Genetic Algorithms
GaN	→ Gallium Nitride
GBP	→ Gain Bandwidth Product
GNSS	→ Global Navigation Satellite System
GUI	→ Graphical User Interface
HEV	→ Hybrid Electric Vehicle
HLGA	→ Holed Land Grid Array

HSD	→ TOYOTA's Hybrid Synergy Drive
HV	→ Hybrid Vehicle
HW	→ Hardware
ICA	→ Independent Component Analysis
ICNN	→ Independent Component Neural Network
ICE	→ Internal Combustion Engine
I^2C	→ Inter-Integrated Circuit, serial communication protocol
IC	→ Integrated Circuit
i.e.	→ id est - the abbreviation of a Latin phrase meaning: "in other words"
IGBT	→ Insulated-Gate Bipolar Transistor
iPM	→ Intelligent Power Management
ISA	→ International Society of Automation - Standard
ISH	→ Industrial Service Hybrid Robots
ITAR	→ International Traffic in Arms Regulations
KL15	→ Terminal 15 or "run bus" which corresponds to the ignition position 1
KL30	→ Terminal 30 or "battery bus" which corresponds to the ignition position 2
LBC	→ Learning-Based Control
LGA	→ Land Grid Array
Li-Ion	→ Lithium-ion
LMI	→ Linear Matrix Inequalities
LNA	→ Low-Noise Amplifier
LPV	→ Linear Parameter Varying
LST	→ Least Significant Bit
LTI	→ Linear Time-Invariant
MAE	→ Mean Absolute Error
MCU	→ Microprocessor Control Unit
MEMS	→ Microelectromechanical Systems
MFC	→ Membership Functions Circuits.
MHEV	→ Mild Hybrid Electric Vehicle
MIF	→ Membership Input Function
MLP	→ Multilayer Perceptron
MOA	→ Midpoint of Area

MOB	→ on-board memory size expressed in bytes
MOF	→ Membership Output Function
MOM	→ Medium of Maxima
MPPT	→ Maximum Power Point Tracker
MSB	→ Most Significant Bit
MSE	→ Mean Square Error
MXAE	→ Maximum Absolute Error
NiMH	→ Nickel-Metal Hydride Battery
NMEA	→ National Marine Electronics Association
NN	→ Neural Network
NRR	→ Number of Registers Read
OA	→ Obstacle Avoidance
ODR	→ Output Data Rate
OEM	→ Original Equipment Manufacturer
OMF	→ Output Membership function
OR	→ Obstacle Recognition
PC	→ Personal Computer
PCB	→ Printed Circuit Board
PHEV	→ Plug-In Hybrid Electric Vehicle
PID	→ Proportional-Integral-Derivative controller
PFC	→ Power Factor Correction
PIR	→ Pyroelectric Infrared Radiation Sensor
PP	→ Polypropylene Film capacitor
PMSM	→ Permanent Magnetic Synchronous Motor
PSU	→ Power Supply Unit
PV-module	→ Photo-Voltaic cells mounted in a frame for work installation
PWL	→ Piece Wise Linear models
PWM	→ Pulse Width Modulation
P/N	→ Manufacturer’s Part Number
RC	→ Radio-Controlled
REESS	→ Rechargeable Energy Storage System
REOMP	→ Reconfigurable Orthogonal Multi-processor Memory

RMSE	→ Root Mean Square Error
ROI	→ Region of Interest
ROM	→ Remotely Operated Machine
ROUV	→ Remotely Operated Underwater Vehicles
ROV	→ Remotely Operated Vehicle
RTL	→ Register-Transfer Level (VHDL schematics view)
R&D	→ Research and Development
Si	→ Silicon
Si_3N_4	→ Silicon Nitride
SiC	→ Silicon Carbide
SIM	→ Serial Interface Mode
SM	→ State Machine
SoC	→ State of Charge
SoF	→ State of Function
SoH	→ State of Health
STNFC	→ Self-Tuning Non-linear Function Control
SW	→ Software
ToF	→ Time-of-Flight
ToV	→ Field of view
UAV	→ Unmanned Arial Vehicle
UART	→ Universal Asynchronous Receiver-Transmitter
UGV	→ Unmanned Ground Vehicle
USAF	→ The United States Air Force
UUGV	→ Utility Unmanned Ground Vehicle
UUV	→ Unmanned Underwater Vehicles
UV	→ Unmanned Vehicle
V_a	→ Vehicle's Velocity in the moment of maximum take-off acceleration
VAR	→ Variable Resistor
VLSI	→ Very Large Scale Integration
VMC	→ Minimum control speed,
VHDL	→ Very High-Speed Integrated Circuit Hardware Description Language
V_{FC}	→ Final Cruise Velocity

V_{SD}	→ Starting Descending Velocity (during the landing manoeuvre)
V_{TD}	→ Touch-down Velocity
WBG	→ Wide Band Gap
WMAE	→ Weighted Mean Absolute Error
WMSE	→ Weighted Mean Square Error
ZTA	→ Zirconia Toughened Alumina
3ph	→ 3 phase (electric motor)

B. Non-Linear System Linearization Technique

A common approach could be to achieve a "Non-Linear Controller" by approximating the "Non-Linear System" with a linear one. To reach this goal could be helpful the use the **Hartman-Grobman Theorem, which states that:**

If the Jacobian of the system $\dot{X}(t) = f(X)$ possess no pure complex or zero eigenvalues, and the system can be locally represented by a linear approximation (Guckenheimer-Holmes).

One method could be utilising a linear system to design the control law for the non-linear one. For example, for a system:

$$\dot{X}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} cos(\vartheta) \cdot u_1 \\ sin(\vartheta) \cdot u_1 \\ u_2 \end{bmatrix}, \qquad U(t) = [u_1, u_2]$$

(Equation 54)

Could be approximated to a new System, such as:

$$\dot{\tilde{X}}(t) = A \cdot \tilde{X}(t) = B \cdot \tilde{U}(t)$$

(Equation 55)

The new system should guarantee the necessary accuracy for the correct "System Working", the accuracy is inversely proportional to the value "ε", and it is defined as:

$$X(t) - \tilde{X}(t) \leq \varepsilon$$

(Equation 56)

The problem could be to understand if the vector fields are close enough to ensure the right trajectory bound. Practically:

$$\begin{matrix} X(t) = f(X, U) \\ F(X, U) - (A \cdot X + B \cdot U) \leq \varepsilon \end{matrix} \quad \Leftrightarrow \quad X(t) - \tilde{X}(t) \leq \varepsilon$$

(Equation 57)

The problem is that with the traditional Jacobian approximation, there is no general rule that can determine an accurate region of validity. An approach could be to use only the vector field information (not considering the trajectories).

$$X(t) = f(X, U)$$

(Equation 58)

$$F(X, U) - (A \cdot X + B \cdot U) \leq \varepsilon$$

(Equation 59)

Then could be used the **Theorem of Taylor,** which says: given a function of several variables $f(X, U)$ the polynomial vector field that better approximate f in a set point (X^*, U^*) is given by:

$$f(X^*, U^*) + \frac{\partial f(X, U)}{\partial X} \cdot (X - X^*) + \frac{\partial f(X, U)}{\partial U} \cdot (U - U^*) + o(2)$$

(Equation 60)

In a general linear system, there are some points of particular interest, called equilibrium points; where for equilibrium point is intended:

$$\dot{X}(t) = 0 = f(X^*, U^*)$$

(Equation 61)

The equilibrium point research is vital to define a correct approximation function; indeed, at the equilibrium point, the states and controls reach the reference. In the specific are researched the matrices:

$$A = \left.\frac{\partial f(X, U)}{\partial X}\right|_{X^*, U^*}$$

(Equation 62)

$$B = \left.\frac{\partial f(X, U)}{\partial U}\right|_{X^*, U^*}$$

(Equation 63)

It is expected that the matrix A is null and to obtain valuable matrices for the research of the equilibrium points, could be necessary to operate a change of coordinate, as could be new variables as $e(t)$, $\overline{U}(t)$ and $\overline{U}_r(t)$. Thus the new system could be written as:

$$\dot{e}(t) = A \cdot e(t) + B \cdot \overline{U}(t)$$

(Equation 64)

Then the following matrices are defined:

$$A^* = \frac{\partial f(e,\bar{U})}{\partial e}\Big|_{e^*=0,\bar{U}^*=0}$$

(Equation 65)

$$B^* = \frac{\partial f(e,\bar{U})}{\partial \bar{U}}\Big|_{e^*=0,\bar{U}^*=0}$$

(Equation 66)

Once it is defined, an LTI system could utilise the pole placement (for instance):

$$\bar{U} = K \cdot e$$

(Equation 67)

Thus it is possible to rewrite the system, according to the relation:

$$\dot{e}(t) = [A^* + B^* \cdot K] \cdot e(t)$$

(Equation 68)

Then it is possible to define the Matrix $A_{lc} = [A^* + B^* \cdot K]$ and recall that in order to find K, in that way it is necessary to calculate the eigenvalues of A_{lc}. To anticipate the existence of the matrix K, could be used the **"Kalman Theorem"** which states that: the pole-placement problem (find matrix K) possesses a solution if the matrix "C" has full-rank. Practically the Theorem states that:

$$X(t) = (A + B \cdot K) \cdot X \;\Leftrightarrow\; C = [B \quad A \cdot B \quad A^2 \cdot B \quad A^3 \cdot B \quad \cdots \quad A^{n-1} \cdot B], \qquad A \in \S^{n\,x\,n}$$

(Equation 69)

Which matrix C is called the *controllability matrix.* "Matlab" allows tools to check controllability, and a way to realize that could be to lose rank through taking both reference controls null.

C. Lyapunov Theorem

Supposing an autonomous non-linear dynamic system: $\dot{x} = f\big(x(t)\big)$ and, $x(0) = x_0$. Where, $x(t) \in \mathbb{D} \subseteq \mathbb{R}^n$ denotes the system state vector, $\mathbb{D}$ an open set containing the origin and, $f: \mathbb{D} \to \mathbb{R}^n$ continuous on $\mathbb{D}$.

Presuming that f has an equilibrium at x_e so that $f(x_e) = 0$, then this equilibrium is alleged to be Lyapunov stable, if, for every $\varepsilon > 0$, there exists a $\delta = \delta(\varepsilon) > 0$, such that, if $\|x(0) - x_e\| < \delta$ for every $t > 0$, will result $\|x(t) - x_e\| < \varepsilon$.

The equilibrium of the above system is said to be asymptotically stable if it is Lyapunov stable and if there exists, $\delta > 0$ such that if, $\|x(0) - x_e\| < \delta$, then $\lim_{t\to\infty} \|x(t) - x_e\| = 0$.

The equilibrium of the above system is said to be exponentially stable if it is asymptotically stable and if there exists, $\alpha, \beta, \delta > 0$ such that if, $\|x(0) - x_e\| < \delta$, then, $\|x(0) - x_e\| < \alpha \cdot \|x(0) - x_e\| \cdot e^{-\beta t}$ for $t \geq 0$.

Practically, the meanings of the above terms are the following: Lyapunov stability of at an equilibrium means that solutions starting "close enough" to the equilibrium (within a distance δ from it) remain "close enough" forever (within a distance ε from it)[187]. Asymptotic stability means that solutions that start close enough, not only remain close enough but also eventually converge to the equilibrium. Exponential stability means that solutions not only converge but converge faster than or at least as fast as a mainly known rate $\alpha \cdot \|x(0) - x_e\| \cdot e^{-\beta t}$.

Designates $y(t)$ the system output, the consequent trajectory x is (locally) attractive if $\|y(t) - x(t)\| \to 0$ for $t \to \infty$ for all trajectories that start close enough, and globally attractive if this property holds for all trajectories.

That is, if x belongs to the interior of its stable manifold, it is asymptotically stable if it is both attractive and stable.

D. Lyapunov Stability Technique

The first step to describe the Lyapunov Stability Techniques is to describe the Lyapunov Functions and Theorems. The first Lyapunov function is called "Weak Lyapunov Function", and the function $V(X)$ is addressable as a "Weak Lyapunov Function" only if:

$$V(0) = 0$$

(Equation 70)

$$V(X) > 0, \forall X \neq 0$$

(Equation 71)

$$V(X) \in \S^1\ (B_r)$$

(Equation 72)

$$\frac{\partial V(X)}{\partial X} \cdot f(X) \leq 0, \qquad \forall\, X \in (B_r)$$

(Equation 73)

[187] It must be true for any ε chosen.

Moreover, the equilibrium point is zero where B_r is the "ball" of radius r, described according to the following relation:

$$B_r = \{X : \|X\| \leq r\}$$

(Equation 74)

To be valid the last statements, the function $V(X)$ should be semi-definite negative along trajectories of the system $\dot{X}(t) = f(X)$, it means that:

$$\dot{V}(t) \leq 0$$

(Equation 75)

Similarly, it is possible to define the "Lyapunov Function", which differs from the "Weak Lyapunov Function" described by the system of Equations built with the Equations 70, 71, 72 and 73. The "Lyapunov Function" differs by a strict inequality of "Equation 73", which is replaced with "Equation 76".

$$\frac{\partial V(X)}{\partial X} \cdot f(X) < 0, \qquad \forall\, X \in (B_r)$$

(Equation 76)

Once that are defined the Lyapunov functions, it is possible to analyse the Lyapunov theorems. **The "First Lyapunov Theorem" says that:** if exists a Lyapunov function smooth and weak, then the system is "Lyapunov Stable". It means that finding $\dot{X}(t) = f(X)$, it ensures stability but not asymptotic stability. In order to guarantee the *asymptotic stability*, *it is necessary to* use the "**Second Theorem of Lyapunov,** which states that: if there exists a "Lyapunov Function" smooth, then the system $\dot{X}(t) = f(X)$ is asymptotically stable. This powerful theorem does not indicate how to obtain the function $V(X)$; therefore, a system might be stable without the "Lyapunov Function". In the study case for the instance:

$$\begin{cases} \dot{x}_1(t) = -x_1(t) - x_1(t)x_2(t)^2 \\ \dot{x}_2(t) = -x_2(t) - x_2(t)x_1(t)^2 \end{cases}$$

(Equation 77)

If the origin is an equilibrium point, $x_1^* = 0$ and $x_2^* = 0$, it will be possible to state that:

$$\begin{cases} 0 = -x_1^* - x_1^* \cdot x_2^{*2} \\ 0 = -x_2^* - x_2^* \cdot x_1^{*2} \end{cases}$$

(Equation 78)

Then be considered the "Lyapunov Function" $V(x_1, x_2)$ that should satisfy the relations of the "Lyapunov Function" (Equations 70, 71, 72 and 76). It means that:

$$\frac{\partial V(X)}{\partial x} \cdot f(X) = 2x_1(-x_1(t) - x_1(t) \cdot x_2(t)^2) + 2x_2(-x_2(t) - x_2(t) \cdot x_1(t)^2)$$

(Equation 79)

Thus it will be possible to write:

$$\frac{\partial V(X)}{\partial x} \cdot f(X) = -2(x_1(t)^2 + x_2(t)^2 - x_1(t)^2 \cdot x_2(t)^2) < 0, \forall\, (x_1, x_2) \neq (0,0)$$

(Equation 80)

If the last relation is verified, it will affirm that the System is "asymptotically stable". A helpful way to produce the "Lyapunov Function" could be to redefine the system, through mathematical artifices, in order to redesign the system in a way that could fit with the definition of the "Lyapunov Function" and "Lyapunov Theorem".

E. CPWL Function introduction

A piecewise linear function is a function composed of some number of linear segments defined over an equal number of intervals, usually of equal size. The process-model output using the CPWL approximation is defined as:

$$y_{mp} \leq \hat{h}_{mp}(\vartheta(t)) = \Theta^T \Lambda(\vartheta(t))$$

(Equation 81)

Where, $\Theta^T \in \Re^{\sigma+1}$ and $\Lambda \in \Re^{\sigma+1}$.

Using the CPWL approximation, any non-linear function "h" can be uniquely represented by the segmentation of its input domain. Let consider the segmentation into σ segments by the parameters α_i , with $\alpha_0 \leq \alpha_1 \leq \ldots \leq \alpha_\sigma$. Additionally, the elements of the primary function can be expressed as:

$$\Lambda = \begin{bmatrix} 1 \\ \frac{1}{2} \cdot (\vartheta - \alpha_0 + |\vartheta - \alpha_0|) \\ \vdots \\ \frac{1}{2} \cdot (\vartheta - \alpha_{\sigma-1} + |\vartheta - \alpha_{\sigma-1}|) \end{bmatrix}$$

(Equation 82)

At the same time, the parameters' vector is : $\Theta^T = [\vartheta_0, \vartheta_1, \ldots, \vartheta_\sigma]$. Clustering algorithms choose the segments' locations, and the vector of the parameters can be calculated using standard least-square algorithms.

F. Fuzzy logic Introduction

The Author of the fuzzy logic was usual to describe his invention with the phrase: <<…computing with words…>>. It was easy to understand, by that definition of fuzzy logic, that this kind of logic is exciting and innovative because it uses qualitative inferences in the design of artificial systems (control or decision support) when the mathematical model is

unknown or does not exist or is too complex to run appropriately in real-time. The main target of fuzzy logic was to find solutions to problems, even complex, through the use of empirical and qualitative rules that affect a world of "grey" or "fuzzy" (hence the term fuzzy logic) actions, instead of the logic "white" or "black". In practice, the traditional logic is characterised by a "bivalent logic", which associates with each element a value that can be "0" or "1", so that indicates that belonging to a given set is true or false. In contrast, the fuzzy logic is "polyvalent", i.e. the degree of membership (Membership) M_I (X) of an element "X" to a fuzzy set "I" can assume any value in the range between 0 and 1. "Membership Function" is defined as the relation that represents this kind of memberships. Those functions are designed upon expert's recommendations or, in the most elementary case, using accessible empirical functions dictated by common sense. These functions could take many forms, but for less complicated cases are preferable to use only triangles and trapezoids. [43]

Usually, the design of fuzzy algorithms is achieved in three steps:

a) acknowledgement of "Membership Input Functions" ("MIF" - fuzzification);
b) acknowledgement of the "FIS";
c) acknowledgement of "Membership Output Functions" ("MOF" - defuzzification).

Fuzzy Logic Membership Input Functions

The "Membership Input Functions" (MIFs) are most commonly associated with physical and sometimes non-physical variables and therefore are not strictly "fuzzified values", but almost undoubtedly, numerical values can be referred to as "crisp parameters". It is necessary to convert each numeric value to the corresponding input fuzzy sets, or in other words, convert to an input fuzzification.

By an input with a generic value x_0 and a fuzzy set A, there is an establishment of a degree of truth of A not exceeding $M_A(x_0)$ and with a sub-set A' of A having as a maximum ordinate $M_A(x_0)$; as illustrated in [43].

$$M_{A'(x)} = MAX\big(M_A(x_0), M_A(x)\big)$$

(Equation 83)

What this means is that if the membership function input is a triangle $M_A(x_0)$, then $M_{A'}(x)$ will be a trapezoid, and this trapezoid will have a maximum value $MAX\big(M_A(x_0)\big)$ which is valid if $0 \leq MAX\big(M_A(x_0)\big) \leq 1$.

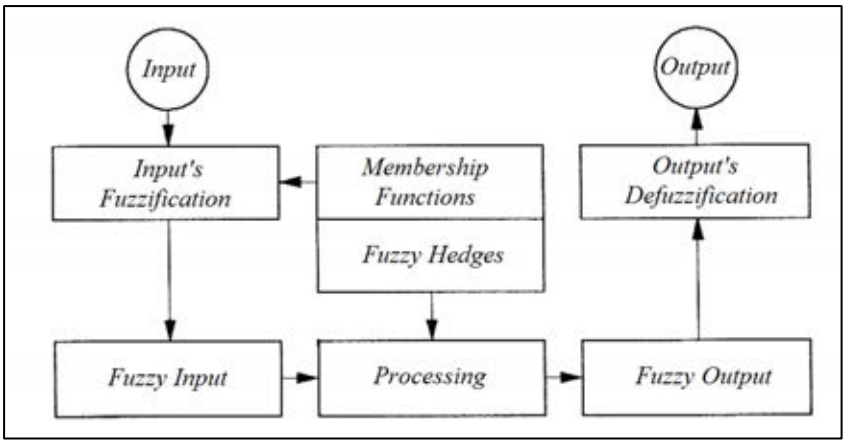


Figure F.1: General purpose fuzzy controller flow chart. [43]

In a standard process, the received crisp values will be the generic values and the input membership functions A that return fuzzy sets triggered by those values A'. In practice, rather than activated sets A' it is preferable to use their maximum degree of truth $MAX\big(M_{A'}(x_0)\big)$, which incidentally coincides with $MAX\big(M_A(x_0)\big)$, with the last result being defined as the fuzzy input. [43]

Fuzzy Hedges

In front of a fully defined fuzzy input set, those values can be computed within a block of assessment rules where each combination of fuzzy input activates a particular rule. To every single rule corresponds a particular degree of activation (weight). This value could be equal to the minimum degree of truth of the fuzzy input sets that define the combination. What illustrated describes a truth table, where to each combination of the fuzzy sets is associated with a unique weight value, which in turn activates a fuzzy set for each type of output with a certain degree of truth. What described is called "Fuzzy Inference", precisely defined as the process that receives the fuzzy controller's inputs and, through the use of the "Fuzzy Rules", returns the fuzzy output sets inferred [43].

In the fuzzy logic environment design tool Xfuzzy 25th (V. 3.5), the "Fuzzy Inference" (FIS) is described by a, or by an assembly of many, "Rulebase".

Fuzzy Output Defuzzification

The union of all output sets defines the "Membership Output Functions" (MOFs). The "Fuzzy Output Block" (FOB) elaborates the output data of the "Assessment Rules Block" (ARB), according to a specific method. Whatever is the method used, in the case of a set

associated with more degrees of truth, by definition, the target is to maximise the values. Generally, most popular approaches used are: the "**composition method**", where the fuzzy output sets obtained are the subject of a logical "OR operation", and the "**sum composition method**", where the fuzzy output sets obtained are, simply, added together.

There is a final step required to get a usable output function, and this is called defuzzification. In this step, a determination is made in establishing the numerical value most representative of the whole final output through the use of a specific method. The most common methods are:

- the **"Centroid method"** (COG) where the defuzzification takes as output value the centroid abscissa of the solid figure bounded by all fuzzy output;
- the **"MAX method"** where the defuzzified output value corresponds with the maximum of the output;
- the **"Medium of Maxima method"** (MOM), where the defuzzified output value is the average of the values corresponding to the maximum of the output.

[43]

G. Fuzzy Logic applications in smart electrical systems

Nowadays, there are ceaseless applications of fuzzy logic in endless fields because this control approach is giving excellent feedbacks, especially for applications on which the process is not available or not modelled (in part or in full), or it is affected by disturbs due to external variables that can influence the model. In fact, in order to achieve an accurate, reliable and stable control for a complex system, the mathematical model $P_{(s)}$ that describes the physical system process could be not appropriate; because it is based on a specific set of hypothesis, and, usually, it is calculated with approximation under specific environmental conditions. It means that should be used for the control design the process:

$$P_{D(s)} = \left\{ \tilde{P}_{(s)} = P_{(s)}\left(1 + \Pi_{(s)}\Delta_{(s)}\right), \left\|\Delta_{(s)}\right\|_{\infty} \leq 1 \right\}$$

(Equation 84)

where $\Pi_{(s)}$ is a weight function and $\Delta_{(S)}$ is an adaptive function with a resonance peak ≤ 1; both functions could have only negative poles and zeros.

It is easy to understand how it could be useful to use fuzzy logic when $1 + \Pi_{(s)}\Delta_{(S)}$ is not very small. Therefore, the control theory and stability theory are based on the LTI hypothesis. Discussed particulars result in a primary advantage of the fuzzy logic control method, making the fuzzy logic a competitive choice when the "Lyapunov Theorem" results

too complicated to comply, or there is not enough information available for the implementation of the “Lyapunov Stability Technique”. In this case, fuzzy logic is very useful for values of ε that are not extremely small.

Some interesting examples of fuzzy logic applications are:

- fuzzy control design for gas absorber system;
- large scale fuzzy controller (Appliances);
- PFC;
- trending and prediction;
- biomedical applications;
- ground vehicle engineering;
- smart modelled fuzzy logic “Maximum Power Point Tracker” (MPPT) for photovoltaic applications;
- application of fuzzy logic in smart distributed power systems or micro-grids with a high penetration of renewable energy.

H. Neuro-Fuzzy introduction

Artificial neural systems’ interpretation may concur to simplified mathematical models of brain-like systems, functioning as parallel distributed computing networks. Nevertheless, creatively to traditional computers, which are programmed to perform a specific task, most neural networks must be prepared or trained. They can acquire new associations, new functional boundaries and new patterns. Although computers outclass both biological and artificial neural systems for tasks based on well-defined and fast arithmetic operations, artificial neural systems express the promising new generation of information processing networks. [48]

The modern techniques of artificial intelligence have potential applications in almost all fields of human knowledge. Despite a great emphasis given to the fundamental sciences, perhaps the most noticeable explanation of the success of these techniques is in the engineering field. Combining the two techniques, neural networks and fuzzy logic, is often preferred for solving engineering problems where the traditional techniques do not provide a comfortable and accurate solution. The neuro-fuzzy term was born by the fusing of these two techniques. As each researcher combines these two tools differently, some confusion occurs on the terminology meaning. Still, there is no absolute consensus, but in general, the neuro-fuzzy term means a type of system characterised for a similar structure of a fuzzy controller

where the fuzzy sets and rules are adjusted using neural networks iteratively tuning techniques with data vectors (input and output system data). The before-mentioned systems show two well-defined behaviours. In the first phase, called the learning phase, it behaves like neural networks that learn their internal parameters offline. Later, in the execution phase, it behaves like a fuzzy logic system. Separately, each of these techniques possesses advantages and disadvantages that, when combined, their cooperage provides better results than the ones achieved using each isolated technique.

After that, the fuzzy systems become successful in industrial applications; the common perception was of a complicated development for designing a fuzzy system with good performance. The problem of finding membership functions and appropriate rules is frequently a tiring process of attempt and error. This lead to the idea of applying learning algorithms to fuzzy systems. The neural networks that have efficient learning algorithms had been presented as an alternative to automate or to support the development of tuning fuzzy systems. The earliest significant studies of the neuro-fuzzy systems date back to the beginning of the 90's decade, and the most significant examples are **Jang, Lin** and **Lee** in 1991, **Berenji** in 1992 and **Nauck** from 1993. The majority of the first applications were in process control. Gradually, its application spread for all the areas of human knowledge, and in particular to data analysis, data classification, imperfections detection and support to decision-making. Neural networks and fuzzy systems can be combined to join their advantages and to cure their defectiveness. Neural networks introduce its computational characteristics of learning in the fuzzy systems and receive from them the interpretation and clarity of systems representation. The capacities of the neural networks compensate for the disadvantages of the fuzzy systems. These techniques are complementary, making practical concurrent use. [49]

Definition of Neuro-Fuzzy modules

Fuzzy logic controllers' tuning methods are an evolution of fuzzy logic. The neuro-fuzzy controller uses the neural network learning techniques to tune the membership functions while keeping the semantics of the fuzzy logic controller intact. Neural networks offer the possibility of solving the problem of tuning. Although a neural network can learn from the given data, the trained neural network resembles a black box. Neither can it be possible to extract structural information from the trained neural network, nor can it integrate certain information into the neural network to simplify the learning procedure.

Contrariwise, a fuzzy logic controller, by assumption, has to work with structured knowledge in the form of rules, and nearly everything in the fuzzy system remains highly transparent and easily interpretable. However, there exists no formal framework for the choice of various design parameters and optimisation of these parameters generally is done by trial and error. [48 and 50]

A combination of neural networks and fuzzy logic offers the possibility of solving tuning problems and design difficulties of fuzzy logic. The resulting network will be more transparent and observable in the form of fuzzy logic control rules or semantics. This approach combines the well-established advantages of both methods and avoids the drawbacks of both. [48]

In general, all the combinations of techniques based on neural networks and fuzzy logic can be called neuro-fuzzy systems. The different combinations of these techniques could be classified, as illustrated by [51], in the following classes:

- **Cooperative Neuro-Fuzzy System**, where there is a pre-processing phase where the neural networks mechanisms of learning determine some sub-blocks of the fuzzy system; [188]
- **Hybrid Neuro-Fuzzy System,** in this category, a neural network is used to learn a wide range of parameters of the fuzzy system (parameters of the fuzzy sets, fuzzy rules and weights of the rules); [189]
- **Concurrent Neuro-Fuzzy System**, where the neural network and the fuzzy system work continuously together (in general, the neural networks pre-processes the inputs, or post-processes the outputs, of the fuzzy system).

Definition of Genetic Algorithm

"Genetic Algorithms" (GA) were invented to mimic some of the processes observed in natural evolution. Many people, biologists included, are astonished that life at the complexity level that we observe could have evolved in the relatively short time suggested by the fossil

[188] For instance, the fuzzy sets and (or) fuzzy rules (fuzzy associative memories [52] or the use of clustering algorithms to determine the rules and fuzzy sets position [53]). Networks pre-process the inputs (or post-processes the outputs) of the fuzzy system. In effect, after that, the fuzzy sub-blocks are calculated, the neural network learning methods are taken away, executing only the fuzzy system. [50]

[189] The majority of the researchers uses the neuro-fuzzy term to refer only hybrid neuro-fuzzy system. After that, the fuzzy sub-blocks are calculated the neural network learning methods are taken away, executing only the fuzzy system. [50]

record. The idea with GA is to use this power of evolution to solve optimisation problems. The father of the original GA was John Holland, who invented it in the early 1970s. [54]

The definition of "Genetic Algorithm" is:

<<… a method for solving both constrained and unconstrained optimisation problems based on a natural selection process that mimics biological evolution. The algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm randomly selects individuals from the current population and uses them as parents to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution…>.

Indeed, the aim is to achieve an adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics (they represent an intelligent exploitation of a random search used to solve optimisation problems). Although randomised, GAs are by no means random; instead, they employ authentic information to instruct the search into the area of better performance within the search space. Conventional techniques of the GAs aim to replicate natural systems processes necessary for the evolution, especially those that follow the principles first laid down by **Charles Darwin** of "survival of the fittest". Considering the nature, competition among individuals for scarce resources results in the fittest individuals dominating over the weaker ones.

I. TYPES OF NEURO-FUZZY SYSTEMS

Cooperative Neuro-Fuzzy Systems

A cooperative system employs the neural networks only in an initial phase. In this case, the neural network regulates the fuzzy system's sub-blocks using training data; consequently, the neural networks are removed, and only the fuzzy system is executed. "Figure I.1" illustrates an example of a cooperative neuro-fuzzy system.

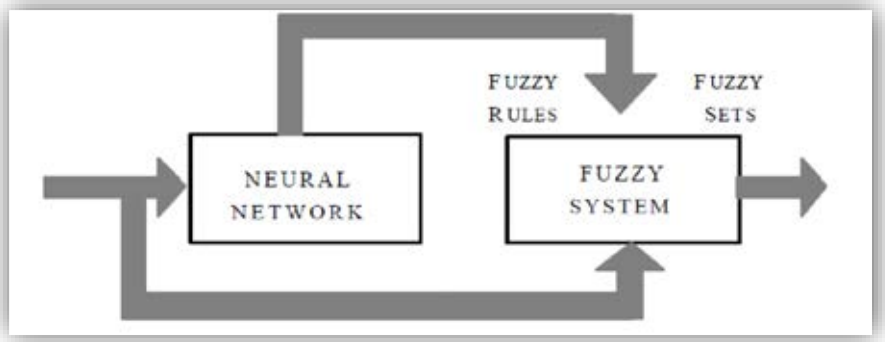


Figure I.1: cooperative system. [52]

Concurrent Neuro-Fuzzy Systems

A concurrent system, by definition, is not a neuro-fuzzy system in strict terms. Because the neural network works in symbiosis with the fuzzy system, it implies that the inputs inflowing the fuzzy system are pre-processed, and then the neural network processes the concurrent system's outputs (or in a reverse way). "Figure I.2" shows a "concurrent neuro-fuzzy system" architecture's example.

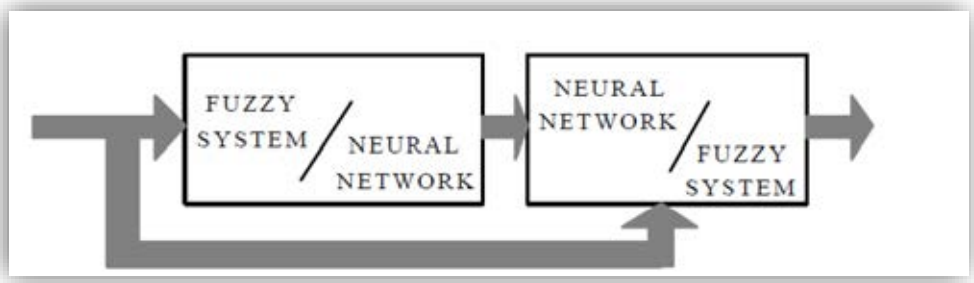


Figure I.2: Concurrent system. [52]

Hybrid Neuro-Fuzzy Systems

In Nauck [57] definition: "A hybrid neuro-fuzzy system is a fuzzy system that uses a learning algorithm based on gradients or inspired by the neural networks theory (heuristical learning strategies) to determine its parameters (fuzzy sets and fuzzy rules) through the pattern's processing (input and output)".

It is possible to identify a neuro-fuzzy system as a set of fuzzy rules. This system can be total created from input-output data or initialised with human knowledge (the same principles of the fuzzy rules). The resultant system by fusing fuzzy systems and neural networks has as advantages of learning through patterns and the straightforward interpretation of its functionality.

There are several distinctive approaches capable of developing hybrid neuro-fuzzy systems; therefore, being a recent research subject, each researcher has defined its particular models. These models are similar, but they present fundamental differences.

Many types of neuro-fuzzy systems are represented by neural networks that implement logical functions. It is not necessary for the application of a learning algorithm in a fuzzy system; however, the representation through a neural network is more convenient because it allows us to visualize the flow of data through the system and the error signals that are used to update its parameters. The additional benefit is to allow the comparison of the different

models and visualize their structural differences. There are several neuro-fuzzy architectures, including:

- Fuzzy Adaptive Learning Control Network
  - (FALCON) C. T. Lin and C. S. Lee [58]
- Adaptive Network-based Fuzzy Inference System
  - (ANFIS) R. R. Jang [59]
- Generalized Approximate Reasoning based Intelligence Control
  - (GARIC) H. Berenji [60]
- Neuronal Fuzzy Controller
  - (NEFCON) D. Nauck & Kruse [61]
- Fuzzy Inference and Neural Network in Fuzzy Inference Software
  - (FINEST) Tano, Oyama and Arnould [62]
- Fuzzy Net
  - (FUN) S. Sulzberger, N. Tschichold and S. Vestli [63]
- Self Constructing Neural Fuzzy Inference Network
  - (SONFIN) Juang and Lin [64]
- Fuzzy Neural Network
  - (NFN) Figueiredo and Gomide [65]
- Dynamic/Evolving Fuzzy Neural Network
  - (EFuNN and dmEFuNN) Kasabov and Song [66]

J. ARTIFICIAL NEURAL SYSTEMS

Artificial neural systems, or neural networks, are (physically) identified as cellular systems which can acquire, store, and utilize experimental knowledge. The knowledge is in the form of stable states or mappings embedded in networks that, in response to the presentation of cues [55], is recallable.

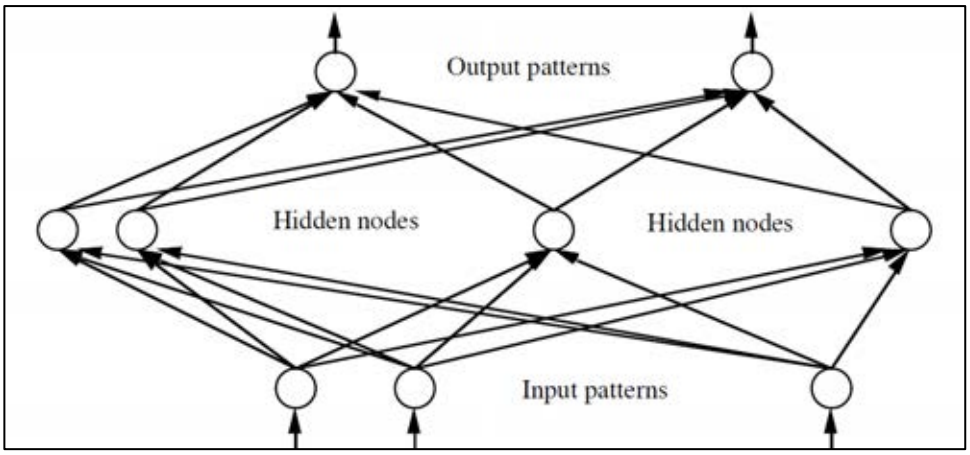


Figure J.1: Multi-layer feed-forward NN.

The primary processing elements of neural networks are called artificial neurons or purely neurons or nodes. Each processing unit has a unique activity level (representing the state of polarization of a neuron), an output value (representing the firing rate of the neuron), a set of input connections (representing synapses on the cell and its dendrite), a bias value (representing an internal resting level of the neuron), and a set of output connections (representing a neuron's axonal projections). Each of these unit's characters is mathematically defined (by real numbers); thus, each connection has a unique intrinsic weight (synaptic strength), which determines the effect of the incoming input on the unit's activation level. The weights may be positive (excitatory) or negative (inhibitory).

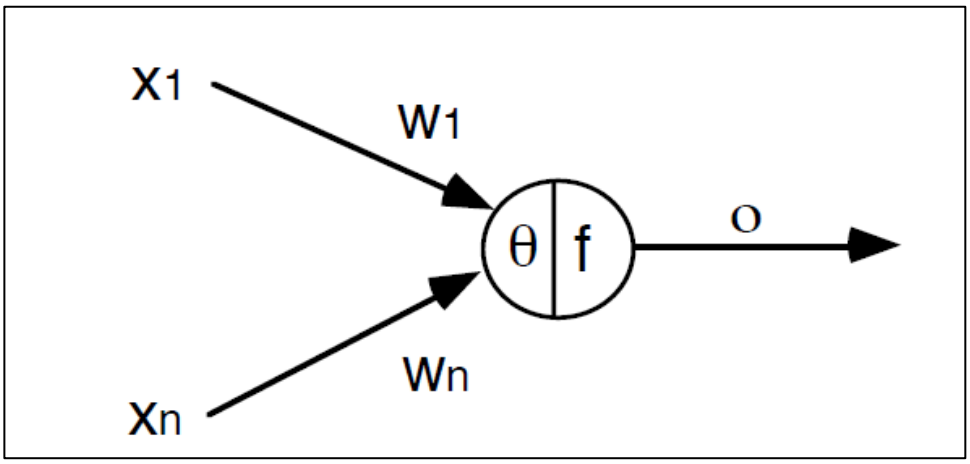


Figure J.2: Processing element with a single output connection. [56]

The signal flow from neuron's inputs, x_j, is considered unidirectional as indicated by arrows, as it is the neuron's output signal flow. The following relationship gives the neuron output signal:

$$o = f(< w, x >) = f(w^T x) = f\left(\sum_{j=1}^{n} w_j x_j\right)$$

(Equation 85)

Where the weight vector is:

$$w = (w_1, w_2, \dots, w_n) \in \mathbb{R}^n$$

(Equation 86)

The function $f(w^T x)$ is often referred to as an activation (or transfer) function. Its domain is the set of activation values, net, of the neuron model, we thus often use this function as $f(net)$. The variable net is a scalar product of the weight and input vectors, according to "Equation 87".

$$net =< w, x >= w^T x = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

(Equation 87)

In the simplest case, the output value o is computed as:

$$o = f(net) = \begin{cases} 1 & if \ \ w^T x \geq \theta \\ 0 & otherwise \end{cases}$$

(Equation 88)

where θ is called threshold-level, and this type of node is called the linear threshold unit. [56]

K. Automotive trend interpretation for EV, PHEV and HEV

At the 2019 Geneva auto show, Gerald Killmann, Toyota's vice president of research and development for Europe, enlightened why the automaker has not embraced EVs yet: battery production capacity. Now, Toyota is not strictly limited in its battery production, although its capacity is significantly lower than Tesla's. It is how Toyota is allocating that production that matters. According to Killmann, Toyota can produce enough batteries for 28,000 electric vehicles each year or 1.5 million HEVs and PHEVs.

For Toyota, selling 1.5 million HEVs and PHEVs reduces carbon emissions by a third more than selling 28,000 EVs. It allows the company to generate a more positive environmental impact by selling many times more HEVs cars than it would be selling far

fewer EVs (consequently far more conventional ICE vehicles) while also providing its customers more practical vehicles[190] at more affordable prices.

The previously enunciated statement does not go more in-depth in mathematical models (for example, details around those carbon emissions calculations). It is difficult to say whether the logic aims to explain away Toyota's irrelevant EV offerings or if it has been Toyota's vision all along to take a pragmatic approach to reduce new-vehicle carbon emissions globally. Nevertheless, results in a significant explanation for understanding the strategy behind distributing Toyota's battery capacity among a more significant number of HEVs (and PHEVs) vehicles than a smaller number of full-electric models. Although Toyota does not target the EVs market now, it does not mean Toyota cannot mass produce EVs. It is merely taking its usual careful, calculated approach to a long game, and hybrids are a vital bridge to that future.

It is essential to highlight that many other automotive OEMs privilege other business models and their interpretations are valid, although they are diverging. The Author's decision to propose Toyota's vision has its roots in the Author's personal beliefs, aligned with Toyota's automotive trends interpretations.

L. ANCILLARY RESULTS FROM THE THESIS WORK

Though not explicitly relating to the proposal's construction, the following annexure outlines a few Author's research that might be worthy of disclosing in the function of potential future researches.

UAV Motor Drive Doctoral Research

The research attempted to eliminate active or heavy passive cooling from the high power density inverter for small UAV BLDC Motor Drive, achieving high efficiency and dissipating generated heat via the inverter PCB and the mechanical structure of the UAV itself. [17]

In an attempt to reduce the inverter's form-factor and weight, the Author attempted to use the first technological generation of "GaN power MOSFETs" on the inverter's "power stage". The use of GaN power MOSFETs requires particular attention because of their well know fragility. A technical overview for this particular WBG technology is briefly covered in [17].

[190] Vehicles capable of avoiding charging anxieties and guarantee high reliability at low fuel consumption.

The research had a focus on the reliability of the whole system and the relative comparison with conventional “Si” based alternatives, investigating the “Power Electronics Packages” and the relative cooling contributions for achieving high reliability and small form-factors; addressing the PCB design principles to follow to minimise problems related to EMI/EMC (a WBG Power MOSFETs typical design challenge).

Particular attention was given to what inherent to the “Gate Driver Circuitry optimisation”. GaN MOSFET’s Gate related issues during the experiments demonstrated to be the principal limitation for reliable and durable operation, especially involving safety-critical applications based on this type of converter technology. Such issues could be associated with the early samples technology used and to the absence, at the time, in the market of a specifically designed “Gate Driver Device for GaN Power MOSFETs”, and in this regard, experiments pursued the adaptation of standard “Si” MOSFET’s “Gate Drivers” in line with recommendations of the GS61008T “GaN Power MOSFET Datasheet/Application Note”.

UAV Motor Drive Doctoral Research Conclusion

The conclusion of [17] shows that:

a) GaN MOSFET GS61008T is definitively an exciting device, as impressive as is the GaN Technology;
b) GaN Technology has been demonstrated within several academic works to be extremely interesting for DC/DC applications since they are capable of work at very high frequencies;
c) Doctoral Researches focused on a precise application (powertrain driver for UAV applications), marked up a critical issue referenced either directly or indirectly, to the MOSFET’s Gates;
d) At that time, investigations address to the GaN Technology that they are not yet mature enough for this specific application as there are no optimised gate driver devices currently available on the market;
e) Optimisation of the gate driver circuitry represents a future research opportunity and will be the focus of future works with GaN.

Concerning the selection of the Motor Drive for a UAV application, it is possible to confirm that, although the GaN technology in future will no doubt be the King of WBG technology and will in time improve and grow its quota within the market, at this moment the most reliable option is still the conventional “Si” Technology.

Mechanical Advantages introduced by the GS61008T package are possible to obtain using:

- for low power/low form-factor applications, the new dual cool packages introduced by Fairchild/ONSEMI (such as FDMT800120DC) and nowadays also by ST (such as STLD125N4F6AG);
- for significant current ratings, the best option is given by IXYS with its SMPD package (such as MMIX1F420N10T or MMIX1F520N075T2);

Few of those conventional components are AEC-Q101 qualified, representing an assurance in terms of consistency and reliability of the product. It is a common opinion that the AEC-101 qualification of a GaN MOSFET will represent a waypoint for GaN Technology and the evidence of technological maturity. A similar conclusion may is valid for “Hybrid Robot’s E-Motor drive applications”.

[17]