Digital Control Systems: Modern Aspects in Modeling and Implementation

DIGITRANS

Co-funded by
the European Union

Co-funded by the
European Union

Digital Control Systems:
Modern Aspects in Modeling and Implementation

Editors: Volodymyr Kazymyr, Nadezhda Kunicina, Anatolijs Zabasta

Anatoliy Prystupa
Maksym Khomenko
Serhii Moroz
Nataliia Yakymchuk
Iryna Trunova
Andrii Rohovenko
Oleksii Krasnozhon
Yosyp Selepyna
Oleksandr Dziubenko
Andrii Hnatov

Riga 2026

The textbook has been developed by the financial support of European Union. The authors from Chernihiv Polytechnic National University, Lutsk National Technical University, Kharkiv National Automobile Highway University and Riga Technical University are responsible for the content of this document. This publication reflects the views only of the authors, and it cannot be regarded as the European Union's official position.

The textbook has been developed in a frame of the project "ERASMUS+ Capacity-building in the Field of Higher Education 2023 Call for Proposals 101127683-DIGITRANS-ERASMUS-EDU-2023-CBHE.

The textbook is devised for students studying in the fields of electrical engineering, computer engineering, and automotive transport. The book comprehensively overviews key concepts, technologies, and applications related to modern digital control systems. It covers theoretical foundations, modeling techniques, practical implementation and recent advances in this field, offering valuable information for students, researchers, and professionals in the computer and electrical engineering.

Key Action: KA2 - Cooperation for innovation and the exchange of good practices

Action: Capacity Building in Higher Education

Action Type: Joint Projects

Deliverable: 2.2. Elaboration of text e-books for students and teachers.

Co-funded by the
Erasmus+ Programme
of the European Union

Project Coordinator: Anatolijs Zabasta

Project Scientific Manager: Nadezhda Kunicina

Editors: Volodymyr Kazymyr, Nadezhda Kunicina, Anatolijs Zabasta

Institution: Riga Technical University

https://doi.org/10.7250/9789934372315
ISBN 978-9934-37-231-5 (pdf)

Contributors

Volodymyr Kazymyr, Dr.Sc, professor, Department of Information and Computer Systems, Chernihiv Polytechnic National University, 95 Shevchenko Str., Chernihiv, 14030, Ukraine, tel.+380 503444377, vvkazymyr@stu.cn.ua

Nadezhda Kunicina, professor, senior researcher of Institute of Industrial Electronics, Electrical Engineering and Energy, Faculty of Computer Science, Information Technology and Energy, Riga Technical University, 12/1 Azenes Str. - 503., Riga, LV 1048, Latvia, tel. +371 26162662, nadezda.kunicina@rtu.lv

Anatolijs Zabasta, senior researcher of Institute of Industrial Electronics, Electrical Engineering and Energy, Faculty of Computer Science, Information Technology and Energy, 12/1 Azenes Str. - 503., Riga, LV 1048, Latvia, tel. +371 29232872, anatolijs.zabasta@rtu.lv

Anatoliy Prystupa, PhD, associate professor, vice-rector for scientific work, Chernihiv Polytechnic National University, 95 Shevchenko Str., Chernihiv, 14030 Ukraine, tel. +380 504652013, anatoliy.prystupa@stu.cn.ua

Andrii Rohovenko, PhD, Head of Information and Computer Systems Department, Chernihiv Polytechnic National University, 95 Shevchenko Str., Chernihiv, 14030 Ukraine, tel. +380 5046522 58, arogovenko@stu.cn.ua

Maksym Khomenko, PhD, associate professor, Department of Electronics, Automation, Robotics and Mechatronics, Chernihiv Polytechnic National University, 95 Shevchenko Str., Chernihiv, 14030, Ukraine, tel.+380 507062770, mr.homax@stu.cn.ua

Oleksii Krasnozhon, PhD, associate professor, Department of Information and Computer Systems, Chernihiv Polytechnic National University, 95 Shevchenko Str., Chernihiv, 14030, Ukraine, tel.+380 666121483, alexey.krasnozhon@stu.cn.ua

Serhii Moroz, PhD in Engineering, Associate Professor of Electronics and Telecommunication Department, Faculty of Computer and Information Technologies, Lutsk National Technical University, Lvivska Str. 75, Lutsk,43018, Ukraine, tel. +380 673897687, s.moroz@lntu.edu.ua

Yosyp Selepyna, PhD in Engineering, Associate Professor of Electronics and Telecommunication Department, Faculty of Computer and Information Technologies, Lutsk National Technical University, Lvivska Str. 75, Lutsk,43018, Ukraine, tel. +380 997201464, y.selepyna@lntu.edu.ua

Nataliia Yakymchuk, PhD in Engineering, Associate Professor of Electronics and Telecommunication Department, Faculty of Computer and Information Technologies, Lutsk National Technical University, Lvivska Str. 75, Lutsk,43018, Ukraine, tel. +380 995462041, n.yakymchuk@lntu.edu.ua

Oleksandr Dziubenko, associate professor of Kharkiv National Automobile and Highway University, 25 Yaroslava Mudrogo Str., Kharkiv, 61002, Ukraine, tel. +380 667684116, dzyubenko.alan@gmail.com

Iryna Trunova, associate professor of Kharkiv National Automobile and Highway University, 25 Yaroslava Mudrogo Str., Kharkiv, 61002, Ukraine, tel. +380 677240653, trunova.irinaserg@gmail.com

Andrii Hnatov, professor, Dr. of Science, Head of Vehicle Electronics Department, Kharkiv National Automobile and Highway University, 25, Yaroslav Mudry Str.street, Kharkiv, Ukraine, 61002, tel. +380 667430887, kalifus76@gmail.com

Foreword

The rapid evolution of technology in the 21st century has transformed the way we approach automation, energy efficiency, and intelligent systems. At the heart of these transformations lie digital control systems (DCSs) — powerful tools that integrate computation, communication, and control into the very fabric of modern engineering. Their relevance extends across robotics, automotive systems, renewable energy, and aerospace, shaping not only industrial progress but also the emerging vision of sustainable development.

This textbook, Digital Control Systems: Modern Aspects in Modeling and Implementation, was created within the framework of the ERASMUS+ DIGITRANS project and reflects a collaborative effort by researchers and educators from universities in Ukraine and Latvia. It presents both theoretical foundations and practical implementations of DCSs, offering readers insight into methods ranging from classical modeling to intelligent approaches based on artificial neural networks and fuzzy logic. The work also addresses the growing challenges posed by Industry 4.0 and Industry 5.0, emphasizing the integration of artificial intelligence and the orientation toward human-centered, sustainable solutions.

The book is structured to provide a comprehensive view:

- from control of DC and BLDC motors,
- through in-vehicle communication systems,
- to photovoltaic converters, unmanned aerial vehicle swarms, and FPGA-based solutions.

The book carefully blends mathematical modeling, simulation, and experimental validation, ensuring that students and researchers not only grasp the principles but also see how these systems are applied in real-world contexts. Each chapter contributes to a comprehensive understanding of the field, combining classical methods with advanced approaches that employ artificial intelligence, neural networks, and fuzzy logic.

This work will serve as a valuable resource for students of electrical and computer engineering, as well as postgraduate researchers, academic staff, and practicing engineers seeking to deepen their knowledge in the field of modern digital control systems. It is our conviction that the presented material will not only facilitate the acquisition of advanced knowledge but will also stimulate innovation, interdisciplinary cooperation, and sustainable solutions in the field of digital control systems.

Chernihiv, Ukraine, September 2021
Anatoliy Prystupa,
PhD, associate professor,
vice-rector for scientific work,
Chernihiv Polytechnic National University

Contents

Introduction

Control systems for technical means and devices have come a long way in their development, from primitive mechanical switches to electrical analog and digital cybernetic systems. In fact, the development of control systems coincides with the change in technological structures, which at the beginning of the 21st century entered the era of cyber physical systems (CPSs). These systems marked the emergence of a new trend in the formation of production systems, which was called Industry 4.0.

CPSs, integrating computational and physical processes, include more physical components than purely embedded systems, and are traditionally denoted by the symbol C3, under which computation, communication and control are combined. But being an integral part of CPSs, control systems under the influence of new technologies themselves become CPS, combining the three aforementioned components (computing, communication and control) within the framework of the general trend of digitalization.

Given that since 2021, in addition to Industry 4.0, a new Industry 5.0 program has appeared, aimed at systemic transformations towards a sustainable development. Under the influence of Industry 5.0, the emphasis in the use of digital control systems (DCSs) is also changing somewhat. They are no longer focused exclusively on production development processes, as is the case in Industry 4.0, but also cover the issues of applying CPSs in applied areas, ensuring their sustainable development taking into account human needs. A characteristic feature of DCSs in Industry 5.0 is also the use of artificial intelligence methods, especially those based on artificial neural networks and fuzzy decision-making rules.

This e-book is devoted to highlighting issues related to modern approaches to creating digital control systems, and, above all, to their modeling and implementation in the era of CPSs. DCS's architectures, models and examples of implementing are considered in the parts of computing, communication and control of objects from different applied fields.

The first five chapters of the book are related to the construction of digital control systems based on the use of microprocessors (MCU).

Chapter 1 considers general issues of DC motor modeling and their optimization. Separately, the ANN-based controller scheme for the positioning system is analyzed, and a description of the process of its simulation and testing is provided. This chapter can certainly be useful for developers of robotic systems where precise adherence to position coordinates is required, for example, in the fields of electron beam welding and other high-precision instrument making.

The 2nd Chapter highlights the issue of building a control system for a brushless DC (BLDC) motor, which is one of the best types of electric drives today. Due to their high output power and efficiency, these motors provide stable operation even in the most difficult production conditions. It combines simplicity of design, high reliability, long service life and maximum efficiency. The emergence of such motors, which are controlled by a controller, is due precisely to the development of electronics. Focused on small-sized objects, they are suitable for use in industry, machine tools, electric vehicles, and are well suited for installation on electric scooters, high-speed electric bicycles, electric tricycles, and other electric vehicles. The section models various BLDC motor control methods, including the Field-Oriented Control (FOC) method, and examples of their implementation.

Chapter 3 is directly related to control systems in the automotive sector, namely the issue of In-Vehicle Network. it is directly aimed at the communication component of the DCS in the triad of its implementation of DCS as a CPS. Various topologies for building in-vehicle communication systems are considered, including CAN Technology, LIN bus and MOST system with all structural components. Modern approaches to building In-Vehicle Network based on Bluetooth and FlexRay technologies, their architecture and features of operating are shown in detail.

The 4th Chapter specifically meets the challenges of Industry 5.0. It is aimed at the ecology sphere and is filled with an example of building an intelligent control system using artificial neural networks (ANN) and fuzzy control methods. Maximum Power Point Tracking (MPPT) in

photovoltaic converters (PVC) is used as the main control principle. Based on analytical models, the architecture of the Fuzzy control system for MPPT is formed, and the properties of the constructed DCS are investigated with mathematical methods and simulation. All stages of the modeling process are considered in detail, including modeling of linguistic variables, forming of rule base, modeling of the PVC control surface, tuning of linguistic variables using ANN, creating a model of the MPPT control system in MATLAB, simulation experiments with the model. To verify the developed model, the simulation results are compared with the results of a full-scale experiment.

Current issues of building control systems for modern UAVs are investigated in Chapter 5. The UAV as part of a drone swarm, which provides defense of a critical infrastructure facility, is considered as a control object. An overview of the hardware architecture of the UAV control system using an additional microcomputer is provided. The components of the software, which perform computational tasks in the control system, are described in detail: processing of target audio signals, video detection and classification of targets, support for inter-drone communication within the swarm and interaction with the flight controller. The chapter includes description of applied model-oriented control method to provide the swarm mission that based on embedded models of control algorithms in E-networks forms. Implementation of the applied control methods is given with test results.

The last 6^{th} Chapter is devoted to the implementation of control systems based on FPGA. Programmable Logic Devices (PLD) represent a fundamentally different approach to building DCS then using of MCU, because it consists in hardware implementation of the main algorithms by implementing them as microcircuits in chip. The chapter examines in detail the features of using FPGA as a control element in comparison with MCU and the analysis of different types of Programmable Logic Devices. The main capabilities of Hardware description languages, such as VHDL and Verilog, as well as the Intel Quartus platform as a tool for designing FPGA are considered. The process of creating FPGA firmware and an example of implementation the Global Maximum Power Point Tracking (GMPPT) algorithm for PVC are considered in detail.

In total, this e-book highlights the main theoretical and applied aspects related to the modeling and implementation of control systems from the perspective of their application in the CPSs. The considered examples of DCSs will contribute to a better understanding of the current trends and prospects for digitalization in the field of control, responding to the challenges of Industry 5.0.

Chapter 1. DC motor digital control Systems

1.1. MCU based DC motor control systems

The electric drive is an integral component of virtually any industrial equipment. In general, all electric drives could be divided into two groups by their application. The first group includes issues associated with obtaining the rotational movement of the operating device with the given parameters such as speed, speed accuracy, speed stability. The second group deals with issues related to the implementation of the rotation or movement of the operating device at a given angle or distance - the so-called positioning.

Positioning electric drives, with DC motor, are widely used in series of machines and mechanisms, such as industrial robots, manipulators, rotary and coordinate tables, packaging equipment, hoisting machinery. There is virtually no industry in which the positioning electric drive has not been used - from microelectronics to the food industry.

Nowadays there are a lot of control techniques and strategies for DC motor position system design. The most wellknown control methods include proportional–integral–derivative (PID) algorithm [11, 12, 13]. This control strategy is easy to implement but it has one serious drawback associated with proper coefficients tuning. So, the optimal tuning algorithm of PID controller is the major issue in this case.

Another approach to DC motor position system analysis and synthesis is based on state space methods. These methods are suitable for multi-input and multi-output systems and could be used to obtain optimal in some sense controller [14], [15]. Generally, the drawback of these methods is a sensitivity to motor parameters changes.

Much research in recent years has been devoted to algorithms based on artificial neural networks (ANNs). The ANN is very powerful tool that could be used in different regions of application such as pattern recognition, function approximation, classification, control and so on. Major advantages of the ANN are its approximation ability, and tuning flexibility. Therefore, this approach is widely used for DC motor control system implementation [16], [17], [18]. The best results could be achieved using ANN in combine with state space methods when the positioning system is designed.

This chapter describes an application of ANN approach in combination with state space method of variable gain for DC motor positioning. Such combination improves robustness with respect to supply voltage disturbances and DC motor parameter variations. The applicability of proposed method is illustrated by simulation results and experimental data.

1.2. Modeling of DC motor DCS with fixed time step

The state space method is widely used for the analysis and synthesis of automatic control systems in modern control theory. The advantages of this method over the traditional frequency approach are unification of onedimensional and multidimensional system description with different types of quantisation, as well as the convenience of representation for the compute solution [1].

Any linear time-invariant system represented in terms of the state space can be described by the following equations written in matrix form [1, 2]

$$\begin{aligned} \dot{x}(t) &= \mathrm{A}x(t) + \mathrm{D}m(t), \\ y(t) &= \mathrm{B}x(t) + \mathrm{G}m(t), \end{aligned} \tag{1.1}$$

where

x – is the state vector;
A – is the state matrix;
D – is the input matrix;
m – is the input vector;

y – is the output vector;
B – is the output matrix;
G – is the feedthrough matrix.

So, if rotation angle is taken as an output variable, the DC motor transfer function has the form:

$$\Omega(p) = \frac{k}{p(p+a)(p+b)}; \tag{1.2}$$

where

p - Laplace transform;

$a = \frac{1+\sqrt{1-4\frac{T_e}{T_m}}}{2T_e}; b = \frac{1-\sqrt{1-4\frac{T_e}{T_m}}}{2T_e};$

$T_e = \frac{L}{R}$ – the electrical time constant of DC motor;

$T_m = \frac{\text{JR}}{C_e k_T}$ – the electromechanical time constant of DC motor.

C_e – the back emf constant;

k_T – the torque constant ($k_T \approx C_e$).

Therefore, the DC motor in state space can be described by equations

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -a & 1 \\ 0 & 0 & -b \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ k \end{bmatrix} m,$$
$$y = [1 \quad 0 \quad 0] \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + [0]m. \tag{1.3}$$

Consequently, $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -a & 1 \\ 0 & 0 & -b \end{bmatrix}$, $D = \begin{bmatrix} 0 \\ 0 \\ k \end{bmatrix}$, $B = [1 \quad 0 \quad 0]$, $G = [0]$ and the modeling scheme of DC motor is shown in Fig. 1.1.

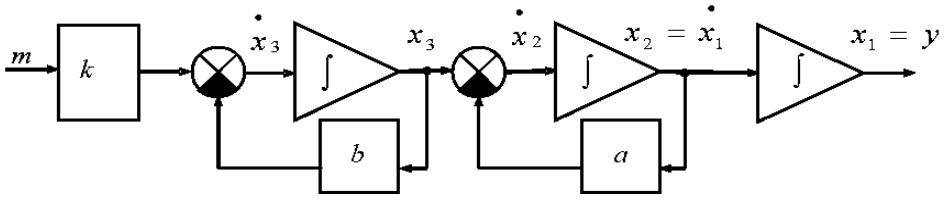


Fig. 1.1. DC motor modeling scheme in terms of state space method.

Now, if the input variables considered in conjunction with the state variables of the system, the column state vector and the state matrix of the increased dimension can be written, taking the assumption that *m* is a step function [2], we will obtain

$$V = \begin{bmatrix} m \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -a & 1 \\ k & 0 & 0 & -b \end{bmatrix}. \tag{1.4}$$

Let the initial conditions are given by the vector V (0 +), then the state of the system at any time t can be determined by the equation

$$V(t) = \Phi(t)V(0+), \tag{1.5}$$

where $\Phi(t)$ is called transition matrix and can be derived from one of the two following equations [3]

$$\Phi(t) = e^{At}, \tag{1.6}$$

$$\Phi(t) = L^{-1}\{[pI - A]^{-1}\}, \tag{1.7}$$

where L^{-1} – operator of the inverse Laplace transform, I – the identity matrix.

This approach to the analysis of control systems can also be applied to discrete systems by moving from continuous to discrete time with period *T*, then the equation of transfer takes the form [2]

$$V((n+1)T) = \Phi(T)\mathrm{BV(nT)}, \tag{1.8}$$

where the matrix **B** is made on the basis of the transition states of the system.

Consider the structure of digital control system (DCS) with sampling period *T* and the feedback shown in Fig.1.2.

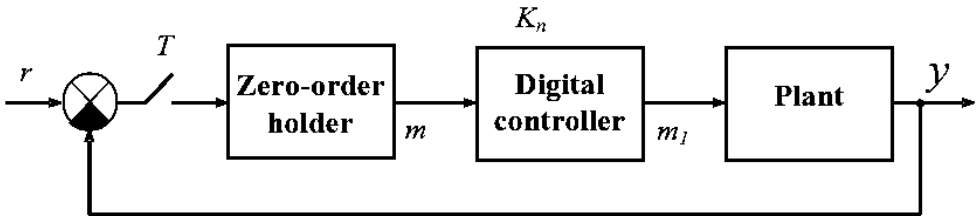


Fig. 1.2. Digital control system structure.

Such DCS is fed an input step function *r* with assume that the plant is linear and its parameters are known. In this case, the controller synthesis issue reduces to finding the transfer function capable to obtain the transition process that satisfies the specified quality criteria.

A method of variable gain for the control system is described in [2]. The essence of this method is to consider a digital controller as an amplifier with a gain *Kn*, which can take different values at each sampling period. The input signal for this amplifier is *m*, and the output *m1*. These signals are linearly related by a factor *Kn*, hence the transition matrixes of the system for different switching periods are functions of the permanent gains *Kn*. Using one of the equations (1.6) or (1.7) to find the transition matrixes of the system, the transition equation (1.8) and the vector of initial conditions, the system of equations can be obtained and the values of the gain for each switching period can be found. Transfer function of the digital controller can be derived from the expression

$$D(z) = \frac{M1(z)}{M(z)} = \frac{\sum_{j=0}^{n} K_j m(\mathrm{jT+})z^{-j}}{\sum_{j=0}^{n} m(\mathrm{jT+})z^{-j}}, \tag{1.9}$$

where *n* – the order of the plant differential equation.

The main advantage of this method is customizable optimum response without overshoot. However, it should be noted that this system does not provide an optimal time of the transient process for whole range of input signals. This is due to the fact that transient duration is determined by the supply voltage and the maximum positioning angle. Therefore, positioning time duration weakly depends on the given angle, and remains virtually the same for both large and small angles. Consequently, the transient process is close to optimal only for the large rotation angles.

1.3. Optimal control algorithm with variable time step

The minimum time for the transient process for different given angles can be obtained if the digital controller feeds the DC motor with maximum in absolute voltage on each control step, but the lengths of each step is different and depends from the given rotation angle.

The block diagram of calculation in the control system shown in Fig. 1.3. The calculations of control steps duration for the scheme can be derived from the differential equations and the equations of transition states (excluding the controller)

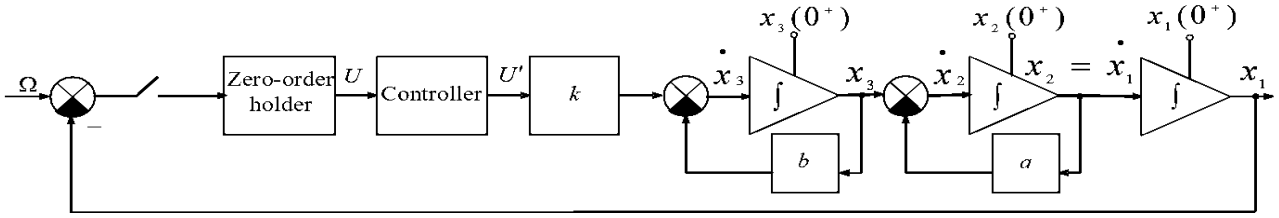


Fig. 1.3. The block diagram of control system.

$$\dot{\Omega} = 0;\ \dot{x_1} = x_2;\ \dot{x_2} = x_3 - \mathrm{a}x_2;\ \dot{x_3} = \mathrm{k}U - \mathrm{b}x_3;\ \dot{U} = 0; \tag{1.10}$$

$$\Omega\left(\mathrm{nh}^+\right) = \Omega(\mathrm{nh});\ x_1\left(\mathrm{nh}^+\right) = x_1(\mathrm{nh});x_2\left(\mathrm{nh}^+\right) = x_2(\mathrm{nh});\ x_3\left(\mathrm{nh}^+\right) = x_3(\mathrm{nh});\ U\left(\mathrm{nh}^+\right) = \Omega(\mathrm{nh}) - x_1(\mathrm{nh}). \tag{1.11}$$

The state matrix **A** can be derived from the differential equations (1.10):

$$\mathbf{A}=\begin{bmatrix}0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0\\ 0 & 0 & -a & 1 & 0\\ 0 & 0 & 0 & -b & k\\ 0 & 0 & 0 & 0 & 0\end{bmatrix}. \tag{1.12}$$

The **B** matrix can be derived from the equations of transition states (1.11):

$$\mathbf{B}=\begin{bmatrix}1 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\\ 1 & -1 & 0 & 0 & 0\end{bmatrix}. \tag{1.13}$$

The extended state column vector of the system is

$$\mathbf{V}=\begin{bmatrix}\Omega\\ x_1\\ x_2\\ x_3\\ U\end{bmatrix}. \tag{1.14}$$

The vector of initial conditions is

$$\mathbf{V}(0)=\begin{bmatrix}\Omega(0)\\ 0\\ 0\\ 0\\ 0\end{bmatrix} = \Omega(0)\begin{bmatrix}1\\ 0\\ 0\\ 0\\ 0\end{bmatrix}. \tag{1.15}$$

The transition matrix at the first control step is defined from the state matrix of the system:

$$\mathbf{\Phi}(h_1)=\begin{bmatrix}1 & 0 & 0 & 0 & 0\\ 0 & 1 & M_1 & P_1 & \mathrm{kL}_1\\ 0 & 0 & A_1 & W_1 & \mathrm{kP}_1\\ 0 & 0 & 0 & B_1 & \mathrm{kQ}_1\\ 0 & 0 & 0 & 0 & 1\end{bmatrix}, \tag{1.16}$$

where $A_1=e^{-\mathrm{ah}_1}$; $B_1=e^{-\mathrm{bh}_1}$; $L_1=\frac{\mathrm{abh}_1-a-b+\frac{b^2e^{-\mathrm{ah}_1}-a^2e^{-\mathrm{bh}_1}}{b-a}}{b^2a^2}$; $M_1=\frac{1-e^{-\mathrm{ah}_1}}{a}$; $P_1=\frac{1+\frac{\mathrm{be}^{-\mathrm{ah}_1}-\mathrm{ae}^{-\mathrm{bh}_1}}{a-b}}{\mathrm{ab}}$; $Q_1=\frac{1-e^{-\mathrm{bh}_1}}{b}$; $W_1=\frac{e^{-\mathrm{ah}_1}-e^{-\mathrm{bh}_1}}{b-a}$.

The modified equation of the transition state (1.8) can be used to determine the state of the system at different times. The variable lengths of control step should be considered [2]

$$\mathbf{V}(h_n^+)=\mathbf{B}\mathbf{V}(h_n); \tag{1.17}$$

$$\mathbf{V}(h_n+h_{n+1})=\mathbf{\Phi}(h_n)\,\mathbf{V}(h_n^+) \tag{1.18}$$

The extended state colon vector on the first control step can be derived from (1.17), (1.18):

$$\mathbf{V}(0^+)=\begin{bmatrix}1\\0\\0\\0\\m_1\end{bmatrix}; \tag{1.19}$$

$$\mathbf{V}(h_1)=\begin{bmatrix}1\\ \mathrm{kL}_1m_1\\ \mathrm{kP}_1m_1\\ \mathrm{kQ}_1m_1\\ m_1\end{bmatrix}. \tag{1.20}$$

The following transition matrix is for the second control step:

$$\mathbf{\Phi}(h_2)=\begin{bmatrix}1 & 0 & 0 & 0 & 0\\ 0 & 1 & M_2 & P_2 & \mathrm{kL}_2\\ 0 & 0 & A_2 & W_2 & \mathrm{kP}_2\\ 0 & 0 & 0 & B_2 & \mathrm{kQ}_2\\ 0 & 0 & 0 & 0 & 1\end{bmatrix}, \tag{1.21}$$

where

$A_2=e^{-\mathrm{ah}_2}$; $B_2=e^{-\mathrm{bh}_2}$; $L_2=\frac{\mathrm{abh}_2-a-b+\frac{b^2e^{-\mathrm{ah}_2}-a^2e^{-\mathrm{bh}_2}}{b-a}}{b^2a^2}$; $M_2=\frac{1-e^{-\mathrm{ah}_2}}{a}$; $P_2=\frac{1+\frac{\mathrm{be}^{-\mathrm{ah}_2}-\mathrm{ae}^{-\mathrm{bh}_2}}{a-b}}{\mathrm{ab}}$; $Q_2=\frac{1-e^{-\mathrm{bh}_2}}{b}$; $W_2=\frac{e^{-\mathrm{ah}_2}-e^{-\mathrm{bh}_2}}{b-a}$.

Taking in to account (1.20), (1.21) from equations (1.17), (1.18) the extended state colon vector on the second control step can be obtained:

$$\mathbf{V}(h_1^+)=\begin{bmatrix}1\\ \mathrm{kL}_1m_1\\ \mathrm{kP}_1m_1\\ \mathrm{kQ}_1m_1\\ m_2\end{bmatrix}; \tag{1.22}$$

$$\mathbf{V}(h_1+h_2)=\begin{bmatrix}1\\ \mathrm{kL}_1m_1+\mathrm{kM}_2P_1m_1+\mathrm{kP}_2Q_1m_1+\mathrm{kL}_2m_2\\ \mathrm{kA}_2P_1m_1+\mathrm{kW}_2Q_1m_1+\mathrm{kP}_2m_2\\ \mathrm{kB}_2Q_1m_1+\mathrm{kQ}_2m_2\\ m_2\end{bmatrix}. \quad (1.23)$$

The transition matrix for the third control step can be defined:

$$\mathbf{\Phi}(h_3)=\begin{bmatrix}1&0&0&0&0\\0&1&M_3&P_3&\mathrm{kL}_3\\0&0&A_3&W_3&\mathrm{kP}_3\\0&0&0&B_3&\mathrm{kQ}_3\\0&0&0&0&1\end{bmatrix}, \quad (1.24)$$

where

$A_3=e^{-\mathrm{ah}_3}$; $B_3=e^{-\mathrm{bh}_3}$; $L_3=\frac{\mathrm{abh}_3-a-b+\frac{b^2e^{-\mathrm{ah}_3}-a^2e^{-\mathrm{bh}_3}}{b-a}}{b^2a^2}$; $M_3=\frac{1-e^{-\mathrm{ah}_3}}{a}$; $P_3=\frac{1+\frac{\mathrm{be}^{-\mathrm{ah}_3}-\mathrm{ae}^{-\mathrm{bh}_3}}{a-b}}{\mathrm{ab}}$; $Q_3=\frac{1-e^{-\mathrm{bh}_3}}{b}$; $W_3=\frac{e^{-\mathrm{ah}_3}-e^{-\mathrm{bh}_3}}{b-a}$.

Taking in to account (1.23), (1.24) from equations (1.17), (1.18) the extended state colon vector on the third control step can be obtained:

$$\mathbf{V}((h_1+h_2)^+)=\begin{bmatrix}1\\ \mathrm{kL}_1m_1+\mathrm{kM}_2P_1m_1+\mathrm{kP}_2Q_1m_1+\mathrm{kL}_2m_2\\ \mathrm{kA}_2P_1m_1+\mathrm{kW}_2Q_1m_1+\mathrm{kP}_2m_2\\ \mathrm{kB}_2Q_1m_1+\mathrm{kQ}_2m_2\\ m_3\end{bmatrix}; \quad (1.25)$$

$$\mathbf{V}(h_1+h_2+h_3)=$$
$$\begin{bmatrix}1\\ \mathrm{kL}_1m_1+\mathrm{kM}_2P_1m_1+\mathrm{kP}_2Q_1m_1+\mathrm{kL}_2m_2+\mathrm{kM}_3A_2P_1m_1+\mathrm{kM}_3W_2Q_1m_1+\\ +\mathrm{kM}_3P_2m_2+\mathrm{kP}_3B_2Q_1m_1+\mathrm{kP}_3Q_2m_2+\mathrm{kL}_3m_3\\ \mathrm{kA}_3A_2P_1m_1+\mathrm{kA}_3W_2Q_1m_1+\mathrm{kA}_3P_2m_2+\mathrm{kW}_3B_2Q_1m_1+\mathrm{kW}_3Q_2m_2+\mathrm{kP}_3m_3\\ \mathrm{kB}_3B_2Q_1m_1+\mathrm{kB}_3Q_2m_2+\mathrm{kQ}_3m_3\\ m_3\end{bmatrix}. \quad (1.26)$$

Since the transfer function of the DC motor with rotation angle as an output has the third order, the finite duration process can be realized within three control steps. The conditions below must be met to achieve the finite duration transient process in the system:

$$\begin{cases}x_1(h_1+h_2+h_3)=\Omega\\ x_2(h_1+h_2+h_3)=0.\\ x_3(h_1+h_2+h_3)=0\end{cases} \quad (1.27)$$

A final system of equations is obtained from (26) and (27):

$$\begin{cases}k(L_1m_1+M_2P_1m_1+P_2Q_1m_1+L_2m_2+M_3A_2P_1m_1+M_3W_2Q_1m_1+\\ +M_3P_2m_2+P_3B_2Q_1m_1+P_3Q_2m_2+L_3m_3)=\Omega\\ k(A_3A_2P_1m_1+A_3W_2Q_1m_1+A_3P_2m_2+W_3B_2Q_1m_1+W_3Q_2m_2+P_3m_3)=0\\ k(B_3B_2Q_1m_1+B_3Q_2m_2+Q_3m_3)=0\end{cases}. \quad (1.28)$$

Since the amplitude of the control actions $|m_1|=|m_2|=|m_3|=m$ is known and defined by the limits of voltage for a particular DC motor, the system of equations (1.28) contains three unknown variables

– h_1, h_2, h_3. Therefore, the solution of this system for different given angles Ω, determines the duration of all control steps – h_1, h_2, h_3 and gives the way to implementation of digital controller with time-pulse modulation. However, this system has no analytical solutions and numerical methods should be used to obtain the duration of control steps. This becomes a major issue for implementation of algorithm as a part of embedded system. Artificial neural network (ANN) is one possible solution of this issue.

1.4. ANN based DCS

An artificial neuron is a first approximation that simulates the biological neuron. Model of artificial neuron (see Fig. 1.4) consists of such blocks as weighting, bias, adder, and transfer (activation) function of the neuron [4], [5].

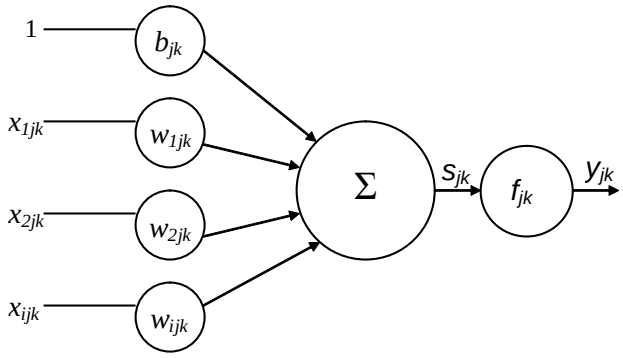


Fig. 1.4. Model of artificial neuron.

The j-th neuron of k-th layer is shown on Fig. 1.4, where:

x_{1jk} ... x_{ijk} – the inputs;
w_{1jk} ... w_{ijk} – weights;
b_{jk} – bias;
f_{jk} – activation function;
y_{jk} – the output;
s_{jk} – a weighted sum.

In mathematical terms the neuron can be described as follows

$$s_{\mathrm{jk}} = \sum_{i=1}^{N} w_{\mathrm{ijk}} x_{\mathrm{ijk}} + b_{\mathrm{jk}}, \tag{1.29}$$

$$y_{\mathrm{jk}} = f_{\mathrm{jk}}(s_{\mathrm{jk}}), \tag{1.30}$$

or in matrix form

$$s_{\mathrm{jk}} = W_{\mathrm{jk}} X_{\mathrm{jk}} + b_{\mathrm{jk}}. \tag{1.31}$$

An activation function f_j can have different shapes depending on the problem is solved by the neuron. Most widespread activation functions are shown in Fig. 1.5 and described by the following expressions.

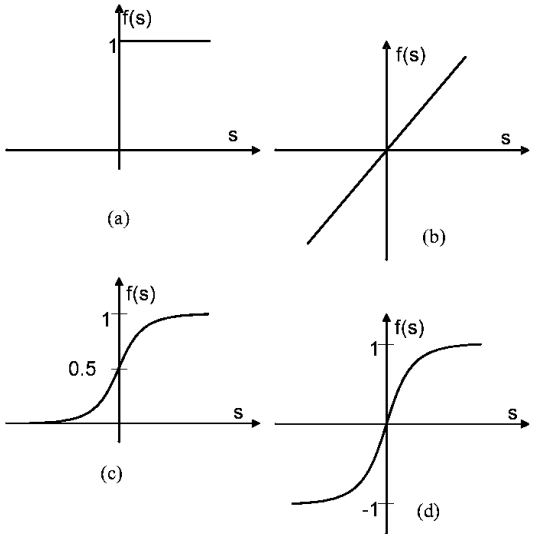


Fig. 1.5 Activation functions:
(a) threshold (b) linear (c) logistic (d) hyperbolic tangent.

Below the formal definitions of activation functions are presented:

- threshold function:

$$f(s) = \begin{cases} 0, s < 0 \\ 1, s \geq 0 \end{cases}. \tag{1.32}$$

- linear function

$$f(s) = s. \tag{1.33}$$

- logistic function

$$f(s) = \frac{1}{1+e^{-s}}. \tag{1.34}$$

- hyperbolic tangent function

$$f(s) = \frac{e^{s}-e^{-s}}{e^{s}+e^{-s}}. \tag{1.35}$$

The logistic (1.34) and the hyperbolic tangent (1.35) functions are widely used because of their properties to amplify weak signals and reduce gain for strong input signals. Thus, the same neuron equally well handles both weak and strong input signals. The advantage of hyperbolic tangent function is probability to take both positive and negative values.

Artificial neural network (ANN) is a set of neurons connected to each other. Typically, activation functions of all neurons in the network are fixed, but the weights are the parameters of the network and can be modified. The topology of the artificial neural network can be divided into three main groups: full-mesh, weakly connected networks, multilayer network. Nowadays it is widely recognised, that multilayer network is the best topology for automatic control systems [5], [6], [7], [8].

Multilayer network, that is shown in Fig. 1.6, consists of three layer types. The input layer does not perform any calculations. The task of this layer is to transfer the input signal to all neurons of the next layer. The hidden layer or layers are the basis of a neural network. All neurons in the hidden layer typically have the same activation function. The output layer provides output signals of the network.

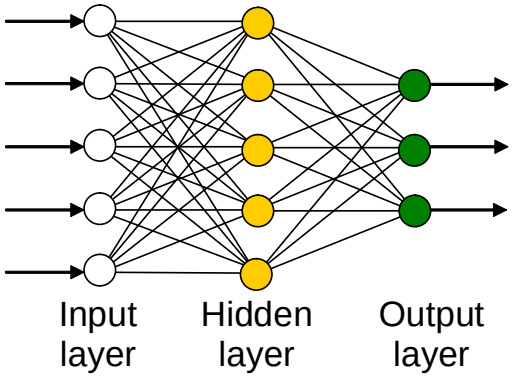


Fig. 1.6. Multilayer ANN.

The key feature of multilayer ANN training is backpropagation algorithm. This is an iterative gradient algorithm that is used to minimize the standard deviation of current output from the desired output in multilayer neural networks. The objective function to be minimized has the form [5]

$$E = \frac{1}{2}\sum_j \sum_m \left(d_j^m - y_j^m\right)^2, \tag{1.36}$$

where

m – number of training vectors;
j – neuron number in the output layer;
y_j^m – neuron output;
d_j^m – the desired output.

Nowadays different varieties of backpropagation algorithm are known, many of them are implemented in MATLAB. The most efficient algorithms based on conjugate gradient method are Gauss-Newton algorithm (GNA) and Levenberg-Marquard (LMA) algorithm.

Three basic issues should be taken in to account when using the multilayer neural network for control prepuces:

- the First of all the feedforward multilayer neural network is static network. The output signal of such network depends on current input signal and does not depend on previous state of the system. Therefore. the additional variables that perform information about system dynamics should be used. It could be a delayed output signal or in case of DC motor positioning system it could be rotary speed and acceleration.
- the second basic issue is scaling. A proper scaling should be used for all network inputs to ensure that all inputs have the same influence on the network output.
- the third major issue is data set for learning algorithm. The data set should be regularly distributed upon whole range of possible input values to ensue the best approximation capability of the network.

The scheme shown in Fig. 1.7 can be proposed for data collection and training of ANN in DCS context. The input signals are various given rotating angles ($\Omega_{in}(k)$). Master controller implements optimal control algorithm with variable time step and generates control action ($V_{cont}(k)$) according to error signal ($e(k)$). Control action feeds the input of DC motor model. The output signals of DC motor model are used to provide feedback ($\Omega_m(k)$ – rotation angle) and additional variable for ANN ($\omega_m(k)$ – rotation speed). ANN according to its input signals produces an output signal which is compared with control action. The result of comparison is an error ($e_{NN}(k)$), which can be used by backpropagation algorithm to adjust the weights of ANN. The task angles that should be applied to the input are shown in Fig. 1.8. Such kind of signal aims to form regularly distributed data set for positive and negative rotation angles.

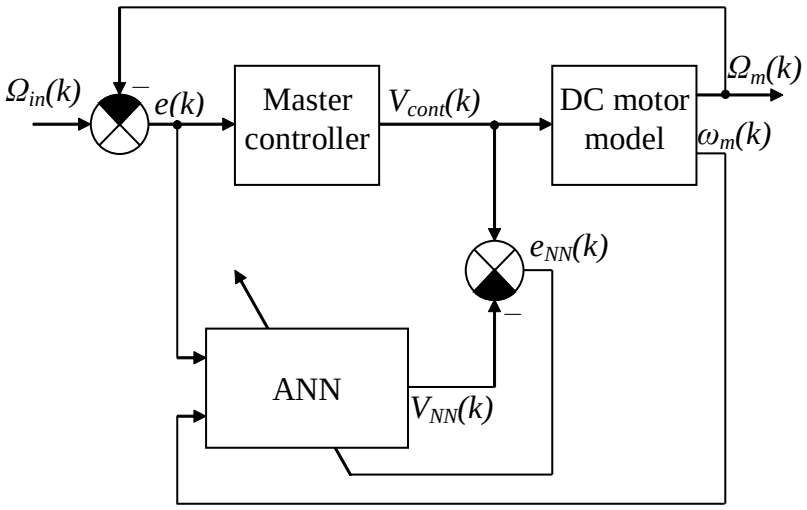


Fig. 1.7. ANN data collection and training scheme.

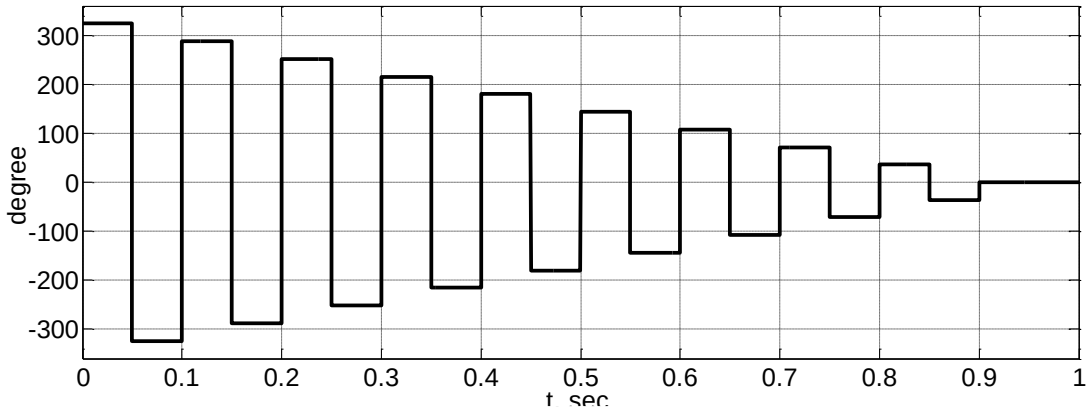


Fig. 1.8. Task angles for data collection.

As it has been shown in [9] and [10] almost any function can be approximated by the multilayer neural network with at least one hidden layer and one output layer. Also, neurons should have non linear activation function in hidden layer and linear activation function in output layer. However no special rule is given for neuron number and number of hidden layers selection. Next recommendations can be given to select the structure of ANN. The sufficient quantity of neurons should be selected to provide accuracy of approximation and this quantity should be small enough to be implemented in embedded system.

Modeling different structures shows that the best ratio of neurons quantity to approximation error gives ANN controller structure presented in Fig. 1.9. This ANN has two neurons in input layer, two hidden layers with three neurons in each hidden layer and one neuron in output layer. All neurons in hidden layers have a hyperbolic tangent activation function. The output neuron has a linear activation function. The hyperbolic tangent activation function has been chosen for neurons in hidden layers because it is well suitable for bipolar operation range of positioning system.

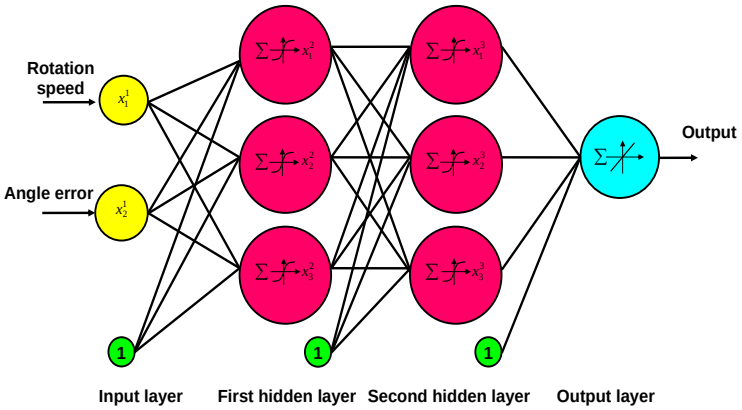


Fig. 1.9. ANN controller structure.

Fig. 1.10 shows the MATLAB Simulink model of ANN based DC motor positioning system.

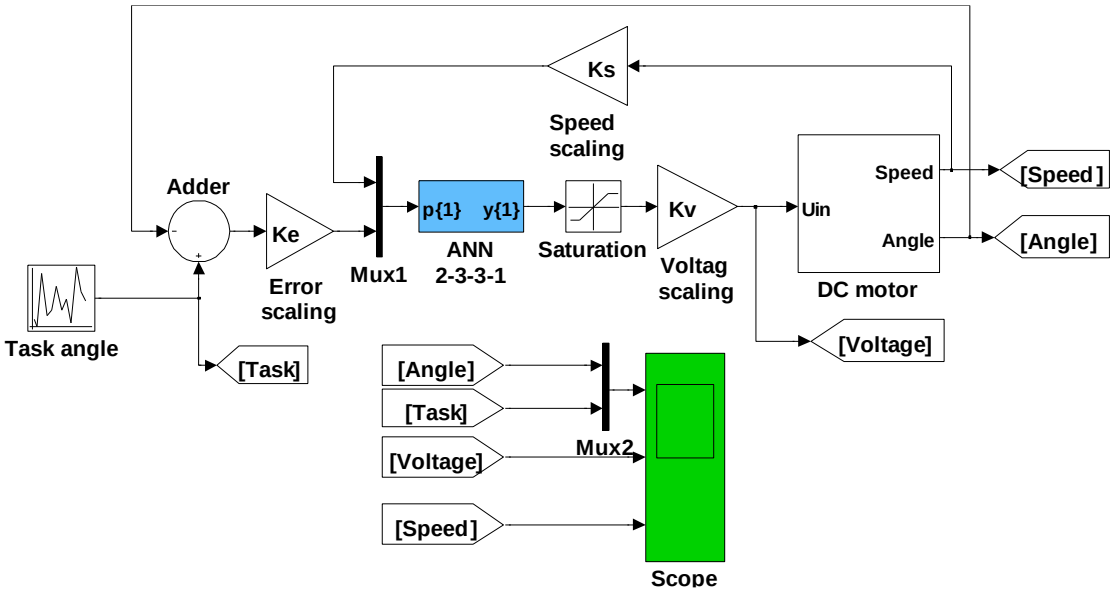


Fig. 1.10. The Simulink model of ANN based DC motor positioning system.

1.5. DCS simulation

The master controller parameters and ANN weights have been obtained for DC motor with permanent magnets that has anchor winding resistance R=0.7 Ohm, anchor winding inductance L=90x10^{-6} H, back emf constant C_e=0.056 V·s/rad, the rotor inertia J=130x10^{-7} kg·m^2.

Table 1.1 lists numerical values of weights and biases obtained for ANN. Fig. 1.11 shows Simulink model response for different given task angles. Transient process duration depends on rotation angle; hence system performance is close to optimal for all given angles. Moreover, transient response has no overshoot for all rotation angles.

The DC motor parameters used for system synthesis are not constant values and can changes during motor life cycle. Therefore, the control system should be robust to changes in DC motor parameters. The resistance of the anchor winding is frequently changeable parameter due to the anchor winding heating, by the current flowing through it.

Table 1.1 ANN parameters

Layer	Neuron #	Weights			Biases
First hidden	1	49.1173	-72.1763	–	-63.6570
	2	-2.3686	34.1922	–	0.9480
	3	3.6988	28.0580	–	7.3235
Second hidden	1	-5.5974	-8.7650	-5.6571	-3.1629
	2	-17.2153	-43.6494	10.2892	20.9795
	3	9.3155	15.2586	-0.3732	-1.6034
Output	1	-0.0183	0.5085	0.4813	-0.5089

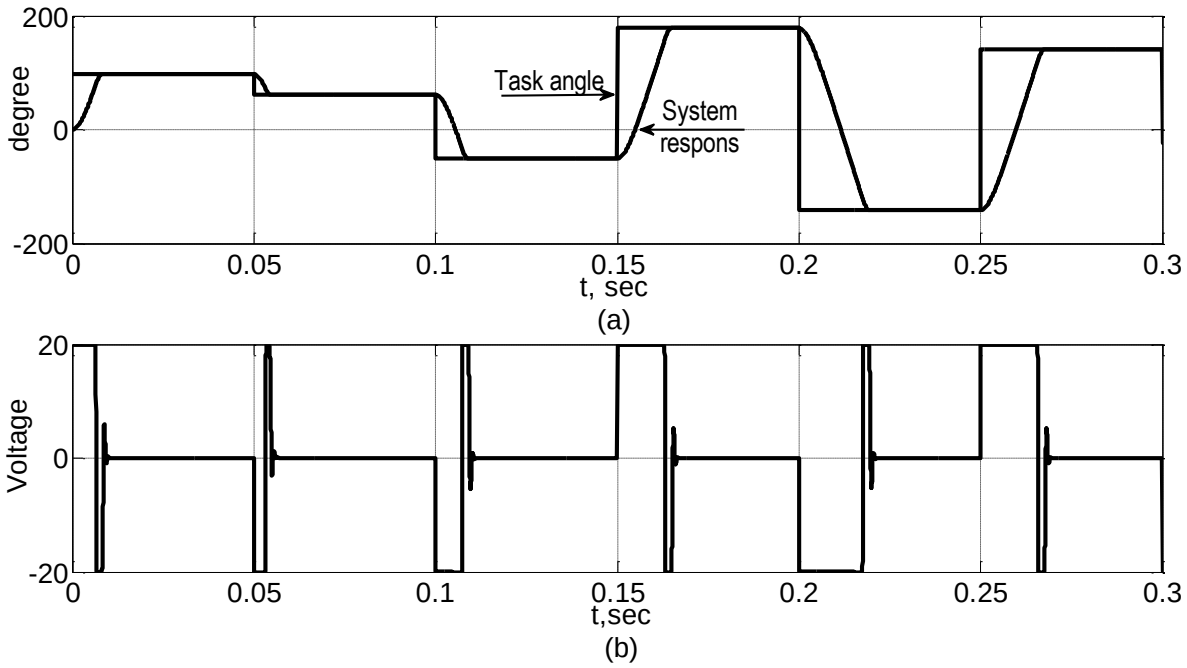


Fig. 1.11. Simulated transient process in ANN based DC motor positioning system: (a) step response for different task angles (b) applied voltage.

As can be seen from the diagrams shown in Fig. 1.12, even a twofold change in resistance does not lead to a significant deterioration in the quality of the transient process. The steady state error 0.1 degree or 0.057% has been obtained for the anchor resistance equal to the nominal value and half times greater than the nominal value. The steady state error 0.13 degree or 0.073% has been obtained for the anchor resistance two times greater than the nominal value. Transient process duration 20 ms has been obtained for the DC motor with a nominal anchor resistance and anchor resistance half times greater than the nominal. The transient process takes 20.1 ms for the DC motor with anchors resistance two times greater then the nominal. Also 0.55% overshoot approximately has been seen in this case.

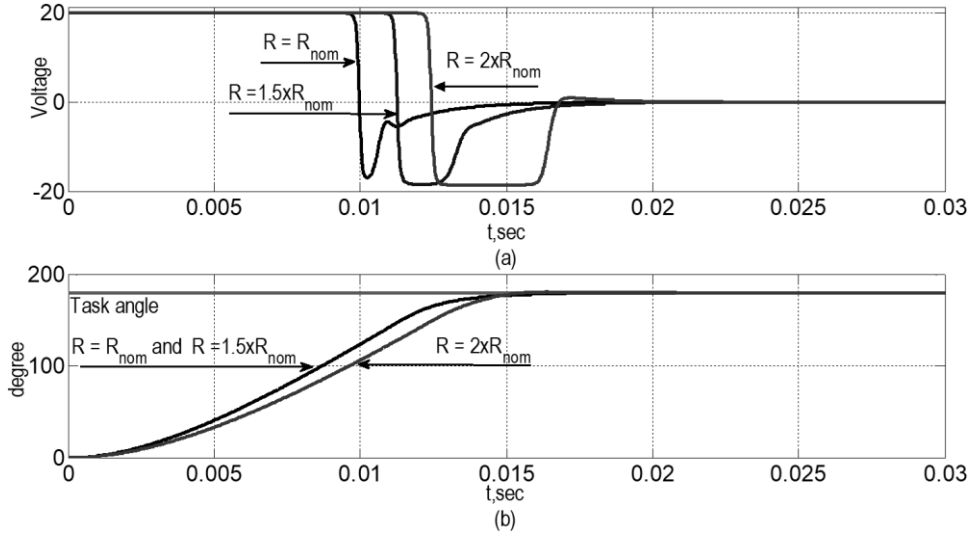


Fig. 1.12. Simulated transient process in ANN based DC motor positioning system for different rotor resistance value: (a) applied voltage (b) step response.

However, parameters of the DC motor are not the only one issue that should be taken into account when precision positioning system is being designed. The external factor such as supply voltage should be considered too. High frequency noises are the most common issue for all pulsed power sources which is widely used nowadays because of there high efficiency. Disturbances such as voltage drawdown are more likely being seen in the portable equipment power sources. They are caused by the battery partial discharge when it can not longer provide peak load currents.

Simulation results for the ANN based DC motor positioning system with voltage source disturbances such as 20%pulsations of rated voltage and 40% the maximum voltage drawdown are shown in Fig. 1.13. Significant disturbances to the supply voltage do not lead to the deterioration in the quality of the transient process. The steady state error is less than 0.04 degree. Hence the requirements to power supply can be reduced, which in turn may lead to lower costs, dimensions and weight of whole system. Such properties of the ANN based system determine it as sensible approach to the design of embedded and portable applications.

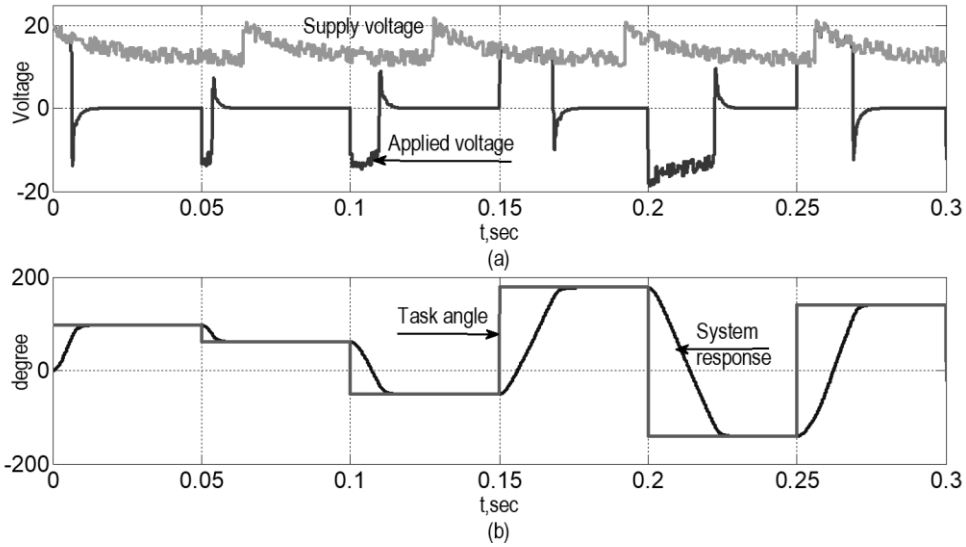


Fig. 1.13 Simulated transient process in ANN based DC motor positioning system with disturbances in supply voltage: (a) supply voltage and applied voltage (b) DC motor response.

1.6. Physical experiment

Fig. 1.14 shows main modules of the experimental system.

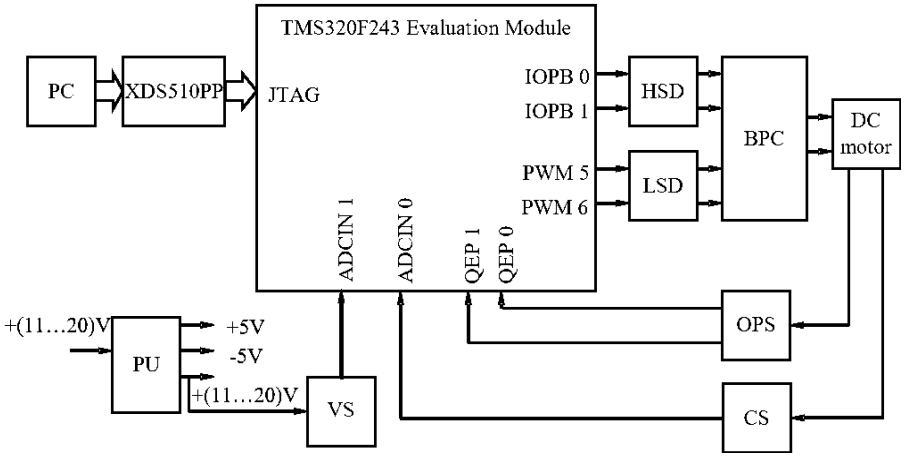


Fig. 1.14. The architecture of experimental system.

Here:

- *PC* – personal computer;
- *XDS510PP* – in-circuit emulator;
- TMS320F243 Evaluation Module;
- *PU* – power unit;
- *VS* – voltage sensor;
- *BPC* – bridge pulse convertor;
- *HSD* – high side driver of bridge pulse convertor;
- *LSD* – low side driver of bridge pulse convertor;
- *OPS* – optical position sensor;
- *CS* – current sensor.

The physical experiment system view is shown in Fig. 1.15.

Fig. 1.15. Physical experiment system view.

The architecture of experimental system is based on TMS320F243 Evaluation Module manufactured by Spectrum Digital Inc. for the F243 family of Texas Instruments signal processors. This evaluation module supports program developing and debugging in real time via JTAG interface.

The *HSM150* DC motor with maximum supply voltage 24 V, nominal current 8.5 A, nominal power 140 W and nominal speed 3650 rpm has been used. The MOSFETs have been used in bridge pulse convertor to perform high frequency pulse-width modulation (PWM). Such modulation can be obtained by built in hardware module that offloads the processing core for more priority tasks execution. The input-output lines *IOPB0* and *IOPB1*of signal processor drive high side MOSFETs, while low side MOSFETs are driven by the built-in hardware PWM thru lines *PWM5* and *PWM6.*

Voltage sensor delivers the information about device supply voltage to the internal analog-to-digital converter (ADC) thru pin *ADCIN1*. This information can be used to evaluate the influence of supply voltage quality on system parameters. The second input of the ADC via pin ADCIN2 is connected to the current sensor output, so the information about motor current can be obtained.

The specific internal module of signal processor is called quadrature encoder. It makes easy interface with optical position sensor. So, the quadrature encoder has a positive influence on system performance. It uses two signals QEP0 QEP1 that are shifted by 90 degree relative to each other and allows rotor speed and position tracking.

The personal computer performs a series of preparation tasks such as system simulation, ANN controller training, development of application software, loading compiled code in the program memory, real time debugging but do not influence on control process. So, the experimental design can be considered as fully functional and flexible embedded system.

The flowchart of ANN based control algorithm is shown in Fig. 1.16.

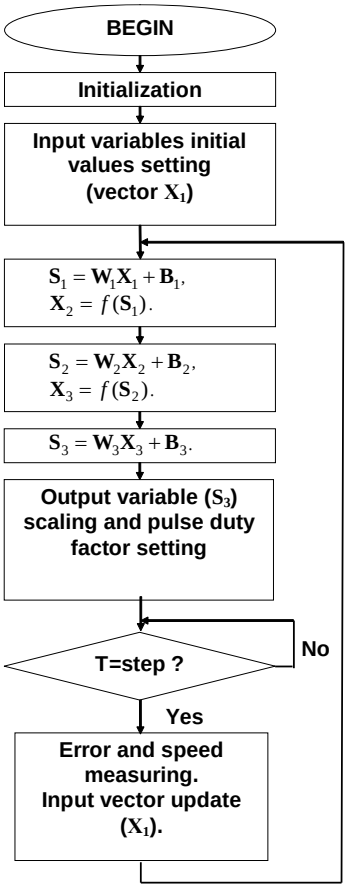


Fig. 1.16. Flowchart of ANN based control algorithm.

Here:

- X1 – column vector of input data that have dimension 2x1 (corresponding to the number of ANN inputs);
- W1 – matrix of weights for the first hidden layer that have dimension 3x2;
- B1 – column vector of biases that have dimension 3x1;
- S1 – column vector of weighted sums for the first hidden layer that have dimension 3x1;
- f – the hyperbolic tangent activation function that is given in tabular form;
- X2 – column vector that have dimension 3x1 is output of the first hidden layer, which is the input to the second hidden layer.

All other matrixes and vectors have the same meaning but for the following network layers and they have dimensionality: W2 - 3x3; B2 - 3x1; S2-3x1; X3 - 3x1; W3 - 1x3; B3 - 1x1; S3-1x1. Since the output layer has the linear activation function, a weighted sum S3 is also the output of the network.

Fig. 1.17 shows the waveforms of positioning for given angle equals 180 degrees.

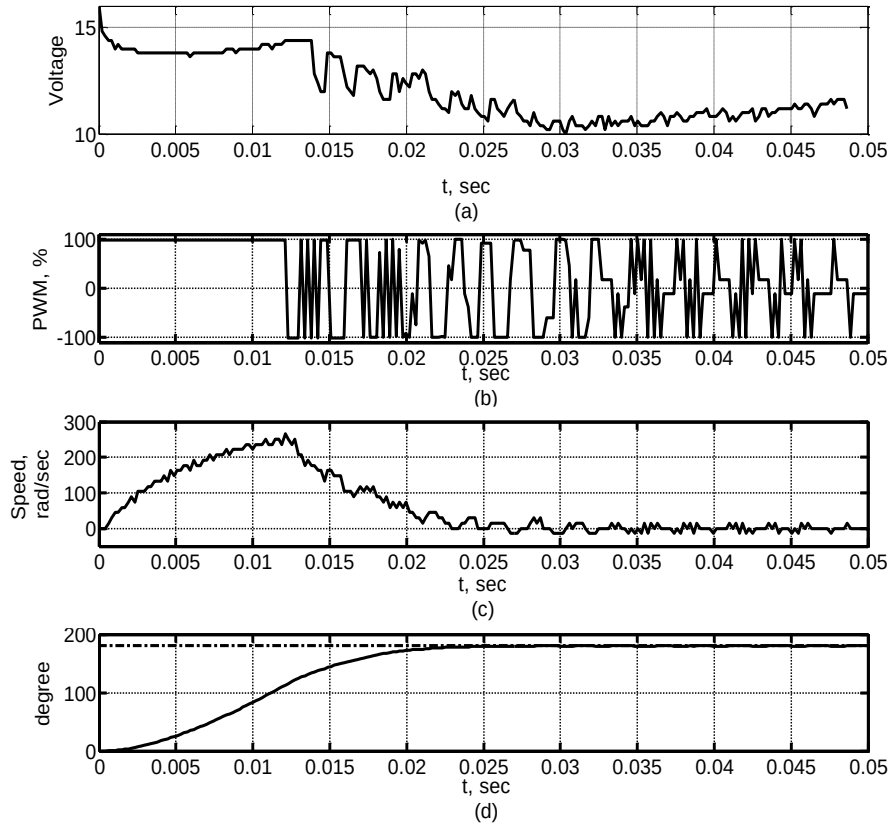


Fig. 1.17. Transient process by experimental data:
(a) supply voltage (b) signed PWM duty factor in percent (c) rotation speed (d) position angle.

The nominal supply voltage has been set to 16 volts. The sign of pulse duty factor determines the polarity of voltage applied to the anchor winding of the DC motor as shown in Fig.1. 17 (b). The static performance of the system (absolute positioning accuracy) and dynamic (the quality of the transient process) have been analyzed in this experiment. The transient response duration 26 ms has been obtained as shown in Fig. 1.17 (d). The maximum error in the steady state has not exceeded ±

0.54 degree (± 0.3%). Also, the transient process has no overshoot. In addition, the robustness to the supply voltage disturbances has been verified. Fig. 1.17 (a) shows that even a voltage drawdown up to 37% has no crucial influence on system response.

Fig. 1.18 shows the transient processes for the angles 90 and 45 degree as the result of computer simulation and experimental data. Obviously, the diagrams of computer simulation and experimental data are quite similar. Differences in the rise area of the transient curve can be explained, in particular, by the difference between real DC motor parameters and parameters that have been used for the development of a master controller. In addition, the friction has not been taken into account in simulink model. However, the friction influence over transient process. Its influence can be seen from the curves as a delay in the initial stage of acceleration for experimental system and its faster braking than the simulation data shows.

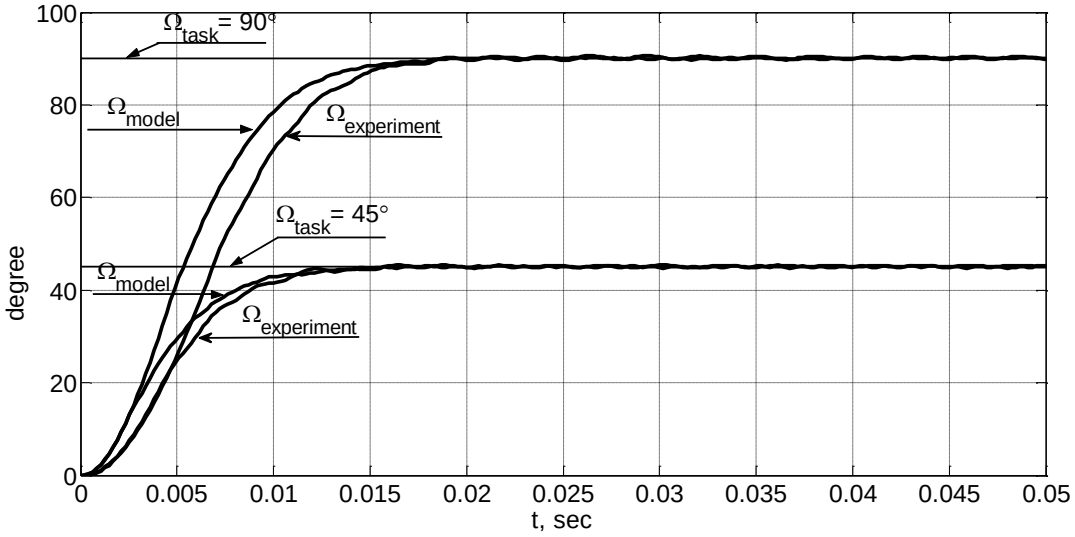


Fig. 1.18. Comparative diagram of computer simulation and experimental data.

1.7. Summary

The embedded system based on neural network algorithm has been implemented on the signal processor TMS320F243. The experimental results fully confirm the theoretical calculations and computer simulations, so the approach based on artificial neural networks in combination with variable gain method could be suggested for DC motor positioning system syntheses.

The experimental system is robust to supply voltage disturbance and load parameters variation. The achieved transient process is close to optimal without overshoot.

Further improvements could be suggested to enhance system dynamics and minimize steady state error. Using more performance controller will lead to increase the sampling rate of control algorithm and improve the quality of the positioning for small angles. Using a sensor with higher resolution will reduce the speed quantization error which is very important especially for the low-velocity areas such as braking zone near the given angle.

References

1. Benjamin C. Kuo, Digital control systems, Holt, Rinehart and Winston, 1980.
2. Julius T. Tou, Modern Control Theory, McGraw-Hill Education, 1964.
3. Jay C. Hsu, Andrew U. Meyer, Modern control principles and applications, McGraw-Hill, 1968.
4. Roland S. Burns, Advanced Control Engineering, Linacre House, Jordan Hill, Oxford. – 2001.

5. Simon Haykin, Neural Networks: A Comprehensive Foundation (2nd Edition), Prentice-Hall, Inc. – 1999.
6. Rios-Gutierrez F., Makableh Y.F., Efficient position control of DC Servomotor using backpropagation Neural Network, Natural Computation (ICNC), 2011 Seventh International Conference on, 2011, Vol.2, pp. 653 - 657.
7. M.N. Cirstea, A. Dinu, J.G. Khor, M. McCormick, Neural and Fuzzy Logic Control of Drives and Power Systems, Linacre House, Jordan Hill, Oxford, 2002.
8. V. Shanmuga, T. Jayabarathi, Load Frequency Control Using PID Tuned ANN Controller in Power System, Electrical Energy Systems (ICEES), 2011 1st International Conference, 2011, pp. 269 – 274.
9. Cybenko G. Approximation by superposition of a sigmoidal function. Mathematics of Control, Signals, and Systems, 1989. Vol. 2. pp. 303 – 314.
10. Kurt Hornik Approximation Capabilities of Multilayer Feedforward Networks, Neural Networks, 1991. Vol. 4, pp. 251–257.
11. Duman S., Maden D., Guvenc U., Determination of the PID controller parameters for speed and position control of DC motor using Gravitational Search Algorithm, Electrical and Electronics Engineering (ELECO), 2011 7th International Conference, 2011, pp. I-225 - I-229.
12. Yanzhu Zhang, Jingjiao Li, Fractional-order PID controller tuning based on genetic algorithm, Business Management and Electronic Information (BMEI), 2011 International Conference, Vol.: 3, 2011, pp. 764 – 767.
13. Ashry M.M., Kamalova Z.Z., Breikin T.V., Tuning of digital PID controller parameters using local optimal control, Control and Automation, 2008 16th Mediterranean Conference, 2008, pp. 587 – 592.
14. Gwo-Ruey Yu, Ming-Hung Tseng, Yuan-Kai Lin, Optimal positioning control of a DC servo motor using sliding mode, Control Applications, 2004. Proceedings of the 2004 IEEE International Conference, Vol. 1, 2004, pp. 272 – 277.
15. Moradi M., Ahmadi A., Abhari S., Optimal control based feedback linearization for position control of DC motor, Advanced Computer Control (ICACC), 2010 2nd International Conference Vol. 4, 2010, pp. 312 – 316.
16. Cas J., Klobucar R., Safaric R., Neural network-based control of micro-manipulator, Advanced Motion Control, 2008. AMC '08. 10th IEEE International Workshop 2008, pp. 444 – 449.
17. Rios-Gutierrez F., Makableh Y.F., Efficient position control of DC Servomotor using backpropagation Neural Network, Natural Computation (ICNC), 2011 Seventh International Conference Vol. 2, 2011, pp. 653 – 657.
18. Castañeda C.E., López-Mancilla D., García J.H., Reátegui R.J., Huerta G., Zárate R.C., Position control of dc motor based on recurrent high order neural networks, Intelligent Control (ISIC), 2010 IEEE International Symposium, 2010, pp. 1515 – 1520.

Chapter 2. BLDC motor digital control systems

2.1. Brushless DC Motors

Brushless DC (BLDC) motors, also known as Electronically Commutated Motors (ECMs), have become an important actuator in modern engineering, with widespread applications ranging from consumer electronics to industrial automation, robotics, and aerospace systems. Their popularity is due to a number of significant advantages that make BLDC motors an optimal choice for high-performance and reliability-critical systems.

BLDC motors exhibit superior energy efficiency, primarily due to the elimination of frictional losses associated with the absence of a brush-commutator assembly. This design improvement enables more efficient energy conversion and significantly reduces thermal dissipation. In addition, electronic commutation enhances the overall performance by allowing for precise and adaptive control of motor operation [1].

The brushless architecture also significantly contributes to the durability and reliability of BLDC motors. By eliminating mechanical contact between brushes and the commutator, a major source of wear is removed, thereby extending operational lifespan and improving dependability in continuous and mission-critical applications. This, in turn, lowers maintenance demands and reduces the frequency of failures, ultimately resulting in considerable cost savings for industrial plants.

Precise speed and torque control is another key advantage. Thanks to electronic commutation, BLDC motors offer high-precision control, which is critical in applications where specific and constant speeds are required, such as robotics, electric vehicles and industrial equipment. In addition, the design of BLDC motors helps reduce noise and vibration during operation, which is an important factor in areas such as medical equipment, heating, ventilation, and air conditioning systems (HVAC) and household appliances.

BLDC motors also feature high power density, delivering significant power output in a compact size, making them ideal for space-constrained applications such as electric vehicles and drones. [2] Their versatility and adaptability allow for customization of the number of rotor poles, stator windings, and control electronics, making them suitable for a wide range of industries, including robotics, automotive, and aerospace.

BLDC motors are used in a wide range of sectors. In the automotive industry, they are used to power electric vehicle drive systems and in steering systems. In industrial automation, they drive robotic manipulators, conveyor belts, and computer numerical control (CNC) machine tools. In consumer electronics, they are integrated into HVAC systems and household appliances for energy efficiency and quieter operation. The aerospace and aviation industries employ them in propulsion systems for both fixed-wing and multirotor drone platforms. They are also an integral part of wind turbine generators in renewable energy and medical equipment, which require low noise and high reliability.

The operation of a BLDC motor is based on the interaction of a magnetic field generated by permanent magnets on the rotor and an electromagnetic field produced by the stator windings [3]. Unlike traditional commutator motors, BLDC motors consist of a rotor made of permanent magnets and a stator containing coils that are powered in series.

Rotational motion is achieved by sequentially energizing the stator windings. An electronic controller plays a crucial role in controlling this sequence of switching the stator windings to ensure smooth rotation. Electromagnetic interaction produces torque that drives the rotor rotation. The rotor, which contains permanent magnets, follows the electromagnetic fields of the stator when they are activated, ensuring continuous motion [4].

2.2. Classification of BLDC motors

BLDC motors, although they share a common basic structure, differ in design, electromagnetic and functional characteristics. Depending on the specific application, size, torque, accuracy or efficiency requirements, different types of BLDC motors are used, each of which has its own advantages and limitations.

By rotor design. One of the most important criteria for dividing BLDC motors is the location of the permanent magnets relative to the rotor. According to this parameter, two main types are distinguished: inner rotor (in-runner) and external rotor (out-runner) (Fig. 2.1).

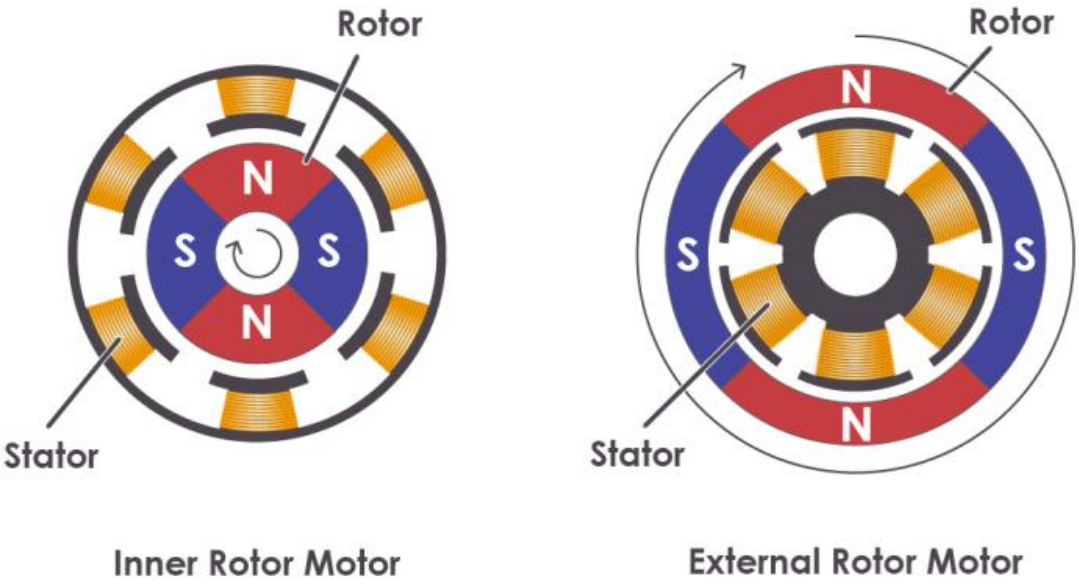


Fig.2.1. BLDC rotor design diagrams [5].

Inner rotor is a classic design option in which permanent magnets are placed on a rotor shaft that rotates inside a fixed stator with windings. This type is typical for most industrial systems, fans, power tools. It allows for high rotation speeds and efficient heat dissipation through a fixed stator. Such motors are typically characterized by a compact design and are compatible with standard controllers.

The outer rotor, on the other hand, has fixed windings located inside, and permanent magnets attached to a cylindrical rotor shell that rotates around the stator. This arrangement allows for increased torque due to a larger radius of rotation of the magnets, so it is often used in drones, small-sized motor vehicles, cooling valves and acoustic devices. The outer rotor usually rotates at a lower speed, but provides more torque without a gearbox.

By number of phases. Another important criterion for classifying BLDC motors is the number of phases of the stator windings that participate in creating the rotating magnetic field. Although three-phase BLDC motors are the most common, in real applications there are both single-phase and even multi-phase models. Each configuration has its own characteristics in terms of structure, control and application.

Single-phase BLDC motors are the simplest configuration in terms of design and control. Typically, such a motor uses one winding with a middle contact (or two symmetrical windings) that form a magnetic field in two directions. Switching is performed by a controller using an H-bridge circuit, and the rotor position is determined using one or two Hall sensors. Due to their simple structure, such motors are often used in fans, coolers, miniature pumps and household appliances.

However, they have significant limitations - uneven torque, low dynamics and limited controllability under variable load.

Three-phase BLDC motors are the most common option, which has become a kind of industry standard due to the optimal balance between efficiency, smooth rotation, simplicity of design and the availability of a wide range of controllers. In a three-phase BLDC motor, three windings are connected in a "star" or "delta" configuration, and are switched through an inverter with three pairs of transistors. This allows for 6-position (trapezoidal) or sinusoidal switching with a high degree of control. Due to the widespread use of this topology, it is supported by most microcontrollers and drivers, which makes it universal for drones, robots, electric vehicles and industrial servo systems.

Multiphase BLDC motors (5, 6, 7 phases and more). In highly specialized systems where ultra-high accuracy, reduced vibration, increased reliability or fault tolerance are important, BLDC motors with a larger number of phases are used. A multiphase system allows for reduced torque ripple, increased efficiency at low speeds, and provides redundancy: even if one or two phases are lost, the motor can maintain functionality (albeit with limited power). However, controlling such systems is more complex and requires specialized software and a more complex inverter with a larger number of power switches. Because of this, multiphase BLDCs are mostly used in aviation drives, defense systems, precision medical technology or high-precision automation.

By the form of back electromotive force (EMF). The principle of operation of a BLDC motor is based on the interaction of the stator magnetic field and the back electromotive force generated by the rotation of the rotor. Depending on the shape of this back EMF, trapezoidal and sinusoidal BLDC motors are distinguished.

In trapezoidal motors, the EMF shape is similar to a stepped (trapezoidal) one, which corresponds to a control scheme with six switching positions. Such motors are easier to control, do not require complex PWM modulation algorithms, and are well suited for systems where torque ripple is not critical - for example, in fans or pumps. They are also more often used in household applications and in low-power equipment due to the cheapness of the controller.

Sinusoidal BLDC motors have a back EMF that is close to a sinusoidal waveform and require appropriate control using sinusoidal modulation. This minimizes vibration, noise, and torque ripple, which is critical for precision servo drives, medical equipment, or precision mechanisms. Such motors are more expensive because they require more sophisticated position sensors and more complex controllers.

By rotor position control method. Another important criterion is the method of determining the rotor position required for correct phase switching. Here there is a division into sensor and sensorless motors.

Sensored BLDCs use built-in position sensors – most often three Hall sensors placed at specific angles (usually 120° apart) that generate signals that accurately determine the position of the magnets. This allows for accurate and stable commutation even at low speeds or starting with a load. Sensored systems have higher reliability in difficult conditions, but are more expensive and require precise calibration.

Sensorless control relies on measuring back EMF or phase currents to estimate rotor position. This reduces motor cost and increases mechanical reliability (fewer components), but makes motor starting more difficult and reduces accuracy at low speeds. This approach is particularly popular in drones, fan systems, and portable equipment where weight and cost minimization are critical.

2.3. BLDC motor control methods

Effective control of BLDC motors necessitates comprehensive knowledge of electronic commutation principles and rotor position sensing methods, as these constitute the foundation for attaining the specified performance characteristics in terms of speed, torque, and overall efficiency.

The speed and torque of the motor are controlled by varying the timing, duration, sequence, amplitude, and polarity of the pulses applied to the stator windings [4].

BLDC motors use electrical switches to switch the current, which ensures continuous rotation of the motor. MOSFET or IGBT transistors are used as switches. These electrical switches are usually of a three-phase bridge structure for a three-phase BLDC motor, as shown in Fig. 2.2.

The transistors in such a circuit are controlled using pulse-width modulation (PWM), which converts a DC voltage into a modulated voltage, which easily and effectively limits the starting current, control speed, and torque. Increasing the switching frequency generally raises PWM-related losses, whereas lowering the switching frequency reduces the system bandwidth and can lead to greater current ripple [6].

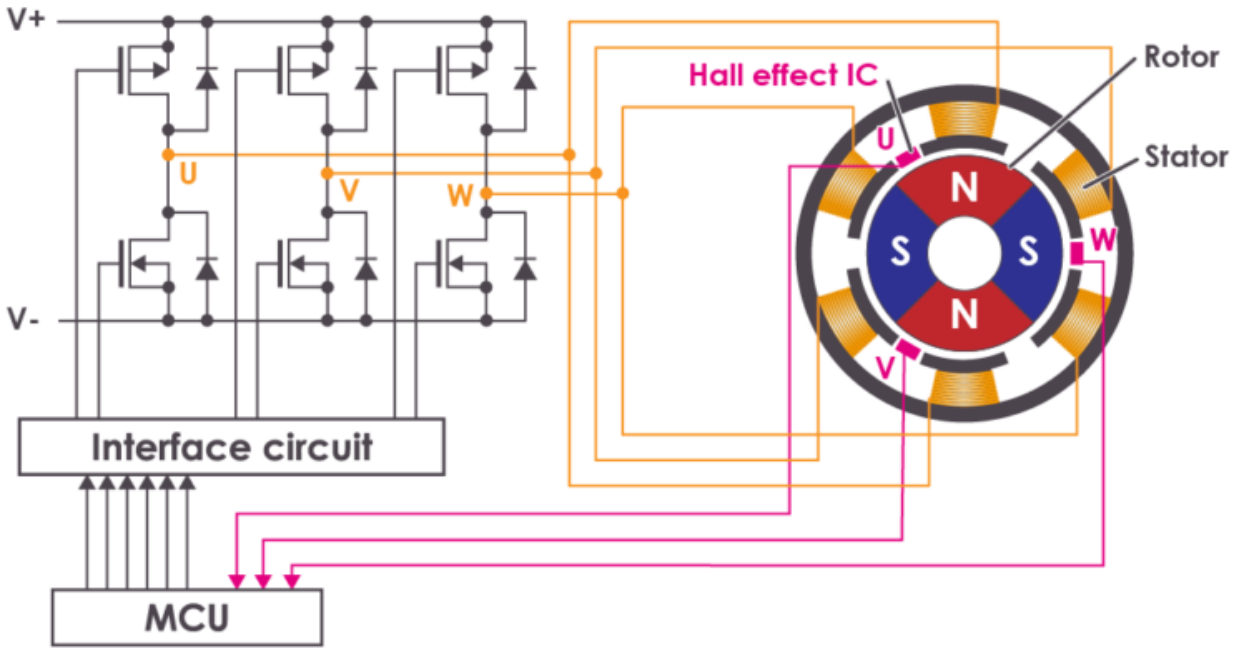


Fig. 2.2. BLDC motor control scheme [5].

2.3.1. Trapezoidal (6-step) control method

This method is one of the most common and easiest to implement [7]. It involves simultaneously powering two phases of the motor, while the third phase remains open (floating) [4]. The magnetic field generated changes by 60 electrical degrees for each step, forcing the rotor to follow it [7].

A three-phase BLDC motor requires three Hall sensors to determine the rotor position. Based on the physical location of the Hall sensors, there are two types of output signals: 60° phase shift and 120° phase shift. The combination of these three Hall sensor signals can determine the exact switching sequence.

The switching sequence of a three-phase BLDC motor control circuit for counterclockwise rotation is illustrated in Fig. 2.3.

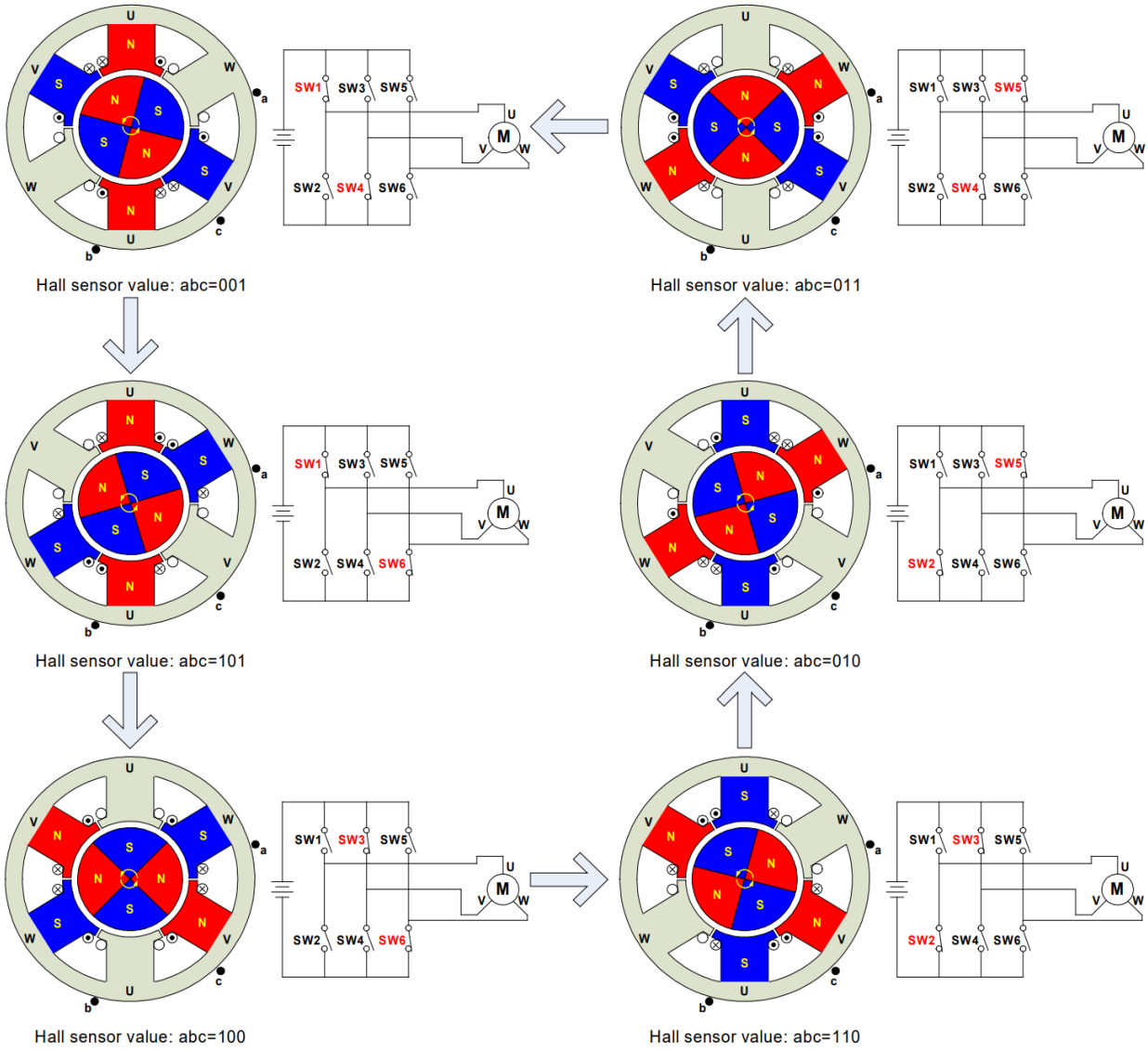


Fig. 2.3. BLDC Motor Commutation Sequence [8].

Three Hall-effect sensors (a, b, and c) are mounted on the stator at 120° angular intervals, while the three-phase windings are connected in a star (wye) configuration. Every 60° of rotor rotation, one of the Hall sensors changes its state, resulting in a six-step commutation sequence for completing a full electrical cycle. In synchronous commutation mode, the phase current is updated every 60°. In each step, one motor output is driven high, the other motor output is driven low, and the third is left floating. Separate controls for the high and low signal drivers allow the high, low, and floating currents on each motor output to be controlled.

However, one signal cycle may not correspond to a complete mechanical rotation. The number of signal cycles to complete a mechanical rotation is determined by the number of pole pairs in the rotor. Each pair of poles in the rotor requires one signal cycle for one mechanical rotation. Therefore, the number of signal cycles is equal to the number of pole pairs in the rotor.

Fig. 2.4 shows timing diagrams where the phase windings: U, V and W are either energized or de-energized based on the Hall sensor signals: a, b and c. This is an example of the Hall sensor signals having a phase shift of 120° relative to each other when the motor is rotating counterclockwise. Generating a Hall signal with a phase shift of 60° or rotating the motor clockwise requires a different timing sequence.

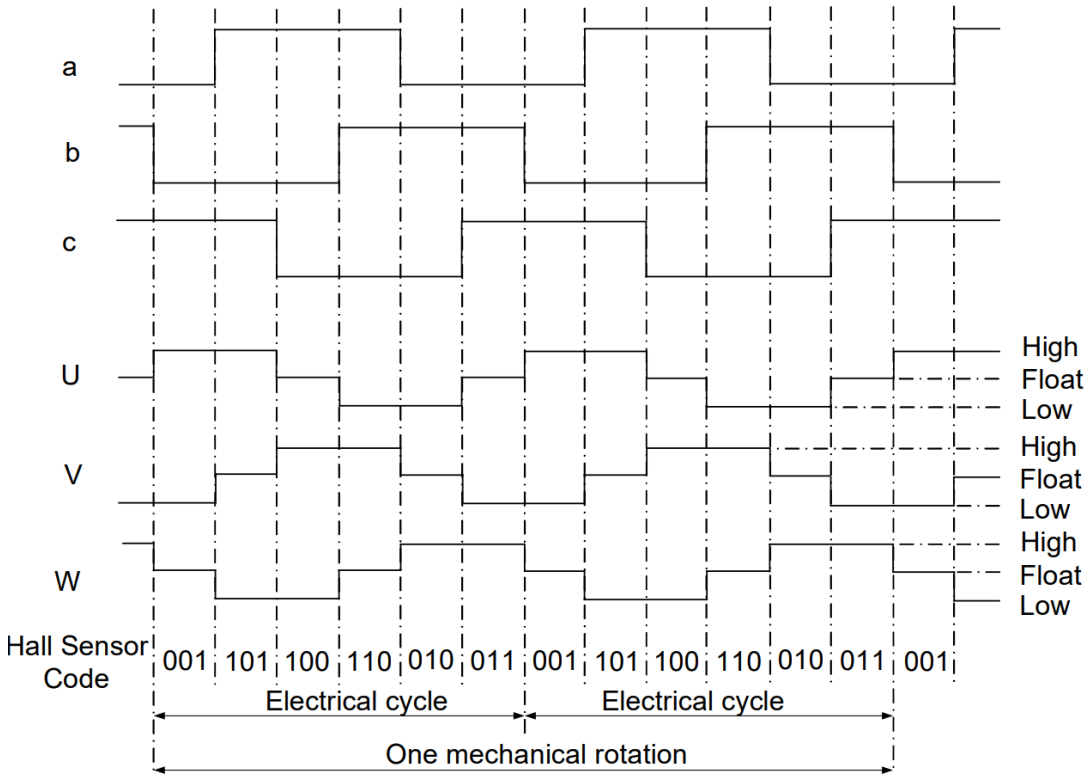


Fig. 2. 4. Timing diagram of six-step commutation control by BLDC motor [8].

To change the rotational speed, PWM signals are applied at a frequency significantly higher than the motor's rotational frequency. Typically, the PWM frequency should be at least ten times greater than the motor's maximum rotational frequency. Another advantage of PWM is that when the DC bus voltage substantially exceeds the motor's rated voltage, limiting the PWM duty cycle allows the stator winding current to be maintained at its nominal value.

Although this approach is relatively simple, it can lead to torque ripple due to imperfect alignment of the magnetic fields, which affects the smoothness of the motor [6].

2.3.2. Field vector control method

The basic idea of the Field-Oriented Control (FOC) method is to convert the alternating three-phase current into a rotating coordinate system that is rigidly related to the rotor position and then control the currents in this system to create the optimal magnetic torque. In contrast to 6-step commutation, FOC is considered the most efficient method of controlling BLDC motors [4]. This method uses three sine waves shifted by 120 degrees to create rotation. The amplitude of these sine waves is the main factor in controlling the speed and torque. FOC is much more complex to implement, but provides smoother operation, less torque ripple and higher efficiency. This is achieved because FOC seeks to maintain the stator magnetic field perpendicular to the rotor permanent magnet field, which maximizes the output torque and reduces ripple [6].

FOC involves transforming a three-phase current system into a two-phase (stationary) coordinate system using the Clarke transformation, and then into a two-phase coordinate system rotating with the rotor using the Park transformation. In this rotating system, currents are regulated, after which the results are converted back into a three-phase system to generate PWM control signals (Fig. 2.5). Typically, either a sinusoidal modulation scheme (SPWM) or a space vector modulation scheme (SVPWM or SVM) is used for this process.

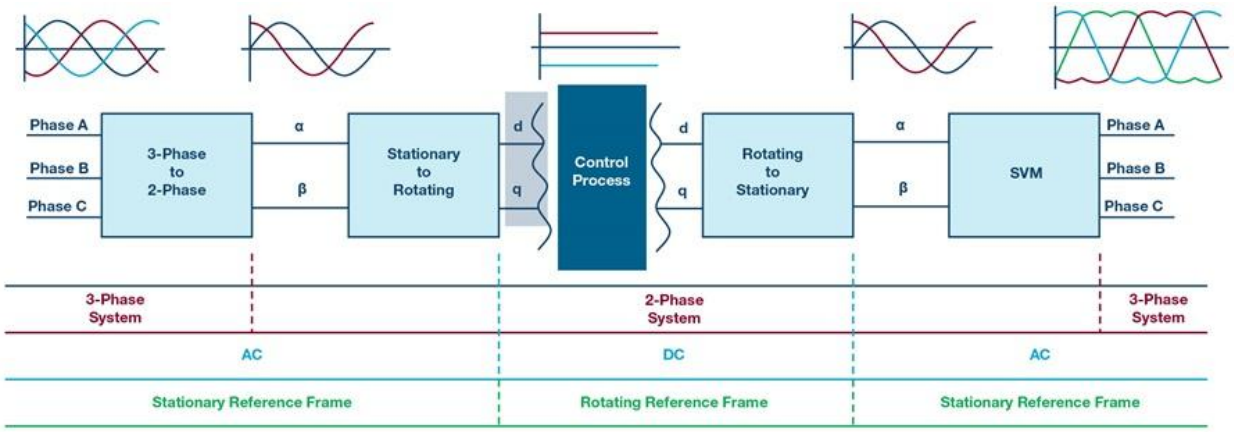


Fig. 2. 5. Transformation of control parameters in FOC [9].

The Clarke transformation aims to translate three-phase currents into a fixed orthogonal α-β coordinate system (a two-dimensional plane).

Suppose that the currents in the windings are denoted as i_a, i_b, i_c, then the Clarke transformation:

$$i_\alpha = i_a, \quad (2.1)$$
$$i_\beta = \frac{1}{\sqrt{3}}(i_a - i_c). \quad (2.2)$$

Or in a more general form:

$$i_\alpha = \frac{2}{3} i_a - \frac{1}{3}(i_b + i_c), \quad (2.3)$$
$$i_\beta = \frac{1}{\sqrt{3}}(i_a - i_c). \quad (2.4)$$

Next, the coordinates i_α, i_β, are transformed into a rotating coordinate system dq, where the d axis is directed along the rotor magnetic field, and the q axis is perpendicular to it. This allows the torque and excitation to be controlled separately.

$$i_d = i_\alpha \cos(\theta) + i_\beta \sin(\theta), \quad (2.5)$$
$$i_q = -i_\alpha \sin(\theta) + i_\beta \cos(\theta), \quad (2.6)$$

where θ is the angle of the rotor position relative to phase a;

i_d – current along the magnetization axis (usually adjusted to 0 for BLDC);

i_q – current that creates a moment.

In dq-space, regulation is carried out (for example, by a PI controller or another method) for each component separately:

- the i_q regulator controls the engine torque;
- the i_d regulator reduces magnetization (or maintains it at a given level).

The result of the regulator is the new values of the voltages v_d, v_q, which after the inverse transformation return to v_α, v_β:

$$v_\alpha = v_d \cos(\theta) - v_q \sin(\theta), \quad (2.7)$$
$$v_\beta = v_d \sin(\theta) + v_q \cos(\theta). \quad (2.8)$$

Then the three-phase system of voltage values v_a, v_b, v_c is determined:

$$v_\alpha = v_a, \tag{2.9}$$

$$v_b = -\frac{1}{2}v_\alpha + \frac{\sqrt{3}}{2}v_\beta, \tag{2.10}$$

$$v_c = -\frac{1}{2}v_\alpha - \frac{\sqrt{3}}{2}v_\beta. \tag{2.11}$$

These voltages are generated by an inverter, the microcontroller of which generates the corresponding PWM signals to control the power transistors to power the motor windings.

Next, to achieve closed-loop control, feedback must be integrated.

To determine the rotor position in BLDC motors, Hall sensors or encoders are traditionally used. Typically, three Hall sensors are embedded in the stator of the motor to detect the rotor position and provide the corresponding information to the controller. Each Hall sensor generates a 180° pulse signal, and since they are offset by 120° relative to each other, their combinations create six unique states indicating the rotor position every 60 electrical degrees [10].

Sensorless control is an alternative approach that eliminates the need for physical position sensors. Instead, it relies on detecting the back electromotive force (Back-EMF or BEMF) induced in the inactive phase of the motor [3]. BEMF is the voltage generated in the stator windings of a BLDC motor due to the rotation of the rotor in a magnetic field, which opposes the applied voltage.

When a BLDC motor rotates, the rotation of the permanent magnet rotor generates an alternating magnetic field. Each phase of the winding experiences BEMF. The key moment to determine the rotor position is to detect the zero crossing of the BEMF in the phase that is not energized (the open-loop phase). This zero crossing indicates the moment when the phases must be switched to continue rotation. After detecting the zero crossing of the BEMF, a delay of 30 electrical degrees must be set before the proper commutation is performed [7].

The advantages of sensorless control are significant: it reduces the cost and size of the system, simplifies wiring (from 8 to 3 wires) as there is no need for additional wires for sensors, and is less sensitive to external factors such as temperature and magnetic fields. This increases the overall reliability of the system.

However, sensorless control has its drawbacks. It requires more complex control systems and algorithms that require specialized knowledge. In addition, sensorless methods can have difficulties at very low speeds and during engine start-up, as the BEMF is too low to be reliably detected.

For simple fans or pumps, 6-step switching may be sufficient. However, for precision robots or electric vehicles, where smoothness and efficiency are key, FOC becomes necessary, despite its complexity. Modern microcontrollers and software libraries (e.g., STM32 Motor Control SDK) are trying to lower the barrier of entry for FOC, making it affordable.

2.4. MCU based BLDC motor control system

2.4.1. STM32 microcontrollers for BLDC control system

The choice of microcontroller is an important step in the development of a BLDC motor control system. Vector control requires high computational performance, accurate current measurement, fast PWM and feedback from rotor position sensors. Therefore, the microcontroller must meet the key requirements listed in Table 2.1.

STMicroelectronics' STM32 microcontrollers have gained wide recognition due to their versatility, performance, and availability of specialized peripherals, making them ideal for such tasks.

Table 2.1 Microcontroller requirements for implementing FOC

Criterion	Requirements for FOC
Computing performance	At least 72 MHz (better >100 MHz)
ADC	At least 2-3 channels, preferably with DMA
PWM	At least 3 PWM phases with "dead time"
Timers	Support for center-symmetrical PWM
Fixed/Floating Point Operations	FPU is desirable
UART / CAN / USB	For debugging or interface
Hall/Encoder support	Interface for reading rotor position

STM32 microcontrollers based on Arm® Cortex®-M cores (M0 to M7 and M33) are particularly well-suited for BLDC motor control due to their broad portfolio, industry-standard compliance, rich analog peripherals, and advanced motor control timers [10].

Computing Power and Speed: To implement complex control algorithms such as FOC and to efficiently process sensor data (e.g., BEMF), the microcontroller must have sufficient computing power [3]. The different STM32 series offer a wide range of central processing unit (CPU) frequencies, from 48 MHz (Cortex-M0) to 480 MHz (Cortex-M7), allowing the microcontroller to be selected according to specific performance requirements [11]. The presence of a floating-point unit (FPU) and digital signal processing (DSP) instructions in the Cortex-M4 and Cortex-M7 cores significantly accelerates complex mathematical calculations that are critical for FOC algorithms.

Pulse Width Modulation: High resolution PWM is essential for smooth control of the power and amplitude delivered to the motor. This allows for finer tuning of the duty cycle, providing more precise speed and torque control. The STM32 advanced timers, such as TIM1 and TIM8, support a special combined 3-phase PWM output mode, which is extremely useful for driving MOSFET bridge drivers [12]. Center-Aligned PWM is preferred for multiphase motor control and power electronics because it reduces switching losses and electromagnetic interference (EMI). In addition, these timers can generate complementary PWM outputs (inverse of each other) and insert dead-time, during which both complementary channels are in a low state. This is critical to prevent shoot-through currents in the half-bridge MOSFET drivers, which can short-circuit the power supply. The STM32 timers also have a Break input for emergency stop, which can instantly turn off all 6 PWM outputs and generate interrupts, providing hardware protection [13].

Input Capture modes for reading signals from Hall sensors and encoders, allowing pulse duration measurement and rotor position determination [14]. Timers can be configured to operate in encoder modes (x2, x4), allowing direct acquisition of rotor angle or position.

Analog-to-Digital Converter (ADC): STM32 microcontrollers feature 12-bit ADCs with high sampling rates (up to 1 Msps or even 3.75 Msps on some series) and the ability to perform simultaneous conversions. This is critical for accurate and timely real-time measurement of winding currents and back EMF (BEMF). Injected ADC channels have higher priority than regular channels and are capable of interrupting ongoing conversions to immediately perform critical measurements, such as current at the moment of PWM switching. This allows for accurate synchronization of current measurements with the PWM cycle, which is vital for accurate control algorithms. The ability to synchronize the ADC with PWM timers ensures accurate reading of current feedback at the right time.

STMicroelectronics company has specialized microcontroller lines designed for electric drive control. For example, STM32G4 is a new line of microcontrollers for motor control:

- powerful core (Cortex-M4, 170 MHz);
- high accuracy of PWM and ADC;
- has specialized peripheral blocks for FOC (HRPWM, HRTIM, FMAC, CORDIC) for fast trigonometric calculations;
- support for Hall sensors, encoders, 3x ADCs.

STMicroelectronics also provides a comprehensive motor control ecosystem, including SDK (X-CUBE-MCSDK), Motor Control Workbench, and Motor Profiler, which greatly simplifies the development process.

2.4.2. FOC implementation on the STM32G431RB microcontroller

Configuring the microcontroller's peripherals. To configure the main peripherals of the STM32G431RB microcontroller, we will use the free STM32CubeMX utility from STMicroelectronics - this is a program for graphically configuring the peripherals of STM32 microcontrollers and automatically generating project initialization code for STM32CubeIDE or other IDEs.

Let's set up the TIM1 timer to generate three synchronous PWM signals (phases A, B, C) to control the inverter transistors:

- Timer type: Advanced-control timer;
- Mode: Center-aligned mode 1 (symmetrical PWM with zero average harmonics);
- Channels: Activate Channel 1, 2, 3 as PWM Generation CHx and CHxN;
- Dead-Time: Enable Dead-Time Insertion (typically 1-2 μs) to avoid through-current;
- Break Input: Enable fault protection (overcurrent);
- Trigger Output (TRGO): Set Update Event or Compare Event - used to trigger the ADC.

Let's set up the TIM2 timer to measure the rotor position:

- Mode: Hall Sensor Mode;
- Specify three inputs (TIM2_CH1, TIM2_CH2, TIM2_CH3);
- Enable interrupts on update and Capture Compare.

Let's configure the ADC1 for measuring phase currents for further Clarke and Park transformations. Works in close conjunction with timer TIM1:

- Channel mode: Injected channels - triggered by an event from the TIM1 timer;
- Number of channels: three;
- Start trigger: Set External Trigger from TIM1 TRGO or Timer1 CCx event;
- Interrupts: Enable Injected End Of Conversion Interrupt;
- Sampling time: 7.5 – 47.5 ADC cycles (compromise between accuracy and speed);
- Operating mode: Independent mode.

Configuring Direct Memory Access (DMA) and Interrupts (NVIC). To quickly copy ADC results to a variable without CPU load or processing priority management.

DMA: Enable for ADC1 Injected Channels.

NVIC (interrupt):

- ADC1_IRQn;
- TIM1_UP_TIM16_IRQn – for PWM and regulator processing;
- TIM2_IRQn – processing the position from the encoder (if required).

GPIO settings:

- device operation control and monitoring;
- inverter operation control (EN, DIS);
- status output (LED);
- engine start/stop control buttons;
- UART interface for parameter monitoring (via CubeMonitor, PC GUI, etc.).

For example:

- Set the necessary pins (GPIO_Output for Enable, GPIO_Input with PullUp for buttons);
- UART1 - for debugging or data transfer.

The microcontroller settings window is shown in Fig. 2.6.

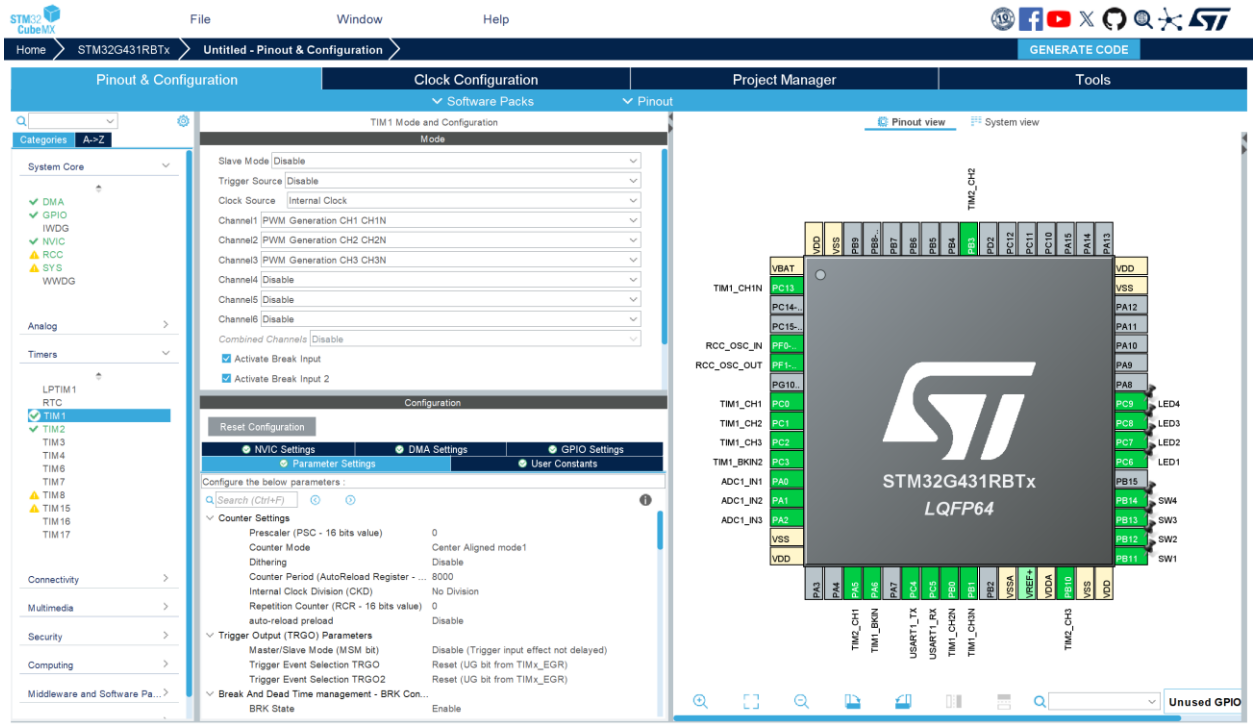


Fig. 2.6. Microcontroller settings window in CubeMX.

The algorithm of operation of the FOC system on STM32G431RB with Hall sensors includes next steps:

1. System initialization.

When starting the program:

- Initialization of microcontroller peripherals (TIM1, TIM2, ADC, GPIO, UART...);
- PWM channels are launched in center-aligned PWM mode;
- The ADC starts in the injected current measurement mode of phases A and B.
- The program enters the waiting state for launch (via UART command or button press).

2. Determining the rotor position (via Hall sensors):

- – TIM2 operates in Hall Sensor Mode;
- Hall sensors provide three logical signals (H1, H2, H3), which form the rotor position code (from 1 to 6);
- a change in the state of the inputs causes a TIM2 interrupt, in which:
- the current rotor position is read;
- the electrical angle (electrical position) of the rotor is updated;
- the direction of rotation is determined (using the transition table);
- starts (if necessary) updating the PWM switching table.

Note: in BLDC on Hall sensors the angle is known discretely (every 60°), so intermediate values of the rotor position angle must be calculated as $\theta = \frac{60 t_{CNT2}}{t_{n-1} - t_{n-2}}$.

3. Current measurement:

- usually 2 shunts are used on phases A and B. In phase C, the current is calculated using Kirchhoff's rule: IC =−IA + IB);
- currents are read via ADC1 in injected mode (trigger from TIM1 update).
- processing of results – in an interrupt or in the main loop, via DMA or polling.

4. Clarke → Park Transformation (FOC).
According to the currents I_A, I_B:

- the Clarke transformation is performed (transition from a three-phase to a two-phase stationary α-β system);
- the Park transformation is performed on the electrical angle θ (obtained from the Halls).

5. Regulation (PI regulators).
Two separate PI controllers:

- for I_d (maintained at zero for maximum torque);
- for I_q (regulates the engine torque);
- generate voltages V_d, V_q.

6. Inverse transformation Park → Clarke → PWM:

- voltages V_d, V_q → are converted back into the α-β system:
- the PWM value is calculated for each TIM1 channel;
- update PWM channels (CCR1, CCR2, CCR3 of timer TIM1).

7. Speed regulation (external loop):

- calculation of speed by the frequency of Hall signal transitions;
- regulation occurs through:
- PI speed controller → I_q_ref setpoint;
- Next, I_q_ref is transferred to the current loop.

8. Protect:

- current protection - through comparators;
- overvoltage – measurement of the DC bus Vbus.

The code structure templates for implementing a BLDC motor control system on the STM32G431RB microcontroller is presented below.

main.c – general structure:

```
#include "main.h"

// Global variables
uint8_t hall_state = 0; // Current state of Hall sensors
float theta_elec = 0.0f; // Electrical angle of the rotor
float I_a, I_b, I_c; // Phase currents
float I_d, I_q; // Currents in the dq system
float V_d, V_q; // Voltages in the dq system
float duty_a, duty_b, duty_c; // Calculated PWM for phases

// Main function
int main(void) {
  HAL_Init();
  SystemClock_Config();
```

```
  MX_GPIO_Init();
  MX_TIM1_Init(); // PWM generator
  MX_TIM2_Init(); // Hall Sensor mode
  MX_ADC1_Init(); // Phase currents
  MX_DMA_Init(); // If DMA is used
  MX_USART2_UART_Init(); // Communication

  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);
  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_3);
  HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_1); // Lower keys
  HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_2);
  HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_3);

  HAL_TIM_Encoder_Start_IT(&htim2, TIM_CHANNEL_ALL); // Hall sensors with interrupt

  HAL_ADC_Start(&hadc1); // Currents

  while (1) {
    // 1. Estimating the rotor position using Hall sensors
    theta_elec = hall_lookup_table[hall_state]; // tabular

    // 2. Reading currents I_a, I_b via ADC
    Read_Current(&I_a, &I_b);
    I_c = - (I_a + I_b); // Calculate the third phase

    // 3. Clark -> Park transformation
    abc_to_dq(I_a, I_b, theta_elec, &I_d, &I_q);

    // 4. PI current regulators
    V_d = PI_Controller_D(I_d, 0); // I_d_ref = 0
    V_q = PI_Controller_Q(I_q, Iq_ref); // The given moment

    // 5. Inverse transformation dq -> abc + SVPWM
    dq_to_abc(V_d, V_q, theta_elec, &duty_a, &duty_b, &duty_c);

    // 6. PWM update
    Set_PWM_Duty(duty_a, duty_b, duty_c);

    // 7. Additionally: display, diagnostics
  }
}
```

2.4.3. Hall sensor signal processing.

The Hall sensors in a BLDC motor output a six-step sequence (called a switching table) that allows for a rough determination of the rotor position.

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim) {
  if (htim->Instance == TIM2) {
    // Read 3 bits from GPIO inputs (e.g. PA0, PA1, PA2)
```

```
   hall_state = (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) << 2)
              | (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1) << 1)
              | (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_2) << 0);
  }
  float theta_elec = get_electrical_angle_from_hall(hall_state);
}
```

Each of the three Hall sensors (H1, H2, H3) can be 0 or 1. Together they form 6 unique combinations out of 8 possible:

001, 011, 010, 110, 100, 101 – this allows you to divide the electrical circuit into 6 sectors of 60°.

Calculating theta_elec

Since we only know the sector, the angle can be given as the midpoint of the sector. For example:

```
float get_electrical_angle_from_hall(uint8_thall_state) {
   float theta = 0;

   switch (hall_state) {
      case 0b001: theta = M_PI / 6.0f; break // 30°
      case 0b011: theta = M_PI / 2.0f; break // 90°
      case 0b010: theta = 5.0f * M_PI / 6.0f; break // 150°
      case 0b110: theta = 7.0f * M_PI / 6.0f; break // 210°
      case 0b100: theta = 3.0f * M_PI / 2.0f; break // 270°
      case 0b101: theta = 11.0f * M_PI / 6.0f; break;// 330°
      default: theta = 0; break; // error or 000/111
   }

   return theta;
}
```

Clarification of angular interpolation.

While the motor is rotating, we can interpolate between sectors based on the update rate (dt) and the rotation speed. For example:

```
theta_elec += omega_elec * dt;
if (theta_elec > 2 * M_PI) theta_elec -= 2 * M_PI;
```

This allows for a smooth change in the angle between Hall sensor transitions. This method is called pseudo-FOC.

read_Current() – reading phase currents from two ADC channels (Ia and Ib), the third Ic is calculated according to Kirchhoff's rule:

```
void Read_Current(float* Ia, float* Ib) {
   uint32_t adc_val_a = HAL_ADC_GetValue(&hadc1); // Channel for I_a
   uint32_t adc_val_b = HAL_ADC_GetValue(&hadc1); // Channel for I_b

   // Convert to current (assuming 12-bit ADC and 3.3V)
   float adc_voltage_a = (adc_val_a / 4095.0f) * 3.3f;
   float adc_voltage_b = (adc_val_b / 4095.0f) * 3.3f;

   // For example, shunt + amplifier = 0.1 ohm + 10x
   *Ia = (adc_voltage_a - 1.65f) / (0.1f * 10.0f); // Centering at 1.65V
   *Ib = (adc_voltage_b - 1.65f) / (0.1f * 10.0f);
```

```
}
```

abc_to_dq() – conversion of currents Ia and Ib into d–q coordinates by the known electrical angle theta_elec:

```
void abc_to_dq(float Ia, float Ib, float theta_elec, float* Id, float* Iq) {
    float alpha = Ia;
    float beta = (Ia + 2.0f * Ib) / sqrtf(3.0f);

    float sin_theta = sinf(theta_elec);
    float cos_theta = cosf(theta_elec);

    *Id = alpha * cos_theta + beta * sin_theta;
    *Iq = -alpha * sin_theta + beta * cos_theta;
}
```

dq_to_abc() – inverse transformation from d–q coordinates to phase values (alpha–beta):

```
void dq_to_abc(float Vd, float Vq, float theta_elec, float* Va, float* Vb, float* Vc) {
    float sin_theta = sinf(theta_elec);
    float cos_theta = cosf(theta_elec);

    float alpha = Vd * cos_theta - Vq * sin_theta;
    float beta = Vd * sin_theta + Vq * cos_theta;

    *Va = alpha;
    *Vb = -0.5f * alpha + sqrtf(3.0f) / 2.0f * beta;
    *Vc = -0.5f * alpha - sqrtf(3.0f) / 2.0f * beta;
}
```

PI_Controller_D() and PI_Controller_Q() - simple PI controller without restrictions:

```
float PI_Controller_D(float Id_meas, float Id_ref) {
    static float integral = 0;
    float error = Id_ref - Id_meas;
    integral += error * 0.001f; // Ts = 1 ms

    float Kp = 0.1f, Ki = 5.0f;
    return Kp * error + Ki * integral;
}

float PI_Controller_Q(float Iq_meas, float Iq_ref) {
    static float integral = 0;
    float error = Iq_ref - Iq_meas;
    integral += error * 0.001f;

    float Kp = 0.1f, Ki = 5.0f;
    return Kp * error + Ki * integral;
}
```

set_PWM_Duty()– bringing values to the TIM1 timer registers:

```
void Set_PWM_Duty(float Va, float Vb, float Vc) {
    // Suppose supply voltage = 24V
```

```
    float Vdc = 24.0f;

    float dutyA = (Va / Vdc + 0.5f) * TIM1->ARR;
    float dutyB = (Vb / Vdc + 0.5f) * TIM1->ARR;
    float dutyC = (Vc / Vdc + 0.5f) * TIM1->ARR;

    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, dutyA);
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2, dutyB);
    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_3, dutyC);
}
```

2.5. Summary

In this chapter, the principles of operation, construction, and control strategies for BLDC motors have been comprehensively discussed. BLDC machines represent a highly efficient and versatile solution for modern electrical drives due to their combination of compact design, high torque-to-weight ratio, and low maintenance requirements.

The review covered both the theoretical background and practical implementation aspects, highlighting the importance of precise current and voltage control in order to achieve smooth torque production. A particular focus was given to Field-Oriented Control (FOC), which has proven to be the most advanced and widely adopted method for high-performance BLDC drives. FOC allows the decoupling of flux and torque components, enabling independent control and maximizing efficiency across different speed ranges.

The analysis also demonstrated the role of position feedback devices such as Hall sensors and encoders, as well as the possibility of sense less techniques. Each method has specific advantages and limitations: Hall sensors are simple and inexpensive, encoders provide high precision, while sensorless methods are attractive for cost-sensitive applications but require sophisticated algorithms.

It was emphasized that advanced microcontrollers such as STM32 offer integrated peripherals for motor control, including high-resolution PWM timers, ADC units, and dedicated motor-control libraries, which significantly simplify the development process. By utilizing STM32CubeMX and Motor Control SDK, engineers can rapidly prototype and implement BLDC control strategies.

Overall, the study confirms that BLDC motor control combines theoretical complexity with practical applicability, making it a central topic in modern electromechanical systems. The adoption of FOC and other vector-based methods ensures high efficiency, robustness, and dynamic response, which are crucial for automotive, industrial, and consumer applications. Future developments will focus on sensorless operation, AI-based optimization of control parameters, and integration of BLDC drives into smart energy systems.

In summary, BLDC motor control is not only a key technological area today but also a rapidly evolving field, bridging classical electrical engineering with modern embedded systems and digital control theory.

References

1. Krishnan, R. Permanent Magnet Synchronous and Brushless DC Motor Drives. – Boca Raton: CRC Press, 2010. – 496 p.
2. Ede, A. N. Brushless Permanent Magnet Motor Design. – 2nd ed. – Magna Physics Publishing, 2005. – 250 p.
3. Pillay, P. Modeling, Simulation and Control of Permanent Magnet Synchronous and Brushless DC Motors. – Concordia University, Canada, 2003.
4. Mohan, N. Electric Machines and Drives: A First Course. – Wiley, 2012. – 288 p.

5. What is a BLDC Motor? – ABLIC, https://www.ablic.com/en/semicon/applications/bldc-motor/what-bldc-motor/
6. Challenges in Speed Control of BLDC Motors – ResearchGate, https://www.researchgate.net/publication/391233364_Challenges_in_Speed_Control_of_BLDC_Motors
7. Jahns, T. M., Kliman, G. B., Neumann, T. W. Interior Permanent Magnet Synchronous Motors (IPMSM) and BLDC Drives // IEEE Transactions on Industrial Applications, 2001.
8. AN047 – Brushless DC Motor Fundamentals / Prepared by Jian Zhao, Yangwei Yu // Monolithic Power Systems, 2011. https://www.monolithicpower.com/media/document/Brushless_DC_Motor_Fundamentals.pdf?srsltid=AfmBOooxClByPiMUctXy0PxWmBJY0II1znOHqi24DcymreWEoDZIMiGA&utm_source=chatgpt.com
9. Charles Frick. Brushless DC Motors Introduction for Next-Generation Missile Actuation Systems Outline // ANALOG DEVICES, Oct 1 2018. https://www.analog.com/en/resources/technical-articles/brushless-dc-motors-introduction-for-next-generation-missile-actuation-systems-outline.html
10. STM32-based brushless DC motor drive design – Atlantis Press, https://www.atlantis-press.com/article/17489.pdf
11. STM32 Ecosystem for Motor Control – STMicroelectronics, https://www.st.com/content/st_com/en/ecosystems/stm32-motor-control-ecosystem.html
12. STM32 3 Phase PWM (Center-Aligned) Example Code, https://deepbluembedded.com/stm32-3-phase-pwm-center-aligned-example-code/
13. STM32 for Motor Control Applications, https://d1.amobbs.com/bbs_upload782111/files_10/ourdev_265522.pdf.
14. STM32 BLDC Control with HALL Sensor – Mikrocontroller.net, https://www.mikrocontroller.net/articles/STM32_BLDC_Control_with_HALL_Sensor.

Chapter 3. In-vehicle digital control systems

3.1. In-Vehicle Network Technologies

A modern automobile is a complex, multifunctional system incorporating numerous heterogeneous electronic components. The design of its subsystems enables the regulation of a wide range of input and output parameters, making the vehicle a multi-parameter control object. The acquisition and processing of operational data require electronic control units (ECUs) and a variety of sensors. Control actions, generated in response to ECU signals, are implemented through dedicated actuators.

Contemporary vehicles integrate ECUs across drivetrain and chassis systems, passenger compartments, and for radio, communication, and navigation functions. Data exchange between these systems and their respective ECUs is facilitated by digital data buses. These buses reduce the total wiring length, minimise the number of required sensors, and increase both the speed and reliability of information transfer.

Growing demands for road safety, driving comfort, reduced exhaust gas emissions, and fuel efficiency necessitate increasingly intensive information exchange between ECUs.

Historically, electrical components in automotive systems were linked through direct analogue connections, in which a voltage signal was transmitted directly to and received by the ECU, making the sensor and receiver part of a single electronic circuit. To reduce susceptibility to interference and improve signal processing, pulse-width modulation (PWM) was introduced.

PWM transmits parameters as signal pulses of defined duration, repeated at fixed intervals. This approach mitigates distortions caused by resistance and noise, with the signal value encoded in the switching instant within each interval. The connected conductor need only toggle between energised and de-energised states, with longer “on” times corresponding to higher transmitted values.

Current technology employs digital, binary signal transmission. An analogue-to-digital converter encodes the measured value into a binary code, whose accuracy depends on the number of bits used. The conductor now switches at higher frequencies to convey the digital code, which can be easily stored and transmitted to other ECUs without direct wiring between them.

Instead, a principle long used in electrical distribution is applied: a common conductive bus supplies power to multiple components, to which devices can be connected as needed. The same concept is extended to digital information transfer, where the “bus” – derived from “bus-bar” – denotes a wiring system with integrated control elements for data exchange between electronic modules.

Until recently, each data signal in control systems was transmitted via a dedicated conductor. Consequently, the number of wires and ECU connector contacts increased with every additional data channel, making such an approach viable only for limited information exchange.

To enable the transfer of large data volumes while maintaining a system that was both visually traceable and space-efficient, an optimised technical solution was required. Modern vehicles therefore employ highly complex network architectures that facilitate data exchange between multiple ECUs.

The term vehicle network architecture now encompasses not only the wiring and bus systems but also the ECUs themselves, the communication protocols, and the associated software.

The achievable data rate is inherently limited. An ideal rectangular waveform is possible only in theory; in practice, effects such as inertia, self-inductance, and electromagnetic radiation produce a trapezoidal signal shape. Transmission distance is a key factor: as conductor length increases, its resistance causes progressive signal attenuation.

Electromagnetic radiation further degrades the signal. Maintaining quality would require higher transmission voltages, which in turn increase power consumption and emissions, and reduce speed due to signal inertia. In digital transmission, the transceiver determines logical “0” or “1” by comparing the received signal to a defined voltage threshold.

A single-wire bus, though inexpensive and simple, offers limited data rates and is more susceptible to interference, as signals are referenced to the vehicle ground. This requires higher transmission power. In contrast, a two-wire bus transmits as the voltage difference between conductors, providing substantial noise suppression, enabling lower voltages and higher data rates, albeit at greater cost and complexity.

Optical fibres are immune to electromagnetic effects, enabling very high transmission speeds. However, light can only enter or exit at the fibre ends, requiring a light source at the transmitter and a photosensitive element at the receiver. As this transmission is unidirectional, bidirectional communication requires two fibres and, in each transceiver, both a light source and a photodetector.

Radio communication eliminates the need for physical conductors between transceivers, but suffers from limited reliability, as devices may interfere with one another, and offers no higher data rates than wired bus systems [1].

Bus systems are categorised primarily by achievable data rate, with transmission reliability also playing a significant role. These class designations should not be confused with those of CAN buses – initially related, but now applied independently. The classification of data bus systems is presented in Table 3.1.

Table 3.1 Classes of Data Transmission Buses

Data Transmission Speed	*Diagnostic Buses*	*Class A*	*Class B*	*Class C*	*Class C+*	*Multimedia Class*
Transmission rate	< 10 kbit/s	< 25 kbit/s	25–125 kbit/s	125 kbit/s – 1 Mbit/s	> 1 Mbit/s	> 10 Mbit/s
Application area	Connection of diagnostic equipment	Comfort systems (seat adjustment, power windows)	Air conditioning systems	Powertrain and chassis systems	Braking and steering systems	Central display, audio/video navigation systems, telephony, etc.

In details:

- In the diagnostic class, very low data rates are acceptable, and transmission reliability is of minor importance. Errors are simply corrected through retransmission;
- class A allows slightly higher speeds but is used only for non-critical functions such as window lift control or basic switch signals;
- class B bus systems achieve moderate data rates and are employed primarily in automotive comfort and convenience systems;
- class C offers both high speed and high reliability, making it suitable for transmission and chassis electronics;
- for safety-critical functions, Class C+ buses are used. In braking and steering systems, data loss is unacceptable and messages must be transmitted with minimal delay;
- multimedia buses occupy the highest tier in terms of data rate, as required by the large volumes of audio and video information; here, reliability is not a critical factor.

Electronic control units can be interconnected using different topologies. The simplest is a peer-to-peer link between two devices via a single conductor. More commonly, several devices share a bus, which must be terminated at both ends with resistors to prevent signal reflections.

In a star topology, a central node connects individually to all other devices. If this node is merely a wiring junction – a potential distributor – the central connection remains short. If the node

is an ECU with interfaces for peripheral devices, the arrangement becomes a set of multiple peer-to-peer links.

A ring topology is typical when optical fibres are used. Each device forwards the received optical pulses to the next, allowing all devices to exchange data in a closed loop of unidirectional links.

In ECU communication, the raw data alone is insufficient. Additional information is required – such as the data's meaning, its source sensor, priority level, and whether it should be forwarded to other ECUs.

To this end, the ECU appends identifiers and priority codes to the message, while the transceiver adds technical information for reliable delivery. Examples include synchronisation bits, enabling the receiver to align its clock with the transmitter, and message length indicators, informing the receiver when the transmission ends. Error-checking data allows verification of message integrity.

All such elements form a continuous bit stream in which the receiver must locate and extract the essential data. This is achieved through standardised message structures known as protocols. Each protocol defines the format and sequence of data fields. Transceivers operating with different protocols are mutually incompatible.

The type of data transmission is an important criterion in distinguishing bus systems.

Synchronous transmission can be compared to a telephone conversation: while one unit transmits, the other listens and processes the information immediately. Its drawback is that the process cannot be interrupted; during transmission, the bus is occupied by the ECU and all receivers remain blocked until the message is fully sent. Often, the transmitter waits for an acknowledgement and will retransmit the entire message if confirmation is absent or an error is reported.

Asynchronous transmission is more akin to correspondence: short messages are sent at defined intervals, first collected by the ECU and then processed. Between messages, the bus can be used by other ECUs, distributing the load evenly and enabling the rapid, complete transfer of large volumes of data.

To maintain synchronisation, control signals are periodically inserted into the data stream, although their proportion is small compared to information data. These control elements are added even to short messages, which increases the total transmitted volume.

Multiple ECUs must not transmit to the bus simultaneously; such collisions result in message loss. Bus access must therefore be regulated. In master–slave configurations, one ECU acts as the master, initiating all communications. At regular intervals, it polls the slave ECUs, which then have a brief time window to respond. All slave devices must be registered with the master, requiring an initialisation phase in modern bus systems.

When access control is managed at the protocol or message level, any ECU may transmit when the bus is idle. If two devices start simultaneously, an arbitration process determines which continues while the other delays transmission. Arbitration often relies on a binary priority field at the start of the message. Without regulation, low-priority messages may be delayed excessively, so managed arbitration is used to ensure timely delivery.

Most vehicles employ multiple bus systems, selected according to various criteria. The physical medium is significant: fibre-optic cables are lightweight and support high data rates but are fragile and unsuitable for sharp bends. The maximum error-free distance is another key factor; for example, in heavy vehicles, several metres of cable may be needed between front and rear ECUs, and some bus systems are only suitable for short runs.

In certain cases, a direct link between two ECUs is sufficient. The number of devices a bus can support is also important. For audio, video, or safety systems, high data rates are essential, whereas seat adjustment controls require only low speeds. In some applications, reliability outweighs speed – for instance, in braking systems where transmission errors could have severe consequences. In others, occasional delays or noise are acceptable.

Integration into the existing vehicle architecture and service framework is also a consideration, encompassing protocol compatibility, access control methods, and the ratio of control to user data [1].

3.2. Overview of the K-line Bus System

The K-Line–also referred to as the K-Bus–represents the earliest generation of in-vehicle serial communication systems in European automotive engineering. Initially developed to provide a standardised interface for vehicle diagnostics, the K-Line was formally codified under ISO 9141 in 1989. Its enduring relevance lies in its simplicity of design, low implementation cost, and straightforward compatibility with external computing equipment, making it an important foundation for subsequent, more complex vehicular communication protocols.

From a physical standpoint, the K-Line is implemented as a single-conductor communication link, with signal voltage referenced to the vehicle's electrical ground. Unlike differential systems such as CAN or FlexRay, it does not employ a twisted pair or dedicated shielding; instead, it relies on the vehicle chassis as the common return path. The standard imposes no explicit limitation on cable length, which permits considerable flexibility in implementation, particularly for diagnostic harnesses and test equipment.

Functionally, the K-Line is intended primarily for bidirectional communication between two nodes–typically an external diagnostic tester and an electronic control unit (ECU). In contemporary heavy-duty vehicle applications, however, modified implementations allow a single K-Line bus to sequentially address multiple ECUs via selective addressing protocols.

The system operates at relatively low data rates, generally between 1.2 and 10.4 kbit/s, which are sufficient for the transmission of diagnostic parameters but inadequate for high-speed control or real-time applications. Given that it is dedicated to diagnostics rather than mission-critical control, the protocol tolerates longer transmission times and does not require the high reliability standards demanded by real-time control networks. Under certain operational conditions–such as full data frame transfer at the lowest baud rate–the reception of a complete diagnostic message can take up to five seconds.

Access control in the K-Line system follows an asynchronous, master–slave architecture. The external diagnostic tester functions as the master, initiating communication and controlling bus access, while the ECU or other onboard devices act as slaves, responding to queries or transmitting requested data. Error detection is handled at the protocol level, but without redundancy measures such as those found in modern multi-master networks.

Although the K-Line has been largely superseded by faster, more robust communication standards–such as Controller Area Network (CAN), Local Interconnect Network (LIN), and FlexRay–it remains in service for legacy vehicle platforms and as a low-cost diagnostic interface in certain modern systems. Its longevity illustrates the effectiveness of a design philosophy prioritising simplicity, ease of implementation, and compatibility across multiple generations of automotive electronics [2].

3.3. CAN Technology

The Controller Area Network (CAN) was developed by Bosch in the late 1980s to enable robust transmission of measurement, control, and monitoring data. First deployed in 1991 as a Class C vehicle data bus, it is defined by ISO 11898.

CAN employs a twisted two-wire configuration, with the signal level represented by the voltage difference between the conductors. This design reduces susceptibility to interference and minimises emissions. At low transmission speeds, the bus can continue operating on a single conductor in the event of a break.

The permissible cable length is determined primarily by propagation delay and data rate: at 1 Mbit/s, a CAN bus can be up to 40 m, while at 10 kbit/s it can exceed 1 km in theory. Because access control occurs at the message level, the number of ECUs is not inherently limited, though in practice it is constrained by transceiver design.

Data rates are not fixed by the standard. In Mercedes-Benz vehicle electronics, low-speed CAN operates at up to 125 kbit/s, while high-speed CAN exceeds this rate. Multiple mechanisms for error prevention, detection, and correction are implemented; ECUs can report faulty messages or disable themselves if persistent transmission errors occur. As a result, CAN is regarded as one of the most reliable bus technologies.

The maturity and long-standing standardisation of its parameters keep production and operating costs low, making CAN increasingly used for diagnostics as well. CAN functions synchronously, with access managed through message-level arbitration. Each possible message has a unique identifier, with the first bit determining priority. If two devices transmit simultaneously, the higher-priority message continues while the other is deferred. Identifiers are defined with 29 reserved bits, resulting in a higher proportion of control data in short messages relative to payload.
A detailed examination of the CAN data bus architecture follows [3].

3.3.1. CAN Data Bus Network Configuration

From the 1980s onwards, rapid advances in digital electronics led to increasing integration of electronic control units (ECUs) in vehicles [4]. To enable inter-ECU communication, Bosch developed the CAN data bus (Controller Area Network), a dedicated serial bus allowing data exchange between networked control modules (Fig. 3.1).

Vehicle powertrain components form an integrated system comprising: ECU 1 – engine control, ECU 2 – automatic transmission control, ECU 3 – chassis systems control. Comfort-related components also form a network, including the central ECU 4 and door ECUs 5.

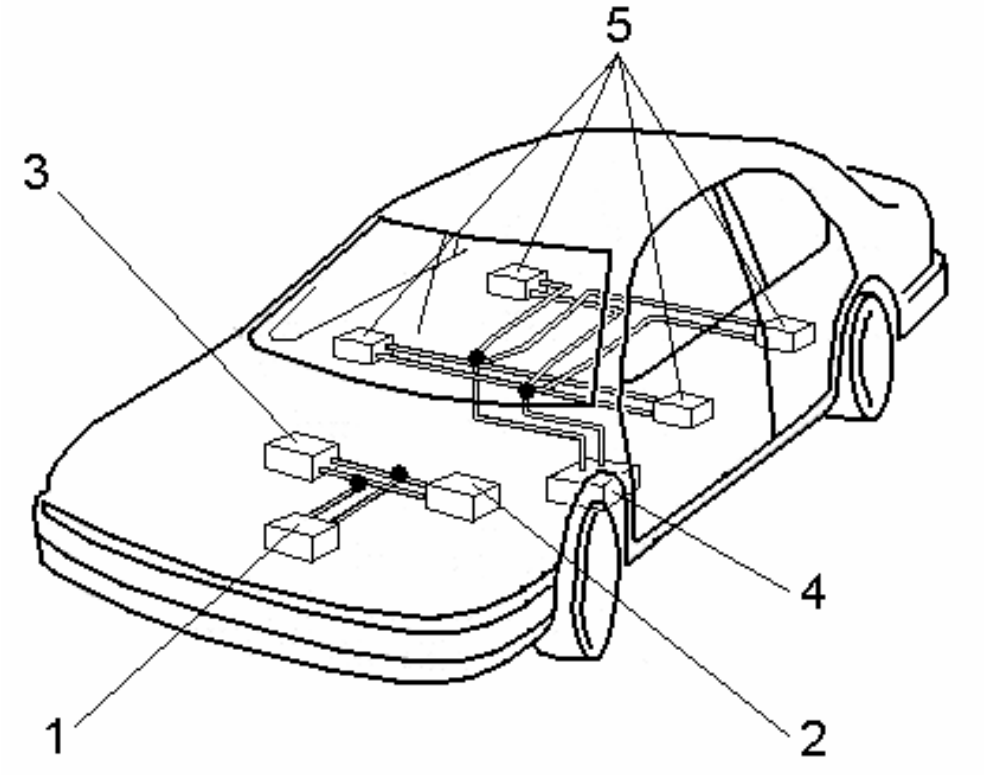
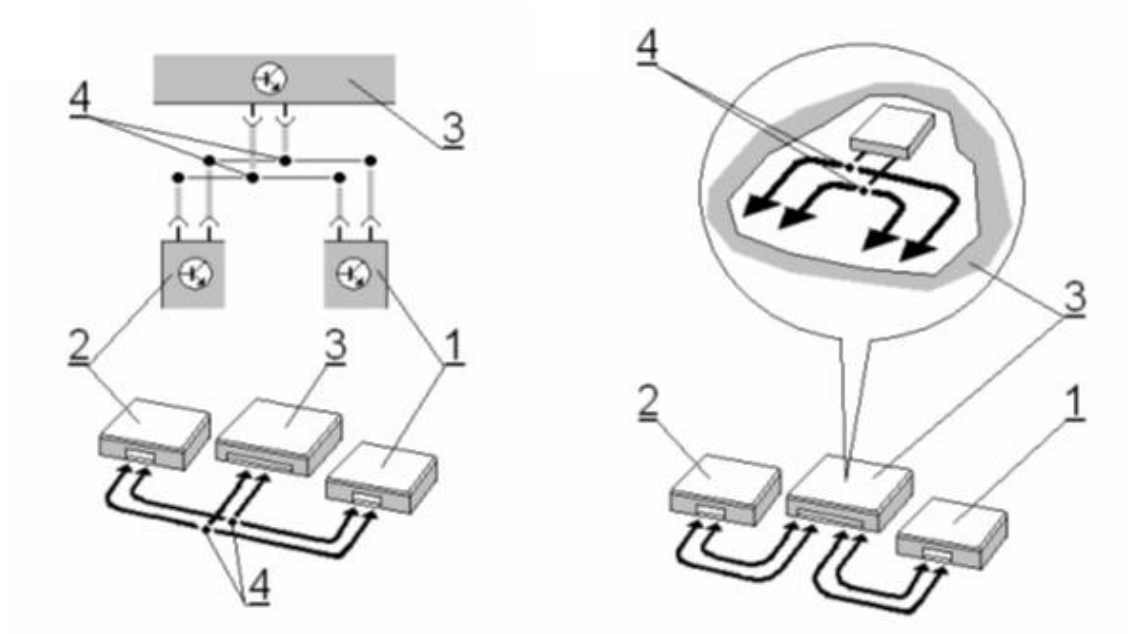


Fig. 3.1. Communication diagram of control units via the CAN bus.

CAN network topologies vary (Fig. 3.2). Typically, the junction node is located outside the ECU within the wiring harness (Fig. 3.2 a), though in some cases it may be integrated into the engine ECU (Fig. 3.2 b).



a) b)

Fig. 3.2. Variants of connecting control units into a single network:

(a) the network node is located outside the control unit;(b) the network node is located in the engine control unit; 1 – brake system control unit; 2 – automatic transmission control unit; 3 – engine control unit; 4 – network node

Information is transmitted over two bidirectional conductors, regardless of the number of ECUs or volume of data. This process is analogous to a telephone conference: one ECU broadcasts data, all others receive it, and each determines its relevance. The greater the system-wide information available to an ECU, the more effectively it can coordinate specific functions [4].

3.3.2. Network Architecture Principle

A CAN network interconnects multiple ECUs via transceivers (Fig. 3.3), ensuring identical operating conditions for all nodes. This “multi-drop” architecture grants all ECUs equal status, with no inherent priority. Information is exchanged as sequential signals.

Although CAN can operate over a single conductor, a second is usually provided, carrying inverted signals. This differential signalling significantly improves immunity to external electromagnetic interference. For clarity in subsequent explanations, a single-conductor configuration is used as the reference model [4].

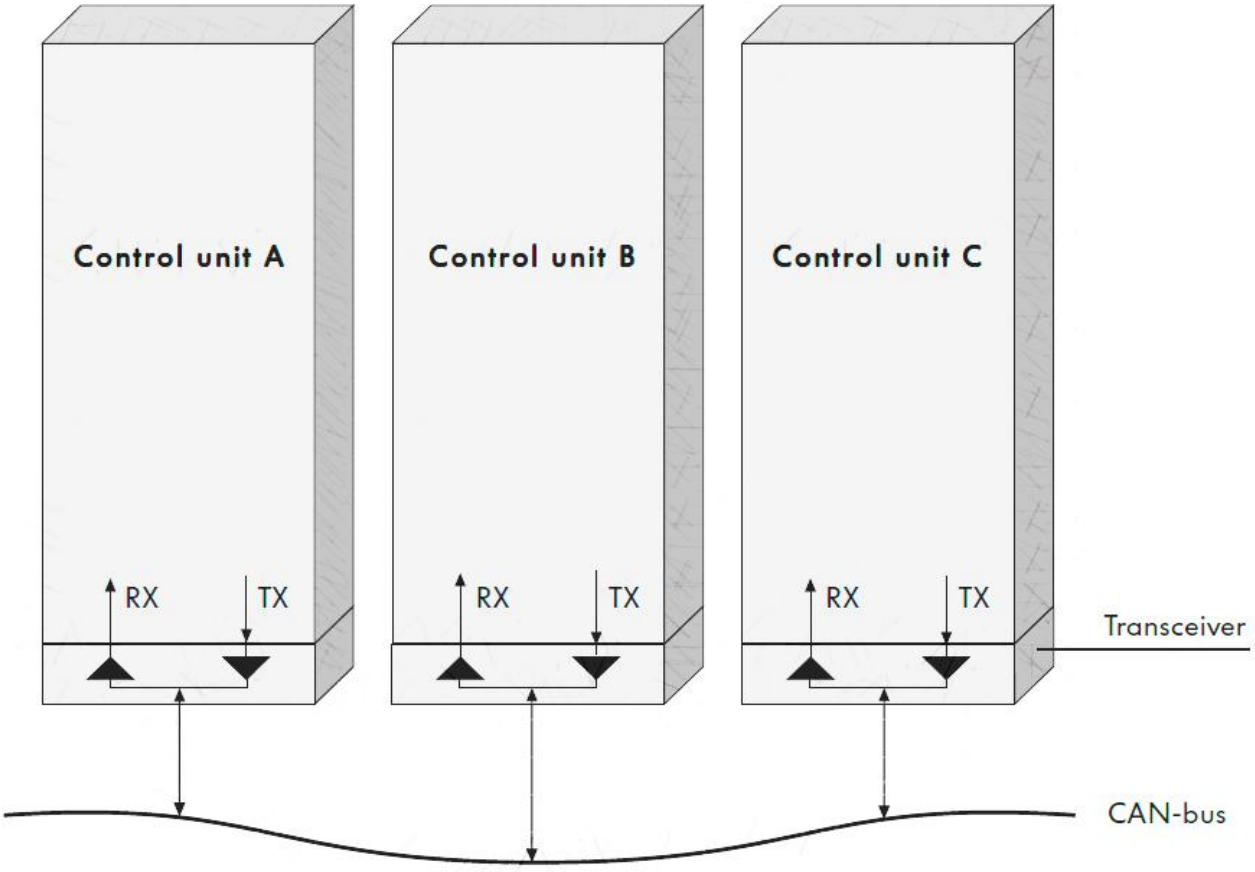


Fig. 3.3. Principle of network formation [4].

3.3.3. Information Exchange Process

The data exchanged within a CAN network comprises discrete messages, each of which can be transmitted or received by any control unit. Each message conveys information on a specific physical parameter, such as crankshaft rotational speed, encoded in binary form as a sequence of zeros and ones (bits). For example, an engine speed of 1800 rpm may be represented as the binary value 00010101 (Fig. 3.4).

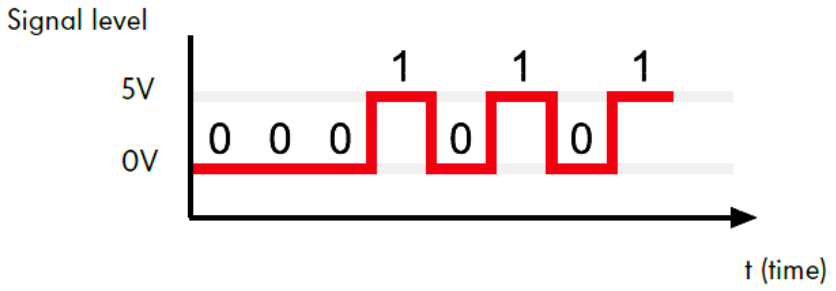


Fig. 3.4. Time-based transmission of electrical signals [4].

During transmission, each binary value is converted into a serial stream of pulses (bits) via the TX (transmit) line to the transceiver, which transforms these current pulses into corresponding voltage signals for sequential transmission along the bus (Fig. 3.5).

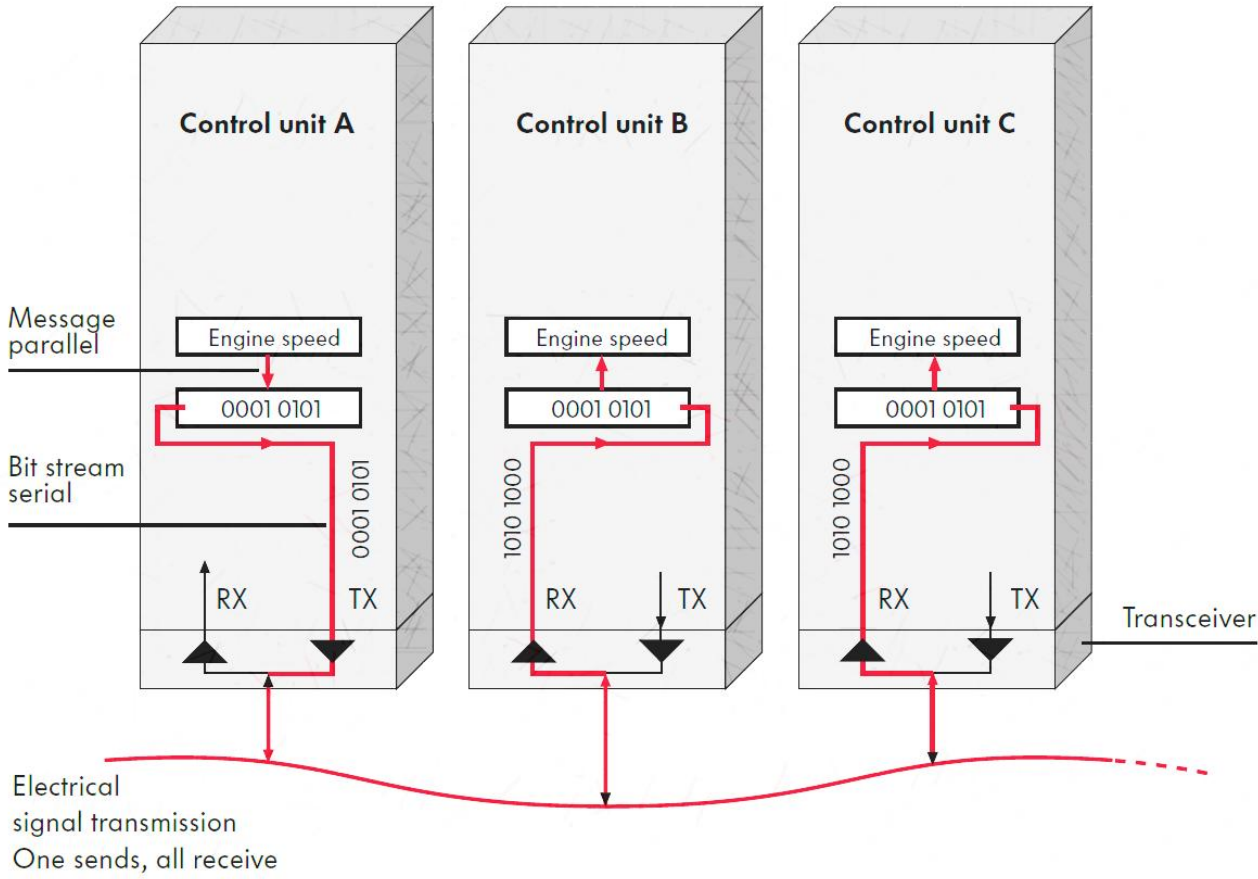


Fig. 3.5 Information transfer over the CAN bus (broadcast principle) [4].

During reception, the transceiver converts voltage pulses from the bus into bit sequences and forwards them via the RX (receive) line to the control unit, where the binary data is decoded into the original parameter value. Thus, the binary code 00010101 is interpreted as 1800 rpm.

Any transmitted message may be received by all control units. This "broadcast" transmission principle is analogous to a radio station whose signals are accessible to all receivers in range. It ensures that all units connected to the network have access to identical, up-to-date information at any given moment [4].

3.3.4. Components of the Data Bus

1. K-Cable.

The K-cable serves to connect the diagnostic tool to the vehicle's system during servicing operations.

2. Control Unit.

A control unit (CU) receives sensor signals, processes them, and transmits the corresponding control signals to actuators. Its key components include a microcontroller with input and output memory units, and a memory device for storing the software (Fig. 3.6).

Signals from sensors – such as temperature sensors or crankshaft speed sensors–are periodically read and sequentially recorded into the input memory unit. Within the microcontroller, these inputs are processed in accordance with the embedded software routines. The resulting output signals are written into the output memory unit, from which they are transmitted to the respective actuators.

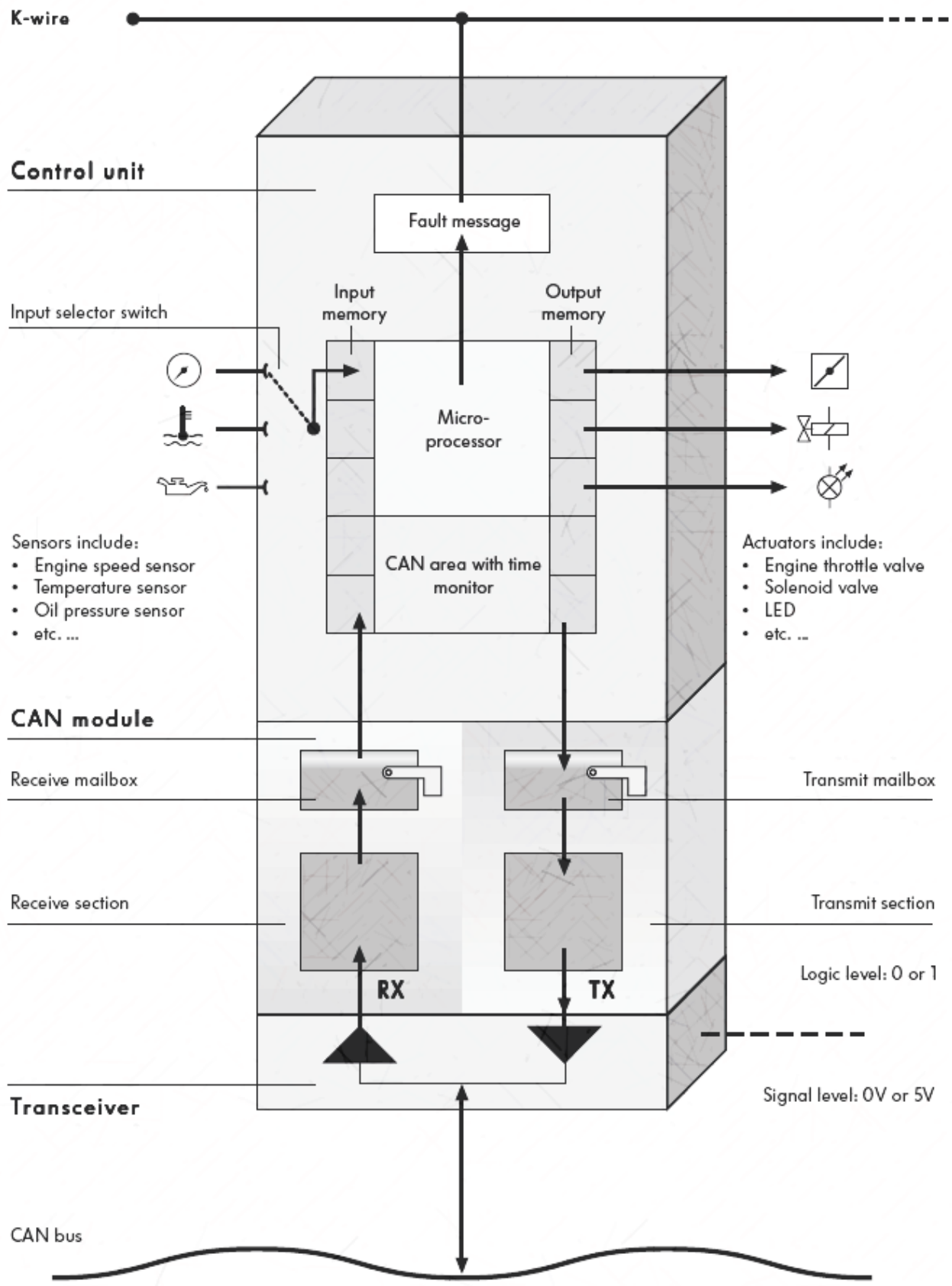


Fig. 1.6 Functional components: control unit, CAN system module and transceiver [4]

For processing messages received from, or destined for, the CAN bus, each CU is equipped with an additional memory device for storing both incoming and outgoing messages.

3. CAN System Module.

The CAN system module facilitates data exchange via the CAN bus and is divided into two functional areas: a receive section and a transmit section. Communication between the CAN module and the CU takes place via dedicated “mailboxes” for incoming and outgoing messages. This module is typically integrated into the CU’s microcontroller chip [4].

4. Transceiver.

The transceiver is a bidirectional communication device that also performs signal amplification. It converts the binary signals (at the logical level) received from the CAN system module into electrical voltage pulses, and vice versa. This allows data to be transmitted via copper conductors in the form of electrical impulses.

The transceiver connects to the CAN system module via the TX (transmit) and RX (receive) lines. The RX line is linked to the CAN bus through an amplifier, enabling continuous monitoring (“listening”) of signals transmitted along the bus.

A distinctive feature of the TX line connection to the CAN bus is its implementation via an open-collector cascade (Fig. 3.7). This arrangement enables two possible bus states:

State 1 (Passive): The transistor is switched off (open switch). The bus line is connected to the current source through a high-resistance path, yielding a bus voltage level of logic “1”.

State 0 (Active): The transistor is switched on (closed switch). The bus line is connected to ground through a low-resistance path, producing a bus voltage level of logic “0” [4].

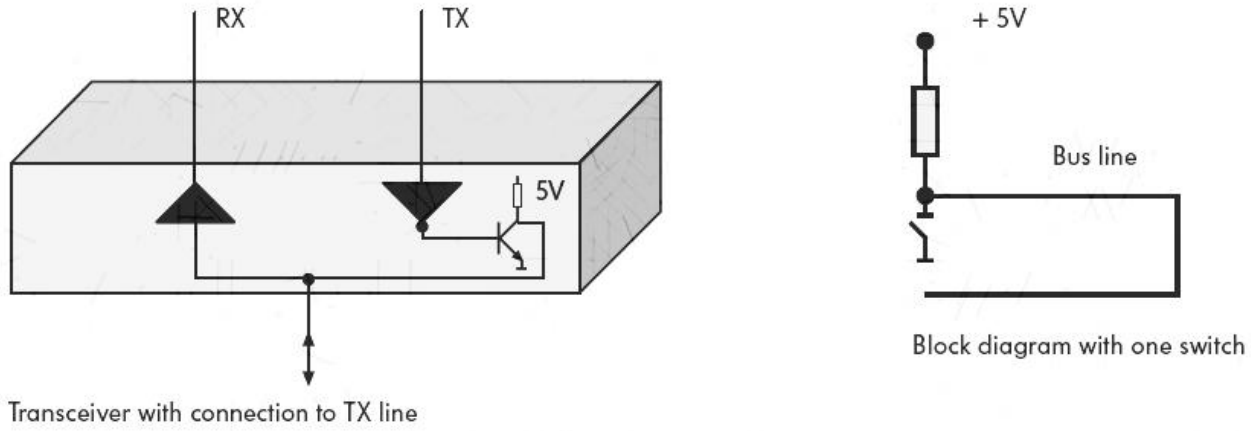


Fig. 3.7. Characteristics of the transceiver [4].

In the example of three transceivers connected to the bus line (schematic in Fig. 3.8), transceiver C is in the active state.

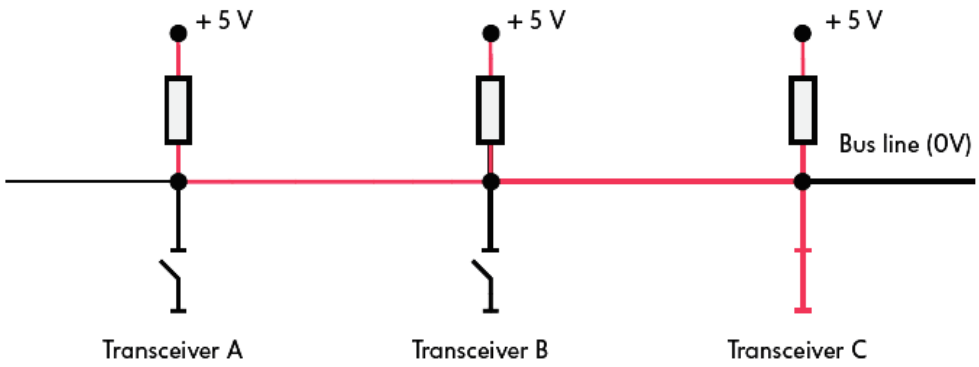


Fig. 3.8. Connection of three transceivers to the bus line.

When the switch is open, the bus is at logic “1” (passive), and when the switch is closed, the bus is at logic “0” (active). The possible bus states for this configuration are summarised in Table 3.2.

Table 3.2 Variants of the bus states [4]

Transceiver A	Transceiver B	Transceiver C	Bus line
1	1	1	1 (5B)
1	1	0	0 (0B)
1	0	1	0 (0B)
1	0	0	0 (0B)
0	1	1	0 (0B)
0	1	0	0 (0B)
0	0	1	0 (0B)
0	0	0	0 (0B)

Bus properties:

- if any switch is closed, current flows through the resistors, and the bus voltage is 0 V;
- if all switches are open, no current flows, no voltage drop occurs across the resistors, and the bus voltage is 5 V.

Consequently, if the bus is in the logic “1” (passive) state, any connected station can force it into the logic “0” (active) state. The passive state is referred to as recessive, and the active state as dominant.

These relationships are significant in the following cases:

a) transmission of error frames during data transfer;

b) arbitration, i.e., resolving simultaneous transmissions from multiple stations.

3.3.5. Data Protocol

The CAN bus transmits a data protocol, comprising an ordered sequence of bits, between control units (CUs). The protocol (Fig. 3.9) consists of seven fields, with its total length determined by the size of the data field (Field 5). The structure is identical on both CAN-High and CAN-Low lines.

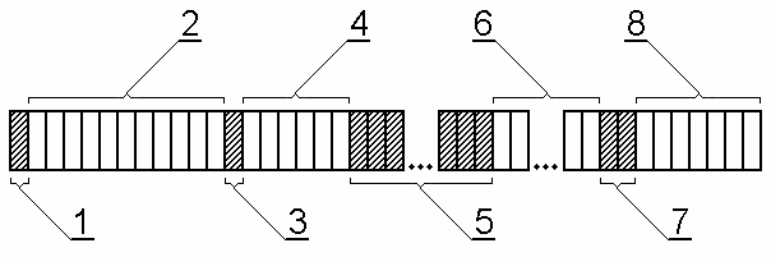


Fig. 3.9. Structure of the data protocol [5]:

1 – start field (1 bit); 2 – identifier/status field (11 bits); 3 – one unused bit; 4 – control field (6 bits); 5 – data field (maximum 64 bits); 6 – protection field (16 bits); 7 – acknowledgement field (2 bits); 8 – end field (7 bits)

The purpose of each data frame field is as follows:

- Start Field (1) – marks the beginning of the data frame. A single bit is transmitted: on the CAN-High line, the signal is approximately 5 V; on the CAN-Low line, it is 0 V (as defined by the system).
- Arbitration Field (2) – determines the priority of the data frame. If two control units (ECUs) attempt transmission simultaneously, the frame with the higher priority prevails.
- Control Field (4) – specifies the number of data messages contained within the data field, enabling each receiver to verify complete message reception.
- Data Field (5) – contains the transmitted messages intended for other ECUs. As shown in Table 3.3, each additional bit doubles the amount of transferable information; for example, in measuring engine coolant temperature, "0" corresponds to 0 V and "1" to 5 V.
- CRC Field (6) – used for error detection during transmission.
- Acknowledgement Field (7) – receivers confirm correct frame reception to the transmitter. If an error is detected, receivers immediately notify the transmitter, which then retransmits the frame.
- End Field (8) – marks the conclusion of the data frame and provides a final opportunity for error detection; if an error is found, retransmission is initiated.

Table 3.3 Variants of Transmitted Information

1-bit variants	Possible information	2-bit variants	Possible information	3-bit variants	Possible information
0 V	10 °C	0 V, 0 V	10 °C	0 V, 0 V, 0 V	10 °C
5 V	20 °C	0 V, 5 V	20 °C	0 V, 0 V, 5 V	20 °C
–	–	5 V, 0 V	30 °C	0 V, 5 V, 0 V	30 °C
–	–	5 V, 5 V	40 °C	0 V, 5 V, 5 V	40 °C
–	–	–	–	5 V, 0 V, 0 V	50 °C
–	–	–	–	5 V, 0 V, 5 V	60 °C
–	–	–	–	5 V, 5 V, 0 V	70 °C
–	–	–	–	5 V, 5 V, 5 V	80 °C

Table 3.4 illustrates encoding of throttle position using 8 bits, allowing for 256 unique bit sequences. This provides positional data in the range 0 °– 102 ° with a resolution of 0.4 °.

Table 3.4 Variants of Transmitted Information

Bit Sequence	Throttle Valve Position
0000 0000	Throttle valve opening angle – 000.0 °
0000 0001	Throttle valve opening angle – 000.4 °
0000 0010	Throttle valve opening angle – 000.8 °
…	…
0101 0100	Throttle valve opening angle – 033.6 °
…	…
1111 1111	Throttle valve opening angle – 102.0 °

When multiple ECUs attempt to transmit simultaneously, arbitration ensures that the highest-priority frame is transmitted first. For instance, a braking system ECU transmitting safety-critical data is prioritised over a transmission ECU sending comfort-related information.

Priority is determined in the Arbitration Field, which contains an 11-bit identifier encoding the frame's priority. Each bit has an assigned "significance" level, either dominant or recessive. All competing ECUs begin transmission simultaneously, with bit-by-bit comparison occurring on the CAN bus. If an ECU sends a recessive "1" but detects a dominant "0" from another ECU, it ceases transmission and switches to receive mode. For example, braking system frames, beginning with "00", override engine control frames ("01"), which in turn take precedence over transmission control frames ("10") [5].

3.4. LIN Technology

The Local Interconnect Network (LIN) emerged in the late 1990s as the result of a joint development initiative among several leading automotive manufacturers seeking a cost-efficient alternative to low-speed Controller Area Network (CAN) systems.

Its conception was driven by the recognition that many vehicular subsystems do not require the high bandwidth, advanced arbitration, or fault-tolerance capabilities offered by CAN, particularly in applications where functional safety is not critical. LIN was therefore designed to deliver adequate communication performance for such peripheral subsystems while significantly reducing both hardware complexity and production costs [6].

A notable feature of LIN technology is its integration of basic error detection and correction mechanisms. These provide sufficient data integrity for non-safety-critical applications while avoiding the processing overhead associated with more sophisticated protocols. In its operational philosophy, LIN bears certain similarities to the older K-line protocol, particularly in its use of a simple signalling structure and its focus on minimising implementation cost. However, unlike the point-to-point K-line, LIN supports true multi-node networking.

Communication between LIN sub-networks within a vehicle is typically coordinated through gateway functionality embedded in a CAN-connected control unit. This gateway translates LIN messages into CAN frames and vice versa, enabling seamless integration into the vehicle's broader communication architecture (Fig. 3.10).

From a physical layer perspective, LIN utilises a single-wire configuration for data transmission, a choice that substantially reduces wiring complexity and cost. The conductor, generally insulated in violet with additional colour-coded markings for identification, typically has a cross-sectional area of 0.35 mm². Given the relatively low data rates and robust signalling tolerances of the protocol, electromagnetic shielding is not required. The single-wire approach also simplifies harness design and allows easier routing within the vehicle body, making LIN especially attractive for distributed control applications such as window lifters, seat adjustments, climate control flap actuators, and lighting subsystems.

The network topology follows a single-master, multiple-slave architecture, in which a central LIN master control unit orchestrates all communication on the bus. This master is responsible for generating the time base, initiating data frame transmission, and polling up to sixteen connected LIN slave control units in a cyclic or event-driven sequence. Slave nodes do not initiate communication independently but instead respond deterministically to the master's requests, ensuring a predictable and collision-free data exchange environment.

In its intended scope, LIN thus occupies a clearly defined niche within the hierarchy of automotive communication standards, complementing high-speed and low-speed CAN networks by offloading simpler, non-time-critical tasks. This targeted design strategy has allowed LIN to remain a viable and cost-effective solution for over two decades, with continued deployment in modern

vehicles where minimal hardware requirements and predictable performance are valued above raw transmission speed [6].

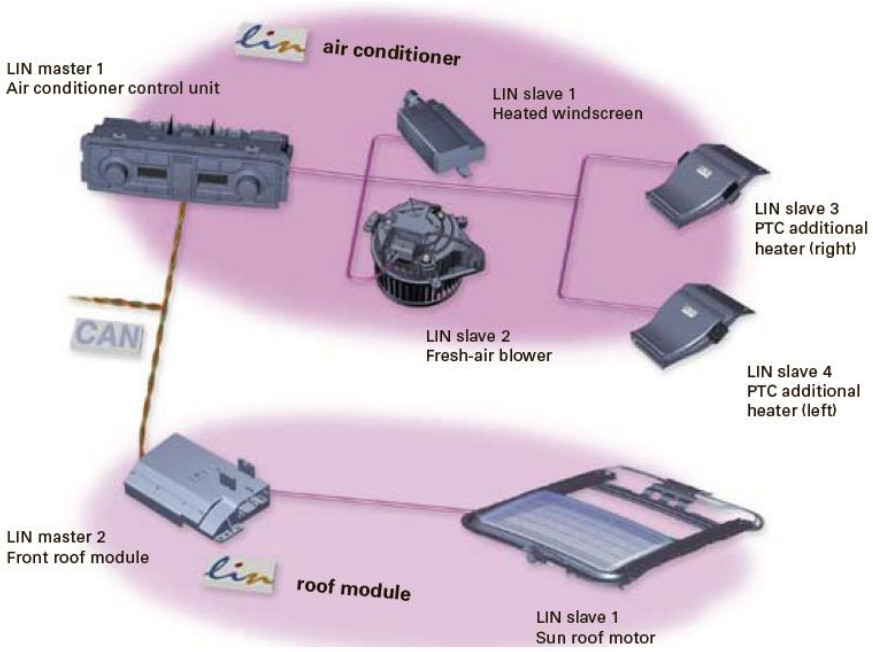


Fig.3.10. Communication diagram of control units via the LIN bus [6].

The LIN master, connected to the CAN data bus, acts as a “gateway” between local LIN sub-systems and the CAN network, and is the sole LIN unit with direct CAN bus access (Fig. 3.11).
It governs data transfer and transmission rates, and its software defines the cycle–specifying when, how frequently, and which messages are transmitted on the LIN bus.
Diagnostics of connected LIN slave units are carried out exclusively via the master control unit.

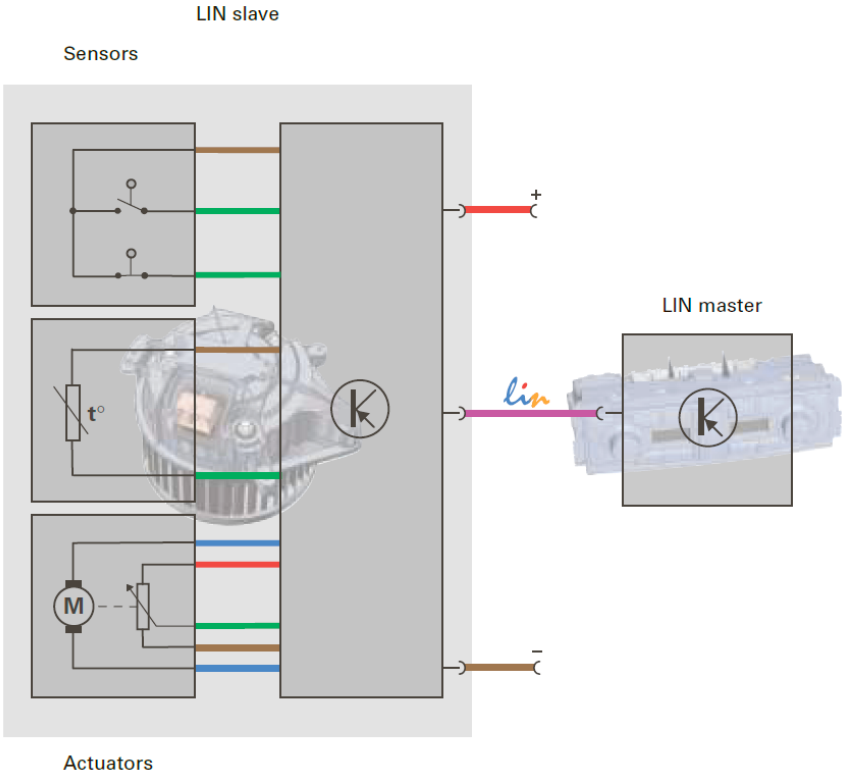


Fig. 3.11. Connection of the LIN-master control unit [6].

Within a LIN bus system, slave control units may include dedicated controllers–such as fan motor modules–as well as integrated sensors and actuators, for example, an inclination sensor or a DWA alarm siren. Sensors typically contain embedded electronic components that process measured values locally before transmitting them as digital signals over the LIN bus.
Multiple sensors or actuators can be connected via a single pin within the LIN master control unit's connector. The data transmission rate, defined in the software of LIN control units, ranges from 1 to 20 kbit/s–approximately one-fifth of the rate used in the CAN-Comfort bus [6].
Signal levels on the LIN bus are defined as follows (Fig. 3.12):

- Recessive level: When no telegram or recessive bit is transmitted, the bus line voltage is maintained close to the vehicle battery voltage.
- Dominant level: To transmit a dominant bit, the transmitting control unit connects the bus line to ground via its transceiver.

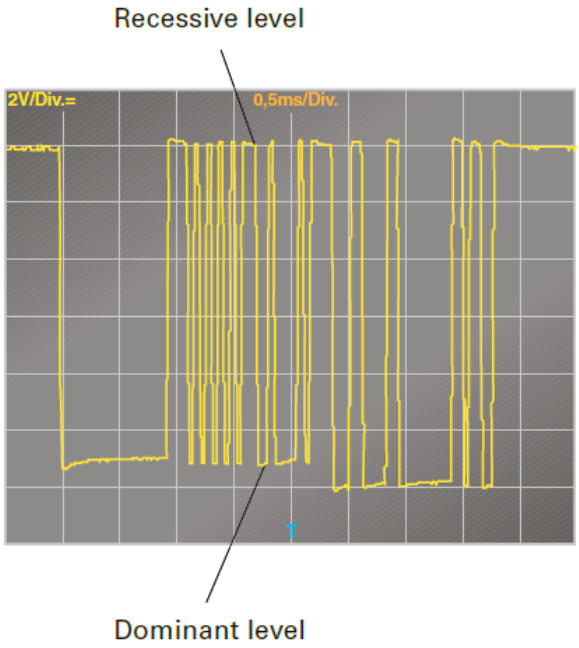


Fig. 3.12. Recessive and dominant signal levels on the LIN bus [6].

Stable data transmission is ensured by defining tolerance margins for both recessive and dominant levels during signal transmission and reception (Fig. 3.13).

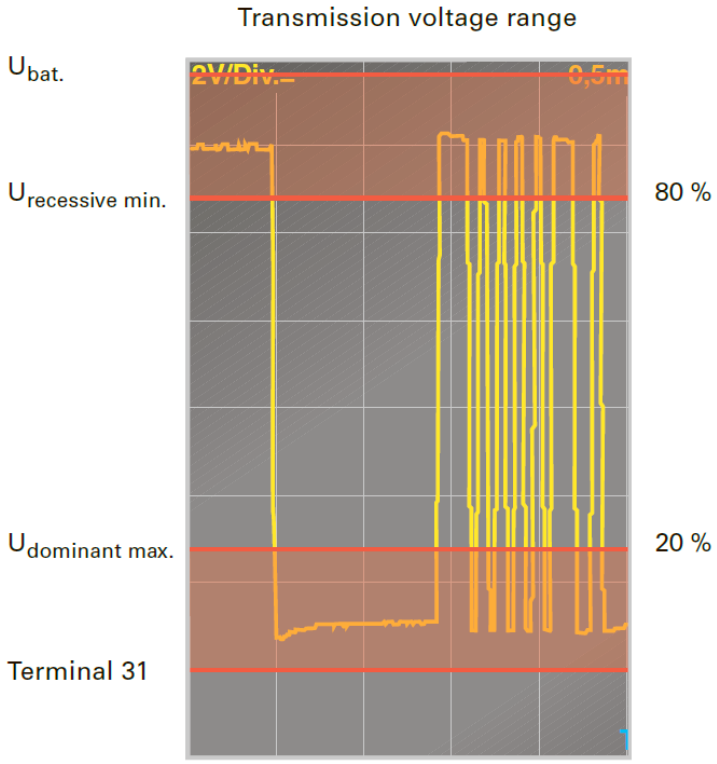


Fig. 3.13. Tolerances for transmission and reception at the recessive and dominant levels [6].

To improve signal reliability and eliminate interference caused by electromagnetic coupling ("cross-talk"), the permissible voltage range for data reception has been extended (Fig. 3.14).

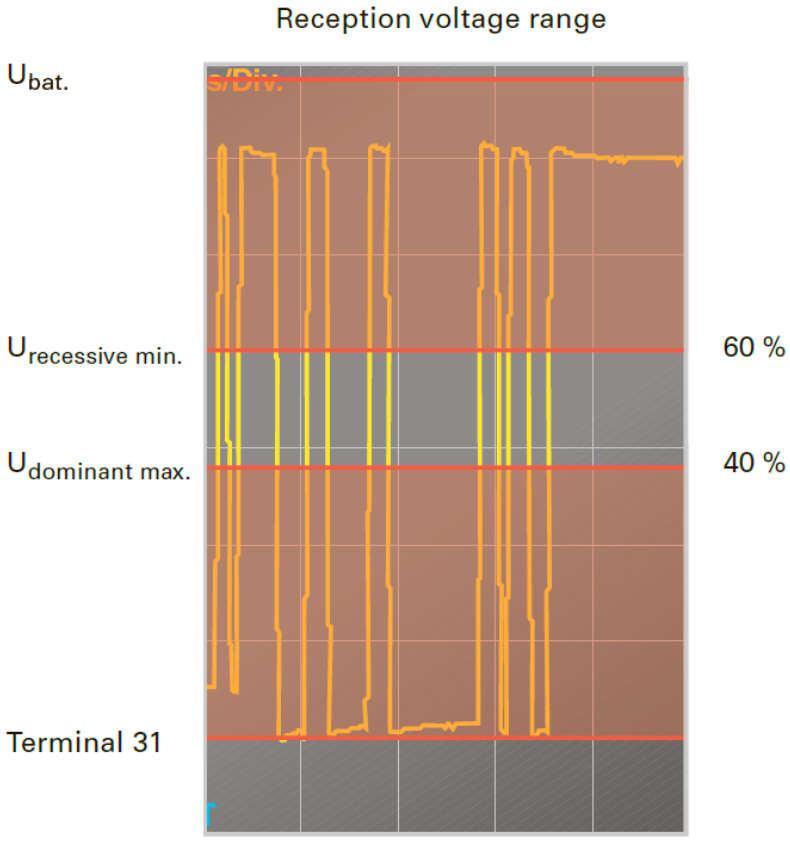


Fig. 3.14. Extension of the tolerance limits at the recessive and dominant levels [6].

Information in the system is transmitted in the form of telegrams, consisting of the Telegram Header (Header) – transmitted by the LIN-Master control unit – and the Telegram Content (Response) – transmitted either by the LIN-Master control unit or by a LIN-Slave control unit.
Two types of telegrams can be distinguished:

- Slave Response Telegram. In the header, the LIN-Master control unit requests the LIN-Slave control unit to send information, such as the status of switches or measured values. The response is sent by the LIN-Slave control unit.
- Master Command Telegram. In the header, through the identifier, the LIN-Master control unit requests the corresponding LIN-Slave control units to process the data contained in the response. The response is sent by the LIN-Master control unit.

In both cases, the LIN-Master control unit cyclically transmits the telegram header, which is subdivided into four phases (Fig. 3.15):

- Synchronization Break;
- Synchronization Delimiter;
- Synchronization Field;
- Identifier Field.

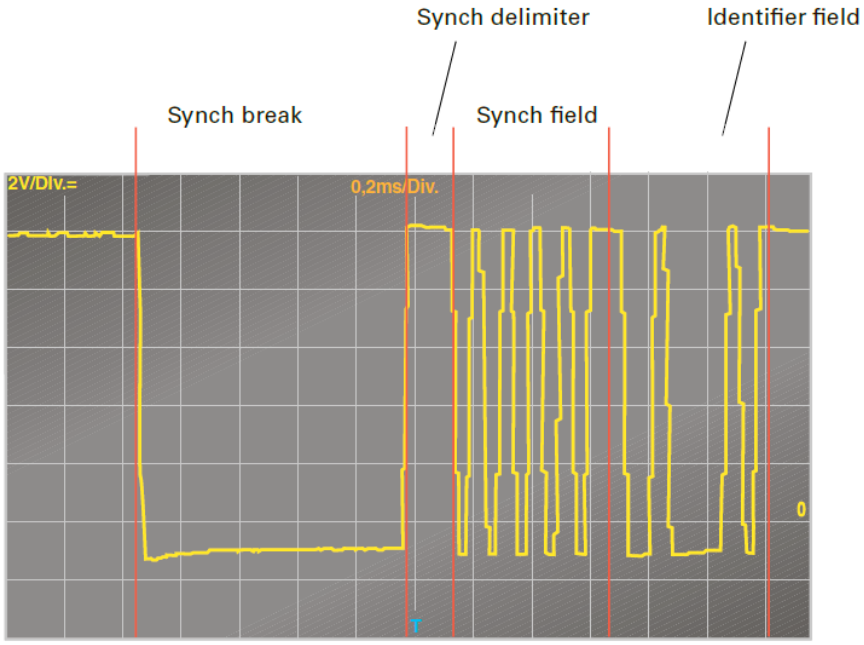


Fig. 3.15. The LIN-Master control unit header [6].

The minimum duration of the synchronization break (“synch break”) corresponds to the transmission time of 13 bits and is transmitted at the dominant level. This duration is required to unambiguously indicate to all LIN-Slave control units the start of a telegram transmission.

In the subsequent parts of the telegram, a maximum of nine dominant bits may be transmitted consecutively. The minimum duration of the synchronization delimiter (“synch delimiter”) equals the transmission time of one bit. The synchronization delimiter is a recessive signal (≈ UBat).

The synchronization field (“synch field”) consists of the bit sequence 0101010101. This sequence enables all LIN-Slave control units to adjust (synchronize) their internal clocks to the system timing. Accurate synchronization of all control units is ensured under uninterrupted data transmission. If synchronization is lost, bit positions in the telegram are misaligned at the receiver, leading to data transmission errors.

The identifier field has a duration equal to the transmission time of 8 bits. The first six bits contain the telegram identifier and the number of data fields in the response, which may range from

0 to 8. The final two bits contain either an error detection code for the transmission or a checksum of the first six bits. The checksum ensures that, in the event of identifier transmission errors, the telegram is not misclassified [6].

In a telegram containing a Slave response, the LIN-Slave control unit appends the Response with relevant information, based on the identifier. In a telegram requesting data from the Master, the LIN-Master control unit appends the response.

Depending on the identifier, the respective LIN-Slave control units interpret the data to execute their designated functions (Fig. 3.16).

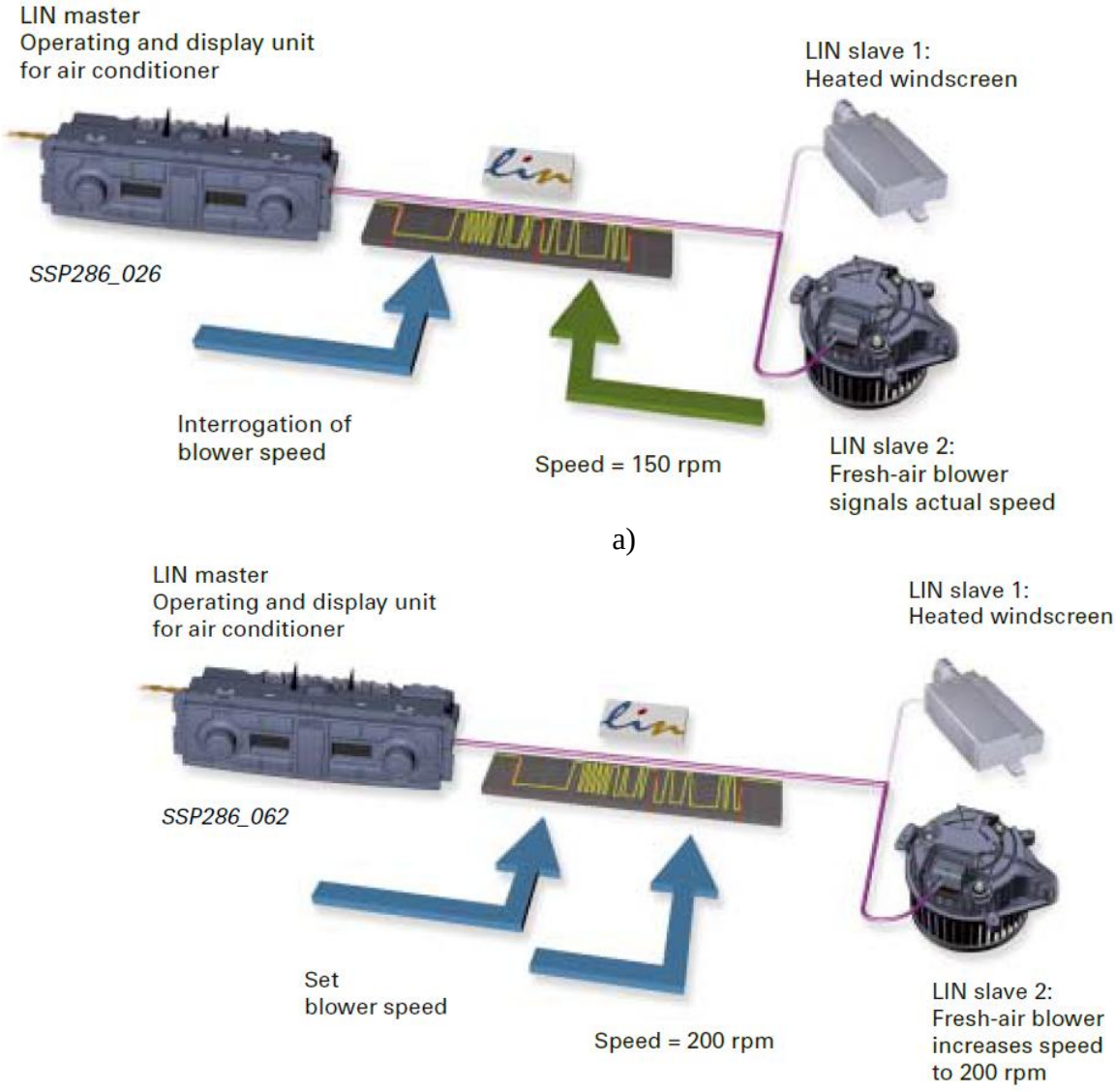


Fig. 3.16. Example of LIN operation in an automotive climate control system [6]**.**

(a) the LIN master (operating and display unit of the air conditioner) interrogates the blower speed; the LIN slave (fresh-air blower) reports the current speed of 150 rpm; (b) the LIN master sets the required blower speed; the LIN slave increases the blower speed to 200 rpm, while the second LIN slave (heated windscreen unit) remains connected to the bus.

The response comprises between one and eight data fields (Fig. 3.17).
Each data field consists of 10 bits, including one dominant start bit, one information byte, and one recessive stop bit.
The start and stop bits are used for subsequent synchronization and for preventing transmission errors.

Following the sequence defined in its software, the LIN-Master control unit cyclically transmits telegram headers on the LIN bus, while master telegrams also include corresponding responses.
Frequently used information is transmitted at significantly shorter intervals [6].
The transmission order of telegrams may vary depending on a range of factors, including the LIN-Master control unit itself. Examples of such factors include:

- ignition ON/OFF;
- diagnostic mode activated/deactivated;
- front position lights ON/OFF.

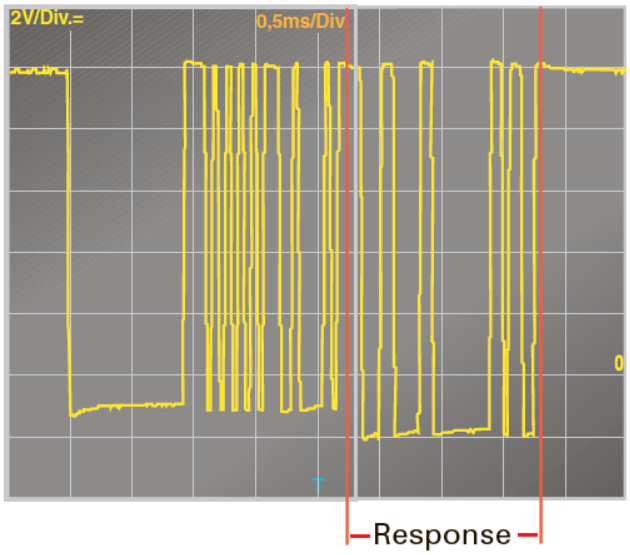


Fig. 3.17. Response data fields [6].

To reduce the variety of LIN-Master control unit variants, the master sends telegram headers for all control units of a fully equipped vehicle onto the LIN bus.
If certain control units for optional equipment are absent, the oscilloscope trace will show telegram headers without responses (Fig. 3.18). This has no impact on system performance.

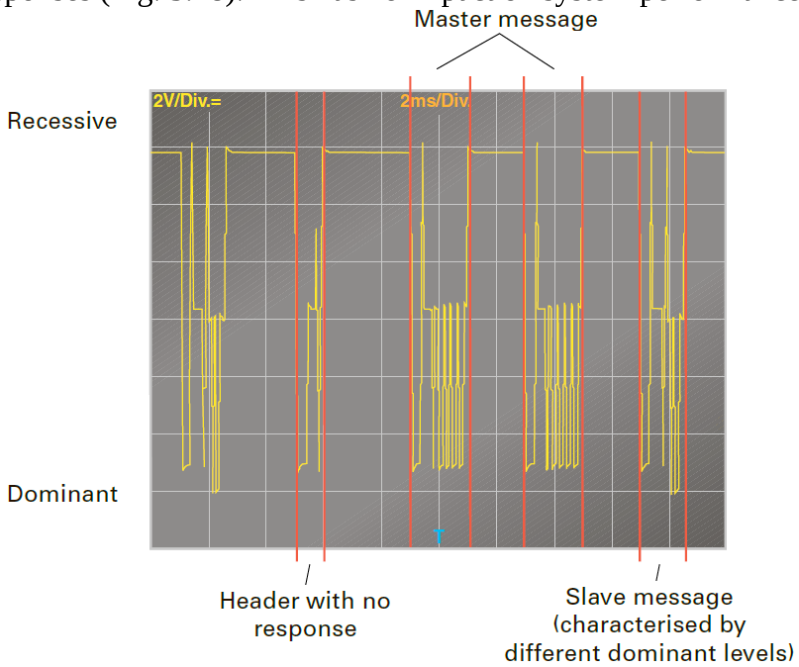


Fig. 3.18. Typical telegram structure [6].

Data transmission on the LIN bus occurs only when the LIN-Master sends a telegram header with the corresponding identifier. Manipulation of the LIN wiring located behind the vehicle's exterior trim is impossible, as all telegrams are fully controlled by the LIN-Master control unit.

A LIN-Slave unit can only transmit responses and cannot initiate commands. For example, it is not possible to unlock vehicle doors via the LIN bus. This design principle allows LIN-Slave control units to be installed in exterior components of the vehicle (e.g., a control unit for a garage door opener drive in the front bumper).

LIN system diagnostics are performed via the address word of the LIN-Master control unit. Diagnostic data transmission from a LIN-Slave to the LIN-Master occurs via the LIN bus.
All LIN-Slave control units are capable of executing the full range of self-diagnostic functions.

3.5. MOST Technology

Media Oriented Systems Transport (MOST) is currently employed in the automotive sector for data exchange between infotainment components. Unlike the Controller Area Network (CAN), where telegrams with specific addresses are directed to a single recipient, MOST enables the transmission of large volumes of data.
Originally developed by Oasis Silicon Systems, the technology was later standardised by MOST Cooperation, established in 1998, for application in multimedia network architectures within vehicles. The MOST bus supports data transfer rates of up to 21.2 Mbit/s, in contrast to CAN, whose maximum rate is 1 Mbit/s and is therefore suitable only for control signal transmission (Fig. 3.19). The adoption of an optical MOST bus facilitated fully digital communication between infotainment devices.

Fig. 3.19. Communication diagram of control units via the MOST bus [6].

Optical transmission offers several advantages over radio frequency methods: shorter wavelengths, immunity to electromagnetic interference, and elimination of emissions. This approach not only reduced wiring complexity and overall vehicle weight but also substantially increased transmission speed. As a result, MOST technology combines high data throughput with robust interference resistance.

3.5.1. Components of MOST Systems

Components of MOST Bus Control Units (Fig. 3.20):

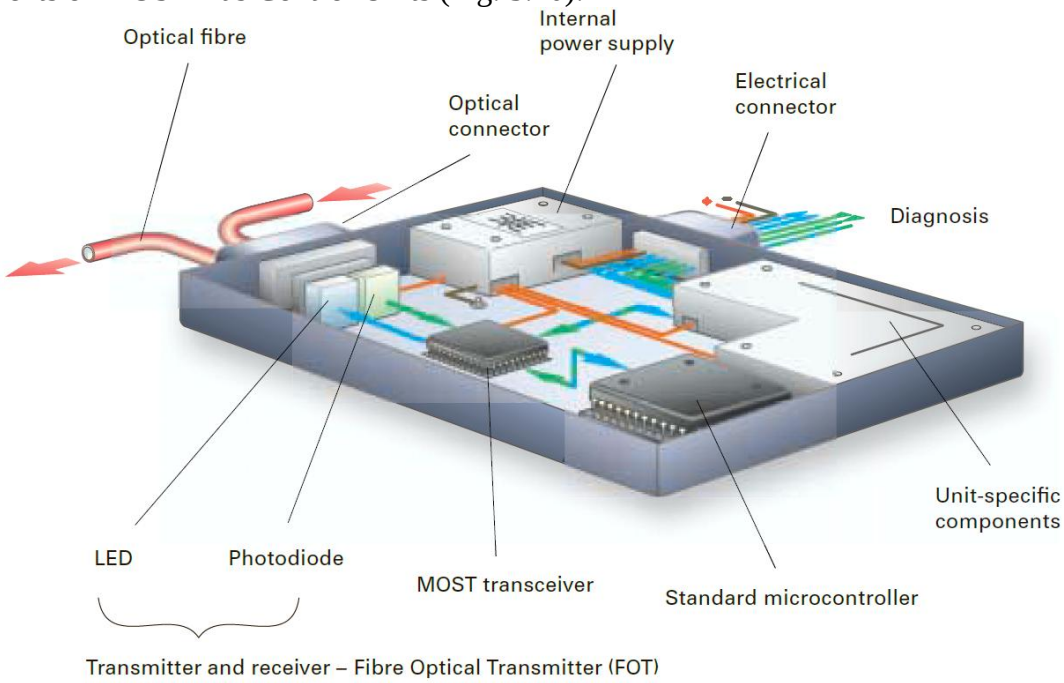


Fig. 3.20. Layout of the control units [6].

- Optical Fibre (LWL) – Optical Connector. This connector enables the transmission of optical signals into the control unit and the forwarding of generated signals to the next device in the MOST network.
- Electrical Plug Connector. Provides power supply, line-break diagnostics, and input/output signal interfaces.
- Internal Power Supply. Through the electrical connector, supply voltage is distributed within the control unit. Selected components can be switched off to reduce quiescent current consumption.
- Fibre-Optic Transmitter (FOT). The FOT comprises a light-emitting diode (LED) and a photodiode. Incoming optical signals are converted by the photodiode into voltage signals for the MOST transceiver. Conversely, the LED converts voltage signals from the transceiver into optical signals with a wavelength of approximately 650 nm, visible as red light. Data transfer is achieved by modulating these light waves, which are then transmitted via the optical fibre to the next control unit.
- MOST Transceiver. The transceiver incorporates a transmitter and receiver. The transmitter forwards telegrams as voltage signals to the FOT. The receiver processes incoming voltage signals, delivering relevant data to the unit's microcontroller, while non-relevant telegrams are passed through unchanged to the next unit.
- Standard Microcontroller (Central Processor). Serves as the control unit's computational core, integrating a microprocessor responsible for all primary operational functions.
- Device-Specific Components. Perform functions unique to the given control unit (e.g., CD drive, radio tuner).
- Photodiode. Converts optical waves into voltage signals via the internal photoelectric effect (Fig. 3.21). The photodiode's p-n junction generates electron-hole pairs when exposed to light, producing a current proportional to light intensity. In reverse bias and in series

with a resistor, increased illumination raises current flow, resulting in a higher voltage drop across the resistor, thus achieving optical-to-electrical signal conversion.

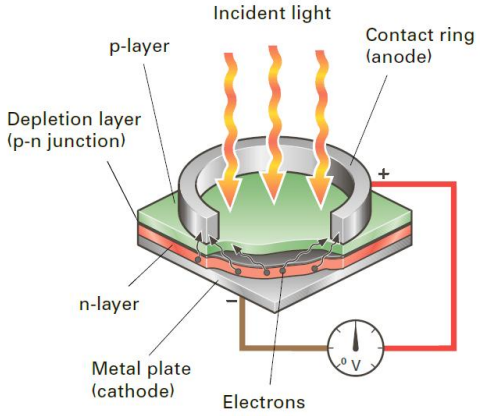


Fig. 3.21. Photoelectric effect [6].

- Optical Fibre (LWL). Transfers light waves from one control unit's transmitter to another's receiver (Fig. 3.22). Design considerations include straight-line propagation of light, with guided bending through fibre curvature; operational distances of several metres; mechanical resilience under vibration and handling; reliable function under automotive temperature extremes (–40 °C to +85 °C). Performance requirements are: low signal attenuation; bending tolerance without significant loss; flexibility for installation.

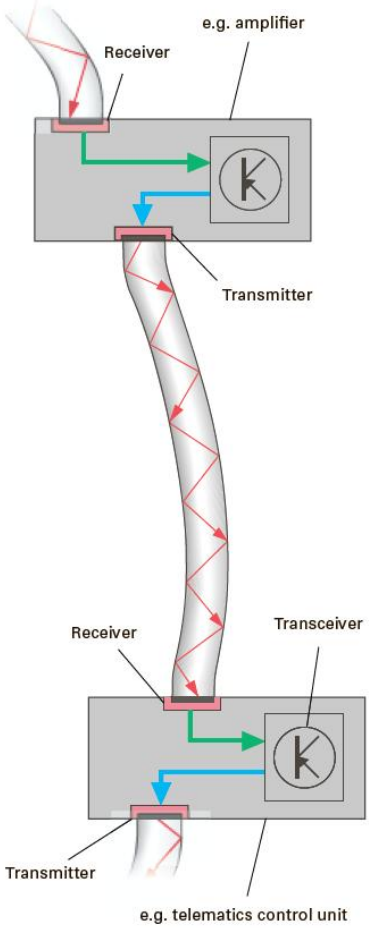


Fig. 3.22. Light guide device [6].

The optical fibre consists of multiple layers (Fig. 3.23). The core, made of polymethyl methacrylate (PMMA), functions as the primary light guide, transmitting light via the principle of total internal reflection with minimal losses. This reflection is ensured by a transparent cladding of fluoropolymer surrounding the core.
A black polyamide sheath shields the core from external light sources, while an outer coloured jacket provides mechanical protection, thermal resistance, and facilitates cable identification.

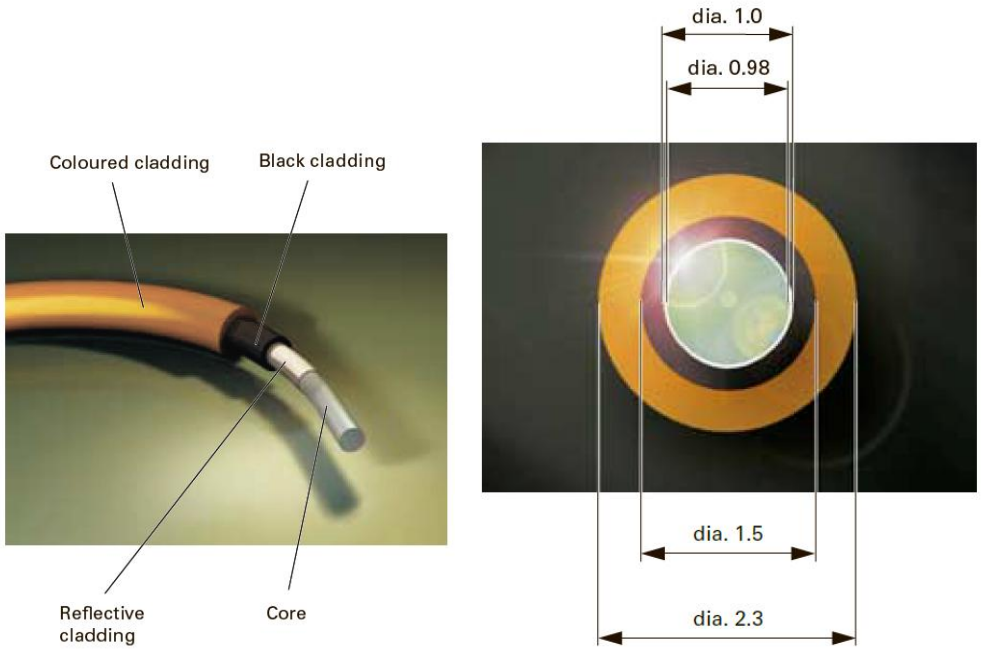


Fig. 3.23. Optical fibre structure [6].

Special optical connectors are used to interface the fibres with control units (Fig. 3.24).

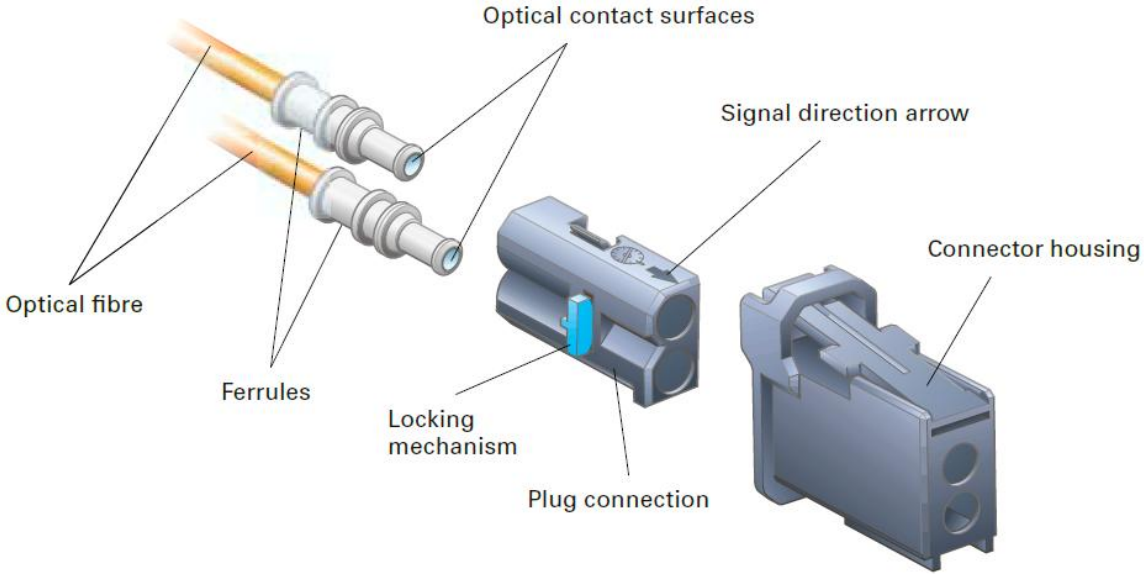


Fig. 3.24. Optical plug connection [6].

The plug housing mates with the control unit, and directional arrows on the connector indicate the signal input towards the receiver. Light transmission occurs through the polished end face of the core, which makes direct contact with the transmitter or receiver in the control unit. During

manufacturing fibre ends are secured within the connector housing either by laser-welded plastic tips or by crimp-fitted brass caps.

A defining characteristic of the Media Oriented Systems Transport (MOST) bus is its ring topology, in which control units transmit data unidirectionally over an optical fibre to the next node in the sequence. The data stream continues through each node until it returns to the originating control unit, thus completing a closed-loop structure (Fig. 3.25). This architecture inherently supports deterministic latency and predictable bandwidth allocation, making MOST particularly suitable for high-quality multimedia and infotainment applications.

Diagnostics are performed through a dedicated MOST Data Bus Diagnostic Interface in combination with Controller Area Network (CAN)-based diagnostic tools. The CAN interface serves as a gateway, enabling diagnostic access to MOST network components without requiring direct optical-layer interaction from the service tool.

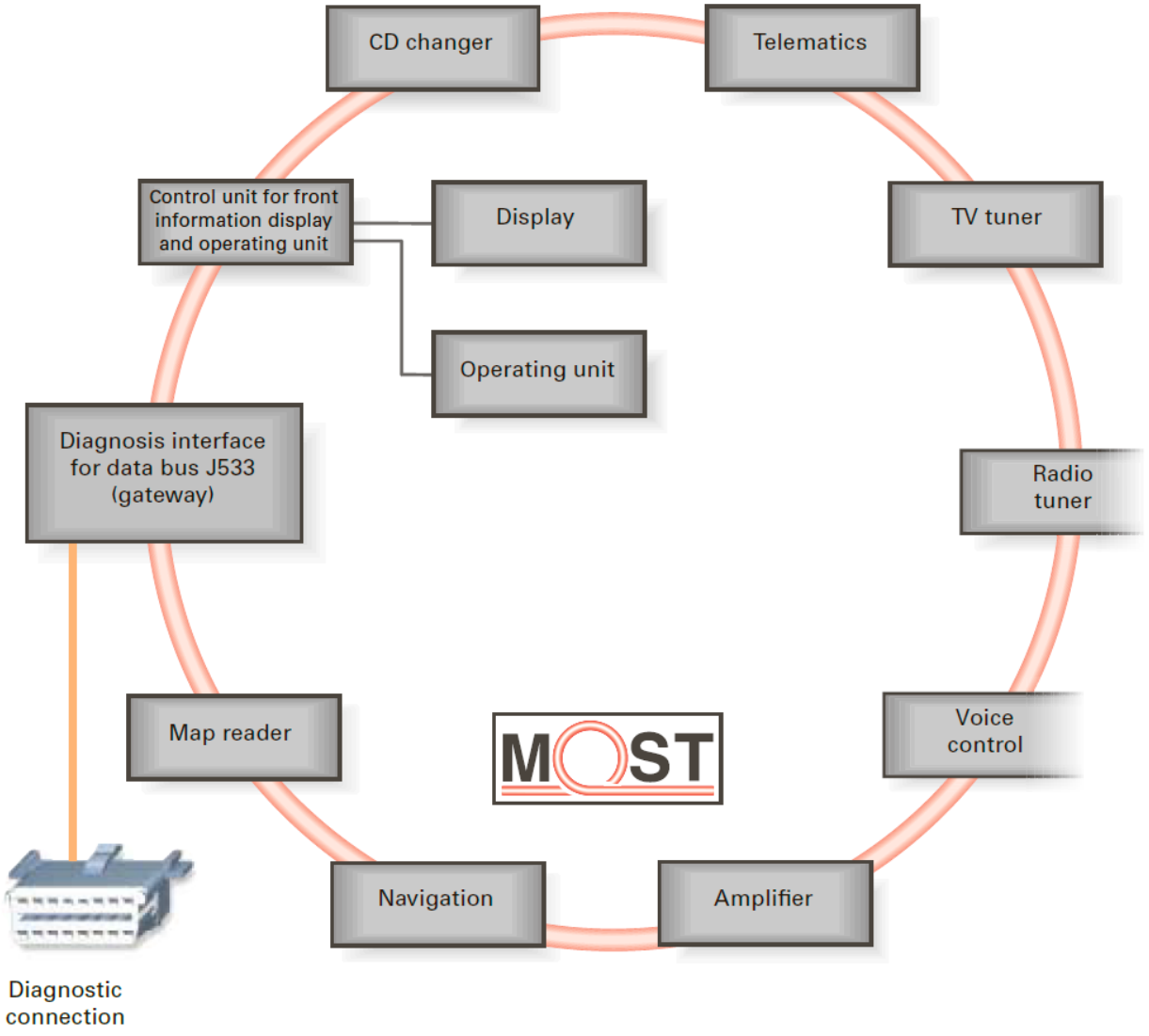


Fig. 3.25. Ring Topology of the MOST Bus [6].

3.5.2. Operational States

The MOST Bus Operating States are distinguished:

1. Sleep Mode. In this low-power state, no data exchange occurs, and all connected devices remain in a state of standby readiness. The network can be reactivated by an optical wake-up pulse generated by the system administrator node. Quiescent current draw is kept to a minimum to reduce battery load. Conditions for entry into Sleep Mode:

- all control units signal readiness for sleep;
- no requests are received via the Gateway from other networks;
- no diagnostics are in progress.

Additional triggers include:

- gateway command from the battery management system when the starter battery is discharged;
- activation of transport mode via a diagnostic tester.

2. Standby Mode. The system appears inactive from a user perspective, yet the MOST network remains operational. Output devices such as displays and audio amplifiers are powered down to conserve energy. This state is typically active during system idle periods and immediately after initial wake-up. Activation occurs via:

- other data buses through the Gateway (e.g., unlocking the driver's door, switching ignition to ON);
- a MOST network control unit (e.g., incoming telephone call).

3. Power-On Mode. All control units are fully active, with complete data exchange and full functionality available to the user. Conditions:

- the system is in Standby Mode;
- activation by other data buses via the Gateway (e.g., S-contact, display activation);
- user selection of a function (e.g., multimedia control E380) [6].

3.5.3. Telegram Segments and Clocking

Data transfer in MOST networks is tightly synchronised through a master clock generated by the system administrator node. This node continuously transmits telegram segments–also referred to as frames–around the ring at a constant frequency of 44.1 kHz. This clock rate matches the standard sampling frequency of digital audio systems (e.g., CD/DVD players, DAB radio), allowing direct and lossless integration of audio data streams.
Each frame has a fixed length of 64 bytes (Fig. 3.26).

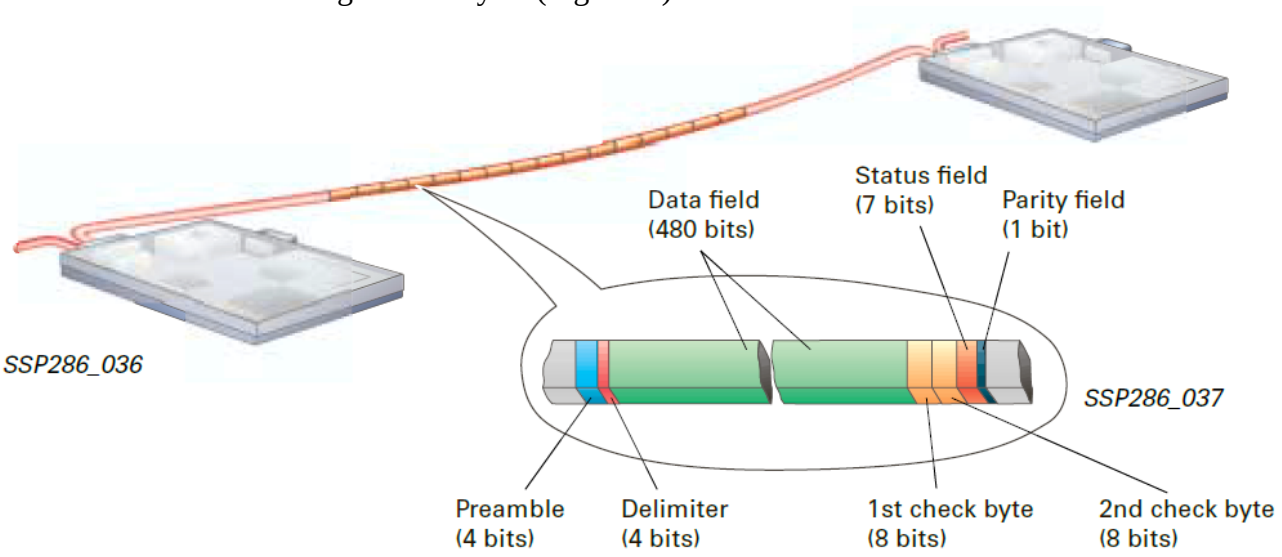


Fig. 3.26. The telegram section structure [6].

A preamble marks the start of each frame (Fig. 3.27). A delimiter field separates the preamble from the subsequent data field, which can carry up to 60 bytes.

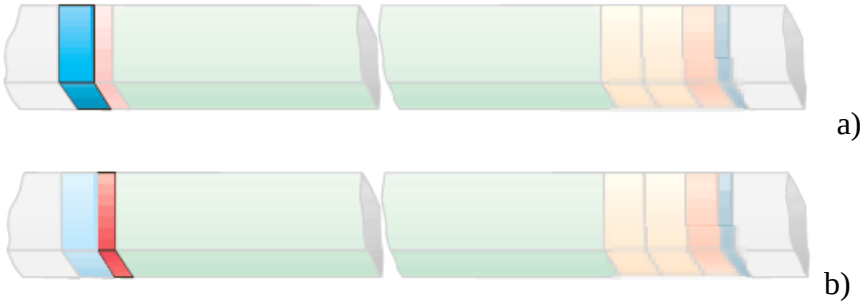


Fig. 3.27. Frame Structure: a) preamble; b) delimiter field [6].

MOST differentiates between two main types of transmitted data (Fig. 3.28):

- synchronous data – Primarily audio and video streams, requiring precise and consistent timing intervals;
- asynchronous data – Such as still images, computational results, and textual information.

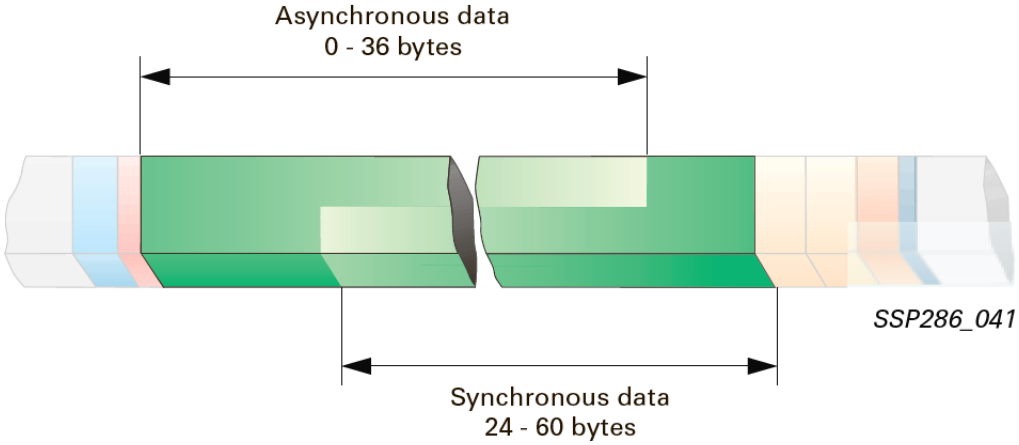


Fig. 3.28. Telegram data types [6].

Allocation within the data field is dynamic: synchronous data may occupy 24–60 bytes depending on system configuration, always taking priority over asynchronous traffic. Asynchronous data is organized in 4-byte quadlets and is individually addressed.

The payload also contains two control bytes that carry metadata, including:

- transceiver Address (Identifier) – unique address of the sending device;
- receiver Control Commands – for example, volume control or mode switching commands for an amplifier.

Control bytes from multiple consecutive frames are aggregated into a control frame, with 16 frames grouped into a block (Fig. 3.29). Control frames facilitate operational commands and diagnostic data transfer between specific source and destination nodes (addressed data transfer).

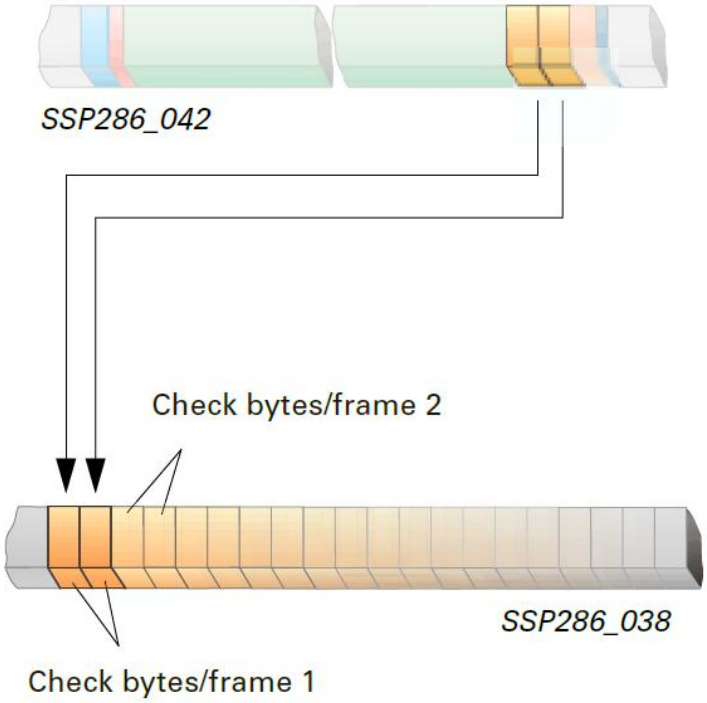
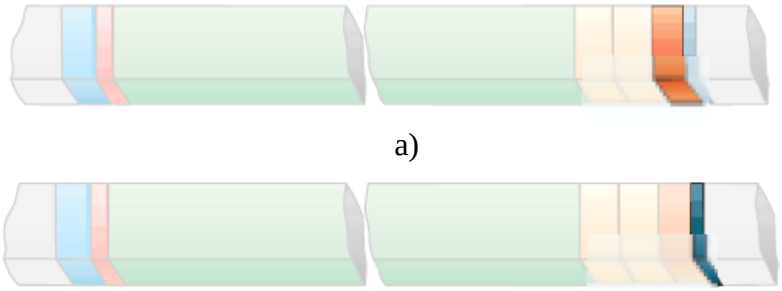


Fig. 3.29. Frame Structure: control frame [6].

The status field informs the receiver about the frame's transmission status, while the parity field provides a final integrity check, indicating whether retransmission is required (Fig. 3.30).



a)

b)

Fig. 3.30. Frame Structure: a) status field; b) parity field.

3.5.4. Process Flow in the MOST Bus

The flow of processes in the MOST bus includes a sequence of steps the first of which is System Start-Up (Wake-Up).
When the MOST bus is in Sleep Mode, the wake-up process first transitions the system into Standby Mode. If a control unit other than the system administrator initiates wake-up, it sends a modulated optical Slave signal to the next control unit in the ring (Fig. 3.31).

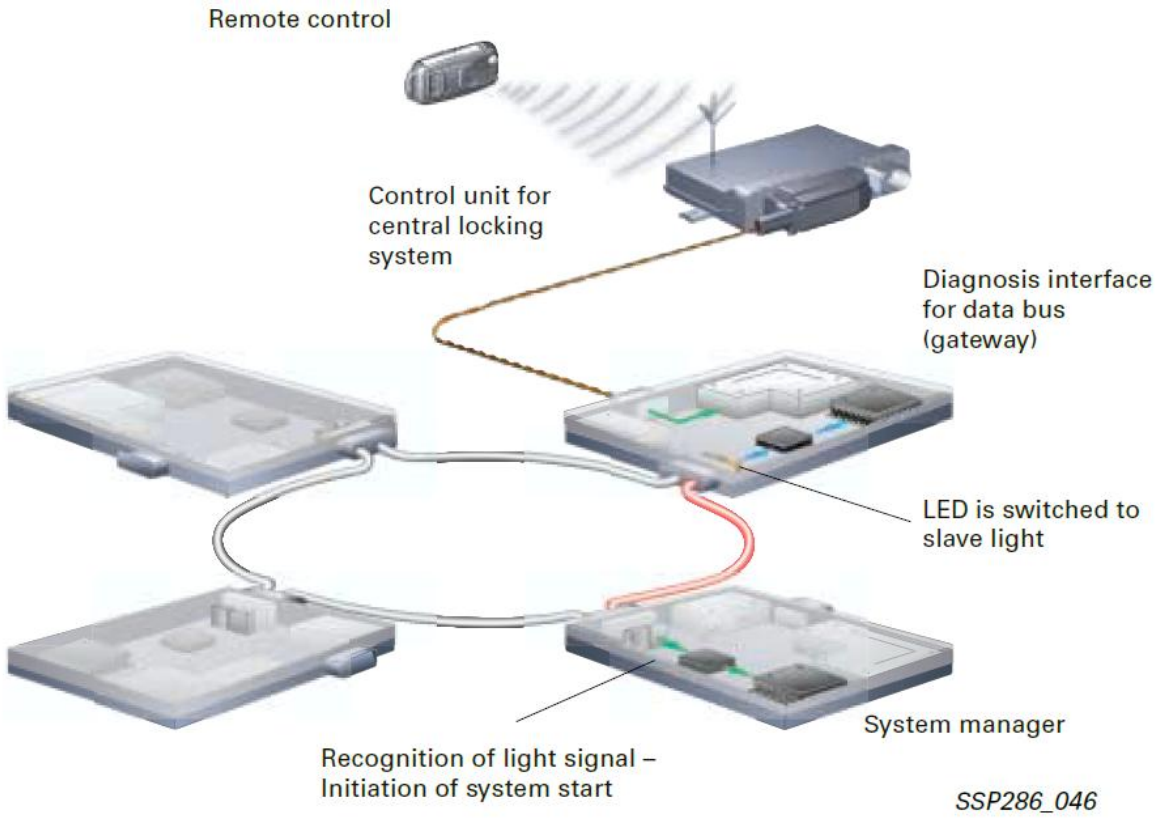


Fig. 3.31. System Start-Up in the MOST Bus [6].

Owing to LEDs active in Sleep Mode, the next control unit receives the Slave signal and forwards it to the subsequent unit, continuing until the signal reaches the system administrator. Upon receiving the Slave signal, the administrator interprets it as a start-up command and responds by transmitting a modulated optical Master signal to the next unit.

All control units relay the Master signal in sequence. Once the system administrator receives its own Master signal back through the fibre-optic transmitter, it confirms the ring is closed and begins frame transmission (Fig. 3.32).

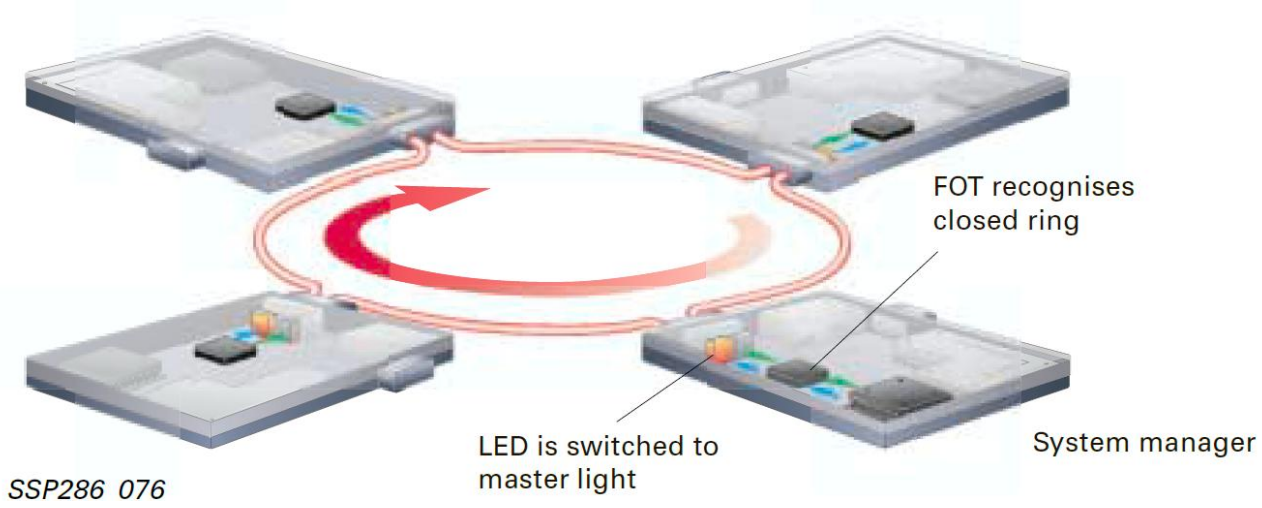


Fig. 3.32. The processes of frame transmission [6].

The initial telegram segments contain commands instructing control units to perform self-identification. Based on the identification results, the administrator distributes the current system sequence (actual configuration) to all control units in the ring, enabling addressed data transfer. The diagnostics administrator then compares the registered control units with the stored list of installed units (nominal configuration).

If discrepancies are found between the actual and nominal configurations, a corresponding fault entry is recorded in the error memory. The wake-up process is then complete, and normal data transmission can commence (Fig. 3.33).

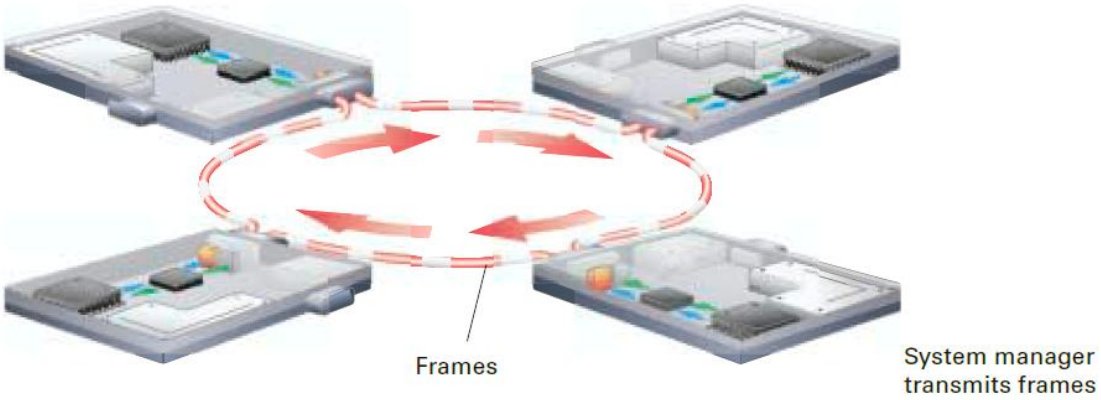


Fig. 3.33. The processes of data transmission [6].

3.6. Bluetooth Technology

Bluetooth is a short-range wireless communication standard originally developed by Ericsson, combining radio-frequency transmission with networking protocols to facilitate the formation of small-scale, ad hoc wireless networks. Operating in the globally available, licence-free 2.4 GHz Industrial, Scientific, and Medical (ISM) band, Bluetooth shares its spectrum with devices such as cordless telephones, Wi-Fi equipment, and microwave ovens, which can result in susceptibility to electromagnetic interference under certain conditions (Fig. 3.34).

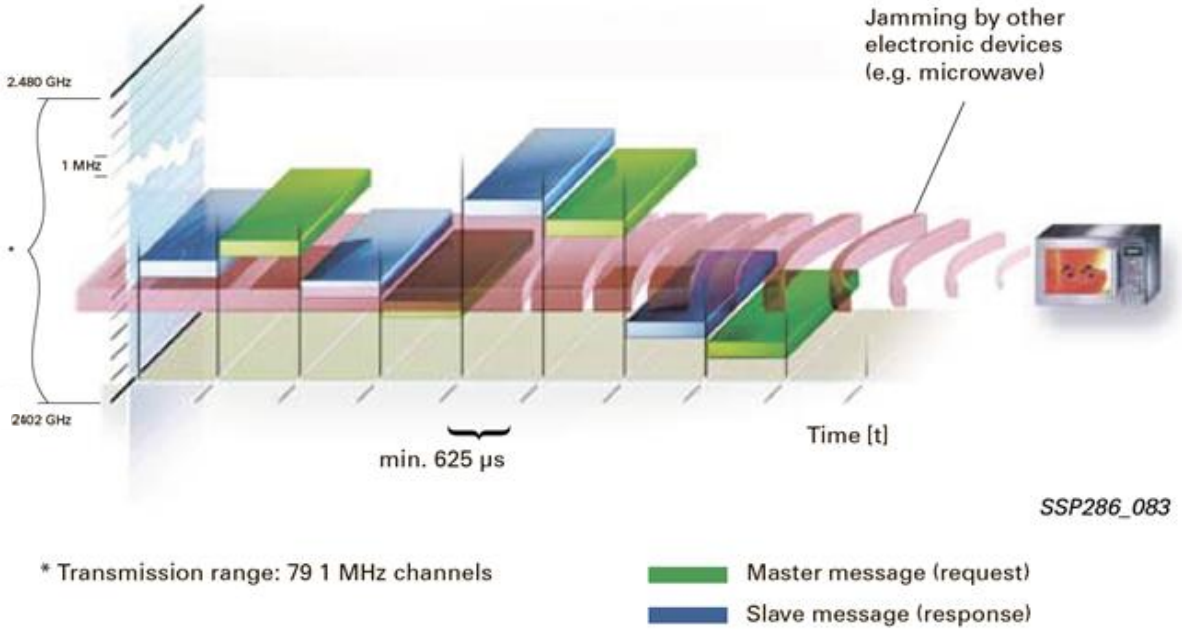


Fig. 3.34. Bluetooth Technology [6].

The effective communication range of a Bluetooth link is determined by the transmission power class of the device. Class 3 modules typically operate within 1–10 m, Class 2 devices achieve ranges of approximately 10–30 m, and Class 1 modules can exceed 100 m under favourable conditions.

A Bluetooth piconet can theoretically register up to 260 devices, although the protocol restricts simultaneous active participation to eight devices – one acting as the master and up to seven as active slaves. The remaining devices maintain time-synchronisation with the network and can be activated on demand without requiring a full reconnection sequence.

Under optimal conditions, the nominal data rate exceeds 700 kbit/s, though actual throughput often falls below this value due to environmental interference, network congestion, and the overhead associated with error correction and encryption [6].

The Bluetooth control module performs the following key functions:

- Segments information into short, flexible data packets (~625 μs duration).
- Verifies packet integrity via a 16-bit checksum.
- Automatically retransmits corrupted packets.
- Employs robust speech coding to convert voice into digital signals.

The Bluetooth radio transceiver implements frequency hopping spread spectrum (FHSS) at a rate of 1,600 frequency changes per second, following a pseudo-random hopping sequence known to both communicating devices. This technique significantly reduces the probability of prolonged interference on any single channel and enhances co-existence with other ISM band technologies.

Data security was a core design priority, with protection against manipulation, interception, and unauthorised listening. Bluetooth employs 128-bit encryption for data confidentiality and receiver authentication. Devices exchange a unique passkey to recognise each other, and a new session key is generated for every connection. The limited operating range (typically 10 m) further mitigates the risk of external manipulation.

In addition to these physical and cryptographic safeguards, advanced coding techniques, layered security mechanisms, and network protocols are implemented by manufacturers to enhance data protection [6].

3.7. FlexRay Technology

The FlexRay Consortium was founded in 2000 by a group of manufacturers to develop a new data exchange protocol. Over time, additional members joined, including Volkswagen.

The aim of the FlexRay protocol is to meet the growing demands placed on modern automotive communication systems, particularly in terms of speed, real-time capability, and reliability. It extends the scope of in-vehicle data exchange, enabling applications such as vehicle dynamics control, adaptive cruise control, and video data processing [7].

Unlike its predecessors CAN, LIN, or MOST, FlexRay operates on a fundamentally different principle, illustrated by the analogy of a cable car system (see Table 3.5). In this analogy cable car stations represent the bus participants (control units as transmitters/receivers); cable cars correspond to messages; passengers represent the data within those messages.

Each FlexRay participant has a precisely defined time slot for transmission, and receivers know exactly when to expect messages. As with cable cars departing on schedule regardless of passenger load, a FlexRay time slot remains reserved even if no new data is available. In such cases, the control unit retransmits previous data rather than leaving the slot empty, as an “empty cabin” would be interpreted as a transmitter fault. Newly transmitted data is marked by an Update Bit.

A key feature of FlexRay is its use of Time Division Multiple Access (TDMA), a multiplexing method that allocates exclusive time intervals for each node’s transmissions. Outside its assigned time slot, a node cannot transmit. This eliminates collisions and blocks unauthorised messages, rendering attacks such as CAN Bomber ineffective, as all out-of-slot frames are ignored.

Table 1.5 Comparison of CAN and FlexRay Buses [7]

Characteristic	CAN	FlexRay
Connection	electrical, two-wire	electrical, two-wire
Signal states	"0" – dominant, "1" – recessive	"idle", "Data 0", "Data 1"
Transmission rate	500 kbit/s	10 Mbit/s
Access organisation	event-driven	time-triggered
Topologies	bus, passive star	active star, point-to-point, daisy chain[1]
Arbitration	a message with higher priority is transmitted before one with lower priority	not used, data transmission follows a predefined schedule (time-triggered)
Acknowledgement of reception	the receiving device confirms correct reception of the data transmission protocol	the transmitting device does not receive information as to whether the data transmission protocol has been correctly transmitted (received) or not
Error protocol	an error can be indicated in the network using an error protocol	each receiving device independently checks the correctness of the received data transmission protocol
Length of data transmission protocol	no more than 8 bytes of user data	no more than 256 bytes of user data
Use	• as required; • the time at which the CAN bus can be used depends on the bus load; • increasing load may result in CAN bus congestion	• the time at which the data transmission protocol may be used is predefined; • duration of usage is predefined; • the data transmission slot always remains reserved, even if it is not used
Reception time	unknown	known

A FlexRay communication cycle is divided into several segments, which are further subdivided into time slots.

The FlexRay communication cycle is divided into the following segments:

1. Static Segment – ensures real-time operation and collision-free transmission based on the TDMA principle. Each node must transmit its message within its assigned time slot.
2. Dynamic Segment – allows event-triggered data transmission. Here, FlexRay functions similarly to the CAN bus but still follows a predefined slot schedule, implementing Flexible Time Division Multiple Access (FTDMA).
3. Symbol Window – reserved for service messages, such as initiating a network wake-up.

The Dynamic Segment and Symbol Window are optional; the Static Segment is mandatory and forms the backbone of the protocol due to its deterministic TDMA scheduling.

Combination of segments in one communication cycle is shown in Fig. 3.35.

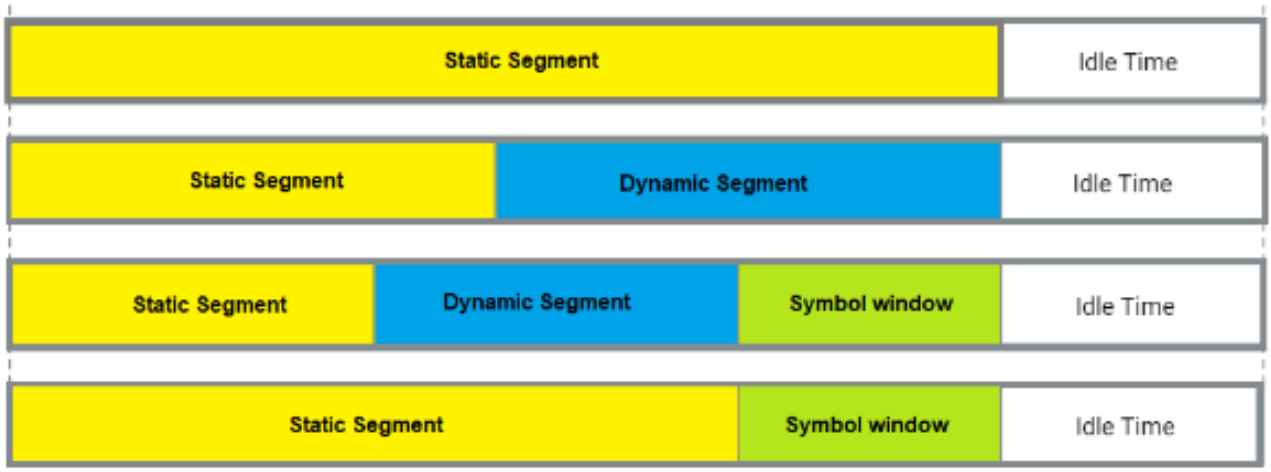


Fig. 3.35. Combination of segments in one communication cycle [7].

The static segment transmits user data between bus components (Fig. 3.36). It is divided into 62 fixed time slots, each assigned to a specific device. Only the designated device may transmit during its slot, though all participants receive all static-slot messages, including those not addressed to them.

Each static slot is exactly 42 bytes long, and the slot sequence is fixed. Devices may transmit different data in successive communication cycles while retaining the same slot assignment. The slot structure is repeated in every cycle, regardless of whether new user data is present. In Audi's implementation, devices always fill their allocated slots, using an Update Bit to indicate new messages.

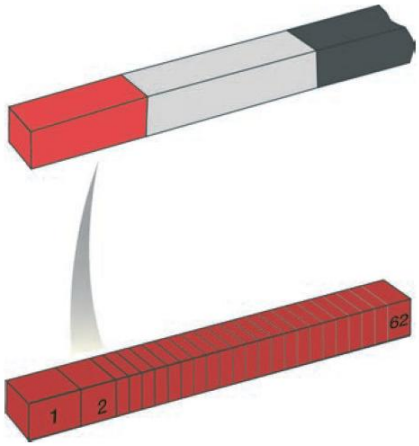


Fig. 3.36. Static Segment [7].

The dynamic segment is subdivided into mini-slots and is accessible to all devices (Fig. 3.37). It provides reserved bandwidth for event-driven transmissions within the communication cycle.

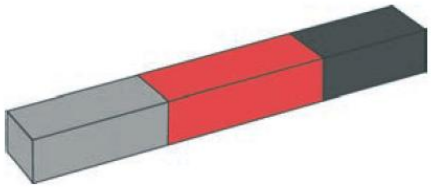


Fig. 3.37. Dynamic Segment [7].

Network idle time is the period when no messages are transmitted on the FlexRay bus (Fig. 3.38). It is used by the J533 diagnostic interface to synchronise the internal clocks of all bus participants with the system time.

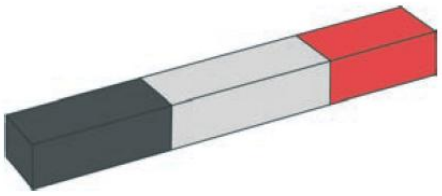


Fig. 3.38. Network Idle Time [7].

3.7.1. FlexRay Bus Architecture and Operation

The FlexRay communication bus implemented in the 2010 Audi A8 employs a hybrid topology that combines an active star configuration with point-to-point and daisy-chain interconnections. Specifically, branch 3 adopts a direct point-to-point layout, while branches 1, 2, and 4 utilize daisy-chain links (Fig. 3.39). The central coordinating role in the network is performed by the J533 Data Bus Diagnostic Interface, which acts as the primary bus controller. This interface is equipped with connectors for all four branches, thereby enabling a centralised architecture in which each branch integrates additional control units into the network [7].

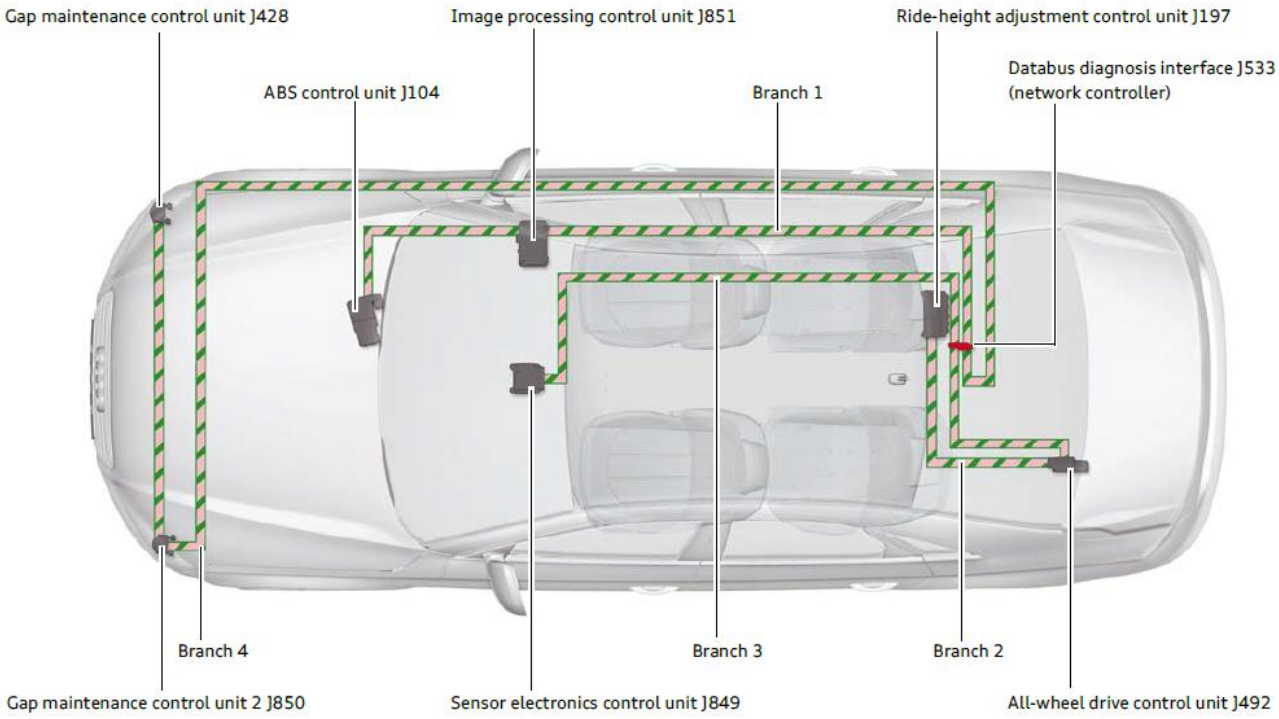


Fig. 3.39. FlexRay Bus Architecture [7].

Intermediate control units within a daisy-chain branch incorporate four connector pins: two dedicated to forwarding the FlexRay signal to the next unit in the chain, and two for linking the control unit itself to the network (Fig. 3.40). Terminal control units–such as the ABS Control Unit J104–are equipped with only two contacts, as no further forwarding is required. The physical structure of the network is designed such that no more than two control units are present per branch, ensuring predictable signal propagation characteristics.

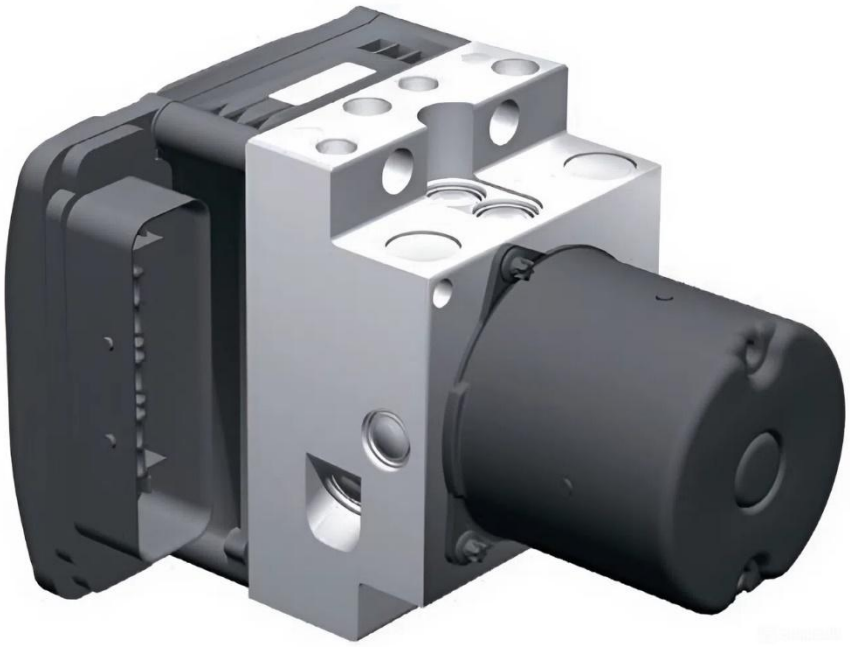


Fig. 3.40. Intermediate control units [7].

From an electrical standpoint, active star modules and terminal control units present low internal resistance, which facilitates robust signal integrity. In contrast, intermediate control units are designed with higher internal resistance to ensure correct termination and signal reflection control within the chain topology [7].

3.7.2. Operational States

The FlexRay protocol defines distinct operational phases to manage power consumption, synchronisation, and data exchange efficiency [7]:

1. Wake-Up – In the low-power sleep state, the network is activated by the transmission of a dedicated wake-up symbol from any capable control unit. Upon receiving this signal, the bus transitions into standby mode, in which physical connectivity is active but no application data is exchanged.

2. Start-Up – Actual data communication begins only after a start-up process is initiated by a designated starting node. The first starting node to transmit effectively triggers network synchronisation. Only starting and synchronising nodes possess the capability to initialise the network. In the Audi A8 configuration, these include:

- J533 Data Bus Diagnostic Interface;
- J104 ABS Control Unit;
- J849 Sensor Electronics Control Unit.

Non-starting nodes cannot initiate the network and only transmit after at least two other units have begun transmission (Fig. 3.41). Examples include:

- J428 Adaptive Cruise Control Unit;
- J850 Adaptive Cruise Control Unit 2;
- J851 Image Processing Control Unit;
- J492 All-Wheel Drive Control Unit;
- J197 Level Control System Unit (non-starting, but participates in synchronisation).

Fig. 3.41. Non-starting nodes [7].

3.7.3. Signal States and Transmission Characteristics

In a FlexRay communication bus, the two conductors of the differential pair are supplied with complementary voltages, conventionally referred to as the “positive” and “negative” lines. The absolute voltage level on each conductor varies within the range of approximately 1.5 V (minimum) to 3.5 V (maximum).

The FlexRay protocol operates with three distinct signal states, each defined by the voltage relationship between the two lines:

- Idle — both conductors are held at an equal potential of 2.5 V, indicating the absence of active data transmission.
- Data 0 — the voltage on the positive line is lower than the voltage on the negative line.
- Data 1 — the voltage on the positive line is higher than the voltage on the negative line.

Each bit occupies a time interval of 100 ns. The actual transmission time is influenced by both the physical length of the cable and the signal transition time of the bus driver circuitry. Signal transmission is strictly differential; consequently, both conductors are required for proper data communication (Fig. 3.42).

The receiving node determines the current bit state by measuring the instantaneous voltage difference between the two conductors. Under normal operating conditions, the typical differential voltage amplitude lies between 1.8 V and 2.0 V. At the output of the transmitting device, the minimum permissible differential voltage is 1200 mV, while at the receiving end this value must not fall below 800 mV to ensure reliable detection.

If no bus activity is detected within a time window ranging from 640 μs to 2660 μs, the FlexRay network autonomously transitions to the idle state.

During the wake-up sequence, the control unit initiating the process transmits a so-called wake-up symbol onto the FlexRay bus. Prior to transmission, a mandatory idle period is observed to ensure that no ongoing communication is present on the bus and that all connected control units are indeed in the sleep mode. This precaution prevents collisions and guarantees synchronisation during network reactivation [7].

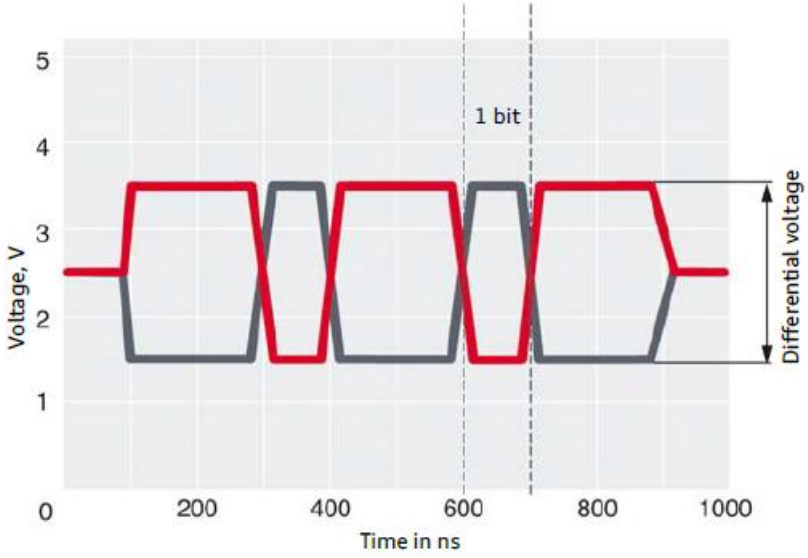


Fig. 3.42. Signal States FlexRay bus [7].

In the FlexRay communication network, the primary start-up control unit, which initiates the network activation process, commences data transmission based on its own internal clock, operating without prior synchronisation corrections. The secondary start-up control unit subsequently aligns its internal timing parameters with the data stream received from Start-Up Node 1. Only when at least two start-up nodes are actively exchanging data does the non-start-up control unit (Non-Start-Up Node) of the FlexRay bus begin its own synchronisation process. This staged initialisation sequence ensures that the network achieves temporal coherence before data traffic from standard nodes is introduced, thereby preventing timing conflicts and maintaining deterministic communication.

3.7.4. Diagnostics

The data bus diagnostic interface J533 is capable of detecting faults within the network and initiating corrective measures to ensure that unaffected network segments remain operational. Faults may be confined to a single branch of the network or, in more severe cases, may propagate throughout the entire communication structure [7].

Using a diagnostic tester, the following categories of FlexRay bus faults can be detected (address word 19 – Data Bus Diagnostic Interface):

- Control unit – no communication detected.
- FlexRay data bus fault.
- FlexRay data bus initialisation failure.
- FlexRay data bus signal fault.
- FlexRay Bus Behaviour Under Fault Conditions

FlexRay Bus Behaviour in Case of Faults:

1. Short-circuit of one conductor to ground

The J533 diagnostic interface detects a constant voltage difference across the bus conductors. The affected branch of the network is electrically isolated until an idle state (i.e., the quiescent 2.5 V voltage level) is detected again, indicating that normal conditions have been restored.

2. Short-circuit between conductors

The diagnostic interface detects a constant idle state across the bus pair. Communication between nodes on this branch becomes impossible due to the loss of differential signalling capability.

3. Continuous idle transmission from a control unit

If a node persistently transmits an idle state, the diagnostic interface identifies the anomaly and deactivates the corresponding branch to prevent disruption of the remaining network segments.

FlexRay Bus Cable Repair. The FlexRay bus cable, similar to that used in Controller Area Network (CAN) systems, is constructed as a twisted-pair conductor, encased within an additional outer jacket. This jacket does not function as an electromagnetic shield; rather, it provides environmental protection, minimising the influence of external factors such as humidity and temperature on the cable's characteristic impedance.

Partial replacement of a FlexRay bus cable is, in principle, feasible. However, repair procedures must strictly adhere to specified parameters for (Fig. 3.43):

- Untwisting length (1) – the permissible length of untwisted conductor before termination;
- Jacket removal length (2) – the section of protective sheath removed for splicing or connector installation.

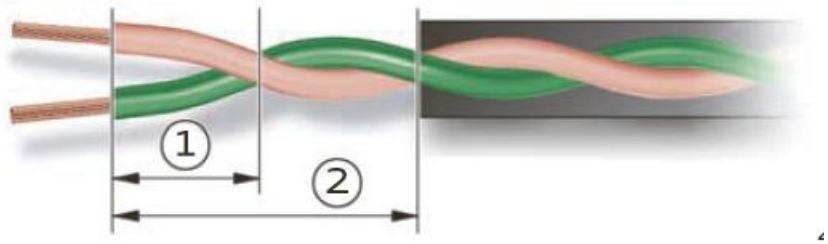


Fig. 3.43. FlexRay Bus Cable Repair [7].

Compliance with these dimensional requirements is critical to maintaining the impedance characteristics of the cable and ensuring the signal integrity necessary for high-speed deterministic communication [7].

The main characteristics of the data buses under consideration are summarised in Table 3.6.

Table 3.6 Characteristics of Data Transmission Buses

Type of bus	Data transmission medium	Transmission distance	Number of devices	Transmission rate
K-bus	single-wire	not specified	2	1.2–10.4 kbit/s
CAN	two-wire	several hundred metres	> 100	10 kbit/s – 1 Mbit/s
LIN	single-wire	up to 40 m	max. 16	1–20 kbit/s
MOST	optical fibre	not specified	max. 64	25 Mbit/s
Bluetooth	radio waves	several metres	max. 260	700 kbit/s
FlexRay	two-wire	up to 40 m	max. 128	10 Mbit/s

3.8. Summary

This chapter offers a comprehensive examination of digital control systems in modern vehicles, with particular attention to in-vehicle network technologies. It situates the automobile as a highly complex, multi-parameter system requiring the coordinated operation of numerous electronic control units (ECUs), sensors, and actuators. The central argument is that efficient and reliable

communication between these subsystems is indispensable for achieving safety, comfort, fuel efficiency, and reduced emissions, thereby making the evolution of digital bus technologies a cornerstone of contemporary automotive engineering.

The discussion begins with an historical overview of in-vehicle communication, tracing the transition from direct analogue wiring to pulse-width modulation and, subsequently, to digital transmission. This evolution reflects the growing need to handle larger data volumes while minimising wiring complexity, interference, and energy losses. The authors emphasise that the adoption of bus-based communication architectures represents a paradigm shift, where shared transmission media, regulated by protocols, replace point-to-point wiring. The chapter outlines the classification of bus systems by transmission speed and reliability, noting their varying suitability for diagnostic, comfort, safety-critical, and multimedia applications.

A detailed analysis of specific communication protocols follows. The K-Line system is introduced as the earliest standardised bus, designed primarily for diagnostics, notable for its simplicity and low cost but limited by low data rates. By contrast, the Controller Area Network (CAN), developed by Bosch, is presented as the industry standard for robust, high-speed, and fault-tolerant communication, particularly in safety-critical powertrain and chassis systems. The chapter offers a clear account of CAN's arbitration mechanisms, error detection capabilities, and data protocol structure, underlining its enduring relevance and cost-effectiveness.

The Local Interconnect Network (LIN) is described as a complementary, low-cost solution for peripheral subsystems such as window lifters and climate control. With its master–slave topology, single-wire physical layer, and modest data rates, LIN is positioned as a pragmatic alternative for non-safety-critical applications. In contrast, Media Oriented Systems Transport (MOST) represents a high-capacity, fibre-optic solution for multimedia and infotainment networks. Its ring topology, synchronised frame transmission, and optical immunity to electromagnetic interference allow efficient handling of audio and video data.

The chapter then addresses Bluetooth as a wireless standard, highlighting its short-range, ad hoc networking capabilities, frequency-hopping spread spectrum technique, and robust security protocols. While offering flexibility, it is limited in reliability and throughput compared to wired alternatives. Finally, the FlexRay protocol is examined as a next-generation technology, designed to meet demands for deterministic, high-speed, real-time communication in advanced driver assistance and vehicle dynamics control. Its use of time-triggered scheduling, fault isolation, and hybrid topologies is presented as a decisive advance over event-driven systems such as CAN.

The chapter concludes with a comparative synthesis, summarising the capabilities, limitations, and application domains of each bus technology. Collectively, these systems form a layered communication hierarchy in which diagnostic, comfort, infotainment, and safety-critical subsystems are integrated into a coherent vehicular network. The analysis underscores the essential role of digital control systems in ensuring functional integration, safety, and efficiency in modern automotive design, while also providing a clear conceptual framework for understanding the interplay between different bus technologies.

References

1. Rajesh R., Thomas M. On-Board Diagnostics (OBD-II) Data Bus Analysis and Applications for Automotive Systems. IEEE Access, 2021. Vol. 9. P. 117356–117369. DOI:10.1109/ACCESS.2021.3108732.
2. Etschberger K. Controller Area Network and K-Line Diagnostics in Modern Vehicles. SAE Technical Paper Series, 2020. DOI:10.4271/2020-01-5032.
3. Miller J., Smith D., Zhang Y. Next-Generation In-Vehicle Networking: CAN FD, Automotive Ethernet, and Diagnostics Integration. IEEE Vehicular Technology Magazine, 2022. Vol. 17, No. 2. P. 45–55. DOI:10.1109/MVT.2022.3145789.

4. 4. VAG. VAG SSP 238 – Data Exchange On The CAN Bus I Basics [online]. Available at: https://procarmanuals.com/pdf-online-vag-ssp-238-data-exchange-can-bus-basics/ [Accessed 10 Aug. 2025].
5. Etschberger K. Controller Area Network: Basics, Protocols, Chips and Applications. 2nd ed. Wiesbaden: Hüthig, 2019. 430 p.
6. VAG SSP 286 – New data bus systems – LIN, MOST, Bluetooth [online]. Available at: https://procarmanuals.com/pdf-online-vag-ssp-286-new-data-bus-systems-lin-bluetooth/ [Accessed 15 Aug. 2025].
7. VAG SSP 459 – Audi A8 2010 Electrical and network systems [online]. Available at: https://procarmanuals.com/pdf-online-vag-ssp-459-audi-a8-2010-electrical-network-systems/?__cf_chl_rt_tk=43zf.eobjgYeq0BGC7rf2p4YbldZpj5GoTTDBPninNY-1754820602-1.0.1.1-seole9fxQiXM0LjeThs_oflx579tBWHvAocC02noi24 [Accessed 25 Aug. 2025].

Chapter 4. Photovoltaic converter digital control systems

4.1. Photovoltaic converters

Photovoltaic converters (PVCs), better known as solar panels, are actively used today. The use of PECs allows to reduce the amount of electrical energy consumed from the power grid, which in turn reduces the negative impact of the power industry on the ecological state of the environment [1]. The experience of different countries indicates the possibility of large-scale application of PVCs through the creation of solar power plants.

The efficiency of a PVC within 5-20 % but, for example, the efficiency of a thermal power plant is about 60%. In fact, this means that, in order to generate equivalent electrical power, a solar power plant must occupy a much larger surface area. Another significant disadvantage is that the level of electrical power generation by a PVC is almost completely determined by the meteorological conditions of the location: the density of the incident light flux from the Sun and the ambient temperature [1].

The generation of maximum electrical power of PVC is possible only under certain conditions: for each value of the density of the incident light flux, the PVC load resistance must be a fixed value (the so-called maximum power point). Therefore, today there is a certain class of Maximum Power Point Tracking (MPPT) systems which provide the maximum power point for the PVC depending on the ambient conditions. It is obvious that MPPT is one of the most important subtasks of any control system at a solar power plant.

The process of converting solar energy into electricity using a PVC is based on the photoelectric effect that occurs in inhomogeneous semiconductor structures under the influence of solar radiation. The conversion efficiency depends on the electrophysical characteristics of the heterogeneous semiconductor structure, as well as the optical properties of the PVC among which the most important role has a photoconductivity as an internal photo effect in semiconductors under their irradiation with sunlight [2].

The appearance of a PVC based on single-crystal silicon is shown below, in Fig. 4.1. The most efficient mode of their operation is achieved under the condition of constant environmental parameters (i.e., complete or partial darkness worsens its characteristics). The indisputable advantages include the highest efficiency of converting light energy into electrical energy, characterized by an efficiency of 15–20% [3].

Fig. 4.1. Appearance of a single-crystal silicon-based PVC [1].

The appearance of a polycrystalline silicon-based converter is shown below, in Fig. 4.2. Such converters have a lower average cost, since the complexity of manufacturing crystals is lower here. Complete or partial darkening of the PEP also reduces the output power, but this is less pronounced than in the previous type. They are characterized by an average conversion efficiency: they have an efficiency of 10 - 14% [3].

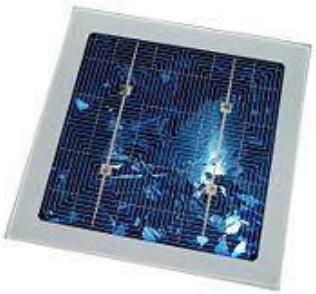


Fig. 4.2. Appearance of a polycrystalline silicon-based PVC [1].

The appearance of a converter based on amorphous silicon is shown below, in Fig. 4.3. Such converters have the lowest cost, since the complexity of obtaining amorphous silicon is low. Thanks to this structure of silicon, it is possible to manufacture a converter with a large light absorption area, as well as flexible elements. Complete or partial obscuration of the PEP has almost no effect on its characteristics - this type of converter is the most stable to such fluctuations, but it is characterized by the lowest efficiency, at the level of 5 - 6% [3].

Fig. 4.3. Appearance of a PVC based on amorphous silicon [1].

Given the advantages and disadvantages of each of the considered types of PEDs, it is easy to understand that their areas of application will also differ.

4.2. Modeling of PVC

To estimate the parameters and characteristics of a photovoltaic device, it is necessary to have its model. Any photovoltaic device is a number of photovoltaic (PV) cells connected together in a certain sequence: in order to increase the efficiency. It should also be taken into account that the dependences between the state parameters of the semiconductor material are not linear [4].

The equivalent electrical circuit based on the general mathematical model of a PV cell, is shown in Fig. 4.4.

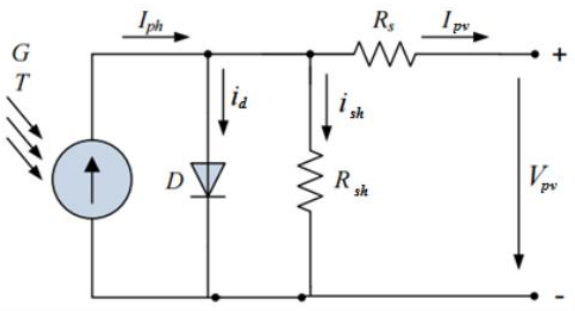


Fig. 4.4. General equivalent circuit of a PV cell [4].

This scheme corresponds to the general mathematical model of the volt-ampere characteristic of a PV cell:

$$I = I_{PH} - I_S \cdot e^{\frac{q\cdot(V+I\cdot R_S)}{k\cdot T_C\cdot A}-1} - \frac{V+I\cdot R_S}{R_{SH}}, \quad (4.1)$$

where I_{PH}– the magnitude of the current generated under the influence of solar radiation at a certain temperature (approximately 5 A);

q– the charge of the electron (1.6 10-19 C);

k– the Boltzmann constant (1.38 10-23 J/K);

T_C– the working temperature of the cell (usually 20 ºC or 293 K);

R_{SH}– the magnitude of the shunt resistance;

R_S– the magnitude of the series resistance (approximately 0.001 Ohm);

I_S– the reverse saturation current of the diode (approximately 0.0002 A);

A– the idealization coefficient, which shows how much the values obtained by modeling correspond to the actually measured ones.

In practice, a typical equivalent electrical circuit is widely used [4], which is shown in Fig. 4.5.

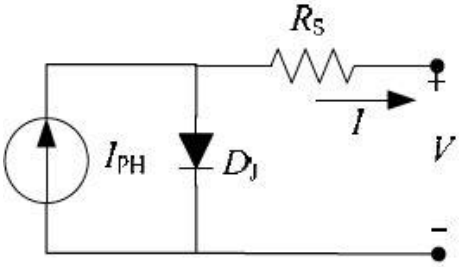


Fig. 4.5. Typical equivalent circuit of a PV cell [4].

The volt-ampere characteristic of such circuit is determined by the formula:

$$I = I_{PH} - I_S \cdot e^{\frac{q\cdot(V+I\cdot R_S)}{k\cdot T_C\cdot A}-1}. \quad (4.2)$$

The use of a typical equivalent electrical circuit of a PV cell for the purpose of further construction and modeling of a maximum power point tracking system is acceptable since an equivalent circuit provides sufficient accuracy.

4.3. Maximum power point of PVC

The maximum power point of a PVC is a point on the volt-ampere characteristic where the derivative of the power by voltage is zero (at this point the power stops increasing, but does not decrease yet), i.e. [5]:

$$\frac{dp}{du} = 0, \quad (4.3)$$

where *p* is the instantaneous value of the output power of the PVC.

The condition given by the formula (4.3) is equivalent to the following

$$\frac{di}{du} = -\frac{i}{u}, \quad (4.4)$$

where i is the instantaneous value of the PEP output current;
u is the instantaneous value of the PEP output voltage.

Figure 4.6 shows the combined characteristic with the point of maximum power.

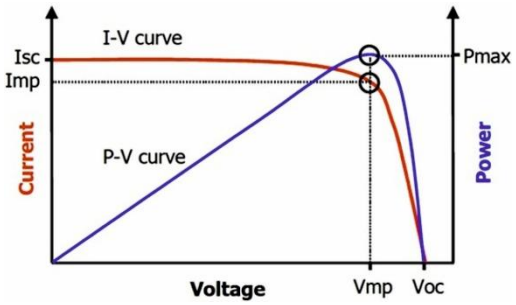


Fig. 4.6. Combined characteristic of the PVC [3].

The coordinates of the point of maximum power of the PVC are most significantly influenced by:

- working temperature of the PVC;
- density of the incident solar radiation flux.

Since the PVC still does not have a sufficiently high efficiency, the task of obtaining the maximum possible power is very relevant.

The following PVC control methods are distinguished:

- the method of tracking the coordinates of the maximum power point based on voltage feedback;
- the method of tracking the coordinates of the maximum power point based on current feedback;
- the method of tracking the coordinates of the maximum power point based on observations of power fluctuations.

4.4. Fuzzy control system for MPPT

4.4.1. MPPT control system architecture

Based on analyse given in [1 – 6], it can be concluded that the most optimal option for the implementation of the PVC maximum power point tracking system is the use of a fuzzy logic device. It allows to operate the object in conditions of insufficient amount of information.

In fact, the maximum power point tracking system under development is a MPPT control system where PVC acts as the control object. Proposed architecture of the MPPT control system is shown in Fig. 4.7.

Achieving the point of maximum electrical power of PVC at a given “Operating temperature” (“Temperature”) and “Density of the incident solar radiation” (“Solar”) linguistic variables is ensured by selecting the value of the “Load resistance” (“Resistance”) logistic variable of the fuzzy regulator. So, as a result of performing all fuzzy operations, the optimal value of the load resistance will appear at the output of the module.

Adjustment of the fuzzy control to the optimal functioning can be carried out by training an Artificial Neural Network (ANN) with the appropriate structure. Based on the analysis of the obtained curves, it is possible to calculate the value of the load resistance at which the power will be maximum.

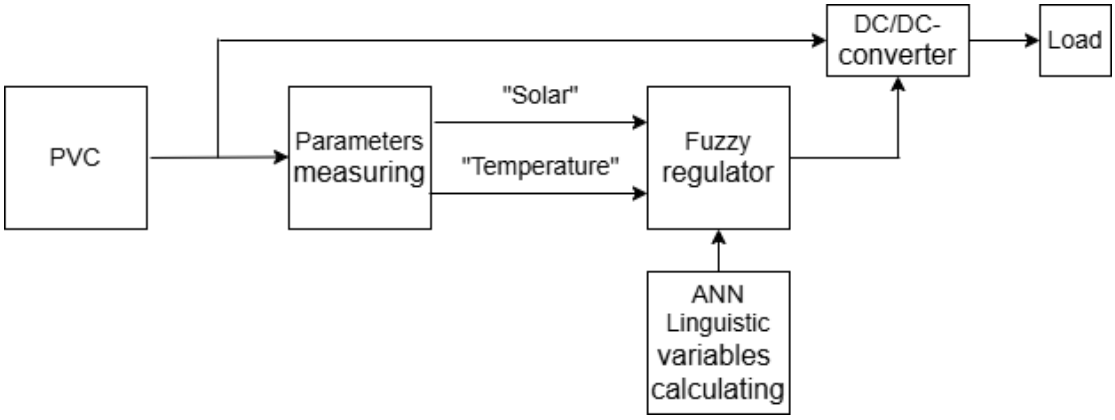


Fig. 4.7. MPPT fuzzy control system architecture

4.4.2. Fuzzy regulator

The central element of fuzzy control system is a fuzzy regulator. Figure 4.8 shows its typical structure [7, 8]. It consists of the following components: fuzzification block, rule base, logical inference block, and defuzzification block.

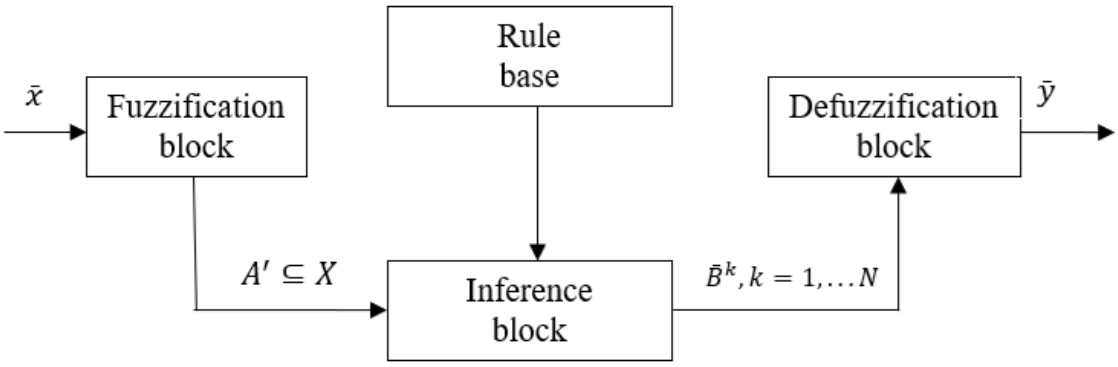


Fig. 4.8. Typical structure of the fuzzy regulator [7].

Since the fuzzy logic control system operates with fuzzy sets, the specific value of the input signal of the fuzzy regulator is subject to the fuzzification operation, as a result of which the input clear value will be assigned a pair of values of the form:

$$A = \{(x, \mu_A(x)); x \in X\}, \quad (4.5)$$

where $\mu_A(x)$ is the membership function that quantifies the grade of membership of the element *x* to the fuzzy set *A*, $\mu_A(x)$: $X \rightarrow [0,1]$.

The Fig. 4.9 demonstrates some examples of fuzzy set memberships.

The task of the Inference block is to transform the input fuzzy set into the corresponding output fuzzy set, such a transformation is carried out using fuzzy logic methods. As a result of the transformation the set of N fuzzy sets $\bar{B}^k$ with membership functions $\mu_{\bar{B}^k}(y)$ is formed as the output of the block.

The rule base is a set of fuzzy rules *R(k) (k = 1, ..., N)* of the form:

$$IF\left(x_1 \; is \; {A_1}^{(k)} AND \; x_2 \; is \; {A_2}^{(k)}\right) THEN \; (y_1 \; is \; {B_1}^{(k)} \; AND \; y_2 \; is \; {B_2}^{(k)}), \quad (4.6)$$

where x_i, y_i are the input variables;

$A_i^{(k)}$, $B_i^{(k)}$ are fuzzy sets;
N is the total number of fuzzy rules.

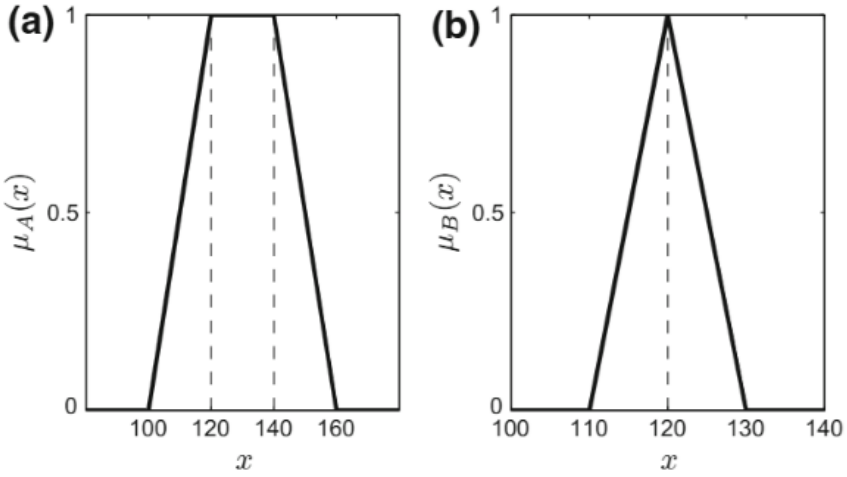


Fig. 4.9. Examples of membership functions: a – trapezoidal, b – triangular [7]

In the MPPT control system considering the nonlinearity of the control process, which is due to the influence of external factors, instead of the general approach it is advisable to use the Takagi-Sugeno (T-S) fuzzy algorithm [8]. The rules used in this algorithm are fuzzy only in the IF part, while in the THEN part there are functional dependencies:

$$IF\left(x_1\ is\ A_1{}^{(k)} AND\ x_2\ is\ A_2{}^{(k)}\right)\ THEN\ y_1 = f^{(k)}(x_1, x_2), \tag{4.7}$$

where $f^{(k)}$ – ordinary non-fuzzy function.

If a signal is applied to the input of T-S fuzzy regulator then the output signal is a normalized weighted sum of individual outputs, which is calculated by the formula

$$\bar{y} = \frac{\sum_{k=1}^{N} w^k \bar{y}^k}{\sum_{k=1}^{N} w^k}, \tag{4.8}$$

The normalized weighted sum allows to see which output, other than 0, gives the largest value.

The task of the defuzzification block is to map fuzzy sets $\bar{B}^k$ into a single value y, which represents the control influence applied to the input of the control object. In T-S fuzzy algorithm, as rule, defuzzification occurs using the center of gravity method, which corresponds to the expression (4.8). The essence of the method is illustrated in Fig. 4.10.

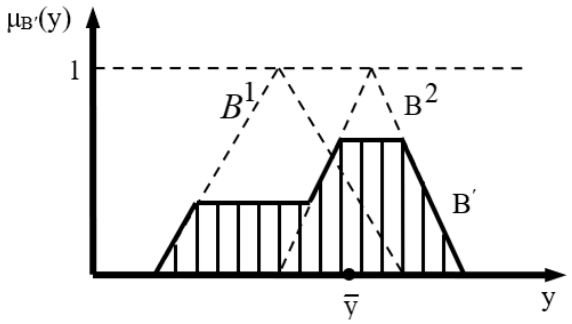


Fig. 4.10. Illustration of defuzzification using the centre of gravity method [7].

4.5. Simulation of MPPT fuzzy control system

Simulation of MPPT fuzzy control system can be performed with the MATLAB Fuzzy Logic Designer tool that provides the configuring fuzzy systems T-S type control [9 – 12]. Simulation process consists of the following consecutive stage:

- modeling of linguistic variables;
- forming of rule base;
- modeling of the PVC control surface;
- tuning of linguistic variables using ANN;
- creating a model of the MPPT control system;
- simulation experiments with the model.

4.5.1. Modeling of linguistic variables

linguistic variables in Zadeh definition [13] are “variables whose values are not numbers but words or sentences in a natural or artificial language”. In formal definition linguistic variable corresponds to the tuple

$$L = (X, U, T, M), \tag{4.9}$$

where X – the name of the linguistic variable;

U – the actual physical domain of the X linguistic variable values;

T – the set of linguistic values (terms);

M – semantic rule that relates each linguistic values in T with a fuzzy set in U.

Fuzzy controller of MPPT control system receives in the input the values of two linguistic variables: "Density of incident solar radiation" (“Solar”) and " Operating temperature” (“Temperature").

Given the nonlinear nature of the MPPT control system behaviour, the Gaussian function should be used to model membership functions instead of simpler trapezoidal or triangular functions. The structure of the input linguistic variables in the MATLAB representation is shown in Figures 4.11 and 4.12, and the parameters of the fuzzy membership functions are in Table 4.1.

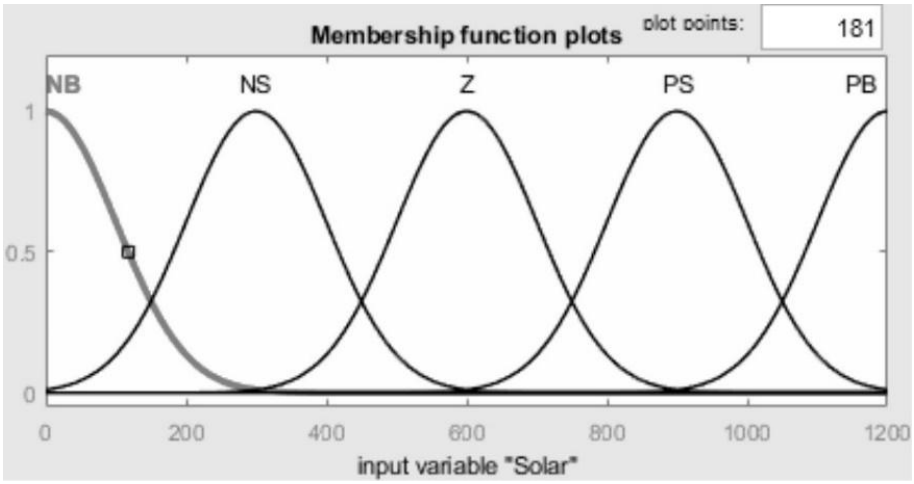


Fig. 4.11. Input linguistic variable "Solar "

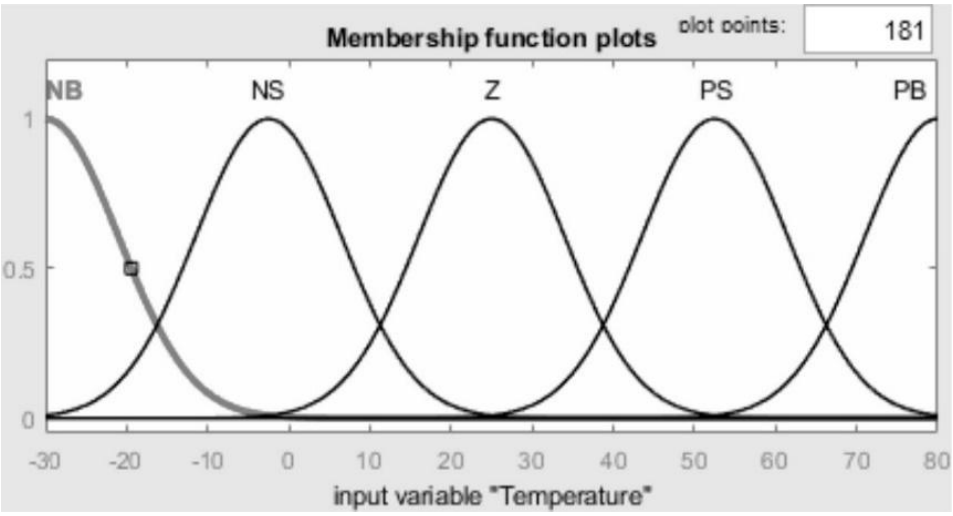


Fig. 4.12. Input linguistic variable "Temperature" of the second fuzzy control system

Table 4.1 Parameters of fuzzy membership functions of input linguistic variables

Linguistic variable name	Linguistic values	Linguistic variable function domain parameters (standard deviation estimation)
"Solar"	NB	[100 0]
	NS	[100 300]
	Z	[100 600]
	PS	[100 900]
	PB	[100 1200]
"Temperature"	NB	[9 -30]
	NS	[9 -2.5]
	Z	[9 25]
	PS	[9 52.5]
	PB	[9 80]

The range of permissible values of the output linguistic variable "Load resistance" ("Resistance") is determined solely by which PVC was chosen. For most high-power PVCs (more than 200 W), it is in the range of [1 ... 40] Ohms. The structure of this variable based on T-S inference algorithm is shown in Fig. 4.13. This linguistic variable consists of 25 fuzzy constant membership functions.

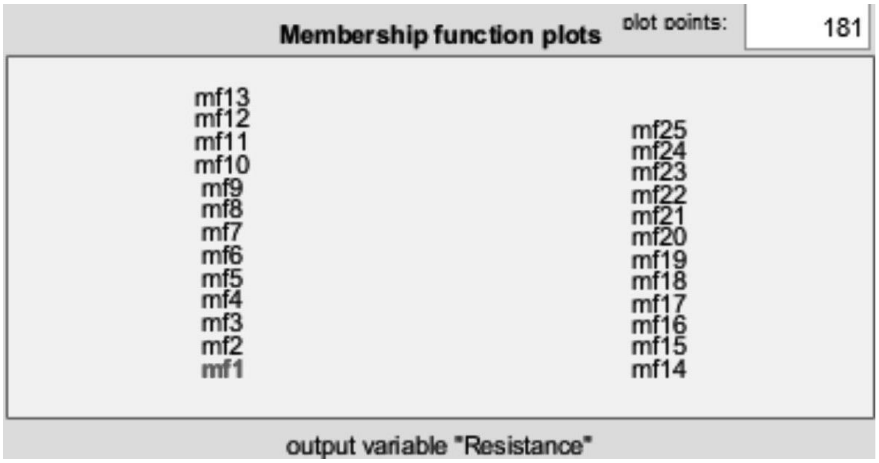


Fig. 4.13. The structure of the output linguistic variable "Resistance"

4.5.2. Forming of rule base

The table of fuzzy rules of the control module must contain a number of rules that would cover all combinations of all terms of linguistic variables. In our case such rules must be created 25 rules, since each of the two input linguistic variables has 5 membership functions. Presentation of fuzzy rules developed using MATLAB Fuzzy Logic Toolbox is given in Fig. 4.14.

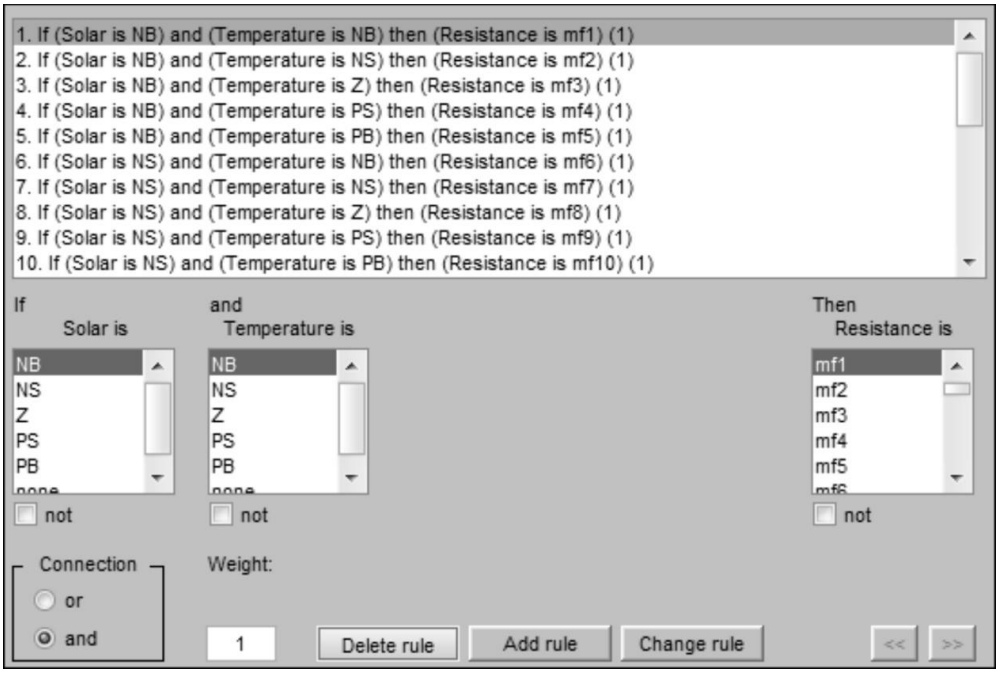


Fig. 4.14. Presentation of the rule base in Fuzzy Logic Toolbox

This model uses a modification of the rule (4.7) where the conclusion is represented as a constant:

$$IF\left(x_1 \; is \; {A_1}^k \; AND \; x_2 \; is \; {A_2}^k\right) THEN \; y = c^{(k)}. \qquad (4.10)$$

Table 4.2 contains the content of the developed rule base, where at the intersection of the selected row and column there is the number of the corresponding constant (*mfi*, where *i* = 1, 2, 3, ..., 25). The weight of each generated fuzzy rule equal 1 by default.

Table 4.2 Contents of the knowledge base for the fuzzy control system

Membership functions of the variable "Solar"	Membership functions of the variable "Temperature"				
	NB	**NS**	**Z**	**PS**	**PB**
NB	mf1	mf2	mf3	mf4	mf5
NS	mf6	mf7	mf8	mf9	mf10
Z	mf11	mf12	mf13	mf14	mf15
PS	mf16	mf17	mf18	mf19	mf20
PB	mf21	mf22	mf23	mf24	mf25

It should be noted that the initial selection of the form and parameters of membership functions for linguistic variables must be adjusted with ANN.

4.5.3. Modeling of the PVC control surface

One of the simplest ways to ensure the operation of the PVC power generation mode control device is to set the value of the output resistance for it based on an analytical expression. For this, it is necessary to obtain a dependence of the form:

$$R = f(S, t), \tag{4.11}$$

where R– PVC load resistance, for which the generation of the maximum possible power of electrical energy is ensured, Ohm;

S– solar radiation flux density falling on the PVC surface, W/m2;

t– working temperature of PVC, C.

If there is no possible to measure the initial parameters of PVC in nature, the model experiment can be used. Fig. 4.15 shows the example of experimental MATLAB model.

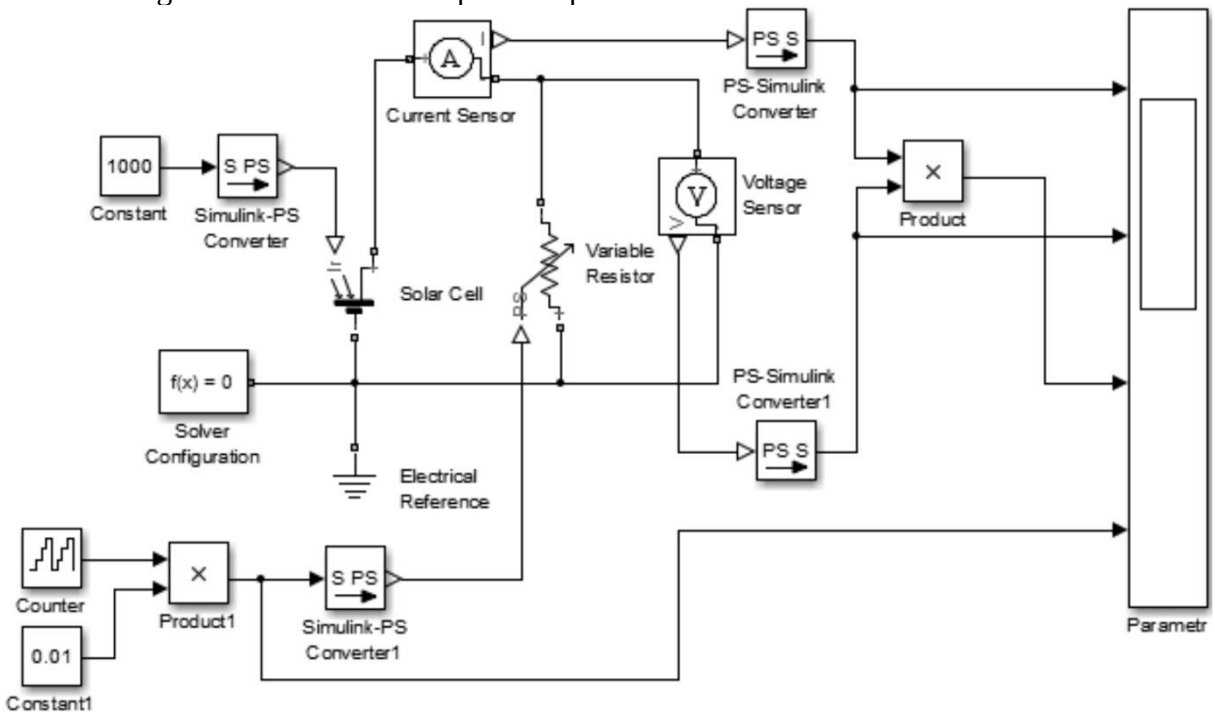


Fig. 4.15. Experimental model of the PVC for obtaining initial parameters.

Tables 4.3a and 4.3b contain the values of the optimal PVC load resistances for a fixed working temperature and density of the incident solar radiation.

Table 4.3a Optimum PVC load resistances (for flux density from 100 W/m2 to 600 W/m2)

Operating temperature, oC	Density of incident solar radiation, W/m2					
	100	200	300	400	500	600
-30	32.3000	16.1500	10.7400	8.0200	6.3850	5.2850
-20	33.5000	16.7400	11.1125	8.2850	6.5800	5.4400
-10	34.7100	17.3200	11.4750	8.5400	6.7750	5.5900
0	35.9500	17.8750	11.8250	8.7850	6.9500	5.7300
10	37.0900	18.4300	12.1600	9.0150	7.1200	5.8500
20	38.2500	18.9700	12.4850	9.2300	7.2700	5.9550
30	39.4200	19.4825	12.7950	9.4250	7.4100	6.0600
40	40.5400	19.9900	13.0750	9.6150	7.5250	6.1300
50	41.6550	20.4700	13.3500	9.7750	7.6300	6.1950

60	42.7550	20.9300	13.6050	9.9250	7.7050	6.2300
70	43.8300	21.3700	13.8250	10.0450	7.7650	6.2550
80	44.8700	21.7800	14.0250	10.1350	7.8050	6.2500

Table 4.3b Optimum PVC load resistances (for flux density from 700 W/m2 to 1200 W/m2)

Operating temperature, oC	Density of incident solar radiation, W/m2					
	700	800	900	1000	1100	1200
-30	4.5000	3.9100	3.4550	3.0900	2.7850	2.5350
-20	4.6300	4.0150	3.5400	3.1600	2.8450	2.5850
-10	4.7500	4.1150	3.6150	3.2200	2.8950	2.6250
0	4.8500	4.1950	3.6800	3.2700	2.9350	2.6550
10	4.9450	4.2650	3.7350	3.3100	2.9600	2.6700
20	5.0250	4.3200	3.7750	3.3350	2.9800	2.6800
30	5.0900	4.3650	3.7950	3.3450	2.9750	2.6700
40	5.1350	4.3900	3.8050	3.3450	2.9650	2.6500
50	5.1650	4.3950	3.8050	3.3250	2.9350	2.6150
60	5.1750	4.3850	3.7750	3.2950	2.9000	2.5800
70	5.1650	4.3650	3.7400	3.2450	2.8550	2.5500
80	5.1400	4.3200	3.6950	3.2050	2.8250	2.5450

Checking the adequacy of the proposed model by calculating the t-criterion shows: for a sample of 41 values, the t-criterion is 0.22 (and its critical value is 2.021), the confidence level is 0.827, and the confidence interval is from 5.058 to 10.327. That is, the proposed model is adequate and can be used to obtain initial data for the purpose of their further processing.

Data from Table 4.3 gives to get a dependency in form:

$$R_i = f(S)|t_i, \tag{4.12}$$

where R_i– load resistance of PVC, with a variable value of S and temperature t_i;

i– the number of the constructed function.

So, based on data in Table 4.3 a linearly interpolated experimental curve can be built using, for example, interpolation function *linterp()*.

The resulting graphs of interpolated experimental data for the given temperature are shown in Fig. 4.16.

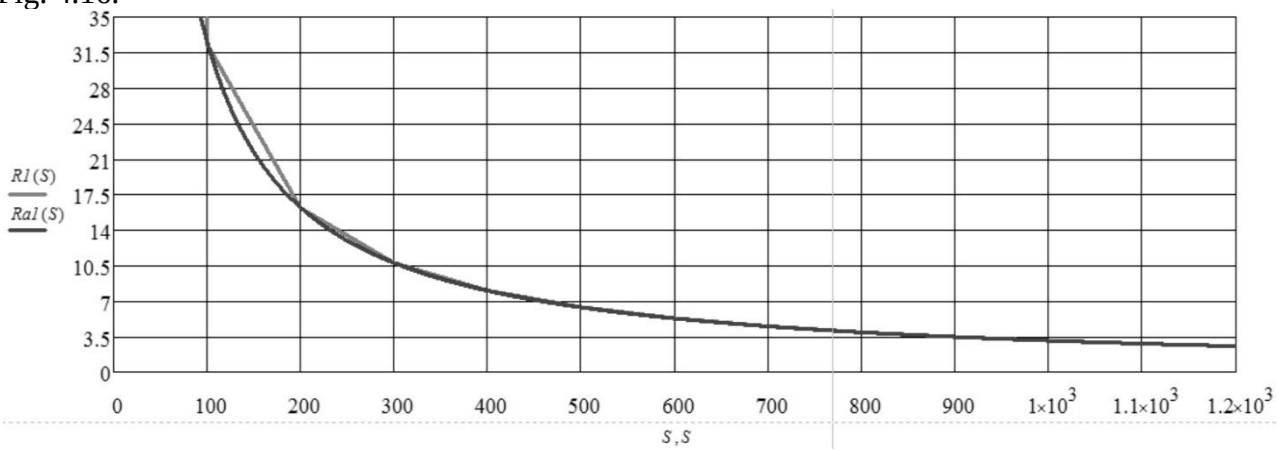


Fig. 4.16. Resulting approximation of dependence (4.12) at temp = -30 °C

The Fig. 4.16 demonstrates a hyperbolic view of the expression (4.12) in form:

$$\mathrm{Ra}_i = (K_i(S)/S)|t_i, \tag{4.13}$$

where Ra_i– approximation of the dependence of the PVC load resistance on the value of the variable S at a constant temperaturet_i;

$K_i(S)|t_i$- proposed correction function.

The coincidence of interpolation with approximation is ensured by using a correction function of the form:

$$K_i(S)|t_i = k \cdot S + c, \tag{4.14}$$

where k– slope coefficient of the straight line;

c is a free member that determines the constant component.

After performing a number of experiments, it was established that the most optimal and accurate dependence of the coefficient$k$from the temperature can be approximated by a second-order polynomial of the form:

$$\mathrm{ka}(t) = 0.0396 \cdot t^2 + 6.0269 \cdot t + 277.1614. \tag{4.15}$$

Therefore, the expression (4.15) is the general form of the approximating polynomial dependence.

Using similar considerations when searching for an approximating polynomial for the dependence $c(t)$:

$$ca(t) = \frac{880.7}{70} \cdot t + 3663.743. \tag{4.16}$$

The final expression can be obtained by performing some transformation for formula (4.13):

$$\mathrm{Ra}(S,t) = \frac{\mathrm{Ka}(S,t)}{S}. \tag{4.17}$$

Also, $K_a(S,t)$ can be found using the previously obtained formula:

$$\mathrm{Ka}(S,t) = \frac{\mathrm{ka}(t)}{700} \cdot S + \mathrm{ca}(t). \tag{4.18}$$

To increase accuracy other hyperbolic function can be used:

$$\mathrm{Ra}_i = \frac{b_i}{S^{K_i(S)}}|t_i, \tag{4.19}$$

where $K_i(S)|t_i$– proposed corrective function;

b_i– some constant value;

i– the number of the constructed curve.

The correction function $K_i(S)$ is a polynomial:

$$K_i(S)|t_i = \sum_{n=0}^{m} a_n \cdot S^n. \tag{4.20}$$

When the degree of polynomial equal to 1 the correction function has such form:

$$K_i(S)|t_i = a_1 \cdot S + a_0, \tag{4.21}$$

In this case, the approximation function for some value of the working temperature of PVC will take the form:

$$\mathrm{Ra}_i = \frac{b_i}{S^{a_1 \cdot S + a_0}}|t_i, \tag{4.22}$$

Thus, the correction function for each of the twelve experimental curves will be approximated using a linear function of the form (4.21). Coefficients a_1 and a_0 are calculated on data Table 4.3. The value of a constant value b_i. Selection of the required value x_i is performed manually, and the fitting process ends when the value of the approximation function completely coincides with the interpolated dependence curve $R_i = f(S)|t_i$.

The final expression will look like this:

$$R(S,t) = Ra(S,t) = \frac{ba(t)}{S^{a_1 a(t) \cdot S + a_0 a(t)}}. \tag{4.23}$$

The general view of the control surface obtained by approximation according to the expression (4.23) is shown in Fig. 4.17.

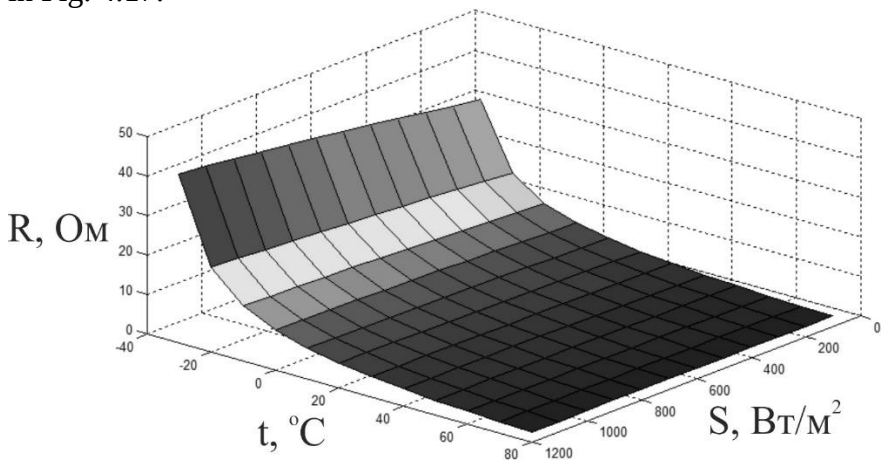


Fig. 4.17. The general view of the control surface.

The analysis of the expression of the form (4.23) shows that at all permissible values of the working temperature of the PVC, the relative error does not exceed the value of 1.6% for the density of the flux of incident solar radiation in the range of reasonable use (300 - 1000 W/m2). This is a much better indicator than provided by the approximation (4.13) that more 2%.

4.6. Tuning of linguistic variables using ANN

After two fuzzy PVC maximum power point coordinate tracking systems were developed and initialized, it is necessary to tune them for optimal functioning. This can be achieved with the help of ANN. The training sample for ANN can be created based on mathematical model of the control surface. The ANN learning process carry out using a set of tools Neuro-Fuzzy Designer included in MATLAB.

4.6.1. ANN learning process

The Neuro-Fuzzy Designer toolkit requires the formation of a sample for ANN training, which should be presented in the form of three columns separated by tabs. Each of the columns contains a corresponding value that a specific linguistic variable can take. The order of the columns from left to right corresponds to the following order of linguistic variables is as follows: “Solar” → “Temperature” → “Resistance”.

Using the expression (4.23), and stored training sample in a separate file (extension *.dat), it ia needed to run m-file which receives numerical values for the training sample. The text of m-file script is given bellow:

```
S= 100:100:1200;      % This is S - irradiance
T= -30:10:80;% This is T - temperature
fid = fopen('c:\altera\Training\Mynet\Polinom_1\Learning_by_S_and_T_P1.dat','wt');
for i = 1:numel(S)
        for j= 1:numel(T)
                chisl= -9.1168e-005*T(j)^3-0.0538*T(j)^2+6.1233*T(j)+3356.9;
                a=      1.7288e-012*T(j)^3+2.7858e-009*T(j)^2+3.6226e-007*T(j)+1.9018e-
005;
                b= -2.5943e-006*T(j)^2 -2.7593e-004*T(j)+0.9852;
                Z(i,j)= chisl/S(i)^(a*S(i)+b);
                Q= [S(i), T(j), Z(i,j)];
```

```
            dlmwrite('c:\altera\Training\Mynet\Polinom_1\Learning_by_S_and_T_P1.dat', Q,
'delimiter', '\t', 'precision', 9, '-append');
            end
        end
        fclose('all');
```

As a result of the execution of the script, a file with a training sample with a capacity of 144 lines was obtained.

After starting Neuro-Fuzzy Designer and loading the structure of the developed fuzzy systems (a file with the extension *.fis), an ANN with the corresponding structure was automatically created (Fig. 4.18). Note: the structure of such a network does not depend on the type of membership functions.

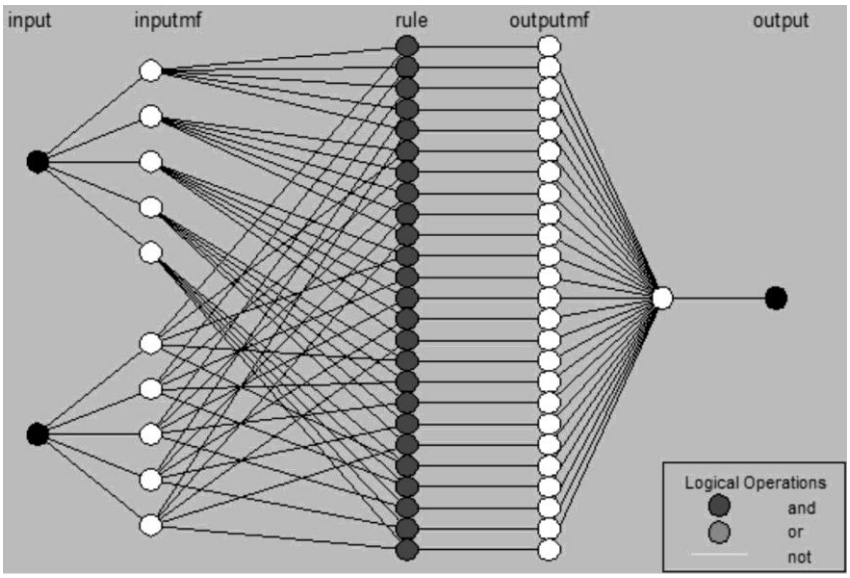


Fig. 4.18. The structure of ANN for tuning of linguistic variables.

Then it is necessary to indicate which method will be used for ANN training (chosen through the Optim. Method drop-down menu): backpropagating error (backprop) or hybrid (hybrid). For example, it may be backpropagation.

Table 4.4 shows the results of the ANN training with membership functions of the Gaussian type with backpropagation method.

Table 4.4 ANN training results

Teaching method	Number of iterations	Relative learning error, %
Backpropagation of errors	1000	10,1762
	5000	0.45549
	10000	0.15853
	15000	0.12184
	20000	0.10543
	25000	0.095626
	30000	0.88986
	35000	0.84146
	40000	0.080436

Fig. 4.19 illustrates the ANN learning process with the most optimal option. The Epochs field determines the number of iterations during which the selected training method will be applied.

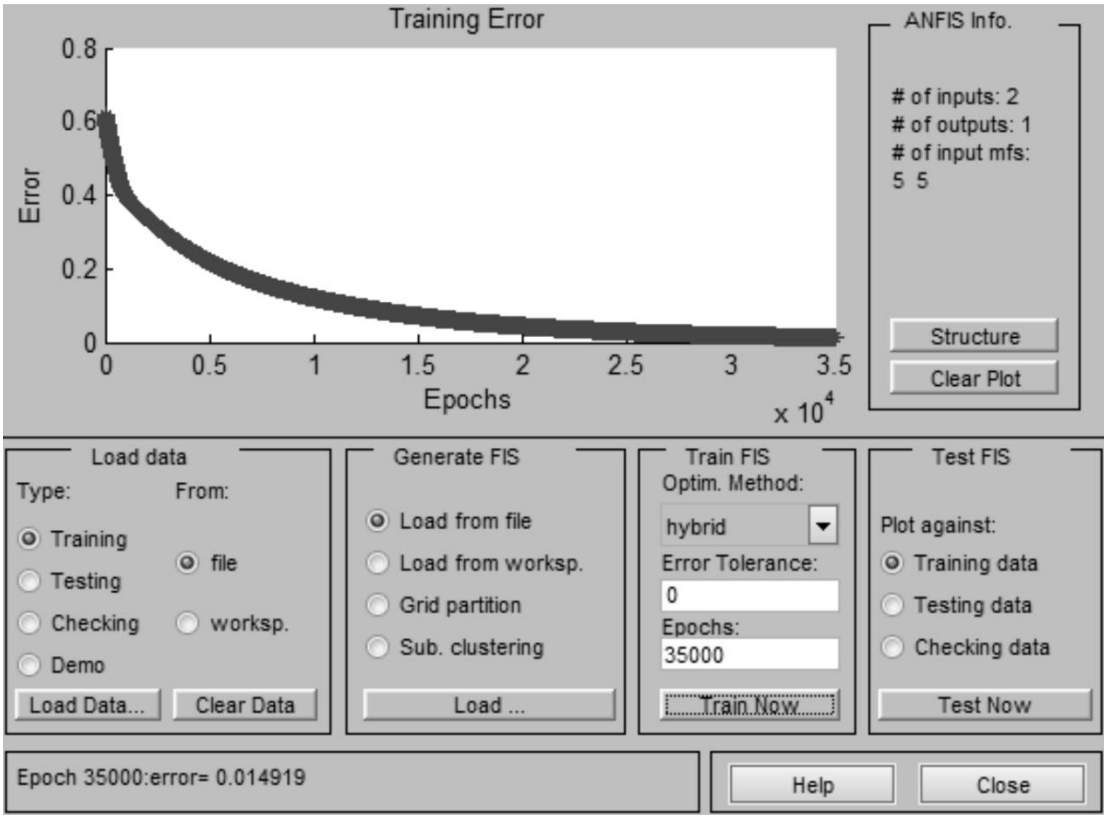


Fig. 4.19. ANN learning process

As a result, a fully configured fuzzy system for tracking the maximum power point of the PVC is obtained. Below, in Fig. 4.20 and 4.21 the general appearance of the fuzzy membership functions for the input linguistic variables of the configured system is depicted. Their parameters are given in Table 4.5.

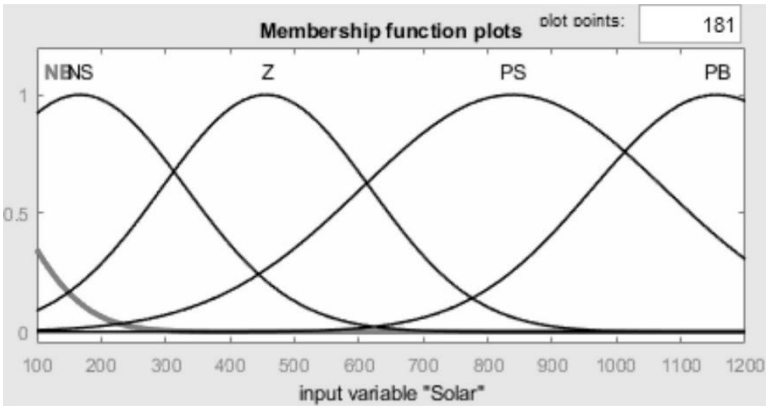


Fig. 4.20. The structure of the adjusted input linguistic variable “Solar”

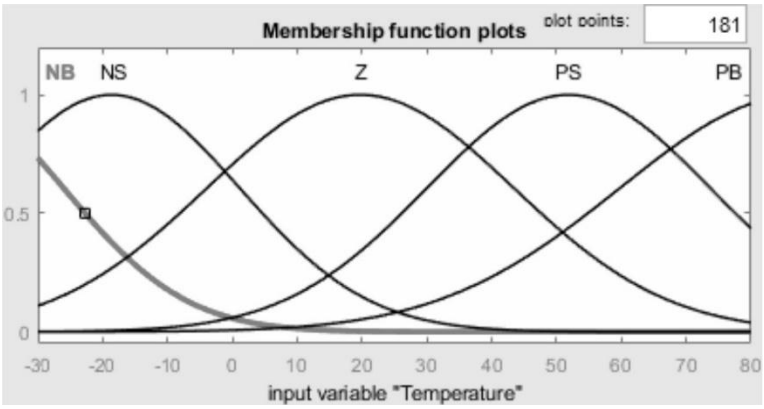


Fig. 4.21. The structure of the adjusted input linguistic variable "Temperature"

Table 4.5 Fuzzy membership function parameters for input linguistic variables after tuning

The name of the input variable	The name of the unclear function	Fuzzy function parameters (standard deviation estimation)
"Solar"	NB	[112.3 -63.38]
	NS	[164.7 167.1]
	Z	[164.7 167.1]
	PS	[234.6 839]
	PB	[192 1155]
"Temperature"	NB	[18.73 -44.76]
	NS	[19.74 -18.62]
	Z	[23.66 19.75]
	PS	[21.84 51.85]
	PB	[27.9 87.67]

The numerical values that make up the “Resistance” variable are listed below in Table 4.6. In fact, this is the load resistance for the selected PVC corresponding to the point of maximum power.

Table 4.6 Numerical values the linguistic variable “Resistance”

Names of constants	Numerical value, Ohm
mf1	78.54
mf2	91.23
mf3	97.82
mf4	108.2
mf5	117.1
mf6	13.62
mf7	15.78
mf8	16.94
mf9	18.78
mf10	20.44
mf11	5.621
mf12	6.429
mf13	6.796
mf14	7.359
mf15	7.479
mf16	3.722

Names of constants	Numerical value, Ohm
mf17	4.193
mf18	4.368
mf19	4.603
mf20	4.364
mf21	1.994
mf22	2.154
mf23	2.15
mf24	2.077
mf25	1.585

4.6.2. Creating a model of the MPPT control system

To evaluate the effectiveness of the developed and configured fuzzy system the most optimal way is simulation in MATLAB Simulink.

Using the Simulink shell component libraries, let's create two models: the first model of the maximum power point tracking system for the PVC, in which the optimal value of the load resistance under given conditions will be calculated based on the analytical expression (4.23) (see Fig. 4.22) and the second model of the system, in which the load resistance will be calculated using a tuned fuzzy system T-S type trained with ANN on Backpropagation method for 35,000 iterations (see Fig. 4.23).

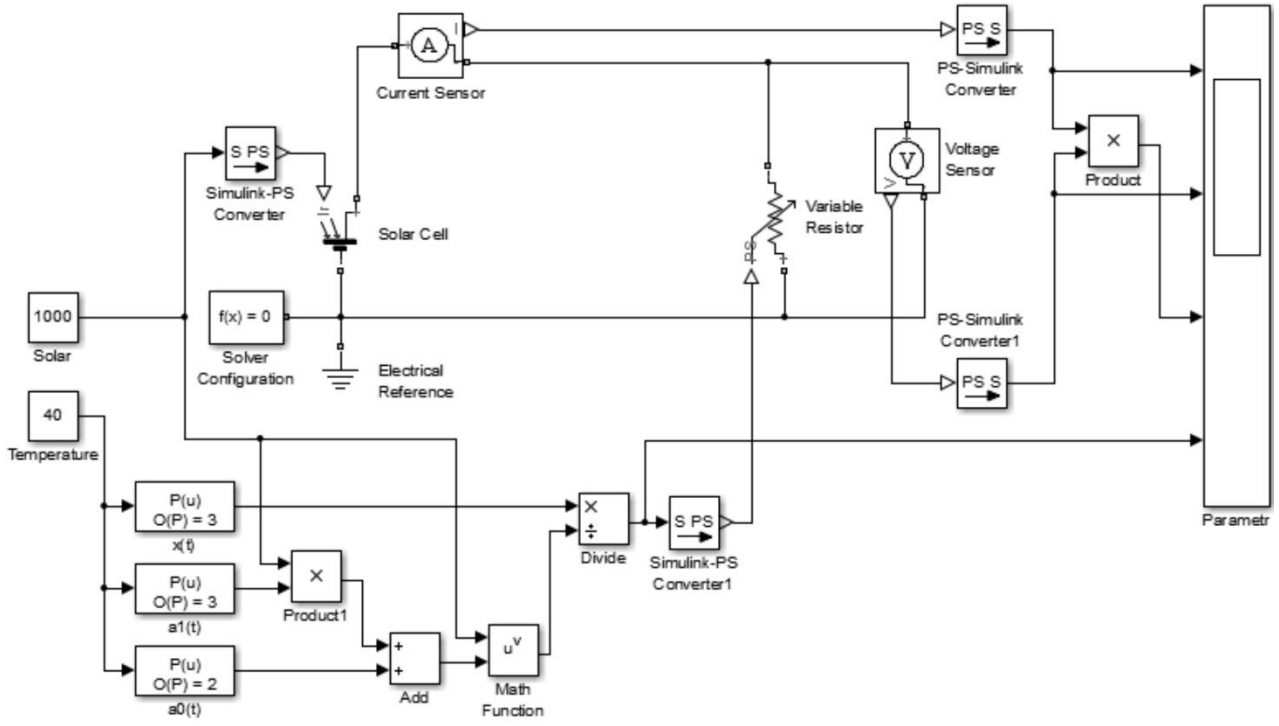


Fig. 4.22. The MATLAB model of the MPPT control system based on hyperbolic power approximation.

The elements that make up the simulation model given in Fig. 4.22 have the following purpose:

- Solar, Temperature – set the parameters of the surrounding environment: the density of the falling flux of solar radiation and the working temperature of PVC;

- x(t), a1(t), a2(t) – calculate the polynomial values of the numerator and denominator of the proposed hyperbolic power approximation at the given value of the working temperature of PVC;
- Simulink-PS Converter, Simulink-PS Converter1 – perform conversion of physical values of temperature and density of incident solar radiation into Simulink model parameters;
- Solver Configuration – sets initial conditions and constraints for simulation;
- Solar Cell – directly the model of the PVC;
- Product, Product1 – perform an arithmetic multiplication operation;
- Electrical Reference – electrical grounding, which is necessary for the correct calculation of voltages and currents generated by PVC;
- Add – performs an arithmetic addition operation;
- Current Sensor, Voltage Sensor – measure the instantaneous values of output current and output voltage of PVC, respectively;
- Math Function – performs the exponentiation operation to calculate the denominator of the selected hyperbolic power approximation;
- Divide – performs an arithmetic division operation;
- PS-Simulink Converter, PS-Simulink Converter1 – perform conversion of Simulink shell model parameters into physical values of working temperature and incident solar radiation density;
- Variable Resistor –performs the role of PVC load resistance to obtain the point of maximum power;
- Parameter – a virtual oscilloscope that displays the instantaneous values of the output current, voltage, PVC power and the value of the optimal load resistance at the given parameters of the surrounding environment.

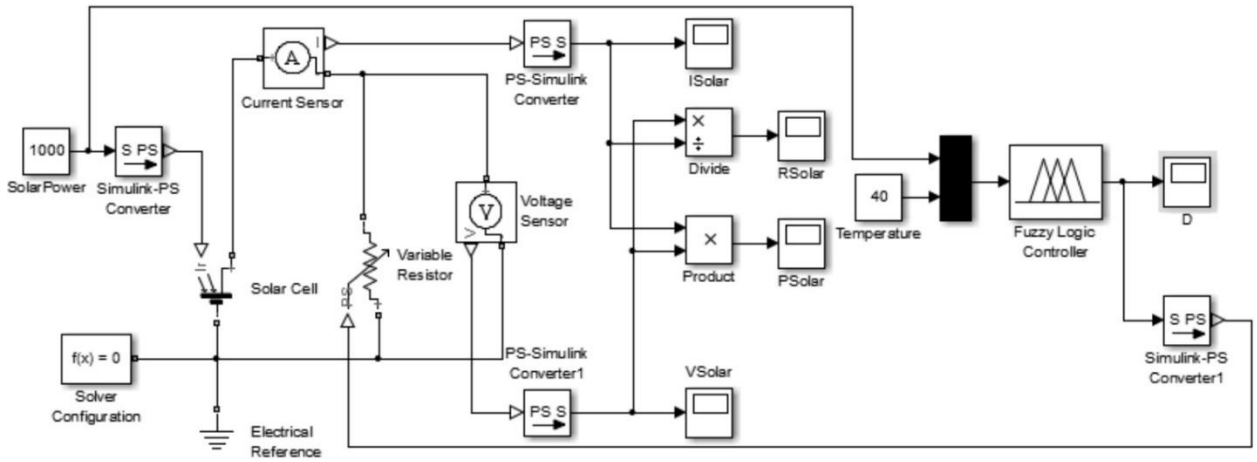


Fig. 4.23. The MATLAB model of the MPPT control system tuned fuzzy system T-S type trained with ANN on backpropagation method.

Most of the components of the simulation model shown above in Figure 4.23 are similar to the model in Figure 4.22. Only a few are different:

- *ISolar, VSolar, RSolar, PSolar, D* – virtual oscilloscopes that display the instantaneous values of the output current, voltage, current load resistance, output power of the PVC and the optimal load time at the given parameters of the environment, respectively (which was calculated by the neuro-fuzzy controller);

- *Fuzzy Logic Controller* – a model of a fuzzy controller, the setting of which was carried out by training the ANN using the training sample created on the basis of the obtained mathematical model of the control surface.

4.6.3. Simulation experiments with the model.

Now it is needed to simulate the operation of both created models and to estimate the relative error of the functioning of the developed fuzzy control system based on the obtained values of the load resistances,

Results of MPPT control system simulation with calculated relative error are given in Table 4.7.

Table 4.7 Results of MPPT control system simulation.

Density of the falling flux of solar radiation, S, Вт/м2	Working temperature of PVC, T, °C	Load resistance of the first model, R1, Ohm	Load resistance of the second model, R2, Ohm	Relative error,, ε, %
100	30	38,6481	38,6443	0,0098323074
200	30	19,2889	19,2903	0,0072580603
300	30	12,7207	12,7117	0,0707508235
400	30	9,3992	9,4161	0,1798025364
500	30	7,3898	7,362	0,3761942136
600	30	6,0418	6,0662	0,4038531563
700	30	5,0744	5,0678	0,1300646382
800	30	4,3465	4,3314	0,3474059588

In Table 4.7 the relative error between any corresponding values of load resistances was calculated using the formula:

$$\varepsilon_i = \frac{|R1_i - R2_i|}{R1_i} \cdot 100\%, \quad (4.24)$$

where *i* is the serial number of the calculation being performed.

Analysis of the results proves that the maximum relative error of calculating the load resistance of the PVC through hyperbolic power approximation and through the configured fuzzy system in terms of modulus does not exceed 0,4038% in the range of the PVC intended functioning.

Now, let's evaluate the effectiveness of the proposed fuzzy control system for tracking the coordinates of the point of maximum power of a photovoltaic cell by modeling. For this, another experimental model shown in Fig. 4.24 will be used. Note: it was created based on the model shown in Fig. 4.22, so the purpose of its components is similar.

Using the specified model, we first obtained numerical values of the maximum possible electric power generated by a photovoltaic cell providing a constant of load resistance for curtain constant working temperature. During the modeling the load resistance is 20 Ohms and does not change. This value was chosen arbitrarily. Same, let's choose arbitrary working temperature 30 °C. Also, simulate for this value of working temperatures, but for changing values of load resistance calculated using the model shown in Fig. 4.22. Results of simulation are given is shown in Table 4.8.

Based on the numerical data from Table 4.8 for greater clarity, below, the dependences of P1max (indicated by squares) and P2max (indicated by circles) on the density of the incident solar radiation flux, as well as the difference of these powers: P2max(S) – P1max(S) (indicated by triangles) are shown in Fig. 4.25.

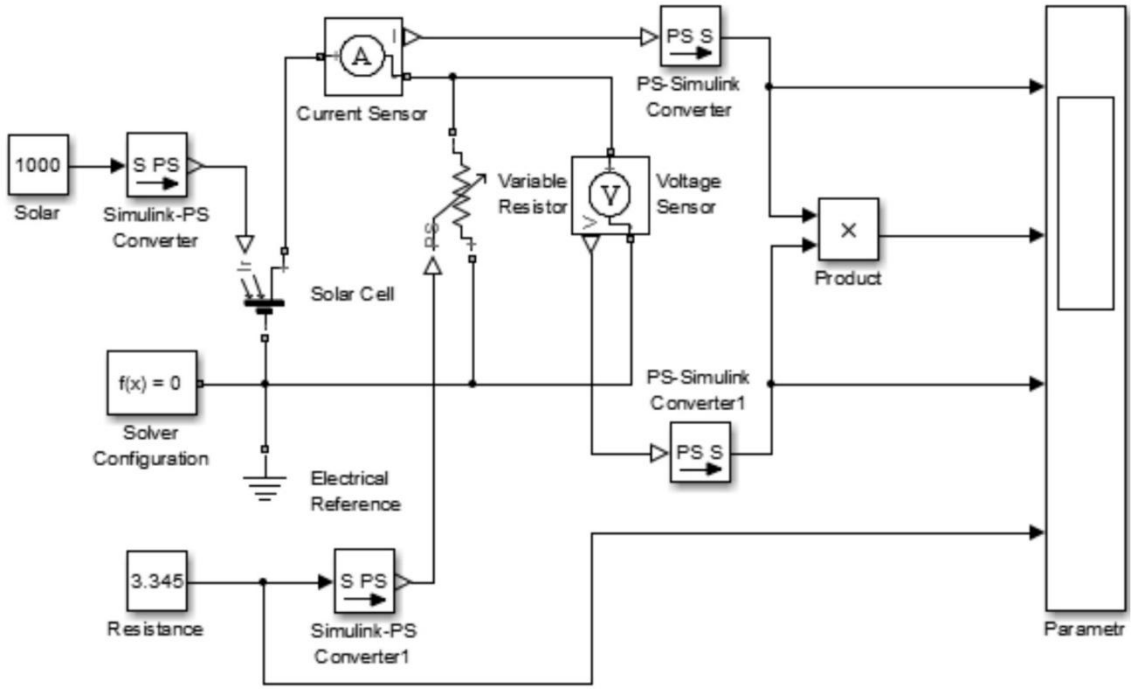


Fig. 4.24. Experimental model for checking the simulation results.

Table 4.8 The maximum electrical power at the working temperature 30 °C

Density of the falling flux of solar radiation, S, Вт/м2	Maximum electric power at constant load resistance P1max, W	Maximum electric power at variable load resistance P2max, W
100	15,3125	28,7842
200	56,4827	56,9783
300	60,5265	84,2063
400	62,6278	110,4151
500	62,2867	135,5203
600	62,7585	159,5380
700	63,1264	182,4586
800	63,4281	204,2480

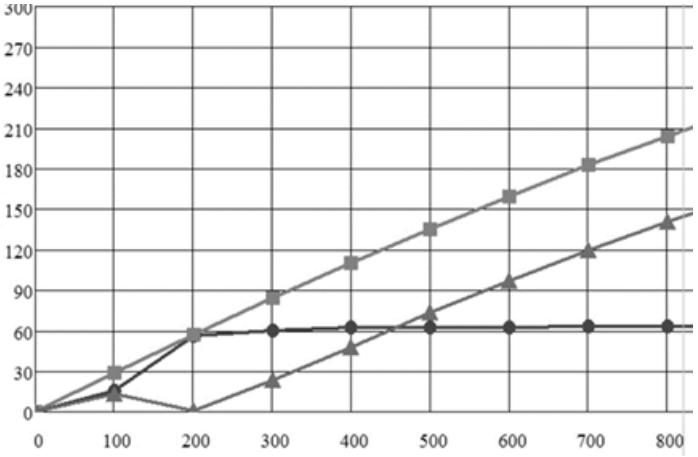


Fig. 4.25. Dependence of the maximum electric power on the density of the incident solar radiation flux, variable load resistance for working temperature 30 °C

Also, Fig. 4.26 shows the relative increase in the maximum electrical power under the condition of a variable load resistance relative to its generation at a constant load at an working temperature 30 °C.

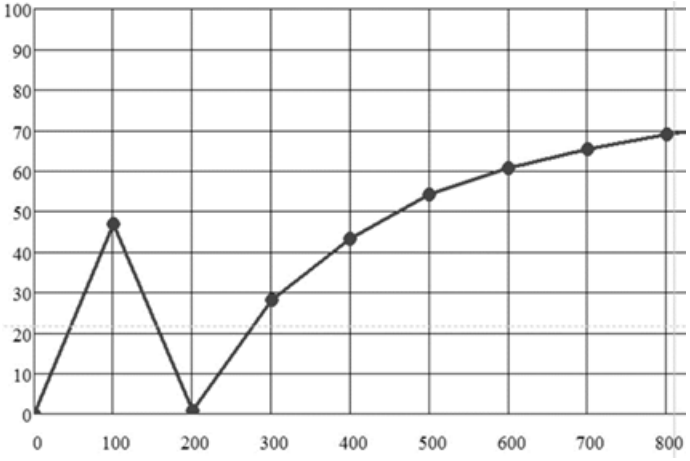


Fig. 4.26 Relative increase in the generation of maximum electrical power of a photovoltaic system with a variable load resistance relative at working temperature 30 °C.

The calculation of the specified relative power increase was carried out based on the formula:

$$\varepsilon_{P_{\max}} = \frac{\left|P2_{\max}(s) - P1_{\max}(S)\right|}{P2_{\max}(S)} \cdot 100\%. \quad (4.25)$$

From the analysis of the results, it is seen that the relative increase in the maximum electric power generated by the PVC with a variable load resistance is on average higher by more than 50% compared to the mode when the load resistance is a constant value. This means that calculating the load resistance value with high accuracy depending on the state of the environment using a fuzzy system with an ANN is very appropriate, because it allows to increase the efficiency of the PVC by adjusting the load resistance.

4.7. Verification of the models by physical experiment

To assess the reliability of the obtained simulation results, let's conduct a physical experiment using laboratory setup in the composition:

- PVC Solibro HNS-SD140;
- ammeter E-514/3;
- voltmeter M45M0M3;
- FLUS ET-965 illumination and temperature measuring device;
- Maynuo M9712B programmable electronic load.

Solibro HNS-SD140 generates a maximum power of 140 W and is made of amorphous silicon.

The density of the incident solar radiation flux was calculated based on the measured value of the illumination on the surface of the FEP based on the formula:

$$E = h \cdot \nu, \quad (4.26)$$

where E is the energy value of the incident light wave;

ν is the frequency of the incident light wave;

h is Planck's constant, which is equal to 1.054·10-34 J·s.

The results of measuring the parameters of the PVC are given in Table 4.9, but Table 4.10 shows the results of comparing the data obtained by conducting a physical experiment with simulation data.

Table 4.9 Measurement results of the Solibro HNS-SD140 FEP parameters.

Working temperature, ºC	Illumination, Klk	Flux density, W/m2	Maximum electrical power, W	Load resistance, Ohm
20,4	28,19	409,02	66,4	62,1
21,2	38,18	553,97	88,0	44,2
22,1	46,59	676,00	101,9	34,0
22,9	51,07	741,00	110,4	30,4
24,1	56,44	818,92	122,2	27,1
26,0	60,93	884,06	129,7	24,6
26,8	62,99	913,95	130,7	23,5
28,2	66,31	962,12	137,3	22,7

Table 4.10 Comparison of the results of simulation with results of physical experiment

Load resistance according to the experimental results, Ohm	FEP load resistance according to simulation results, Ohm	Relative error, %
62,1	63,2	1,77
44,2	44,7	1,13
34,0	33,5	1,47
30,4	30,8	1,32
27,1	26,6	1,85
24,6	24,4	0,81
23,5	23,6	0,43
22,7	23,0	1,33

The final analysis and comparison of the results obtained experimentally and through the application of the proposed information technology to the real-life PVC Solibro HNS-SD140 showed that the error between them is no more than 2%.

4.8. Summary

PVC are actively used today, but the generation of maximum electrical power of PVC is possible only under certain condition when the PVC load resistance is on maximum power point. This provided by using of Maximum Power Point Tracking (MPPT) control systems.

Taking into account the nonlinear nature of the control system caused by the influence of many external factors, MPPT follows the principle of fuzzy control, which requires the definition of linguistic variables, a rule base and a Takagi-Sugeno defuzzification algorithm.

The adjustment of the linguistic variables can be implemented with using of ANN training by results of analytical modeling of the PVC control surface.

The effectiveness of the designed MPPT fuzzy control system can be assessed by the results of simulation experiments for different modes of the PVC. The analysis of the accuracy and adequacy of the model is carried out by conducting a physical experiment, which confirms the effectiveness of the proposed approach to building a MPPT fuzzy control systems.

References

1. Carrera L. A. I., Garcia-Barajas M. G., Constantino-Robles C. D., Alvarez-Alvarado J. M., Castillo-Alvarez Y., Rodrigues-Resendiz J. Efficiency and Sustainability in Solar Photovoltaic

Systems: A Review of Key Factors and Innovative Technologies // Eng 2025, 6, 50, https://doi.org/10.3390/eng6030050.
2. Fkirin M. A., Gowaly Z. M., Elsheikh E. A. Dynamic Controller Design for Maximum Power Point Tracking Control for Solar Energy Systems // Technologies 2025, 13, 71, https://doi.org/10.3390/technologies13020071.
3. Anya, I.F., Saha, C., Ahmed, H., Huda, M.N., Rajbhandari, S.: Performance improvement of perturb and observe maximum power point tracking technique for solar PV applications. 2020. https://doi.org/10.1007/978-3-030-05578-3_10.
4. Pandey, A.K., Singh, V., Jain, S.: Maximum power point tracking algorithm based on fuzzy logic control using P–V and I–V characteristics for PV array. IEEE Trans. Ind. Appl. **59**, 4572–4583 (2023). https://doi.org/10.1109/TIA.2023.3272536.
5. Sakthi Suriya Raj J. S., Pounraj P., Prince Winston D., 4Ramaraj R., Cynthia Christabel S. Intelligent MPPT Control Technique for Solar PV System // International Journal of Applied Engineering Research, ISSN 0973-4562 Vol. 10 No.55 (2015).
6. Dawahdeh A., Hussein Sharadga, Sunil Kumar. Novel MPPT Controller Augmented with Neural Network for Use with Photovoltaic Systems Experiencing Rapid Solar Radiation Changes // Sustainability 2024, 16(3), 1021, https://doi.org/10.3390/su16031021.
7. Jezewski M., Czabanski R., Leski J. Introduction to Fuzzy Sets // In book: Theory and Applications of Ordered Fuzzy Numbers, 2017. Part of the book series: Studies in Fuzziness and Soft Computing, volume 356, pp. 3–22. DOI: 10.1007/978-3-319-59614-3_1.
8. Murilo Franco Coradini, Luiz Henrique Vitti Felao, Stephany de Souza Lyra, Marcelo Carvalho Minhoto Teixeira, Claudio Kitano. Takagi–Sugeno Fuzzy Nonlinear Control System for Optical Interferometry // Sensors 2025, 25, 1853. https://doi.org/10.3390/s25061853.
9. Murillo-Yarce D., Alarcon-Alarcon J., Rivera M., Restrepo C., Munoz J., Baier C., Wheeler P. A Review of Control Techniques in Photovoltaic Systems // Sustainability 2020, 12, 10598, https://doi.org/10.3390/su122410598.
10. Surendra V., Radha Krishna S., Prasad P., Galib Shareef Sk., Dr. Karthik N. Solar Panel Monitoring Using IoT and AI Techniques // IJCRT 2024, Vol: 12, Issue: 4, p. 178-189.
11. Isknan, I., Asbayou, A., Adaliou, A.H., Ihlal, A., Bouhouch, L.: Comparative study and simulation of advanced MPPT control algorithms for a photovoltaic system. Indones. J. Electr. Eng. Comput. Sci. 30, 46–56 (2023). http://doi.org/10.11591/ijeecs.v30.i1.pp46-56
12. Jiang, P., Luan, Y., Zhang, W.: MPPT tracking control based on variable step perturbation and observation method. In: Proc.—2019 Chinese Autom. Congr. CAC 2019, pp. 106–110. (2019). https://doi.org/10.1109/CAC48633.2019.8996703
13. Zadeh L.A. Fuzzy sets // Information and Control, 1965 Vol. 8, pp. 338–353.

Chapter 5. UAV swarm digital control system

5.1. MCU based UAV control system architectures review

Rapid technical developments have made Unmanned Aerial Vehicles (UAVs), or drones, much more accessible and widely used by people and commercial companies. Also, as the events of recent years show, UAVs are becoming a significant tool for military operations. Terrorist attacks at the state level are especially dangerous, when the number of drones can reach hundreds. As a result, drones pose a serious threat to people and infrastructure.

Obviously, it is more effective to counter a large number of drones using not single drones, but groups of drones operating according to swarm rules [1]. Drones in a swarm interact with each other, exchange information and coordinate their actions. This allows for an increase in the speed of response to events and efficiency compared to single drones. However, working in a group requires the drone to meet certain requirements, which leads to the need to develop new hardware and software. In particular, this applies to the control system, since in single drones the control system has very limited functionality, configured for autonomous operation.

Regarding the control system itself, the work [2] proposes the following types of different architectures for UAV group control system which can be used for swarm drone-based defense systems. First of all, it is centralized control with a Leader, which is controlled by a central system (pilot or an AI-based system) with constant communication support. The main capabilities of this system are constant control and monitoring of one UAV that controls the entire group at a stationary object.

Another solution is collective self-governance with information exchange (multi-agent architecture). For an air defence system built on the basis of UAVs, a multi-agent architecture can be most effective, as it allows each drone to perform work both individually and as part of a group (swarm). In addition, when any UAV fails, it will be easy to replace, since they do not differ from each other in functionality. However, the main problem in this architecture remains the complexity of developing a decision-making system for the entire group, and not just for each agent individually. Based on the limitations and requirements for the elements of a multi-agent system, it is necessary to choose a control system architecture for drone agents that would allow for them as part of a swarm.

For UAVs with digital control the as basic is the scheme with the Gyrodyne QH-50 DASH that is used, for example, in American single-rotor UAVs. This control system does not have elements that can autonomously make decisions and generate flight control commands. There is constant control of the UAV from a ground control station, which practically eliminates its use in a multi-agent system. For our problem, we should consider the control systems with flight controller which controls drone during flight.

One of the first flight controller was MultiWii [3]. This is an open-source project based on the Arduino platform, which is used both for polling sensors and for generating control signals. In fact, the flight controller is a board with contacts to which sensors and communication modules need to be connected. The board is based on ATMega2560 or STM32 microcontrollers. This flight controller has low computational power, and although provide some flight control functions, the operator must always control the movement of the drones. Thus, using such control systems in a multi-agent system is practically impossible.

The next control system is based on the specialized CopterControl3D (CC3D) controller [4], which, unlike MultiWii, is already oriented towards use in multicopters and has its own configuration software. Since the controller is based on the STM32F103 microcontroller, there is practically no increase in computing power. Therefore, this control scheme has found widespread use in FPV drones, but is completely unsuitable for a multi-agent system.

On more example of drone's control system is based on the specialized ArduPilot Mega flight controller from Arduino [5]. This system has a two-way telemetry system. The controller is

programmed in the Arduino programming language and is more flexible for programming tasks. It is based on the ATMega32U2 microcontroller, whose performance is very limited and does not allow for the implementation of multi-agent system agent tasks.

Finally, the most common UAV control system is based on the Pixhawk flight controller [6] shown in Fig.5.1.

Fig. 5.1. View of the PixHawk flight controller [13].

Instead of previously considered flight controllers, Pixhawk has a real-time operating system, supports the Mission Planner simulation system, has a large number of interfaces, and is available as a complete module in a case, which can be integrated into a project with minimal adaptation. The architecture of the control system based on Pixhawk is shown in Fig. 5.2 [7].

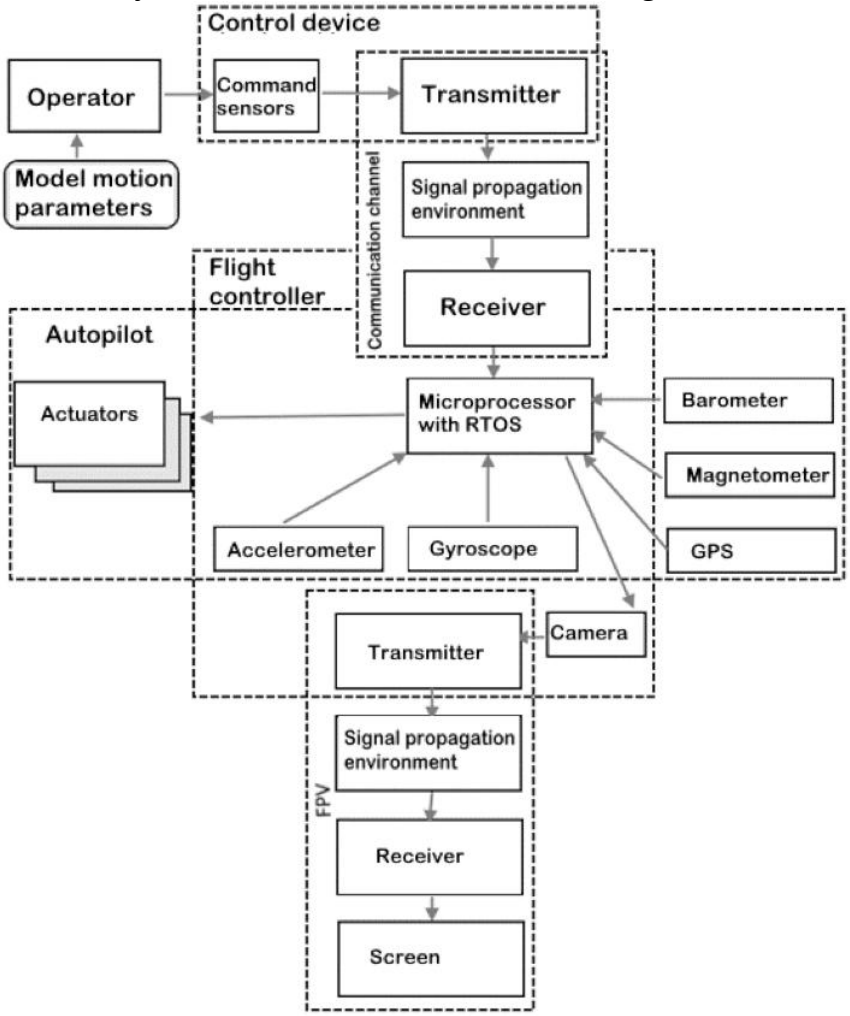


Fig. 5.2. Control system architecture based on Pixhawk flight controllers [7].

The Pixhawk board is equipped with an STM32F427 Cortex M4, significantly increasing the computational capacity of the control system. However, this makes it very difficult to implement third-party software within the system to address intelligent flight control tasks. For example, the anti-drone defense system of critical infrastructure facilities, the scheme of which is shown in Fig. 5.3, requires the use of a quadcopter swarm, which operates on the principles of multi-agent systems.

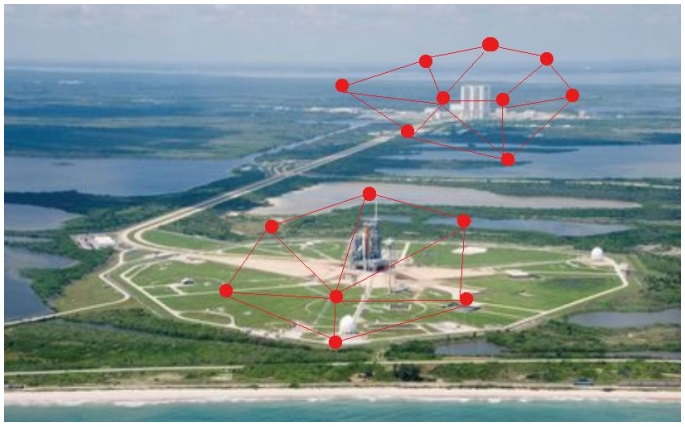


Fig.5.3. The scheme of anti-drone defense system [8]

The software structure of such multi-agent control system must ensure the parallel operation of at least five drone software subsystems presented in Fig. 5.4.

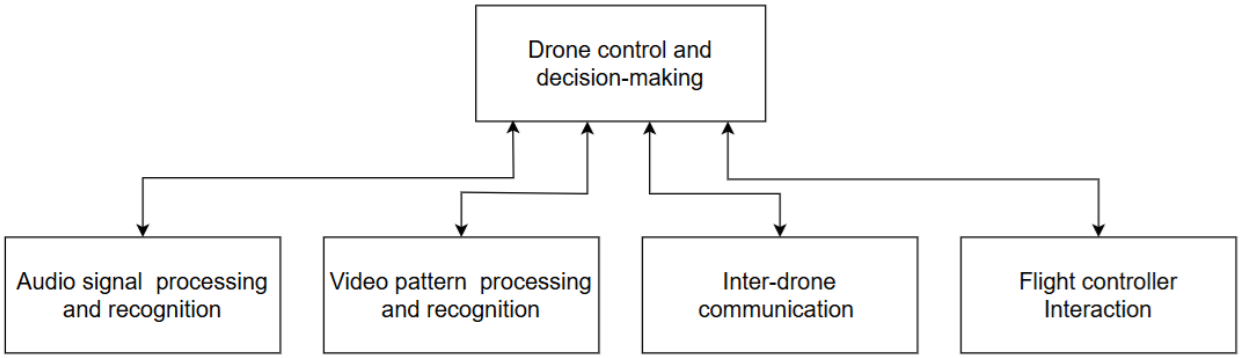


Fig. 5.4. Drone software structure.

Summarizing the results of the analysis, it should be noted that to ensure autonomy of drone in combination with inter drone communication, a flight controller is insufficient. To increase the computational power of the drone's control system, a multi-level hardware architecture and, accordingly, a multi-level software architecture must be developed.

5.2. Single-board computer-based drone control system

To solve the tasks that defined the drone swarm missions shown in Fig. 5.4, a three-level hardware architecture of drone control system with single-board minicomputer can be proposed (see Fig. 5.5):

- the low level consists of actuators and sensors that directly interact with the flight controller;
- the middle level is based on PixHawk flight controller with the PX4 autopilot; it provides a wide range of interfaces for interaction with a single-board computer and sensors.
- the upper level is based on a single-board computer, whose software analyses data from the flight controller, sound and optical sensors, and the power subsystem; based

on the processing of this data, the upper level generates commands for all drone components and supports communication with other agents of the multi-agent system.

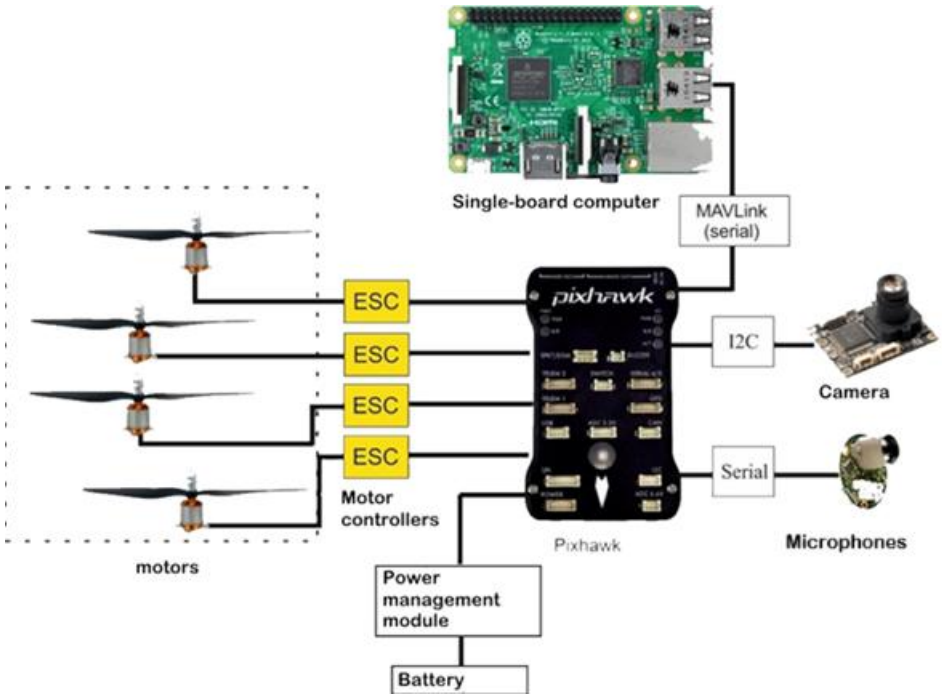


Fig. 5.5. The architecture of a drone control system with single-board computer.

The choice of a single-board computer is limited by certain requirements, the main of which are size, weight, power consumption, performance, availability of interfaces and communication systems, and the presence of developed software tools. In our case, the dimensions of the single-board computer are limited by the size of the quadcopter chassis. Weight is also limited by the capabilities of the chassis.

The interfaces of the single-board computer must provide communication with the flight controller via the MavLink protocol [9], as well as connection of surveillance cameras and microphones. The communication system must be wireless or have a standard interface for connecting wireless communication modules.

The requirements for the single-board computer's software are determined the mission of drone swarm, the availability of development and debugging tools. Therefor the presence of a typical operating system in which software execution and communication can be organized according to general principles is one of the most important requirements. But power consumption should be as low as possible to increase flight time.

Based on the existing constraints, the most suitable choice among single-board computer platforms is NVIDIA Jetson Nano Developer Kit [10]. This single-board computer has the capability to run multiple neural networks in parallel to support applications for image classification, object detection, and audio processing. Also, this platform is easy to use and highly energy-efficient, consuming no more than 5W, which is critical for the autonomy of the drone.

The single-board computer is supported by the NVIDIA JetPack SDK. Its software package provides the necessary resources and capabilities for solving artificial intelligence (AI) tasks: a full Linux desktop with NVIDIA drivers, AI and computer vision libraries, application programming interfaces (APIs), compatibility with NVIDIA's main AI platform for training and deploying AI software.

Structurally, the NVIDIA Jetson Nano computer is a combination of the Jetson Nano Module and a carrier board. The main elements of the processor module are a 128-core NVIDIA GPU with Maxwell architecture and a quad-core ARM Cortex-A57 processor clocked at 1.43 GHz. The module

is cooled by a heat sink that dissipates up to 10W at an ambient temperature of 25°C. If additional cooling is required, the module can be configured to control a system fan (40 × 40 mm). The board contains up to 8 GB of 64-bit LPDDR4 25.6 Gb/s RAM and a microSD card slot. The microSD card is used for writing and installing the Linux operating system image, as well as for data storage. The card must provide a sufficient amount of memory (at least 16 GB) and data transfer speed (UHS-1). The module provides a performance of up to 472 GFLOPS (Floating-point Operations Per Second).

The carrier board, equipped with interfaces such as Gigabit Ethernet, MIPI CSI-2 DPHY lanes camera connector, HDMI 2.0 and eDP 1.4, 4 × USB 3.0, 1 × USB 2.0 (microUSB), GPIO, I2C, I2S, SPI, UART, data storage (microSD card slot), camera, gigabit Ethernet.

Thus, the Jetson Nano Developer Kit is better suited for solving quadcopter control tasks, given all approximately equal parameters, having hardware support for video processing and being oriented towards multi-agents communication. A drone control system architecture based on the available interfaces of the Jetson Nano Developer Kit, is shown in Fig. 5.6.

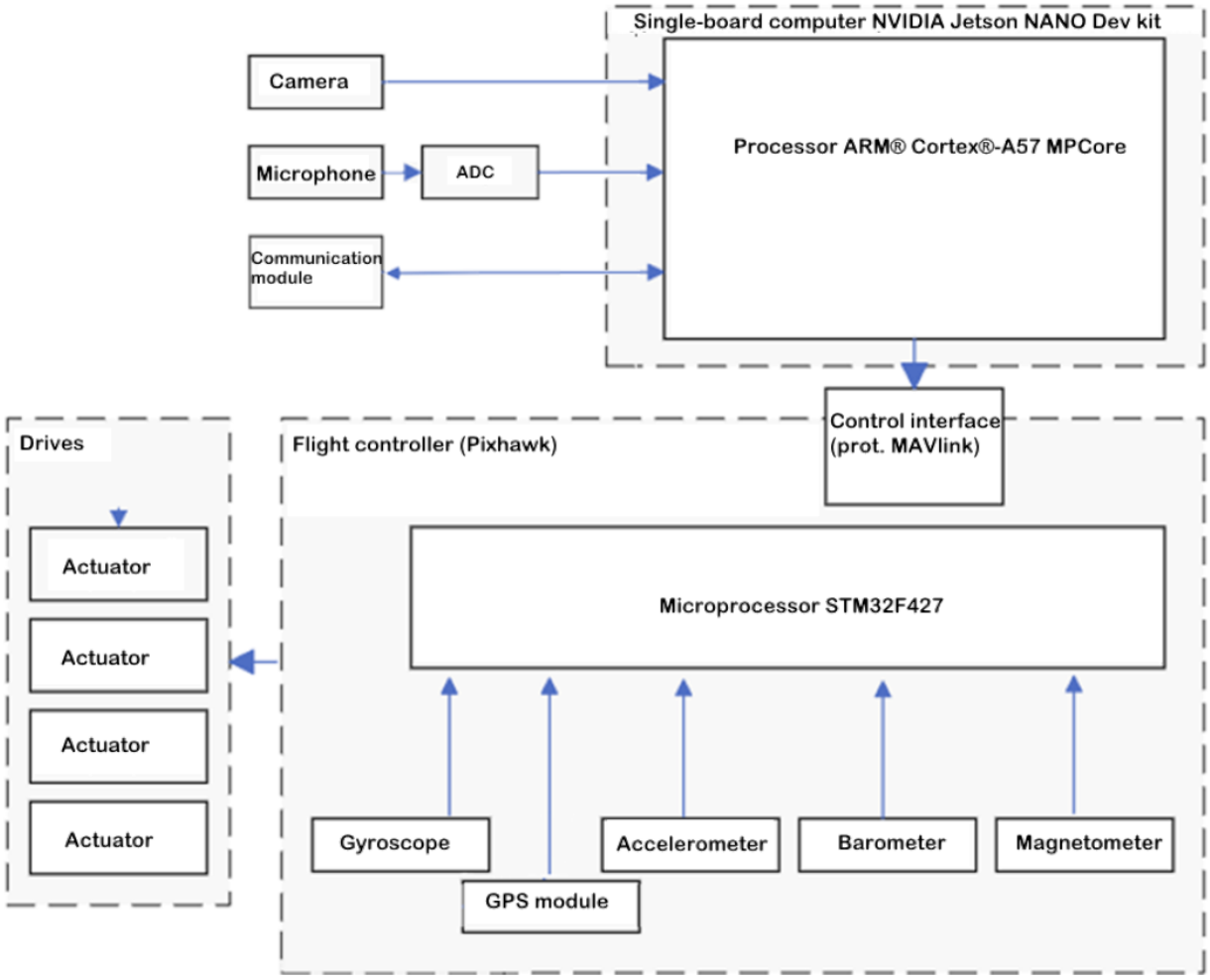


Fig. 5.6. Drone control system architecture based on Jetson Nano Developer Kit.

5.3. Drone control and decision-making software

5.3.1. Drone model-oriented control

Thanks to presence in the structure of drone control system the Jetson Nano microcomputer with an operating system that supports execution of embedded programs, it becomes possible to implement the behaviour of a drone, which acts as an agent in a drone swarm, based on the principles of model-oriented control (MOC).

The essence of MOC is the use of control algorithm models, represented in the form of control networks, directly in the control circuit of the object, in our case - a drone. The essence of MOU is the use of control algorithm models, presented in formal form of Control E-networks (CEN), directly into the control circuit of the object, in our case - a drone. As result, this actually allows not only to automatically convert the control algorithm model into program code in real time, but also to apply verification of control algorithm using its simulation at the stage of control system design.

In general, CENs are an extension of Petri nets for control purposes. However, when they use to describe the behaviour of drones (agents) in a multi-agent system, their definition requires some specification in the form:

$$CEN = (P, T, E, M^0, \tau, \varphi, G, L) \tag{5.1}$$

where each element of the tuple is defined as follows:

- P - a set of positions (places) representing states or steps of the UAV swarm's control process. Each position $p \in P$ can hold a token indicating an active state;
- T - a set of transitions of five types describing a rule for moving tokens between positions. Transitions model the conditions and actions for progressing the control logic from one state to another;
- E - a set of directed edges (arcs) connecting positions to transitions;
- M^0 (Initial Marking): the initial marking function $M^0(p) \in 0,1$ (a binary marking) to indicate which positions have a token at process start time;
- τ - a timing function $\tau: T \to R_{\geq 0}$ assigning an execution delay to each transition. This is useful for modeling-controlled delays or timeouts in drone operations (optional);
- φ - a logical guard condition for transitions. $\varphi: T \to$ {true,false} depending on the state of external variables or probabilistic logic). This allows to model the transition fire condition taking in account the data of global variable updated by a listener;
- G – a set of global variances that save of data about state of control algorithm (data from physical sensors, received messages from other drones of UAV swarm;
- L - a set of listeners that handle external events and signals, forming the bridge between the CEN model and its environment.

Formally, the enabling condition for a transition at *t* time moment can be written as:

$$\text{Enabled}(t) \equiv \bigwedge_{p \in \cdot t} [M(p) \geq 1] \wedge \varphi(t) = \text{true} \tag{5.2}$$

When an enabled transition fires, it consumes tokens from its input positions and produces tokens in its output positions ($t \cdot = , p \mid (t, p) \in E$), after an optional delay $\tau(t)$. The firing can also change the token attributes that together with place marking and global variables define the state of control algorithm. Listeners are special components that can either (a) inject tokens or trigger transitions immediately upon external events, or (b) update shared variables/flags that are checked by transition conditions.

CEN model of drone control algorithm is presented in a simplified form in Fig. 5.7. This model implements three layered control levels—Reactive, Planning, and Cooperative—each showcasing key mechanisms of our approach.

Reactive control level provides the monitoring of critical global state variables and trigger safety actions when needed. Transition X1 checks the drone's global battery status but transition X3 uses position data to determine landing readiness. The drone's altitude listener updates a global position status. In such way these transitions embody reactive safeguards controlled via global state checks.

The planning layer encapsulates the high-level mission sequence that consists of initialization, navigation, target engagement, return, landing, and mission stop while remaining continuously sensitive to events coming from the Reactive and Cooperative levels. Within this chain, all major

CEN transition types are exercised: sequential segments (e.g., initialization → navigation) are implemented with T-type transitions; parallel observation and navigation tasks run under F-type transitions; conditional progression hinges on Y-type transitions whose guards depend on global variables set by listeners; and synchronization with teammate messages is handled by the J-type transition. So, the planning layer is not merely a linear script: it is a hybrid flow in which reactive (battery, position) and cooperative (target confirmation) events dynamically steer the mission path.

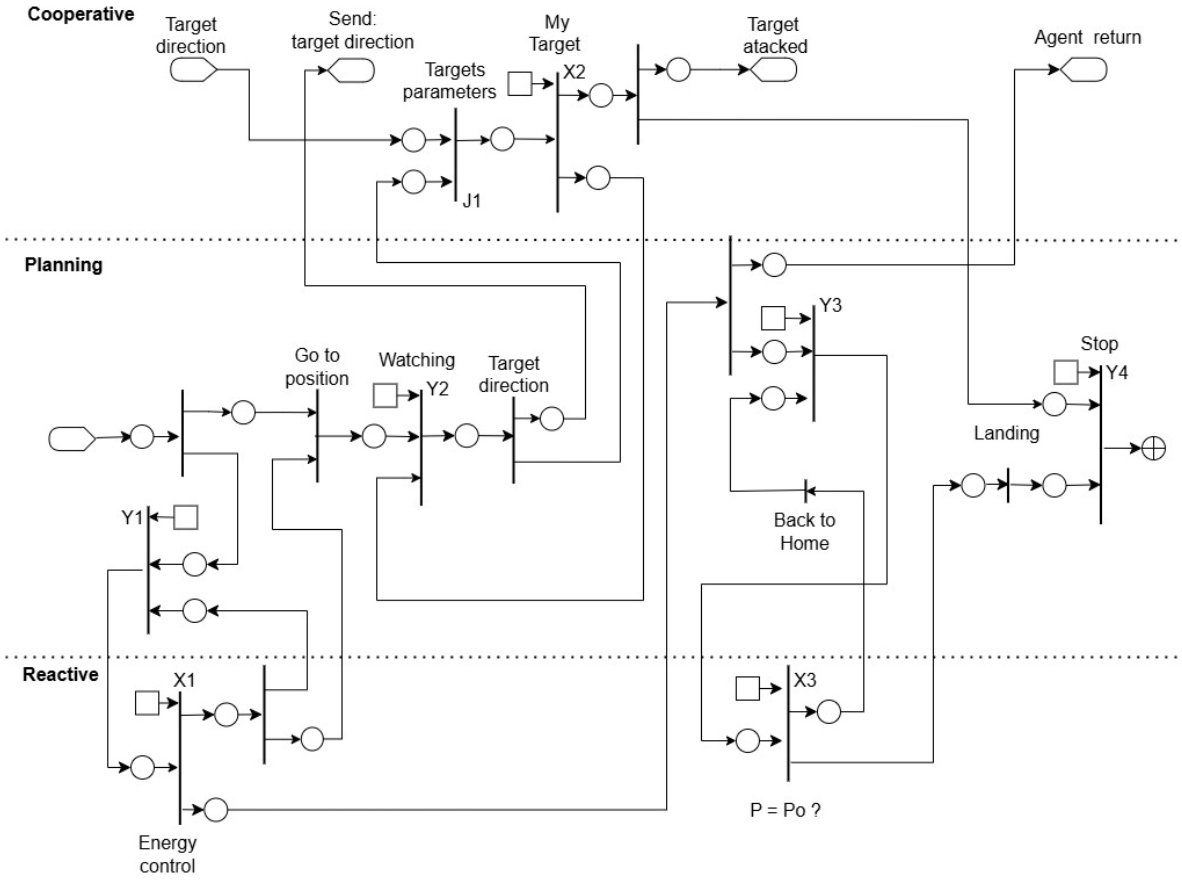


Fig. 5.7. Drone control algorithm CEN model

Cooperative layer: The cooperative control level handles synchronization with external agents or commands. Here, a J1 transition is used to merge local and remote event streams, enabling multi-UAV coordination. It has two input places: one receives a local token (representing the drone's readiness or a local observation, such as detecting a potential target), and the other receives an external token delivered via an inter-agent communication. In our scenario, incoming message is emerged the reactive listener creates a token in the place, causing the model to evaluate J1 without delay. Transition J1 will only fire once both inputs are present, thus synchronizing the drone's local state with the received data. This demonstrates that the architecture can effectively synchronize multi-agent behaviours with minimal latency and correct data merging.

5.3.2. Control algorithm implementation

The first step in implementation process is a definition of classes corresponding to the theoretical elements of the model (see Fig. 5.8):

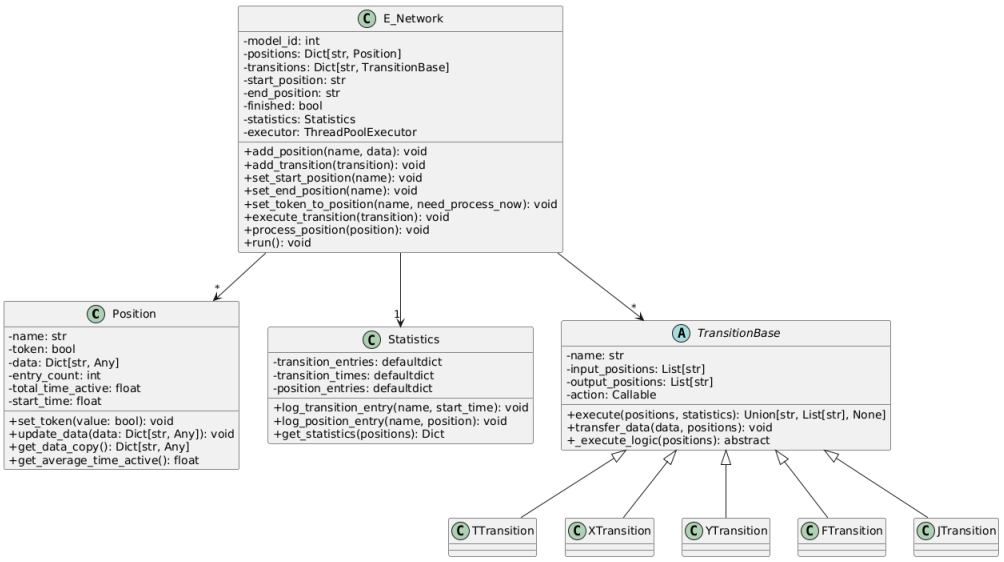


Fig. 5.8. Class diagram of the CEN implementation.

To operate effectively in real time, the CEN model incorporates specialized transition types and execution mechanisms. These mechanisms ensure that both sequential logic and concurrent or asynchronous events are handled properly. The main types of transitions and event-handling strategies are outlined below, aligned with real-time requirements:

Sequential Transitions (T-type). This is the simplest type of transition, corresponding to the traditional Petri net transition with a possible time delay without any additional conditions,

Synchronous Join Transitions (J-type). A J-type transition implements a synchronization barrier, firing only when all of its defined input positions have tokens. This corresponds to a logical AND-synchronization (similar to a Petri net transition with multiple input arcs).

Asynchronous Fork Transitions (F-type). An F-type transition represents a parallel branching (fork) in the control flow.

Conditional Transitions. In addition to the above, the model supports conditional branching transitions. In our implementation X-type choose one of multiple outputs based on a choice function which define the number of outputs that gets the token from input place when the transition fires. This is analogous to a switch/case or an exclusive OR (XOR) split in workflow terms. Against, a Y-type transition can be used to model an OR-join where one of several inputs can pass the token to one output place. These additional transition types extend the expressiveness of the model, though for clarity the core discussion focuses on the fundamental T, J, and F types.

Event Listeners is one of the most significant features in CEN modification to integrate of external events. Here are two modes of external event handling by listeners:

- reactive listeners (immediate mode): In this mode, when an external event or input signal occurs, the corresponding listener in *L* immediately places a token into a designated position (or directly triggers a specific transition) and invokes the processing of that position without delay. This is analogous to an interrupt service routine in embedded systems. The benefit is real-time responsiveness: the event is handled as soon as it occurs, outside of any fixed scanning cycle.
- polling or flag-based listeners (deferred mode): In some cases, rather than injecting a token immediately, a listener may simply update a global variable or flag when an external event occurs. The model's transitions periodically check these variables as part of their guard conditions. It integrates smoothly with transition which fire periodically.

Both types of listeners ensure that the model remains responsive to the outside world. The reactive approach (A) gives immediate insertion of events into the token flow, while the flag-based approach (B) uses regular transition evaluation to pick up events.

The listeners are implemented as callback hooks from drone sensor interfaces or network message handlers. They call set token to position method on the CEN engine when a UAV sends a telemetry update or a ground station issues a command. The output signals correspond to function calls or messages to the drones. When the model generates an output signal, the code would translate that into an API call to a drone (such as changing a drone's flight mode or sending a new waypoint). For example, output transition (like a T-type transition designated as an output action) can directly invoke such code via its action callback.

A Position has a token attribute (boolean in our implementation, indicating whether a token is present), and it can hold a data attributes. This data payload travels with the token, allowing information to be passed between states (for example, one UAV could deposit sensor data in a token that moves to the next state for processing). The Position class also keeps simple metrics: it counts how many times a token has entered that allows the model to maintain stateful information as tokens move.

Transitions base class and specialized subclasses encapsulate the logic of moving tokens from inputs to outputs under certain conditions or actions. In the implementation, an abstract class TransitionBase defines the common interface and properties of transitions (name, list of input positions, list of output positions, and an optional action to perform during firing). Specific types of transitions are implemented as subclasses.:

All transition classes implement an execute() method (inherited from TransitionBase) which in turn calls an internal_execute_logic() specific to each subclass. The execute() method also hooks into a Statistics component (logging transition execution times). Notably, transitions use the transfer_data() helper to copy data to output positions, ensuring that each new token carries forward the data from its origin

The E_Network class represents the entire CEN model instance (essentially, it corresponds to the "Controller" in the MVC analogy, managing the Model's state). It contains dictionaries of positions and transitions of CEN model, marking of places, a thread pool executor for running transitions concurrently and a Statistics object for collecting performance metrics. In fact, it implements the logic to run the network.

The concurrency model of E_Network is partly event-driven (trigger transitions as soon as input tokens appear) and partly multi-threaded (parallel branches). This hybrid ensures that no busy-wait polling is needed for normal transition firing – every token insertion trigger checks immediately – and parallelism is exploited where the model design calls for it. The careful use of locking in J-type transitions and thread pooling in F-type transitions addresses real-time concerns: threads can run simultaneously to handle different UAVs or tasks, but synchronization points are handled safely.

To analyze performance (important in real-time systems to ensure timing constraints are met), the implementation includes a Statistic utility. It records how many times each transition fired and the distribution of execution times, as well as how many times each position was activated and how long tokens stayed there on average. While not part of the core model, this is useful for debugging and tuning the system (for instance, identifying bottlenecks if a transition is slow or gets queued often). Logging is also built into the engine (with model ID tagging) to trace execution for debugging.

In total the drone control algorithm includes 26 positions and 16 transitions, 3 listeners (communication messages, battery status, current position) and 3 global variables (battery level, current position, drone status). Previously, this model was tested with NetLogo multi-agent modeling system [8] to observe how the agent reacts to events and coordinates with a swarm collecting runtime statistics to assess performance.

Based on classes diagram the developed CEN model is implemented as Python project that is embedded in JetsonNano on drone hardware platform. Fragment of CEN model in Python implementation is shown in Fig. 5.9.

```
358  def setup_model(model_id):
359      model = E_Network(model_id)
360      # setup positions
361      model.add_position("P1")
362      model.add_position("P2")
363      ...
364      #  setup transitions
365      j1 = JTransition("J1", ["P1", "P2"], "P3")
366      x1 = XTransition("X1", "P3", ["P4", "P12"], condition=lambda: condition_x1())
367      t4 = TTransition("T4", "P8", "P5", delay=1, action=action_increment_repeat)
368      ...
369      model.add_transition(j1)
370      ...
371      model.set_start_position("P1")
372      model.set_end_position("P15")
373      return model
```

Fig. 5.9. Fragment of CEN model in Python implementation.

5.3.3. Physical experiments with control algorithm model

Physical experiments on a standard X quadrotor (450mm, 4 S Li Po) which avionics combine in real time PixHawk flight controller with a Python driven mission computer, show:

- low-level attitude control is handled by a Holybro Pixhawk 6C (STM32H743, triple-redundant IMU, PX4 v1.14), while high-level CEN logic executes on a Jetson Nano B01 (quad-core A57 @ 1.43 GHz, 4 GB LPDDR4, JetPack 4.6);
- the Pixhawk closes the 400 Hz inner loop and streams MAVLink telemetry over USB-C to the Jetson, which runs the event-driven swarm controller at 20–50 Hz;
- sensors feeding reactive listeners are: an 8 MP IMX219 front camera for target recognition, a MEMS directional microphone for acoustic bearing cues, a downward optical-flow camera for hover stabilisation, and integrated GNSS/barometer modules for global pose and battery data. Inter-agent coordination packets are exchanged via a Digi XBee 3 2.4 GHz mesh radio (< 15 msec latency);
- hard real-time loops remain on the MCU; the Jetson injects events into the CEN as (i) immediate tokens when the camera or mesh-radio triggers, and (ii) flag updates for slower variables such as voltage or acoustic level. This splits yields < 8 ms end-to-end event latency, demonstrating that the reactive, model-centric architecture fits within the resource envelope of a typical quadcopter.

The implementation choices below were made to satisfy real-time constraints and the needs of embedded control:

- **Low Latency Event Handling.** In traditional PLC or polling systems, an external event might wait until the next cycle tick to be handled, which could be tens of milliseconds or more. In our case, by using an event-driven approach, we avoid the latency of a fixed cycle loop as soon as a critical event occurs (e.g., proximity alert from a UAV's sensor), the listener directly triggers the relevant part of the model. This is crucial for swarm safety and reactivity.

- **Deterministic Synchronization.** The use of J-type transitions with locks ensures that even in a highly concurrent scenario: the model's behavior is deterministic and consistent with the formal specification. The cost is a brief locking period, but this is limited to the duration of the join operation (typically negligible compared to sensor or network I/O latencies).
- **Parallel Execution.** Real UAV swarms inherently involve parallel activities (each UAV operates independently yet cooperatively). The thread pool allows our model to reflect this parallelism. This improves throughput and aligns with multi-core processing capabilities of modern embedded platforms. We do impose a limit (e.g., max 10 threads) to avoid overwhelming the system, but this can be tuned based on hardware.
- **Resource Management.** We use additional mechanism to implement queue for handling situations where tasks queue up, ensuring that no data is lost and that processing occurs in order. In a real UAV system, this could correspond to queuing incoming requests or telemetry if the system is momentarily busy. By providing this within the model, we ensure the control logic can include buffering behavior explicitly.
- **Timing and Delays.** The model's inclusion of τ for transitions and the timing recorded by Statistics help to analyze and design the system with real-time deadlines in mind.

5.4. Audio data processing and recognition

5.4.1. Audio data processing

The control system described above employs audio classification of microphone inputs to make decisions about detected target.

Audio data processing is a crucial step in audio classification tasks, as the quality of input data significantly impacts the accuracy and reliability of algorithms. Sound data exhibits significant variability in duration, depending on the nature of the sound source and the context of its occurrence. To process long recordings, they are divided into shorter intervals (usually 1-5 seconds). This simplifies analysis and allows focusing on local sound events. To feed data into a model, it is necessary to ensure that each fragment has the same duration. This is achieved by trimming or padding recordings (e.g., by adding silence).

Noise is one of the main factors that complicate the recognition of sound events. It can arise from external influences (wind, traffic etc.) or hardware defects (noise from a microphone or electronic components). The application of noise reduction methods, such as low-pass and high-pass filters, or specialized algorithms will improve model accuracy. To enhance model adaptability to real conditions, methods of introducing noise into training data (data augmentation) are used. To distinguish between noise and useful signals, spectral analysis is used to isolate the frequency range characteristic of the target sound.

In many real-world scenarios, sound events occur simultaneously, making classification difficult due to partial signal overlap. To address this problem, spectral decomposition is used. This is the separation of overlapping signals into individual components. Models capable of classifying multiple different sound events simultaneously are also used. Threshold methods or clustering of spectral components are used to separate dominant events from less significant ones.

Audio classification is a subfield of audio data analysis that focuses on the automatic categorization or classification of sound signals based on their characteristics. The primary goal of audio classification is to construct models capable of detecting, identifying, and distinguishing between sound events for its real-time detection.

5.4.2. The methods of audio signals analysis

Machine learning algorithms are foundational to the effective analysis of features extracted from audio signals. They enable automated pattern discovery within data and facilitate classification or regression for subsequent decision-making. In the context of audio classification, a variety of algorithms are employed, with Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Random Forest being among the most prevalent. Each method possesses distinct advantages and characteristics, influencing their application based on specific tasks.

One of the primary limitations of these classical methods is the necessity for pre-processing and manual feature selection, which is critical for the successful operation of algorithms. For the application of classical methods like SVM or KNN, it is necessary to select and create representative features from raw data, which is very intensive process. Also, classical methods are unable to automatically extract relevant features from data. These methods can operate effectively only with limited data volumes. They require significant time and resources to process large datasets, such as large audio files or high-dimensional spectral features. For example, in the case of audio classification for detecting drone sounds, classical methods are unable to detect nonlinear or complex sound patterns.

Unlike classical methods, deep learning based on deep neural networks, in particular, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and Transformers are powerful ANN models capable of addressing the aforementioned challenges. Deep learning allows for the automatic extraction of relevant features from raw data, such as audio files, without the need for manual feature selection.

CNNs are among the most powerful deep learning tools that have gained popularity due to their effectiveness in solving computer vision tasks, but are also actively used for classification and analysis of audio data. They are particularly useful when processing structured data, such as images or audio files, where local dependencies and patterns are important and can be effectively extracted through the convolution process [11].

RNNs and their variants, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), are among the most effective tools for processing sequential data, such as audio, text, or video. These models are capable of storing information about previous elements of a sequence, making them particularly useful for tasks where context recognition and temporal dependencies are important. In the field of audio classification, recurrent neural networks allow for effective modeling of the dynamic and temporal aspects of audio signals. The standard RNN architecture includes hidden states that are passed through time, allowing the network to maintain a memory of previous input data. This makes RNNs useful for tasks where sequential dependencies are important, as in the case of audio, where each sound signal may depend on the previous one.

One of the main problems with classical RNNs is the vanishing gradient when the model cannot effectively store information over long time intervals, leading to a deterioration in training quality when processing long sequences [12]. However, LSTM and GRU allow for effective storage of information about long-term dependencies in the sequence. LSTM uses special memory units (cells) that can store information over long sequences. Instead, GRU as a simplified version of LSTM, has two main components: an update gate and a reset gate, that allows achieving similar results with fewer parameters, making it computationally less expensive compared to LSTM.

Transformers quickly gained popularity due to its ability to effectively process sequential data, including text, and have shown significant results in machine translation and text generation tasks. However, due to its versatility and ability for parallel processing, transformers have begun to be actively used in other areas, including audio classification. Transformers use the attention mechanism, specifically self-attention, which allows the model to learn the relationships between elements of a sequence regardless of their distance from each other in time. In traditional recurrent neural networks (RNNs) or LSTMs, this task is performed sequentially, which can lead to the vanishing gradient problem when processing long sequences.

Also, Transformers provide effective modeling of dependencies even at large distances between elements of a sequence, which is especially useful for processing complex audio signals where both local and global contexts are important. The self-attention mechanism allows transformers to focus on the most important parts of the audio signal, even if they refer to distant points in time. This is especially useful for tasks such as classifying complex sound events, where it is important to consider the relationships between different parts of the signal. Transformers have high scalability and can effectively process large amounts of data. They can work with large datasets, allowing for the creation of high-accuracy models for audio classification.

5.4.3. Software tools for ANN model implementation

In the development of audio classification systems, audio data processing, including analysis, filtering, transformation, and visualization, plays a crucial role. A wide range of software libraries exist to facilitate efficient audio processing. Below represents the short description of most popular Python libraries that widely used in modern audio signal processing projects.

Librosa. Audio analysis library which provides a wide range of functions for audio processing, including:

- support for most popular file formats, such as WAV, MP3, and FLAC;
- ability to compute spectrograms, mel-spectrograms, Mel-Frequency Cepstral Coefficients (MFCCs), and other features widely used in classification tasks;
- functions for extracting rhythmic and tempo features, pitch detection, and frequency characteristic analysis;
- tools for visualizing spectrograms and other sound characteristics.

Librosa's simplicity and detailed documentation make it a primary choice for prototyping audio classification systems.

PyDub. Audio file manipulation library focused on manipulating audio files. Its primary capabilities include:

- providing a simple interface for processing audio signals;
- supporting basic operations for improving sound quality;
- working with formats like MP3, WAV, FLAC, etc. (using FFmpeg).

This library is primarily used for pre-processing data, such as trimming long recordings or converting files to the desired format before analysis.

TensorFlow and Keras. TensorFlow is a powerful library for creating and training machine learning models. It supports both simple tasks and complex deep learning architectures. Its integrated Keras module provides a high-level API that simplifies model development.

TensorFlow features:

- use of static and dynamic computation graphs for optimizing execution and training models.;
- ability to run models on CPUs, GPUs, TPUs, as well as on mobile and embedded devices (TensorFlow Lite);
- ability to simultaneously train models on multiple devices or in a cluster;
- offers specialized tools for working with audio data, such as computing spectrograms and Fourier transforms.

Keras features:

- declarative coding style simplifies building models;
- allows for rapid testing of different neural network architectures;
- easily create custom layers, loss functions, and optimizers.

PyTorch. This is a modern deep learning library known for its flexibility and intuitive interface. It provides dynamic computation graphs, making it convenient for research and creating innovative models.

PyTorch features:

- dynamic computation graphs enable modifying the model structure during training, which is critical for experimental research;
- TorchScript is a tool for converting dynamic models into static ones, optimizing them for deployment in production environments;
- integration with CUDA provides easy access to GPU computations for accelerating training;
- TorchAudio is a separate library for audio signal processing, providing file reading, spectrogram computation, and other operations.

Scikit-learn is a library for classical machine learning focused on ease of use and algorithm efficiency. It is suitable for feature analysis, developing basic models, and validation.

Scikit-learn features:

- includes algorithms for classification, regression, clustering, and dimensionality reduction;
- tools for normalization, standardization, handling missing values;
- integrated methods for cross-validation, building error matrices, quality metrics (F1-score, AUC-ROC, etc.).
- Scikit-learn can work with other libraries such as NumPy, Pandas, and Matplotlib.

5.4.4. Implementation of audio processing ANN models

Let's consider the implementation of audio processing ANN models on the example of CNN. General schema of CNN architecture and the training process is shown in Fig. 5.10.

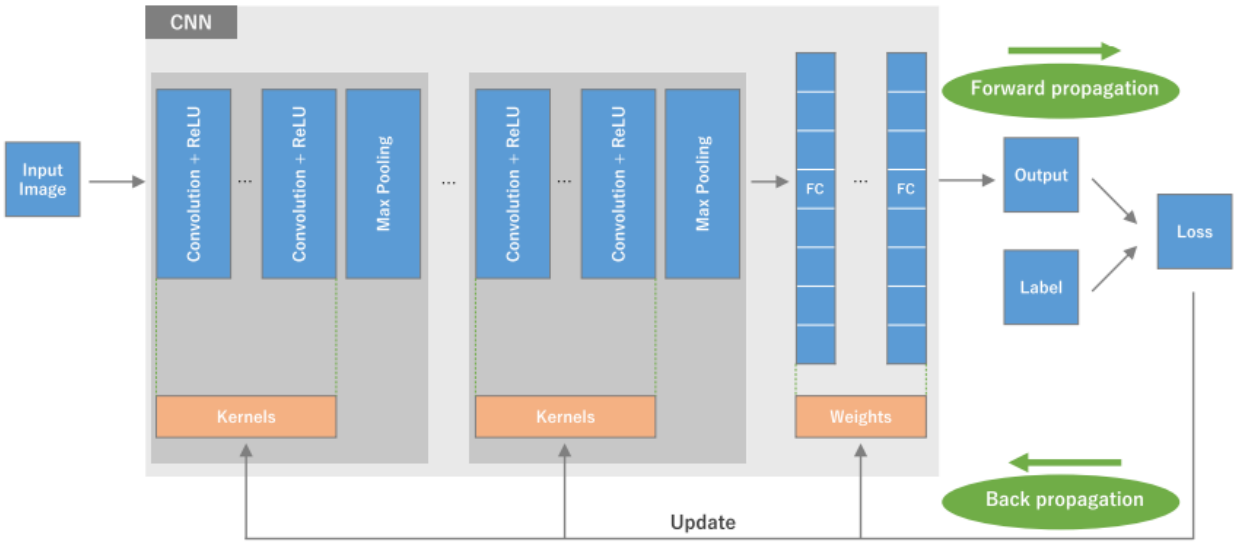


Fig. 5.10. The schema of CNN architecture and the training process [11].

CNN is typically composed of three types of layers:

1. Convolutional layers. In these layers, filters pass over the input data (e.g., spectrograms or mel-spectrograms) and generate activations to extract local patterns from the input data. Filters are trained during training and are able to detect various aspects, such as high-frequency or low-frequency components in audio.

2. Pooling layers (e.g., Max Pooling). After each convolutional layer, pooling layers often follow, which reduce the size of the image or spectrum while preserving the most important characteristics. This helps to reduce the number of parameters and improve the efficiency of the network, while preserving important patterns.

3) Fully connected (FC) layer. At the final stage, a fully connected layer is responsible for the final classification, processing the feature vector obtained after several stages of convolution and pooling.

The result of classification (Output) is calculated through forward propagation on a training dataset but kernels and weights are updated according to the loss value on backpropagation optimization algorithm. ReLU (rectified linear unit) is a piecewise non-negative activation function which facilitates learning of non-linear relationships by rectifying negative activations.

The architecture of implemented CNN model for audio recognition is illustrated in Fig. 5.11.

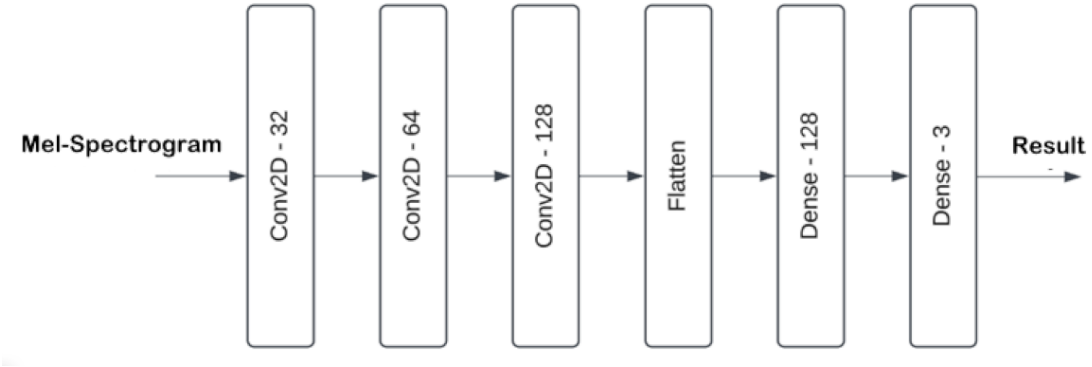


Fig. 5.11. The architecture of implemented CNN model

The initial convolutional layer comprises 32 convolutional filters (neurons) operating on mel-spectrogram inputs. This layer extracts local patterns within the spectrograms, such as characteristic frequencies and periodicities. This layer outputs 32 feature maps, each corresponding to a distinct filter. A MaxPooling layer is subsequently applied to down sample the feature maps by a factor of 2 in both dimensions (from 128×128 to 64×64). This down sampling process mitigates noise and retains salient features extracted by the convolutional filters, thereby reducing computational complexity while preserving essential information. The implementation code listing for this layer is presented below.

```
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 1)))
model.add(MaxPooling2D((2, 2)))
```

The second convolutional layer, comprising 64 neurons, extracts more complex features based on the input from the first pooling layer. The number of filters is increased as the model begins to learn more detailed characteristics, such as frequency combinations or transient processes within the spectrogram. Another pooling layer reduces the output dimensions from 64×64 to 32×32 and mitigates the risk of overfitting by reducing dimensionality while retaining the most salient features. The code for creating this layer is provided below. The code for adding the second layer to the model is:

```
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
```

The third convolutional layer, consisting of 128 neurons, extracts high-level features, such as complex patterns in the spectrogram that may be characteristic of drone sounds. The number of filters is further increased, as at this stage the model requires more capacity to analyze complex characteristics. The final pooling layer reduces the output dimensions from 32×32 to 16×16. It is provided by the next code:

```
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
```

A flattening layer transforms the two-dimensional feature maps into a one-dimensional vector, preparing the data for input to the fully connected layers by the code:

```
model.add(Flatten())
```

The fully connected layers consist of a first layer that detects relationships between high-level features, enabling the model to perform more complex classification tasks; and a second layer that transforms the output values into probabilities for each class. The first of these layers has 128 neurons. The output layer has 3 neurons, corresponding to the number of classes in the data. Code for adding the fully connected layers to the model is:

```
model.add(Dense(128, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

5.4.5. CNN model training and testing

For the task of audio classification of target sounds, the selection of an appropriate dataset is a critical step, as it determines the quality of model training. Publicly available datasets were used to form the training, validation, and test datasets. These datasets contain three main classes of audio signals: the sound of the Parrot Mambo drone, the sound of the Parrot Bebop drone, and other sounds that are not drone sounds.

Audio files from the selected datasets were converted into audio files with the same parameters. The parameters of the audio files included in the training, validation, and test datasets are given in Table 5.1.

Table 5.1 Parameters of audio files

Sampling frequency	Bitrate	Number of channels	Audio file format
16KHz	16KB/s	mono	wav

Source code of a method for loading audio data into memory and converting it to mel-spectrograms is shown below:

```
def load_audio_files(directory, label):
    audio_data = []
    labels = []
    print("start")
    for filename in os.listdir(directory):
        if filename.endswith(".wav"):
            filepath = os.path.join(directory, filename)
audio, sr = librosa.load(filepath, sr=None)
            mel_spectrogram=librosa.feature.melspectrogram(y=audio,sr=sr,n_mels=128, fmax=8000)
            audio_data.append(mel_spectrogram)
            labels.append(label)
    return audio_data, labels
```

For the experiments, the converted audio files were divided into several smaller segments by defining time intervals at which the audio fragment will be segmented. This will allow the deep learning algorithm to learn features more accurately compared to feeding the entire recording at once. Another goal of segmentation was to optimize model training for real-time deployment, where the time required for detection and identification is critical. The experimental data from datast were taken as a basis for determining the impact of audio segment size on overall performance. It contains

information about different segment sizes, such as one second, two seconds, and five seconds. Heuristic observations in this source indicate that segmentation by one second outperformed other time intervals.

This approach may result in the loss of some features from the original audio fragments, but at the time of the experiments, this method was the only available practically proven method for deep learning on audio input with conversion of audio fragments into spectrograms. Various features are then extracted from the resulting spectrograms by an algorithm for training deep learning models. Fig. 5.12 shows an example of a one-second audio clip containing the sound of a drone and an audio clip of random noise, such as typing.

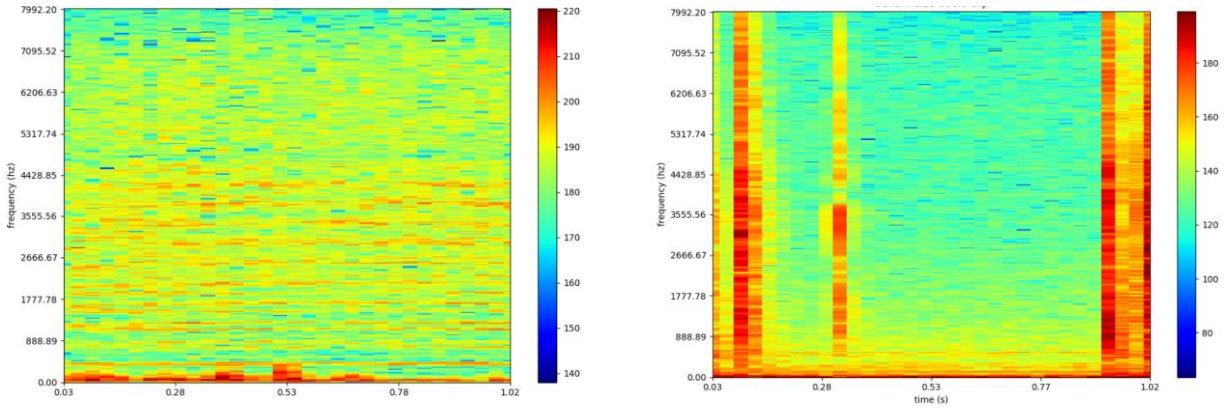


Fig. 5.12. Spectrograms of a segment with drone sound and random noise.

The results of model training are shown in Fig. 5.13.

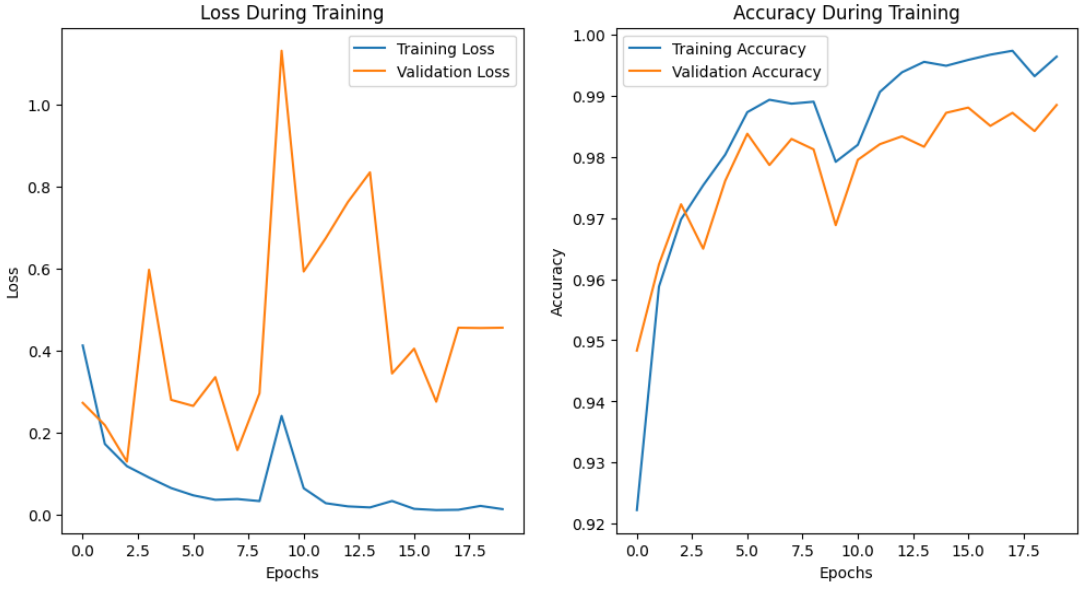


Fig. 5.13. Model training result.

During model training, the training loss gradually decreases, indicating effective model training. The validation loss has significant fluctuations, especially at the beginning of training, but stabilizes later. This may indicate possible overtraining, which was avoided by the end of the epochs. The training accuracy gradually increases and reaches high values, indicating that the model has "remembered" the training data well. The validation accuracy also shows good results, reaching 98-99%. This indicates that the model is able to generalize knowledge on new data, although some fluctuations may indicate small problems with generalization.

The results of model testing are shown in Table 5.2.

Table 5.2 CNN model testing results

Class	Accuracy	Response	F-measure
Mambo	93%	0.90	0.91
Bebop	96%	0.95	0.96
Other sound	99%	1	1

The results of the CNN model testing (Table 5.2) show that the F-measure metrics for all classes exceed 0.90 confirming a good balance between precision and recall.Аналіз результатів

The results of the study other models on methodology similar to the CNN are presented in Table 5.3. Optimization was performed using Nvidia TensorRT.

Table 5.3 ANN models comparison results

Model	Processing time before optimization, ms	Processing time after optimization, ms	Accuracy, %	GPU memory usage, MB	Power consumption, W
ResNet-18	42	17	89.3	256	6
GRU	81	30	85	512	7.7
AST	125	55	92	1024	9

Based on the analysis of neural network training methods for audio recognition, it was found that the use of different architectures, such as CNN, RNN and transformers, allows achieving high efficiency for specific audio signal classification tasks. CNNs demonstrate high efficiency in extracting local frequency features from audio signals, such as spectral patterns or harmonics. They are especially useful for tasks where sounds have distinct frequency characteristics, such as drone engine noise. RNNs, including their modification GRU, specialize in processing sequential data, such as audio signals. They are able to take into account temporal dependencies, which is critical for drone sounds, where the context of frequency changes is important. Transformers, using the attention mechanism, allow for efficient modeling of both local and global dependencies in the signal. They have high scalability and the ability to process large data sets.

The use of the Python programming language ensures efficient implementation of computational tasks for neural networks due to support for optimized libraries and the ability to work with parallel computing. Preliminary results of training and testing ANN models on Nvidia Jetson Nano demonstrate high efficiency, in particular CNN models, which provide processing speed and accuracy of more than 90% even for limited computing resources. Optimization using Nvidia TensorRT has achieved a 50% reduction in data processing time.

For optimal solution of tasks related to audio processing on limited computing resources, it is desirable to use combined approaches that take into account the advantages of each architecture.

5.5. Video data processing and recognition

5.5.1. Video data processing

Video recognition is an important link, after audio monitoring, in systems for protecting against aerial targets using drone swarms. In critical infrastructure protection systems, small and medium-sized aircraft can be considered as potential detection targets. The detection problem is exacerbated by the diversity of different types of drones, such as quadcopters and octocopters, each

of which has unique flight patterns. Traditional video detection systems often have difficulty identifying and tracking these agile machines, especially smaller models that can manoeuvre with a low radar plane. This breakthrough is made possible by the YOLOv8 ANN model, which is the latest version of the famous YOLO (You Only Look Once) series and is famous for its lightning-fast object detection capabilities. YOLOv8 is responsible for identifying and tracking enemy drones in real time, combining advanced machine learning techniques with the urgent need for aerial surveillance [13].

Current algorithms and approaches for using YOLOv8 for drone detection in video streams provide computational efficiency and high accuracy in solving aerial target detection problems. First of all, they involve the use of a transfer learning paradigm, initializing YOLOv8 with weights pre-trained on a comprehensive dataset to take advantage of the pre-established feature detection capabilities.

Further fine-tuning on a drone-specific dataset, including different UAV classes and contextual backgrounds, aims to enhance the model's discriminative ability. The fine-tuning process involves careful optimization of hyperparameters, seeking an economical balance between detection latency and accuracy. Validation is performed using stratified k-fold cross-validation to assess generalizability to unseen data.

This iterative approach to training, based on constant performance monitoring using loss metrics and mAP scores, ensures that the model is improved to optimal operational efficiency. In the future, the main focus will be on detecting drones with an average size of 347.5×283×107.7 mm (L×W×H) and a speed not exceeding 15 m/s. The input parameters of the detection objects were selected taking into account the fact that the most common drone on the battlefield, the DJI MAVIC 3, has the following characteristics. The IMX219 camera with a sensor resolution of 3264×2464 is used as the main means of obtaining the video stream, which provides an average possible detection range of 100 m. When training and testing the model, an attempt is made to fill the set of raw data with images with different types of environments: from empty clear skies to urban landscapes. The model's performance is also tested in different environmental scenarios to analyse which ones are the most favourable and in which ones the model has difficulty in the detection process.

5.5.2. YOLOv8 architecture and functionality

YOLOv8, an advanced iteration in the YOLO series, utilizes a deep CNN for efficient object detection. The architecture comprises three main components: the backbone, the neck, and the head [12].

- Backbone: The backbone is designed for feature extraction. It uses a series of convolutional layers to process the input image and extract a set of feature maps at different scales. This part of the network is critical for identifying various attributes of objects within the image.
- Neck: The neck connects the backbone to the head. It processes the feature maps from the backbone, refining and recombining them. This step is crucial for preparing the features for precise object detection.
- Head: The head of the network is responsible for making predictions. It uses the processed feature maps to predict bounding boxes, object classes, and abjectness scores. The head employs a set of anchor boxes, predefined shapes that help the model in detecting objects of various sizes and shapes efficiently.

The full YOLOv8 architecture is presented in the Fig. 5.14. The architecture uses a modified CSPDarknet53 backbone. A spatial pyramid pooling fast (SPPF) layer accelerates computation by pooling features into a fixed-size map. Each convolution has batch normalization and SiLU activation. The head is decoupled to process objectless, classification, and regression tasks independently [13].

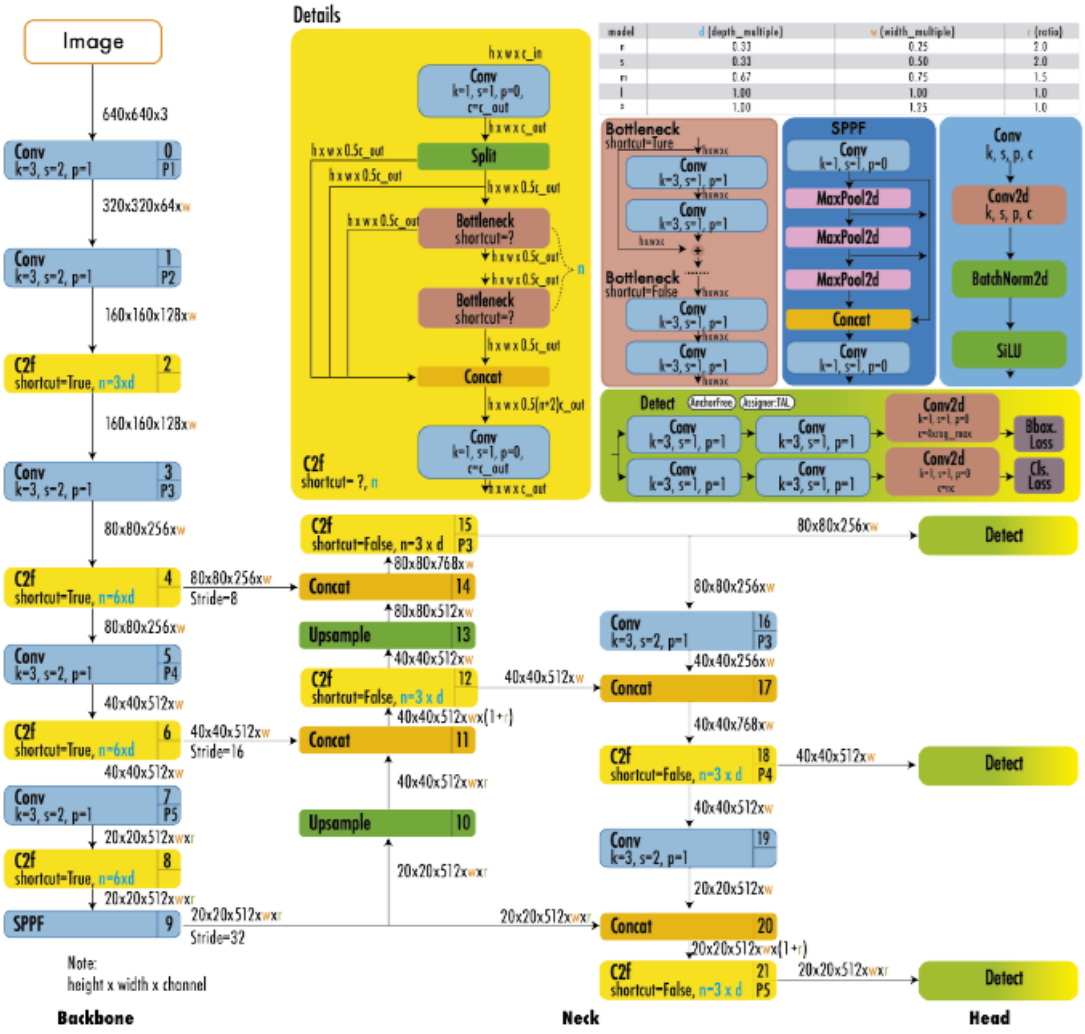


Fig. 5.14. The full YOLOv8 architecture [13].

YOLOv8 also provides a semantic segmentation model called YOLOv8-Seg model.

The backbone is a feature extractor, followed by a C2f module which corresponds to the traditional YOLO neck architecture. The C2f module is followed by two segmentation heads, which learn to predict the semantic segmentation masks for the input image. The model has similar detection heads to YOLOv8, consisting of five detection modules and a prediction layer. The YOLOv8-Seg model has achieved state-of-the-art results on various object detection and semantic segmentation benchmarks while maintaining high speed and efficiency.

YOLOv8 can be run from the command line interface (CLI), or it can also be installed as a PIP package. In addition, it comes with multiple integrations for labelling, training, and deploying.

5.5.3. Training YOLOv8 model to detect drones

The development of a robust drone detection system via YOLOv8 entailed a systematic training methodology, initiated by curating a dataset using the Roboflow platform, which streamlined the collection and pre-processing of diverse image of drones (Fig. 5.15). The platform provides convenient user interface (UI) to automate common steps (e.g., collecting training data, assigning classes, annotating objects.) for training object detection models. The figures below demonstrate the sequence of steps to train ANN using Roboflow UI.

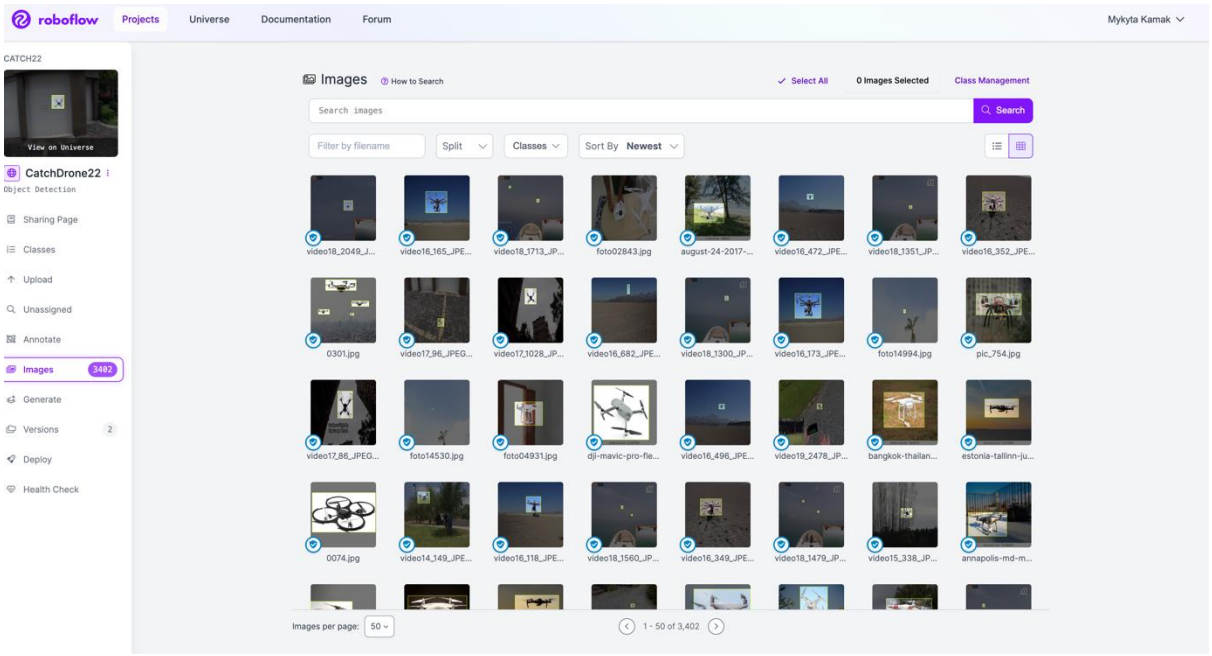


Fig. 5.15. Drone images within Roboflow drone's dataset.

In object detection tasks, it can be advantageous to start with a singular classification category when there is insufficient data to reliably differentiate between subcategories or when the distinctions are not critical to the project's goals. Therefor, the next step in preparing the training dataset for ANN model is annotation of objects we want to detect. For this, each previously uploaded drone image was carefully annotated to ensure accurate training of the model (Fig. 5.16).

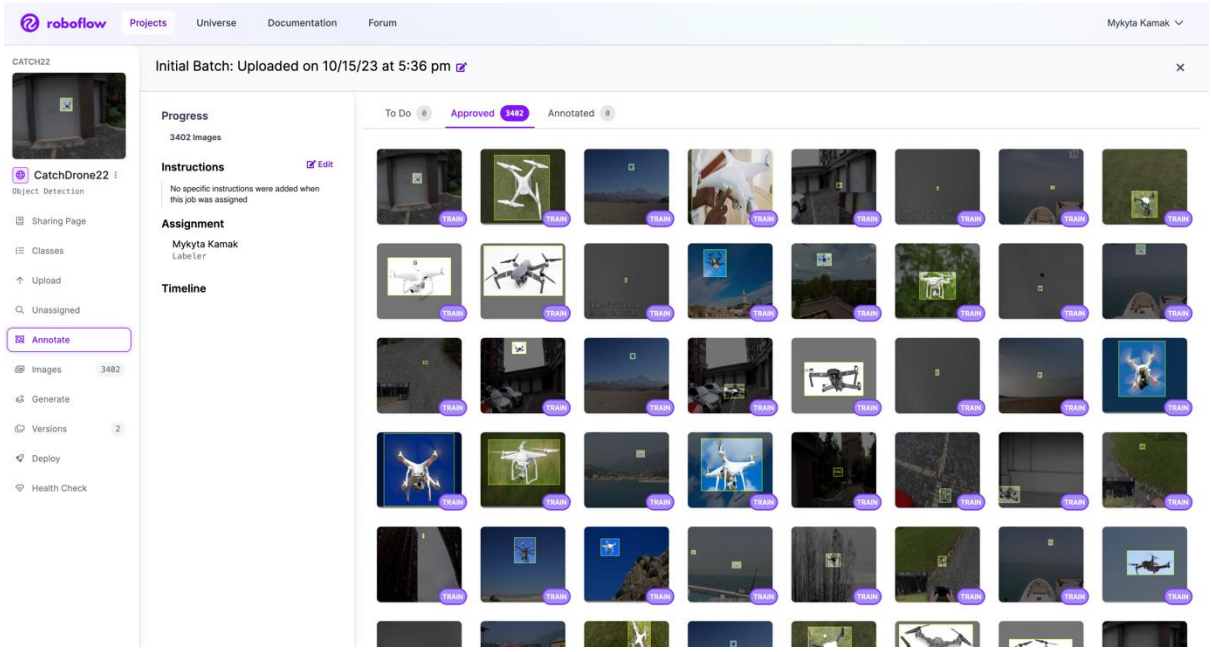


Fig. 5.16. Annotation functional within Roboflow UI

The training process, scripted in a Python, utilized the Roboflow library to retrieve the dataset and implemented a series of commands to train the YOLOv8 model.

Key parameters such as the number of *epochs* and *imgsz* (image size) were configured to optimize model performance for real-world application (Fig. 5.17). Here, the 100 epochs were set to train the model according to recommendation of YOLOv8 developers.

```
[ ]  !yolo task=detect \
     mode=train \
     model=yolov8s.pt \
     data={dataset.location}/data.yaml \
     epochs=100 \
     imgsz=640
```

Fig. 5.17. Key parameters of train process

The image size (imgsz) is also an important hyperparameter in object detection models. It determines the resolution at which the input images are processed. The size of 640x640 pixels is chosen because it's a common standard that balances the need for detail (to detect and classify objects accurately) with computational efficiency (to train and run the model quickly). It's large enough to capture relevant features of the objects but not so large that it would require excessive computational power or memory, which could slow down the training process or make the model less deployable in environments with limited resources that is very important for us, because as general aerial surveillance tools need to be portable, which imposes a resource constraint on them.

So, the complete roadmap of model consists of:

- data collection: acquire diverse drone images;
- data annotation: Manually annotate images with bounding boxes;
- data pre-processing: resize, normalize, and augment data using Roboflow;
- model configuration: set YOLOv8 parameters and hyperparameters;
- model training: train YOLOv8 with the prepared dataset;
- model evaluation: validate using metrics like mAP, precision, and recall;
- Model Optimization: Tune parameters based on validation results;
- deployment: deploy the trained model for real-time detection.

The results of the model training process for the detection of UAVs, specifically quadcopters and octocopters, include both *quantitative* and *qualitative* in nature and divided in two samples: training and validation. Our full dataset of 8220 images (4349 images of quadcopters and 3871 images of octocopters) was divided into samples in the following ratio: 60% for training, 20% for validation and 20% for testing according to best practices in ANN training for small sets (up to a few thousand examples).

The training process results are demonstrated by series of graphs of the model's performance over the training period (see Fig. 5.18).

Loss Metrics. The graphs for *train/box_loss*, *train/cls_loss*, and train/df1_loss alongside their validation counterparts (*val/box_loss, val/cls_loss, and val/df1_loss*) exhibit a typical downward trend, indicative of the model's improving ability to correctly identify and classify objects within the training dataset. As the number of epochs increases, the smooth lines illustrate a consistent reduction in loss, reflecting the model's increasing accuracy.

The *box_loss* graphs calculate the average squared difference between the predicted bounding box coordinates and the true bounding box coordinates across all *N* predictions according to such formular:

$$\text{box\_loss} = \frac{1}{N}\sum_{i=1}^{N}\left(y_{\text{true}_i} - y_{\text{pred}_i}\right)^2. \quad (5.3)$$

In (5.3) each prediction i involves comparing the vector of predicted coordinates y_{pred_i} (such as the centre, width, and height of a box) against the true coordinates y_{true_i} This value measures the accuracy

of the location and size of the bounding boxes prediction. Minimizing Boxloss is crucial for improving the precision of object localization in images.

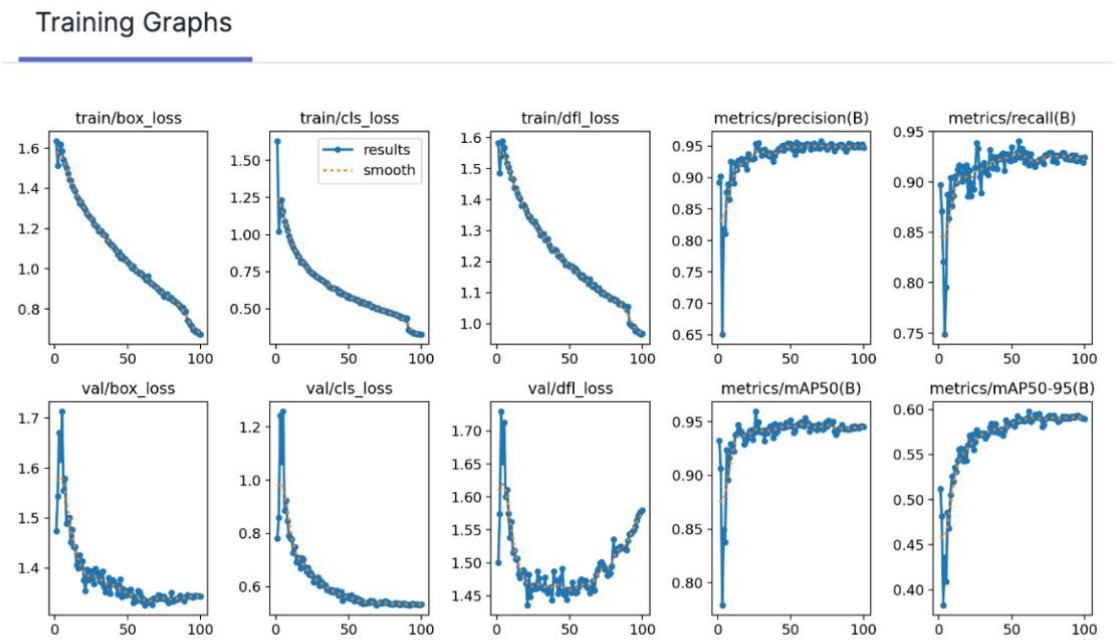


Fig. 5.18. The graphs of model performance over the training period

The *cls_loss* graphs is calculated based on Cross-Entropy Loss function. It calculates the average negative log probability of the correct class C across all N images. Here:

- $y_{\text{true}_{i,c}}$ is a binary indicator (0 or 1) if class label C is the correct classification for observation i;
- $y_{\text{pred}_{i,c}}$ is the predicted probability that observation i belongs to class C.

The cross-entropy loss is crucial for classification tasks as it penalizes incorrect class predictions. It's especially sensitive to confident wrong predictions, which is beneficial in training classifiers to output probabilities close to 0 for wrong classes and close to 1 for the correct class. The cls_loss metric is calculated by the next formalar:

$$\text{cls\_loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{\text{true}_{i,c}} \log\left(y_{\text{pred}_{i,c}}\right) \quad (5.4)$$

Performance Metrics. Precision and recall graphs, *metrics/precision(B)* and *metrics/recall(B)*, remain high throughout the training epochs, maintaining values close to 1 (0.94 based on the graphs). This suggests that the model has a high probability of correctly identifying UAVs when they are present and a low rate of false negatives. The Mean Average Precision (mAP) at different Intersection over Union (IoU) thresholds (*metrics/mAP50(B) and metrics/mAP50-95(B)*) are robust, which is critical for applications where the correct drone detection is paramount.

Precision is a metric that calculates the ratio of correctly predicted positive observations to the total predicted positives (the sum of true positives (*TP*) and false positives (*FP*)) [13]. It's important because it shows how reliable the model's predictions are: the higher the precision, the more you can trust the model's positive predictions. It is particularly important in domains where the cost of false positives is high.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (5.5)$$

In the recall metric *FN* stands for False Negatives - the count of positive cases that were incorrectly predicted as negative. This metric measures the ability of a model to find all the relevant cases (all positive samples).

$$\text{Recall} = \frac{TP}{TP+FN} \tag{5.6}$$

The mAP at a single IoU threshold (like 0.50) is the mean of the Average Precision (AP) scores for all classes Q. AP for a single class is computed by integrating the area under the precision-recall curve for that class. mAP is an essential metric in object detection because it accounts for both the precision and recall of the predictions, providing a balanced view of the model's overall performance. For our model it is close to 0.95.

$$\text{mAP} = \frac{1}{Q}\sum_{q=1}^{Q} AP_q \tag{5.7}$$

This variant of mAP is averaged over a range of IoU thresholds, typically from 0.50 to 0.95 in increments (like 0.05). It calculates the average mAP across these thresholds, representing a more robust measure of the model's performance at different levels of bounding box overlap. This comprehensive metric is important as it ensures the model is consistently good across various degrees of detection strictness, which is valuable for practical applications where different conditions/environments may require different precision levels for the detected objects. Based on our testing, we obtained a value for this metric that is close to 0.59.

$$\text{mAP}_{\text{IoU=0.50:0.95}} = \frac{1}{T}\sum_{t=1}^{T} mAP_{\text{IoU}_t} \tag{5.8}$$

*5.5.4. **Testing the YOLOv8 model in different environments***

Testing results provide a visual affirmation of the YOLOv8 model real-world efficacy. The example outputs from the test set feature the model's predictions superimposed on images of various environments.

The ability of the model to detect drones in different environmental conditions is extremely important for its practical application in realistic systems, especially in the military sector, as the accuracy of the model can affect the outcome of a decision or the lives of personnel and the integrity of property. Therefore, we have conducted a number of tests. Below is a photo report of some tests to illustrate the process, as well as a table describing the other tests that were not included in the photo report for reasons of keeping the material short.

The Fig. 5.19 demonstrate the used Roboflow UI for the test trained YOLOv8 model.

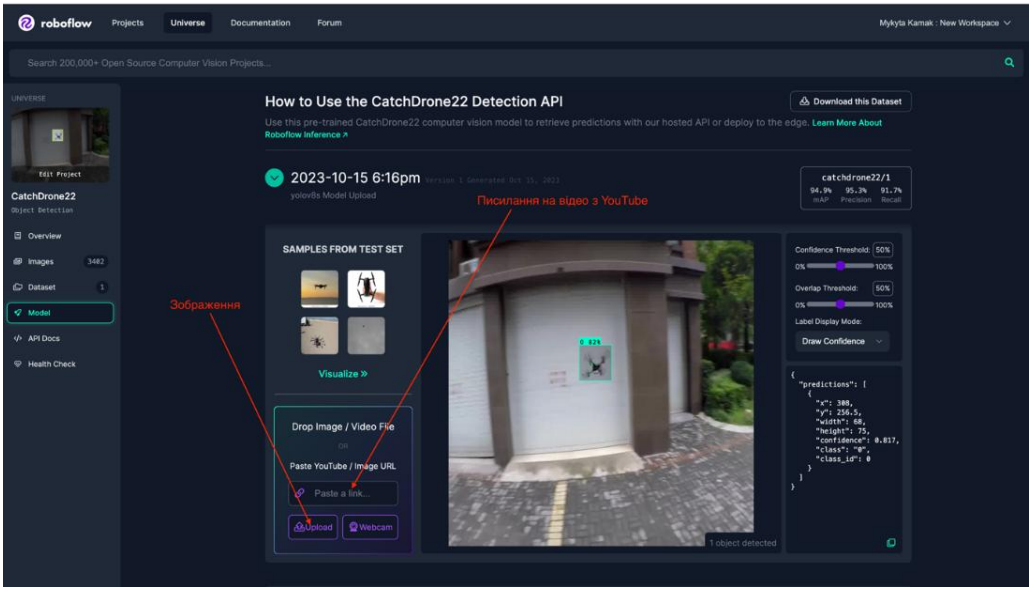


Fig. 5.19. Roboflow UI for model testing.

Red arrows indicates that we have ability to use images or video stream as input parameters of the model to test in in different environments.

The Fig. 5.20 –5.23 provide the photos of test report. Real values of such parameters as confidence, class of detection, boxes coordinate can be find on each figure in the low right corner in red box.

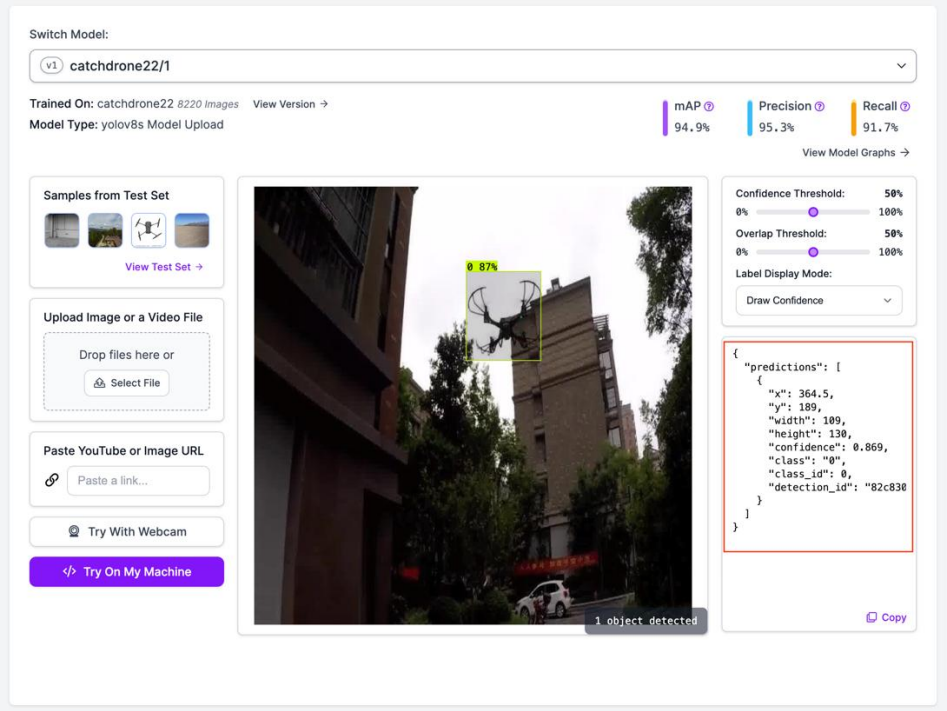


Figure 5.20. Result of drone detection in an urban environment.

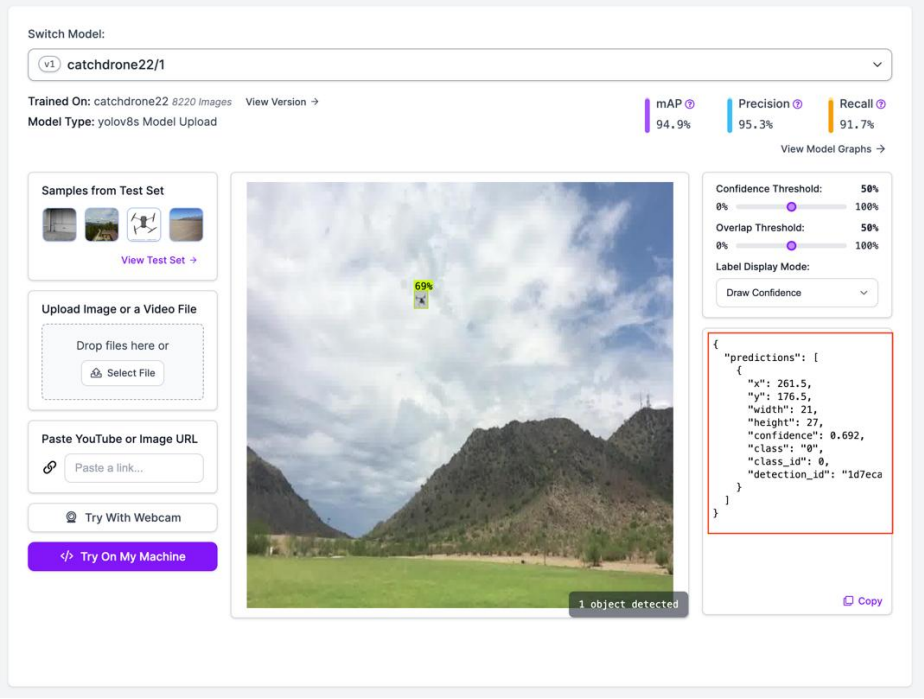


Fig. 5.21. Result of drone detection in a mountains environment.

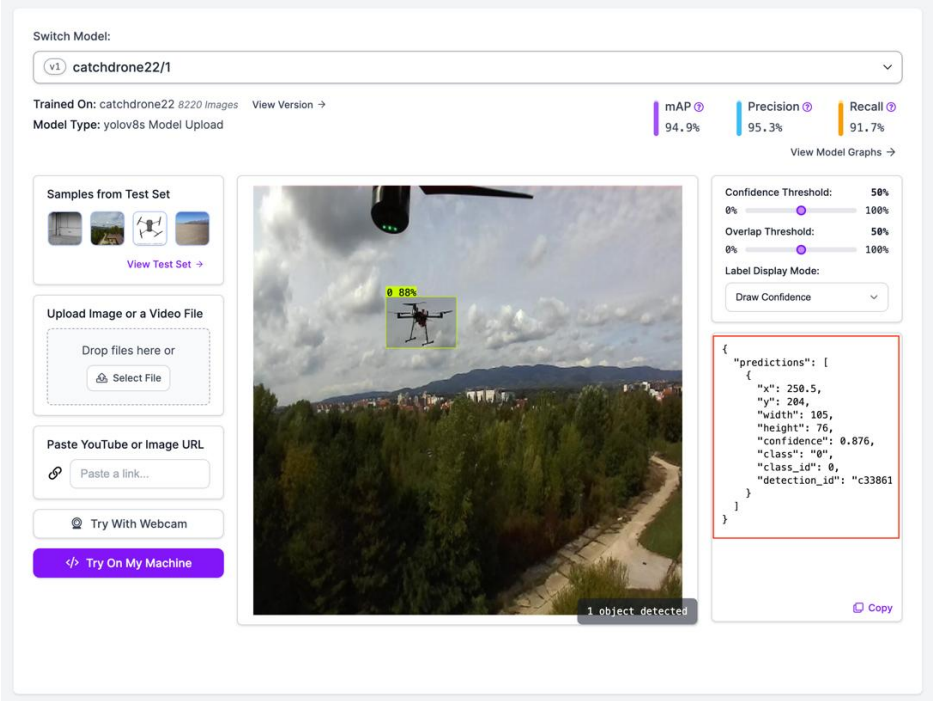


Fig. 5.22. Result of drone detection in a mix environment from another drone camera

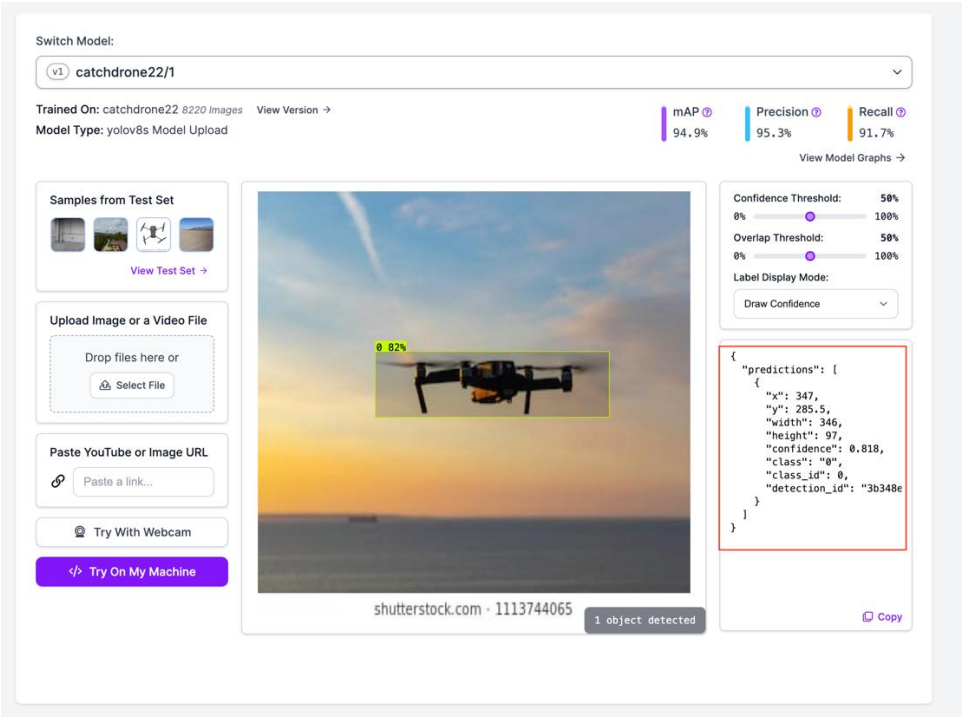


Fig. 5.23. Result of drone detection in a sea environment.

The Table 5.4 contains quantity results of drone detection tests for different environments. These results show:

– the best result of detection can be gotten in case when drone is in high contrast background such as sky, sea, or earth. It is expected because in this case the model can find more futures of object and distinguish it from a background;

– Detection of drones on earth background take worse result then sky because in most out test cases drone has black or dark colure and it increased the chances that the drone's features would blend in with the color of the ground;

– Detection in low light was real challenge for the model. We should think about how we can increase confidence for this env in the next version of the model.

Table 5.4 Results of drone detection test in different environments

Environment	Count of tests	Avg confidence in %
Urban	30	81
Mountains	20	68
Mixed	50	79
Sky	200	89
low light sky	100	53
low light urban	20	65
Sea	20	82
Earth (view from top to down)	40	73

5.6. Inter-drone communication

5.6.1. Mesh Network for communication channel organization

One of the key components of effective drone swarm control is a reliable and efficient network subsystem that provides communication between drones. Effective communication plays a critical role in coordinating movement, data transmission and task performance. The use of a Mesh network [14] based on XBee radio modules [15] provides a convenient and effective mechanism for ensuring this key aspect, allowing drones to communicate with each other directly and reliably even in difficult conditions. This approach guarantees optimal use of network resources and maximum efficiency of drone swarm control.

The Mesh network is designed for the creation of wireless networks with self-extending and data routing capabilities through network nodes. The primary objectives of this subsystem include:

1. **Network formation.** The Mesh Networking Subsystem enables devices to automatically discover each other and establish a continuous network without the need for centralized control.
2. **Data routing.** When a device transmits data, the Mesh Networking Subsystem determines the most optimal route for data delivery to its destination. This implies that data can be automatically routed through various network nodes to ensure efficient delivery.
3. **Autonomous network expansion.** When a new device joins the network, the Mesh Networking Subsystem automatically detects and integrates it into the network, expanding its coverage area.
4. **Automatic Network Recovery**. In the event of a communication loss with one of the network nodes, other nodes can automatically reroute data through available paths, ensuring network resilience.

Regarding drone swarm application, the Mesh network can be utilized for communication between two devices or for establishing a network with numerous devices.

For a clearer understanding of Mesh operation, let us consider a simple two-device interaction scenario:

- when the first device wishes to send a message to the second, it forms a data packet and transmits it to its immediate neighbor within the Mesh network;
- When the packet reaches its destination, the second device receives the message and can send an acknowledgment of successful reception back to the first device.

This scenario is schematically illustrated in Fig. 5.24.

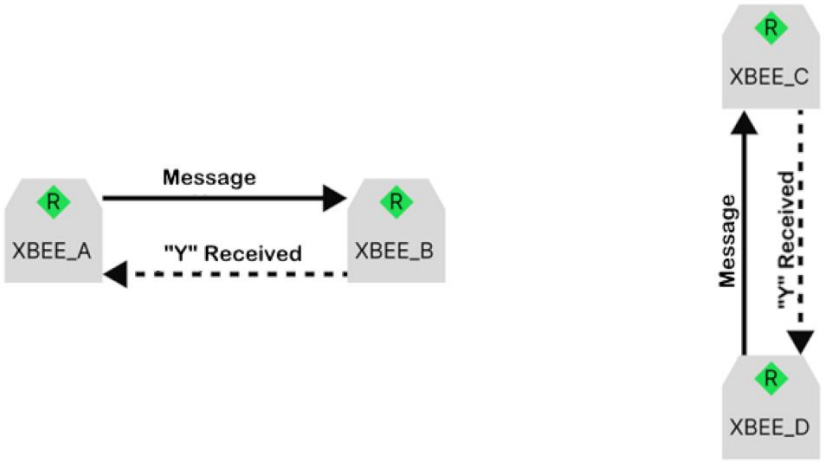


Fig. 5.24. Scenario of communication between two devices.

In the case of a broadcast message to all nodes within the Mesh network, the process can be described as follows:

- the initial node wishing to broadcast a message to all other nodes constructs the data packet and transmits it to its immediate neighbours within the Mesh network;
- these immediate neighbours receive the message and subsequently forward it to their respective neighbours, who, in turn, perform the same action;
- the message is automatically propagated through the network nodes until it reaches all nodes;
- each node that receives the message can transmit an acknowledgment of its reception back to the originating node. This allows the originating node to determine which nodes have successfully received the message.

This scenario broadcasting a message is illustrated in Figure 5.25.

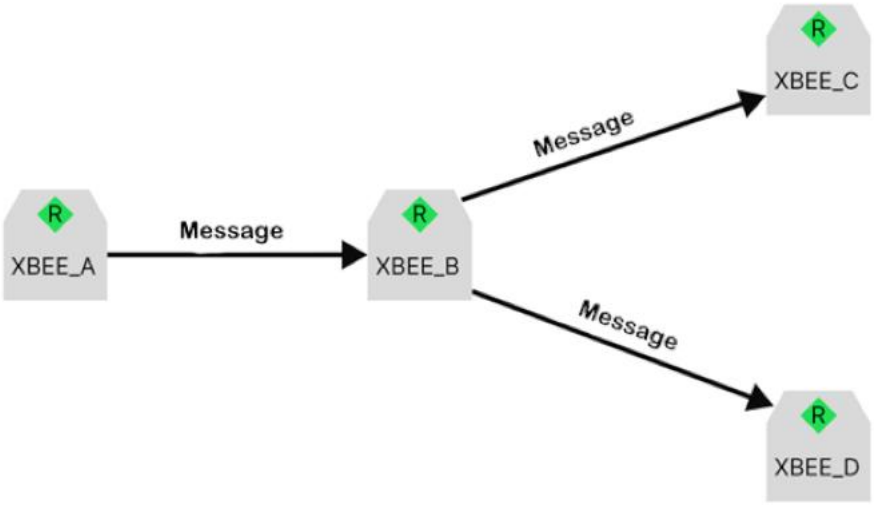


Fig. 5.25. Scenario of broadcasting a message to all devices.

Consequently, the message is autonomously propagated across the Mesh network nodes to its intended recipient, guaranteeing reliable delivery irrespective of direct sender-receiver connectivity.

5.6.2. Zigbee wireless communication module

To build a Mesh network based on ZigBee technology, the Digi Mesh Kit is used, which is supplied as follows:

1. Digi XBee Grove Development Board which allows to easily connect XBee modules to any device or microcomputer via the Grove interface.
2. Digi XBee 3 Zigbee SMT Modules. These XBee 3 Zigbee SMT modules are programmable which can operate in 802.15.4 and DigiMesh standards. They provide wireless connectivity and communication between devices in a Zigbee network.
3. Micro-USB Cables which are used to connect the Digi XBee Grove Development Board to a USB port on a host device such as the Jetson Nano.
4. Antenna: The antennas connecting to the XBee modules provide optimal wireless communication in the network. They help ensure stable and long-range communication between devices.

XBee SMT Grove Development Board layout is shown in Fig. 5.26.

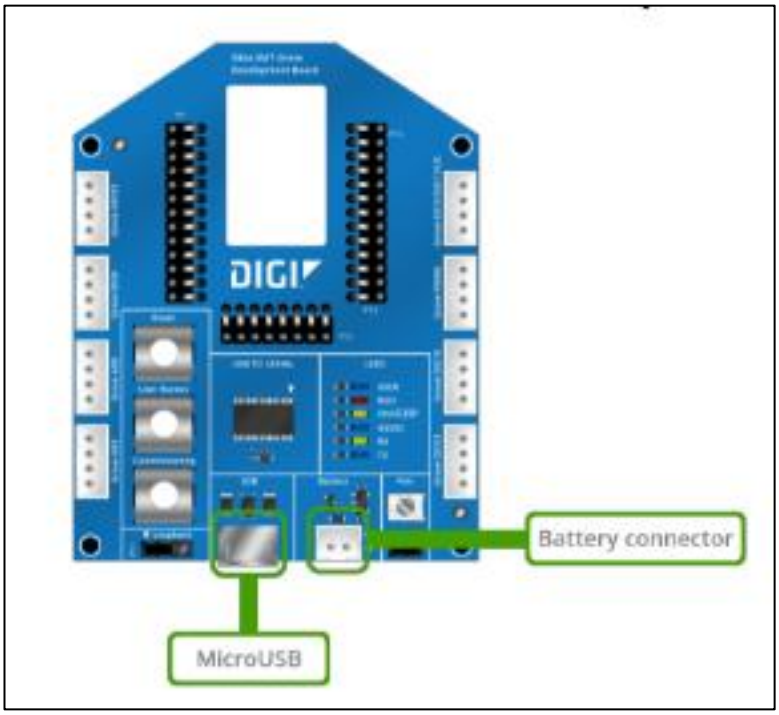


Fig. 5.26. Digi XBee SMT Grove Development Board layout.

Digi XBee 3 modules [15] provide rapid and reliable wireless connectivity for Original Equipment Manufacturers (OEMs). Built on cutting-edge technology, these modules offer the flexibility to switch between various frequencies and wireless protocols. Digi XBee 3 802.15.4 modules are particularly well-suited for applications demanding low latency and predictable communication.

In terms of size and flexibility, the new Digi XBee®3 micro form factor, measuring 13 mm x 19 mm, enables the creation of more compact and portable devices. It serves as a unified module for all protocols, including Zigbee, 802.15.4, DigiMesh, and BLE. All these protocols can be configured via Digi XCTU.

Digi XBee 3 modules eliminate the need for external microcontrollers, allowing for the creation of intelligent end nodes using MicroPython. They also support Bluetooth Low Energy for beacons, connection to Bluetooth sensors, and local configuration using the Digi XBee Mobile app. Moreover, they incorporate Digi TrustFence for inherent IoT security, providing a multi-layered approach to securing edge devices through the IoT gateway, ingress, and egress.

The appearance of the assembled module is shown in Fig. 5.27.

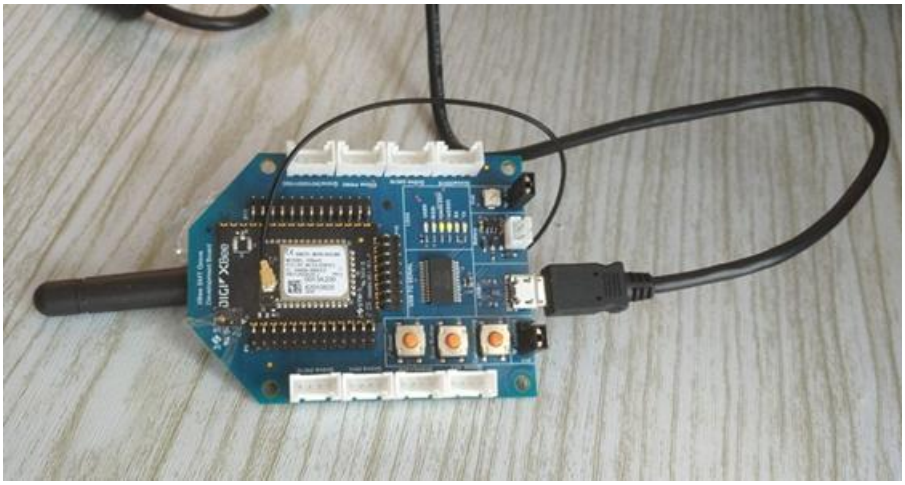


Fig. 5.27. Assembled module Digi Mesh Kit.

To integrate Digi Mesh Kit components with Jetson Nano, several key steps must be taken to ensure proper connection and functionality.

Upon examining Fig. 5.22, it becomes evident that the Jetson Nano can be connected to the XBee SMT Grove development board using the USB 3.0 connector on the Jetson Nano [10]. For this purpose, the microUSB port on the XBee SMT Grove development board is utilized. This establishes a serial connection between the Jetson Nano and XBee SMT Grove via USB.

At the operating system level, Ubuntu running on the Jetson Nano recognizes the microUSB port on the XBee SMT Grove as a serial port. In Ubuntu, serial ports are typically represented as files in the /dev directory with the tty prefix. For instance, if the XBee SMT Grove is connected to the Jetson Nano via microUSB and recognized by the operating system, it might be accessible at /dev/ttyUSB0 or a similar address.

This enables data transmission between the Jetson Nano and the XBee SMT Grove development board through the serial port, using standard operating system commands and interfaces. Such a connection ensures reliable data exchange and facilitates convenient device control from the software running on the Jetson Nano.

5.6.3. Communication module software

Digi provides the Digi XBee Python library, which is well documented and has an active community that facilitates rapid code development. At the same time, Jetson Nano officially supports Python, which provides easy integration with its development environment and specialized libraries for image processing, machine learning, and sensor work.

The list of main developed functions include:

- *handleconnect.* This function creates an instance of the device and starts it on the specified port;
- *handle_send function.* This function is directly responsible for sending messages. After sending a message, it is planned to start a resend thread for more reliable communication between nodes;
- *resend_message.* This function is responsible for resend thread. It receives a list of nodes to send a message. If the recipient is still in this list, message send to it again;
- *message_callback.* The activation of callback is implemented in the library provided by Digi. All that remains is to implement this function for our needs: receive the message,

identify the sender, respond to him that the message has been received, and then forward the message to other nodes.

For example, Fig. 5.28 presents the algorithm of resend_message function operation.

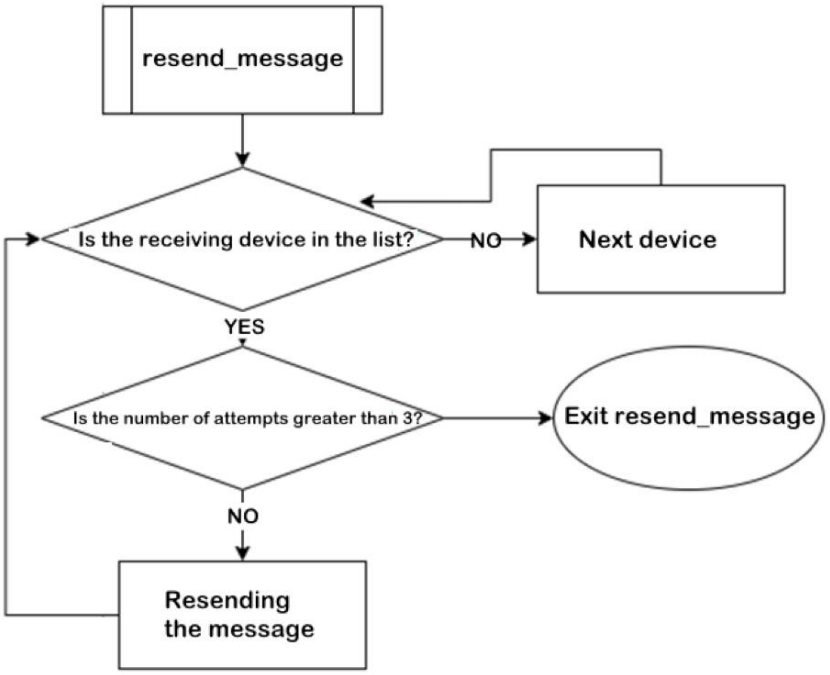


Fig. 5.28. Algorithm of the resend_message function operation.

Since the Digi library is more geared towards IoT devices than drone communication, it is needed to implement some functionality independently. The main difference is that IoT devices are mostly static.

For example, below is definition of Communicator class:

```
class Communicator:
    def __init__(self):
        self.device = None
        self.xbee_network = None
        self.received_message_ids = set() # List of IDs
        self.message_count = 0 # Current number
        self.clear_list_after = 5  # Limit number
        self.current_discovered_devices = set()
        self.devices_to_send = {}
        self.timer_flag = False
```

5.6.4. Initial configuration of the communication module

The initial configuration of the communication module is done using the XCTU software. XCTU is a user-friendly program from Digi International that is used to configure and manage Digi XBee modems. The procedure for configuring devices in XCTU under Windows is described below.

To commence, the device is connected to the PC using a USB cable. Within the XCTU software, the "Discover radio modules" function is activated. This triggers a scan of the designated ports to identify any connected Digi modules, as it is shown in Fig. 5.29.

Upon establishing a connection, the device configuration was initiated. The following parameters were set for the device named "ROUTER":

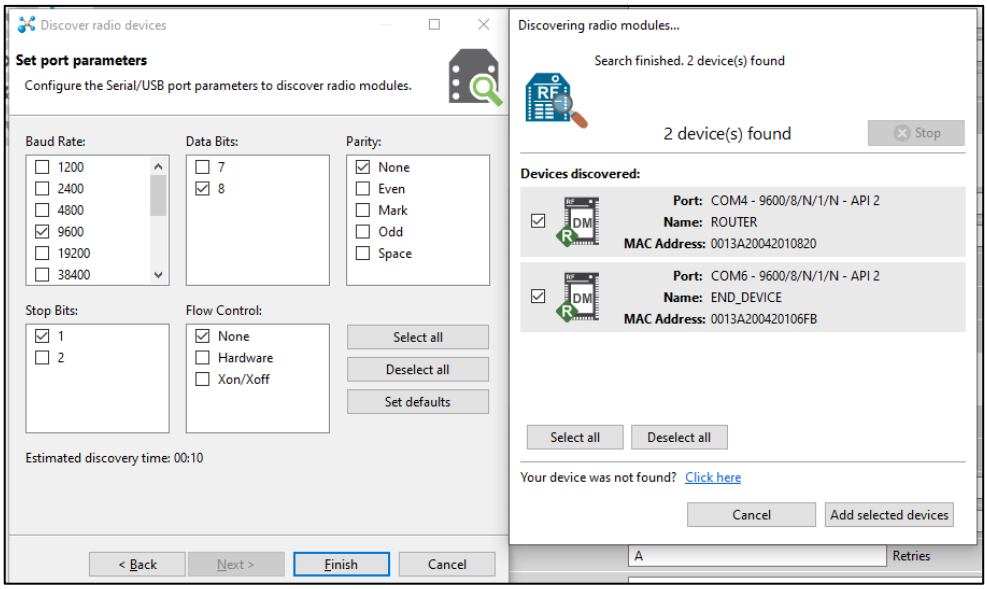


Fig. 5.29. Scanning Digi modules on a PC.

- Personal Area Network (PAN) ID: Set to 2022, this identifier uniquely identifies the specific wireless network to which the device belongs. All devices within a network must share the same PAN ID for communication;
- Node Identifier: Assigned the value "ROUTER", this parameter provides a unique name for the device within the network, aiding in device identification;
- Digi Device Type Identifier: Set to 14002, this parameter is used to identify the specific type of Digi device and can be used to differentiate between various XBee-based products during network scanning;
- Network Discovery Timeout: Set to 20 (in manufacturer-defined units), this parameter determines the duration the device will wait for a response from other devices during the network discovery process;
- Encryption Key: Set to 1A8, this cryptographic key is used to secure data transmission within the network, ensuring confidentiality and integrity;
- Broadcast Radius: Set to 3E1151, this parameter defines the maximum number of hops a broadcast message can traverse;
- Guard Time: Set to 96F5, this parameter is used for synchronization between data transmissions.

A connectivity verification was conducted to assess the integrity of the established link. By selecting a specific module and initiating a device discovery procedure (Fig. 5.30), connected devices are identified.

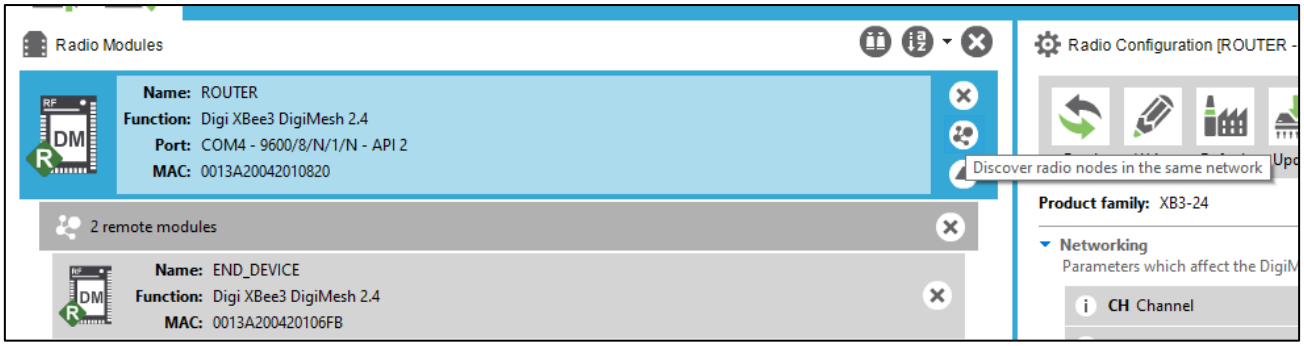


Fig. 5.30. Search for connected devices.

To evaluate link quality, a range test was subsequently performed. This involved selecting the target device and executing a standardized test sequence (Fig. 5.31).

Fig. 5.31. Result of the Range Test.

The test results yielded the following key metrics:

- Local RSSI: This metric quantifies the signal strength received by the local module, with higher (less negative) dBm values indicating stronger signals.
- Remote RSSI: This metric represents the signal strength received by the remote module.
- Percentage: This metric indicates the percentage of successfully transmitted packets, serving as a direct measure of link quality.

The observed Local and Remote RSSI values of -37 dBm suggest a robust signal strength between the modules. Furthermore, the 100% packet transmission success rate confirms the reliability of the link. The test results demonstrate that the link is configured correctly and functions optimally at the specified distance. Next, the device can be programmed.

5.6.5. Testing the communication module

To test the communication module, use the IDE VS Code and its terminal. To do this, the code is divided into three parts for three separate devices. Then it is run for execution. After that, the connect command is entered from the terminal. The result of this step is shown in Figure 5.32.

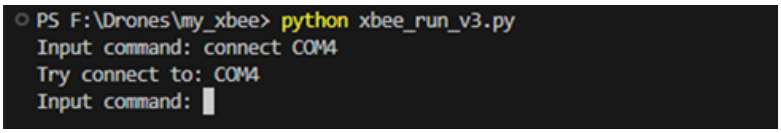


Fig. 5.32. Running the script and connecting to the serial port.

The result is a message about a successful connection to the specified port.

Next, the *list* command is entered to verify that the connection has been established. The result of this step is shown in Fig. 5.33.

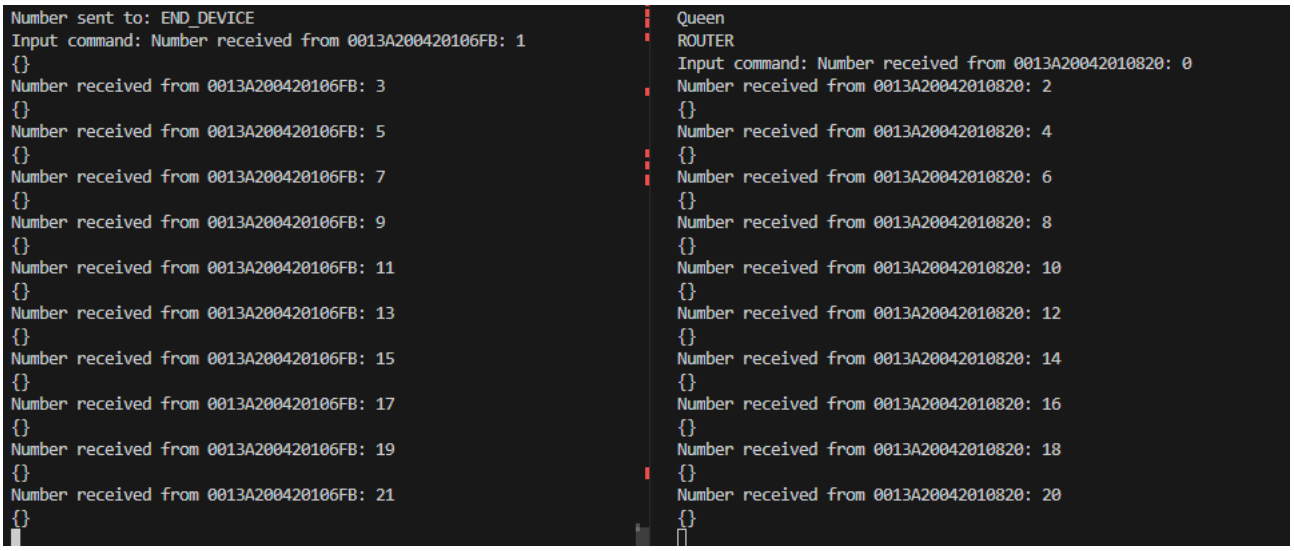


Fig. 5.33. Result of processing the *list* command

Next, the *send* command is tested for all devices. One example is shown in the Fig. 5.34.

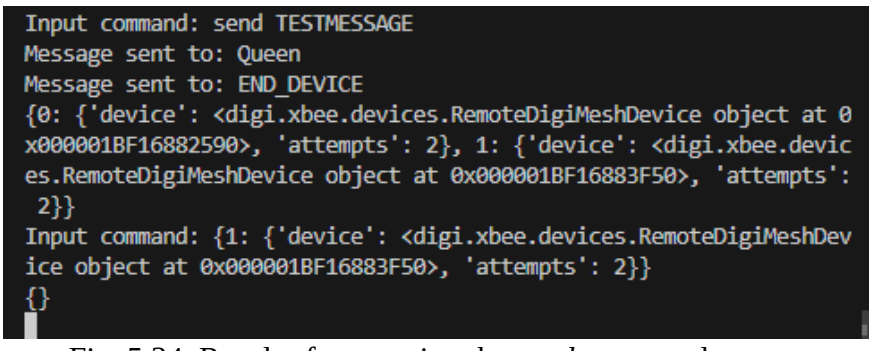


Fig. 5.34. Result of processing the *send* command

5.7. Flight controller interaction

5.7.1. Pixhawk flight controller features

Pixhawk flight controller is a key element for implementing the drone control system. When choosing a controller, the following tasks were set:

- Popularity of the model, which provides access to a large community and support;
- Availability of high-quality documentation to simplify the development process;
- the relevance of the model, which continues to be supported and produced by the manufacturer;
- Ability to connect a GPS module for spatial orientation;
- Availability of several telemetry ports for integration with other components;
- Optimal price that takes into account the project budget.

Among popular flight controllers, it is worth paying attention to the manufacturer Holybro and its Pixhawk series:

Pixhawk 4 – Launched in 2018, Pixhawk 4 is a reliable controller for a wide range of tasks. It has sufficient performance and a large number of ports, but with the advent of newer models, its technology is gradually becoming obsolete.

Pixhawk 5X – A high-performance model designed for complex projects with high demands on computational resources and connections. Pixhawk 5X supports significantly more peripherals but has a high price, as well as features that may be redundant for simpler tasks.

Pixhawk 6C – An updated version of Pixhawk 4, which retains its ease of use but with improved performance, modern ports, and better compatibility with new standards. Pixhawk 6C is a more affordable and optimized option for general tasks.

Based on the analysis of documentations, the following conclusions can be drawn for each model:

Pixhawk 4

- Pros: Easy to use, suitable for basic projects.
- Cons: Outdated processor and sensors, limited performance.
- Recommendation: Should only be considered for small budgets or simple tasks.

Pixhawk 5X

- Pros: High performance, large number of ports, support for the latest peripherals.
- Cons: High price, excess features for simple projects.
- Recommendation: Suitable for complex or industrial applications.

Pixhawk 6C

- Pros: Improved performance compared to Pixhawk 4, up-to-date sensors, affordable price, optimized for a wide range of tasks.
- Cons: May be less performant in complex scenarios compared to 5X.
- Recommendation: The best choice for most modern projects due to its balance of price, performance, and functionality.

Based on the above, the following conclusion can be drawn. The Pixhawk 6C is ideally suited for drone swarm tasks, as it combines modern technologies, sufficient performance and ease of use. It is an updated version of Pixhawk 4 that retains simplicity but eliminates the technical limitations of the older model.

The complete technical specifications of the Pixhawk 6C are presented below.

Processors & Sensors:

- FMU Processor: STM32H743;
  32 Bit Arm® Cortex®-M7, 480MHz, 2MB memory, 1MB SRAM;
- IO Processor: STM32F103;
  32 Bit Arm® Cortex®-M3, 72MHz, 64KB SRAM;
- On-board sensors:
  Accel/Gyro: ICM-42688-P;
  Accel/Gyro: BMI055;
  Mag: IST8310;
  Barometer: MS5611.

Electrical data:

- Voltage Ratings:
  Max input voltage: 6V;
  USB Power Input: 4.75~5.25V;
  Servo Rail Input: 0~36V;
- Current Ratings:
  Telem1 Max output current limiter: 1.5A;
  All other port combined output current limiter: 1.5A.

Mechanical data:

- Dimensions: 84.8 * 44 * 12.4 mm;
- Weight (Aluminum Case): 59.3g;
- Weight (Plastic Case): 34.6g.

Interfaces

- 16-PWM servo outputs (8 from IO, 8 from FMU) with hardware switchable 3.3V or 5V signal mode;
- 3 general purpose serial ports;
  Telem1 - Full flow control, separate 1.5A current limit;
  Telem2 - Full flow control;
  Telem3.
- 2 GPS ports;
  GPS1 - Full GPS port (GPS plus safety switch);
  GPS2 - Basic GPS port;
- 1 I2C port;
  Supports dedicated I2C calibration EEPROM located on sensor module;
- 2 CAN Buses;
- 2 Debug port;
  FMU Debug;
  I/O Debug;
- Dedicated R/C input for Spektrum / DSM and S.BUS, CPPM, analog / PWM RSSI;
- Dedicated S.BUS output;
- 2 Power input ports (Analog).

Other Characteristics

- Operating temperature: -40 ~ 85°c.

Placements of the Pixhawk 6C ports is shown in Fig. 5.35. In drone control system not all the ports are used, but only some of them:

- POWER1 - provides power for the controller;
- TELEM1 (full flow control) - used for communication with the onboard Jetson Nano computer (will be described later);
- TELEM3 (limited control) - provides antenna connection and communication with the remote control;
- GPS1 - for connecting the M9N GPS sensor;
- I/O PWM OUT - motor connection;
- Micro SD card slot - used for recording logs and analyzing them;
- USB - used for firmware debugging and controller configuration.

Single-board computer Jetson Nano transmits and processes a telemetry data, as well as control flight tasks in interaction with the controller. Besides, it performs the following tasks:

- analyses audio data from the microphone to identify acoustic events or commands;
- connects the camera for continuous image reading with subsequent real-time object recognition;

- provides communication with other agents via the Digi XBee radio module.

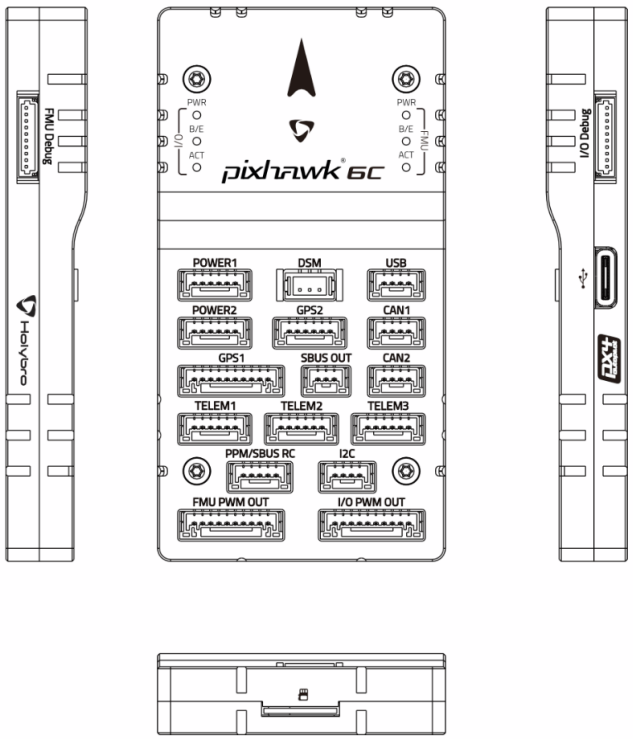


Fig. 5.35. Ports of the Pixhawk 6C flight controller.

5.7.2. MAVLink communication protocol

Communication between the Jetson Nano computer and the Pixhawk 6C flight controller is facilitated by the MAVLink protocol [9]. MAVLink (Micro Air Vehicle Link), developed in 2009 by Lorenz Meier, is designed for unmanned aerial systems. Its primary goal is to provide a lightweight, versatile, and reliable communication method between drone components such as autopilots, ground stations, and sensors. Due to its simplicity and flexibility, MAVLink has rapidly become the de facto standard for unmanned systems and is used in projects like PX4, ArduPilot, QGroundControl, and others.

MAVLink is a lightweight and reliable communication protocol specifically designed for real-time transmission of telemetry data and control commands. Its primary functions include:

- *Telemetry data transmission.* Exchanging information about the drone's state (coordinates, speed, altitude, battery level, etc.).
- *Drone control.* Receiving commands from a ground station or another device (e.g., navigation commands or mission execution).
- *Component integration.* Communication between various devices on board the drone, such as cameras, radio modules, or additional sensors.

The key advantages of MAVLink are:

- *Universality.* MAVLink is supported by many autopilots, ground stations, and external devices.
- *Minimized bandwidth.* The protocol is optimized for operation even in low-speed networks.
- *Real-time.* Ensures the transmission of critical data in real time.
- *Ease of implementation.* Open source and simple structure allow for easy integration of MAVLink into various systems.
- *Extensibility.* Ability to create custom message sets.

However, MAVLink has certain limitations:

- *Security.* MAVLink does not have built-in encryption, so external data protection methods (e.g., VPN) must be used.
- *Minimal integrity checking.* CRC protection exists but is basic and not a full-fledged solution for protecting against forged data.
- *Throughput.* MAVLink is optimized for small amounts of data and is not suitable for transmitting large files or high-quality video.

Currently, there are two versions of the protocol: 1 and 2. The first, unlike the second, has a limited set of commands and messages as it is the initial version. The second version, in turn, has the following advantages:

- *New features.* Support for packet routing, allowing multiple devices to be connected.
- *Backward compatibility.* Can work with MAVLink 1.0 messages.
- *Improved security.* Uses signed messages to verify their authenticity.
- *Larger command set.* Supports advanced features for modern systems.

Since the use of one or another version is limited by the flight controller, and Pixhawk 6C supports it, we have chosen the second version. This parameter is configured in the flight controller firmware in the port configuration section. It is also necessary to remember to use the correct import when writing code, as the libraries support both versions, and it is better to explicitly declare which version the program will use.

5.7.3. Implementation of the Flight controller interaction

In the world of drone technology, there are many libraries and wrappers for working with the MAVLink protocol in various programming languages: C++, C, Java, JavaScript, and others. However, for rapid development and convenient project debugging Python is an extremely popular choice. One of the most common libraries for working with MAVLink in Python is MAVSDK (formerly known as DroneCore Python). This is a high-level wrapper over C++ that allows developers to get convenient access to all the main features of the MAVLink protocol without delving into low-level details.

The following algorithms must be implemented:

1. Connecting to the drone.
2. Subscribing to receive the drone's status.
3. Waiting for the connection status to the drone.
4. Checking readiness to execute commands.
5. Arming the motors.
6. Commanding an automatic takeoff.
7. Waiting for 10 seconds (while the drone ascends to the desired altitude).
8. Commanding an automatic landing.
9. Unsubscribing from receiving the drone's status.

To perform the implementation of the algorithms an asynchronous execution model of the standard *asyncio* library should be used that provides with Python version 3.7 or higher.

The code below demonstrates the implementation of these algorithms.

```
import asyncio
from mavsdk import System
async def run():
    """
    This asynchronous function orchestrates the drone flight, including connection, arming, takeoff, hovering, and landing.
    """
    # Create a System object, the primary interface for interacting with MAVSDK.
```

```
    drone = System()

    # Connect to the drone via serial port. Other options include UDP, TCP, etc.
    await drone.connect(system_address="serial:///dev/ttyUSB1:57600")

   # Start a coroutine to print status messages asynchronously.
    status_text_task = asyncio.ensure_future(print_status_text(drone))

    # Wait for the drone to connect.
    print("Waiting for drone to connect...")
    async for state in drone.core.connection_state():
        if state.is_connected:
            print(f"-- Connected to drone!")
            break

    # Wait for the drone to have a global position estimate.
    print("Waiting for drone to have a global position estimate...")
    async for health in drone.telemetry.health():
        if health.is_global_position_ok and health.is_home_position_ok:
            print("-- Global position estimate OK")
            break

    # Arm the drone.
    print("-- Arming")
    await drone.action.arm()

    # Take off.
    print("-- Taking off")
    await drone.action.takeoff()

    # Hover for 10 seconds.
    await asyncio.sleep(10)

    # Land.
    print("-- Landing")
    await drone.action.land()

    # Cancel the status printing task.
    status_text_task.cancel()

async def print_status_text(drone):
    # Asynchronously subscribes to telemetry.status_text() and prints status messages.
    try:
        async for status_text in drone.telemetry.status_text():
            print(f"Status: {status_text.type}: {status_text.text}")
    except asyncio.CancelledError:
        return
if __name__ == "__main__":
    asyncio.run(run())
```

Below is detailed overview of MAVSDK commands used in the example:

1. ***drone.connect(system_address="...")***

Description: connects to a drone at the specified address.
Parameters (example):

- *serial:///dev/ttyUSB0:57600 (serial port)*
- *udp://:14540 (UDP, often used for the PX4 SITL simulator)*
- *tcp://192.168.1.10:5762 (TCP, e.g., Wi-Fi module on the aircraft)*

Behaviour:

- *Successful connection.* The drone is identified as the target system; drone.core.connection_state() will subsequently return is_connected=True.
- *Address error/device absence.* Program may throw an exception or wait indefinitely for a connection if the address is unreachable.
- *Connecting to multiple drones.* MAVSDK can work with multiple systems simultaneously, although usually only one drone is connected in System().

2. drone.core.connection_state()

Description: A method (from the Core API) that returns an asynchronous stream of information about the connection state.
Parameters: none.
Behaviour:

- If the drone is connected, *connected = "True"*.
- If the drone is disconnected, *connected= "False"*.
- Reconnection: The same stream can be reused to detect a new connection.

3. drone.telemetry.health()

Description: a method (from the Telemetry API) that returns an asynchronous stream with data about the drone's "health": GPS status, battery level, sensor readiness, etc.
Parameters: none.
Behavior:

- *health.is_global_position_ok.* It is "True" if the drone has a valid position (GPS is fixed).
- *health.is_home_position_ok.* It is "True" if the home position is defined.
- *Pre-flight checks.* Most autopilots require a valid GPS and Home, otherwise they will not allow *arm()*.

4. drone.action.arm()

Description: a method (from the Action API) that activates the drone's motors. Without this command, most autopilots will not allow takeoff.
Parameters: None (called without arguments).
Behavior:

- Successful arming. The drone transitions to the Armed state; motors start spinning at idle speed (for multicopters).
- Error or arming denial. It may occur if safety conditions are not met (e.g., weak GPS signal, low battery, uncalibrated sensors).
- Repeated *arm()* calls. Usually have no effect if the drone is already armed.

5. drone.action.takeoff()

Description: this method (from the Action API) initiates an automatic takeoff to a pre-programmed altitude. The takeoff altitude is configured in the autopilot parameters (e.g., MIS_TAKEOFF_ALT in PX4, TKOFF_ALT in ArduPilot).

Parameters: None. The altitude and other parameters are stored in the controller's firmware or set by separate methods.
Behaviour:

- *Arming before takeoff.* Typically, arm() must be called first; otherwise, takeoff() will result in an error.
- *Successful takeoff.* The drone will ascend to the configured altitude and hold its position (GPS Hold/Loiter).
- *Insufficient GPS or barometer erro.* takeoff() may be ignored or cause a failure.
- Hovering: After reaching the altitude, the drone will "hover" and wait for further commands (if the autopilot is configured for position hold mode).

6. asyncio.sleep(10)

Description: This function introduces a 10-second delay. In a real-world scenario, the drone would remain in the air during this time.
Parameters: 10 - the number of seconds.
Behaviour:

- *Normal delay*. The drone will continue to stabilize automatically in the air.
- *Changing external conditions.* Wind, GPS signal, etc., may cause the autopilot to adjust the drone's position.

7. drone.action.land()

Description: this method (from the Action API) initiates an automatic landing on the current terrain or at the Home point (depending on the autopilot configuration).
Parameters: none.
Behaviour:

- *Successful landing.* The drone will descend to the ground, perform a Touchdown, and the autopilot often executes disarm() automatically.
- *Interrupting landing.* If drone.action.takeoff() or another command is called during landing, the result depends on the firmware and mode - in most cases, the operation is interrupted, but a hard override may be required.
- *Landing without precise GPS.* The landing process may be rougher (the autopilot relies on the barometer), or the command may not be executed.

8. Possible return codes/exceptions

All calls to drone.action may raise exceptions such as:

- *ActionError.* A general execution error (the drone refused the command).
- *ConnectionError.* No connection to the drone.
- *TimeoutError.* If the drone did not respond to the command within a specified time.
- Some methods also have return codes like ActionResult (SUCCESS, COMMAND_DENIED, NO_SYSTEM) that can be handled to determine the result of execution.

As demonstrated in the example, working with the MAVLink protocol using the MAVSDK library offers several advantages:

- *Asynchronous Model.* The entire logic is built around events and deferred execution. Thanks to async/await, multiple data channels (telemetry, statuses, commands) can be processed concurrently without complex threading.
- *Connection Flexibility.* We've demonstrated a serial port, but other options (UDP, TCP) are available for various scenarios (real drone, simulator).
- *Simple Commands.* Commands like arm(), takeoff(), and land() are high-level, simplifying the learning curve. However, MAVSDK provides a much richer API for advanced actions.

- *Minimal Working Example.* This script is a basic demonstration to get started. In real-world applications, you can add mission processing, takeoff/landing behavior, speed control, flight path construction, and more.

With this simple example demonstrates the structure of an asynchronous Python application using MAVSDK and the basic steps to prepare a drone for flight which include:

- connecting to the system;
- checking readiness;
- activating motors (Arming)
- Takeoff and flight actions;
- Landing.

For more details on parameters, methods, and additional capabilities (e.g., starting/stopping missions, gimbal control, camera operation), refer to the official MAVSDK documentation [15].

5.8. Summary

The use of UAV swarms requires the construction of a drone control system based on the principles of multi-agent control, which is associated with the implementation of a whole range of tasks for processing data from sensors, inter-drone communication, planning actions to achieve the mission goal, and operational interaction with the flight controller.

These tasks require the creation of an architecture for the UAV control system using two microprocessor devices: the Pixhawk flight controller and the Jetson Nano microcomputer, which performs computing, communication, and control functions.

The control system involves the modular construction of software that implements audio and video data processing functions in Python based on modern methods and technologies, including the using of ANNs. To support inter-drone communication, a Mesh Network based on Digi XBee modules is organized, and interaction with the flight controller is carried out using the MAVLink protocol, also implemented in Python.

During the developing the main control module for a drone in a swarm, a model-oriented approach was used, which consists in directly embedding algorithm models, represented in the form of Control E-networks embedded into the control loop. This ensures prompt response to sensor data without using traditional cyclic polling.

Thus, the use of embedded models and ANNs directly in the control loop transfers the UAV control system into an intelligent one. The presented results of model and full-scale tests confirm the effectiveness of this approach.

References

1. Konert A., Balcerzakb T. TOWARDS AVIATION REVIVAL Military autonomous drones (UAVs) - from fantasy to reality. Legal and Ethical implications. //Transportation Research Procedia 59 (2021) 292–299 10th International Conference on Air Transport – INAIR 2021
2. European Global Navigation Satellite Systems (EGNSS) for drones operations. //White paper. European GNSS Agency (GSA). 2023 DOI:10.2878/52219.
3. Secom. Official Website [online]. Available at: http://www.secom.co.jp [Accessed 16 Sep. 2025]
4. Lim, Jia Tian. Development and control of mini-quadrotor. //DRNTU Engineering Electrical and electronic engineering Control and instrumentation Nanyang Technological University, 2013
5. ArduPilot. Official Website [online]. Available at: https://ardupilot.org/ardupilot/ [Accessed 16 Sep. 2025]
6. Pixhawk. Official Website [online]. Available at: https://pixhawk.org [Accessed 16 Sep. 2025]

7. K. Yang G Yang I Huang. Research of control system for plant protection UAV based on pixhawk. //Procedia Computer Science Volume 166, 2020, Pages 371-375
8. Kazymyr, V., Oleksiienko, P., Chornonoh, O., Rohovenko, A. (2024). Modeling of Defensive Drone Swarms with NetLogo. In: Kazymyr, V., et al. Mathematical Modeling and Simulation of Systems. MODS 2023. Lecture Notes in Networks and Systems, vol 1091. Springer, Cham. https://doi.org/10.1007/978-3-031-67348-1_28
9. MAVLink. MAVLink Documentation [online]. Available at: https://mavlink.io/en/ [Accessed 16 Sep. 2025].
10. NVIDIA. Jetson Nano Developer Kit User Guide [online]. Available at: https://developer.download.nvidia.com/embedded/L4T/r32-3-1_Release_v1.0/Jetson_Nano_Developer_Kit_User_Guide.pdf [Accessed 16 Sep. 2025]
11. Ayeni J. A. Convolutional Neural Network (CNN): The architecture and applications // Applied Journal of Physical Science, 2022. - Volume 4(4), pages 42-50.
12. Yamashita R., Nishio M., Kinh Gian Do R., Togashi K. Convolutional neural networks: an overview and application in radiology // Insights into Imaging, 2018, volume 9, pages 611–629.
13. Juan Terven, Diana-Margarita Córdova-Esparza 2 and Julio-Alejandro Romero-González. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Mach. Learn. Knowl. Extr. 2023, 5(4), 1680-1716.
14. Digi International. Zigbee Wireless Standard [online]. Available at: https://www.digi.com/solutions/by-technology/zigbee-wireless-standard [Accessed 10 Aug. 2025].
15. Digi International. Getting Started with XBee Zigbee [online]. Available at: https://www.digi.com/resources/documentation/Digidocs/90001942-13/#containers/cont_getting_started_with_xbee.htm?TocPath=Get%20started%20with%20XBee%20Zigbee%7C_____0 [Accessed 16 Sep. 2025]
16. MAVSDK. MAVSDK Documentation [online]. Available at: https://mavsdk.mavlink.io/main/en/index.html [Accessed 10 Aug. 2025].

Chapter 6. FPGA-based digital control systems

6.1. FPGA overview

6.1.1. FPGA and Microcontroller comparting

In modern engineering, Field Programmable Gate Arrays (FPGAs) have emerged as a powerful platform for implementing digital control systems due to their flexibility, parallelism, and real-time processing capabilities. Unlike traditional microcontrollers, FPGAs allow designers to create custom hardware architectures tailored to specific control tasks, enabling high-speed and deterministic performance.

FPGAs, unlike microcontrollers, are typically manufactured and sold in relatively small quantities, so they are more expensive per unit. Since microcontrollers are designed for specific purposes, they are very easy to configure. A basic microcontroller design can be developed in a few hours. In contrast, programming an FPGA takes much longer. Instead of microcontrollers where code is writing in C programming language, FPGA requires using of special language, such as Verilog or VHDL. With C, the code is written at a higher level than in Verilog or VHDL, where you have to specify individual gates and wires directly in the code.

So, a microcontroller is often simpler than an FPGA. Another factor to consider is the amount of power the device consumes. Since a microcontroller is designed for a specific use, it can be optimized to consume incredibly low power, such as a single battery. FPGAs, with all their routing capabilities, simply cannot compete with microcontrollers in terms of power consumption [1]. Fig. 6.1 demonstrates the main features of FPGA in comparison with microcontroller.

	FPGA	Microcontroller
Cost (low quantities)	Moderate	Cheap
Cost (high quantities)	Moderate	Cheap
Speed	Fast	Moderate
Power	Moderate	Low
Flexibility	High	Low
Ease of use	Medium	Easy

Fig. 6.1. Comparison FPGA with microcontroller (based on [1]).

These advantages make FPGD much less limited in their use for solving a wide range of digital control problems. Often a microcontroller will work very well, but sometimes it simply won't work due to speed or flexibility issues. In such situations, FPGAs is a better decision.

Despite microcontrollers almost always dominate in terms of cost, ease of use, and power consumption, there are other factors that determine the advantages of FPGAs [2]:

1. Parallel Processing: FPGAs can execute multiple control loops or signal processing tasks simultaneously, which is crucial for systems requiring high responsiveness.
2. Low Latency: Hardware-level implementation of control algorithms ensures minimal delay, making FPGAs ideal for time-critical applications.
3. Reconfigurability: Control logic can be updated or modified without changing the physical hardware, allowing for rapid prototyping and iterative development.

4. Integration: FPGAs can integrate multiple system components – such as sensors, actuators, and communication interfaces – on a single chip.

6.1.2. FPGA manufacturing trend

FPGAs are typically manufactured and sold in relatively small quantities before they are more expensive per unit then microcontroller.

Today, there are quite a large number of different manufacturers of FPGA chips. The most common and well-known among them are the following [3]:

1. Xilinx from AMD market leader with a share of about 50% by 2020.
2. Altera from Intel ccupied about 37% of the market.
3. Lattice Semiconductor – about 10%, specializes in low-power solutions.
4. Microchip Technology – a smaller share, but is actively developing in the areas of space, defense and automotive industries.
5. Achronix, QuickLogic, Efinix –low level players with a focus on AI, 5G, edge computing.

The market shares of FPGA chip manufacturers in 2024 is shown in Table 6.1 [4,5].

Table 6.1 FPGA Market Share by Manufacturers in 2024

Manufacturer	Market share (%)	Main products / applications
Xilinx (AMD)	~45–47%	Versal ACAP / telecommunications, defense, AI, video processing
Altera (Intel)	~38–40%	Agilex 7, Stratix, Arria / data centres, PCIe 5.0, CXL, telecommunacations
Lattice Semiconductor	~10%	MachXO3, Nexus / IoT, low-power, consumer electronics
Microchip Technology	~4–5%	PolarFire, Igloo / space, cars, industry
Achronix	~1–2%	Speedster7t / AI, machine learning, speech processing

There are some interesting facts [4]:

- Total FPGA market size in 2024 is probably $12.1 billion.
- Expected growth by 2029: up to $25.8 billion, with an average annual growth rate of 16.4%.

Key growth drivers: AI, IoT, 5G, autonomous systems, data centres, telecommunications.

Key FPGA market trends for 2025 are the next [5]:

- 1. Xilinx (AMD) maintains leadership with high-performance FPGAs for telecommunications, defense, and imaging.
- 2. Intel (Altera) is actively investing in new products, including Agilex 7 for data centers and PCIe 5.02.
- 3. Lattice focuses on energy-efficient solutions for IoT, automotive, and consumer electronics.
- 4. Microchip is developing radiation-hardened FPGAs for space applications.

6.1.3. FPGA, CPLD and ASIC devices comparing

FPGA using trend would not be complete without a comparison with other digital logic devices namely Complex Programmable Logic Device (CPLD) and Application-Specific Integrated Circuit (ASIC).

FPGA. Key features of FPGA devices are:

1. Programmability. Users can change the configuration of an FPGA to implement different digital circuits.

2. Parallel Processing. Unlike processors that execute instructions sequentially, FPGAs can perform many operations simultaneously.

3. Uses in tasks requiring high-speed processing: such as telecommunications, video processing, artificial intelligence, cryptography, and robotics.

An FPGA architecture is given in Fig. 6.2.

FPGA device consists of:

1. Logic Blocks (CLBs) – the basic elements that implement logic functions.

2. Programmable Interconnects – allow logic blocks to be connected into the desired circuit.

3. I/O Blocks – for interacting with other devices.

It has also inside these architectures, the introduction of some dedicated blocks such as RAM for storing data or configuration.

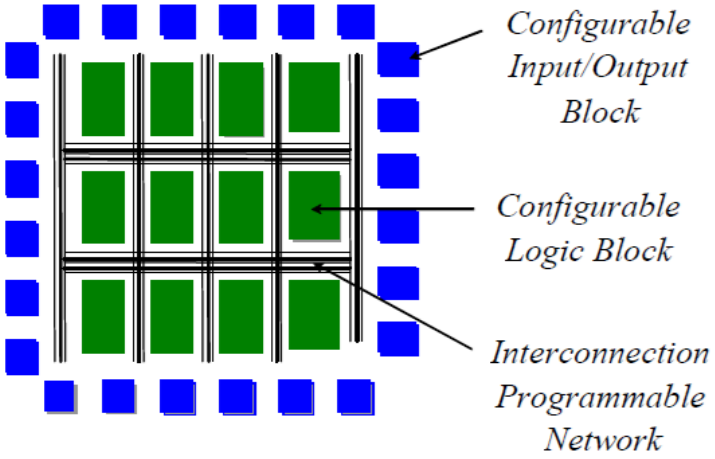


Fig. 6.2. Architecture of an FPGA [6].

CPLD. It is a type of programmable logic chip, similar to an FPGA, but with some important differences [7].

Key features of CPLD devices are:

1. Lower complexity. CPLDs typically have fewer logic elements than FPGAs and are suitable for simpler tasks.

2. Predictable delay. Signal delay in CPLDs is more stable, which is useful for certain types of digital circuits.

3. Small memory size. Usually used to implement simple logic functions like decoders, multiplexers, and controllers.

4. Configuration is retained. Unlike most FPGAs, CPLDs retain their configuration even after power is turned off (because it has non-volatile memory onboard).

Each CPLD device consists of:

1. Macrocells which are the basic “building blocks” of a CPLD. Each macrocell typically contains: a programmable logic array (PLA) or AND-OR array for implementing logic functions; a flip-flop or latch for storing state (used in sequential logic); optional output logic (e.g., tri-state buffers etc.).

2. Programmable interconnect array which connects macrocells to each other, allows flexible routing of signals between logic blocks.

3. Input/output blocks which are the interface between the CPLD and external devices and can be configured as input, output, or bidirectional.

Also, CPLD has control logic which manages global signals (like clocks, resets, and enables) and Non-Volatile Configuration Memory that stores the logic configuration permanently. This allows the CPLD to retain its functionality even after power is removed (see Fig. 6.3).

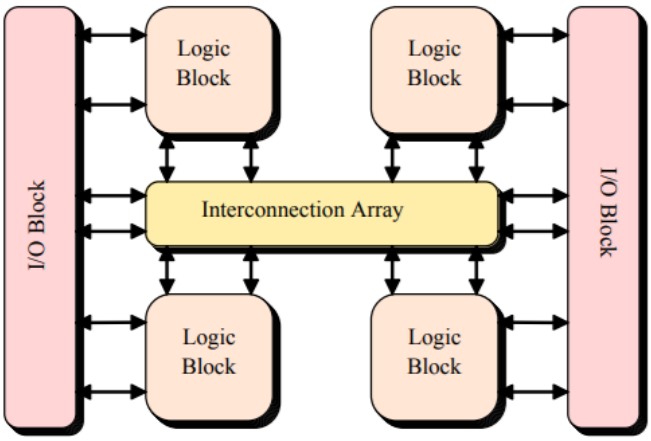


Fig. 6.3. Architecture of an CPLD [7].

ASIC. It is an integrated circuit designed to perform a specific task or function, unlike general-purpose chips like FPGAs. ASICs are designed and manufactured for a specific customer or application and can include digital, analog, and mixed-signal components on a single chip [8].

Key features of ASIC devices are:

1. Specialization. It designed to perform one specific task (e.g., video processing, cryptography, device control etc.).

2. High performance. Since the logic is optimized for a particular task, it operates faster and efficiently.

3. Low power consumption due to optimization, it consumes less power than general-purpose solutions.

4. High development cost, but low unit cost in mass production.

During the chip design the architects come up with a system level view of how the chip should operate, what components are required, how the data should flow inside the chip. An example of ASIC architecture design is shown in Fig. 6.4.

As rule, ASIC device consists of:

1. Logic gates and functional blocks which are the core digital circuits (AND, OR, NOT, etc.) arranged to perform the specific task the ASIC is designed for. In complex ASICs, these are grouped into functional modules (e.g., video decoder, encryption engine).

2. Standard cells and/or custom cells. Standard cells are pre-designed logic blocks used in semi-custom ASICs. Full-custom cells are designed from scratch for maximum performance and efficiency.

3. Memory blocks may include ROM (Read-Only Memory), RAM (Random Access Memory), Register files which used for storing data, instructions, or configuration.

4. Clock management circuitry which includes clock generators, dividers, and distribution networks to synchronize operations across the chip.

5. Input/output interfaces allow to communicate with other components or systems, can include GPIOs, serial interfaces (e.g., SPI, I^2C, UART), or high-speed interfaces (e.g., PCIe, USB).

6. Power management units which regulate and distribute power efficiently across the chip and may include voltage regulators, power gating, and clock gating.

7. Analog/mixed-signal blocks (optional) include analog components like ADCs (Analog-to-Digital Converters), DACs (digital-to-Analog Converters), PLLs (Phase-Locked Loops), etc.

8. Test and Debug Logic consists of built-in self-test (BIST), scan chains, and JTAG-interfaces for manufacturing testing and debugging.

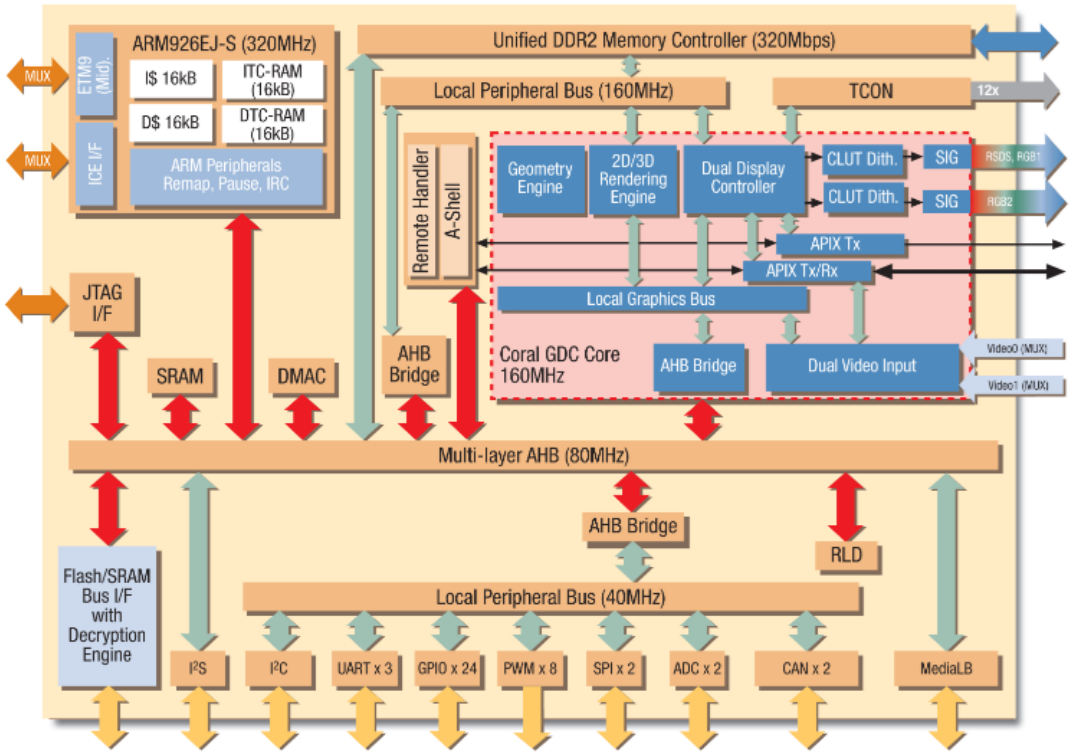


Fig. 6.4. An example of ASIC architecture design [8].

The comparative characteristics of the devices are given below, in Table 6.2. From this comparing, it follows that each of these considered devices has its own advantages and disadvantages as part of digital control systems which allowed to determine the main areas of their application [9].

1. FPGA application areas are:
   - artificial intelligence and machine learning (hardware acceleration for ANN,
   - signal and image processing (filtering, decoding, transformations;
   - telecommunications (routers, switches, base stations);
   - aerospace and defense (control systems, radar, avionics);
   - automation and robotics (motion control, sensor data processing);
   - digital circuit prototyping (rapid testing).
2. CPLD application areas are:
   - glue logic (interfacing between digital components);
   - I/O control (managing ports, buffers, and interfaces);
   - simple digital circuits (implementing decoders, counters, multiplexers);
   - discrete logic replacement (replacing 74xx series logic chips);
   - small embedded systems (where fast response and instant-on are needed).
3. ASIC application areas are:
   - smartphones and tablets (GPUs, image processors, communication modules);
   - cryptocurrency mining (specialized chips for hash computations (e.g., SHA-256);
   - automotive electronics (ABS systems, engine control, sensors);
   - medical devices (heart monitors, implants, diagnostic systems);
   - consumer Electronics (TVs, audio systems, smart home devices);
   - industrial systems (process control, safety, energy management).

Table 6.2 Detailed comparative of FPGA, CPLD and ASIC

Characteristic	FPGA	CPLD	ASIC
Logic Capacity	High (millions of gates)	Low to moderate (thousands of gates)	Very high (custom logic)
Flexibility	Very high (fully reprogrammable)	Moderate (limited reprogrammability)	None (fixed logic)
Performance	Moderate to high (parallel processing)	Stable (predictable delays)	Very high (task-optimized)
Power Consumption	Moderate to high	Low	Very low
Configuration Retention	Volatile (requires configuration on startup)	Non-volatile (retains configuration)	Non-volatile (fixed at manufacture)
Development Cost	Low to moderate	Low	Very high
Unit Cost (Mass Production)	High	Moderate	Very low
Development Time	Short	Short	Long
Reprogrammability	Yes	Yes	No
Typical Use Cases	Prototyping, AI, signal processing, custom hardware	Glue logic, I/O control, simple logic replacement	Smartphones, automotive, medical devices, mining
Integration Complexity	High	Low	Very high
Analog Support	Limited	Rare	Yes (if designed)
Startup Time	Requires configuration loading	Instant-on	Instant-on
Security	Moderate	Low	High
Design Customization	Moderate (via HDL)	Low	Very high (full custom design)

In general, ASICs beat FPGAs on power consumption and have a slight edge on speed, but they lose to FPGAs on flexibility and ease of use. As result, during developing control device the design is typically started from prototyping with an FPGA, and then it produces with ASIC [1].

6.2. Hardware description languages and tools

6.2.1. VHDL and Verilog HDL

Nowadays, there are 2 most popular hardware description languages (HDL) which are used for description of FPGA: Verilog HDL and Very High-Speed Integrated Circuits HDL (VHDL). Detailed comparative of the FPGA description languages is given in Table 6.3. [10].

From this comparative it follows:

1. VHDL is ideal for large-scale, safety-critical, and formally verified systems. Its strong typing and modularity make it robust but harder to learn.

2. Verilog is preferred for rapid development, prototyping, and commercial applications, especially in FPGA and ASIC design. It’s easier to learn but less strict.

The developer of digital control systems usually uses the tools of Computer-Aided Design (CAD) systems in the usual form (structural or schematic diagram, HDL description) to describe the required set of devices and obtain a file that is used to configure the FPGA. Programming consists in setting the necessary parameters for the FPGA functional converters and the connections between them. Such a design/manufacturing cycle takes little time, and changes can be made at any stage. The

introduction of new design tools at the initial stage requires practically no material costs due to the low cost of microcircuits and the availability of free, fully functional open-source versions of CAD.

Altera, a structural unit of Intel, has produced and produces FPGA and CPLD of several families, which are fundamentally different [11]:

- MAX3000A, MAX7000, MAX9000, MAX V, MAX 10;
- Stratix I, II, III, IV, V, 10;
- Cyclone I, II, III, IV, V, 10;
- Agilex 3, 5, 7, 9.

In particular, Altera (Intel) offers developers a system for automated design of digital devices based on programmable logic chips Quartus Prime. This is a CAD that supports work with all new FPGA families, provides access to all chip resources and allows you to design digital devices and systems of any level of complexity.

Table 6.3 VHDL and Verilog comparative

Aspect	VHDL	Verilog HDL
Origin	Developed by the U.S. Department of Defense in the 1980s	Developed by Gateway Design Automation in 1984
Standardization	IEEE 1076	IEEE 1364
Syntax Style	Ada/Pascal-like, verbose and strongly typed	C-like, concise and loosely typed
Typing System	Strongly typed – strict type checking	Weakly typed – more flexible but error-prone
Modularity	Supports packages, libraries, and user-defined types	Supports modules and parameters, limited package support
Design Units	Entity and Architecture	Module
Concurrency	Explicit with processes and signal assignments	Implicit – all blocks execute concurrently
Sequential Logic	Described using “process” blocks with sensitivity lists	Described using “always” blocks
Simulation Constructs	“assert”, “report”, “wait”, “after”	“$display”, “$monitor”, “initial”, “#delay”
Testbench Support	Strong, with structured verification	Strong, with procedural flexibility
Data Types	Rich: bit, boolean, integer, real, std_logic, array, record	Limited: wire, reg, integer, real, time
Abstraction Levels	Behavioural, RTL, Structural, Gate-level	Behavioural, RTL, Structural
Synthesis Compatibility	Requires strict RTL coding discipline	More forgiving, but synthesizable subset must be respected
Tool Support	Widely supported by major EDA tools	Equally supported by major EDA tools
Learning Curve	Steep – due to verbosity and strict typing	Gentle – easier for software engineers
Preferred Use Cases	Aerospace, defense, safety-critical systems	Commercial IC design, rapid prototyping, FPGA development

Code Reusability	High – due to modularity and strong typing	Moderate – less formal structure
Community & Documentation	Strong in academia and defense sectors	Strong in industry and commercial sectors

6.2.2. Intel Quartus platform

The Intel Quartus software is a full-fledged cross-platform design environment (there are versions for both MacOS and MS Windows, as well as for various versions of the Linux OS), which provides all stages of designing digital control systems on FPGAs using various digital device hardware description languages. The Quartus software includes tools for all phases of design using both FPGA and CPLD structures. Fig. 6.5 presents a generalized structure of the design process in the Intel Quartus Prime environment.

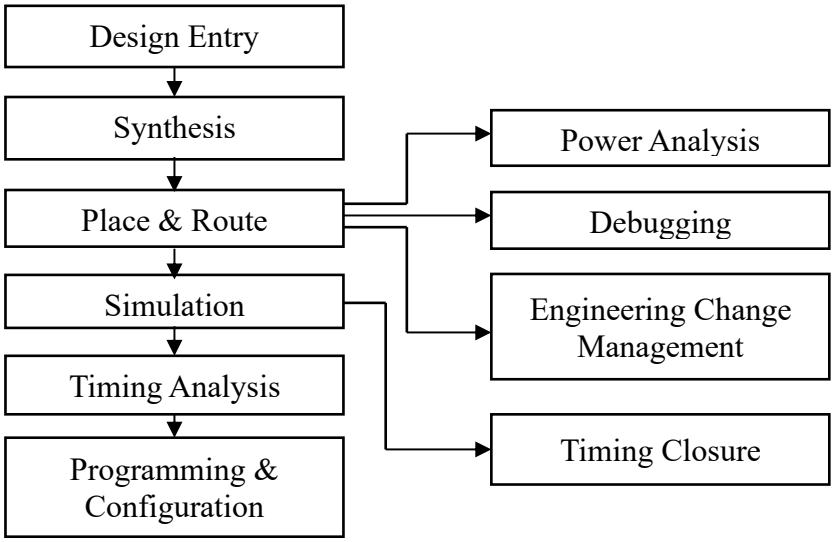


Fig. 6.5. Generalized structure of the design process in the Intel Quartus CAD

Intel Quartus Prime is specially adapted to work with FPGAs (so-called EDA Tool - Electronic Design Automation Tool), that is, it supports the entire design cycle of digital systems based on programmable logic, which provides the ability to reconfigure the internal structure (for example, for all FPGA chips of all Cyclone and Stratix families). The Quartus Prime CAD includes:

- various tools for developing digital devices (graphic editor, circuit editor, VHDL and Verilog HDL editor, etc.);
- a compiler and editor for placing the developed system/device on the logic of the target FPGA;
- tools for analysing timing characteristics (signal propagation delay time between input and output, maximum clock frequency, signal setup time and hold time, etc.);
- a timing diagram editor for testing and debugging digital devices/systems;
- a programmer for transferring the configuration from the project to the FPGA.

Analogues of the Intel Quartus Prime CAD are Quarus II, MAX+PLUS II (the predecessor of Quartus II), and competitors are Vivado and ISE from Xilinx (AMD) and Active HDL from Aldec. Additionally, Quartus Prime provides the ability to use a graphical user interface, EDA or command line interface in each phase of development. You can use any interface for the entire design process or different settings for each individual phase of development.

6.3. Program developing process

6.3.1. Project concept in Intel Quartus Prime CAD

The term “project” in the Intel Quartus Prime CAD refers to a set of files related to the designed digital system, in which two groups of files are distinguished [11]:

1. Logical files that describe the structure or algorithm of work (Design Files).
2. Auxiliary files (Ancilary Files).

A project can contain either one logical file or a set of them that form a hierarchical description of the module/device/system being designed. In the hierarchical description, among the set of logical files, the following are distinguished:

1. The top-level file in the description hierarchy (Top-level Design File).
2. Files of lower (one or more) levels (Low-level Design files).

The top-level file of the hierarchy specifies the architecture of the module, determines the set of modules that are part of it as components, and their relationships. Descriptions of these modules are contained in the logical files of the lower level of the hierarchy. In turn, they may include modules in the form of components, the descriptions of which are given in logical files of an even lower level of the hierarchy, etc.

The project name must match the name of the upper-level module in the hierarchy of descriptions, and therefore the name of the logical file in which this description is contained. The names of the modules of the lower levels of the hierarchy, in turn, must match the names of the files in which they are implemented.

Logical files may include files of one of the following types:

1. Block Diagram / Schematic File (*.bdf), contains a structural diagram of the device/system created in the Quartus Prime CAD system.
2. Graphic Design File (*.gdf), contains a schematic electrical diagram created in the CAD.
3. AHDL Text Design File (*.tdf), contains a text description of the module in the AlteraHDL language.
4. Waveform Design File (*.wdf), contains timing diagrams of input and output signals created in Quartus Prime.
5. VHDL Design File (*.vhd), contains a text description of the module in VHDL.
6. Verilog Design File (*.v), contains a text description of the module in Verilog HDL.
7. Orcad Schematic Files (*.sch), contains a diagram created in ORCAD CAD.
8. EDIF Input Files (*.edf), contains a description of EDIF 200 or 300.
9. Xilinx Netlist Format File (*.xnf), contains a description of the module obtained in Xilinx CAD.

Additional files store additional information about the project. Their names coincide with the project name, and the extensions may be different.

6.3.2. Structural strategies

The Intel Quartus Prime design environment allows you to implement both bottom-up and top-down structural design strategies.

Both the first and second strategies involve the use of behavioural and structural descriptions of modules/devices/systems. In the structural description, the module/device/system is represented as a set of interdependent components of a lower level of the description hierarchy. In the behavioural description, the algorithm of their operation is specified.

Bottom-up design is used when a detailed structural description already exists for the control system being developed (usually, this is an electrical schematic diagram, which is implemented on paper and consists of medium-level integration microcircuits), executed in an element base different from the base available to the FPGA developer.

In this case, the developer solves the following tasks:

- creation of functional analogues of the elements used in the existing structural description;
- configuration of the created components;
- integration of the created components into a single module;
- modeling and debugging the operation of the device as a whole.

During the design process, the developer first creates lower-level modules in the description hierarchy, and then the upper-level module. Hence the name of the design strategy.

The top-down design strategy is used when the operating algorithm (in the form of a behavioural description) of the device being developed and a set of system requirements (maximum clock frequency, delay for signal propagation from inputs to outputs, energy consumption, cost, etc.) are specified. In this case, the behavioural description can be both formalized (i.e., presented in the form of an algorithm diagram, graph, transition table, output table, etc.) and informal (text description). The implementation of top-down design is based on the iterative execution of structural decomposition.

In a simplified form (based on the functionality of the Quartus Prime CAD), the top-down design procedure provides:

- development of the system architecture: the behavioural description is converted into a structural one (a structural diagram), the elements of which are architectural modules;
- architectural modules are either described at the behavioural level (for example, using the AHDL, VHDL, Verilog HDL), or their structural decomposition is carried out and a structural description is created, the elements of which are functional modules;
- after this, functional modeling of the operation of modules with behavioural descriptions is carried out;
- functional modeling of the operation of modules with a structural description (since modules with a behavioural description are included in them as components);
- modeling and debugging of the device as a whole.

Thus, in the design process, the developer descends from the top level of the description hierarchy, that is, the FPGA level, to the lower levels. Hence the name of the design strategy.

It is worth noting that the top-down design strategy has undeniable advantages both in terms of development time and the quality of project development.

Regardless of the choice of design strategy, it is advisable to use a text description created in high-level hardware description languages (VHDL, Verilog) to implement the structures and algorithms of the modules/devices.

6.3.3. Intel Quartus Prime CAD user interface

After run Intel Quartus Prime, the main CAD window will appear on the monitor, the appearance of which is shown in Fig. 6.6, it consists of the following areas [12]:

- window title containing the project name and its working directory (indicated by the number I);
- the main CAD menu (indicated by the number II);
- the CAD toolbar (indicated by the number III);
- the project navigator, which displays the project hierarchy, project files and quick start commands (indicated by the number IV);
- the project compilation procedure status window (indicated by the number V);
- the message window (or console), which displays information about the execution of operations, errors and warnings (indicated by the number VI);
- the main window, which displays reports, project files and other project elements (indicated by the number VII).

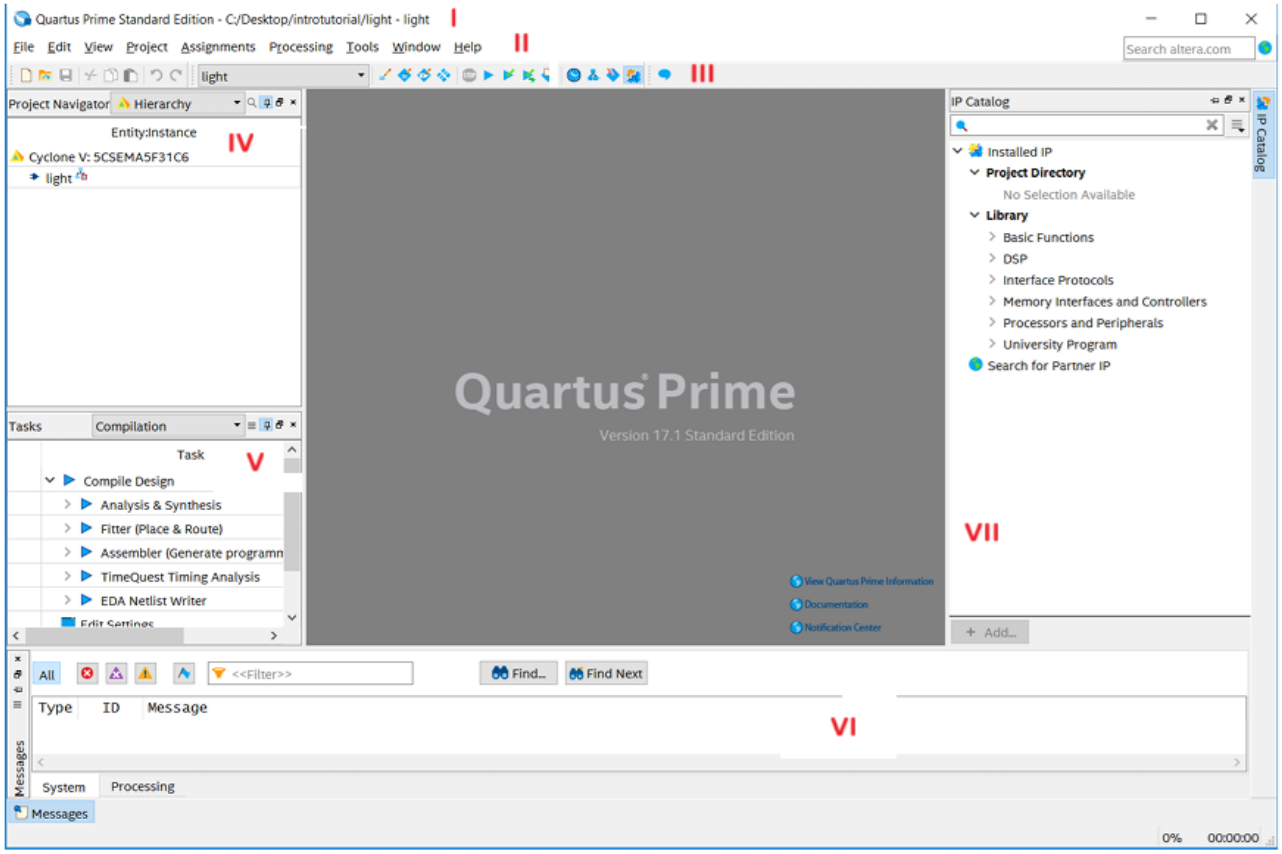


Fig. 6.6. Quartus Prime main window [12]

It is worth noting that the window shown in Fig. 6.6 will appear each time you run the Quartus Prime CAD. Here areas IV (Project Navigator) and VI (Messages) are empty, since no project is currently open for work, in addition, area V (Tasks) is unavailable (highlighted in gray) because there are no tasks to compile. The Recent Projects list in area VII of the window may contain links to projects that were previously opened (no more than four links). Clicking on the appropriate link will allow you to open the required project more quickly.

Fig. 6.7 shows an example of the main window of the Quartus Prime CAD system while working on an open project.

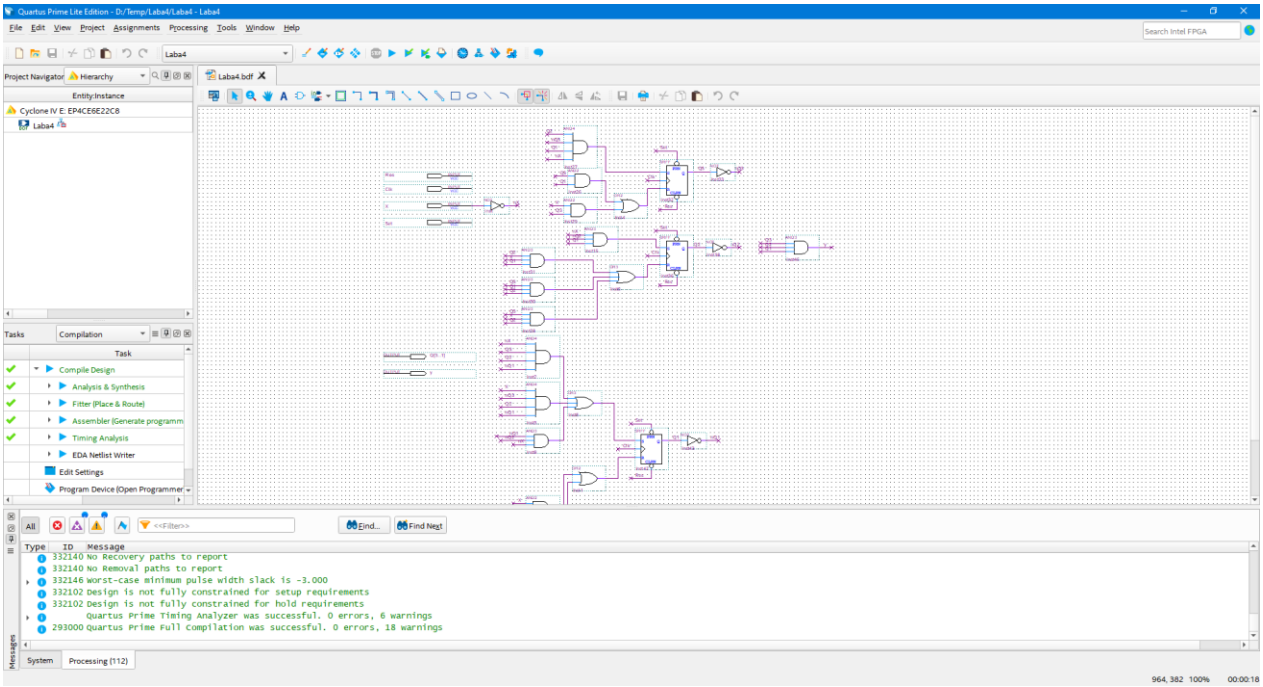


Fig. 6.7. Quartus Prime main window with open project

6.3.4. Logical analysing of signals

To debug the operation of digital control systems, it is often necessary to be able to simultaneously view a large number of signals for their changes in real time. Usually, to solve this problem in digital systems, logic analyzer s are used. For microcontroller systems, such analyzer s are implemented as external devices. Similar capabilities are provided by CAD programs designed for design on the basis of FPGAs.

The built-in logic analyzer Signal Tap Logic Analyzer, like a conventional logic, includes a memory for storing signal samples and a control system. The Signal Tap Logic Analyzer uses the built-in FPGA memory for storing signal samples and a control circuit implemented inside the FPGA, which determines the algorithm for recording signal samples. A personal computer is used to display information, which is connected to the debug board using a boot cable. It is controlled using the Intel Quartus Prime.

To use Signal Tap, you must perform the following sequence of actions:

1. Create a new or open an existing project.
2. Add the Signal Tap Logic Analyzer module to the project and configure it.
3. Define the conditions for starting data capture (Data Triggering).
4. Compile the project.
5. Download the project to the FPGA chip.
6. Run Signal Tap Logic Analyzer in the Quartus Prime.
7. Record data to the buffer memory (Data Capture).
8. Analyse the recorded data (Data Analysis).

In order to add the Signal Tap Logic Analyzer module to the project, you must create a new logic analyzer file, which will automatically be added to the open project.

To create a new logic file, you can use one of two options:

1) select the Tools → main menu item (Fig. 6.8);

2) the File → New → Signal Tap II Logic Analyzer File menu item (Fig. 6.9).

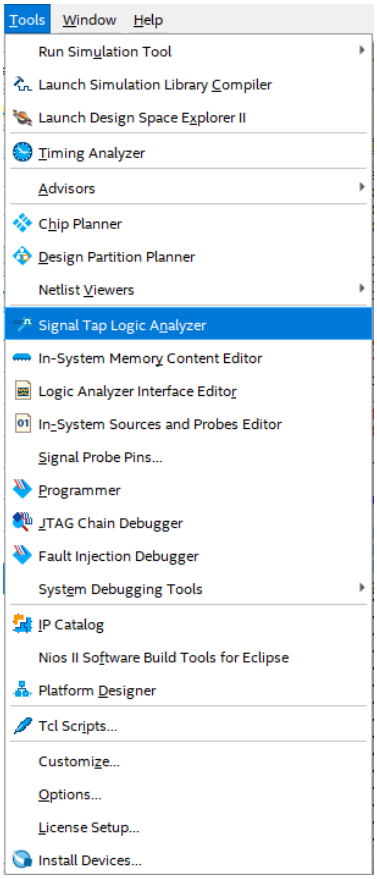


Fig. 6.8. Creating a new Signal Tap Logic Analyzer file across Tools menu

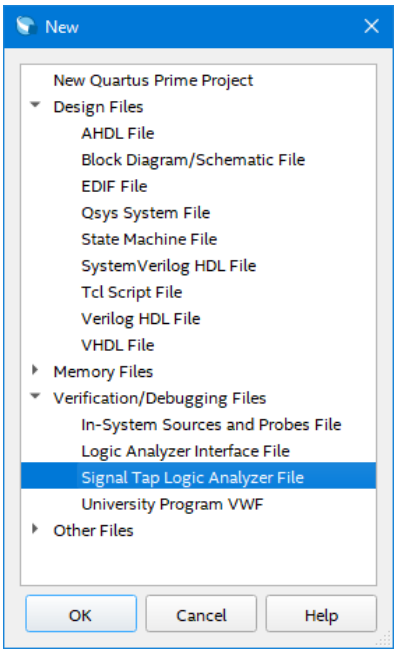


Fig. 6.9. Creating a new Signal Tap Logic Analyzer file across File menu

In any case, this will open the logic analyzer window (Fig. 6.10).

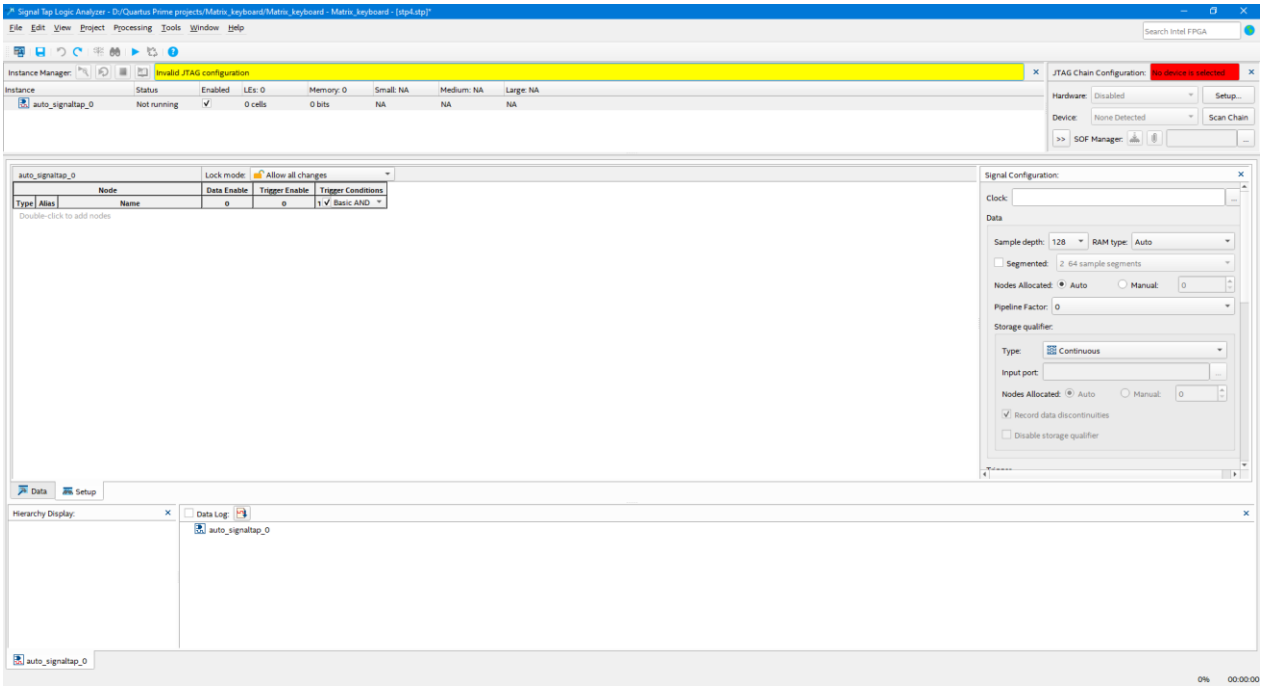


Fig. 6.10. Signal Tap logic analyzer window.

Let's take a closer look at two dialogs - the signal list with the Data and Setup tabs. The list of signals from the Setup tab is shown in Fig. 6.11.

Node			Data Enable	Trigger Enable	Trigger Conditions
Type	Alias	Name	126	126	1 Basic AND
		Add0~0			
		Add0~2			
		Add0~4			
		Add0~6			
		Add0~8			1
		Add0~10			1
		Add0~12			0
		Add0~14			0
		Add0~16			
		Add0~18			
		Add0~20			0
		Add0~22			1
		Add0~24			1
		Add0~26			0
		Add0~28			
		Add0~30			
		Add0~32			
		Add0~34			
		Add0~36			
		Add0~38			

Fig. 6.11. List of signals in the Signal Tap II logic analyzer.

To add signals, double-click in the Node field. The standard Node Finder dialog will open, in which Signal Tap II: post-fitting or Signal Tap II: pre-synthesis signal filtering is performed. In the first case, signals that are present in the project after the project synthesis and placement on the chip will be shown. Such signals will be shown in blue. In the second case, signals that are obtained after the first stage of compilation - syntax checking and building the project hierarchy - will be shown. Such signals will be shown in black. After full compilation and optimization of the project, a situation is possible when some of the signals disappear. Such signals will be shown in the signal list in red.

After a signal is added, it is possible to configure how the signal will be analysed. For this, the Data Enable and Trigger Enable buttons are used. The Data Enable button allows or prohibits recording data from the selected signal. This makes it possible to increase or decrease the amount of memory used by the logic analyzer. If the Trigger Enable button is disabled, this means that the selected signal cannot be used in creating recording conditions. In turn, this reduces the analyzer's logic circuit. A signal with the Data Enable button disabled but Trigger Enable enabled is not recorded in memory, but is included in the recording conditions of other signals.

Here it is necessary to clarify the meaning of the concept of trigger. During its operation, data passes through the logic analyzer, but the recording is not stopped. As soon as the recording condition is met, the signal recording stops and the recorded signals are displayed on the screen. The recording condition is called the trigger.

Next, the window contains a list of signal recording conditions. Up to ten different recording conditions can be selected for each signal. The number of recording conditions is determined in the trigger parameters dialog and is described below. You can choose one of two recording conditions - general (Basic) or advanced (Advanced).

The general method of recording involves performing a logical AND function on all signals or groups that are in a given column. To change the signal value in the context menu, you can select the following values:

1. "Don't Care" – the signal level does not matter.
2. "Low" – low signal level.
3. "Falling Edge" – trailing edge of the signal.
4. "Rising Edge" – leading edge of the signal.
5. "High" – high signal level.
6. "Either Edge" – any change in the signal.
7. "Insert Value" – entering the value of the signal group.

The selected signals will be recorded every time all the conditions written in the column are met during the edge of the clock signal.

The Advanced option allows you to create more complex logic equations to start recording signals. When you select this option, a graphical editor opens, in which a logic equation is built using simple blocks.

The Signal Configuration window, located to the right of the pin list, allows you to define the clock signal, the size of the data buffer and its properties, and the recording conditions.

The first is the assignment of the clock signal (Clock). The logic analyzer will read signals on each edge of the clock signal. It is advisable to choose a signal that performs the function of a clock in the clock domain to which the signal under study belongs. The higher the frequency of the clock signal, the greater the accuracy of the logic analyzer. However, this will lead to an increase in the amount of memory required to store signal samples.

The memory volume (Sample depth) and its type (RAM type) allow you to determine the size of the buffer for storing signal samples and the type of built-in memory elements. The buffer size can vary from 0 to 128 K. But it is not advisable to set the maximum value, as this may lead to the fact that the project with a logic analyzer will not fit in the FPGA. The memory type depends on the family of microcircuits used for implementation. For example, for Stratix IV family microcircuits these are memory blocks of types MLAB, M9K, M144K, and for Cyclone IV family – M9K. If the memory type is not important, then you can leave the value Auto.

The next parameter that needs to be defined is the buffer type. By default, the cyclic (Circular) buffer is set. The developer can also select a segmented buffer (Segmented) and define the size of the segments.

As a result of performing all these operations in the digital control system, it will be possible to view the status and change of the necessary signals, which will simplify and speed up the process of its debugging and configuration.

In the next paragraph, an example of designing a digital control system based on FPGA will be considered in more details (see https://habr.com/ru/articles/801191/).

6.4. FPGA based GMPPT algorithm for DC-DC converter

6.4.1. Global MPPT algorithm

Renewable energy sources (PV panels) require detailed analysis of the power dependence. Despite the Maximum Power Point Tracking (MPPT) algorithm the global tracking of the maximum power is an important issue. However, an unequivocal decision is not available in the research library.

Furthermore, the algorithm of global MPPT depends on the selected topology. Usually, the PV application accompanied by the AC or DC grid connection, as it is shown in [13 – 15]. The method of MPPT based on the stability analysis with using Lyapunov quadratic function is discussed in [16]. The Lyapunov function is generated from the variation of energy stored in the DC link capacitor.

The novel MPPT and Global MPPT (GMPPT) scenario for the single-phase qZS-based on the grid-connected inverter is descripted in [17]. The proposed control approach performs a stable operation of DC-link voltage, shoot through duty cycle decoupling and high MPPT efficiency.

The novel family of buck-boost DC-AC converters were proposed in [18]. The connection to grid of the buck-boost inverter with unfolding circuit was considered in [19]. The inverter based on unfolding circuit can operate in the wide input voltage and input power ranges. This solution is suitable for applications with a high-power density requirement. The interleaved features of the buck-boost inverter based on unfolding circuit is discussed in [20]. The input capacitor *CIN* filters the low current ripple and supplies the DC current at the input side.

A new technique of GMPPT for interleaved buck-boost DC-DC converter is shown in Fig. 6.12. The main idea is to decrease the time of the GMPPT.

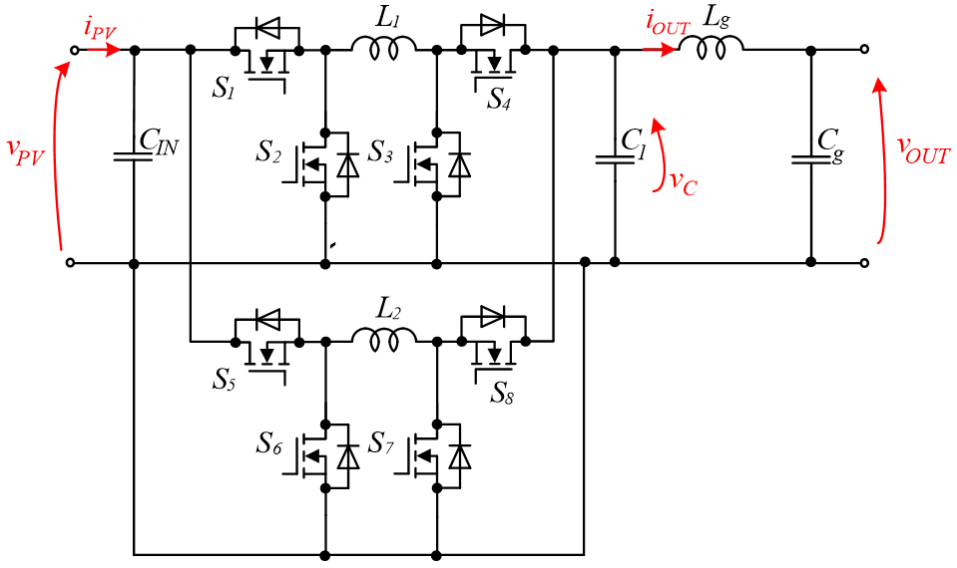


Fig. 6.12. The interleaved buck-boost dc-dc converter with synchronous switches

The interleaved buck-boost converter can operate in a wide range of the input voltage. The minor advantage is a contribution the input current between buck-boost cells because each inductance is limited by current and a reducing of the current allows to fit in limits. The input power is bounded by the input source parameters. This research devotes to the renewable energy source, particularly to PV array as the input side. For instance, the solar panel HNS-SD140 of Hanergy company provides 140 W in a case of the normal environment conditions. The main parameters of consider solar panel are collected in Table 6.4.

Table 6.4 Electrical performance of solar panel HNS-SD140.

Parameter	Value
Nominal Power, W	140
Maximum Power Voltage, V	59
Maximum Power Current, A	2.17
Output Circuit Voltage, V	75
Short Circuit Current, A	2.58

The serial set of the PV string consists of 3 panels. Thus, the open circuit of the set can reach up to 225 V, while the maximum input voltage in normal operation is 177 V. The maximum given PV current depends on the weather state. The maximum current according to panel document is possible to get only in a clear weather, without any clouds. However, the probability of weather state depends on the location, where the panels are placed. The clouds and a high temperature led to a great voltage drop of the solar panel. Besides, the abnormal weather condition caused to reduce the maximum PV current. The theoretical characteristic of the serial set of 3 panels HNS-SD140 is shown in Fig. 6.13.

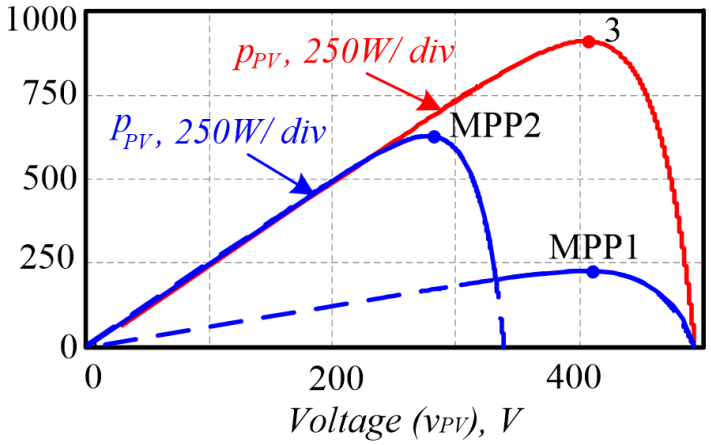


Fig. 6.13. The theoretical characteristic of the serial set of 3 panels HNS-SD140.

The Fig. 6.13 (red curve) shows the power line of the selected PV installation with one local maximum (point 3). The simple MPPT provides success out coming on the necessary power point. However, several local maximums are appearing during the bad weather (partial shadowing of several panels) is shown as blue line. Fortunately, if MPPT goes to the suitable power point, but a likelihood is 50%. Therefore, the global MPPT is applied by researchers to find the necessary power point.

The traditional GMPPT gradually exposes changes power from zero to maximum and at the end to zero again. Thereby, the control system is founding the maximum power point during the power estimation of the PV string. The system goes to the find point after GMPPT. As a rule, the time of the GMMPT takes up to 10 seconds. From one side, if MPPT cannot detect the global maximum, the GPPT brings a favour. But on the other hand, the MPPT can fluctuate around the global maximum, thus the GMPPT is not necessary. The using GMPPT in the last case carries energy loss that can be given to the load.

The main idea of the proposed GMPPT is to set out the short circuit occurrence of the PV panel and measure only SC current. The maximum power point as a rule regards to 80-85% of the short circuit current. So, the idea is to get a short circuit current of the panel and after set the current of the 80% from the maximum possible value. Fig. 6.14 demonstrates time conversion of the GMPPT with comparison between traditional and proposed method.

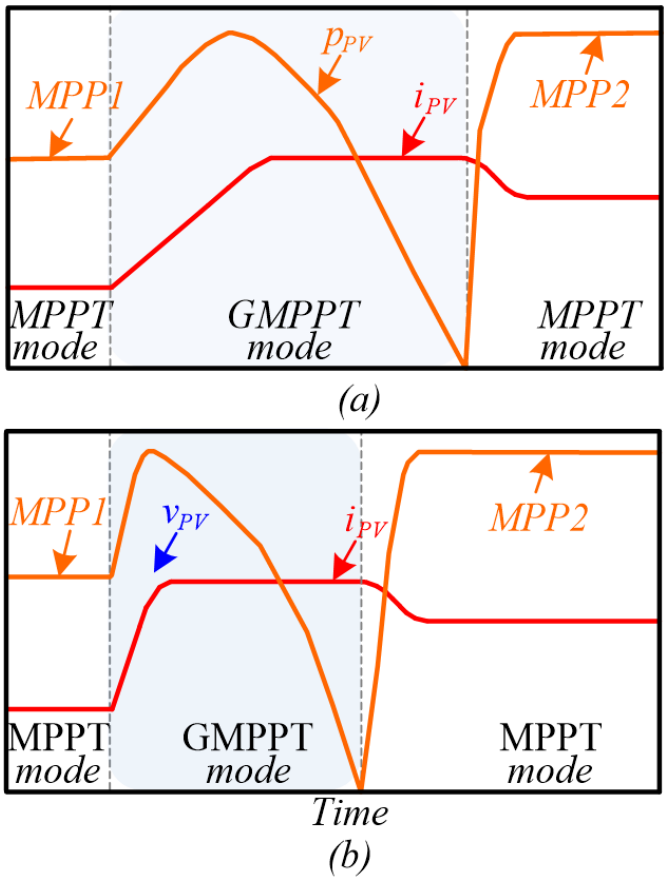


Fig. 6.14. MPPT operation within the GMPPT: (a) - traditional method, (b) - proposed method.

The issue of the proposed GMPPT is to manage discharging of the input capacitor, because it is accompanied by a huge discharging current. The filled areas correspond to power losses. Finally, the major hypothesis is to reduce the overall power loss by cutting the GMPPT time.

The proposed GMPPT can be implemented by achieving a short circuit condition of the PV string: only by transistors, trough the input inductor with or without active decoupling or set the higher power at the input side. A high current spike occurs in cases of short-circuit without decoupling, that can burn the transistors or even the inductance.

The active decoupling is a good way to manage energy between triangle of PV string, battery (or small capacitor) and converter. However, it requires additional switches with driver circuits. SC current can be obtained only through the input transistor, but it caused to a high current spike of the input capacitor. The last way is to change the input power immediate by increasing the output voltage in a case of the resistance load. Thus, a fast-changing input power is considered in selected control systems.

Fig. 6.15 shows the block diagram of algorithm that is devoted to GMPPT. The load is a simple resistor; thus, it is possible to change the input power only by the level of the output voltage. The main regulator is a PI regulator for supplying the output voltage. Despite that PI controllers are used for GMPPT, the MPPT manages the output voltage with respect to the necessary current at the input side based on a PI regulator as well.

The level of the output voltage can be set due to the value of the load value:

$$iin = \frac{|v_{OUT}^2|}{R_{LOAD} \cdot v}. \tag{6.1}$$

where R_{LOAD} is the load resistance.

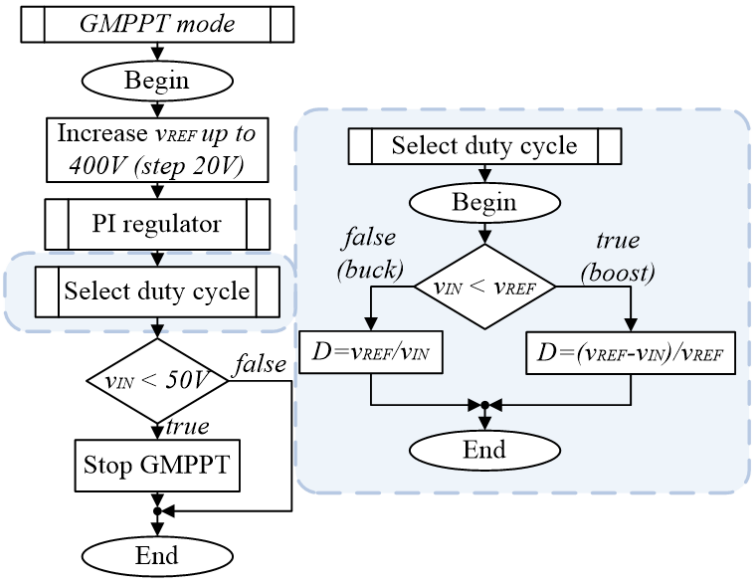


Fig. 6.15. The block diagram of the GMPPT algorithm.

Thus, only voltage measurements are taking part for the reference signal generation. The modulation signal is also generated by using duty cycle dependences for buck and boost cases:

$$D_{BUCK} = \frac{v_{OUT}}{v},\ D_{BOOST} = \frac{v_{OUT}-v}{v}. \qquad (6.2)$$

The signals for the second buck-boost cell are shifted on 180 degrees. The control system includes itself Phase-Lock-Loop (PLL) as a filter. PLL calculates the magnitude and phase of the output voltage.

The strategy of the GMPPT begins with the sharp increasing of the output reference. The increasing of the output voltage leads to raising the input power and the PV current. The system can obtain the SC current as soon as the PV voltage drops significant. All switches are off during some delays after a suitable current detection because it needs to return PV string in the normal voltage. Finally, the system sets out current at the input side with according the founded GMMPT and starts the MPPT for measured current.

6.4.2. GMPPT algorithm simulation

The selected system was modelled in PowerSim 11. Traditionally, the MPPT or GMPPT asks a huge time interval (see Fig. 6.16). Therefore, a big number of points occupy the physical memory of the personal computer. The additional small resistance of each passive element was included during the simulation.

The comparison between the traditional and proposed GMPPT was done for verification the theoretical hypothesis. Fig. 6.17 shows the simulation results for GMPPT. The traditional method took more time of conversion for raising PV current to short-circuit mode (fig. 6.17 (b)) in comparison with proposed technique (fig. 6.17(e)). Figures 6.17(c-f) demonstrate a changing the PV power consuming from a smaller local maximum to a greater local extremum. The open circuit voltage is the same as in the case study system. The smaller local maximum corresponds to 1.6 A of the input current. The greater local extremum relates to 2.17 A that was set after GMPPT interval. The time conversion of the GMPPT of the conventional way reaches to 30 milliseconds. While the time conversion of the proposed GMMPT shorted to 10 milliseconds.

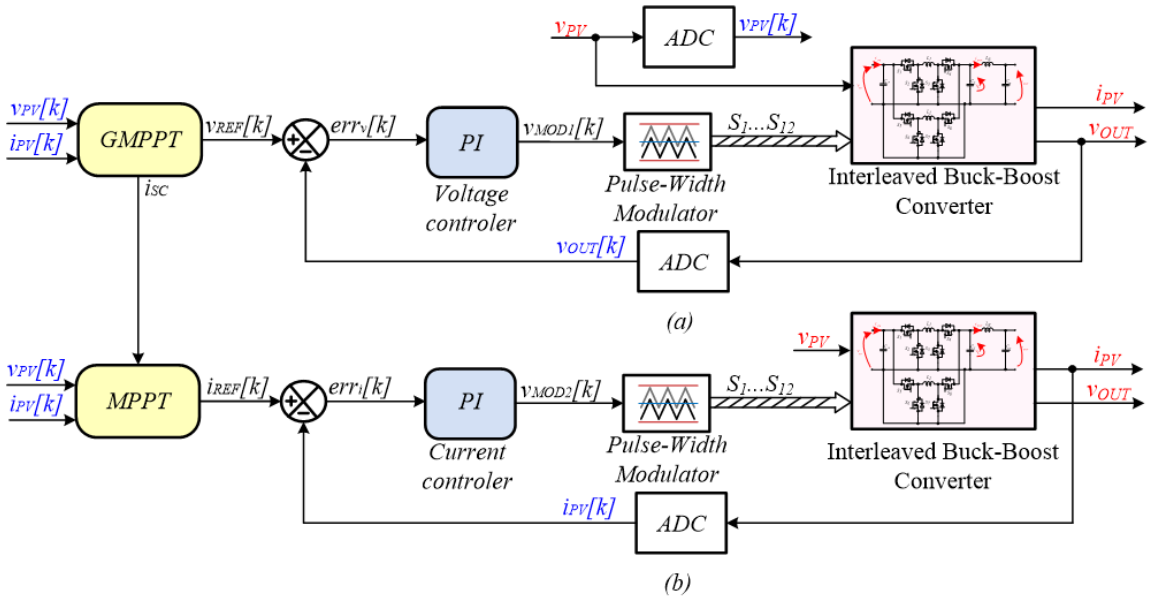


Fig. 6.16. Structure of the close-loop system based on the PI controller in a case of: GMPPT (a), MPPT (b)

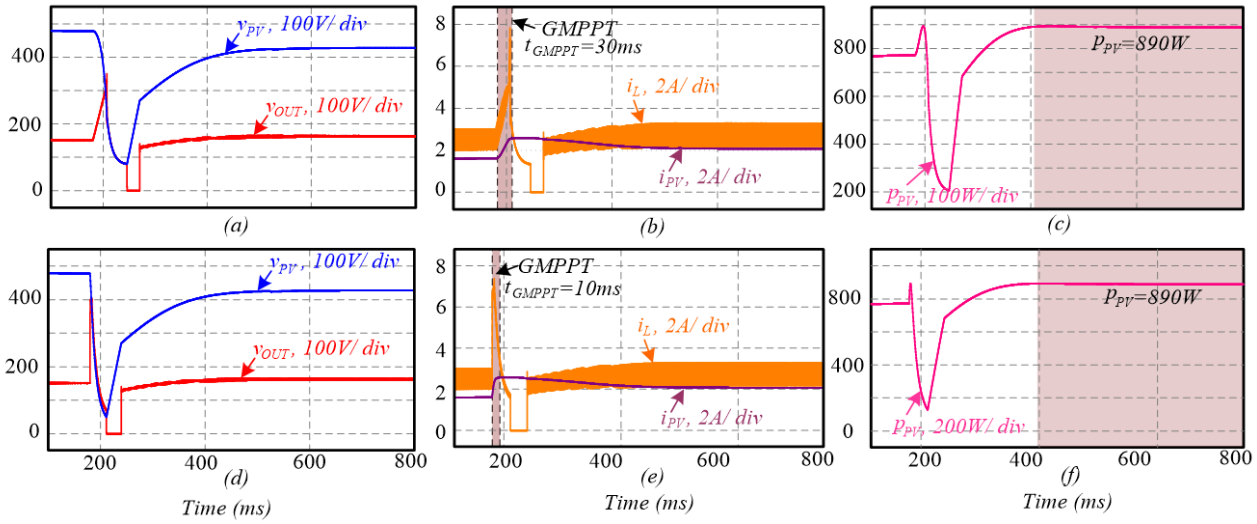


Fig. 6.17. Simulation results of buck-boost converter for the traditional GMPPT: the PV and output voltages (a), the PV and inductor currents (b) and PV power (c); and for proposed GMPPT: the PV and output voltages (d), the PV and inductor currents (e) and PV power (f)

The parameters for the simulation are listed in Table 6.5. Note that the sample frequency is so high and equals to 62.5 kHz.

The system sets the output voltage level as soon as possible up to 400 V in proposed global MPPT, while the traditional technique is accompanied by slowly output voltage accretion. It needs to notice, that the inductor current spikes were occurred during GMPPT. However, those current spikes didn't exceed than 8 A.

Table 6.5 Parameters of the system during the simulation and experiment

Parameter	Value
Output Capacitor *C1*	1.0 µF
Input Capacitor *CIN*	300.0 µF
Grid Capacitor *Cg*	100 nF
Input Inductor *L1*	600.0 µH
Grid Inductor *Lg*	680.0 µH
Input Resistance *RIN*	0.1 Ω
Grid Resistance *Rg*	0.1 Ω
Input Inductor Resistance *RL*	1.0 Ω
Ouput Capacitor Resistance *RC*	0.1 Ω
Ouput Inductor Resistance *RLg*	5.0 Ω
Resistance of *S1 – S8*	0.5 Ω
Resistance of *S9- S12*	0.3 Ω
Sample Frequency *fsample*	62.5 kHz
PWM Frequency *fPWM*	62.5 kHz
Load Resistance for buck case *RLOAD*	30 Ω

6.4.3. GMPPT algorithm experimental testing

The experimental testing requires a designing the necessary boards at start. The proposed algorithm was chosen the same as in simulation part. The verification by the experiment helps to prove the hypothesis that was considered in 1case study system. Therefore, the real solar string was used in the experiments. The PV string with the 7 serial panel is located on the roof of the Chernihiv Polytechnic National University as it is shown in Fig. 6.18. The wide rag covers two solar panels. Thus, the imitation of the shadowing condition is provided by the panels covering. The shadow panels were in the middle of the string.

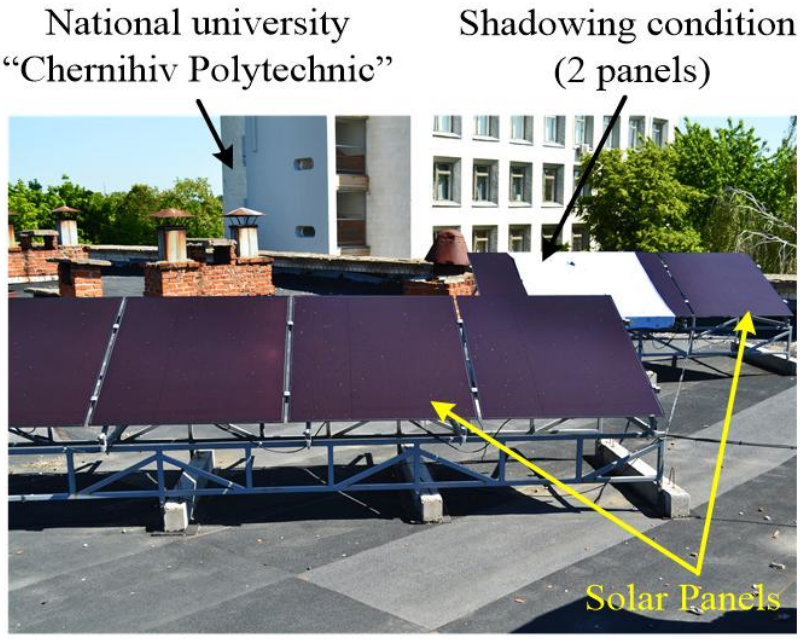


Fig. 6.18. The PV string station based on 7 serial panels.

The electronic load Maynuo M9712B helps to get the real power characteristics of the PV string. The experiment confirms the theoretical statements about the power change in case of normal and with shadow condition. The difference of the real and theoretical points is that there is a huge

power drop in case of shadowing, even if only two panels are covered. Besides the power specifications have small distortions (see Fig. 6.19). The MPPT1 corresponds to the available maximum power of the string without fog or any bad wheatear conditions. The MPPT2 and MPPT3 are related to local maximums with shadow panels.

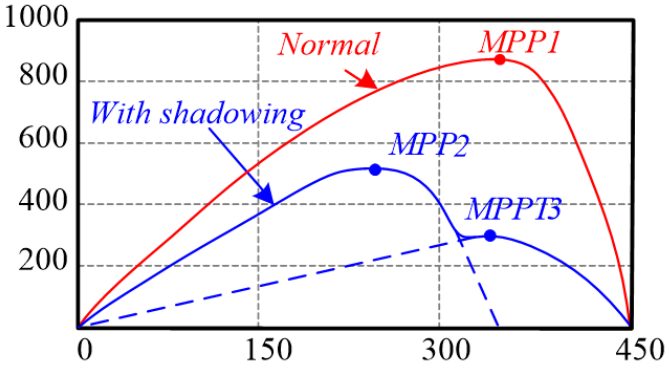


Fig. 6.19. The real power performance of the 3 serials connected panels HNS-SD140

Thus, the real PV specification allows to estimate the converter works in real environment conditions. The location of the string provides a good power generation by PV, because the low rainfall is going during each month on the north part of Ukraine. Besides, the number of sunshine days prevails than the cloudy days. The maximum SC current equals 2.7 A in case of the sun at the zenith. However, the SC current drops to 1.5 A closer to evening. Hence, the power of the PV decreases in the evening up in two times.

Other experimental research is implementing a control system and interleaved buck-boost dc-dc converter. Fig. 6.20 shows the experimental setup of the proposed system.

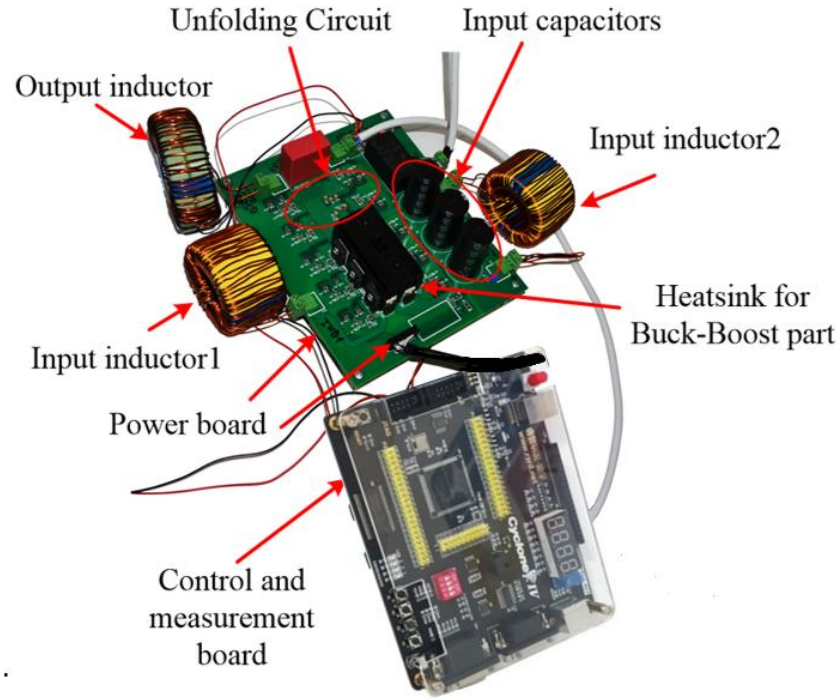


Fig. 6.20. Experimental control system withing
Cyclone IV EP4CE6E22C8N control and measurement board

The power board regarding to interleaved buck-boost converter has no significant heatsink for buck-boost part (6 transistors and 2 diodes). The input capacitors consist of 3 electrolytic capacitances, each of them equals to 100uF. All inductors were wound by hand. Two inductances have limits by current, that equals to 10 A. The grid inductor can pass the current up to 15 A.

The control and measurement boards have 5 channels of the voltage sensor. The 2 voltage sensors provide ac voltage measurement with range of voltage from -500V to 500V. Other sensors were designed only for single direction measuring. The three current sensors are also included in the board. All sensors have filter circuits with analogue differential outputs. The brain of the control board is Cyclone IV EP4CE6E22C8N FPGA of the Altera Intel. The selected FPGA has such specification:

- power supply: 5V, USB;
- Crystal oscillator: 50MHz;
- I/O lines: 84 (42+42);
- FLASH: 16Mbit;
- IO voltage level: 3.3V;
- USB - UART on the CH340 chip.

I/O performance supports several system interfaces, such as the high-speed I/O interface, external memory interface, and the PCI/PCI-X bus interface. I/Os using the SSTL-18 Class I termination standard can achieve up to the stated DDR2 SDRAM interfacing speeds. I/Os using general-purpose I/O standards such as 3.3-, 3.0-, 2.5-, 1.8-, or 1.5-LVTTL/LVCMOS are capable of a typical 200 MHz interfacing frequency with a 10 pF load. The device operates in the commercial junction temperature range (0°C to 85°C) [21].

Implementation of shown in Fig. 6.15 GMPPT algorithm was performed on VHDL withing Quartus Prime. The fragment with PI regulator code implementation is shown in Fig. 6.21.

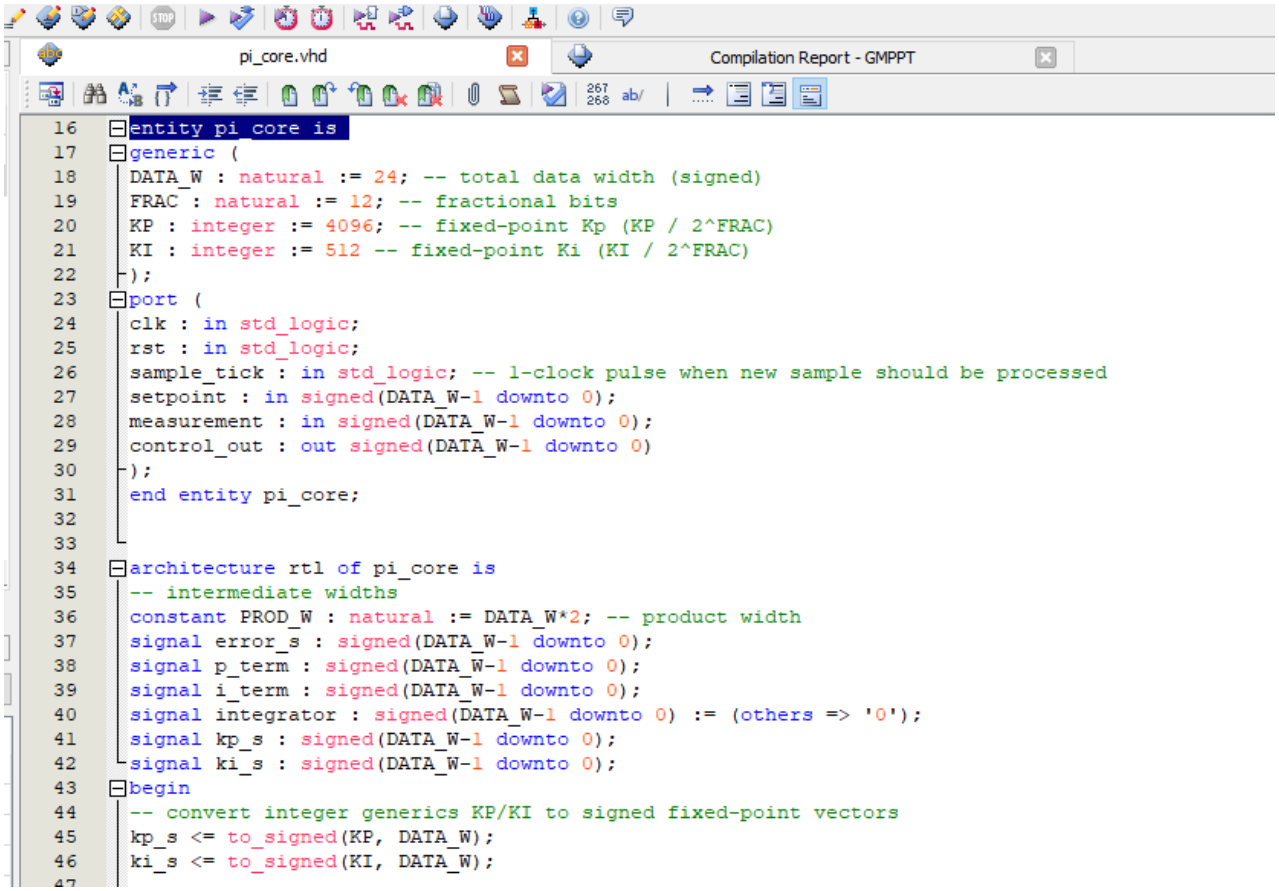


Fig. 6.21. The fragment with PI regulator code implementation on VHDL.

After project verification and configuration Quartus creates output file (see Fig. 6.22) with firmware code for FPGA.

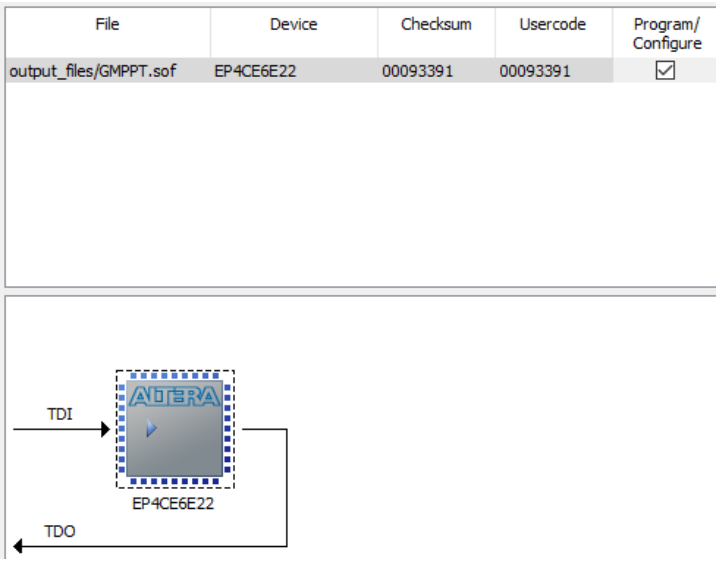


Fig. 6.22. Fragment of window with created output file for FPGA flashing.

The FPGA flashing and testing were performed with Shared Remote Experiment Environment (SREE) laboratory of Chernihiv Polytechnic National University created in DIGITRANS project. One of the SREE lab FPGA stands is shown in Fig.6.23.

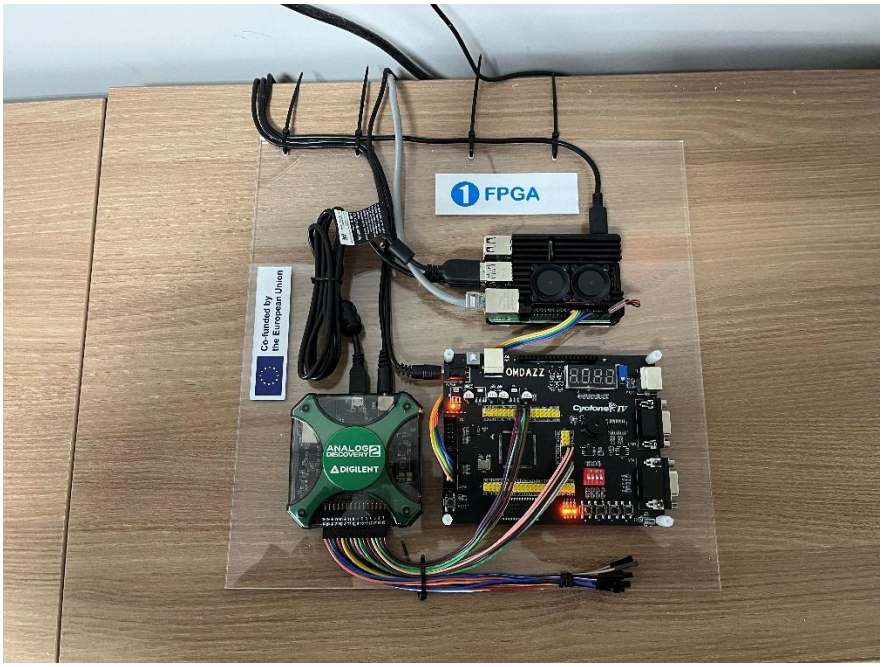


Fig. 6.23. FPGA stand of SREE laboratory

This stand consists of FPGA Cyclone IV EP4CE6E22C8N device, Analog Discovery multi-function device with USB oscilloscope and logic analyzer to test the FPGA and Raspberry Pi 4 microcomputer to connect the FPGA device and Analog Discovery within web-interface in remote mode.

Firmware file download occurs via SREE web interface as it is shown in Fig.6.24.

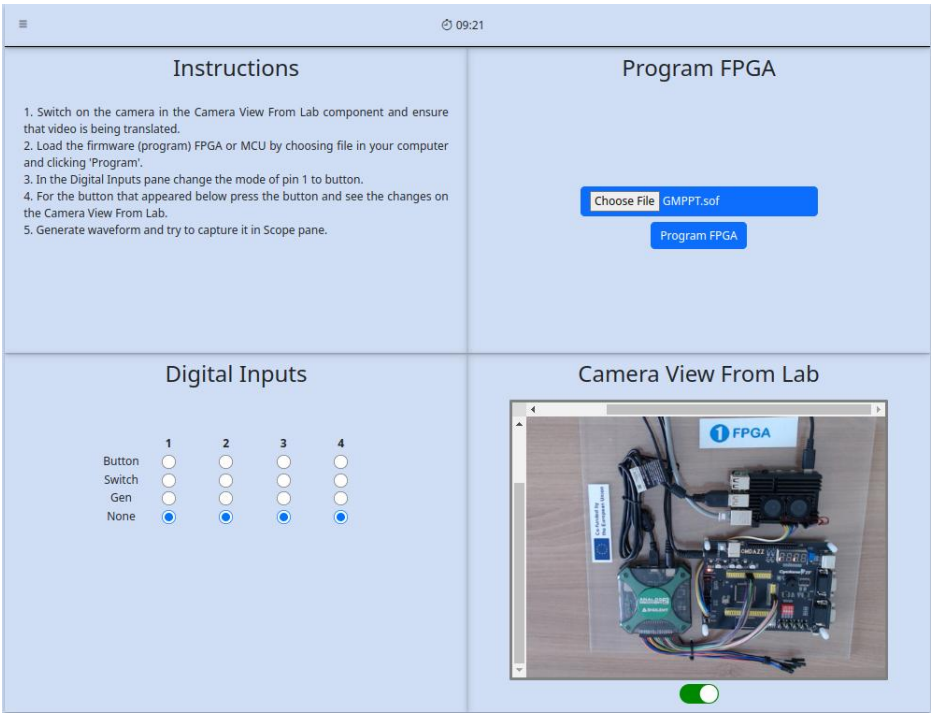


Fig. 6.24. SREE web interface

The Fig. 6.25 shows the experimental results of the proposed GMPPT method for interleaved buck-boost converter implemented on experimental control system.

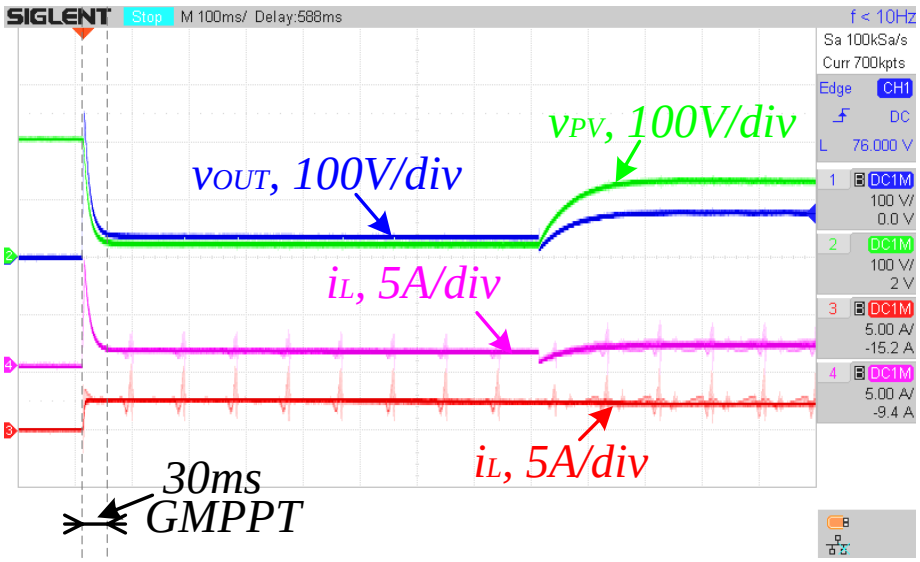


Fig. 6.25. The experimental results of the global MPPT of the interleaved buck-boost converter with using real PV string on the input side (temporary results with half of the PV set)

The time of the GMPPT depends on the current time of the day. Because if the SC current is less, the transient process time of the setting SC condition is decreasing. On the other hand, the AC case takes more time, because it needs to wait PV open circuit regime a several periods of the grid voltage. The red line corresponds to PV current of the real string. The SC current was 2.5 A. The green line regards to the PV voltage. The experiment showed that the open-circuit voltage was 210

V. The time of GMPPT took 25-30 ms. The experimental tests confirmed the theoretical hypothesis and the proposed technique.

The new technique for GMPPT is presented in this paper. The interleaved buck-boost DC-DC converter was used as the instrument for the MPPT managing.

The simple current PI regulator was chosen for control the PV current in case of simple MPPT. Contrariwise, the similar PI regulator was intended in the control system for the output voltage tuning of the GMPPT. The main strategy of the proposed GMPPT is fast set out the SC current of the panel and return the MPPT mode to the maximum power.

The simulation results within the experimental implementation confirmed the proposed technique. The real PV station was used in experiment as the input source. The control and measurement board within the power board were designed for experiment verification. The experiments showed that shadow conditions of the two solar panels decrease the maximum power down in two times. The time of the short circuit current setting of the proposed GMPPT equals to 10 milliseconds.

6.5. Summary

FPGA-based DCSs are the second direction, after MCU-based DCSs, which is becoming increasingly popular in a number of practical applications.

Although NCUs almost always dominate in terms of cost, ease of use and power consumption, programmable logic devices have the advantage of speed and flexibility. They implement algorithms in hardware and can perform many calculations in parallel.

FPGAs are the most common class of such devices, which are designed for specific tasks in Verilog or VHDL languages. Although this may take more time than programming a microprocessor in C, modern tools such as Intel Quartus Prime make this process quite simple and fast. Also, firmware created for FPGAs can be flashed, verified and tested in online mode, thanks to the Shared Remote Experiment Environment laboratory created in the DIGITRANS project.

The considered example of implementation the improved MPPT algorithm for the photovoltaic converter control system, proves the effectiveness of this approach, even in green energy field. The presented data of the model and full-scale experiments fully confirm this.

References

1. Comparing Common Digital Logic Components. [Online]. Available: https://nandland.com/lesson-2-fpga-vs-micro-vs-asic/.
2. FPGA Advantages and Disadvantages: A Comprehensive Overview. [Online]. Available: https://www.rfwireless-world.com/terminology/fpga-advantages-and-disadvantages . Accessed on July 2025.
3. List of FPGA companies. [Electronic source]. [Online]. Available: https://hardware-bee.com/list-fpga-companies.
4. FPGA Manufacturers. [Online]. Available: https://www.marketsandmarkets.com/ResearchInsight/fpga-market.asp.
5. Leading FPGA companies taking the global stage. [Online]. Available: https://fpgainsights.com/fpga/top-fpga-companies-globally.
6. Monmasson E., Cirstea M. FPGA Design Methodology for Industrial Control Systems—A Review. IEEE Transactions on Industrial Electronics 54(4):1824 – 1842, 2007.
7. Hussein F. Hexarray: A Novel Self-Reconfigurable Hardware System. Thesis for: Doctor of Philosophy in Electrical and Computer Engineering, 2017. 2017. DOI: 10.13140/RG.2.2.25809.02406.

8. ASIC Design Flow. [Online]. Available: https://www.chipverify.com/verilog/asic-soc-chip-design-flow.
9. PAL vs. CPLD vs. FPGA: What are the differences between them? [Online]. Available: https://www.electroniclinic.com/pal-vs-cpld-vs-fpga-what-are-the-differences-between-them/.
10. Verilog vs VHDL: A Comprehensive Comparison. [Online]. Available: https://www.wevolver.com/article/verilog-vs-vhdl-a-comprehensive-comparison.
11. Intel Quartus Prime Pro and Standard Software User Guide. [Online]. Available: https://www.intel.com/content/www/us/en/support/programmable/support-resources/design-software/user-guides.html.
12. Intel Quartus Prime Standard Edition User Guide: Platform Designer. [Online]. Available: https://www.intel.com/content/www/us/en/docs/programmable/683364/18-1/creating-a-system-with.html.
13. Guo B., Su M., Sun Y., Wang H., Liu B., Zhang X., Pou J., Yang Y., Davar P. Optimization Design and Control of Single-Stage Single-Phase PV Inverters for MPPT Improvement. IEEE, Transactions on Power Electronics, 2020.
14. Vinnikov D., Roasto I., Quasi-Z-Source-Based Isolated DC/DC Converters for Distributed Power Generation. IEEE Trans. Ind. Electron., Vol. 58, no. 1, pp. 192-201, 2011.
15. Husev O., Strzelecki R., Blaabjerg F., Chopyk V., Vinnikov D.. Novel family of single-phase modified impedance-source buck-boost multilevel inverters with reduced switch count. IEEE Trans. Power Electron., vol.31, no. 11, pp. 7580-7591, 2016.
16. Aamri F. EL, Maker H., Sera D., Spataru S., Guerrero J. M., Mouhsen A. A Direct Maximum Power Point Tracking Method for Single-Phase Grid Connected PV Inverters. IEEE, Transactions on Power Electronics, 2017.
17. Husev O., Vinnikov D., Roncero-Clemente C., Pimentel S., Strzelecki R. MPPT and GMPPT Implementation for Buck-Boost Mode Control of quasi-Z-Source Inverter. IEEE Transactions on Industrial Electronics, Volume: 69, Issue: 11, 2022. DOI:10.1109/TIE.2021.3125658.
18. Husev O., Matiushkin O., Roncero-Clemente C., Blaabjerg F., Vinnikov D. Novel Family of Single-Stage Buck-Boost Inverters Based on Unfolding Circuit. IEEE, Transactions on Power Electronics, 2018.
19. Matiushkin O., Husev O., Vinnikov D., Roncero-Clemente C. Model Predictive Control for Buck-Boost Inverter Based on Unfolding Circuit. IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON), 2019.
20. Fesenko A., Matiushkin O., Husev O., Khandakji K., Velihorskyi O. Feasibility Study of Interleaving Approach for Buck-Boost Inverter with Unfolding Circuit. IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON), 2019.
21. Altera. Cyclone IV Device Datasheet. [Online]. Available: https://cloud.smarts.zp.ua/s/exPdENX3f36BSJm?dir=/09_Datasheets&openfile=true