MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
DNIPRO UNIVERSITY OF TECHNOLOGY

I. Laktionov, O. Vovna, G. Diachenko

DIGITALIZATION AND INTELLECTUALIZATION OF INDUSTRIAL ECOSYSTEMS. IN 2 PARTS. PART 1: INTERNET OF THINGS

Textbook

Dnipro
DniproTech
2024

UDC 004.75
L 18

Approved for publication by the Academic Council of Dnipro University of Technology as a textbook for PhD students in the speciality 123 Computer Engineering (Protocol No. 10 of 10.09.2024).

Reviewers:

A. Perekrest, Dr. Sc. (Tech.), Professor, Head of the Department of Computer Engineering and Electronics (Kremenchuk Mykhailo Ostrohradskyi National University, Kremenchuk).

S. Vasylets, Dr. Sc. (Tech.), Professor, Professor in the Department of Automation, Electrical Engineering and Computer-Integrated Technologies (National University of Water and Environmental Engineering, Rivne).

Laktionov I.

L 18 Digitalization and Intellectualization of Industrial Ecosystems [Electronic resource]: textbook: in 2 parts. Part 1. Internet of Things / I. Laktionov, O. Vovna, G. Diachenko; Ministry of Education and Science of Ukraine, Dnipro University of Technology. – Dnipro: DniproTech, 2024. – 241 p.

The fundamental theoretical provisions and practical principles of creating IoT systems and networks of diverse architectures and applications are presented. The theoretical and applied aspects of modern sensor, microcontroller and network technologies, as well as software tools in synthesising IoT systems and networks, are considered, taking into account modern world achievements in the field of information and computer technologies.

The textbook is recommended for applicants for the degree of Doctor of Philosophy in the speciality 123 Computer Engineering. It can benefit higher education students in many specialities of information and technical direction and practitioners in computer, information, and digital technologies.

UDC 004.75

TABLE OF CONTENTS

INTRODUCTION

Current challenges in the intellectualization, computerization and digitization of technological and business processes in various types and forms of production demand a constant search for ways to optimise and improve them. One of the most progressive information technologies that allows the implementation of comprehensive software and technical retooling of productions to improve their long-term sustainability and competitiveness is the Internet of Things (IoT). An IoT concept transforms ordinary objects in the physical world into smart things for people and allows their integration into a unified infocommunication network, solving tasks related to remote monitoring and control of various objects, processes and phenomena with intelligent support decisions.

The academic course ‘Digitalization and Intellectualization of Industrial Ecosystems. Part 1: Internet of Things’ is one of those studied by a significant number of students in technical universities, especially those specialising in IT-related fields. This academic course aims to develop students' knowledge and skills in understanding and competently applying the theoretical and applied principles of the operation of comprehensive software and hardware solutions of Internet of Things systems and networks at the physical, network and software levels, making IT professionals competitive in the domestic and global job markets.

The textbook for the course ‘Digitalization and Intellectualization of Industrial Ecosystems. Part 1: Internet of Things’ has been designed considering the integration of modern global achievements in the fields of sensor, microprocessor, networking, cybersecurity and software technologies when justifying and synthesising optimal comprehensive software and hardware solutions of Internet of Things systems and networks.

The theoretical part of this academic course, as reflected in the relevant chapters of the textbook, involves the assimilation of five main content sections, which are further subdivided into subsections for effective, logical and structured assimilation of the material.

In the first chapter 'General Information about the Internet of Things' the fundamental provisions regarding the formation of the Internet of Things, the key application areas of IoT systems and networks, their global development prospects, as well as basic terms and definitions in the field of IoT, are given.

In the second chapter ‘Structural and Algorithmic Organisation of Internet of Things Systems and Networks’ theoretical and applied principles of architectural organisation and the creation of standard models of IoT systems and networks in different application areas are given. In addition, the chapter discusses the main mechanisms for ensuring the cybersecurity of IoT devices.

The third chapter ‘Hardware and Software Tools at the Level of Physical Devices’ provides structured information about sensors, microcontrollers, microcomputers and actuators used in modern hardware and software solutions of IoT systems and networks when constructing the field level of data aggregation and processing.

In the fourth chapter ‘Methods and Means for Creating the Network Level’ the main technical and functional capabilities of modern architectural approaches, network protocols and IoT gateways for implementing approaches for exchanging data and information messages in IoT systems and networks are given and substantiated.

The fifth chapter ‘Data Aggregation and Processing Technologies’ reflects the theoretical and applied principles of centralised and distributed data aggregation and intelligent data processing at different hierarchical levels of IoT systems and networks for their implementation in various industrial, infrastructure and business sectors.

Significant attention in the academic course ‘Digitalization and Intellectualization of Industrial Ecosystems. Part 1: Internet of Things’ is devoted to the practical component, which allows students to acquire skills in the development, computer modelling and testing of IoT systems and networks of various complexity and functional purpose. The practical part of this academic course, which is outlined in the sixth chapter ‘Practicum’, involves the completion of eight works using computer modelling methods in the ergonomic open-source software of CupCarbon IoT. The practical works focus on developing and testing of basic algorithms for the

operation of IoT network components, developing and testing functional algorithms for wireless sensor networks, developing and testing models and algorithms for data exchange in wireless sensor networks, developing and testing models and scenarios for data routing in wireless sensor networks, developing and testing models and scenarios for group data exchange in IoT networks, developing and testing models and functional algorithms of the digital sensors in IoT networks, developing and testing the functional algorithms of analogue sensors when designing IoT networks, investigation of typical models of radio modules and wireless data exchange technologies when designing IoT networks.

Successful completion of the academic course ‘Digitalization and Intellectualization of Industrial Ecosystems. Part 1: Internet of Things’ involves students acquiring the following knowledge and skills: understanding and effective use of modern methods and means for developing software and hardware of IoT systems and networks; substantiation of the component base as well as synthesis of structural and algorithmic organisations of IoT systems and networks for various applications; testing and validation of IoT systems and networks of different architectures, including the use of computer modelling approaches; development of measures and approaches for optimising software and hardware solutions of IoT systems and networks.

The textbook ‘Digitalization and Intellectualization of Industrial Ecosystems. Part 1: Internet of Things’ has been designed and written taking into account the authors' extensive practical experience in participating in a significant number of scientific and educational IT projects.

The authors express their deep gratitude to the reviewers of the textbook: Professor Andrii Perekrest and Professor Sviatoslav Vasylets for their valuable comments and objective assessment of the materials in the textbook.

The authors invite scientists and practising professionals to further cooperate in scientific research and educational projects in the field of computer, information and digital technologies.

CHAPTER 1
GENERAL INFORMATION ABOUT THE INTERNET OF THINGS

1.1 The Formation of the Internet of Things

The term ‘Internet of Things’ (IoT) began to be used relatively early, in the late 1990s. It was coined to popularise the revolutionary RFID (Radio-Frequency Identification) technology of that time. At that period of time, the ‘things’ mainly referred to various products in stores and warehouses that were equipped with radio-frequency tags (RFID tags). After that, the term ‘IoT’ gained widespread usage and soon started to be applied to almost all Machine-to-Machine (M2M) communication technologies, including sensors in typical SCADA systems and infocommunication networks.

For the next decade (from 2000 to 2010), the Internet of Things (IoT) existed in a relatively passive mode. This technology occupied a small niche while the mobile boom was actively gaining momentum. According to available statistical data, there were approximately 6.3 billion people on Earth and around 500 million devices were connected to the global Internet in 2003. The ratio of connected devices to world population was 0.08 in 2003. Therefore, according to analytical research by Cisco IBSG the Internet of Things was in its early stages of development in 2003 (see Fig. 1.1).

According to Cisco IBSG’s analysis, there were 25 billion connected devices in 2015 and this number reached 50 billion in 2020. It’s important to note that the calculations visualised in Fig. 1.1 take into account the entire global population, but the majority of people at that time and even now do not have access to the Internet. If only those people who have access to and use the Internet are considered, the number of connected devices per user is significantly higher. It is estimated that approximately 4.8 billion people are currently using the Internet. Therefore, the number of connected devices per Internet user is about 10.5.

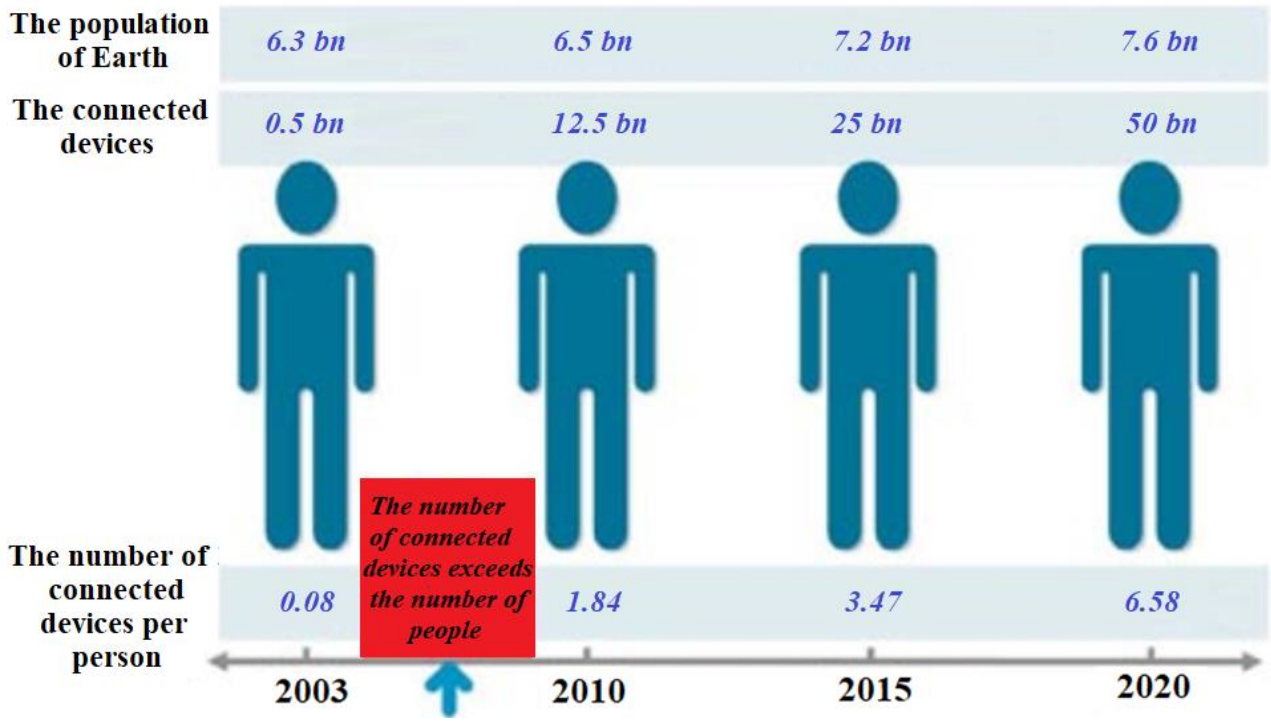


Fig. 1.1. Infographic illustrating the dynamics of connected devices to the Internet

Therefore, it can be noted that it required almost a decade for the term 'Internet of Things' to be actively used in everyday life (see Fig. 1.2). The Internet of Things, along with artificial intelligence (AI), has become a leading direction in the development of information technologies since 2000. In 2008, the IPSO Alliance initiated the cooperation of companies supporting the development of technologies related to the Internet of Things. In turn, this fact stimulated the development and implementation of global companies in the field of information and digital technologies.

The next milestone in the development of the Internet of Things was the implementation of data collection on Wi-Fi network usage by the Google Street View service in the summer of 2010. This served as a stimulus for experts to develop a new data transmission protocol that would allow devices to exchange data. In the same year, China announced its plans to include the Internet of Things in its list of priority research areas for the next five years. Therefore, during this period, it became evident that the collection, processing, analysis and storage of data were of interest not only to large-profile corporations but also to government structures.

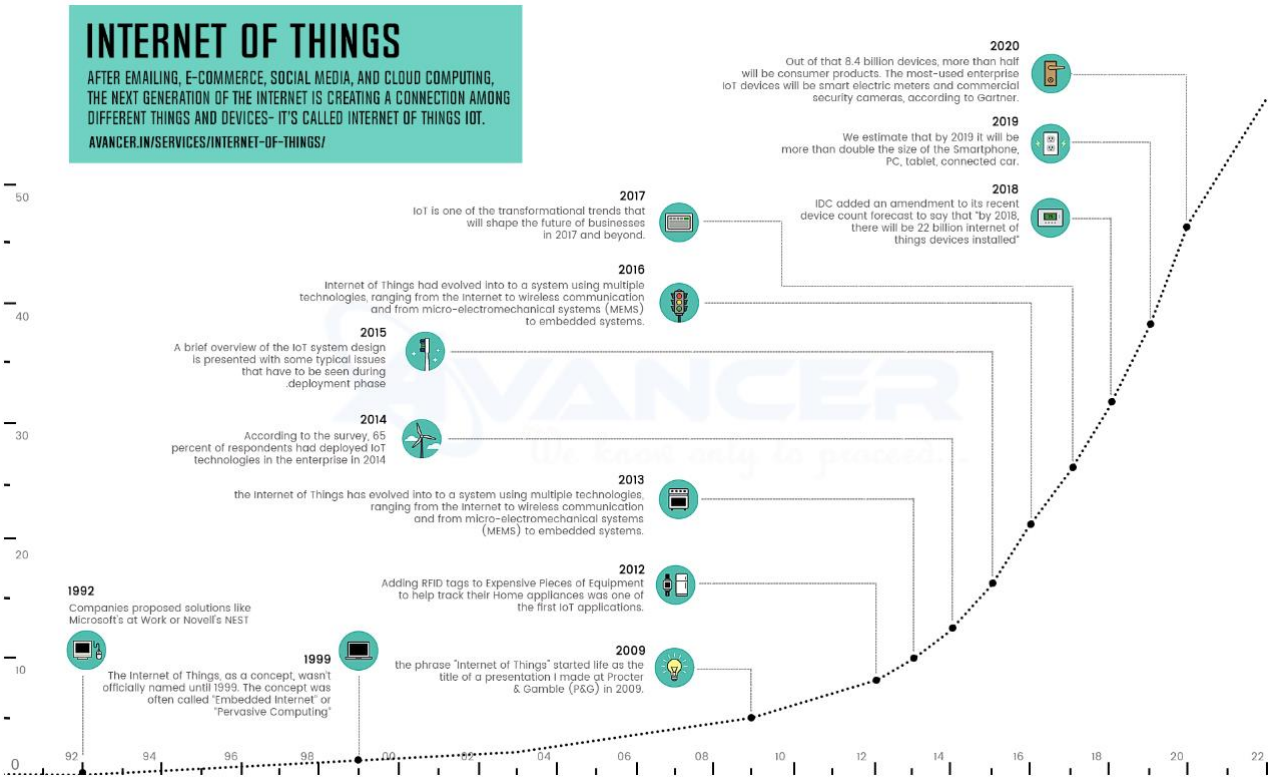


Fig. 1.2. Dynamics of the IoT development

In 2011, the globally renowned company Gartner, specialising in digital technology market research, included IoT in its list of the most promising emerging technologies.

In 2012, the largest European Internet conference, LeWeb, was entirely dedicated to the topic of the Internet of Things. The world-authoritative journals such as Forbes, Wired and Fast Company started actively using the term 'Internet of Things'. The global scientific community started discussing the Internet of Things and companies started a technology race. In 2013, IDC company published its results of analytical research, predicting that the growth of the Internet of Things technology market will reach 8.9 trillion dollars by 2020.

In January 2014, Google took over the company Nest for $ 3.2 million. Nest was involved in the development of 'smart home' technologies and the creation of devices and systems for industrial building automation. This event is seen as a recognition by the global market of digital technologies that the Internet of Things is the immediate future. In the same year, the largest American technology exhibition, the Consumer

Electronics Show, took place in Las Vegas under the topic of the Internet of Things. This marked a dynamic development in the devices, systems and networks of the Internet of Things.

As of today, the pace of development of the Internet of Things (IoT) is exceeding expectations, as IoT technologies simultaneously solve several key practical tasks. These include how to efficiently aggregate data with minimal resource expenditure, process data quickly, transmit data securely with minimal costs and optimise data storage.

1.2 Key Terms and Definitions in the Internet of Things

The term ‘Internet of Things’ (IoT) was first proposed and used by Kevin Ashton, one of the three founders of the Auto-ID Centre at the Massachusetts Institute of Technology, in 1999. As of today, there are several definitions of the term ‘Internet of Things’, but each of them, according to the general consensus among practising experts and scientists in the field of information and digital technologies, is not precise enough.

At present, the most successful definition is considered to be the one formulated and proposed by Gartner company: The Internet of Things is a network of physical objects equipped with embedded technologies to interact with the external environment, transmit information about their state and receive data from the outside.

Another fairly common definition that allows understanding the general structure and principle of operation of IoT systems and networks is as follows: the Internet of Things is a network concept consisting of interconnected physical devices containing embedded sensors and software that enable the transmission and exchange of data between the physical world and computer systems using standard infocommunication protocols. In addition to sensors, such a networked organisation can include executive devices integrated into physical objects and connected via wired or wireless technologies. These interconnected hardware and software tools are able to read and act on objects and their actuators, support programming, scaling and

identification functions and enable human involvement through the use of intelligent human-machine interfaces.

The main commonly used terms in the field of the Internet of Things are as follows:

The IoT ecosystem is a comprehensive integration of software and hardware tools, services and technologies used in the creation of Internet of Things systems and networks.

IoT devices are objects that essentially constitute the ‘things’ in the global Internet of Things technology. Typically, these are small-scale autonomous computing devices capable of data aggregation or being used to control other devices. They act as an interface between the physical and digital worlds. IoT devices can vary in size, design and technological complexity, depending on their specific application within a given deployment of IoT systems and networks.

Smart technologies are technologies that utilise artificial intelligence, machine learning and big data analytics to achieve the goal of cognitive awareness of objects.

Operational technology is the hardware and software used on a technical object for tracking, diagnosing, monitoring and controlling equipment, processes and infrastructure.

Cloud technologies are technologies of distributed processing of digital data, with the help of which computer resources are provided to Internet users as an online service.

Fog computing technologies are a type of infocommunication technologies characterised by the distribution of computations among devices within the network organisation of the Internet of Things. When using fog computing technologies, secondary data is generated through shared resources and transmitted to the cloud server.

A physical device is a real Internet of Things device connected to the centre of the network or system.

An end device is a device without connected devices, typically end devices connect to a gateway.

IoT hub is a fully controlled service that provides reliable and secure bidirectional communication between IoT devices and the back-end of an IoT solution.

IoT agent is a component of the IoT runtime environment that is responsible for deploying and monitoring modules.

An IoT device template is a schema that defines the characteristics and behaviour of a specific type of device connected to the system or software.

The physical level of IoT is one of the fundamental functional levels of IoT systems and represents the systemic implementation of hardware, including sensors, microcontrollers and network devices.

The IoT application level is one of the fundamental functional levels of IoT systems, which includes protocols and interfaces used by devices for identification and communication among themselves.

The analytical factor of IoT is the systematic integration of software that implements the processing and analysis of data received from IoT devices.

It is worth noting that the above list of terms and definitions is not exhaustive but constitutes a key foundation for understanding the general structure and purpose of IoT systems and networks. In the following sections of this textbook, detailed and specialised definitions in the field of the Internet of Things are provided according to their thematic orientation.

1.3 The Formula of the Internet of Things

As mentioned above, Internet of Things (IoT) systems and networks are aimed at the comprehensive implementation of a range of functions involving the generation, processing, transmission and intelligent analysis of measurement data related to physical objects, processes and phenomena. Therefore, the following formula underlies such network organisations:

$$\begin{aligned}&\textit{Physical objects}\ \left(\textit{processes, phenomena}\right)\ +\\ &+\textit{Hardware devices}\ \begin{pmatrix}\textit{sensors, microcontrollers,}\\ \textit{actuators, network equipment}\end{pmatrix}\ +\\ &+\textit{Software for data processing + Internet = Internet of Things.}\end{aligned}$$

A graphic interpretation of this formula, which describes the fundamental principles of operation of IoT systems and networks, is shown below in Fig. 1.3.

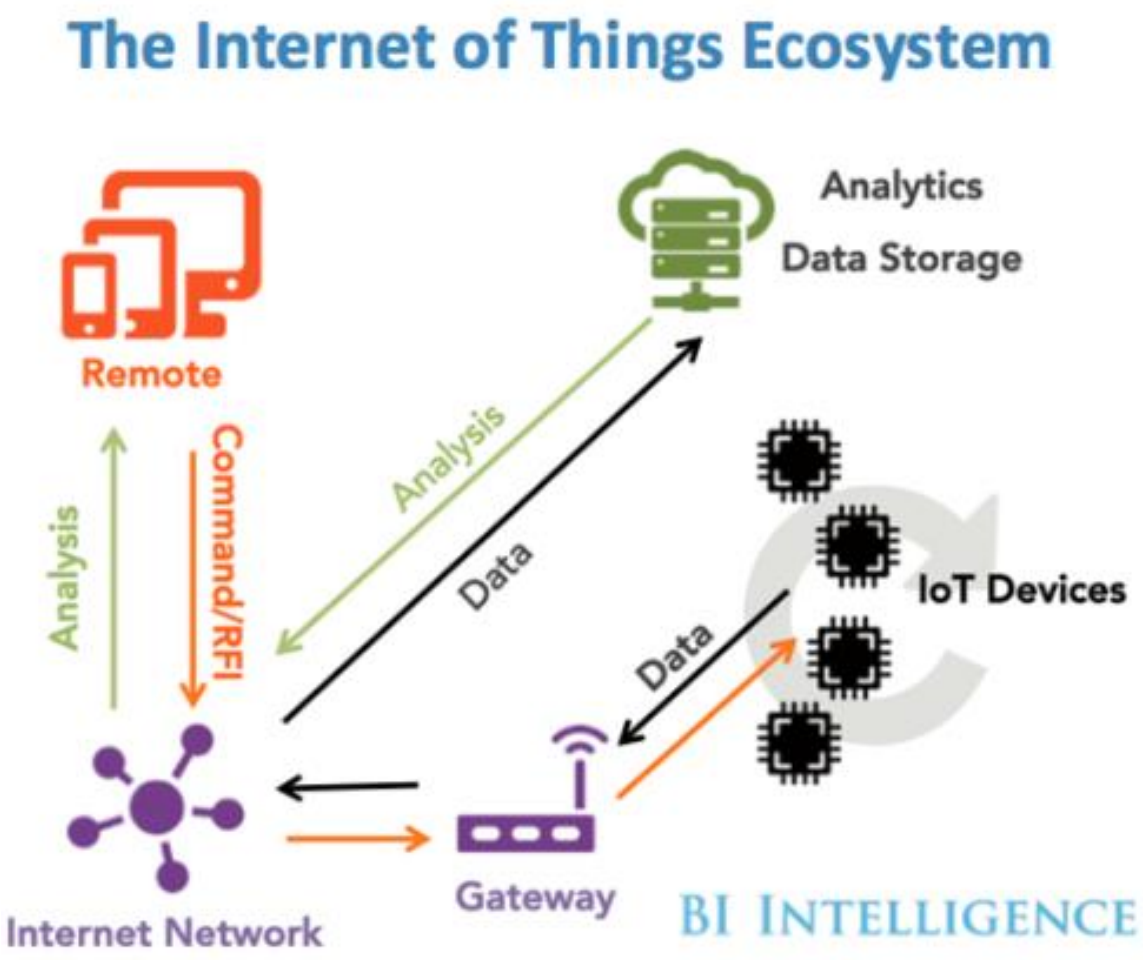


Fig. 1.3. Graphical interpretation of the global operating principle of IoT networks

As can be seen from the formula provided above, the foundation of the Internet of Things is a set of modern hardware and software technologies that implement this formula (see Fig. 1.4).

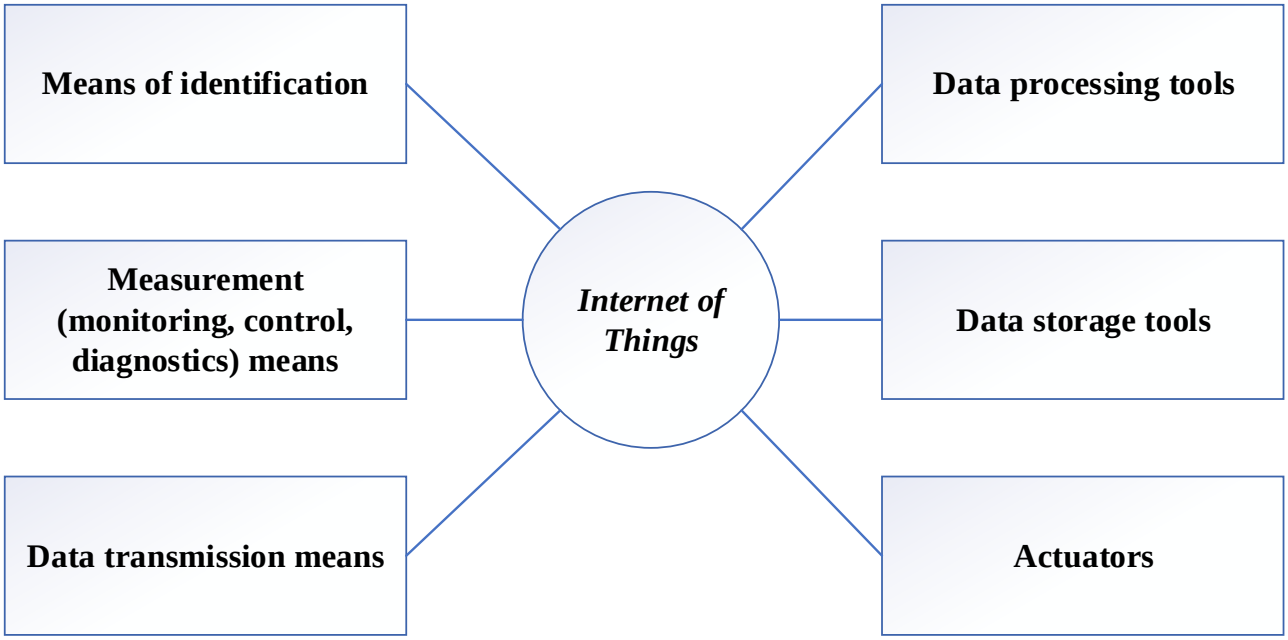


Fig. 1.4. Hardware and software implementation of the Internet of Things formula

Means of identification. All objects (processes, phenomena) in the physical world participating in the organisation of the Internet of Things, whose data may not currently be entering the Internet, must have a unique identifier. Various well-known methods can be used for the automatic identification of objects: radio frequency, optical, infrared, etc. The main problem arising during the identification of Internet of Things objects lies in standardising methods based on different physical principles and technologies.

Measurement (monitoring, control, diagnostics) means. These technical means must enable the transformation of measurement data regarding a wide range of physical, chemical and biological parameters into data suitable for further processing. In the concept of the Internet of Things, measurement tools can be built using both individual sensors and complex measurement installations, including intelligent ones. The main task concerning measurement devices within IoT systems and networks is to achieve a high degree of autonomy in operation through the use of alternative energy sources.

Data transmission means. In order to solve the task of network exchange of measurement data, a significant number of wired and wireless technologies can be used today. The main task related to data transmission means that are part of IoT systems

and networks is to achieve high reliability and fault tolerance during network data exchange, as well as information security.

Data processing tools. One of the main features of IoT networks that distinguishes them from other infocommunication technologies is the systematic integration of intelligent algorithms for processing measurement data at different hierarchical levels of such networks (cloud, fog and edge computing). The main task of the data processing tools that are part of IoT systems and networks is to ensure high throughput and responsiveness to specific situations.

Data storage tools. With over fifty billion devices currently connected to the Internet, they have the potential to generate 45 billion terabytes of data, necessitating the physical implementation of storage solutions. The key role of data storage tools within IoT systems and networks is to ensure high-performance access to data, as well as information security.

Actuators. These are technical means designed to convert digital signals coming from the informational levels of IoT systems and networks into actions that influence physical objects (processes, phenomena). The main task in improving such devices as components of IoT systems is to ensure high compatibility in terms of hardware, software and design with intelligent sensors, as well as to increase the operational and precision characteristics of the processing of control signals.

1.4 Key Areas of Application of the Internet of Things

As mentioned above, there has been a rapid growth in the number of devices connected to the Internet in recent years. These devices are applicable across a wide range of industries. Primarily, the key application areas of IoT systems and networks are as follows: technological processes in industrial facilities, transportation and logistics, infrastructure objects (smart homes and cities, educational and healthcare institutions), municipal services, agriculture and energy, as well as the military-industrial complex. The main application areas of Internet of Things systems and networks are systematised and presented in Fig. 1.5.

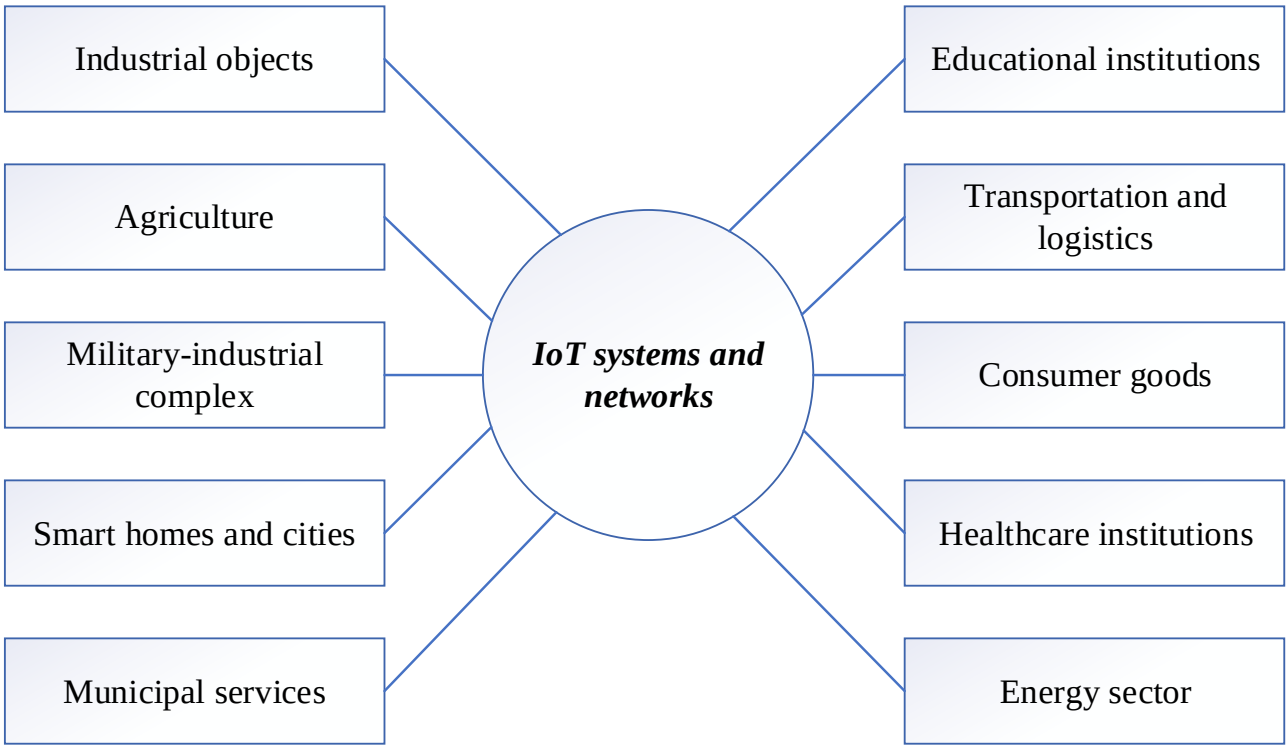


Fig. 1.5. Main application areas of IoT systems and networks

IoT in industry. This application area of IoT systems and networks is leading the spectrum of practical applications, as evidenced by the emergence of the digital technology direction known as the Industrial Internet of Things (IIoT). The Industrial Internet of Things represents a systematic organisation of interconnected computer and cyber-physical networks, along with connected industrial and technological objects equipped with embedded sensors and integrated software for data aggregation, exchange and transformation. It enables remote monitoring and adaptive automatic control.

IoT in agriculture. Smart devices and systems are actively used in all areas of agriculture. These technological tools help improve the efficiency of resource utilisation, preserve crops and protect plants and animals from the negative effects of destabilising environmental factors. Among the most popular IoT devices in agriculture are: computerised monitoring devices for soil and climatic parameters, as well as for diagnosing the health and location of domestic animals and birds, drones for carrying out agricultural procedures, robotic tools for soil cultivation and

harvesting, etc.

IoT in healthcare. This application of IoT systems is one of the fastest developing. IoT devices significantly improve the diagnostic indicators of diseases and the speed and reliability of decision-making regarding treatment techniques and approaches. Such intelligent healthcare systems enable the assessment of individual approaches to the treatment of individual patients, taking into account a significant number of factors influencing health.

Smart homes and cities. This area is one of the most popular and promising in terms of the use and sustainable development of the Internet of Things as a global concept of intellectualization and digitization. The implementation and use of IoT technologies in household devices in smart homes and in the infrastructure of smart cities significantly improve safety and ergonomics, enhance the ecological conditions of the environment, optimise urban transport, rationalise the use of energy and resources, etc.

IoT in energy. Smart technologies in the energy sector are used at all hierarchical levels, from power plants to end consumers, including consumer meters that send monitoring data to the electricity supplier; generation and tracking of payments assigned to users; remote monitoring and control of power plant and distribution network equipment; planning of current equipment maintenance; continuous equipment condition diagnostics with decision support; control of loads on electrical networks in all their sections, etc. The use of IoT technologies in the energy sector improves the indicators of stability, reliability and safety of operation, as well as enhances the quality of services provided and minimises losses. It is worth noting that smart technologies in the energy sector are directly related to environmental aspects, such as ‘green’ types of electricity and electric transport.

IoT in educational institutions. Currently, a significant number of IoT technologies are being implemented in various segments of the educational process. Among the most popular smart devices in the educational process are as follows: smart classrooms, reliable access systems to educational institutions, digital blackboards, voice assistants, artificial intelligence tools for diagnosing current performance, etc.

The integration of IoT devices into physical and cyber-physical educational environments significantly improves the indicators of informatisation and interconnectedness of participants in the educational process.

IoT in transportation and logistics. As for now, a significant number of transportation and logistics companies are adopting advanced telematics tools based on IoT technologies. Modern IoT devices and remote monitoring and control systems for transport, in organic combination with cutting-edge developments in the field of intelligent logistics algorithms, allow achieving the following effects: continuous tracking of the location of moving objects; access to comprehensive information during the movement of objects (duration, number of stops, route, fuel consumption, etc.); optimisation of routes and all logistics-related processes; real-time and ergonomic information for customers about the status of their orders at each stage of the journey, etc. IoT telematics technologies improve the efficiency and productivity of the transport and logistics segment by optimising operational costs and improving the efficiency of vehicle fleet management. They also increase the informativeness, ergonomics and promptness of the services provided by such companies to end users.

IoT in the military-industrial complex. The dynamic development of IoT technologies drives the digital transformation of almost all sectors of industry, business and daily life and the military-industrial complex is no exception. Thanks to digital technologies, the methods and principles of managing and applying armed forces in both peacetime and wartime are undergoing radical transformations. The main global trends in the digital transformation of the military-industrial complex include weapon and military equipment digitization; big data analytics in defence; artificial intelligence and blockchain technology in defence; robotic military equipment; 3D printing technologies for the defence industry; automated support-decision systems and technologies in the defence complex; identification and readiness systems for chemical and biological threats; cybersecurity; virtual military medical technologies; virtual and augmented reality technologies for personnel training. The aforementioned areas of IoT technologies for the military-industrial complex are at different stages of development, but predictive analytical research by scientists and leading companies in

the defence sector worldwide demonstrates the priority and significant dynamics of development in these directions of information and digital technologies.

The list of applied areas provided above is not exhaustive but highlights a significant spectrum and priority directions for the development, research and implementation of hardware and software solutions of IoT devices, systems and networks in the context of global trends towards digitization. It is also important to emphasise that the Internet of Things is a relatively young and expanding field in the sustainable development of information and digital technologies. This fact gives rise to a number of scientific and technical challenges related to standardisation, certification and compatibility of IoT technology solutions at the hardware, software and design levels. At present, several leading companies, including Amazon, Huawei, Cisco, NEC, Intel, Samsung, Qualcomm and others, support and contribute to the development of the OneM2M standard. The primary goal of this standard is the standardisation and interoperability of IoT solutions at the enterprise level.

1.5 Global Perspectives on the Development of the Internet of Things

As mentioned above, the Internet of Things is a dynamic field in the sustainable development of information and digital technologies. Hardware and software solutions of IoT systems are finding more and more diverse applications in various sectors of the national economy, the social sphere and the military-industrial complex. Currently, analysts and researchers in the field of IoT technologies have identified five main global trends in the development of such systems, as shown in Fig. 1.6. These trends are related to all architectural levels of IoT networks, namely:

– development and integration of intelligent algorithms for distributed computing at the physical level of IoT systems and networks;

– improvement of the efficiency and reliability of network data exchange;

– scaling IoT systems and increasing the computational capacity of cloud services;

– synthesis of cyber-physical systems complementary to monitored and

controlled physical objects (processes, phenomena);

– improvement of data protection mechanisms and prevention of cyber-attacks on IoT networks.

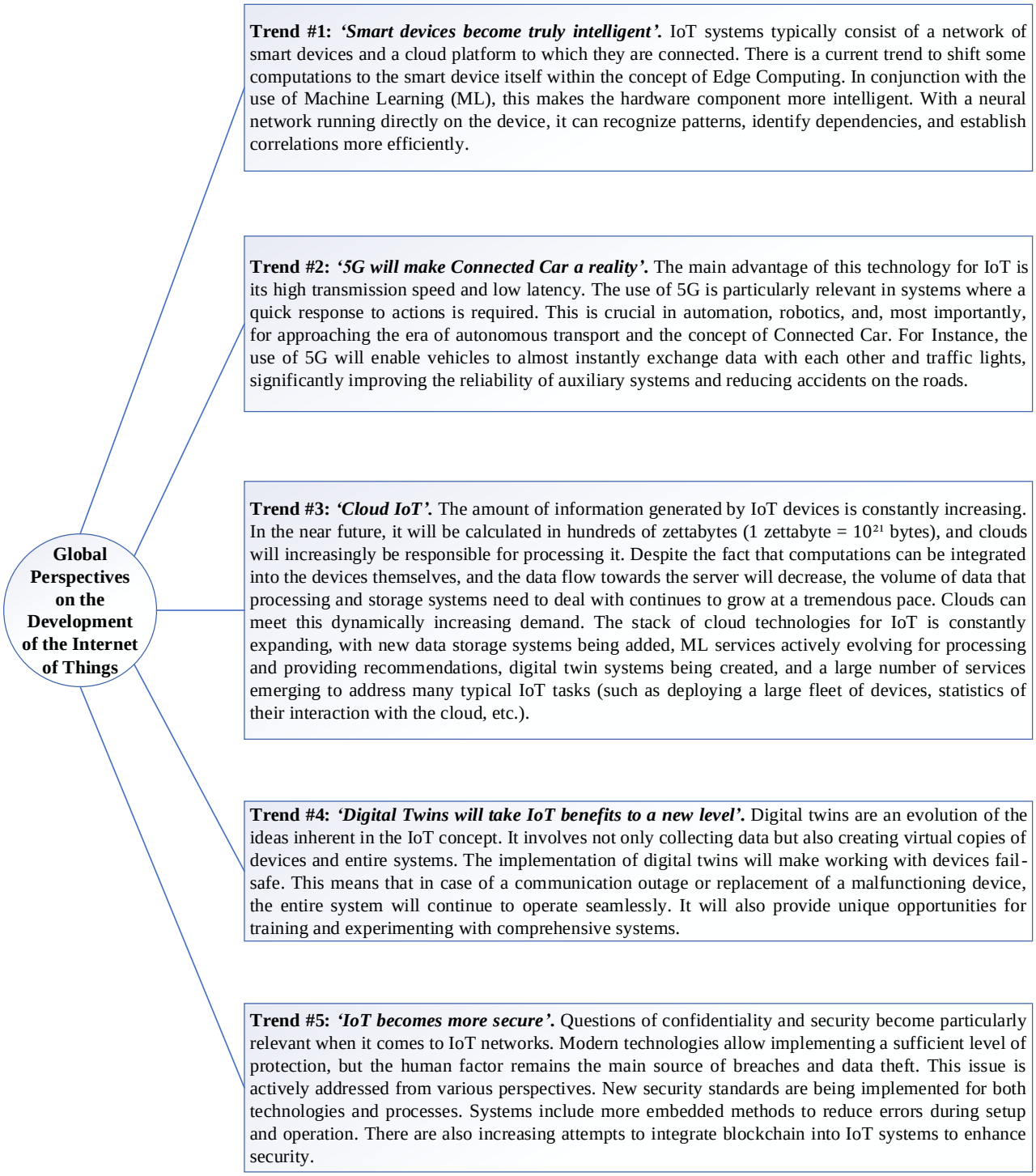


Fig. 1.6. Trends in the development of IoT technologies

In addition, it is important to note that the Internet of Things is one of the highly

effective directions in the global digitization of industry and business within the framework of the Industry 4.0 concept. It integrates seamlessly with other information technologies aimed at the intellectualization of manufacturing, technological and business processes, as shown in Fig. 1.7.

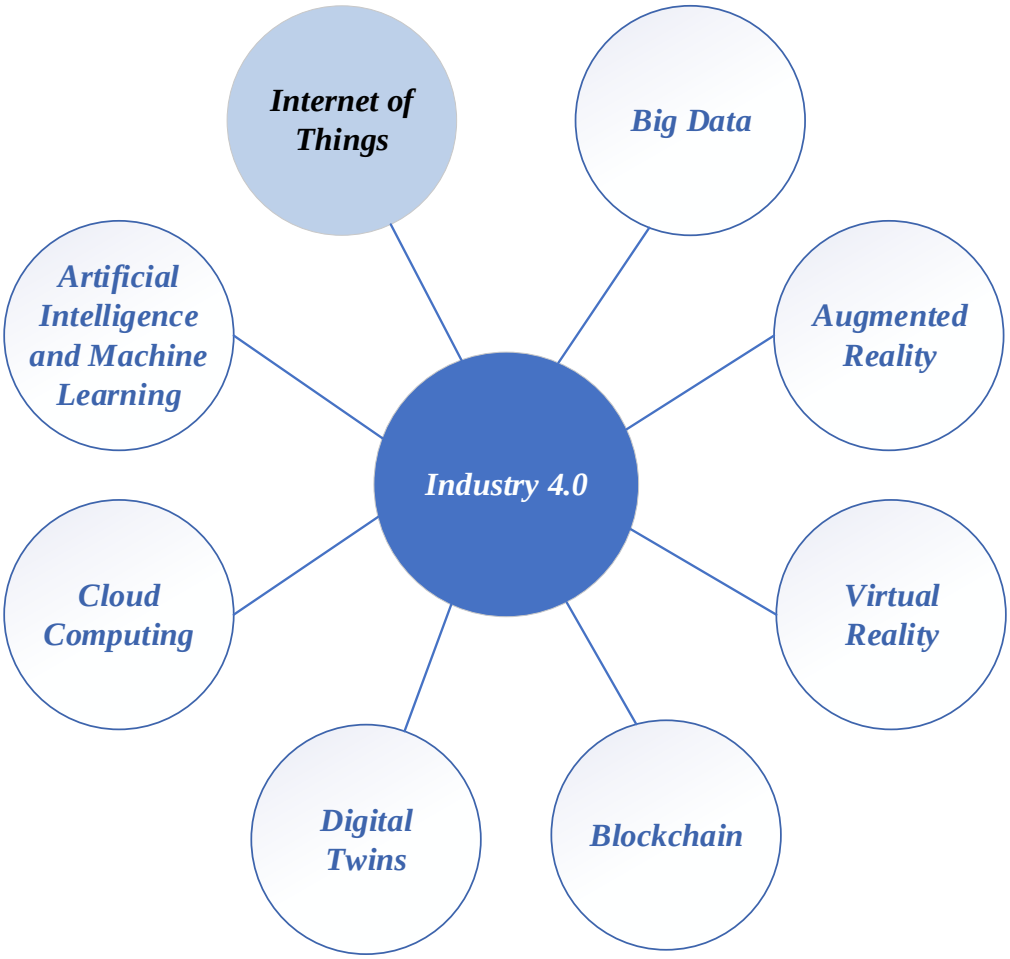


Fig. 1.7. Digital transformation technologies for industry and business within the Industry 4.0 concept

Augmented Reality (AR) is the result of introducing any sensory data into the perception field for the purpose of supplementing information about the surroundings and enhancing the perception of information.

Virtual Reality (VR) is a world created by technical means and conveyed to people through their senses. Virtual reality simulates both influences and corresponding reactions to these influences. In order to create a convincing sense of

reality, computer synthesis of properties and reactions of virtual reality is carried out in real time.

Big Data is a description of the handling of structured and unstructured data of significant volume and diversity that undergo effective processing by software tools. The key principles of working with big data include:

– Horizontal scalability: this is the fundamental principle of processing big data, involving the necessity to increase the number of computing nodes that process the data without compromising performance.

– Fault tolerance: given the potentially large number of computing nodes in a cluster (up to dozens of thousands) and their dynamic growth, the probability of nodes failing increases. Therefore, methods for working with large amounts of data must take such situations into account and implement preventive measures.

– Data locality: since data is distributed across numerous computing nodes if the data physically resides on one server and is processed on another, the costs of data transfer can be unreasonably high. Therefore, storing and processing data should be performed on the same computer.

Cloud Computing is a model that enables ubiquitous and convenient on-demand access to a shared pool of configurable computing resources (such as networks, servers, storage, applications and services) that can be rapidly provisioned by a service provider.

Artificial Intelligence (AI) is a direction of information technology whose task is to replicate intelligent reasoning and actions through computational systems and other artificial devices. It is also the ability of a system to correctly interpret external data, learn from such data and use the acquired knowledge to achieve specific goals and tasks through flexible adaptation.

Machine Learning (ML) is a branch of artificial intelligence theory that explores methods and tools for constructing algorithms capable of learning.

Digital Twin is a software counterpart of a physical device that implements the complete cycle of modelling the internal processes, technical characteristics and behaviour of real objects (processes, phenomena) under the influence of the

environment and destabilising factors. A characteristic feature of a digital twin is that its inputs are measurement data from sensors of a real device that operates in parallel. The operation is possible in both online and offline modes. This approach can be used to compare information from virtual sensors of the digital twin with sensors of real devices, detect anomalies and identify the causes of their occurrence.

Blockchain is a continuously linked chain of serial blocks containing information and built according to specific rules. This technology can be applied to any object that can be represented as interconnected informational blocks.

An essential and sustainable direction in the development of the Internet of Things (IoT) and Industry 4.0 concepts is the Industrial Internet of Things (IIoT). It represents a system of interconnected computer networks and associated industrial objects and production processes with integrated sensors and applied software for data aggregation, network exchange and transformation. It enables remote monitoring and control in an automated mode. The graphical interpretation of the structural and functional organisation of IIoT networks is shown in Fig. 1.8 and a comparative characteristic of the main application areas of IIoT and IoT systems and networks is shown in Fig. 1.9.

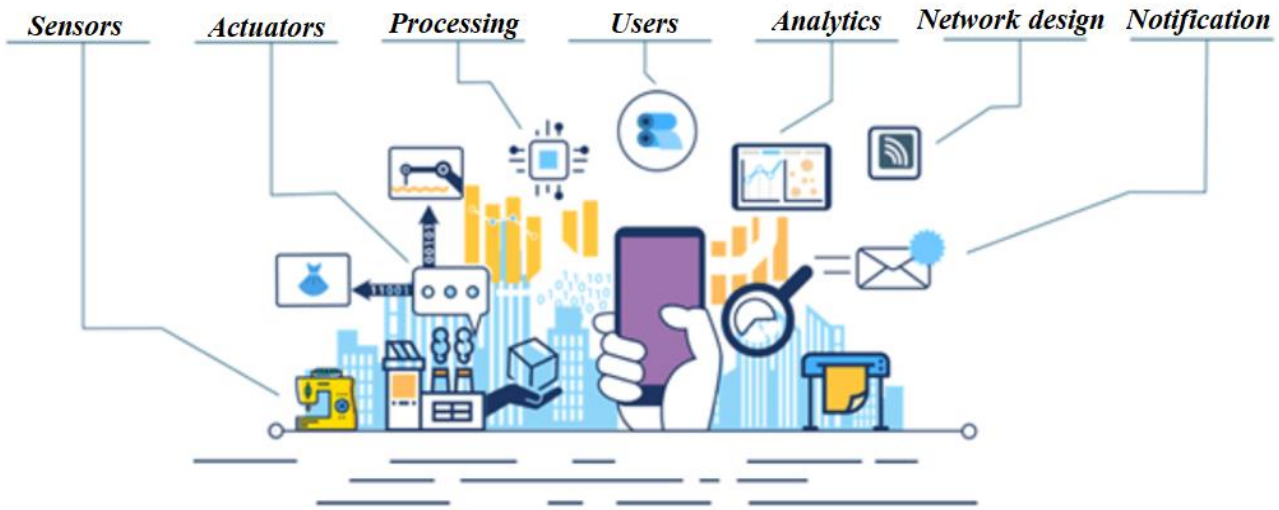


Fig. 1.8. Structural and functional organisation of IIoT systems and networks

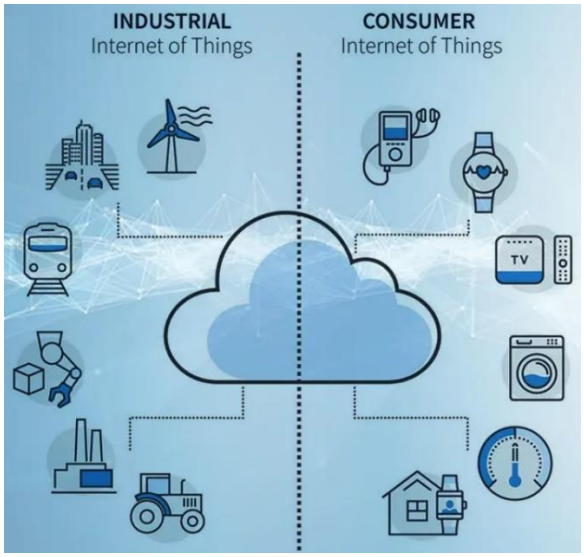


Fig. 1.9. Application areas of IIoT and IoT systems and networks

Sustainable development of IIoT, according to Craig Resnick, a leading analyst at ARC Advisory Group, is currently characterised by six main trends and technologies, as shown in Fig. 1.10.

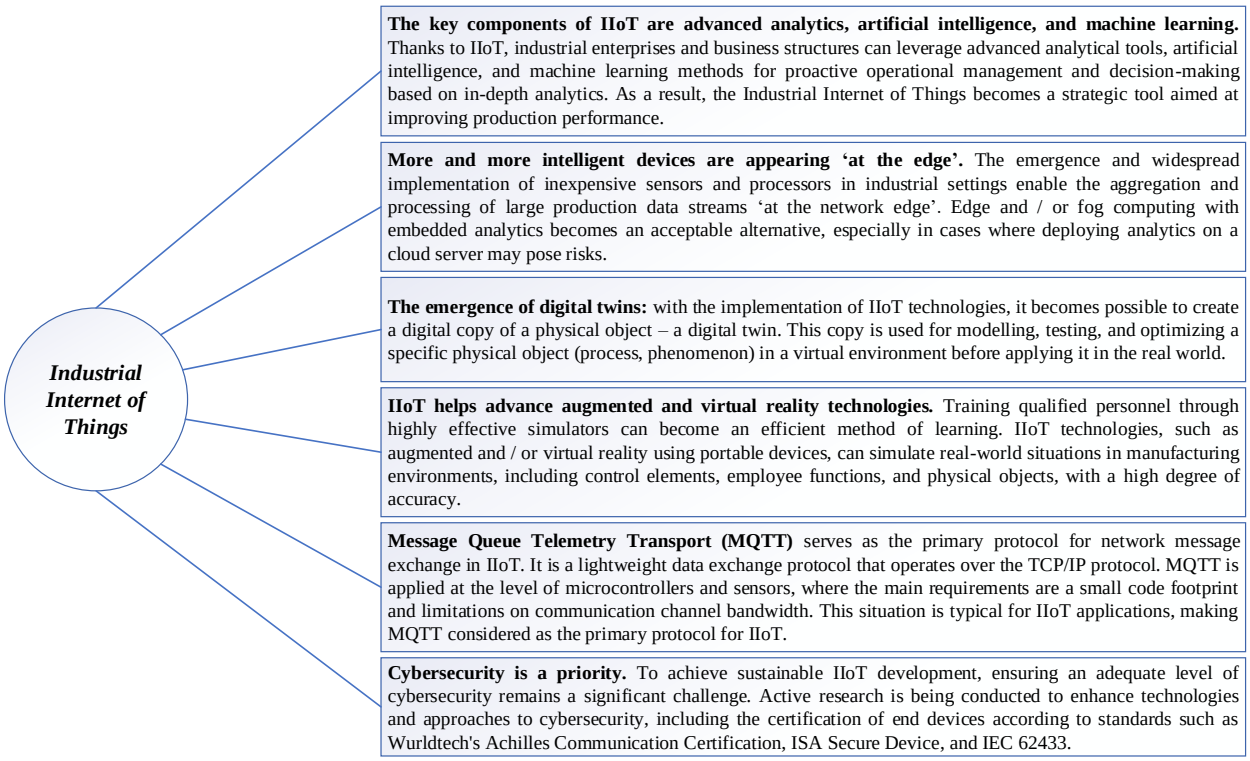


Fig. 1.10. Sustainable development trends in IIoT technologies

Thus, the Industrial IoT is an effective tool for creating new and upgrading

existing productions, making them more adaptive, efficient, resilient to destabilising factors and attractive for investments. The implementation of IIoT hardware and software solutions as components of digital ecosystems enables the transformation of industrial enterprises from isolated systems that independently implement all necessary production and business processes into open systems that bring together all participants in the relevant spheres of production and service provision.

Self-assessment Questions on Material Comprehension

1. Provide an overview of the key stages in the development of the Internet of Things as a conceptual direction in digital transformation.

2. Define the following terms: IoT device, smart technology, operational technology, cloud technologies, fog technologies, physical device, end device, IoT centre, IoT device template, physical level of IoT, IoT application level and IoT analytics factor.

3. Explain the meaning of the Internet of Things formula.

4. Provide a detailed description of the hardware and software implementation of the Internet of Things formula.

5. Characterise the key application areas of Internet of Things systems and networks.

6. Provide a detailed overview of current trends in the development of IoT technologies.

7. Provide a detailed overview of the key technologies of digital transformation in industry and business related to IoT within the Industry 4.0 concept.

8. Describe the main functional features and application areas of the Industrial Internet of Things (IIoT) technology.

List of Recommended Literature for the First Chapter

1. Huawei: The history of the emergence of the Internet of Things and prospects

for development. Available at: https://e.huawei.com/kz/ict-insights (accessed on December 21, 2023).

2. Perenio: History of the Internet of Things. Available at: https://perenio.com/blog/the-history-of-the-internet-of-things (accessed on December 23, 2023).

3. Vovna O.V., Laktionov I.S., Lebediev V.A. Computer-integrated monitoring and control in industrial greenhouses: current results and research prospects: monograph. Pokrovsk: SHEI 'DonNTU', 2020. 255 p. (in Ukrainian).

4. TI40: Industry 4.0 technologies. Lectures (by Oleksandr Pupena). Available at: https://pupenasan.github.io/TI40/%D0%9B%D0%B5%D0%BA%D1%86/intro.html (accessed on November 25, 2023).

5. IT Enterprise: Internet of Things. Available at: it.ua/knowledge-base/technology-innovation/internet-veschej-internet-of-things-iot (accessed on November 27, 2023).

6. Microsoft: Glossary of IoT terms. Available at: https://learn.microsoft.com/uk-ua/azure/iot/iot-glossary (accessed on November 30, 2023).

7. Zhurakovsky B.Yu., Zeniv I.O. Internet of things technologies. Study guide [Electronic resource]: education manual for students speciality 126 'Information systems and technologies', specialisation 'Information support of robotic systems'. Kyiv: KPI named after Igor Sikorskyi, 2021. 271 p. (in Ukrainian).

8. Tibco: What is Industrial Internet of Things (IIoT). URL: https://www.tibco.com/glossary/what-is-the-internet-of-things-iot (accessed on October 23, 2023).

9. Moco Smart: Difference between IIoT and IoT. Available at: https://www.mokosmart.com/uk/iiot-vs-iot-technologies/ (accessed on December 15, 2023).

10. Bravos G., Cabrera A.J., Correa C., Danilovic D., Evangeliou N., et al. Cybersecurity for Industrial Internet of Things: Architecture, Models and Lessons Learned. *IEEE Access*. 2022. Vol. 10. P. 124747–124765

CHAPTER 2
STRUCTURAL AND ALGORITHMIC ORGANISATION OF INTERNET OF THINGS SYSTEMS AND NETWORKS

2.1 Generalised Architecture of Internet of Things Systems and Networks

At present, applied hardware and software solutions of the Internet of Things (IoT) systems and networks are dynamically implemented and used in various industries, businesses, social spheres and others, as detailed in subsection 1.4 ‘Key Areas of Application of the Internet of Things’. This sustainable development of IoT technologies has allowed for the formation of generalised approaches to synthesising the architecture of IoT systems and networks, aiming to facilitate data aggregation and exchange between things at any time and any place, utilising a wide range of physical devices, network technologies and software services, as shown in Fig. 2.1.

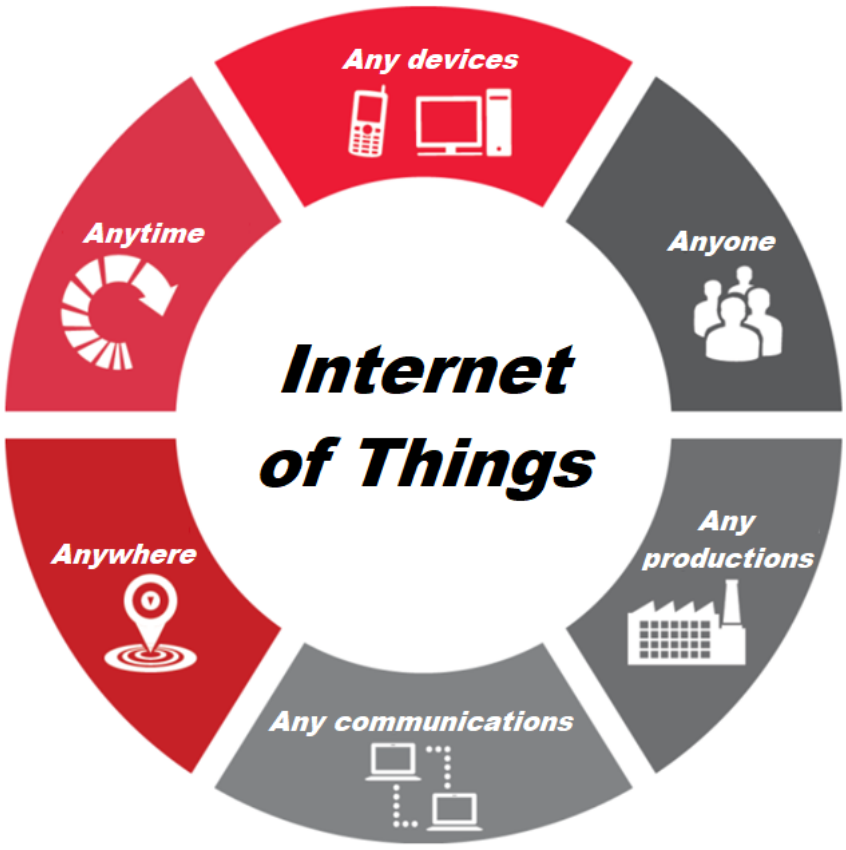


Fig. 2.1. Visualisation of the key conceptual foundations of the Internet of Things

Applied hardware and software solutions that implement the conceptual foundations of the Internet of Things (IoT) vary depending on their intended purpose but generally correlate with a typical architecture that realises the full cycle of intelligent transformation of measurement data (see Fig. 2.2): collection – local processing – network exchange – aggregation – in-depth intelligent analysis – formation of a database and knowledge – ergonomic visualisation and user notifications. Creation of the IoT systems and networks based on such architecture enables a bidirectional flow of data / information: from things to end users and vice versa.

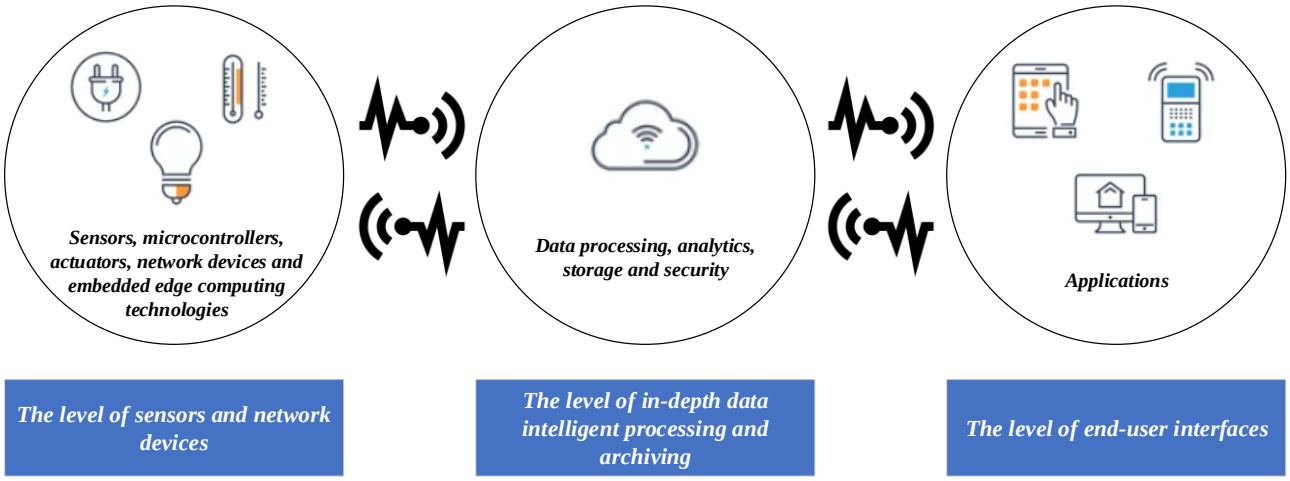


Fig. 2.2. Generalised architecture of the IoT systems and networks

The level of sensors and network devices. Hardware and software means for detection, precision measurement monitoring, embedded processing, network connection and control signal generation implement the systemic organisation of the level of sensors and network devices in IoT systems.

The level of in-depth data intelligent processing and archiving. Data transmitted or received from the level of sensors and network devices undergo intelligent processing using cloud and / or fog computing technologies, then stored as big data.

The level of end-user interfaces. In the final stage of information transformation, data from IoT devices are utilised in applications to assist individuals or organisations in obtaining real-time information and making better decisions or performing specific

actions. Such applications send processed information from cloud services to users' personal computers or mobile devices, where this information is visualised in a user-friendly format. The application level is crucial for users as it serves as their interface to the IoT network, allowing them to monitor and control IoT system components, including in real time.

In a more detailed view, the architecture of IoT hardware and software solutions, developed by researchers Keyur K. Patel and Sunil M. Patel, is shown in Fig. 2.3. This architecture demonstrates the interaction of hierarchical levels during various deployment scenarios and operation of Internet of Things systems.

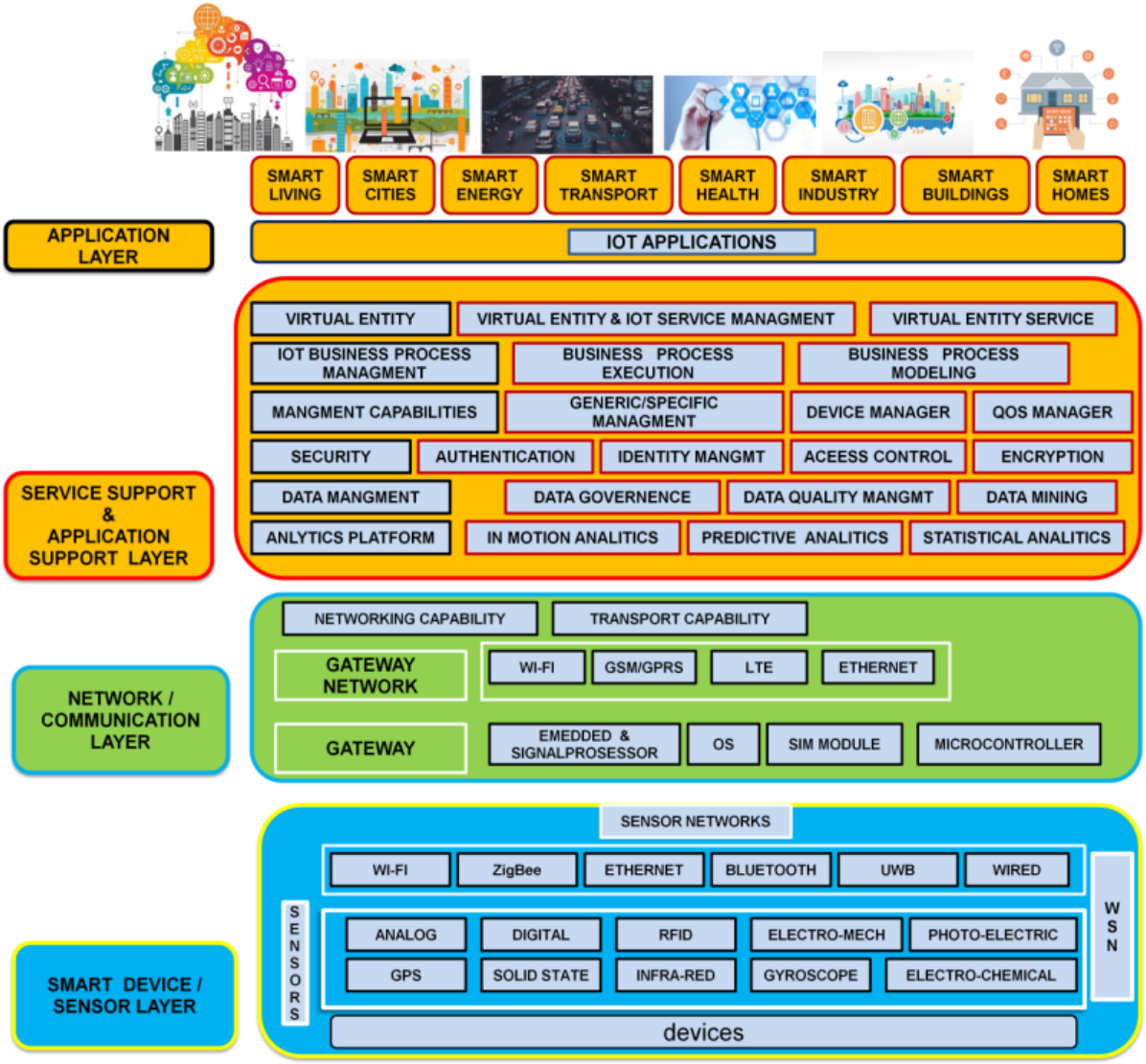


Fig. 2.3. Detailed architecture of IoT systems

Among the fundamental technical and functional characteristics of such IoT hardware and software solutions are:

Measurement, monitoring and control: sensor devices used in IoT technologies detect and monitor any changes in the surrounding environment, as well as generate measurement data regarding their status. Therefore, IoT technologies transform passive networks into active ones.

Interconnection: any things (objects, processes, phenomena) must have the ability to be interconnected with the global information and communication infrastructure. Network connectivity involves establishing a stable connection between all things and the IoT platform deployed on a local or cloud server. After establishing the connection between IoT devices, high-speed message exchange between the devices and the cloud is essential to ensure reliable, secure and bidirectional data and / or information exchange.

Data analysis and artificial intelligence: after connecting all relevant devices and transmitting information, real-time analysis of measurement data occurs, utilising them to create effective analytical models. The term either ‘smart’ or ‘intelligent’ can be used for such systems.

Active integration: IoT systems and networks integrate various technical means and models to improve interaction with end-users, enabling active interaction between connected technologies, products or services.

End-device control: it is crucial to have the ability for automatic and reliable remote control of objects (processes) within IoT networks, as otherwise, it can lead to complete failure and malfunction of the entire system.

Heterogeneity: devices within IoT systems and networks are heterogeneous as they are based on different hardware platforms and network protocols. However, they must interact with other devices or service platforms over different network technologies.

Dynamic reconfiguration: the state of IoT devices within the network structure changes dynamically, such as active and passive operating modes, connection and / or disconnection of nodes, as well as the location and speed of device movement.

Data security: since IoT technologies are networked hardware and software solutions, it is essential to consider the need for ensuring data cybersecurity, including the security of personal data, endpoint protection and safeguarding data in transit between them. All of this necessitates the creation of a scalable security paradigm.

In order to implement the architectural solutions of IoT systems and networks and to ensure the fulfilment of their fundamental functions described above, a wide range of highly efficient achievements in sensor, infocommunication, microprocessor, software and web technologies can be utilised, as shown in Fig. 2.4.

Fig. 2.4. Technologies used for constructing IoT systems and networks

Therefore, it can be observed that the practical aspects of creating and using Internet of Things (IoT) systems and networks entail certain characteristic features in the structural and algorithmic organisation of specific hardware and software solutions

for IoT. However, the key conceptual principles are similar, namely: remote devices send and receive processed data about objects (processes, phenomena) equipped with sensor and microprocessor technologies, integrated into a network organisation through infocommunication and cloud technologies. Thus, IoT systems represent multi-level networked software and technical structures that perform the full spectrum of procedures involving data collection, processing, transmission and storage, as well as remote control of real objects and processes.

2.2 The Reference Model of the Internet of Things Y.2060

The creation and development of IoT technologies are currently undertaken not only by leading device manufacturers but also by specialised organisations, including the International Telecommunication Union (ITU), the Industrial Internet Consortium (IIC) and the Internet Engineering Task Force (IETF).

In the Y.2060 recommendations, developed and proposed for use by the International Telecommunication Union (ITU), titled ‘Overview of the Internet of Things’, the Internet of Things (IoT) is defined as a global infrastructure that provides comprehensive information and communication services by connecting physical and virtual things based on existing and evolving compatible infocommunication technologies. In this formulation, the term ‘thing’ in the Internet of Things refers to an object in the physical and / or virtual world that can be identified and connected to the network. A device in the context of the Internet of Things is referred to as a hardware and software component that has mandatory communication capabilities and can perform measurements. Such a component can trigger under certain conditions, detect, input, process and store measurement data / information messages. The structural model of the interaction between real and virtual objects in the context of the Internet of Things concept is shown in Fig. 2.5 and the relationship between different types of devices and real things in IoT technologies is shown in Fig. 2.6.

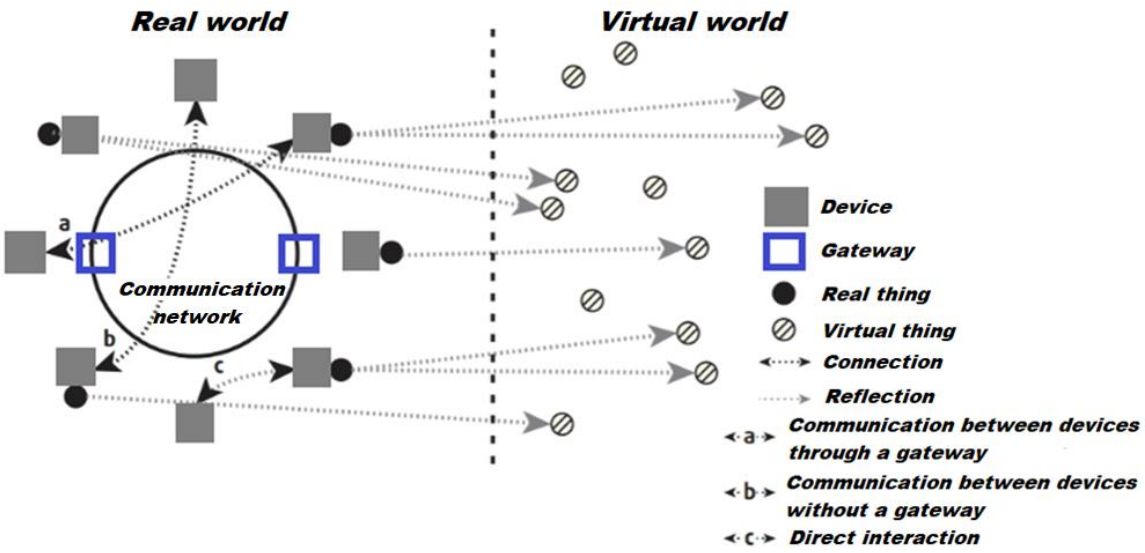


Fig. 2.5. Model of interaction of real and virtual IoT objects

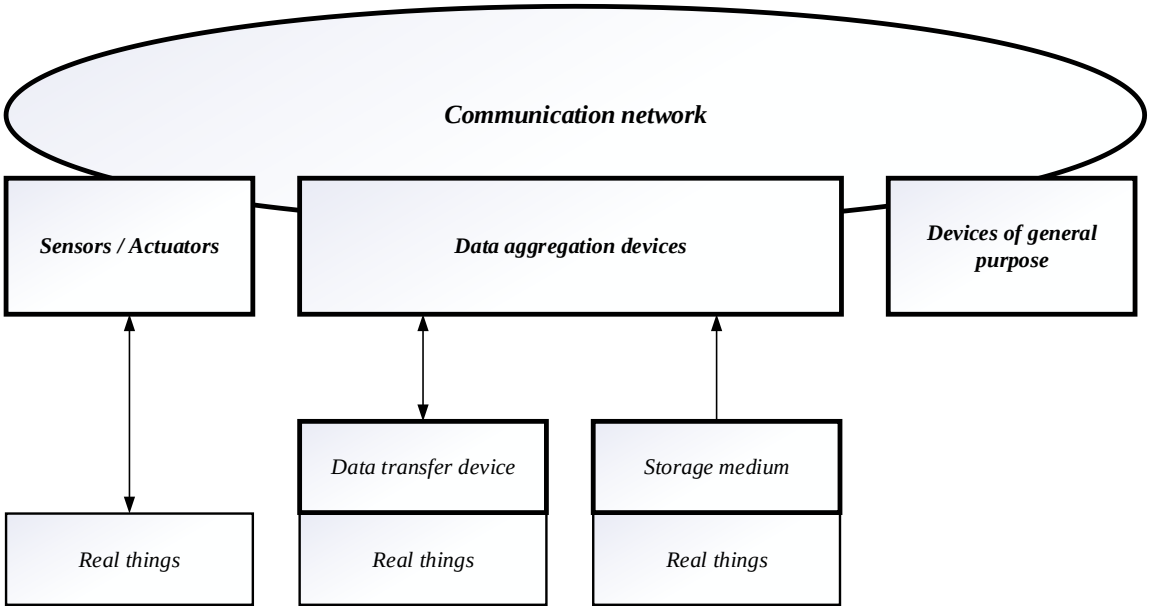


Fig. 2.6. Structural model of the relationship between different types of devices and real things in IoT technologies

In accordance with the recommendations of the Telecommunication Standardisation Sector of the International Telecommunication Union (ITU-T), the Internet of Things (IoT) is a network organisation of hardware and software devices that are closely related to things. Sensor devices and actuators interact with real things in the surrounding environment. Data aggregation (collection) devices read information from real things and / or record it on real things, as well as directly interact

with storage mediums or devices for transferring measurement data.

Therefore, according to the logic of the ITU-T, IoT represents a systemic organisation of either real or virtual, as well as both real and virtual objects, sensors, microcontrollers, actuators and the Internet. In this ITU-T model, a real instance of an IoT component is an object characterised by the following properties:

– intelligent: has a microcontroller (microcomputer) and software for data processing and generating control signals;

– capable of informing or acting: includes a sensor for measuring any physical parameters and an actuator whose operation can be controlled;

– network-enabled for data exchange with other IoT objects.

The reference model of the Internet of Things which has been designed by ITU-T, consists of four main hierarchical levels (device level, network level, service and application support level, as well as application level), along with control and security capabilities that operate across these four main levels of the Y.2060 reference model, is shown in Fig. 2.7.

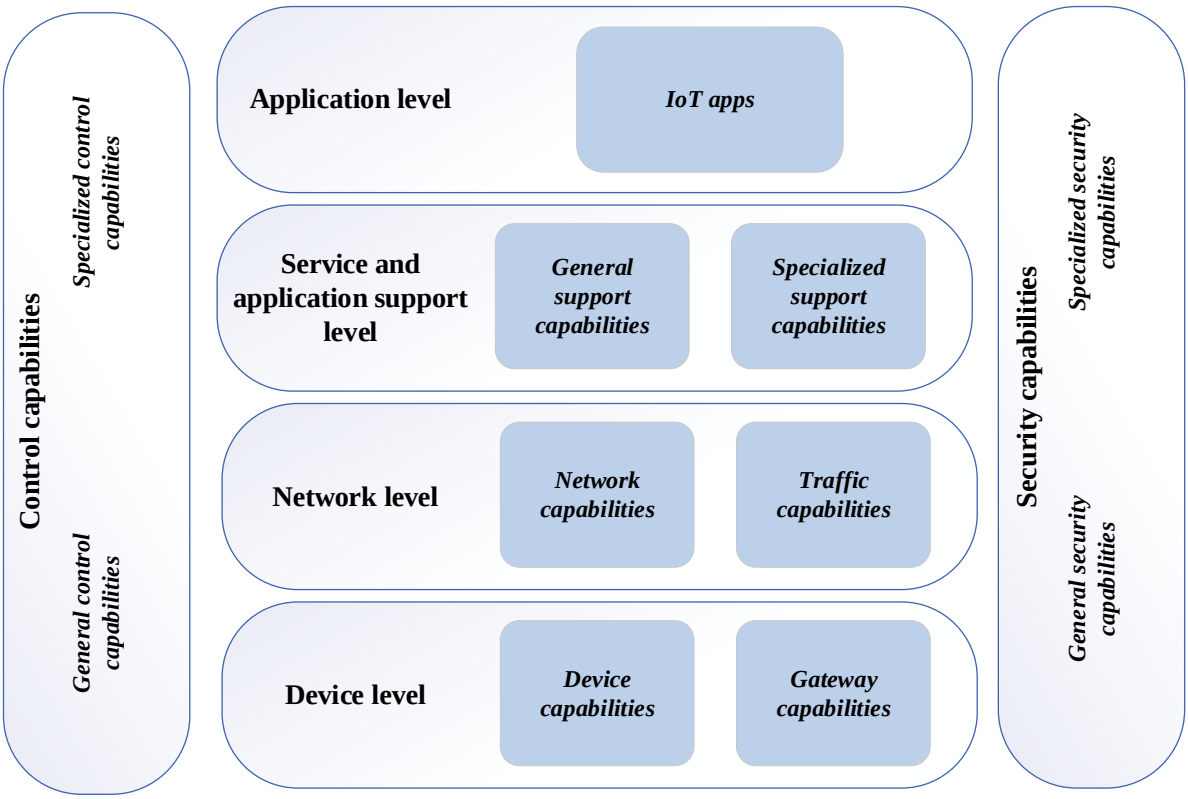


Fig. 2.7. The reference model of the Internet of Things Y.2060

An exemplary model of the Internet of Things has been introduced and comprehensively described in the recommendations Y.2060 by the International Telecommunication Union. This model serves as a structural and algorithmic foundation for standardising IoT systems and networks. Such a reference architecture provides developers with an understanding of the functions that should be implemented in IoT technologies and how they should interact.

The device level: drawing an analogy between the IoT reference model Y.2060 and the well-known OSI network model, the device level (Y.2060) corresponds to the physical and data link levels (OSI). The capabilities of this level, based on logical purposes, can be divided into two types of capabilities:

– Device capabilities that include, but are not limited to, the following: direct interaction with the communication network: devices implement the capability to collect and upload information directly to the communication network without using gateway capabilities, they can also directly receive information from the communication network (e.g., control commands or synchronisation); indirect interaction with the communication network: devices implement the capability to collect and upload information to the communication network indirectly (using gateway capabilities), devices can also receive information from the communication network indirectly; implementation of self-configuring dynamic networks: in the implementation of certain operational scenarios that require increased scalability and rapid deployment, devices can organise networks in specialised ways (e.g., mesh or ad-hoc); sleep and wake modes: device capabilities may support sleep and wake modes for power saving purposes.

– Gateway capabilities, which include but are not limited to the following: support for multiple interfaces: at the device level, gateway capabilities support devices connected using various wired or wireless technologies (CAN, ZigBee, Wi-Fi, Bluetooth, etc.), as well as at the network level, gateway capabilities define data exchange approaches using different technologies (PSTN, 3G, LTE, DSL, Ethernet, etc.); protocol transformation: there are two typical situations where gateway capabilities are needed, the first situation occurs when communication at the device

level involves different protocols, the second situation arises when communication at both the device and network levels may use different types of network protocols.

The network level: drawing an analogy between the IoT reference model Y.2060 and the well-known OSI network model, the network level (Y.2060) corresponds to the network and transport levels (OSI). The capabilities of this level can logically be divided into two types:

– Network capabilities: these capabilities provide relevant connection monitoring and control functions (authentication, authorisation, accounting, as well as monitoring and control of access, transport resources and mobility).

– Transport capabilities: focused on ensuring connectivity during the transport of IoT services and program-specific data, as well as transporting information related to monitoring and control procedures.

The level of service and application support provides the following capabilities utilised by IoT applications:

– General support capabilities designed for executing various IoT software, such as data collection, processing and / or storage.

– Specialised support capabilities designed to meet diverse software requirements, comprising various detailed subgroups of capabilities to perform different support functions for various IoT software.

The application level consists of all applications that facilitate the interaction of IoT devices, as well as the execution of procedures for processing, storing and visualising measurement information.

The level of control capabilities is designed to perform control functions of the network organisation, including fault detection and resolution, configuration, monitoring of operational indicators and ensuring security. Typical tasks of general control capabilities are outlined in the IoT reference model Y.2060, as shown in Fig. 2.8. Specialised control capabilities are directly related to application requirements, such as reliability and real-time monitoring requirements for Smart Grid power lines.

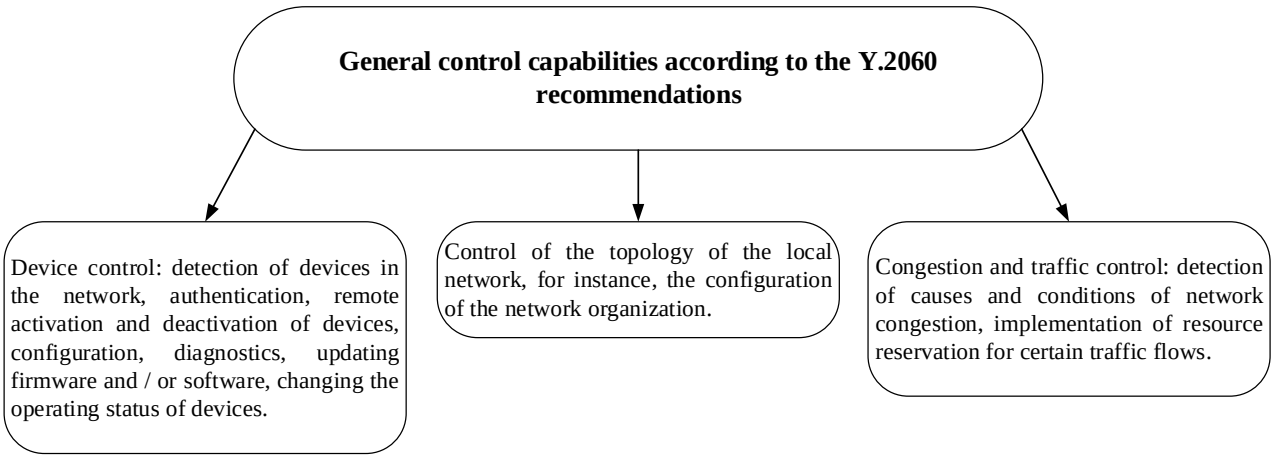


Fig. 2.8. Detailing of the Y.2060 general control capability sublevel

The level of security capabilities includes general security capabilities that are independent of applications and specialised security capabilities that are directly defined by application requirements, such as the requirements of cybersecurity mechanisms for mobile payment systems. The typical tasks of general security capabilities are outlined in the IoT reference model Y.2060, as shown in Fig. 2.9.

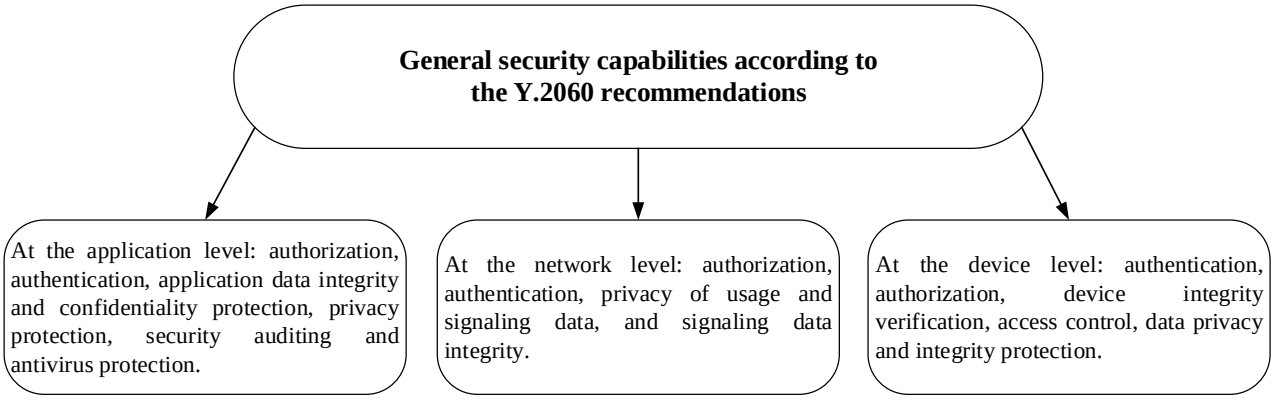


Fig. 2.9. Detailing of the Y.2060 general security capability sublevel

Moreover, in the Y.2060 recommendations, a series of possible business models for organising and ensuring the sustainable operation of IoT systems and networks have also been identified, as outlined below.

In the first model, Subject A ensures the operation of devices, networks, platforms and applications, as well as directly services application users, as shown in Fig. 2.10. In real practical terms, mobile operators or large vertically integrated

enterprises are able to act as Subject A.

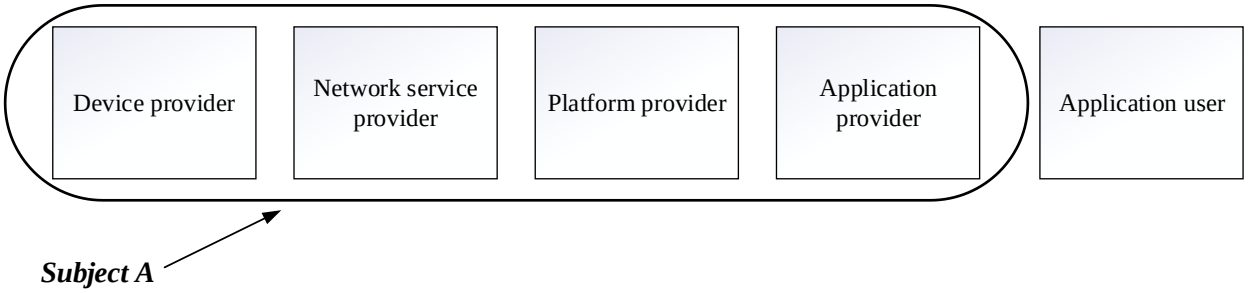


Fig. 2.10. The first version of the model of the organisation and functioning of IoT systems and networks

In the second model, Subject A ensures the operation of devices, networks and platforms, while Subject B ensures the operation of applications and provides direct services to application users, as shown in Fig. 2.11. In real practical terms, mobile operators or other service providers in the field of information and communication technologies are able to act as Subject A, while Subject B could be other entities providing application-related services.

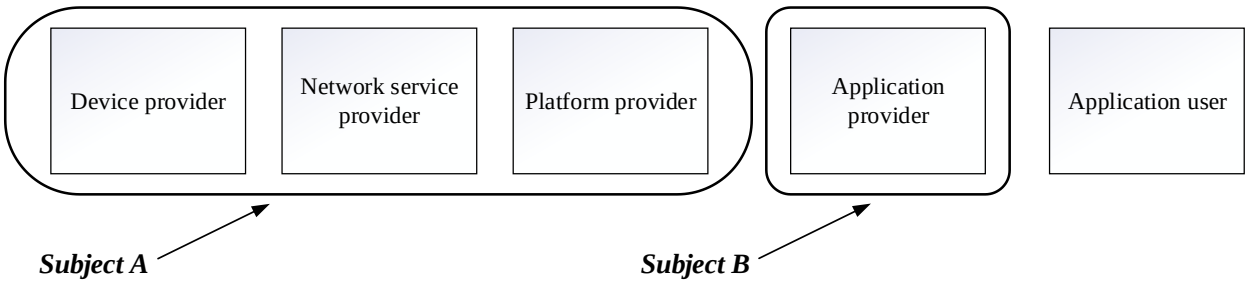


Fig. 2.11. The second version of the model of the organisation and functioning of IoT systems and networks

In the third model, Subject A ensures the operation of the network and platform, while Subject B ensures the operation of devices and applications and also directly services application users, as shown in Fig. 2.12. In real practical terms, mobile operators or other service providers in the field of information and communication technologies are able to act as Subject A, while Subject B could be other entities

providing device and application-related services.

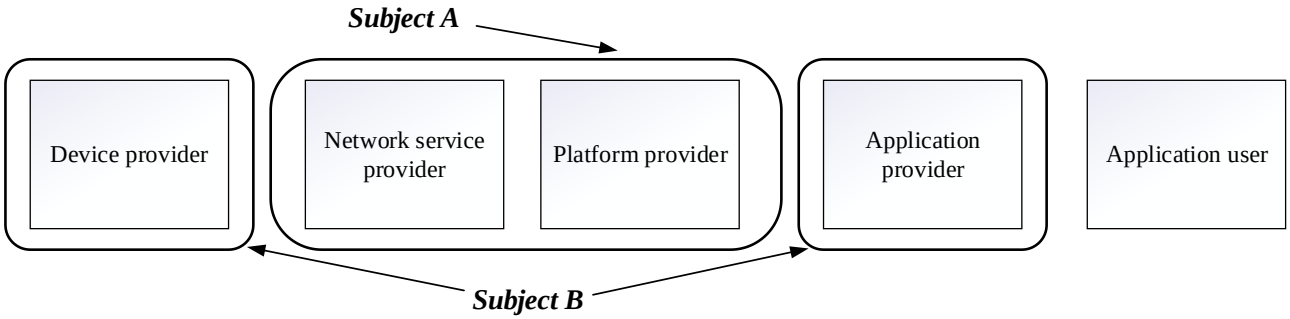


Fig. 2.12. The third version of the model of the organisation and functioning of IoT systems and networks

In the fourth model, Subject A ensures the operation of the network, while Subject B ensures the operation of devices, platforms and applications and also directly services application users, as shown in Fig. 2.13. In real practical terms, mobile operators or other service providers in the field of information and communication technologies, or vertically integrated companies, are able to act as Subject A, while Subject B could be other entities providing device, platform and application-related services.

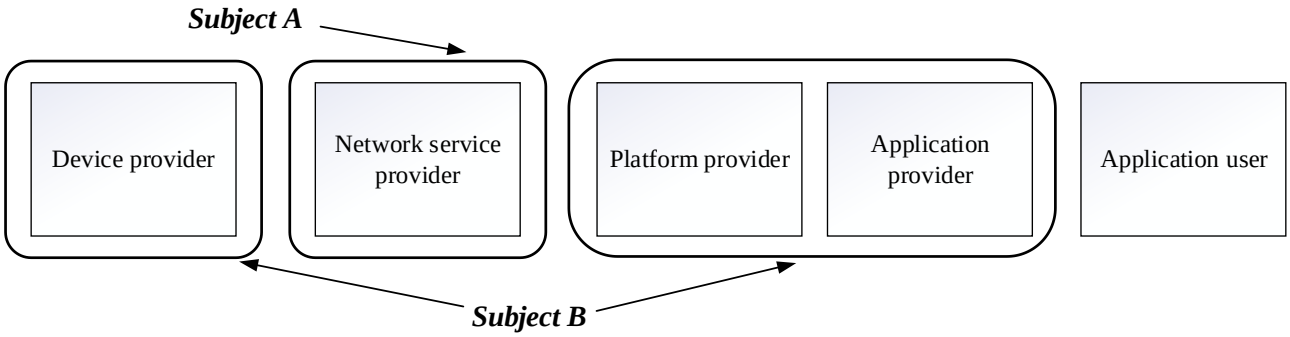


Fig. 2.13. The fourth version of the model of the organisation and functioning of IoT systems and networks

In the fifth model, Subject A ensures the operation of the network, Subject B ensures the operation of the platform and Subject C ensures the operation of devices

and applications, as well as directly services application users, as shown in Fig. 2.14. In real practical terms, mobile operators are able to act as Subject A, other service providers in the field of information and communication technologies are able to act as Subject B and vertically integrated companies are able to act as Subject C.

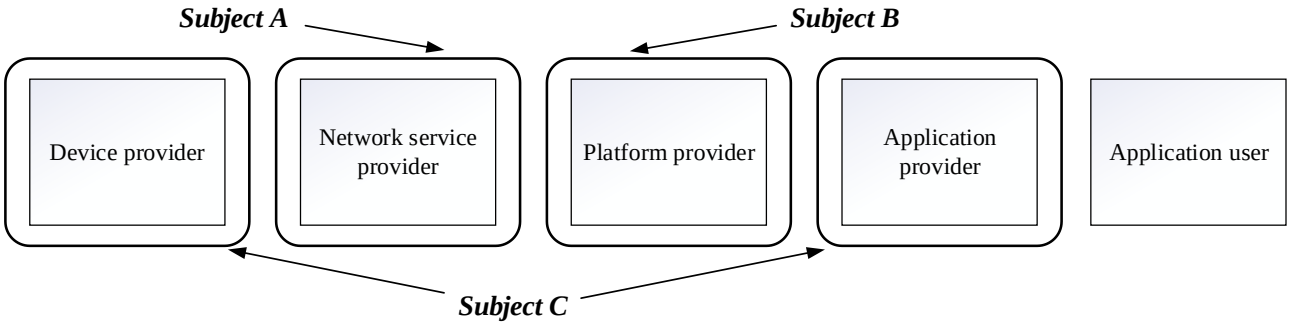


Fig. 2.13. The fifth version of the model of the organisation and functioning of IoT systems and networks

Therefore, the structural diagrams of business models presented in Figs. 2.10–2.14 allow for the practical implementation of effective deployment and sustainable operation of IoT networks across various scales and application domains.

2.3 The Reference Model of the Internet of Things by IWF

The Committee on Information and Digital Technology Architecture of the Internet of Things World Forum (IWF), which includes industry leaders such as Intel, Cisco and IBM, outlined a reasoned model of IoT systems and networks in October 2014. This model serves as a comprehensive structural and algorithmic solution aimed at improving the efficiency and speed of deploying IoT hardware and software solutions. The main objective of this model is to substantiate ways to create reproducible, standardised and unified technical solutions for IoT systems and networks of various applications. This reference model is a logical complement to the Y.2060 model developed by the International Telecommunication Union (ITU-T).

The main difference between these models (IWF and Y.2060) lies in the fact that the Y.2060 recommendations focus primarily on the lower levels of devices and gateways, placing the greatest emphasis on the substantiation of the concept of supporting the development and implementation of standards for the interaction of IoT devices. In the model proposed by IWF, the main attention is paid to the development of applications and software for IoT technologies, as well as the functions and tools to support the corporate Internet of Things. This model, unlike the Y.2060, consists of seven main levels, as shown in Fig. 2.15.

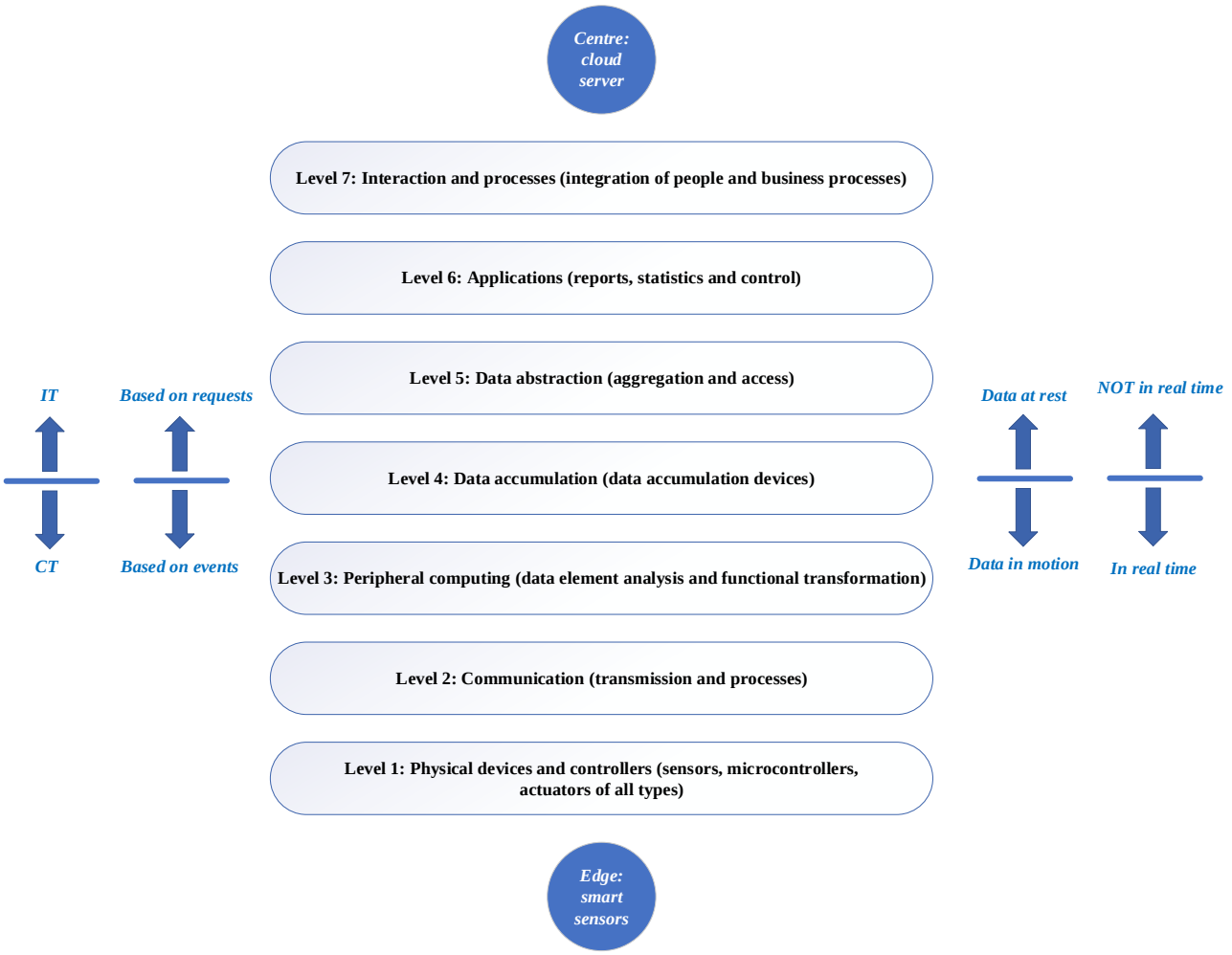


Fig. 2.15. The Reference Model of the Internet of Things by IWF

The detailed description and analysis of the IWF model, which has been published by Cisco, reveal its key features, which include the following:

– simplification: the model helps break down comprehensive systems and

networks into functional components to enhance understanding of the IoT operating principles;

– clarification: the model provides additional information for more precise identification of levels in IoT systems and networks, contributing to the development of commonly used terminology;

– identification: the model aligns aspects where various types of data processing are optimised in different structural components of the system;

– standardisation: the model represents the first step toward enabling developers and information technology providers to create interoperable IoT products;

– organisation: the model makes technical solutions for IoT systems and networks tangible and accessible, moving beyond abstract concepts to practical implementation.

Level 1: Physical Devices and Controllers. In the IWF model, this level corresponds to the term ‘things’ within the Internet of Things (IoT) concept. From the perspective of designing IoT systems, the structural elements at this level are not the real things themselves but rather hardware devices that directly interact with real entities (sensors and actuators).

Among the additional functional capabilities of devices at this level are analogue-to-digital and digital-to-analogue conversion, generation of measurement data and remote monitoring and control. The integration of conceptual principles from edge intelligence theory at the level of physical devices and controllers enables data exchange between devices at this level and devices involved in low-level processing of measurement data (Level 3).

The hardware and software foundation of this level also includes network devices (routers, gateways and firewalls) used in the design of information and computer networks, as well as for connecting to the Internet.

This level implements the capability of network communication between devices. Additionally, through higher logical levels, it enables the exchange of measurement data with application platforms such as personal computers, mobile devices, dispatch control panels and others.

Level 2: Communication. This level encompasses a range of procedures designed to implement the transportation of data from edge node devices to the cloud server. It also involves tasks related to mapping field data onto logical and physical technologies used in IoT systems and networks, as well as feedback from cloud servers to the local network. During the implementation of technical tasks related to communication organisation, a significant variety of options for creating both wired and wireless networks can be utilised.

Level 3: Peripheral Computing. In many hardware and software solutions implemented in IoT systems, the distributed network of sensors is capable of generating large volumes of measurement data.

Therefore, it is practical to perform as much of the transformation and processing of measurement data at the field level of intelligent sensors and microcontrollers rather than constantly involving centralised servers for these tasks that are accessible to IoT applications.

Thus, the main task of the peripheral (edge, fog or distributed) computing level is to transform network streams of measurement data into information that is functionally suitable for transformation and storage at higher hierarchical levels of the Internet of Things systems and networks. The primary, although not exhaustive, range of operations at the peripheral computing level, as highlighted by Cisco, is shown in Fig. 2.16.

Elements for processing measurement data at this structural level are typically deployed at the edge of the IoT network in the form of embedded specialised software.

Thus, a certain part of the basic processing of significant data volumes generated by IoT network sensors is shifted from IoT applications deployed in the centre of the network to peripheral devices at the edge of the network.

The main types of such distributed computing today include fog and edge technologies, which are detailed in the fifth section ‘Data Aggregation and Processing Technologies’ of this textbook.

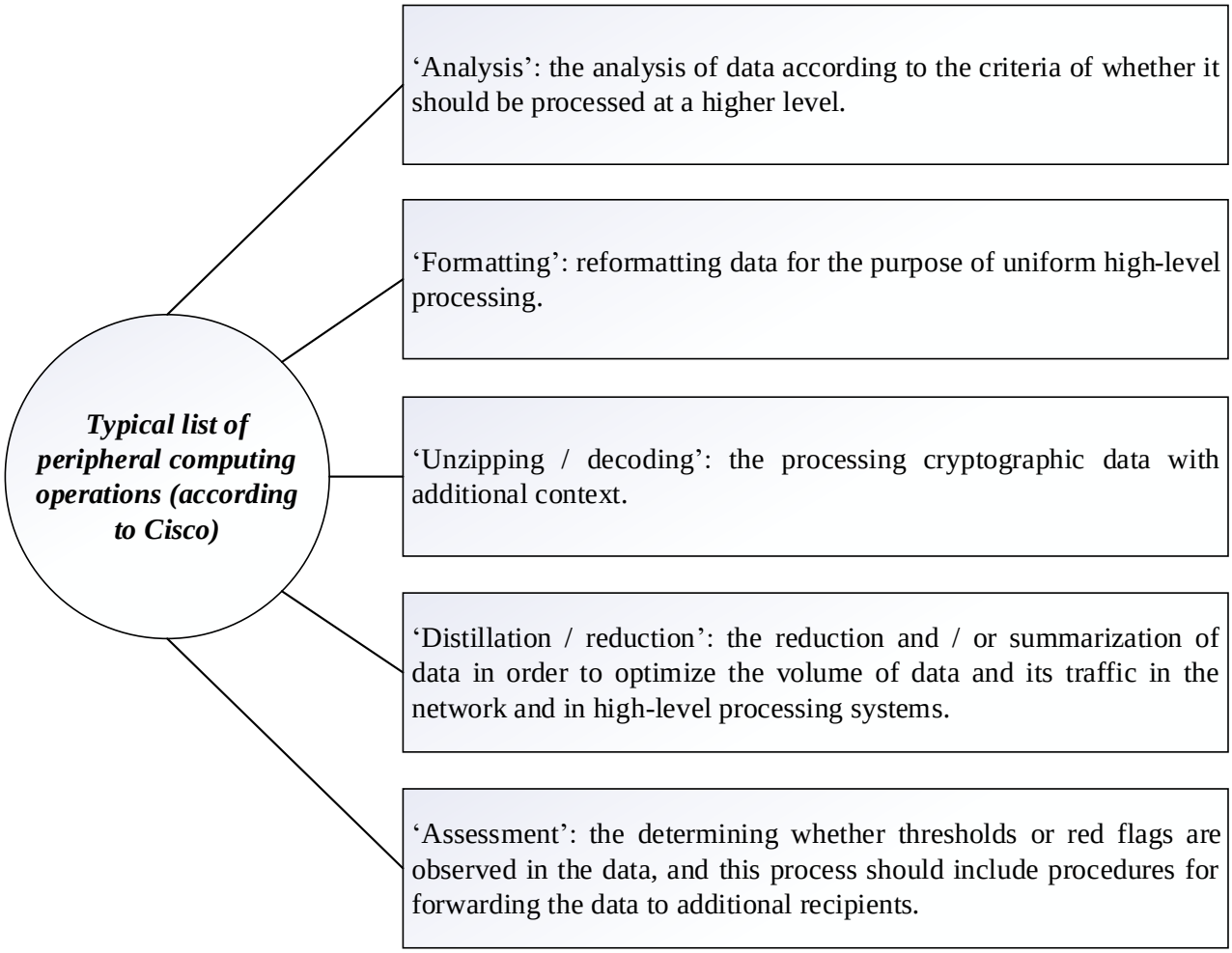


Fig. 2.16. The list of typical operations at the level of peripheral computing (according to Cisco recommendations)

Level 4: Data Accumulation. At this level, data coming from distributed devices, after implementing filtering and processing procedures using peripheral computing technologies, is stored in a repository, from where it is made accessible for higher-level processing.

The principles of data processing at this level fundamentally differ from those used at the low-level (edge) and high-level (cloud) data processing levels in architectural and functional organisation. This is mainly because of the fact that data passing through the IoT network is called ‘data in motion’. Approaches to organising and operating on data in motion are determined by the technical and functional characteristics of the devices generating this data. Data generation mechanisms are implemented based on events or with a certain periodicity.

For an effective aggregation of this data and its further processing, it is necessary to respond to its appearance in a real-time regime. On the other hand, most applications do not require processing data at the speed of network exchange. In real-world practical applications, cloud services or other application platforms are not able to aggregate and process in real-time the significant volumes of data generated by a vast number of distributed networked IoT devices.

Therefore, the functioning of applications relies on ‘data at rest’, namely, data that accumulates in a specialised easily accessible information repository. Thus, the high levels (5–7) of IoT systems and networks operate on queries, while the three lower levels (1–3) operate on events. Basic operations that need to be implemented at the data accumulation level are shown in Fig. 2.17. This list is not exhaustive and can be extended according to the requirements of specific IoT systems and network application solutions.

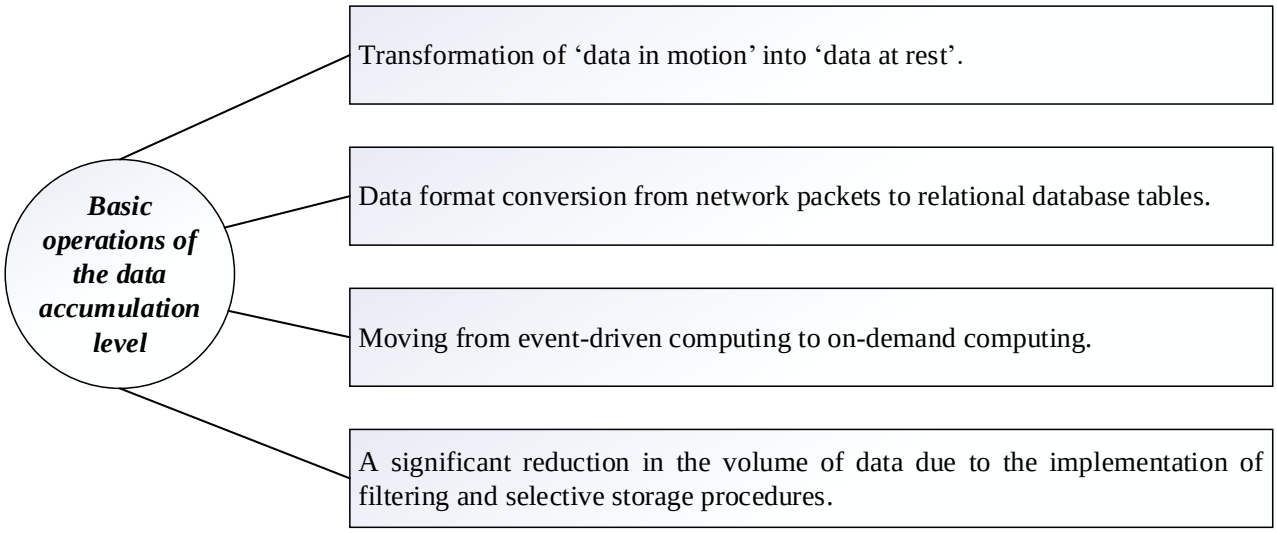


Fig. 2.17. Basic operations of the data accumulation level

Another conceptual characteristic of the data accumulation level is that it represents a certain boundary between information technology (IT) and computing technology (CT). In this context, information technology includes software and hardware, network devices and related services. Computing technology, within the presented architecture of IoT, encompasses software and hardware designed for

detecting or triggering changes through direct monitoring and / or measurement control of objects, processes and phenomena.

Level 5: Data Abstraction. The purpose of this hierarchical level in IoT is to aggregate and format data according to the requirements of the application level, receiving data from the level of peripheral computations to the data accumulation level in various formats.

A list of typical functional operations that need to be implemented at the data abstraction level is shown in Fig. 2.18. Similar to the data accumulation level, this list is not exhaustive and can be extended based on the requirements of specific applications in IoT systems and networks.

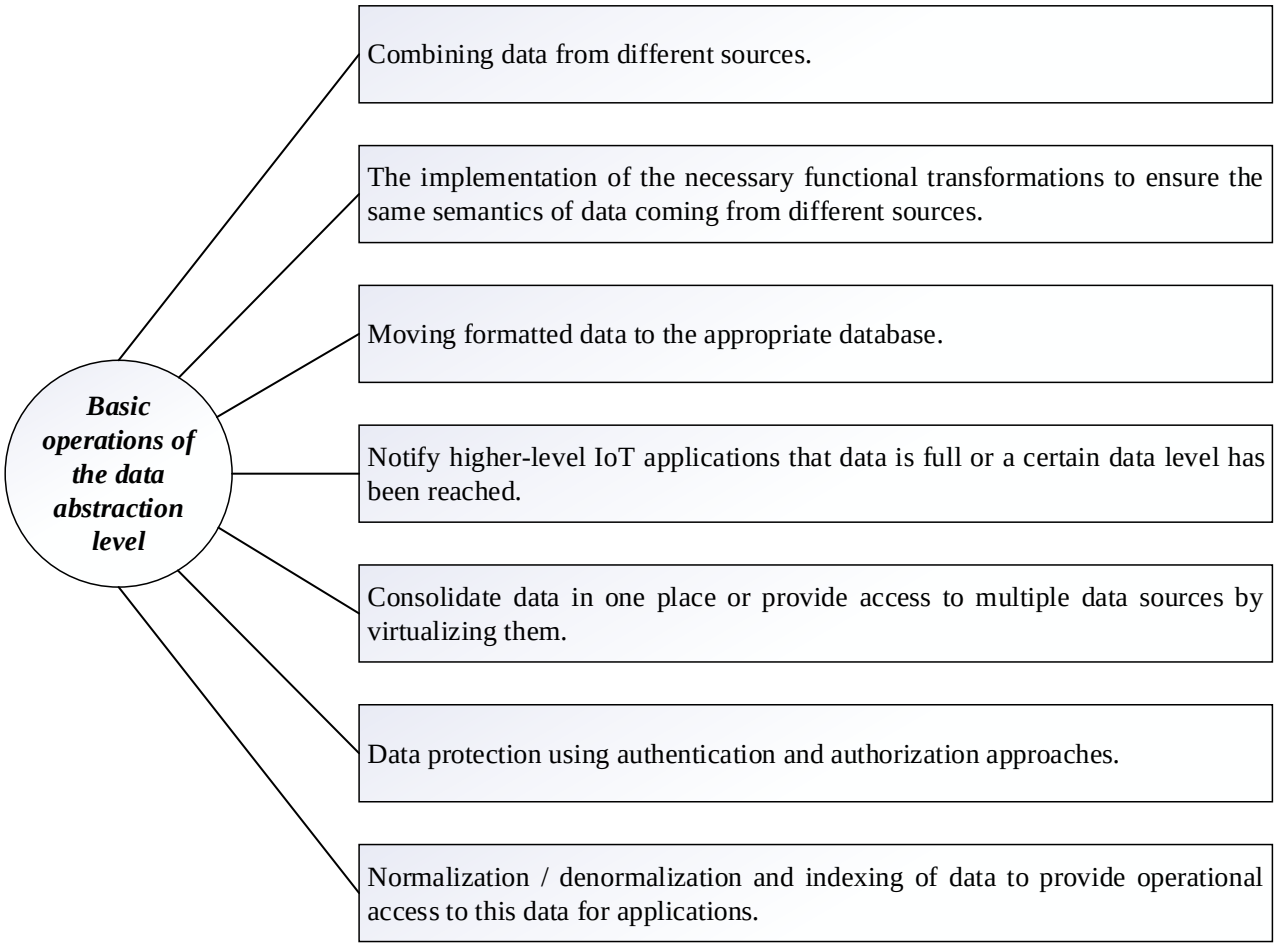


Fig. 2.18. Basic operations of the data abstraction level

The presence of such a level in IoT systems and networks makes application access to data more controlled, responsive and efficient, ultimately reducing the

probability of system errors.

Level 6: Applications. The functionality of this hierarchical level is defined by the specific application of the IoT system or network and may include applications of any type that process input IoT data and / or generate signals and control influences on IoT devices. In most cases, the application level directly interacts with the data abstraction level, namely, processes 'data at rest'. Therefore, there are no specific requirements for the operational speed of functioning compared to the lower levels of IoT systems and networks. Furthermore, in accordance with the recommendations by IWF, when designing IoT hardware and software solutions, the possibility of direct interaction between the application level and the peripheral computing level (level 3) and / or the communication level (level 2) should be provided.

Level 7: Interaction and Processes. This hierarchical level emerged in the general structure of IoT as a recognition of one of the fundamental conceptual principles of the Internet of Things: the ability to interact with humans. This level may include multiple applications and implemented mechanisms for exchanging data and / or information through personal, corporate, or local networks, as well as the Internet.

Thus, a key feature of the reference IoT model by IWF, detailed above, is the substantiation for the development of functional elements of IoT and their subsequent integration into a standardised network organisation.

2.4 The Structural and Functional Organisation of the Web of Things

The Web of Things (WoT) technology is part of a set of standards created and implemented by the World Wide Web Consortium (W3C) to ensure compatibility, fragmentation and enhance usability across various Internet of Things (IoT) applications.

Thus, the Web of Things can be considered a subset of IoT technologies built on standards like REST, HTTP and URIs, enabling the optimisation of network interaction approaches among IoT devices. The typical functional diagram of WoT is shown in Fig. 2.19 and the structural diagram of the interaction between hierarchical levels of

the WoT network is shown in Fig. 2.20.

Fig. 2.19. WoT functional diagram

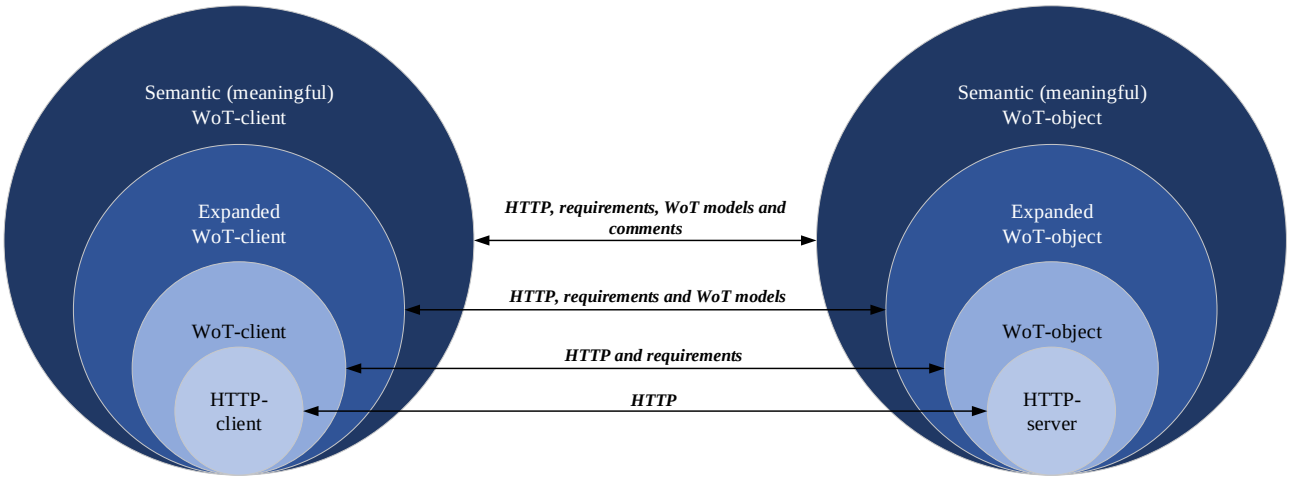


Fig. 2.20. Structural diagram of the interaction of hierarchical levels of WoT networks

The term ‘Web of Things’ (see Fig. 2.20) refers to the digital representation of a real object accessible through a RESTful Web API. An Extended WoT is a virtual object that additionally supports a REST API and a data model defined in this specification, enabling the integration of such objects into more comprehensive systems. A Semantic WoT is a virtual object with a corresponding digital description designed to allow information available on the Internet to be seamlessly transmitted, reused and aggregated by software agents.

As for now, the World Wide Web Consortium has developed three main models for the interaction of WoT-clients with WoT-objects. The choice of the model is determined by the complexity of the architecture of a specific WoT technical solution and the technical and functional capabilities of the devices included in the structure of these solutions.

In the simplest model (direct interaction), WoT-objects are simple Web APIs to which WoT-clients send requests (see Fig. 2.21). In this model, WoT-clients and WoT-objects may be in the same or different networks, but in both cases, WoT-clients send the same requests to WoT-objects. The difference lies in the URL to which the respective request is sent.

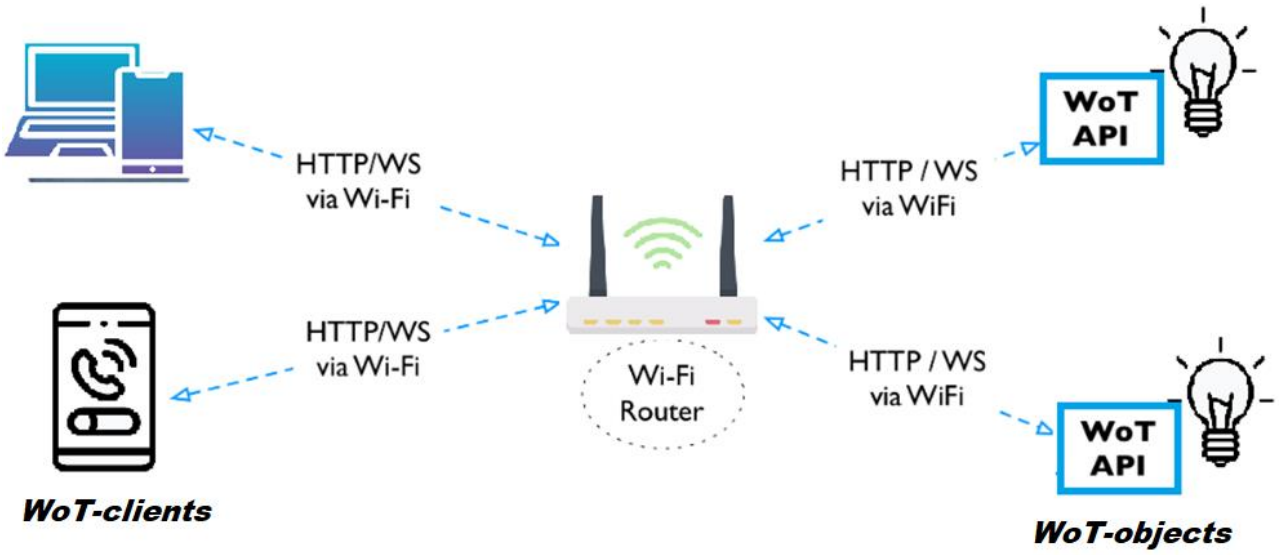


Fig. 2.21. Structural diagram of the direct interaction between WoT-clients and WoT-objects

The second possible model is a connection using a network gateway. This model is used in cases where WoT-objects are unable to directly provide a Web API. In this case, the intermediary is a network gateway that exposes the Web API on behalf of the WoT-object. The structural organisation of the Web of Things interaction model using a network gateway is shown in Fig. 2.22.

The third possible model of the interaction between WoT-clients and WoT-objects is similar to that based on the use of a network gateway, but in this case, a cloud service acts as the gateway rather than a local network device. The structural organisation of the Web of Things interaction model using a cloud server is shown in Fig. 2.23.

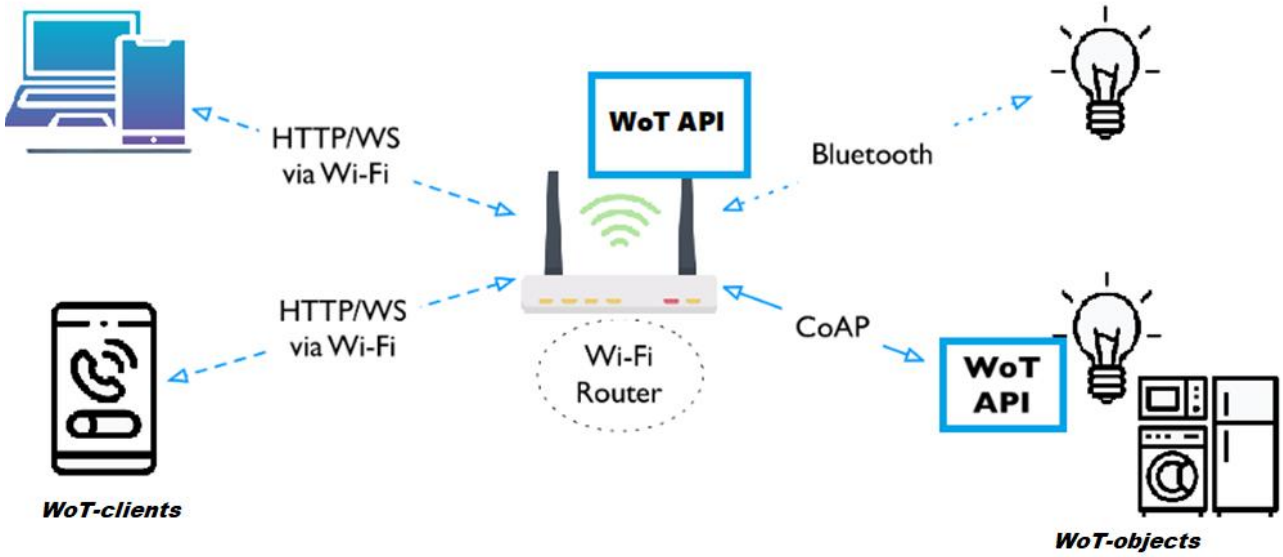


Fig. 2.22. Structural diagram of the interaction between WoT-clients and WoT-objects using a network gateway

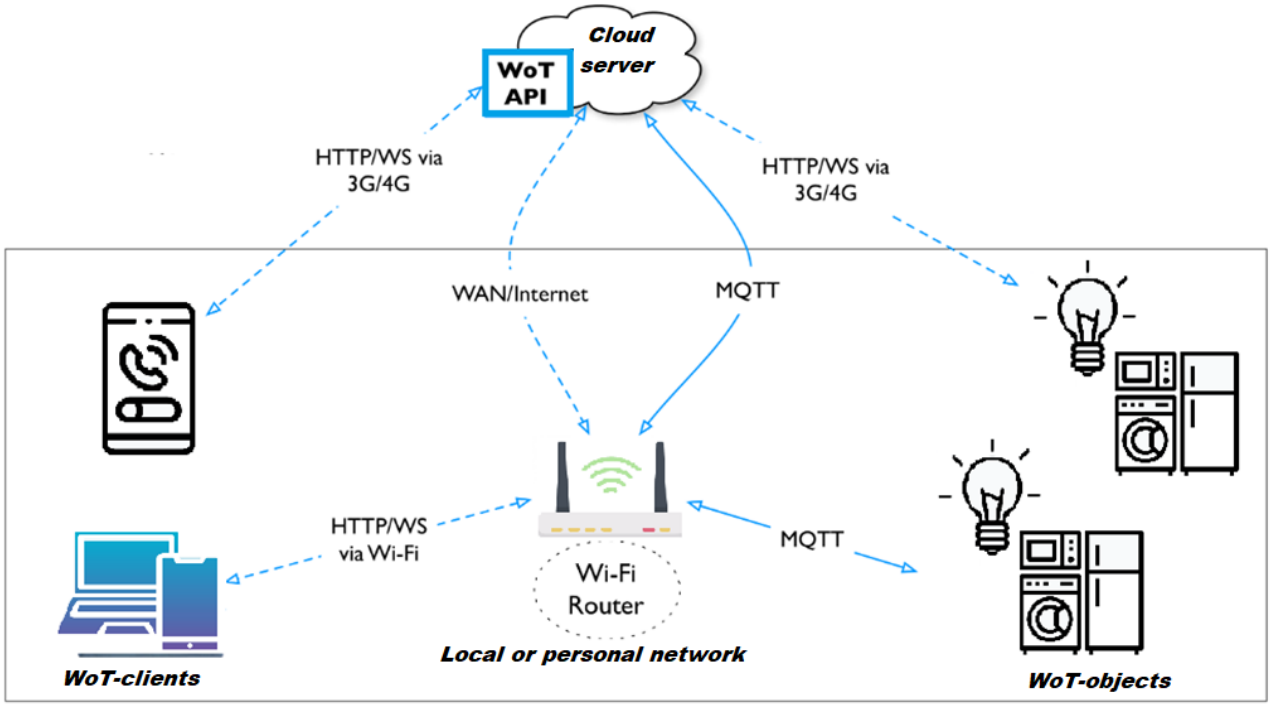


Fig. 2.23. Structural diagram of the interaction between WoT-clients and WoT-objects using cloud technologies

If the key principles of creating and operating Web of Things systems and networks are analysed, they are very similar to those laid down in the concept of the Internet of Things. This is mainly due to the fact that the main purpose of WoT and

IoT is to implement connectivity and data exchange between devices via the Internet. However, there are some differences that explain the purposes and logic of the functioning of these technologies. It is worth noting that the Web of Things functions as an application layer and allows establishing systematic paths for information transmission between networked devices, as well as ensuring compatibility for data generation in the network between clients and objects. The key conceptual principles of the comparative analysis of Internet of Things and Web of Things technologies are shown below in Fig. 2.24.

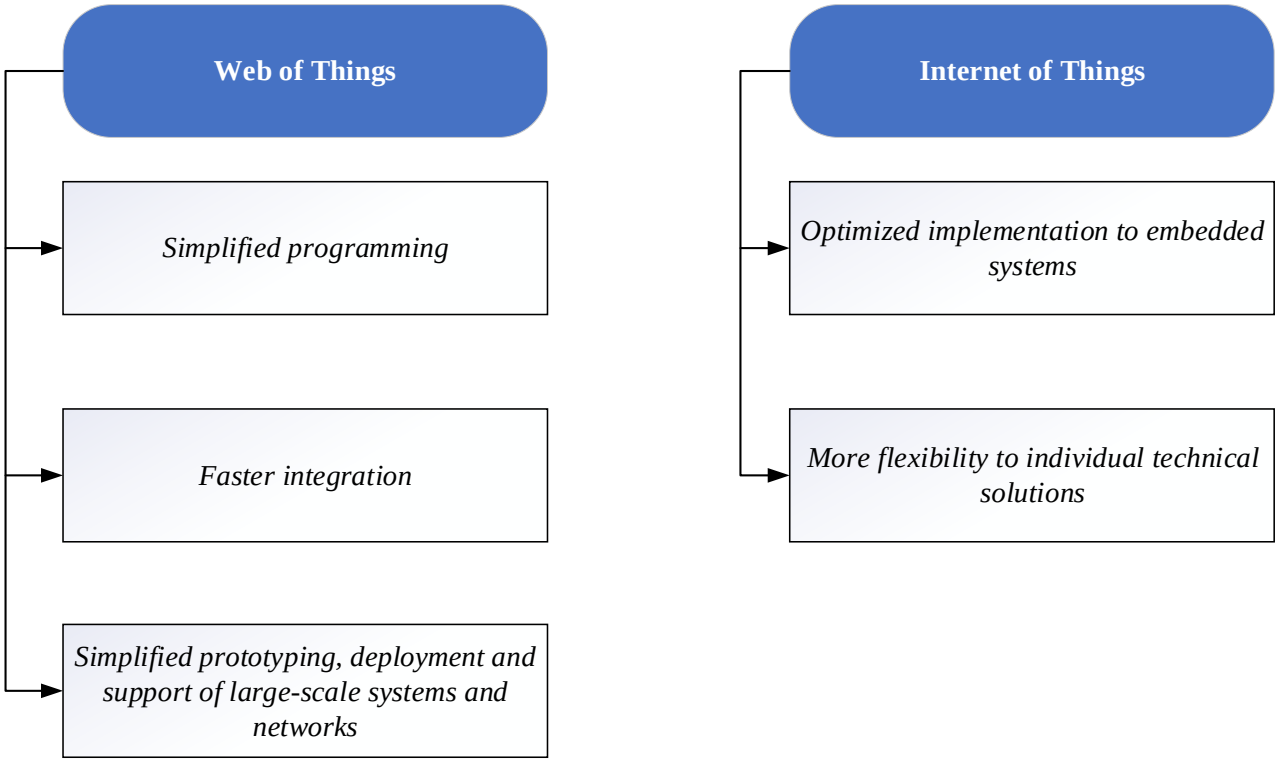


Fig. 2.24. Key functional characteristics of WoT and IoT technologies

Thus, based on the analysis of the information shown in Fig. 2.24, it can be asserted that WoT is not a technology intended to compete with or replace IoT. The primary purpose of WoT is to enhance the functionality of IoT. In essence, WoT is designed to organically complement IoT, primarily through the implementation of approaches to control interaction modes among IoT devices in the global Internet network.

2.5 The Security Framework of the Internet of Things

Since the Internet of Things technology is not standardised, there is no single universal approach to ensure security at present. However, there are some well-known reference models developed and implemented by leading companies in the IoT industry, such as Cisco, Intel, Microsoft, IBM, Symantec, etc.

Security approaches in these reference models for IoT are broadly considered from two perspectives:

– in a multi-level architecture, there is a unified security level covering the entire stack, from the device and network levels to the application level.

– in an end-to-end solution, security mechanisms are implemented in all structural nodes, from the end devices to the cloud server.

It is worth noting that regardless of the security model adopted, the effectiveness of protecting IoT systems and networks from cyber-attacks depends on the level of protection of the devices themselves and ensuring secure connections between them, data storage and the remote IT environment, such as a cloud server. IoT technology developers must consider the need for security at every stage of the hardware and software development process, namely:

– *training stage*: the conceptual foundations of developing cyber-attack-resistant hardware and software are substantiated (secure coding and design, testing and threat modelling);

– *requirements phase*: issues of project confidentiality and security, such as regulatory requirements, are described;

– *design stage*: ensures the establishment of security levels around all software functions;

– *implementation phase*: ensures that the organisation uses coordinated local and network tools, eliminates the possibility of using dangerous functions and performs appropriate data analysis;

– *verification stage*: IoT solution developers ensure that the network solutions used by organisations meet confidentiality and security requirements;

– *release phase:* allows the organisation to track security incidents and respond quickly to them;

– *response stage*: ensures that the organisation can quickly and effectively implement an incident response plan in the event of cyber-attacks or breach of information confidentiality.

An effective strategy for ensuring the information security of IoT solutions should define the security measures that need to be implemented and how they will be monitored or updated over time. The best practices for developing and implementing comprehensive information security strategies for IoT systems and networks include a specific sequence of actions, as shown in Fig. 2.25.

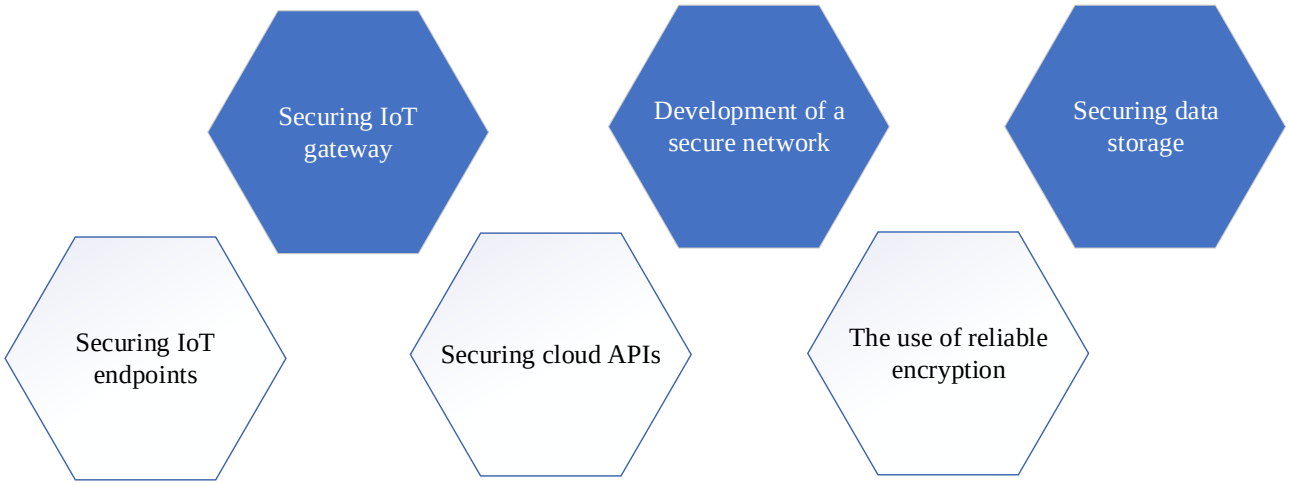


Fig. 2.25. The structure of ensuring comprehensive information security of IoT systems and networks

Securing IoT endpoints involves implementing security approaches on ports with a high degree of risk, such as TCP and UDP protocols, wireless connectivity means and unencrypted communications. Implementing this stage also allows protecting devices at the network edge, giving organisations complete visibility of their network and real-time information about connected devices.

Securing IoT gateways includes the following crucial functions: application control, checking for the secure Hypertext Transfer Protocol (HTTPS) and Secure Sockets Layer (SSL), remote browser isolation and URL filtering. Implementing this

stage helps prevent security risks during web traffic and protects IoT devices from external and internal cyberattacks. Connections can also be secured by threat monitoring solutions that prevent data leaks, as well as Virtual Private Networks (VPNs) that encrypt web browsing data and prevent hackers from tracking user activity on the Internet.

Securing cloud APIs must be implemented through authentication procedures, encryption, the use of tokens and API gateways to protect the data exchanged between the server and network devices.

The development of a secure network ensures proper access control. Such an approach guarantees that only secure, authenticated, or verified devices can connect to the network. Therefore, at this level, it is important to keep authentication keys secure, update antivirus software and continuously monitor network activity to protect devices and users.

The use of reliable encryption is crucial when protecting data 'in motion' between devices or over the Internet. Encryption in IoT is typically implemented using both asymmetric and symmetric encryption methods. Symmetric encryption utilises a single cryptographic key for both data encryption and decryption, while asymmetric encryption utilises open and private keys, providing an elevated level of security.

Securing data storage involves implementing effective, updated antivirus solutions and information monitoring tools that protect the IoT network from threats in a real-time regime. During data storage, it is important to incorporate features such as flexible reporting and scanning, real-time alerts and deep visibility into network activity.

In terms of practical application of the conceptual principles for ensuring comprehensive security of IoT systems and networks described above, Cisco company has developed an Internet of Things security framework. This framework serves as an effective and logical complement to the reference IoT model by IWF. The structural organisation of IoT systems and networks security framework by Cisco is shown in Fig. 2.26.

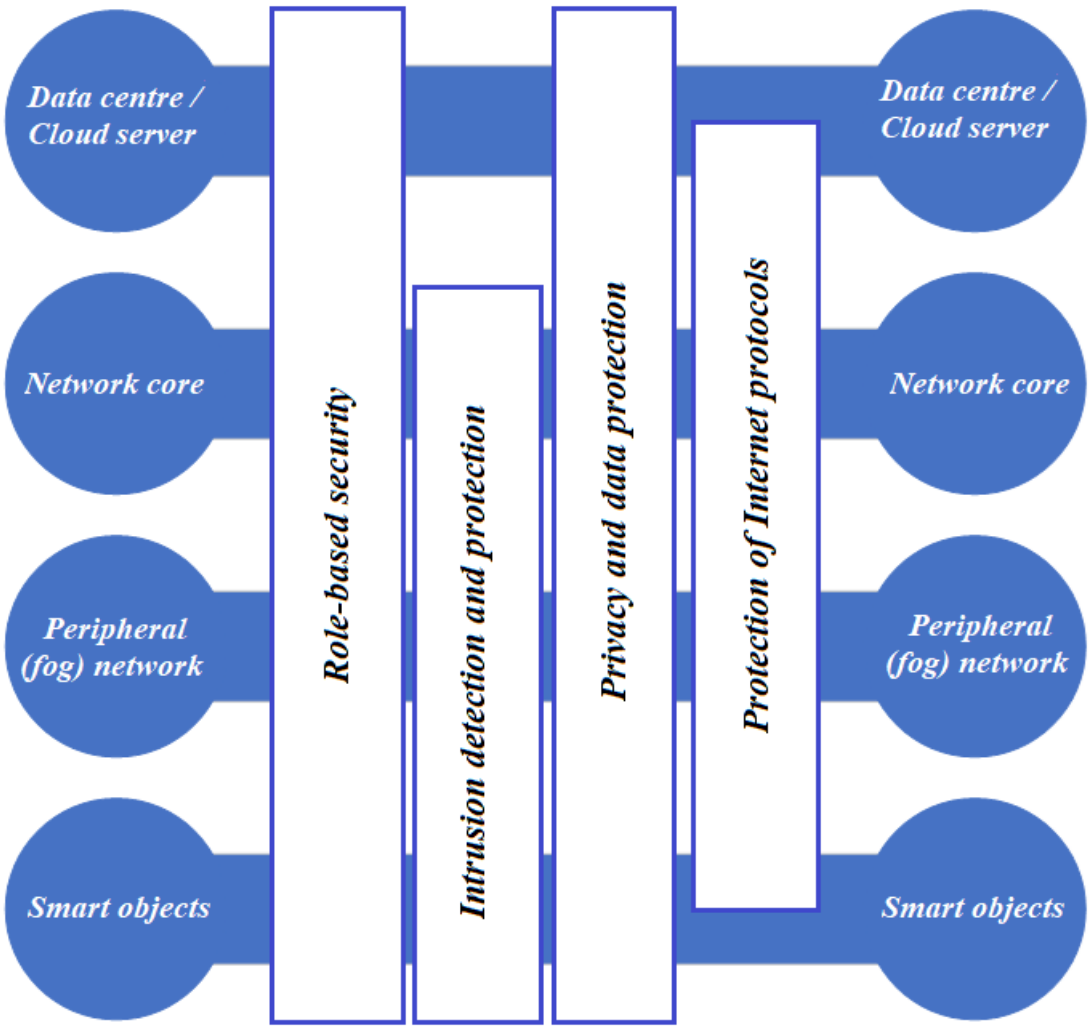


Fig. 2.26. Security framework of IoT systems and networks by Cisco company

As evident from the analysis of the security framework shown in Fig. 2.26, the IoT model by Cisco company is a simplified version of the reference IoT model by IWF. It consists of several levels outlined below.

Smart objects. This level includes sensors, actuators and other embedded devices at the network edge. This part of IoT is the most vulnerable both in terms of physical aspects (devices may be exposed to unprotected and harsh environments for extended periods) and in terms of cyber-physical aspects. Therefore, authentication and data integrity procedures must be applied to all smart objects.

Peripheral (fog) network. The functional elements of this level encompass wired and wireless connectivity technologies for IoT devices. Additionally, some data processing and consolidation may occur at this level using specialised hardware and

software solutions. The key challenge in securing information at this level is the extensive variability of network resources, technologies and protocols on which various IoT devices are based and operate. Thus, there is a need for the development and implementation of a unified security policy.

Network Core. This level implements approaches for data exchange between IoT devices and platforms at the network core. Security challenges at this level are associated with aspects of data exchange in networks with a significant number of endpoints.

Data centre / cloud server. This level consists of specialised application platforms that aggregate and store information, as well as control the network. Thus, security concerns at this level involve issues related to the secure storage and access provision to large amounts of information.

Based on this four-level IoT model, four general cross-level security approaches have been identified by Cisco company, as shown in Fig. 2.27.

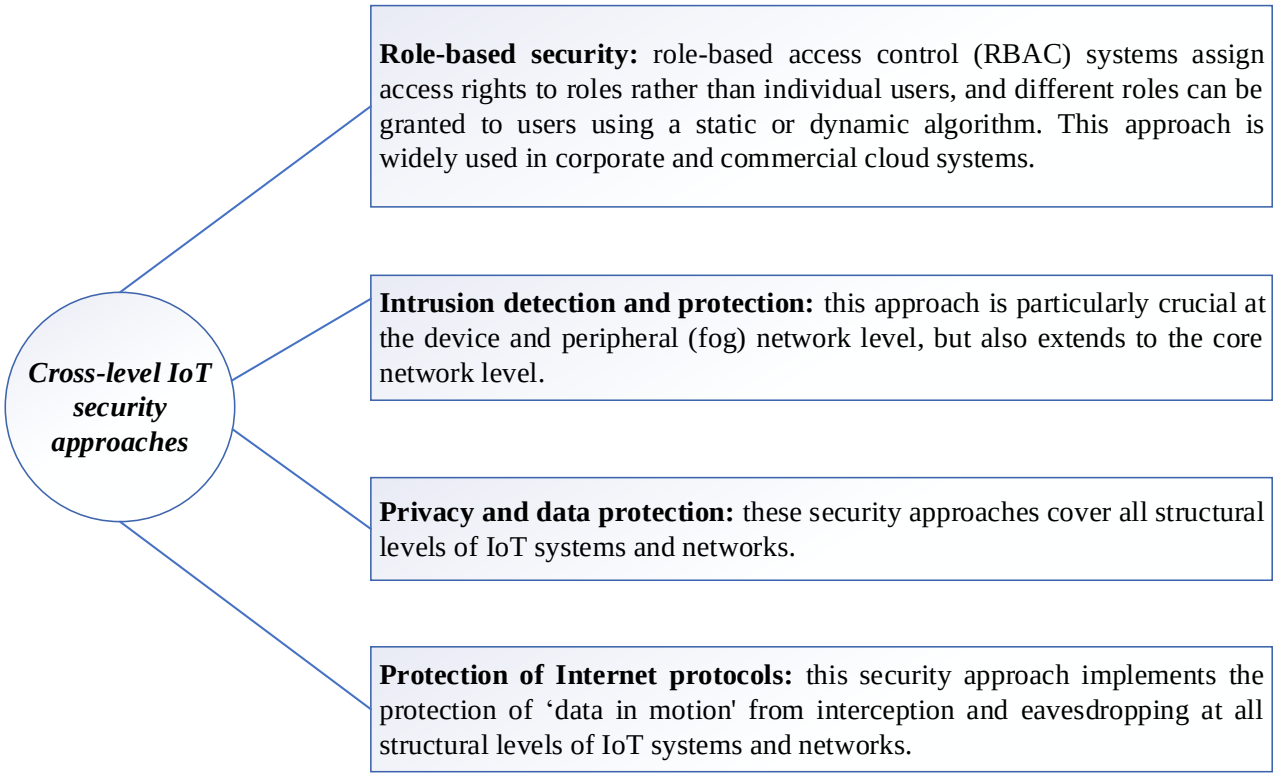


Fig. 2.27. The main cross-level security approaches of IoT systems and networks by Cisco company

Moreover, security components for IoT have been substantiated by Cisco company. These components cover all levels of the model (see Fig. 2.26), in addition to cross-level security approaches (see Fig. 2.27). The specified components in the form of a structure diagram are shown in Fig. 2.28.

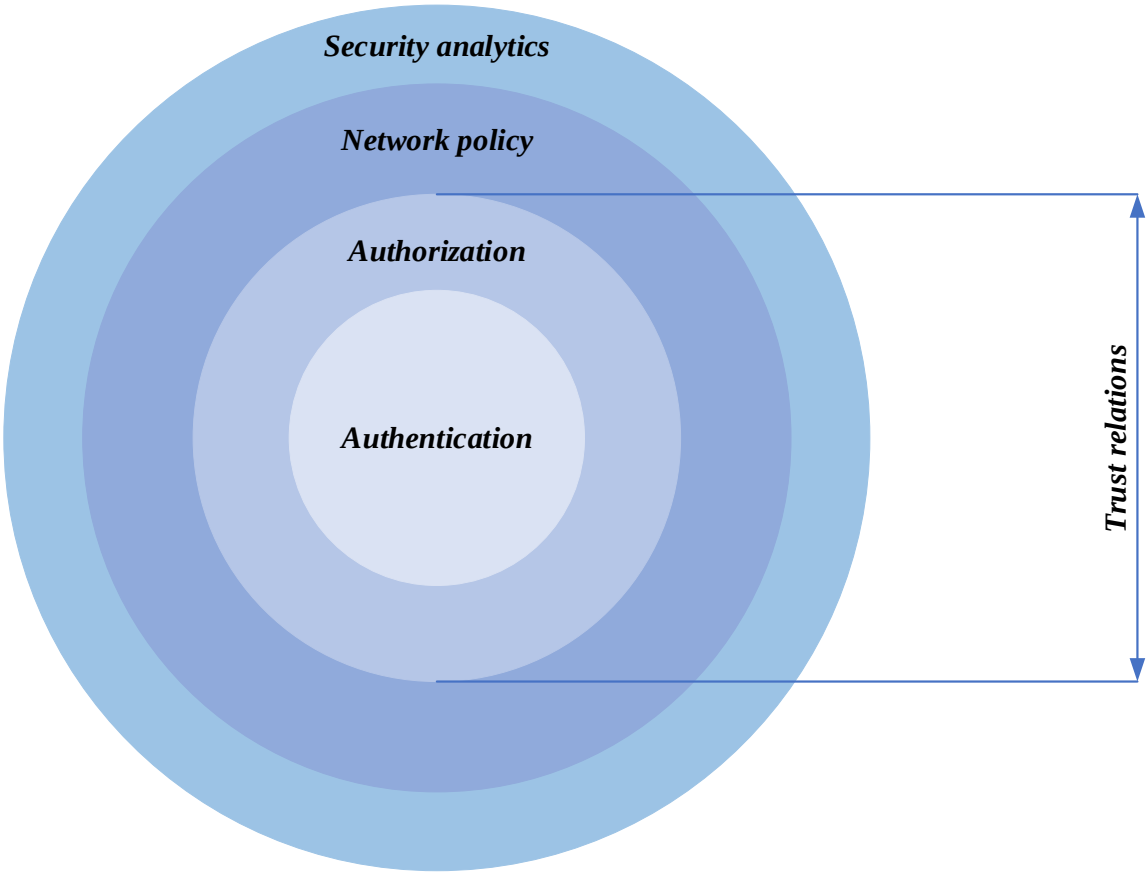


Fig. 2.28. The structural diagram of components of information security of IoT systems and networks by Cisco

Authentication. This security component should cover elements initiating access and is intended for identifying IoT devices in the network. Unlike typical corporate network devices that can be identified by user attributes (login, password, badge, etc.), IoT endpoints should be equipped with authentication means that do not require direct human intervention, such as MAC addresses of endpoint devices, RFID tags and others.

Authorisation. This information security component implements access control functions to devices through the network structure and access control. In conjunction

with authentication, this security component generates the necessary parameters to permit information exchange between devices and between application platforms and devices.

Network Policy. This security component covers all structural levels and nodes implementing routing and traffic transportation functions from IoT endpoint devices throughout the entire network infrastructure.

Security Analytics. This information security component includes all procedures necessary for centralised control of IoT devices, including visibility and function monitoring. Such a security component also enables update and configuration control and allows the implementation of countermeasures to prevent threats.

Furthermore, it is worth noting that an important characteristic of the security framework by Cisco is its reliance on trust relationships. In this context, trust relationships determine the ability of two data-exchanging partners to be confident in each other's identity and access rights. The authentication component, during trust relationships, implements a basic level of trust that should be complemented by the authorisation component.

2.6 The Problem of Standardisation in IoT Technologies

According to the forecasts of analysts and experts in the field of information and digital technologies, diverse hardware and software solutions of IoT systems and networks are expected to significantly outpace IoT technologies based on structurally and functionally compatible standards in their development and implementation. This trend is common for any modern technologies during their dynamic development, formation and deployment phases.

Therefore, the main problem necessitating the search for optimal standardised solutions in the IoT field, as perceived by the global scientific community, is the imbalance between a vast number of devices generating diverse measurement data at a significant speed in various locations and the use of network technologies and cloud services accumulating substantial data in relatively few locations with comparatively

low data update rates.

Eliminating such imbalance through the integration of subsystems for data generation, network transport and remote data storage requires enhanced capabilities of network protocols used at all hierarchical levels of IoT systems, from the physical (the lowest) to the application (the highest) layer.

Numerous leading global organisations in the field of information and digital technology standardisation are actively working to find scientific and applied solutions to solve the aforementioned problem. Their main tasks currently include expanding and adapting Internet protocols for IoT systems and networks, as well as formalising the architecture of IoT technologies.

Self-assessment Questions on Material Comprehension

1. Provide a characterisation of the generalised architecture of IoT systems and networks. Explain the functional purpose of its hierarchical levels.

2. Provide the content of the fundamental technical and functional characteristics of hardware and software solutions of IoT systems and networks.

3. Characterise the reference model of the Internet of Things Y.2060

4. Provide a characterisation of the functional purpose of the main hierarchical levels of the IoT model Y.2060.

5. Provide an overview of business models for the organisation and functioning of IoT systems and networks outlined in the recommendations Y.2060.

6. Characterise the structural organisation of the reference model of the Internet of Things according to the IWF version.

7. Describe the fundamental principles of the hierarchical levels of the reference model of the Internet of Things according to the IWF version.

8. Provide a comparative analysis of the IoT reference models in versions Y.2060 and IWF.

9. Provide the substantive content of the structural and functional organisation of the Web of Things technology.

10. Provide a comparative characterisation of WoT and IoT technologies.

11. Characterise the main stages of ensuring information security for IoT systems and networks.

12. Provide the substantive content of the structure of the process of ensuring comprehensive information security for IoT systems and networks.

13. Provide a characterisation of the IoT security framework by Cisco company.

14. Provide the content of the structural diagram of information security components of IoT systems and networks by Cisco.

15. Describe the problems faced by developers in the search for standardisation paths of IoT systems and networks.

List of Recommended Literature for the Second Chapter

1. Patel K.K., Patel S.M. Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges. International Journal of Engineering Science and Computing, 2016, Vol. 6, No. 5. P. 6122 – 6131.

2. Zhurakovsky B.Yu., Zeniv I.O. Internet of things technologies. Study guide [Electronic resource]: education manual for students speciality 126 ‘Information systems and technologies’, specialisation ‘Information support of robotic systems’. Kyiv: KPI named after Igor Sikorskyi, 2021. 271 p. (in Ukrainian).

3. Links C., Tesla T., Anderton J., Van Hoogstraeten W., Schnaufer D., Warschauer C. Internet of Things: 2nd Special Edition. Hoboken: John Wiley & Sons Inc., 2021. 44 p.

4. Java Point: IoT Tutorial. Available at: https://www.javatpoint.com/iot-internet-of-things (accessed on December 26, 2023).

5. Telesphere: What you need to know about the Internet of Things. Available at: https://www.telesphera.net/blog/iot-likbez.html (accessed on December 26, 2023) (in Ukrainian).

6. Business workshop: Internet of Things: network architecture and security

architecture. Available at: https://www.bizmaster.xyz/2020/12/internet-rechei-merezheva-arkhitektura-ta-arkhitektura-bezpeky.html (accessed on November 16, 2023) (in Ukrainian).

7. Y.2060. Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks. Next Generation Networks – Frameworks and Functional Architecture Models / ITU-T. Geneva: International Telecommunication Union, 2013. 22 p.

8. W3: Web Thing Model. Available at: https://www.w3.org/Submission/2015/SUBM-wot-model-20150824/ (accessed on November 12, 2023).

9. Web Thing Model: Specifications and Standards. Available at: https://webofthings.org/standards/ (accessed on December 22, 2023).

10. Embedded Computing Design: Introduction to Web of Things (WoT). Available at: https://embeddedcomputing.com/technology/iot/introduction-to-web-of-things-wot-heres-everything-you-need-to-know (accessed on December 03, 2023).

11. Fortinet: IoT Vulnerabilities: An Overview. Available at: https://www.fortinet.com/resources/cyberglossary/iot-best-practices (accessed on October 25, 2023).

12. Laktionov I., Vovna O., Kabanets M. Computer-Oriented Method of Adaptive Monitoring and Control of Temperature and Humidity Mode of Greenhouse Production. Baltic J. Modern Computing, 2023, Vol. 11 (1). P. 202–225.

13. Kharchenko V.S. (ed.). Internet of Things for Industry and Human Application. In Volumes 1–3. Vol. 2. Modelling and Development. Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. 547 p.

CHAPTER 3
HARDWARE AND SOFTWARE MEANS AT THE PHYSICAL DEVICE LEVEL

3.1 General Information about Sensors

As mentioned in Chapter 1 ‘General Information about the Internet of Things’, the conceptual basis of IoT involves obtaining measurement data about any physical objects (processes, phenomena). Therefore, one of the crucial roles in the Internet of Things is played by devices that generate measurement data about the parameters and characteristics of objects (processes, phenomena). These measurement data must be presented in a format that can be received and processed by digital computing devices. Thus, the hardware basis for creating the physical devices level of IoT systems and networks, which is responsible for generating measurement data, consists of sensors.

The term ‘*sensor*’ refers to a structurally completed information and measuring device that transforms a controlled physical parameter into a signal represented in a form convenient for further processing in a measuring system.

As of today, the convenient form is considered to be the representation of physical parameters in the form of electrical signals, as it allows satisfying the conditions of telemetry (signal transmission over a distance) and automation of data collection and processing. Thus, one of the main functional tasks of sensors is to convert non-electrical physical parameters into electrical signals or parameters. This approach determines the use of electrical measurement methods in the construction of Internet of Things (IoT) systems and networks, which have the following advantages:

– high accuracy and sensitivity of transformation;

– ergonomics in the implementation of transformation, accumulation and storage procedures;

– insignificant inertia in the transformation process;

– high reliability of circuit solutions;

– continuity and real-time operability of measurements;

– scalability and modularity of the structure;

– wide dynamic measurement range.

The structure of a typical sensor, which performs converting non-electrical physical parameters into electrical ones, is shown in Fig. 3.1. Such a sensor includes a sensitive element (to detect a specific physical parameter) and a transducer for converting non-electrical parameters into electrical signals.

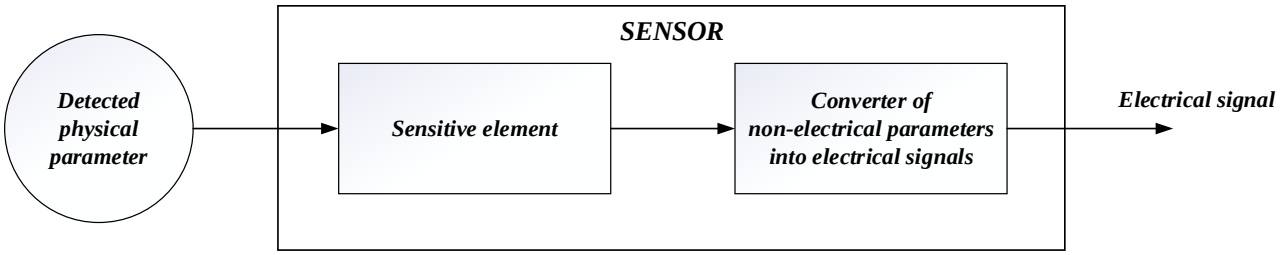


Fig. 3.1. The structure of a typical sensor

Depending on the types of sensors and their functional purposes, the transducer of non-electrical parameters into electrical signals, which is a structural element of sensors, can perform a wide range of tasks, such as amplification, filtering, analogue-to-digital conversion, etc.

To systematise the broad class of sensors known today, several classification features are used, including the type of output parameter (signal), operating principle, nature of the output signal, signal transmission medium, number of output parameters (signals), manufacturing technology and others.

According to the type of output parameter (signal), sensors are divided into:

– generative (active): the output signal is current, voltage, or charge;

– parametric (passive): the output parameter is resistance, capacitance, or inductance.

According to the principle of operation, sensors are divided into:

– mechanical resilient: the principle of operation is based on the relationship between input mechanical forces and the resulting displacements or mechanical stresses in the material of the sensitive element, determined by its elastic properties;

– resistive: the principle of operation involves a change in the electrical resistance of the sensitive element under the influence of mechanical displacements, temperature, illumination, etc.

– capacitive transducers: the physical transformation principle involves the dependence of the capacitance of a capacitor on the distance between its electrodes or on changes in the dielectric permeability of the medium between these electrodes;

– piezoelectric transducers: the principle of operation is based on the use of the phenomenon of polarisation of a piezoelectric material under the action of mechanical forces;

– inductive transducers: the principle of operation is based on the dependence of the total electrical resistance of the winding on changes in the integral magnetic resistance of the core.

– mutually inductive or transformer transducers: the principle of operation is based on the dependence of the electromotive force of the secondary winding on changes in the integral magnetic resistance of the transformer core;

– induction transducers: the principle of operation is based on the use of the phenomenon of electromagnetic induction;

– galvanomagnetic transducers: the principle of operation is based on the use of Gauss or Hall galvanomagnetic transducers;

– thermal transducers: the principle of operation is based on physical effects determined by any thermal processes;

– electrochemical transducers: the principle of operation is based on the dependence of the electrical conductivity of substances on their chemical or physical properties;

– optical transducers: the principle of operation is based on the dependence of the parameters of optical radiation on the value of the converted parameter;

– ionization transducers: the principle of operation is based on the transformation of the intensity of ionizing or X-ray radiation.

According to the nature of the output signal, sensors are divided into:

– analogue: produce a continuous signal over time;

– digital: generate a binary code (sequence) equivalent to the measured parameter;

– signal: generate a signal with only two logical levels (1 or 0).

According to the medium used for signal transmission, sensors are divided into:

– wired;

– wireless.

According to the number of output variables, sensors are divided into:

– one-dimensional;

– multidimensional.

According to the manufacturing technology, sensors are divided into:

– elemental;

– integrated.

It is worth noting that in the context of the Internet of Things concept, the expansion of the functional capabilities of typical sensors is crucial in the following directions:

– intellectualization: implementation of computational procedures at the level of the sensor itself;

– network interaction.

Hardware and software solutions of sensors with advanced functional capabilities allow for the practical implementation of intellectualized machine-to-machine (M2M) interaction of distributed devices in IoT systems and networks.

3.2 Intelligent Sensors

The conceptual principles of the peripheral architecture of IoT systems and networks necessitate the creation of intelligent embedded sensors, enabling the implementation of a range of processing and analytics tasks of measurement data at the physical level of Internet of Things systems. Thus, a typical intelligent sensor usually consists of four main parts: the sensor itself (see Fig. 3.1), an analogue-to-digital converter (ADC), a microcontroller module and a network communication device. The

typical structure of an intelligent sensor is shown in Fig. 3.2.

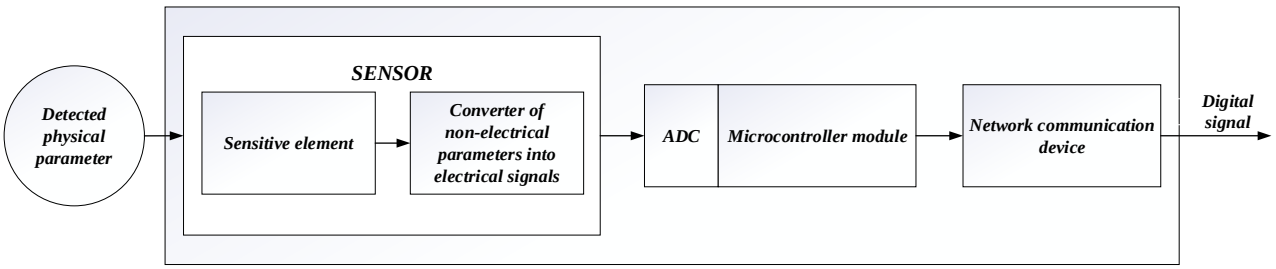


Fig. 3.2. The structure of an intelligent sensor

The operation principle of typical intelligent sensors (see Fig. 3.2) is as follows. The sensitive element performs selective detection of a physical parameter, which, through the converter of non-electrical parameters, is converted into an equivalent electrical signal. This signal is then converted into a digital code using an analogue-to-digital converter (ADC). The digital signal is processed by the microcontroller module using algorithms. The measured and processed data are transmitted to the upper levels of IoT systems and networks through the network communication module.

It is worth noting that in modern circuit solutions, the analogue-to-digital converter (ADC) and microcontroller module are often integrated into a single integrated device.

A crucial factor for further transformation and interpretation of the measurement results at the level of IoT system applications is the timestamping of measurement data. It is therefore essential to associate the code combinations of the measurement results with information about the real-time when these measurement results are obtained by the monitoring system.

In many practical applications of IoT systems and networks, a multi-factor analysis of monitored and controlled objects (processes, phenomena) is assumed. This is made possible by constructing multi-channel devices at the physical level based on the simultaneous connection of a specific set of sensors to one microcontroller module, as shown below in Fig. 3.3.

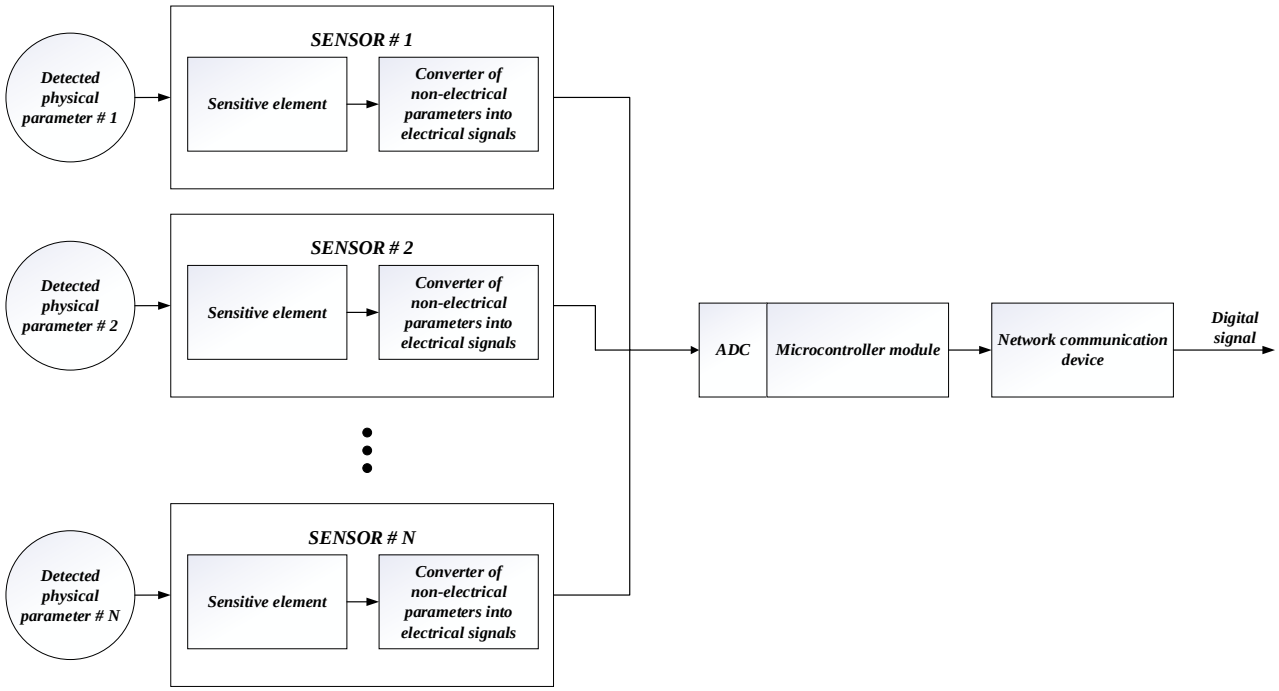


Fig. 3.3. The structure of multi-channel intelligent information and measuring device

In such multi-channel intelligent information and measuring devices (see Fig. 3.3), sensors can detect both homogeneous physical parameters, allowing the analysis of large-scale objects (processes, phenomena) in different locations and heterogeneous physical parameters, allowing a qualitative transition from monitoring parameters to intelligent multi-factor monitoring of the state of objects (processes, phenomena). This approach is a highly effective initiative in the context of the Internet of Things, which, in turn, is a conceptual component of Industry 4.0. It allows the implementation of an intelligent operational transformation of a large amount of measurement data at the peripheral level.

3.3 The Main Parameters and Characteristics of Sensors

At present, various manufacturers have developed a considerable number of sensors for industrial and domestic use, including intelligent sensors. Therefore, the choice of a sensor for specific purposes in practical applications should be based on a multifactor analysis of its technical, functional, metrological and economic indicators.

As mentioned in subsection 3.1 ‘General Information about Sensors’, a sensor,

in terms of its primary aim, can be considered as a transducer that converts the time-varying input physical parameter $x(t)$ into an equivalent output electrical signal $s(t)$.

In a dynamic mode, the variation over time of input parameters and output signals of sensors entails the presence of functional relationships in the form of differential equations, established based on the physical principles of operation and schematic solutions of specific types of sensors. The general form of the relationship equation between input and output variables is as follows:

$$f_1\left(d^{(n)}s/dt^{(n)},\ d^{(n-1)}s/dt^{(n-1)},...,s\right)=f_2\left(d^{(m)}x/dt^{(m)},\ d^{(m-1)}x/dt^{(m-1)},...,x\right), \qquad (3.1)$$

where x is the input parameter; s is the output signal; n and m are the orders of derivative functions.

In a static (steady-state) mode, there is no time dynamics in the input variables and output signals of sensors. Therefore, all derivatives of s and x in the equation (3.1) are assumed to be zero. Thus, equation (3.1) takes the form of an algebraic equation, describing the static transformation characteristic of the sensor. This transformation characteristic is known as the calibration or graduational characteristic of the sensor.

The static characteristics of sensors describe how accurately the output signal s reproduces the measured physical parameter x in a steady-state mode (after the completion of the transient process). The main static characteristics of sensors include accuracy, sensitivity, operating range, linearity, resolution, reproducibility and static gain.

Accuracy of measurements is the main characteristic of measurement quality, reflecting the proximity of the measurement result to the true value of the measured parameter.

Sensitivity is the ratio of the change in the sensor's output signal to the change in the input parameter that causes it. The sensitivity of a sensor in general can be calculated using the following formula:

$$Sensitivity = \frac{ds}{dx}, \tag{3.2}$$

or, in the specific case of sensors with a linear transformation characteristic:

$$Sensitivity = \frac{\Delta s}{\Delta x}. \tag{3.3}$$

Thus, formulas (3.2) and (3.3) can be used for a quantitative assessment of the sensors' operation.

An operating range is the range of values of the measured parameter within which the sensor's errors are normalized.

A linearity is a qualitative characteristic of sensors that is not described analytically but is determined qualitatively based on the shape of the sensor's transformation characteristic graph. The closeness of this graph to a straight line determines the degree of linearity of the sensor.

A resolution is the smallest change in the measured physical parameter that can be detected and accurately reproduced by the sensor.

The reproducibility of measurements is a quality characteristic reflecting the closeness of measurement results for the same quantity taken under different conditions (different times, different places, different methods, or means).

A static gain (gain at a direct current) is the amplification coefficient of the sensor at low frequencies.

The dynamic characteristics of sensors are influenced by a whole spectrum of parameters and factors that are practically not specified in the technical documentation of sensors because of the significant impact of the operating conditions. For the standardisation of sensor testing methods to establish and compare dynamic characteristics, the dynamics of the real physical parameter $x(t)$ is commonly replaced with typical input influences such as harmonic, step, impulse and others. The sensor's response to typical influences can be determined experimentally or analytically based

on Laplace transform or in specialised software packages such as Matlab & Simulink.

Key specific parameters describing the dynamic characteristics of sensors include the time of passage through the insensitive zone, time delay, rise time and time to reach the first maximum. The graphical interpretation of the methodology for determining the dynamic characteristics of sensors based on the graph of the output signal change over time *s*(*t*), is shown in Fig. 3.4.

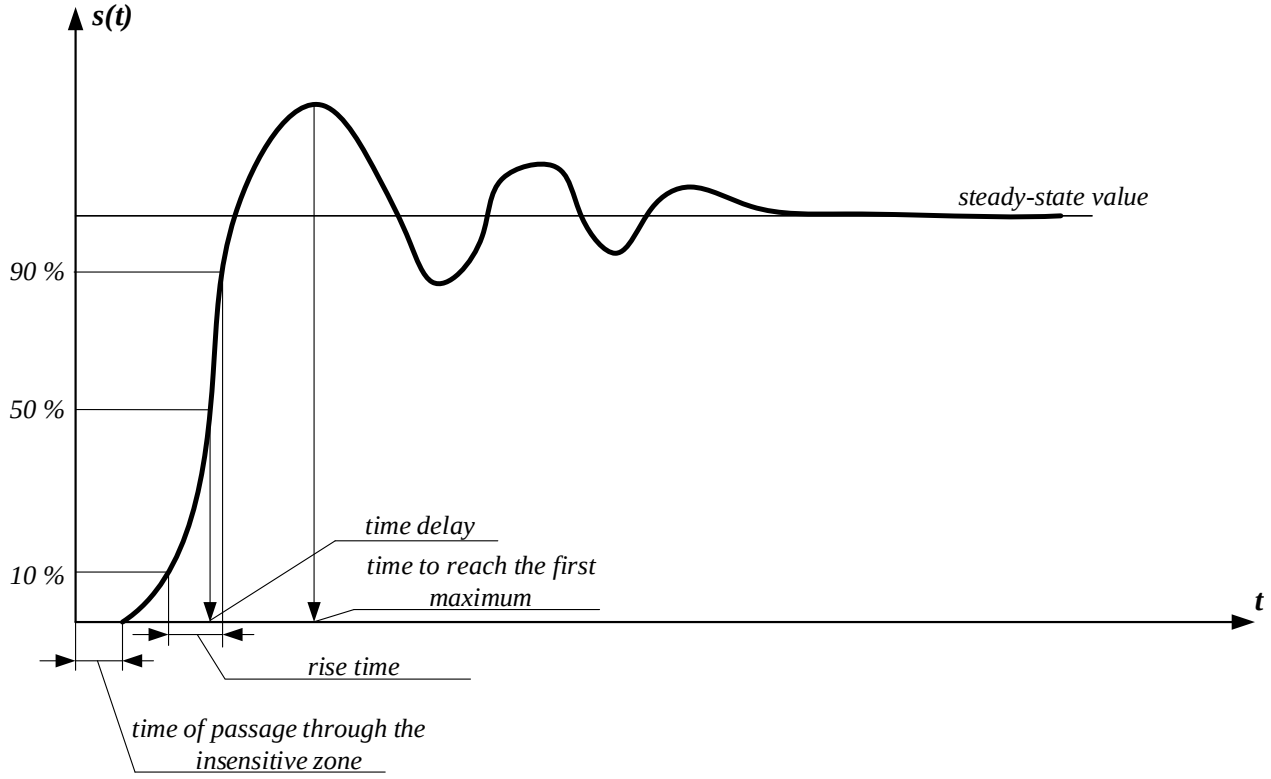


Fig. 3.4. The graphical interpretation of the technique to determine the dynamic characteristics of sensors

The time of passage through the insensitive zone is the time between the start of the input physical parameter detection procedure and the moment of the sensor's response (the onset of the output signal change).

A time delay is the time it takes for the sensor readings to first reach 50 % of the steady-state value of the output signal.

A rise time is the time it takes for the sensor's output signal to increase from 10 % to 90 % of the steady-state value.

The time to reach the first maximum is the time it takes for the sensor's output signal to reach its first maximum.

The above-mentioned dynamic parameters characterise the inertia, speed and accuracy of sensors.

In the context of creating IoT systems and networks, it is essential to obtain reliable and precise data from sensors in real time and promptly diagnose the reasons that may cause sensors to malfunction. One such reason, significantly limiting the reliability and autonomy of the device level of IoT systems and networks, is the need to ensure autonomous power for components, including intelligent sensors, for an extended period.

Therefore, addressing the scientific and applied problem of optimising power consumption modes (adapting passive and active operating modes) of sensors is relevant during the design and implementation of hardware and software solutions for IoT systems.

3.4 Microcontrollers and Microcomputers

During the construction of the device level of IoT systems, microprocessors, microcontrollers and microcomputers are widely used to address tasks such as aggregation, digital processing and generation of information signals.

This approach serves as the hardware and software foundation for implementing advanced global technologies in distributed computing, including fog and edge computing.

A microprocessor is a programmable microelectronic device designed for data collection, processing and control of processes related to the exchange of this data within information, computer-integrated and / or computational systems.

A microcontroller is a specialised programmable microelectronic device designed for use in systems for data exchange or automated control of technological (production) processes.

A microcomputer is a compact computer that uses a microprocessor as the central

processing unit, performing all logical and arithmetic operations. The main distinctive features of microcomputers include the bus organisation of the system, high standardisation of hardware and software and orientation toward a wide range of application tasks.

In modern IoT systems and networks, serial microcontrollers and microcomputers have gained the highest popularity for designing client parts of IoT projects. Most microcontrollers are programmed using high-level languages such as C, C++, Python and others.

Serial microcontrollers support a significant number of input / output data standards and are characterised by relatively low power consumption compared to programmable industrial logic controllers and microcomputers.

Microcomputers are typically system-on-chip devices based on the Princeton architecture, including a central processor, random-access memory (RAM), network adapters and input / output ports.

Modern microcomputers operate on Windows and Linux operating systems. Compared to microcontrollers, microcomputers have greater computational power and more extensive capabilities for connecting peripheral devices. The main drawback of microcomputers is their higher power consumption relative to serial microcontrollers.

Microcomputers are commonly used in IoT projects involving the development of complex information systems, mini-servers, video streaming processing systems, and others.

Currently, a significant number of specialised hardware computing platforms have been developed based on serial microcontrollers and microcomputers. These platforms support a wide range of IoT projects of varying complexity and application areas. Among the most popular platforms are Arduino, ESP, STM, Raspberry and others. These platforms represent a systemic integration of microcontrollers (microcomputers) with expansion capabilities, such as network adapters, data storage, input / output interfaces and others.

Arduino. The family of Arduino computational microcontroller hardware platforms falls into the category of DIY (do-it-yourself). These platforms are equipped

with input / output elements and use the Processing / Wiring development environment in the C / C++ programming language, which is a simplified subset. This family of microcontroller platforms has several implementations (see Fig. 3.5) that differ in their technical and functional characteristics, namely: the number of analogue and digital ports, flash memory size, clock frequency, RAM size, type of USB connector, operating voltage and others.

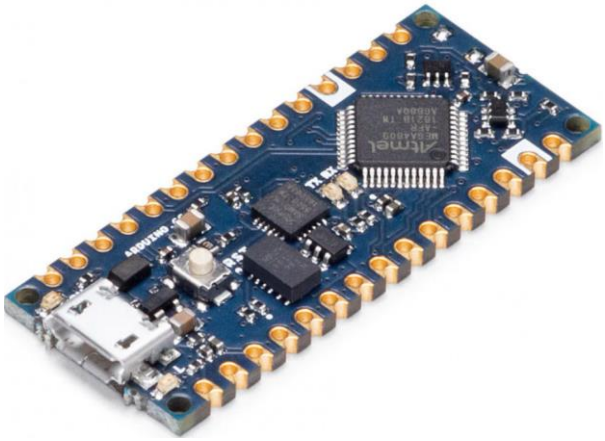
The main technical and functional characteristics of the Arduino microcontroller platform family are given in Table 3.1.

a) Arduino Uno

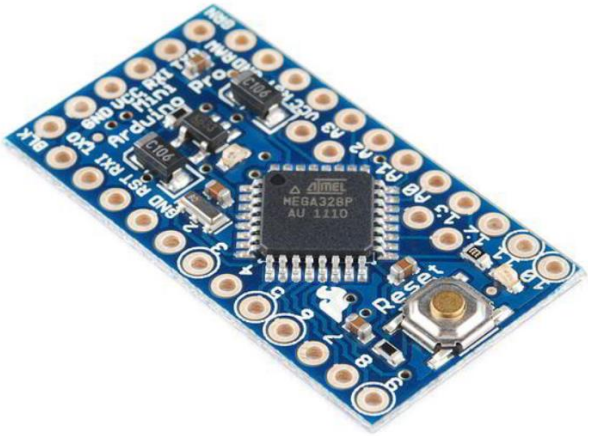
b) Arduino Leonardo

c) Arduino Due

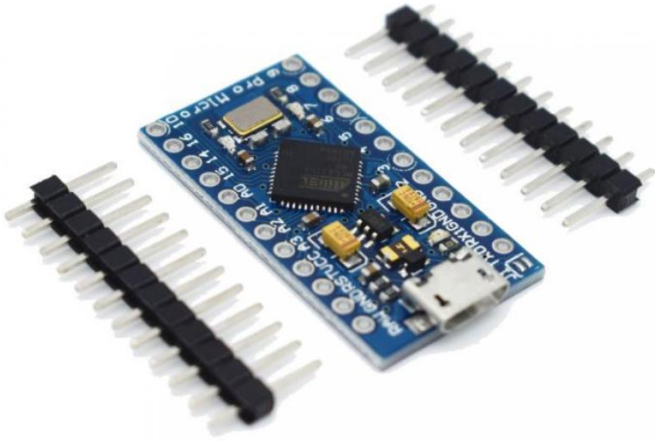
d) Arduino Mega



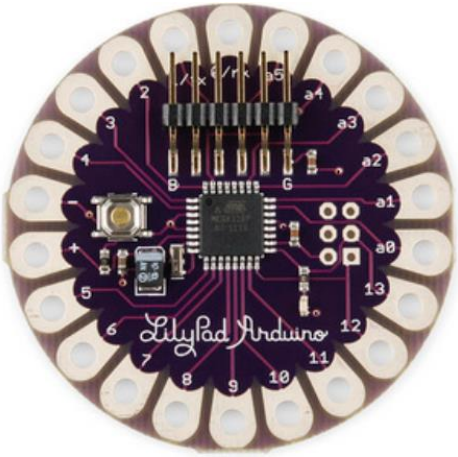
e) Arduino Nano



f) Arduino Mini



g) Arduino Micro



i) Arduino LilyPad

Fig. 3.5. External view of Arduino microcontroller platforms

Table 3.1. Technical and functional characteristics of Arduino microcontroller platforms

Board / **Parameter**	***Uno***	***Leonardo***	***Due***	***Mega***	***Nano***	***Mini***	***Micro***	***LilyPad***
Microcontroller	ATMega 328	ATMega 32u4	Atmel SAM 3X8E ARM	ATMega 2560	ATMega 328	ATMega 328	ATMega 32u4	ATMega 328p
Operating voltage, V	5,0	5,0	3,3	5,0	5,0	5,0	5,0	2,7 – 5,5
Number of digital inputs / outputs	14	20	54	54	14	14	20	20
Number of PWM outputs	6	7	12	14	6	6	7	6

Continuation of Table 3.1

Number of analogue inputs	6	12	12	16	8	6	12	6
Maximum current from a digital pin, mA	40	40	800	40	40	40	40	40
Flash memory size, KB	32	32	512	256	32	32	32	32
RAM size, KB	2	2,5	96	8	2	2	2,5	2
EEPROM-memory size, KB	1	1	–	4	1	1	1	1
Clock frequency, MHz	16	16	84	16	16	16	16	8
Dimensions, mm	69 x 53	75 x 53	102 x 53	102 x 53	18 x 45	30 x 18	48 x 18	D=50
Type of USB connector	A-B	micro	micro	A-B	mini	serial	micro	serial

The Arduino Portenta H7 device (see Fig. 3.6) can be classified into a separate category of Arduino microcontroller platforms. It is capable of simultaneously executing high-level code and real-time computational tasks. The design of this device includes two processors that implement parallel task execution. Therefore, the functional purpose of the Arduino Portenta is dual and consists of the following:

– the board operates such as any other embedded microcontroller board, performing typical tasks of data collection and processing;

– the board acts as the main processor of the embedded computer (microcomputer).

The main technical and functional characteristics of the Arduino Portenta H7 microcontroller platform include:

– microcontroller type: dual-core STM32H747 with an integrated graphics core;

– RAM: 8 MB;

– flash memory: 16 MB;

– operating voltage: 5 V;

– analogue-to-digital converter: 3 x 16-bit;

– digital-to-analogue converter: 2 x 12-bit;

– presence of peripheral devices: 10/100 Ethernet PHY; NXP SE050C2 secure

element; Bluetooth and Wi-Fi interfaces; external antenna; DisplayPort via the USB-C interface.

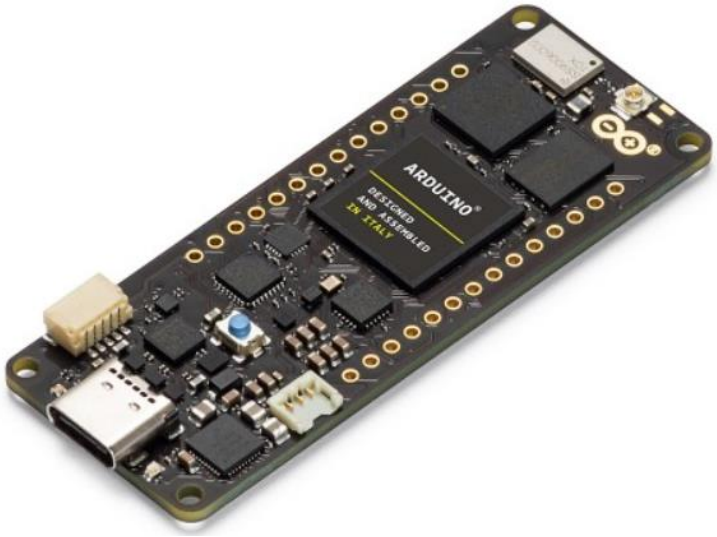


Fig. 3.6. External view of the Arduino Portenta H7

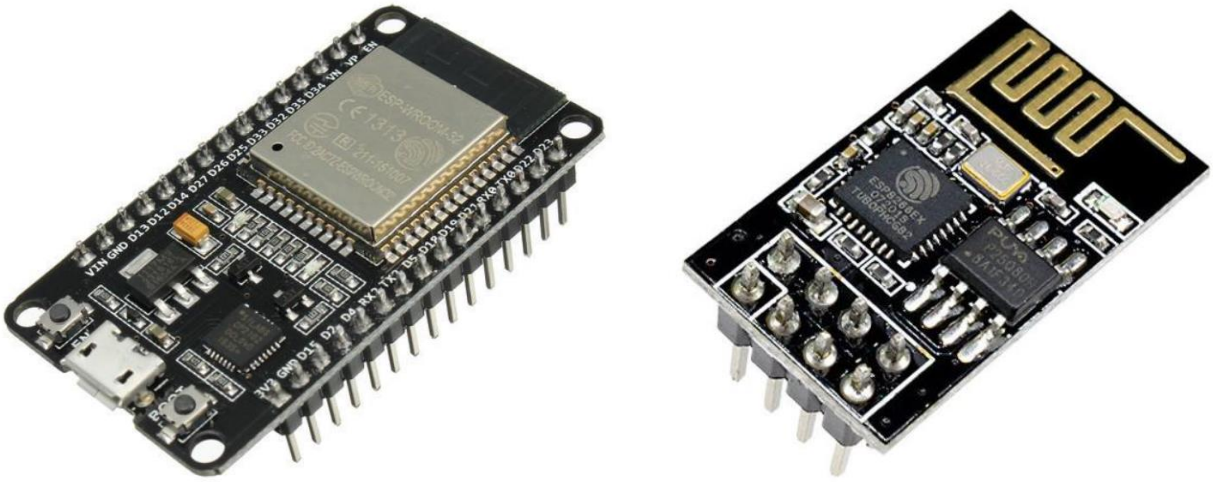
Based on the analysis of the technical and functional characteristics of the Arduino Portenta H7, it can be concluded that this device is potentially suitable for applications beyond the DIY category, requiring enhanced computing performance and main program size.

Therefore, the wide range of Arduino microcontroller platforms available on the electronics market today allows for the implementation of a significant range of applications in computer-integrated monitoring and control at the device level in IoT systems and networks.

ESP. These devices belong to the category of budget-friendly microcontroller Wi-Fi modules, similar to Arduino, designed for the implementation of a wide range of applications in the context of IoT systems and industrial automation. The main characteristic feature of ESP modules is the integrated Wi-Fi interface for creating wireless sensor networks (WSN). Currently, there are two main models of ESP modules in the electronic devices market (see Fig. 3.7): ESP32 and ESP8266. A typical pinout for these modules is shown in Fig. 3.8. A comparison of the technical and functional characteristics of ESP modules is given in Table 3.2.

Table 3.2. Technical and functional characteristics of ESP modules

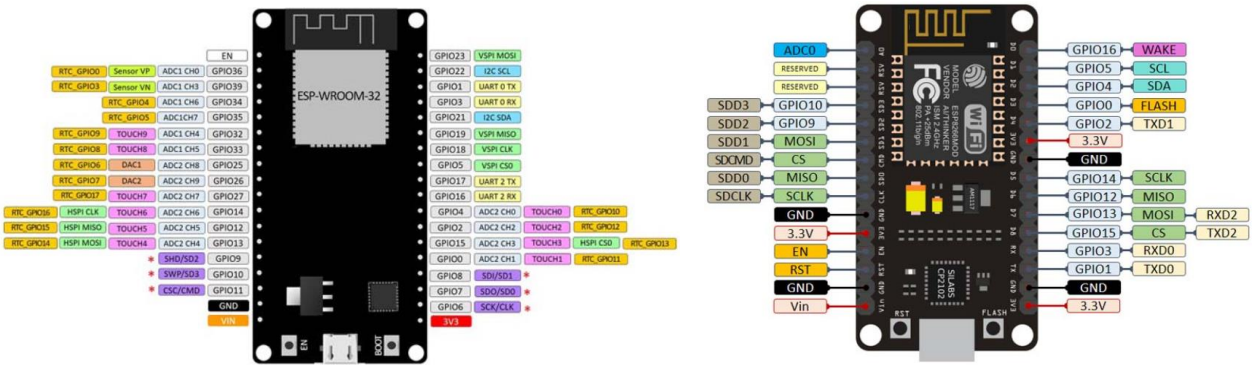
Module / **Characteristics**	***ESP8266***	***ESP32***
Microcontroller type	Xtensa® Single-Core 32-bit L106	Xtensa® Dual-Core 32-bit LX6 600 DMIPS
Wi-Fi 802.11 b/g/n	HT20	HT40
Bluetooth	–	Bluetooth 4.2 і нижче
Clock frequency, MHz	80	160
RAM, KB	160	512
Flash memory, MB	16	16
Number of GPIO	17	36
Number of hardware / software PWM outputs	0 / 8	1 / 16
Number of SPI / I2C / I2S / UART interfaces	2 / 1 / 2 / 2	4 / 2 / 2 / 2
Number of CAN interfaces	–	1
Number of Ethernet MAC interfaces	–	1
ADC resolution	10	12



a) ESP32 b) ESP8266

Fig. 3.7. External view of ESP modules

If the suitability of ESP8266 and ESP32 modules for creating IoT systems and networks is compared, the ESP32 module has better potential capabilities. This is because of the presence of extended functionality, such as Bluetooth and built-in wake-up sources.



a) ESP32 b) ESP8266

Fig. 3.8. Pin assignment of ESP modules

Programming both ESP modules can be done using different approaches. The first possible method is to use the Arduino IDE development environment, to which corresponding libraries for working with ESP modules have been added. The second popular programming method for these modules is to use MicroPython, which is a reimplementation of Python 3 for microcontrollers. Unlike programming using the Arduino IDE technique, most scripts in MicroPython are compatible with both ESP8266 and ESP32 boards, allowing the use of the same code for both without additional adaptation.

STM32. This class encompasses a wide range of 32-bit microcontrollers produced by STMicroelectronics Corp. Nowadays, there is a significant number of development boards based on STM32 microcontrollers. One of the most popular typical solutions is the STM32F030F4P6 development board, the external view of which is shown in Fig. 3.9.

The main technical and functional characteristics of the STM32F030F4P6 development board include:

– core architecture: 32-bit ARM Cortex-M0;

– clock frequency: 48 MHz;

– interface support: SPI, I^2C, UART/USART;

– peripheral support: POR, DMA, PWM, WDT;

– flash-memory size: 16 KB (16 KB x 8);

– data memory size: 4 KB;

– generator type: internal (integrated);

– operating voltage range on digital inputs / outputs: from 2.4 V to 3.6 V;

– nominal voltage on analogue pins: 3.6 V.

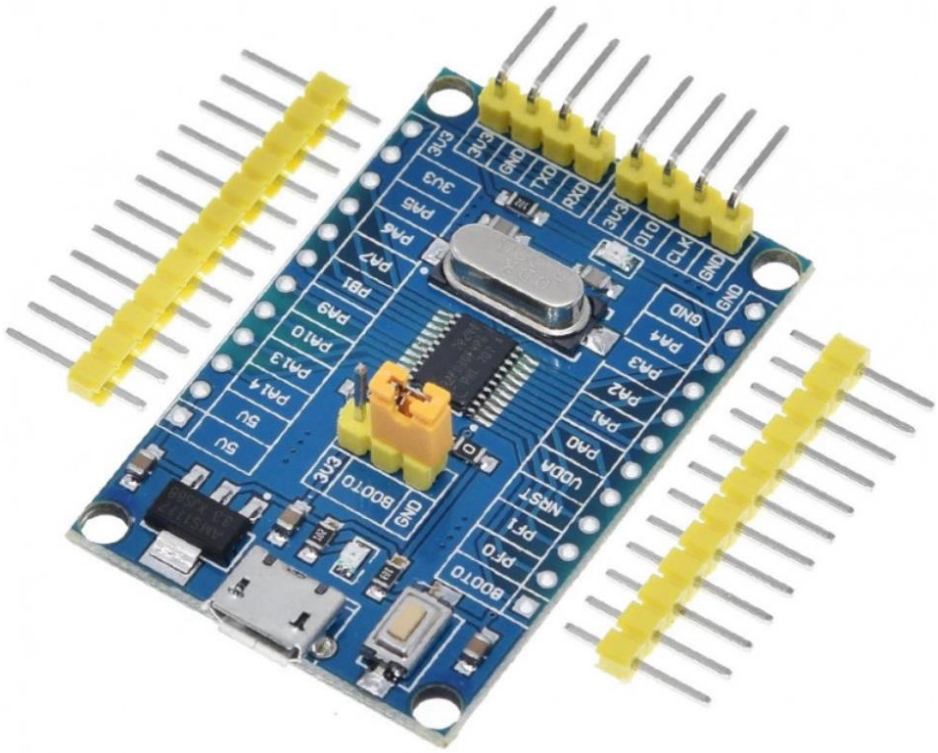


Fig. 3.9. External view of the STM32F030F4P6 development board

Programming STM32 microcontrollers can be implemented using various approaches and corresponding development and testing software. One available approach involves a two-stage development process.

The first stage involves the initial configuration of the microcontrollers and for this aim, tools like STM32CubeMX can be used. STM32CubeMX provides all the necessary tools for generating initialisation code.

The second stage of debugging and testing the software can be performed in a versatile environment such as the IAR Embedded Workbench. This software allows the creation of applications in languages such as C, C++ and assembler for a variety of microcontrollers from different manufacturers.

It is worth noting that the STM32CubeMX initialisation software creates a complete project in the IAR Embedded Workbench environment, including the project

tree, code files and necessary libraries.

The range of applications within the Internet of Things concept, where STM32-type microcontrollers can be used, is quite wide and goes beyond the do-it-yourself (DIY) category when compared to microcontroller platforms such as Arduino and ESP. Specifically, STM32 microcontrollers can find applications in areas such as remote monitoring and control of household and industrial appliances, video reception and processing, digital television, gaming and GPS platforms and others.

Raspberry Pi. This class of devices belongs to the category of microcomputers that operate based on one of the specialised operating systems. Currently, Raspberry Pi microcomputer platforms have undergone four generations of modifications. The external view of the Raspberry Pi 4B, which belongs to the fourth generation of these devices, is shown in Fig. 3.10. Additionally, the assignment of GPIO pins of the Raspberry Pi 4 is shown in Fig. 3.11.

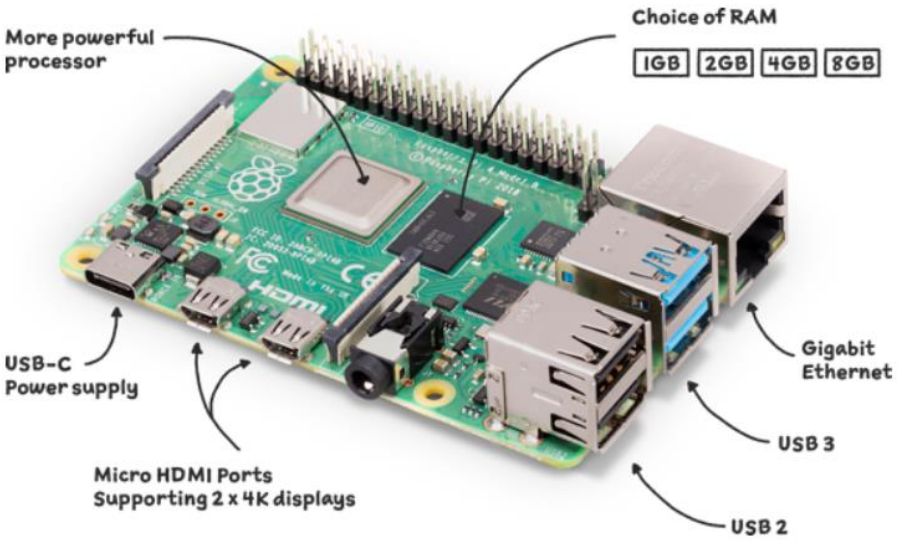


Fig. 3.10. External view of the Raspberry Pi 4B platform

The current dynamics of Raspberry Pi microcomputer characteristics are given in Table 3.3. As evident from this table, the latest versions of these devices are equipped with Wi-Fi and Bluetooth modules, making them more functional in the context of the conceptual theoretical and applied principles of creating Internet of Things systems and networks.

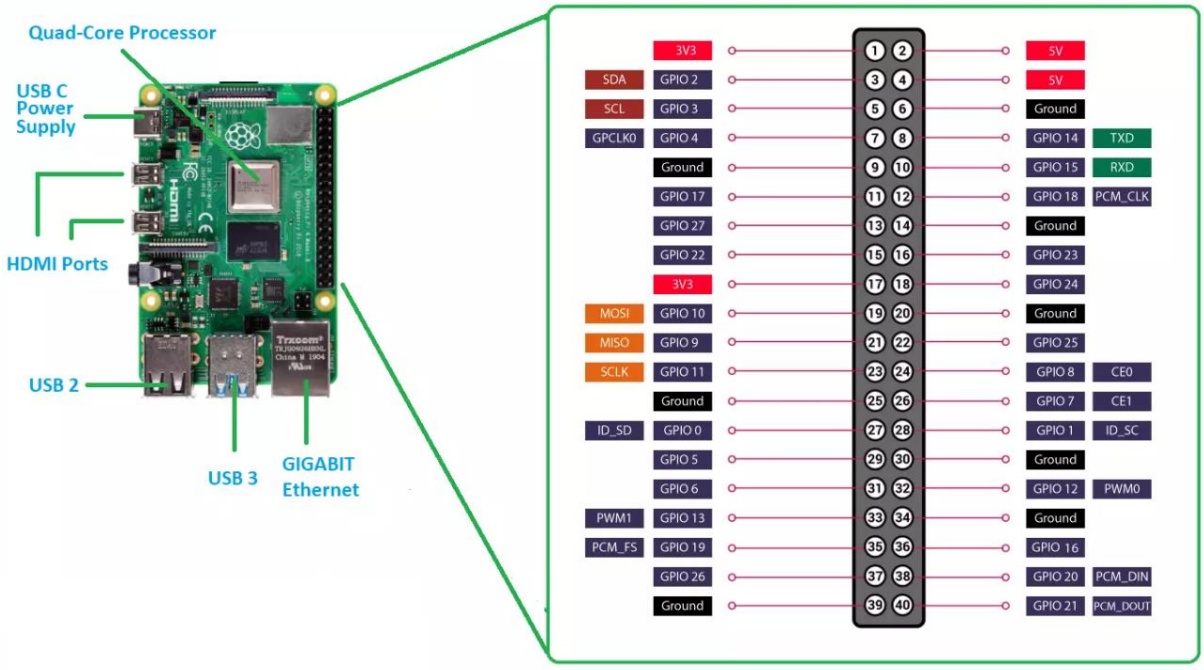


Fig. 3.11. Assignment of GPIO pins of the Raspberry Pi 4

Table 3.3. Technical and functional characteristics of different generations of Raspberry Pi microcomputers

Parameter / **Modification**	***Clock frequency, MHz***	***Number of cores***	***RAM size, MB***	***Number of GPIO***	***Wi-Fi***	***Bluetooth***	***Ethernet***
B (2012)	700	1	512	26			+
A (2013)	700	1	256	26			–
B+ (2014)	700	1	512	40			+
A+ (2014)	700	1	256	40			–
2B (2015)	900	4	1024	40			+
Zero (2015)	1000	1	512	40			–
3B (2016)	1200	4	1024	40	802.11n	4.1	+
Zero W (2017)	1000	1	512	40	802.11n	4.0	–
3B+ (2018)	1400	4	1024	40	802.11n	4.2	+
3A+ (2018)	1400	4	512	40	802.11n	4.2	–
4B (2019)	1500	4	1024; 2048; 4096	40	802.11n	5.0	+

Programming Raspberry Pi microcomputers has its peculiarities compared to microcontrollers discussed earlier (Arduino, ESP and STM). These peculiarities are

associated with the presence of operating systems on which microcomputer platforms operate. One of the most popular operating systems for this class of devices is Raspbian, which is equipped with a special module for configuring and controlling GPIO. This module is a standard addition to the Raspbian OS and is named RPi.GPIO. Another approach to programming Raspberry Pi is to use specialised integrated development environments (IDEs), such as Geany, BlueJ, Lazarus IDE and others.

A significant advantage of utilising microcomputers such as Raspberry Pi in creating Internet of Things systems is the ability to deploy web servers based on them, which is not possible with serial microcontrollers. An additional advantage of introducing microcomputer devices into the structure of IoT systems is the possibility of low-level implementation of complex computational procedures, such as parallel computing and data processing based on artificial neural networks, machine learning and / or neuro-fuzzy logic.

3.5 Actuators and Regulation Modules

One of the fundamental conceptual principles in creating IoT applications is the ability to influence physical objects (processes, phenomena), which can be implemented through the generation of electrical control signals (regulation), including in remote mode (remotely). Such signals are generated based on multi-level processing of measurement data from sensors in IoT systems and networks. Thus, the control signals generated by IoT systems and networks must be directed to the inputs of corresponding actuators and / or regulation modules to exert further influence on physical objects (processes, phenomena).

Among the key actuators (regulation modules) that are part of the applied hardware and software solutions in IoT systems and networks are: servo drives, relays and switches, stepper motors, electromagnetic valves and others. Below is an overview of the main characteristics and parameters of these actuators (regulation modules), which are industrially produced and are structurally and programmatically compatible with a wide range of microcontrollers. They can also be utilised for educational and

research aims in the development and design of do-it-yourself (DIY) IoT technologies.

Servo drives. This category of devices is mechanical drives with automatic state correction through internal negative feedback, according to external parameters.

The primary aim of servo drives is the automatic and precise reproduction and maintenance of the output positions of the actuator based on real-time input control signals.

Therefore, servo drives encompass a wide range of mechanical drives (actuators) that include sensors (speed, displacement, force, position, etc.) in their structure and a control unit (mechanical system and / or electronic circuit) that automatically reproduces and maintains the necessary parameters, in accordance with the initially defined values and external influences.

The external view of typical servo drives, which are structurally compatible with serial microcontrollers and can be used in DIY projects, is shown in Fig. 3.12.

Typically, such servo drives (see Fig. 3.12) are connected to microcontrollers using a three-wire circuit: power supply plus, power supply minus (common wire, ground) and signal wire.

It is worth noting that *digital servo drives* have gained considerable popularity. A distinctive feature of such servo drives is the ability not only to set and maintain the shaft position but also to provide feedback on the parameters of position, supply voltage, holding force, temperature or current consumption.

This implementation can significantly improve the multi-parameter accuracy of automatic control (regulation) units that are part of IoT systems and networks. The digital servo drives are connected to the digital servo drives via a multi-drop half-duplex serial interface, which allows the simultaneous use of several different types of digital servo drives on the same bus.

The main characteristics of servo drives, including digital ones, include torque, operating speed, operating voltage, maximum angle of rotation, housing type, drive type and operating temperature range.

a) MG995 model

b) MG 90S model

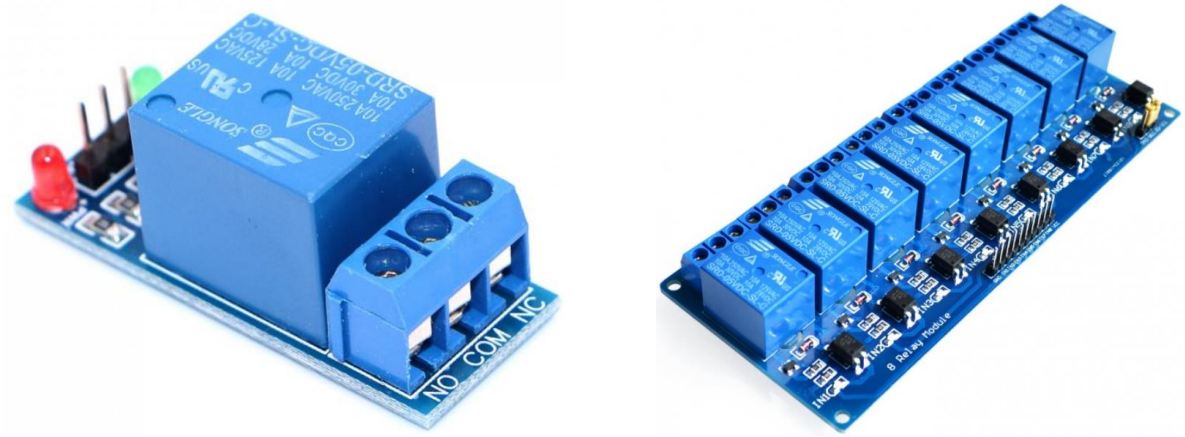
c) DS3115 model

d) SG5010-160 model

Fig. 3.12. External view of typical servo drives for DIY projects

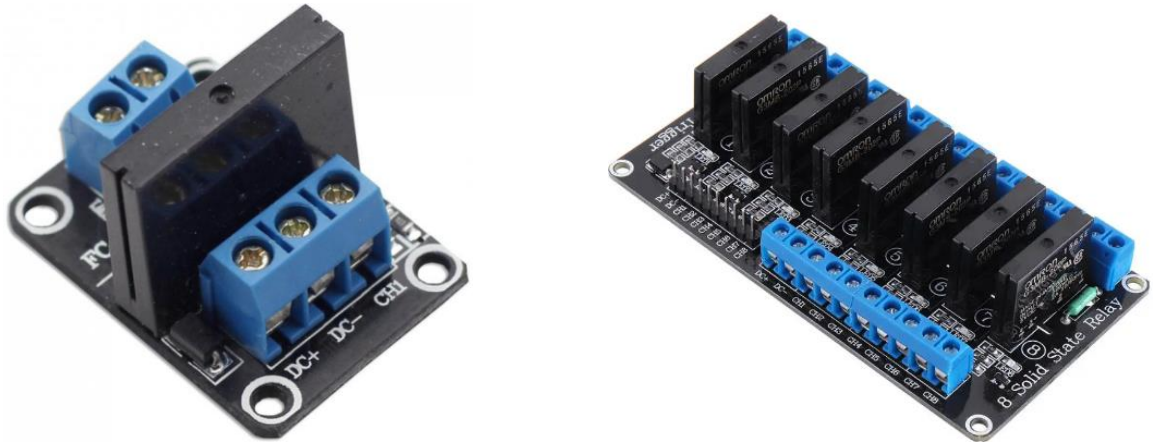
Relays. This category of devices is designed to automatically switch electrical circuits based on the influence of external signals. As for now, two main types of relays are used in IoT systems and networks: electromechanical (see Fig. 3.13) and solid-state (see Fig. 3.14).

To use relay modules in large-scale hardware and software solutions for IoT systems and networks, the industry mass-produces multi-channel assemblies of both electromechanical and solid-state relays. Most modern relay modules are structurally and hardware compatible with commercially available microcontroller platforms by matching the logic levels of the control signals.



a) relay module: 5 V 10 A b) 8-channel relay module: 12 V 10 A

Fig. 3.13. External view of electromechanical relay modules



a) relay module: 5 V 2 A b) 8-channel relay module: 5 V 2 A

Fig. 3.14. External view of solid-state relay modules es

The main characteristics of the relay modules are: switching voltage level, maximum switching current, control voltage value, control current limit and operating current, active logic level, number of channels and operating temperature range.

Stepper motors. Such devices are electric motors in which the power supply by electric current pulses causes a discrete rotation of the rotor by a certain angle of rotation. The stepper motor is controlled by discrete pulses generated by its driver. Depending on the stepper motor model, the angle of rotation as a result of a single pulse can vary in a wide range of values, usually from several degrees to several tens of degrees. A characteristic feature of microcontroller control of stepper motors, unlike ordinary DC motors, is that the controller must switch each of the coils in the motor to

enable it to rotate.

The external view of typical stepper motor models that are compatible with commercial microcontroller platforms is shown in Fig. 3.15.

a) stepper motor 28BYJ-48-12V

b) stepper motor 28BYJ-48-5V complete with ULN2003 driver

Fig. 3.15. External view of stepper motors

The main characteristics of stepper motors are: rated supply voltage, number of phases, number of steps and micro steps, step discreteness, rated frequency, torque, energy saving class, peak current per winding, no-load frequency, friction torque and others.

Electromagnetic valves. This category of devices provides the ability to automatically control the flow of controlled media by commands (supply or reset) in the form of electrical signals. The external view of a typical electromagnetic valve, which is compatible with Arduino microcontroller platforms, is shown in Fig. 3.16, a. It should be noted that at present, electromagnetic mini-valves, the external view of which is shown in Fig. 3.16, *b*, have become very popular in the creation of hardware and software solutions of IoT systems and networks for monitoring gas and non-aggressive liquid media.

а) Electromagnetic valve with ½ inch diameter

б) Electromagnetic mini-valve

Fig. 3.16. External view of electromagnetic valves

The main characteristics of electromagnetic valves (mini-valves) are: rated operating voltage, maximum permissible current, rated minimum holding voltage, normally open or closed state, valve type and material, nozzle diameter, weight and dimensions, operating pressure and temperature range, opening / closing time, productivity and others.

Self-assessment Questions on Material Comprehension

1. Define the term ‘sensor’ and provide a typical block diagram.

2. Give the classification of sensors according to the following features: principle of operation, type of output value, nature of the output signal and the used signal transmission medium.

3. How are the principles of sensor intelligence implemented in practice? Give a block diagram and describe the principle of operation of an intelligent sensor.

4. Describe the main static and dynamic characteristics and parameters of sensors.

5. What is the functional aim of sensors in the context of creating the Internet of Things systems and networks?

6. Define the following terms: microprocessor, microcontroller and microcomputer.

7. Describe the operation principle of the following devices: Arduino, ESP,

STM32 and Raspberry Pi.

8. What are the main differences between microcontroller and microcomputer platforms?

9. What is the functional aim of microcontrollers and microcomputers in terms of creating the Internet of Things systems and networks?

10. Provide a generalised description of the following actuators (control modules): relay, servo, stepper motor and solenoid valve.

11. What is the functional purpose of actuators (regulation modules) in the construction of IoT systems and networks?

List of Recommended Literature for the Third Chapter

1. Laktionov I.S., Vovna O.V. Lecture notes for the disciplines: 'Measurement of Electrical Values and Metrological Support of Electronic Systems', 'Fundamentals of Metrology and Electronic Measurements', 'Measurement of Electrical Values and Metrological Support of Automatic Systems', 'Fundamentals of Metrology' (for full-time and part-time students of all specialities). Pokrovsk: SHEI 'DonNTU', 2017. 80 p. (in Ukrainian).

2. Laktionov I.S., Lebediev V.A., Laktionova H.A. Methodical instructions for practical work for the disciplines: 'Fundamentals of monitoring and information processing in technological production', 'Microprocessor control and information processing devices in power systems', 'Microprocessor technology' (for full-time and part-time students of all specialities). Pokrovsk: SHEI 'DonNTU', 2021. 48 p. (in Ukrainian).

3. Metrology. Terms and definitions: DSTU 2681-94. Effective from 1994-07-26. Kyiv: State Standard of Ukraine, 1994. 68 p. (in Ukrainian).

4. Podzharenko V.O., Vasilevsky O.M., Kucheruk V.Y. Processing of measurement results based on the concept of uncertainty: a textbook. Vinnytsia: VNTU, 2008. 128 p. (in Ukrainian).

5. Electrical engineering online: Sensors of automation systems. Available at:

http://electro-tex.ho.ua/le/m5/5-2 (accessed on December 27, 2023).

6. Arduino. Available at: https://www.arduino.cc/ (accessed on December 26, 2023).

7. Onykiienko Y.O., Ryzhova A.R. Lecture notes ‘Fundamentals of designing Internet of Things systems. The periphery of STM32 microcontrollers: a textbook: electronic network educational edition’. Kyiv: Igor Sikorsky Kyiv Polytechnic Institute, 2022. 127 p. (in Ukrainian).

8. Dnipro University of Technology: What is a microprocessor, microcontroller and programmable logic controller. Available at: https://elprivod.nmu.org.ua/ua/interesting/what_is_mp_mc_plc.php (accessed on January 04, 2024).

9. Raspberry Pi 4. Available at: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ (accessed on January 09, 2024).

10. Lebediev V.A., Laktionov I.S., Vovna O.V., Kabanets M.M., Sahaida P.I., Dobrovolska L.O. Methods of improving technical and functional characteristics of serial budget microprocessor platforms. Journ. Europeen des Systemes Automatises, 2022, Vol. 55 (1). P. 81–88.

11. Vovna O.V., Laktionov I.S., Koyfman O.O., Stashkevych I.I, Lebediev V.A. Study of Metrological Characteristics of Low-Cost Digital Temperature Sensors for Greenhouse Conditions. Serbian Journal of Electrical Engineering, 2020, Vol. 17 (1). P.1 – 20.

12. Vovna O., Laktionov I., Andrieieva A., Petelin E., Shtepa O., Laktionova H. Optimised Calibration Method for Analog Parametric Temperature Sensors. Instrumentation Mesure Metrologie, 2019, Vol. 18 (6). P. 517 – 526.

13. Laktionov I.S., Vovna O.V., Kabanets M.M., Getman I.A., Zolotarova O.V. Computer-Integrated Device for Acidity Measurement Monitoring in Greenhouse Conditions with Compensation of Destabilizing Factors. Instrumentation Mesure Metrologie, 2020, Vol. 19 (4). P. 243 – 253.

CHAPTER 4
METHODS AND MEANS FOR CREATING THE NETWORK LEVEL

4.1 Client-Server Architecture

Currently, two main models are used to implement network interaction between devices:

– centralised computing model: this model can be thought of as an application solution that runs on one or more computers that are not connected to each other;

– distributed computing model: this model can thought of as an application solution that operates on several computers connected simultaneously.

One of the most popular and efficient models for creating distributed infocommunication systems is the client-server architecture. This architecture is based on two main components: the client and the server.

The client is a computer or other user device (such as a mobile or computing device) that is on the user's side. It generates and sends requests to the server to perform specific actions or provide information.

The server is a relatively powerful computer or specialised equipment designed to handle specific tasks, execute program code, provide users with access to certain resources, perform service functions in response to client requests and store information, including creating and managing databases.

Thus, client-server architecture can be defined as a concept of creating and operating an information network in which most of its resources are concentrated on servers that serve a certain number of clients. This architecture defines the following main types of components:

– a set of servers that provide information or other services to applications that access them;

– a set of clients that use the services provided by the servers;

– a network that provides interaction between a certain number of clients and servers.

In accordance with the above, a generalised block diagram of a client-server architecture is shown in Fig. 4.1.

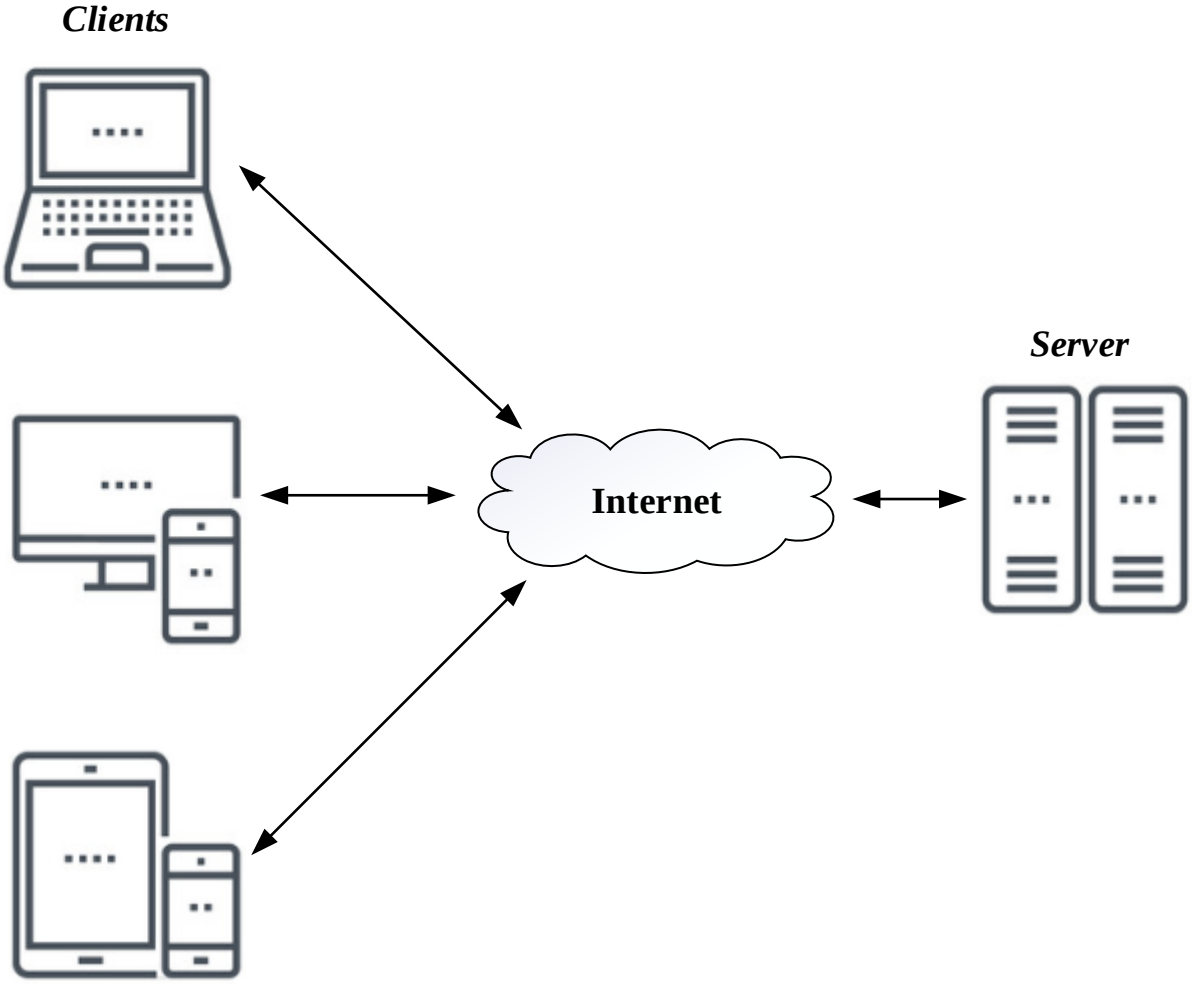


Fig. 4.1. Generalised block diagram of a client-server architecture

Such a model of network organisation means that a client creates a request and sends it to a server, where it is processed and the finished result is sent back to the client. One server can receive and process multiple requests from clients simultaneously. If the server receives more than one request at the same time, they form a queue and are processed sequentially by the server. In certain cases, requests may have priorities and, accordingly, requests with higher priorities should be processed earlier.

In a client-server architecture, it is very important to be clear about what or who can play the role of a client. This can be:

– a client computer from which requests are sent to other computers in the network;

– software that generates and implements requests;

– users who, with the help of appropriate hardware and software, form requests and access the necessary information.

In most cases, clients and servers are the respective software modules that can be located on the same (local server) or different computers.

A set of rules for interaction between the client and the server is called an exchange (interaction) protocol. The main principle of client-server interaction is the division of responsibilities between the client and the server. In general, there are three main levels of such interaction:

– the data presentation level, which is the user interface responsible for presenting data to the user and receiving control commands from the user;

– the application level, which implements the basic logic of data processing using a specific application;

– the data control level, which provides archiving, storage and access to data.

In the practical implementation of the client-server architecture, the following levels of organisation are possible: two-tier, three-tier and multi-tier architectures.

Thus, in general, the distribution of functionality between the client and the server is given in Table 4.1.

Table 4.1. Distribution of functionality between the client and the server

Functions that are implemented on the server side	**Functions that are implemented on the client side**
archiving, storage, access, protection and backup of data;	providing a user interface;
processing of client requests;	creating and sending requests to the server;
sending results (responses) to clients.	receiving query results and sending additional commands (requests to update, add or delete data and others).

The two-tier client-server architecture. Such an organisation involves the interaction of two hardware and software modules: client and server. The server is responsible for receiving requests and sending responses to the client, using only its own resources. In this case, the client is developed to implement and provide the user interface. In general, the principle of the two-tier architecture is that the server receives a request, processes it and responds directly to the client without using any third-party resources. A functional diagram of such an architectural solution is shown in Fig. 4.2.

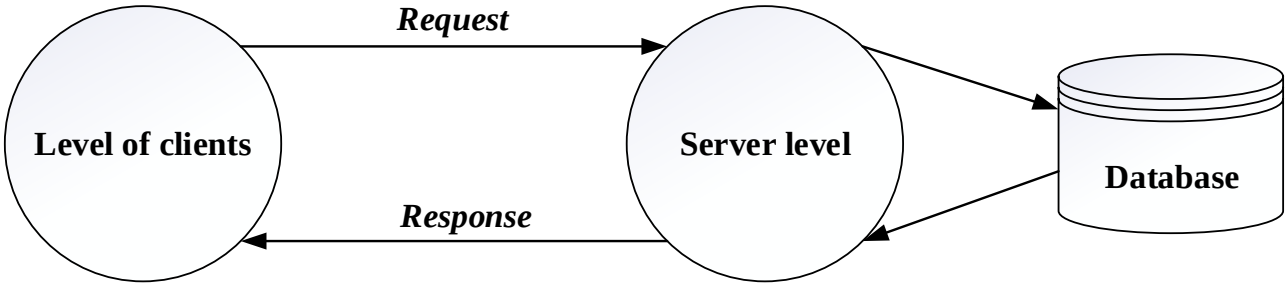


Fig. 4.2. Functional diagram of the two-tier client-server architecture

Depending on how the distribution of functions between the client and the server is implemented, the following organisational models exist:

– thin client model, in which all application and data control logic is implemented on the server and the client application provides only data presentation level functions;

– thick client model, in which the server implements only data control and the application logic and data presentation are performed on the client side.

The main advantages of the two-tier architecture include: ease of configuration and modification of applications; ergonomic user environment; high performance and scalability.

The limitations of this architecture include: a proportional decrease in performance with an increase in the number of users; hypothetical information security issues, as all applications and data are hosted on a central server; all clients are dependent on a single vendor’s database.

The three-tier client-server architecture consists of the following components:

– a data presentation component, which is a specialised user interface;

– an application component, which is an application server;

– a resource control component, which is a database server developed to store and provide information upon request.

In this architecture, a software level is separated, where the application logic is focused and implemented. Intermediate-level applications can be run by specialised application servers, but they can also be run by a regular web server. The logic and data control approaches are implemented by a database server. A generalised functional diagram of the three-tier client-server architecture is shown in Fig. 4.3.

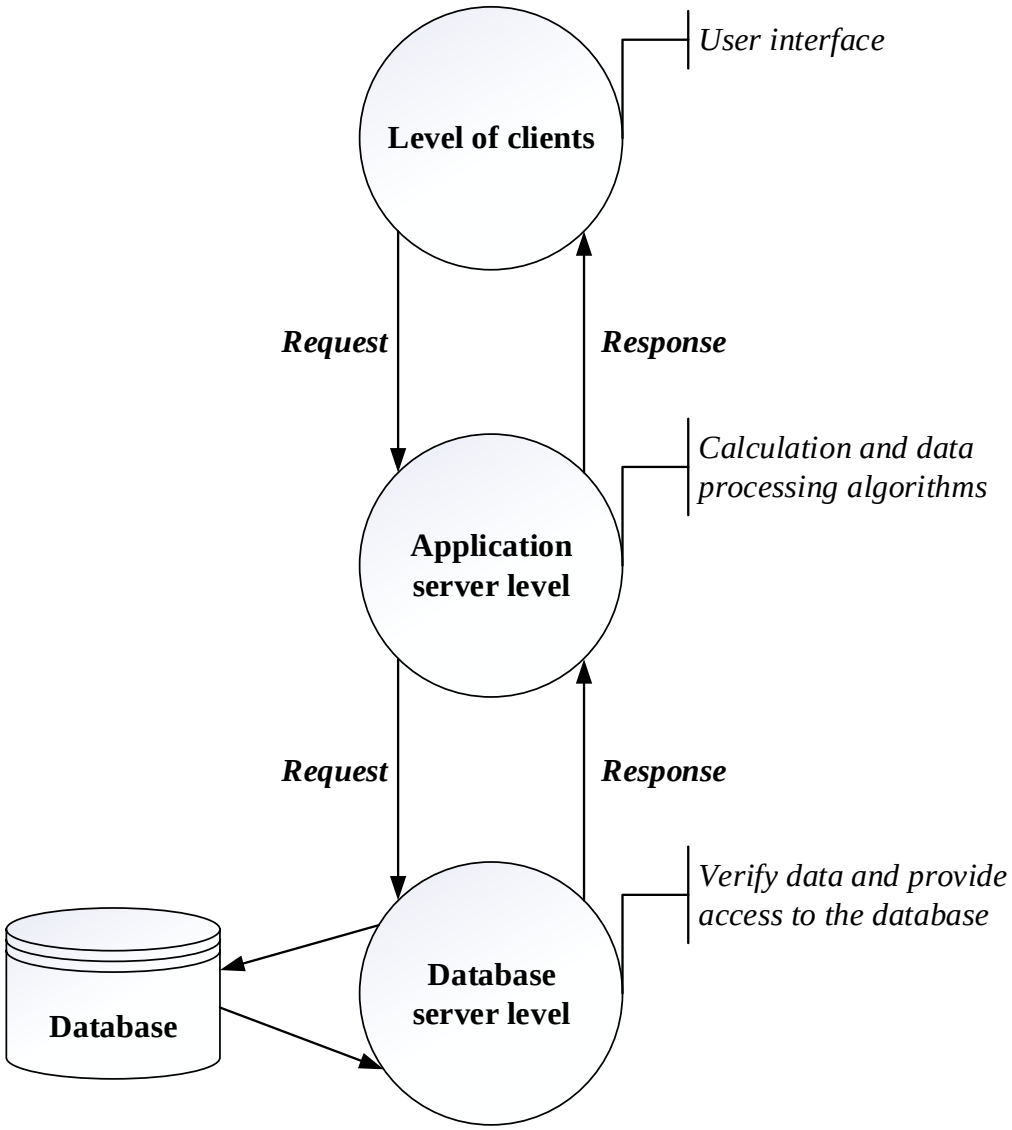


Fig. 4.3. Generalised functional diagram of the three-tier client-server architecture

A characteristic feature of the three-tier architecture is that several servers can simultaneously process a request from one client. This distribution of operations reduces the load on the server. Compared to the two-tier architecture, the three-tier architecture is more complex, as all requests are received and serviced by several servers, but this is why it is more reliable. Because of the fact that all data processing functions in the three-tier architecture are distributed among servers, it has the following advantages: a high degree of adaptability, scalability and performance; and a high level of information security. The disadvantages of this architecture include the complexity of the structural and algorithmic organisation because of the presence of intermediate hardware and software for cross-level information communication.

The multi-tier client-server architecture is a particular variant of horizontal scaling of the three-tiered organisation. This approach can be implemented by installing additional servers at the application level, as shown in Fig. 4.4.

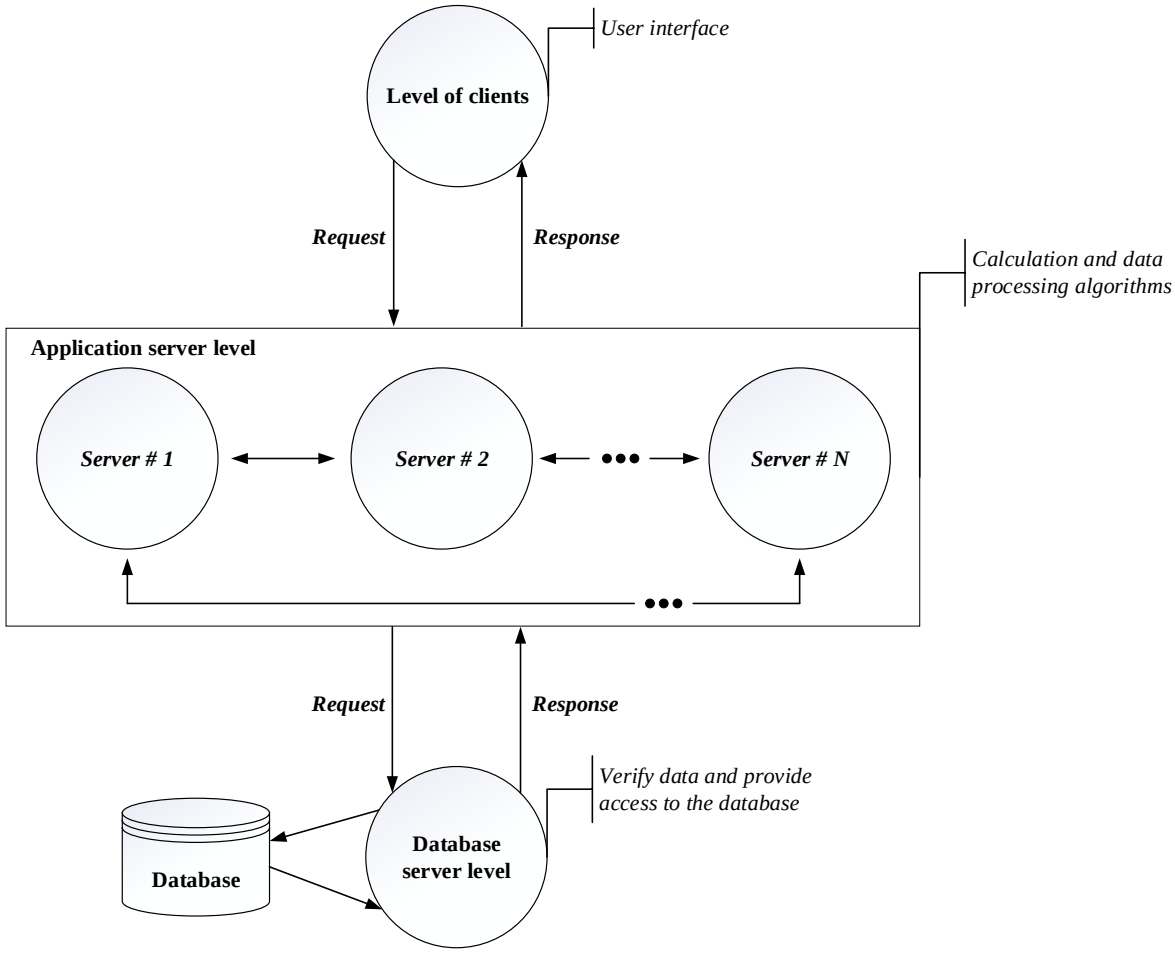


Fig. 4.4. Generalised functional diagram of the multi-tier client-server architecture

The multi-tier client-server architecture can significantly improve the efficiency of infocommunication and computer networks and systems by optimising the distribution of hardware and software computing resources.

In all the above-mentioned architectural solutions of client-server interaction, the server, depending on the practical orientation of IoT technology, can play different roles, as given in Table 4.2.

Table 4.2. The main server roles in the client-server architecture

Server type	The main server roles
Web server	The server that receives HTTP requests from clients (usually web browsers) generates and sends HTTP responses (HTML pages, images, files, streams of media streams and others). Web servers can include both software that performs the functions of a web server and a computer that runs specialised software. Clients access web servers via the URL of the relevant web resource.
Application server	This type of server is developed to run certain application programs. This category also includes the application software that is deployed on the corresponding server.
Database server	This type of server is developed to process user requests in the SQL language. In this case, database control systems are located on the server to which client applications connect.
File server	This type of server is developed to store information in the form of files and provide users with access to it. File servers are also tasked with protecting information from unauthorised access.
Print server	These servers are used to provide and control access to network printers. The main tasks performed by these servers are: controlling printers via web browsers, printing using a URL printer and connecting printers via the Point and Print approach.
Email server	Such servers provide the ability to service user mailboxes, as well as receive and send email from the server.
Remote access server	This type of server provides an entry point to a personal network for remote users. With the help of such servers, routing protocols for WAN and LAN environments can be implemented.

Continuation of Table 4.2

Terminal server	This type of server is a high-powered computer or cluster that is connected to a network with terminal clients. They provide clients with computing resources (disk space, processor time and memory) to solve their respective tasks.
DNS server	The DNS service resolves domain names (FQDNs) to IP addresses.
DHCP server	These servers allow clients to receive their IP addresses on demand and also provide additional data for network configuration: DNS, WINS server addresses, etc.
Streaming media content server	These servers are used to control and deliver multimedia content (video, audio) via the Internet.
Game server	Such servers include those that provide communication between different clients and enable them to communicate within the software of a particular gaming application.

Thus, it can be established that the main idea of the client-server architecture is to divide network applications into several components, each of which implements a specific set of specific functions. Components of such network applications can be stored and executed on different devices when performing client and / or server functions. This approach helps to improve the performance, reliability and security of hardware and software solutions for IoT systems and networks.

4.2 Service-oriented Architecture

When creating large-scale software and hardware solutions for modern distributed information technologies, the use of a vertical structure based on a client-server architecture may not be an effective approach. In this case, it is necessary to use a horizontally distributed organisation in which clients and servers are distributed across multiple computing devices.

Service-oriented programming is a further development of a component

approach to the development and implementation of software systems based on the use of service software with standardised interfaces. Components of software systems can be distributed across different nodes of an infocommunication (computer, telecommunications) network and subsequently utilised as independent, interchangeable or loosely connected application services. The interface of software system components provides encapsulation of the details of the implementation of a particular component from others.

Therefore, service-oriented architecture enables flexible ways of combining and reusing components when creating complex distributed software and hardware systems.

Service-oriented architecture (SOA) is a principle of creating applications that allows a software and hardware system to be represented as a set of interconnected services. Service-oriented architecture creates an infocommunication environment for modules that implement the application's computing logic. SOA defines a way to develop, design, integrate and control discrete units of application logic (services) in a computing environment.

SOA-based applications are developed by linking existing services rather than writing new software code. SOA implements the scalability of services, i.e. the ability to add and / or change the relevant services.

When creating SOA applications, a messaging approach is used. Service-oriented architecture involves three main participants: a service registry, a service provider and a service user. In this case, the service provider registers its own services in the registry and the user accesses the registry based on relevant requests. The functional diagram of the interaction of the architectural components of SOA applications is shown in Fig. 4.5.

The approach to the network exchange of information messages is defined in the service description, which is a specification of the service interface used and includes data types, message formats and transport protocols used in the exchange between user and service provider agents. In addition, the service description may contain a pointer to one or more network nodes from which the service is accessed.

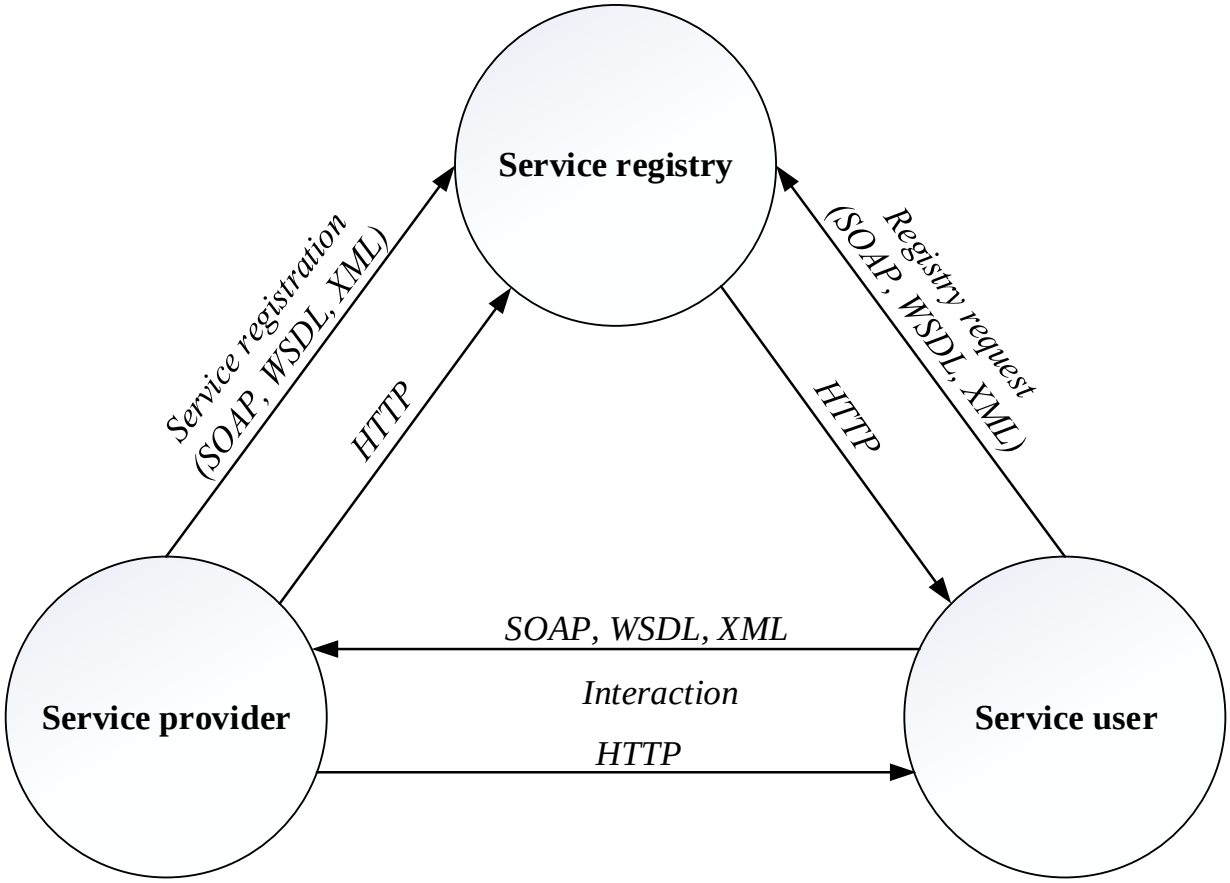


Fig. 4.5. Functional diagram of the interaction of architectural components of SOA applications

When analysing the interaction of SOA application architecture components, it is necessary to note the distinctive characteristics of the following two components:

– service description: defines the format of the request and response during the interaction between the user and the service provider, as well as the required quality of the service;

– service: the service itself, which can be initialised and used by the service user in accordance with the service interface.

The software and hardware implementation of applications based on the service-oriented architecture can be created using the recommended algorithm, the block diagram of which is shown in Figure 4.6. This algorithm contains four main stages (modelling, assembly, deployment and control) which describe its generalised functional underpinnings.

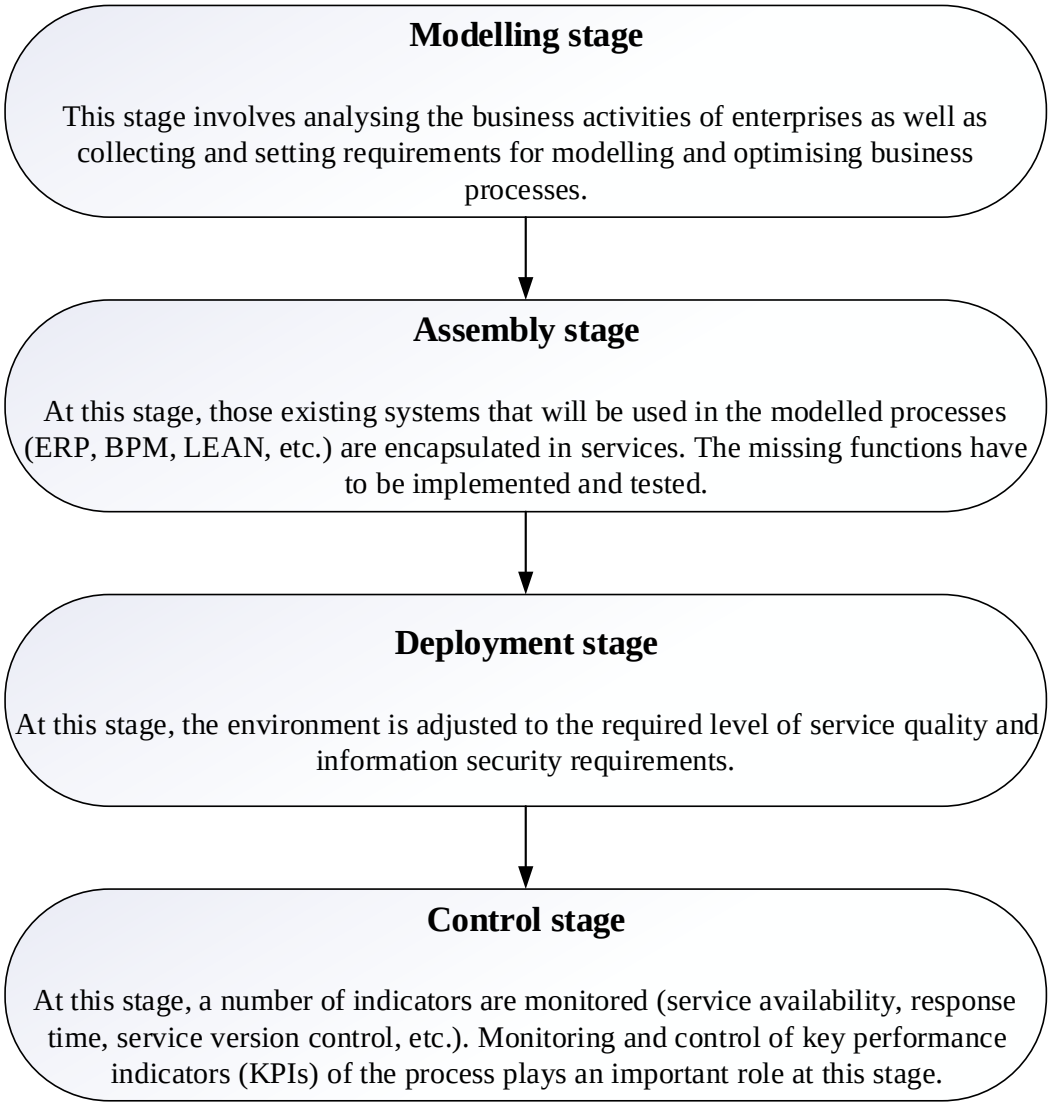


Fig. 4.6. Block diagram of the algorithm for creating SOA applications

The service-oriented approach involves creating a multi-tiered architecture of composite services that are linked to business processes, as shown in Fig. 4.7. Business process flows should be coordinated to combine services into composite and structured applications. To integrate applications, an information messaging system is used to mediate, route and broadcast these services, information flows and their components.

Thus, the following fact can be stated. The evolution of the component architecture of IoT networks is a service-oriented architecture, in particular in the form of web services, which is a modern technology for creating distributed, scalable infocommunication and computer systems.

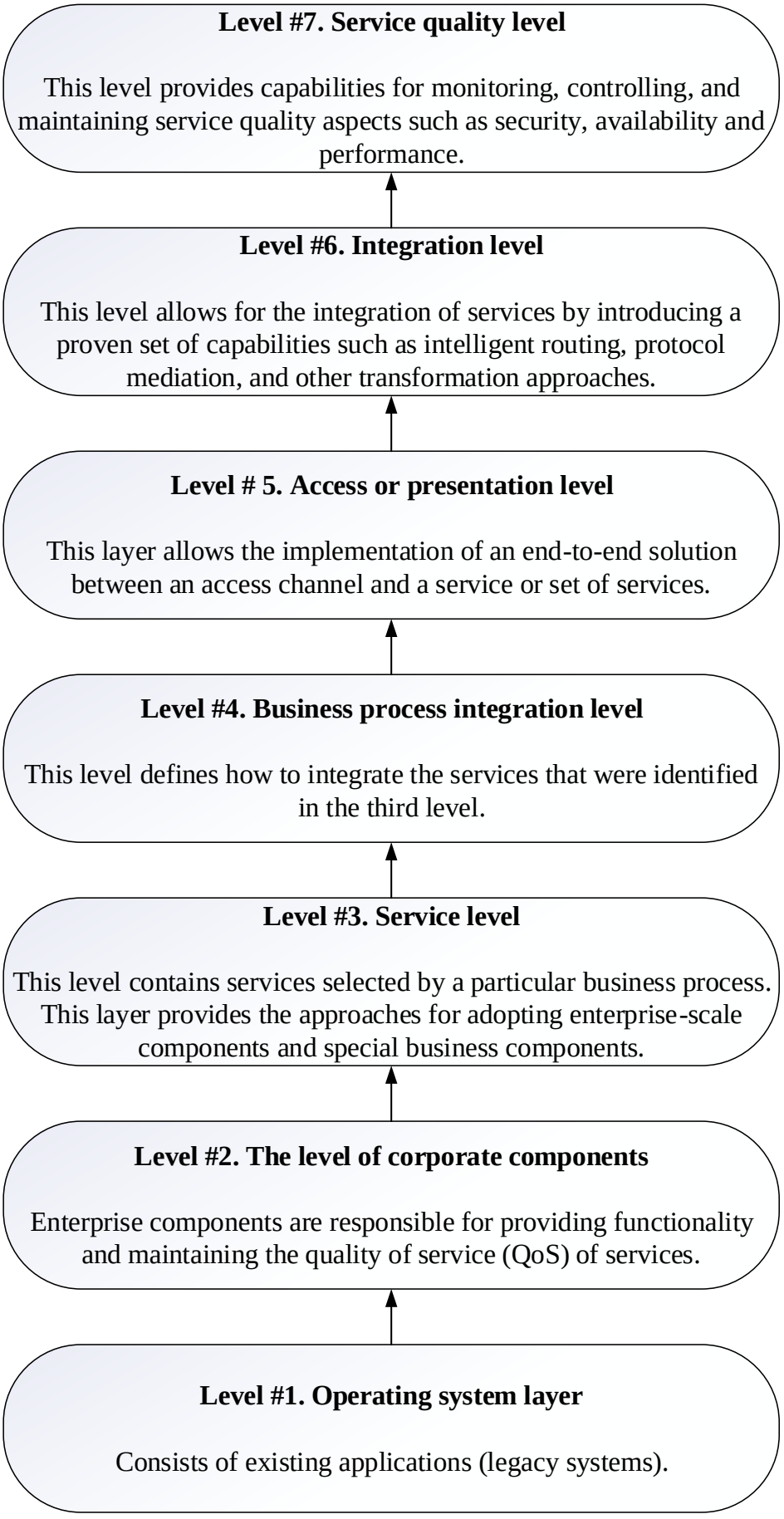


Fig. 4.7. Architecture of multi-tiered SOA applications

The main advantages of this web-based approach include: ensuring interaction between software components and systems regardless of the platform on which they are deployed; the use of open protocols and standards; the ability to ensure interaction between software components and systems through a cross-network interface. The main disadvantages and limitations of this approach are lower performance and higher network traffic compared to CORBA, RMI and DCOM technologies because of the use of textual XML messages.

4.3 Network Protocols

The network exchange of data and information in computer and information technologies is implemented using specialised network protocols. Most IoT systems and networks are based on the operation of several protocols at different hierarchical levels.

A network protocol is a set of system settings that define, implement and regulate the process of information exchange between network devices. The main characteristics of network protocols include standardisation and structuredness.

Today, there are a significant number of network protocols designed to organise and implement the exchange of information messages at different levels, as shown in Table 4.3. In this table, the types of protocols are presented in a hierarchical order, i.e. in accordance with the actions that are implemented at lower levels and subsequently affect the algorithms implemented at new higher levels.

Table 4.3. Classification of the network protocols by levels of use

Level	Principle of operation
Physical	It is a medium in which electrical impulses are converted into binary code and transmitted to higher network levels.
Channel	This level is used to transmit data to the host for processing. MAC address is used to identify information at this functional level.

Continuation of Table 4.3

Network	At this level, IP addresses are updated, which are used to identify users on the Internet. Information is exchanged at this level in the form of data packets.
Transport	This level is aimed at tracking the integrity and correctness of information delivery. Such tasks are implemented using merging or fragmentation algorithms.
Sessional	Protocols at this level ensure that the start and end of a connection are synchronised, that the network session is maintained and that access permissions are checked.
Representative	At this level, the received information is encoded / decoded, files are decompressed / compressed, i.e. data is transformed to a level that meets the requirements of a particular application.
Applied	This layer is utilised to regulate the communication between users and the network, as well as to grant access.

One of the main characteristics that should be taken into account when designing the network level of IoT systems is that the amount of information generated by individual sensor nodes is relatively small, but most IoT applications are based on the aggregation and processing of measurement information from a large number of distributed sensor nodes.

Below are the functional purpose and detailed description of the network protocols that have gained the most popularity in the development of IoT systems and networks of various levels of complexity and application areas.

MQTT. It is a simplified and open protocol for exchanging a sequence of messages containing telemetry data. The main features of this protocol are as follows:

– information messages are exchanged on a publisher-subscriber basis;

– the header size of the information message is 2 bytes and the payload can vary from 1 byte to 260 Mbytes;

– the protocol provides the possibility to select one of three possible service levels.

The block diagram of the MQQT protocol information transfer process is shown in Fig. 4.8.

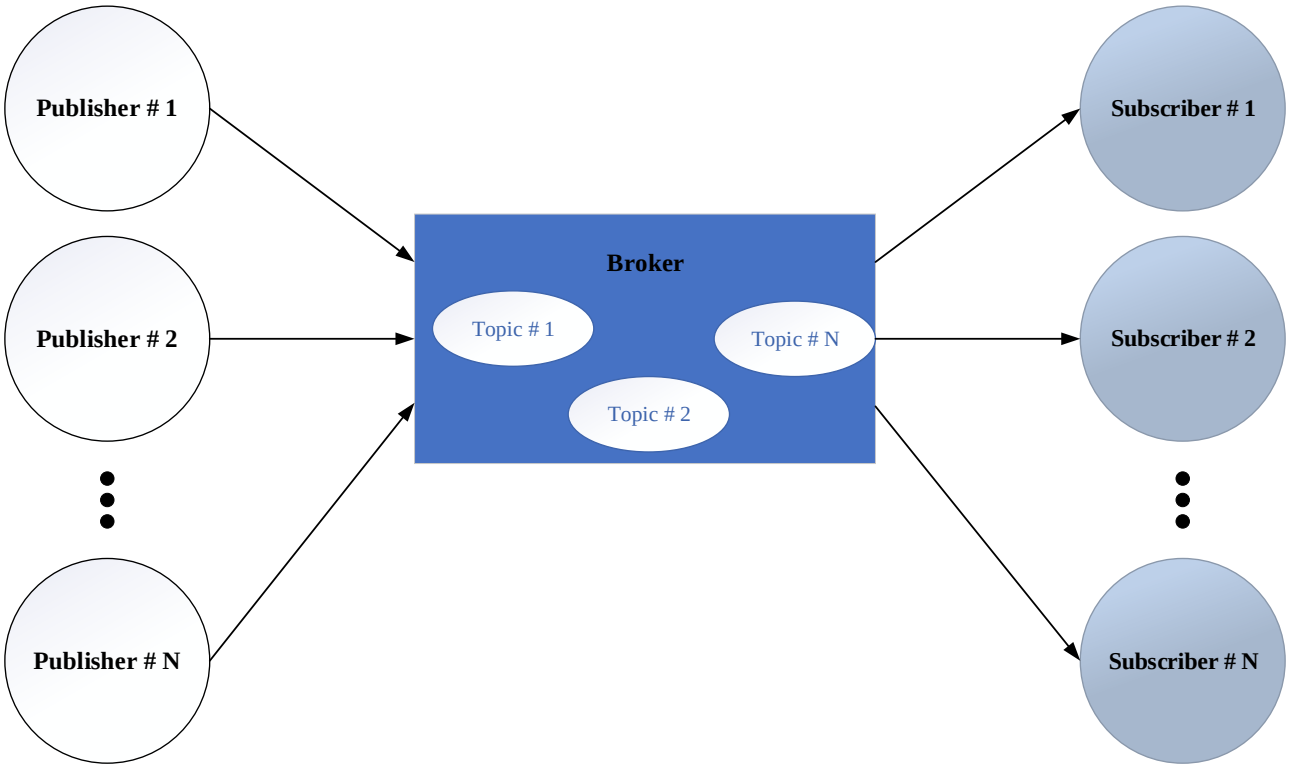


Fig. 4.8. Block diagram of the process of exchanging information messages using the MQTT protocol

In general, the MQTT protocol operates as follows. The publisher transmits an information message with certain data to the broker, indicating the topic to which the data is related. The broker analyses the data to determine which subscribers have subscriptions to the specified topics. The broker sends an information message to those subscribers who have subscribed to the topic specified in the publisher's message. Thus, this approach allows a significant number of subscribers to subscribe to different topics and receive the necessary information without having to contact the publisher directly. Thus, the MQTT protocol can be used in M2M systems. There is also a special version of the MQTT-SN protocol developed for embedded devices based on wireless sensor networks without TCP/IP support.

DDS. This protocol belongs to the M2M type and is used to create real-time

systems, which is similar to MQQT and is based on the publisher-subscriber model. The main task of this protocol is to connect devices to each other via a bus, as shown in Fig. 4.9.

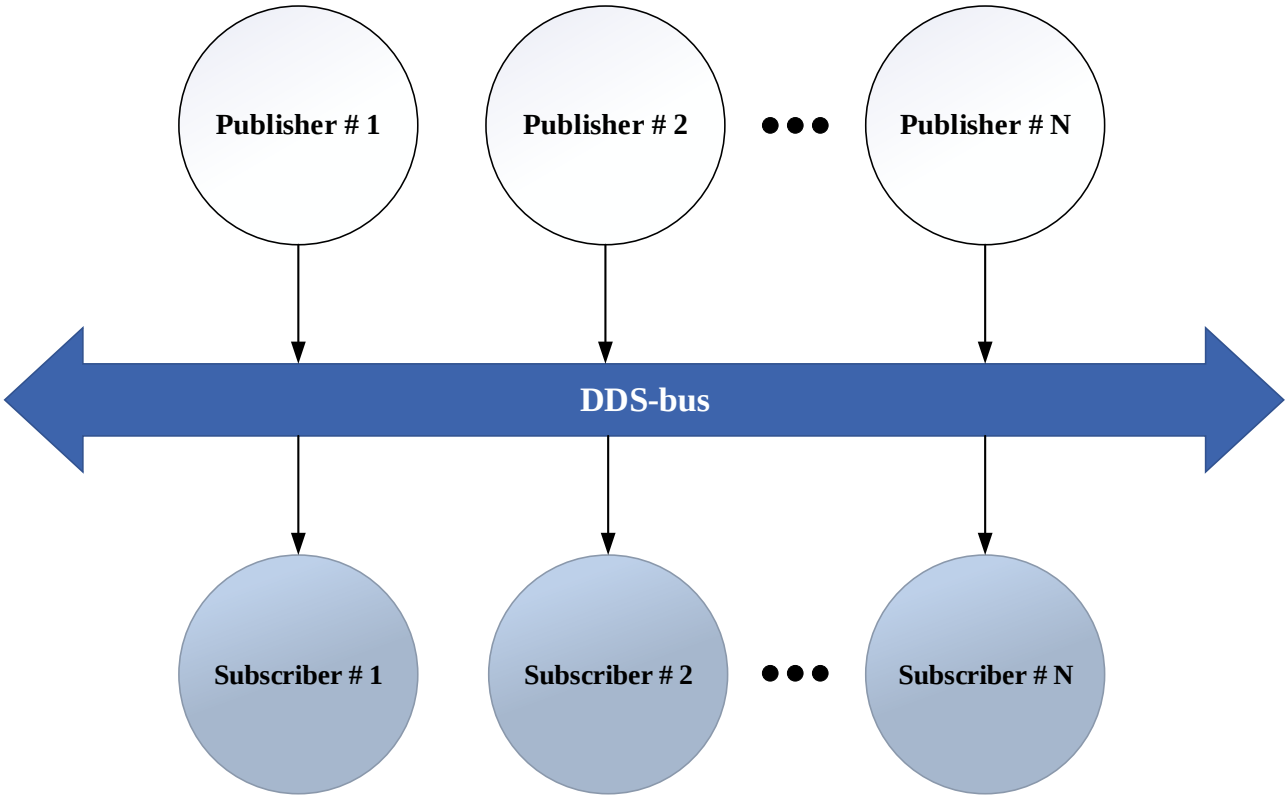


Fig. 4.9. Block diagram of the process of exchanging information messages using the DDS protocol

Devices in most IoT systems need to communicate using complex topologies and thus a simple and reliable two-point TCP/IP communication protocol can be a limiting factor in data transfer. The DDS protocol can efficiently deliver millions of messages per second in a synchronous format while providing satisfactory quality of service control, adaptability, scalability and multicast. The DDS protocol also provides high indicators of filtering and selection of data according to their destination addresses. The hardware implementation of the DDS protocol involves bus communication between devices based on a relational data model.

CoAP. This is a specialised protocol that is designed for use in resource-constrained networks and M2M applications. CoAP can be thought of as a functional

complement to HTTP, but unlike HTTP, the CoAP protocol is designed for use in devices with limited speed and computing resources. A generalised functional diagram showing the principles of the CoAP protocol is shown in Fig. 4.10.

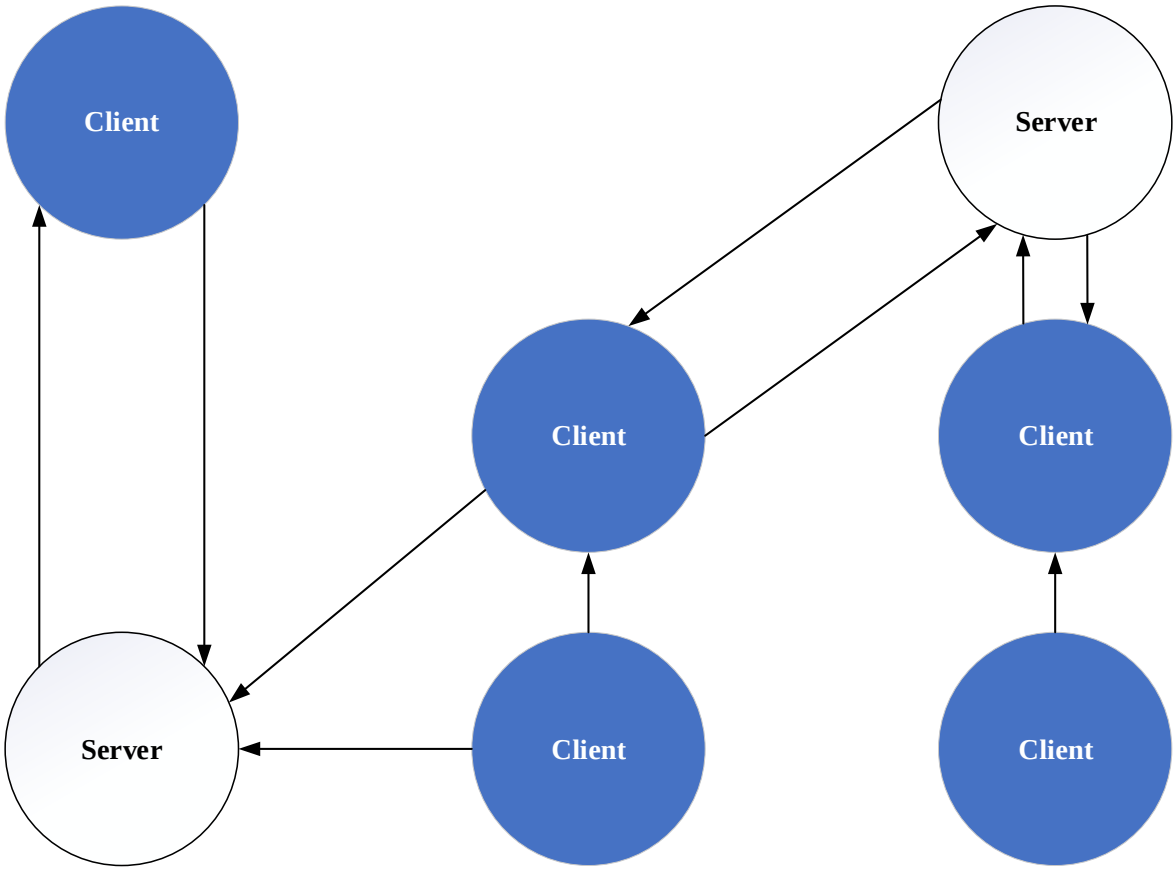


Fig. 4.10. Functional diagram of the CoAP protocol principle

The CoAP protocol is based on the UDP protocol. The range of variations of information messages used by the CoAP protocol is insignificant. Most of them follow the request-response format: CONNECT (establish a connection), PUT (set a new task on the resource), GET (get information about the resource), POST (change actions on the resource) and DELETE (delete active properties of the resource).

Clients use information messages to monitor and control a particular resource. Upon receipt of a request, the corresponding monitoring flag is set and the server begins responding to the request after the initial message has been sent. In IoT systems and networks, this allows servers to stream measurement data from sensors at the physical level.

XMPP. This type of protocol is open and based on XML. Its main purpose is the rapid exchange of information messages over an accessible network in quasi-real time. XMPP supports the network exchange of text, audio and video information. It operates using TCP or HTTP over TCP.

A block diagram of a typical network organisation using the XMPP protocol is shown in Fig. 4.11. It includes clients, servers and a transport level based on a specialised service.

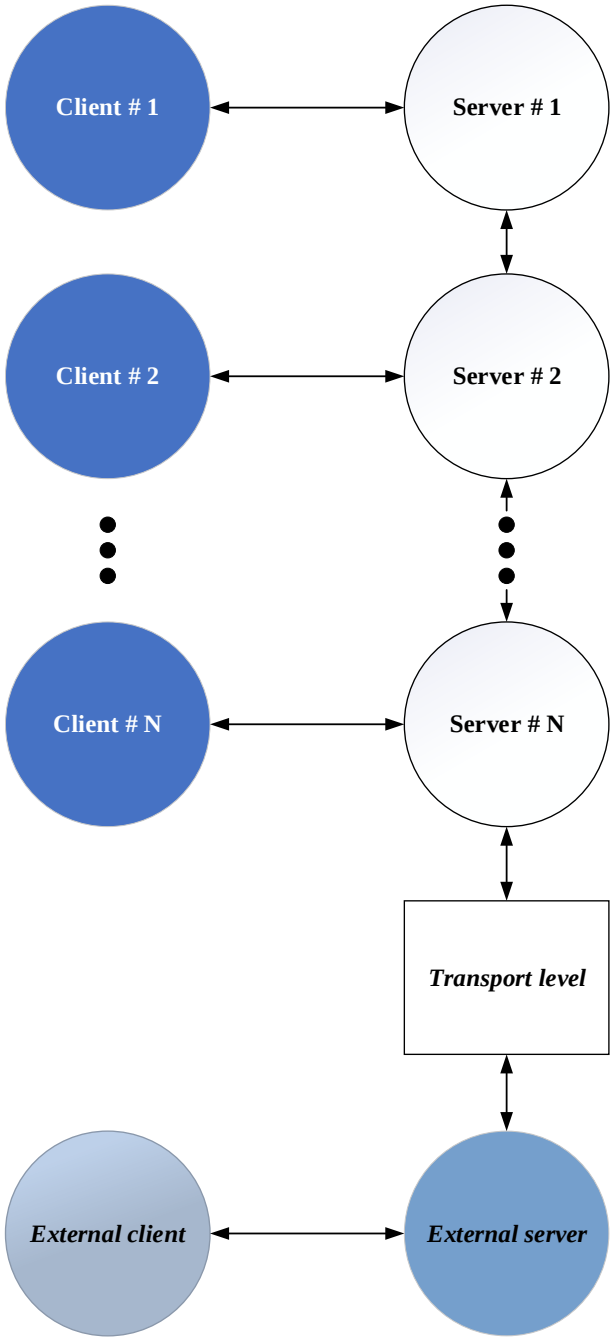


Fig. 4.11. Block diagram of a typical network organisation using the XMPP protocol

The main tasks performed by the server are connection (session) control and routing of network traffic. Most clients are connected directly to servers via a TCP connection and use the XMPP protocol to access the server and the corresponding services associated with it.

It is worth noting that several resources of one client can be connected to the server at the same time. The transport level is a specialised service that runs on the server side and is responsible for converting XMPP to an external network protocol and vice versa. Since in this structure, each server is identified by its own network address and since server-to-server communication is a direct extension of the client-to-server protocol, this system is actually a set of servers interacting with each other.

The main advantages of the XMPP protocol include the addressing method based on the Jabber ID, which contains information about the node, domain and resource. This protocol supports a wide range of infocommunication models, including request-to-response, publisher-to-subscriber and others. The positive aspects of this type of network protocol also include information security and scalability. The main disadvantage of XMPP is the slow speed of data exchange.

SOAP. This protocol is developed for either the structured or unstructured exchange of XML-based information messages in a distributed infocommunication environment. The SOAP protocol operates on a basic connection model that implements a coordinated exchange of messages between a transmitter and a receiver.

This model also allows for the presence of intermediaries that can perform the functions of processing part of the messages or adding additional elements to the messages. A typical structure of a network organisation using the SOAP protocol is shown in Fig. 4.12.

The SOAP protocol supports two possible access approaches: SOAP RPC (a simple request-response protocol using the Call object) and SOAP MESSAGE (a protocol for sending and processing messages using the Message object).

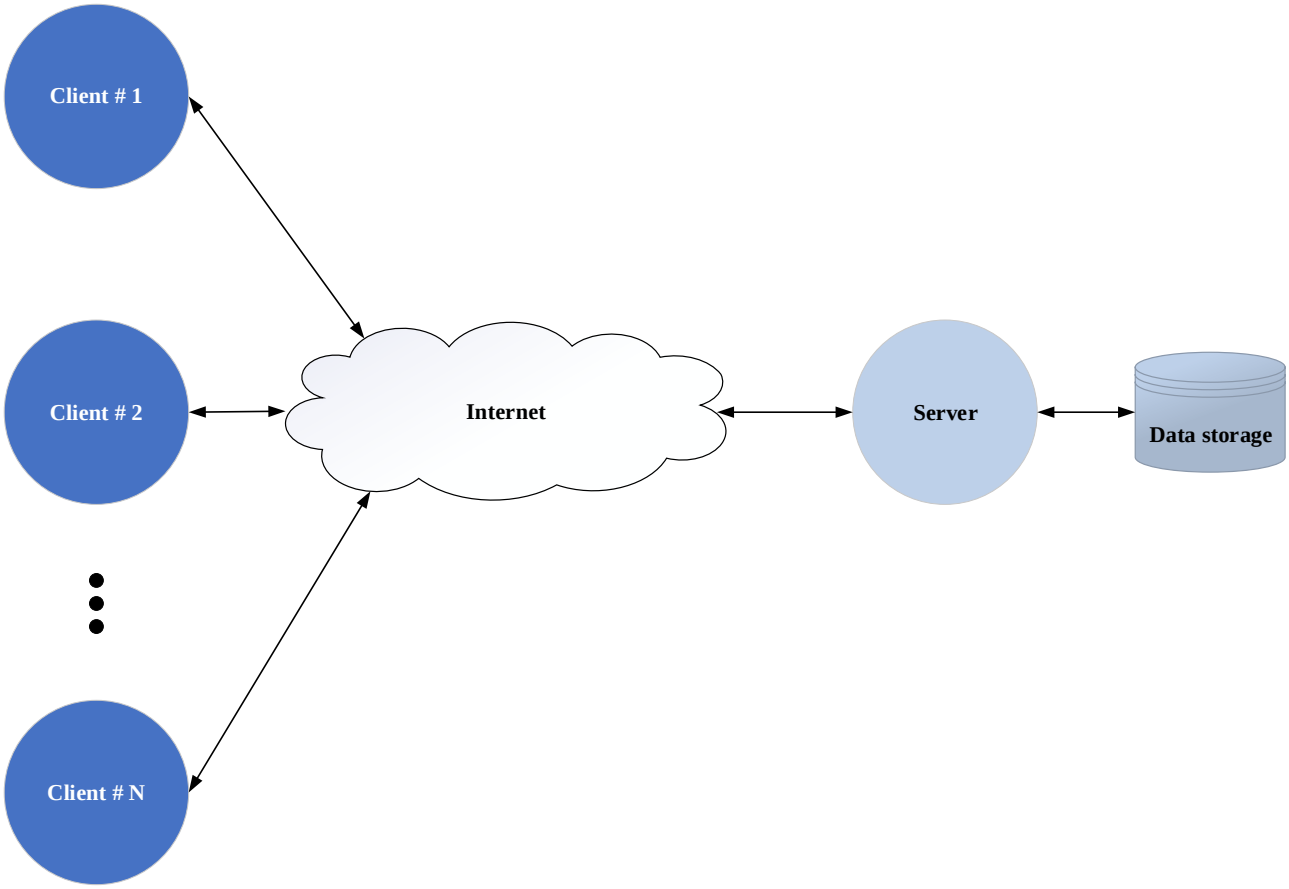


Fig. 4.12. Block diagram of a typical network organisation using the SOAP protocol

Due to the presence of certain types of information messages in the ‘request – response’ category (GET, SOAP ACTION and SOAP ACTION – RESPONSE), this protocol can be used in conjunction with application-level protocols (FTP, HTTPS and SMTP).

STOMP. This protocol belongs to the category of simple publisher-to-subscriber information messaging protocols that support interaction with many infocommunication platforms and programming languages. The STOMP protocol is used when it is necessary to implement the exchange of information messages in networks designed on different types of equipment and platforms. The basic principle of the STOMP protocol is shown in Fig. 4.13.

The STOMP protocol is similar to the MQTT protocol in terms of its principle of operation, except that STOMP implements broker-server interaction, whereas MQTT provides end-to-end publisher-broker-subscriber communication.

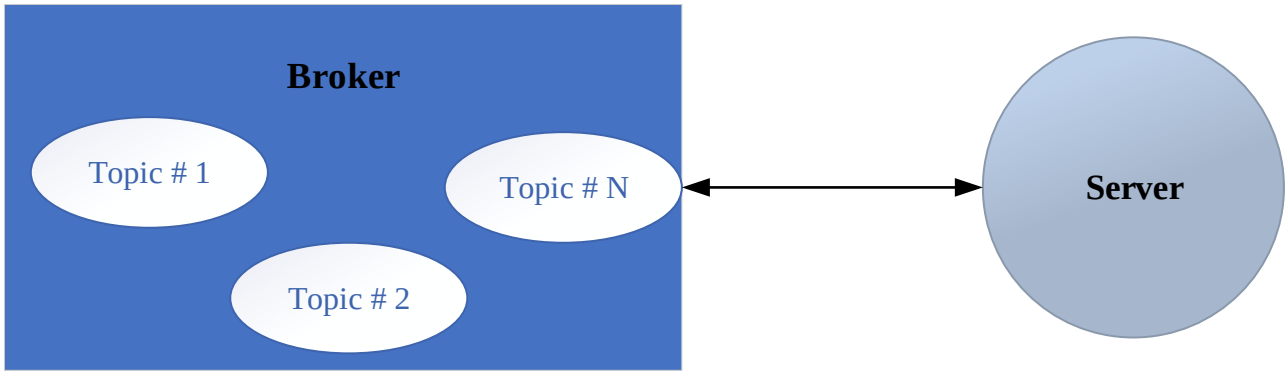


Fig. 4.13. The basic principle of the STOMP protocol

AMQP. It is an open application-level protocol for the exchange of information messages between components of infocommunication systems and networks. The main functionality of this protocol is to ensure the exchange of any number of messages between subsystems (independent software modules) of a network using an AMQP broker. The tasks of the broker are, in turn, routing, possible delivery guarantee and subscription to certain types of messages, as well as data distribution. A typical network organisation using the AMQP protocol is shown in Fig. 4.14.

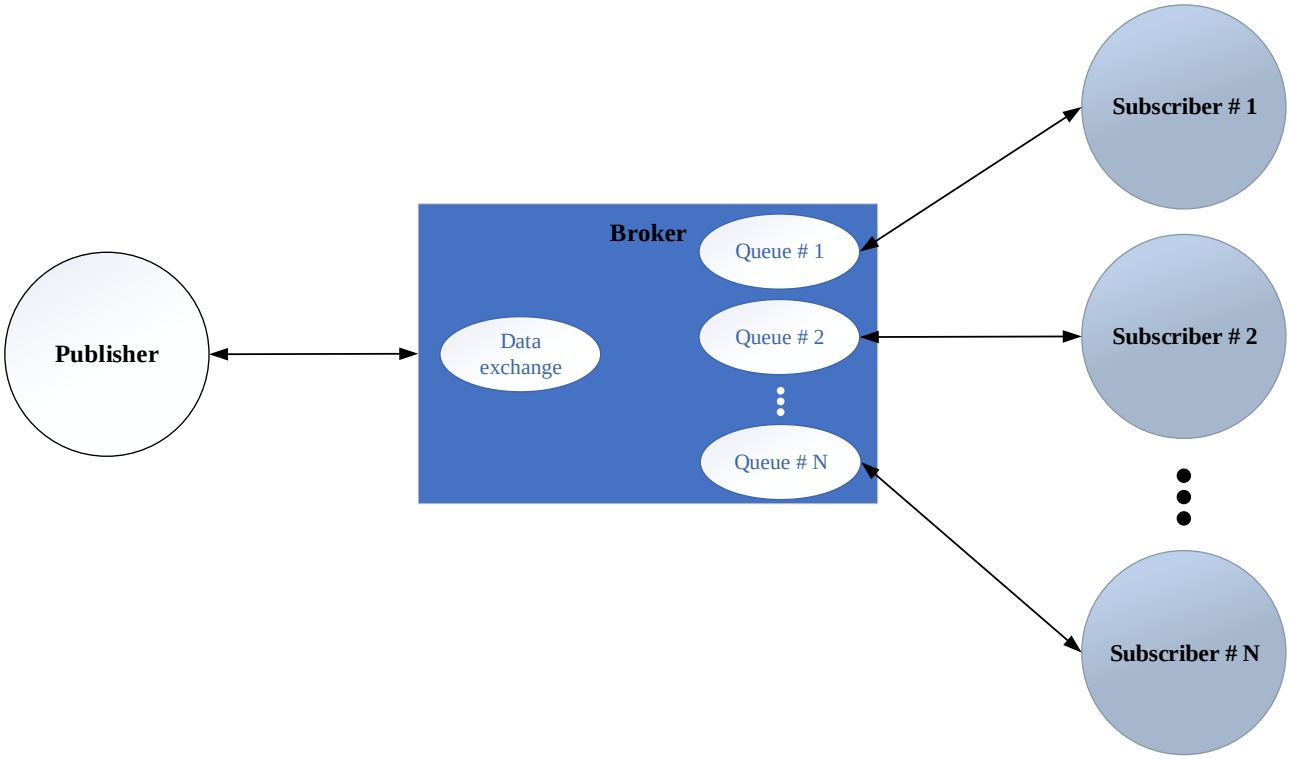


Fig. 4.14. A typical network organisation using the AMQP protocol

The AMQP protocol can be used in a wide range of practical applications where there is a need to ensure high-quality and secure message delivery between applications and processes. This protocol can be used to solve the following networking tasks: monitoring and sharing updates, establishing interaction between different types of systems, ensuring fast server response to requests, simultaneous transmission of information messages to multiple recipients, connecting offline clients for further data retrieval, ensuring system functionality in asynchronous mode and improving application reliability.

The above list of protocols that can be used to create IoT systems and networks is not exhaustive. These protocols have been analysed in order to establish the width of the range of possibilities that can be implemented today in IoT technologies of various scales and applications. A comparative description of the protocols analysed above is given in Table 4.4.

Table 4.4. A comparative characteristic of the protocols used in IoT

Protocol	**Typical areas of application**	**Operations performed**	**Characteristic features**
MQTT	Networks with a significant number of distributed devices and a broker.	Processing of online publications and subscriptions.	Support for queuing and different classes of service quality.
DDS	Networks that require load balancing.	Receiving and transmitting data.	Implementation of direct bus communication based on a relational data model.
CoAP	Networks with limited computing and speed resources.	Receive and record specific indicators and parameters.	It is a simplified binary version of the HTTP protocol.
XMPP	Small personal networks.	Requesting information, receiving and updating data.	It is possible to provide fast asynchronous data exchange.
SOAP	Distributed computing networks.	Parameter request and remote method / function calls.	Support for multiple access approaches and compatibility with different platforms.

Continuation of Table 4.4

STOMP	Networks with the ability to combine several different protocols with the transmission of messages through a broker.	Processing of network publications and subscriptions, as well as transaction operations.	Compatible with a large number of programming languages and infocommunication platforms.
AMQP	Networks that need to ensure high quality and security of data exchange.	Monitoring information messages and sending them to multiple users simultaneously.	Functioning in the asynchronous mode and fast server response to requests, establishing interaction between different systems.

It is worth noting that in addition to the protocols mentioned above, network protocols of various levels are widely used in the development and implementation of IoT systems and networks, which are typical for most modern computer, telecommunications and information networks. These protocols include:

– *TCP / IP* is a set (stack) of data transmission protocols and it is also a designation of the entire network that operates on the basis of two protocols: TCP and IP.

– *TCP* is a protocol used to establish a reliable connection between two network devices, transfer data and acknowledge receipt.

– *IP* is a basic Internet protocol responsible for the correct delivery of information messages to the specified address, in which data is divided into packets that can be delivered in different ways.

– *MAC* is a protocol used to identify network devices.

– *ICMP* is a protocol responsible for network information exchange but is not used for data transmission.

– *UDP* is a protocol that controls the process of information transfer but does not verify the information as it is received.

– *HTTP* is a hypertext transfer protocol used to transfer information to websites.

– *FTP* is a protocol developed to transfer files from a dedicated file server to a

user’s personal device.

– *SSH* is a protocol used to allow remote control of the system over a secure channel.

– POP3 is a standard email connection protocol responsible for delivering email.

– SMTP is a protocol that defines a set of rules for sending email and is responsible for confirming or returning email during delivery.

Thus, when building IoT systems and networks today, a significant number of protocols of different levels and applications can be used. One of the main tasks in developing hardware and software solutions for IoT systems and networks is to choose an effective protocol or combination of protocols that will optimise the performance of the IoT technology being developed in terms of speed, information security and reliability of data exchange.

4.4 Wireless Network Technologies

Today, wireless data exchange technologies are widely used in the development of systems and networks. Wireless technologies are a class of information technologies developed for the wireless transmission of data over a certain distance between two or more objects. Such technologies operate on the basis of appropriate devices implementing appropriate network access points. Data and / or information in such technologies can be transmitted using radio waves, infrared or laser radiation. To date, a significant number of wireless technologies and associated hardware and software solutions have been developed and implemented, for instance: Wi-Fi, ZigBee, Bluetooth, LoRa WAN, NB-IoT, etc. Each of these wireless technologies has certain technical and functional characteristics that determine the scope of its application.

A classification of wireless networks based on coverage area is shown in Fig. 4.15. According to this characteristic, wireless networks can be divided into four categories: WPAN (personal area networks), WLAN (local area networks), WMAN (metropolitan area networks) and WWAN (wide area networks). Also, examples of wireless technologies that fall into the respective categories are shown in Fig. 4.15.

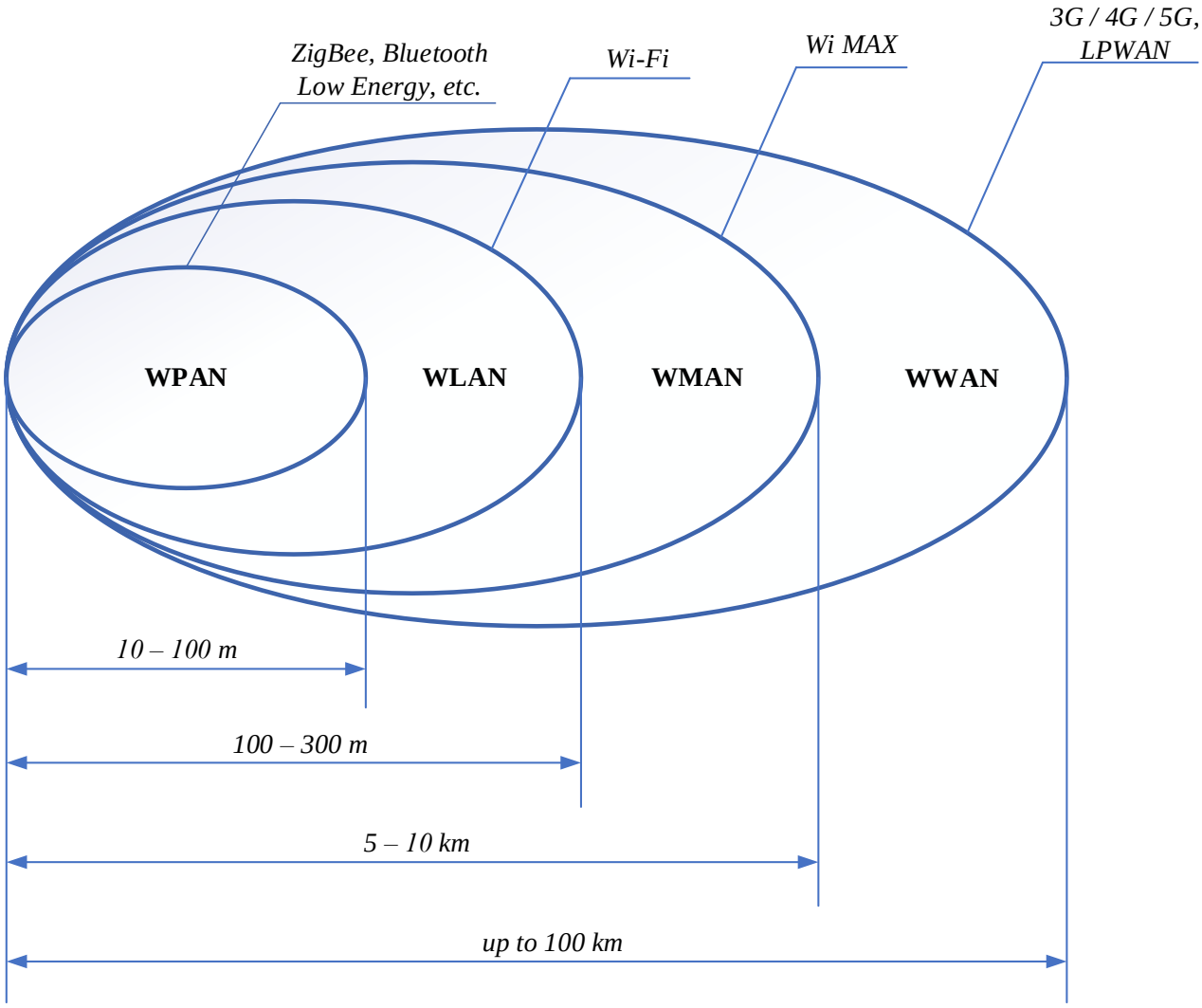


Fig. 4.15. Classification of wireless networks based on coverage area

The key parameters and characteristics of wireless technologies are as follows:

– signal transmission range;

– energy efficiency;

– data transmission rate;

– signal penetration (ability to overcome obstacles: buildings, signal transmission / reception in basements and mines, etc.);

– signal transmission delay (the time between the generation of a signal at the sensor output and the receiving the processed data by the user / operator);

– battery life;

– number of base stations required to cover a certain area;

– base station performance (number of simultaneously supported subscribers);

– subscription fee, connection cost, cost of equipment.

The following are the results of the analysis of the most popular wireless technologies that have been widely used in the development of IoT systems and networks of various sizes and applications.

ZigBee. This technology is a wireless communication standard developed for data collection and control systems. It allows for self-organising and self-healing wireless networks that support automatic message relaying and interaction between mobile nodes, as shown in Fig. 4.16.

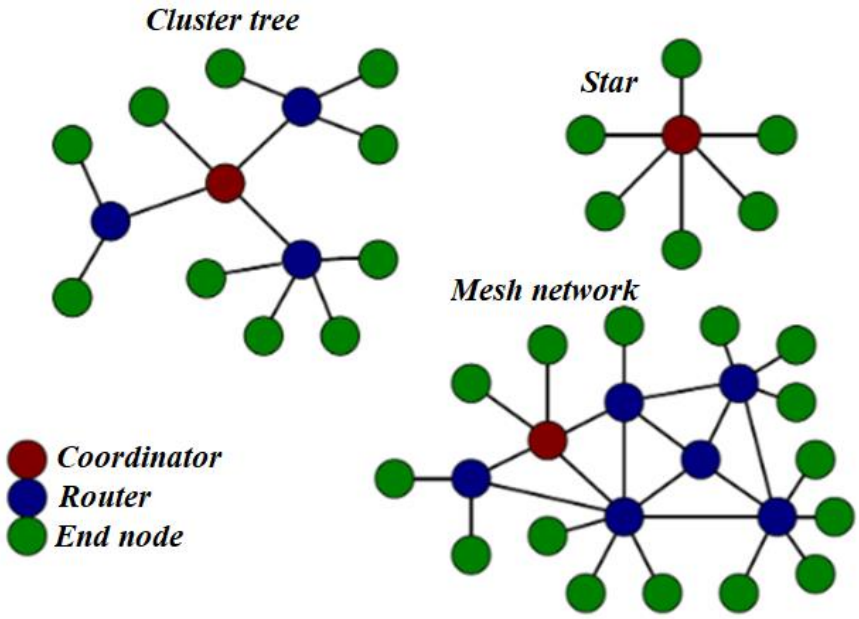


Fig. 4.16. Types of network topologies based on ZigBee technology

The main characteristics of ZigBee technology are as follows:

– support for several network topologies (cluster tree, star, mesh network, etc.);

– low duty cycle, which optimises energy consumption;

– low response delay;

– up to 65 thousand network nodes can be connected;

– 128-bit encryption to ensure a secure network connection.

ZigBee technology provides for operation in the following frequency ranges: 868 MHz, 915 MHz and 2.4 GHz. The highest performance and interference resistance are achieved when using the 2.4 GHz range. This fact determines that most

commercially available microcircuits operate in this range, which includes 16 frequency channels with a 5 MHz step. The maximum data transfer rate declared by most ZigBee chip manufacturers is up to 250 Kbps. The average bandwidth of nodes during the transmission of useful data, depending on the network load and relay capacity, varies from 5 Kbps to 40 Kbps. ZigBee devices can be used in cases where the radio communication range within the line of sight is not sufficiently long and therefore there is a need to increase it when keeping power consumption low. The distance between network nodes can be tens of metres indoors and hundreds of metres outdoors.

The external view of a typical ZigBee module, which is compatible with most commercially available microcontroller platforms, is shown in Fig. 4.17.

Fig. 4.17. The external view of a typical ZigBee module

Today, ZigBee technology is widely used in various areas of the IT industry, but the main segment of this technology is smart homes. Currently, there is no single standard for the hardware and software in the IT industry that will be used in smart homes. The introduction of ZigBee as the main standard can help to solve a number of problems associated with optimising the technical and economic performance of IoT systems and networks intended for implementation in smart homes.

Bluetooth Low Energy (BLE). This technology is an intelligent and cost-effective version of Bluetooth wireless technology that became available for use in the Bluetooth 4.0 version. The main application area of BLE is devices and systems that require low

power consumption, as well as devices that transmit small amounts of data with significant time intervals. BLE is therefore intended for use in low-bandwidth data transmission channels. The main wireless network topologies that can be implemented on the basis of Bluetooth Low Energy technology are shown in Fig. 4.18.

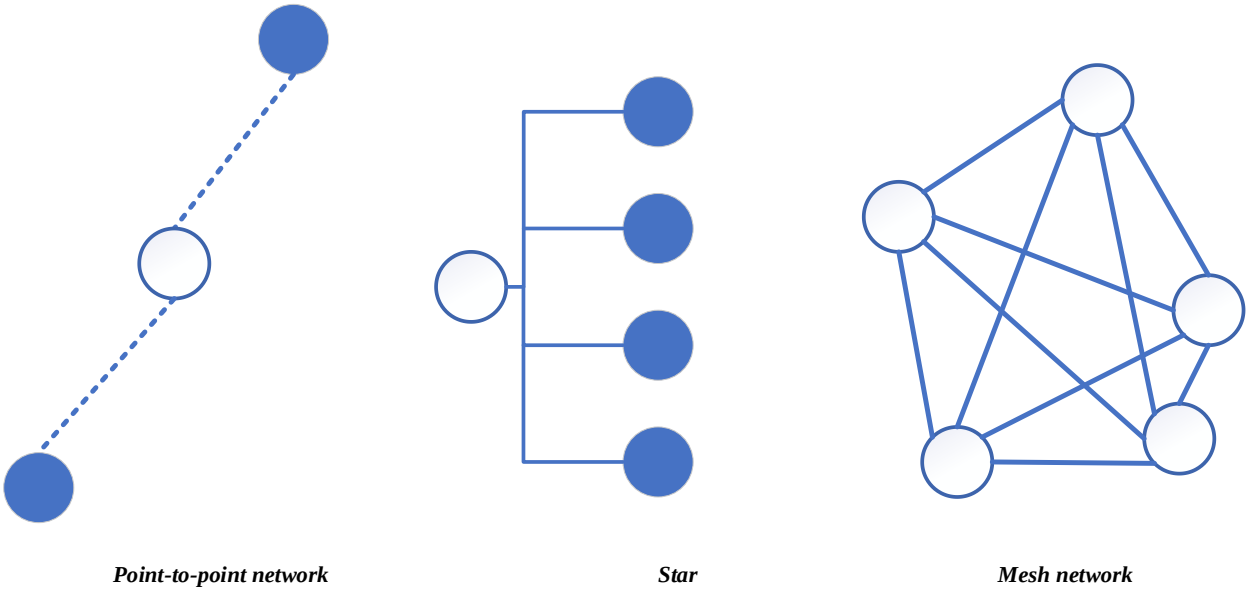


Fig. 4.18. Types of network topologies based on BLE technology

The main technical and functional characteristics of Bluetooth Low Energy are as follows:

– operating frequency range: from 2.4 GHz to 2.4835 GHz;

– the frequency range is divided into 40 channels with a width of 2 MHz;

– maximum data transmission rate over the radio channel: 2 Mbps;

– typical signal transmission range: from 10 m to 30 m;

– current consumption during active mode: 15 mA;

– 128-bit encryption algorithm based on AES-CCM.

The external view of a typical Bluetooth Low Energy module, which is compatible with most commercially available microcontroller platforms, is shown in Fig. 4.19.

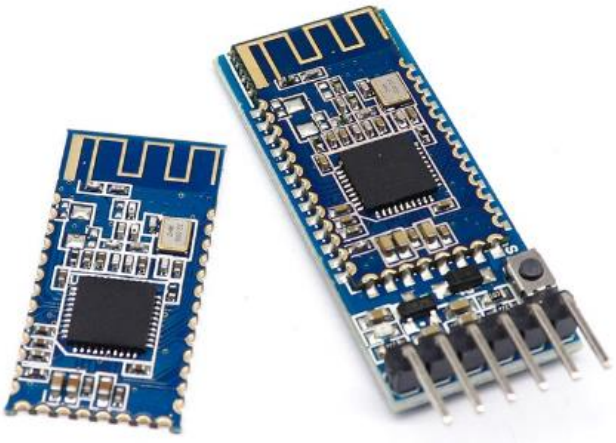


Fig. 4.19. The external view of a typical BLE module

It is worth noting that BLE technology uses a bursty adaptive frequency hopping approach. This enables reliable network communication in noisy environments, such as those found in residential, public and industrial environments.

Wi-Fi. This technology (Radio Ethernet IEEE 802.11) is the first industrial standard that allowed the implementation of a wireless local area network (WLAN) in a limited area where several users have equal access rights to a common data channel. Today, the IEEE 802.11 (Wi-Fi) standard is the basic standard for creating wireless local area networks (WLANs). This standard has been constantly and dynamically improved and therefore there is currently a family of IEEE 802.11 specifications with the corresponding letter indices: a, b, c, d, e, g, h, i, j, k, m, n, o, p, q, r, s, u, v, w. It is worth noting that only five of these categories (a, b, g, i and n) are the main ones and the most popular among network equipment manufacturers, while the rest are additions, improvements or corrections to the adopted specifications.

The main devices that allow organising Wi-Fi networks include:

– a wireless access point, which is used to provide wireless devices with access to the network;

– Wi-Fi adapters are devices that allow wireless signal transmission.

The main limitations of Wi-Fi technology include:

– relatively high power consumption;

– high data security risks;

– the need to disconnect the Wi-Fi connection whenever the server is not in use;

– limited frequency range.

The main advantages of Wi-Fi technology include:

– high scalability and mobility;

– ergonomic configuration of Wi-Fi devices and the network as a whole;

– the ability to connect to the Internet via access points.

Today, the industry produces a significant number of modules that are hardware, structurally and software compatible with serial microcontroller platforms or are complete microprocessor devices themselves. The appearance of such a typical module Wi-Fi module (ESP8266) is shown in Fig. 4.20.

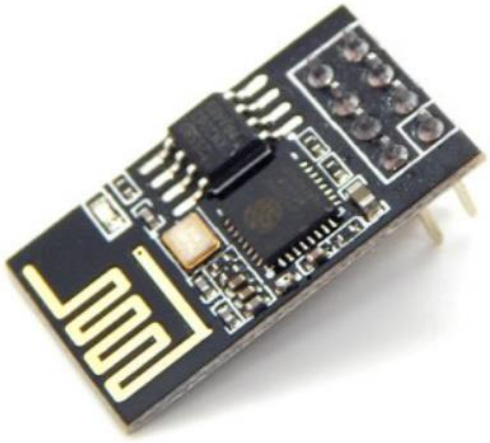


Fig. 4.20. The external view of a typical Wi-Fi module

Wi-Fi technology enables the creation of the following types of wireless networks: episodic network (ad-hoc), basic service area network (BSS) and extended service area network (ESS). Unlike ZigBee and BLE, the main application of Wi-Fi technology is the creation and development of IoT applications that require data transmission over longer distances with the ability to exchange information via the Internet. The results of the comparison of the main parameters and characteristics of the above-described wireless technologies ZigBee, BLE and Wi-Fi are given in Table 4.5.

Table 4.5. The results of the comparison of the main parameters and characteristics of the wireless technologies

Characteristics of the technology (standard)	**ZigBee (IEEE 802.1.5.4)**	**WI-FI (IEEE 802.11b)**	**Bluetooth (BLE) (IEEE 802.15.1)**
Frequency range, GHz		from 2.4 to 2.483	
Bandwidth, Kbps	250	11000	723,1
Protocol stack size, KB	from 32 to 64	more than 1000	more than 250
Continuous battery life, days	from 100 to 1000	from 0.5 to 5	from 1 to 10
Maximum number of nodes in the network	65536	10	7
Operating range, m	from 10 to 100	from 20 to 300	from 10 to 100

It is also worth noting the fact that Wi-Fi technology belongs to the category of WLAN networks, while ZigBee and BLE belong to the category of WPAN (see Fig. 4.15). Therefore, Wi-Fi technology in combination with ZigBee and / or Bluetooth Low Energy allows the implementation of the principles of scaling and adaptation of personal wireless networks and significantly extends their functionality when creating IoT systems and networks.

LPWAN (LoRa, NB-IoT, Sigfox). This is a long-range wireless technology characterised by low power consumption. LPWAN technology combines several significant advantages, in particular: low power consumption, long range and a high level of data security during transmission. Network devices based on LPWAN technology use a highly efficient type of modulation, characterised by high receiver sensitivity (up to –134 dB) and precise data transmission, provided that the informative (carrier) signal is significantly lower than the noise level. Because of these advantages, LPWAN technology is being dynamically implemented in the creation of IoT networks around the world.

It is essential to note that the term LPWAN also refers to a protocol for data transmission in networks with multiple nodes designed in a star or star-of-stars topology, as shown in Figs. 4.21 and 4.22, respectively.

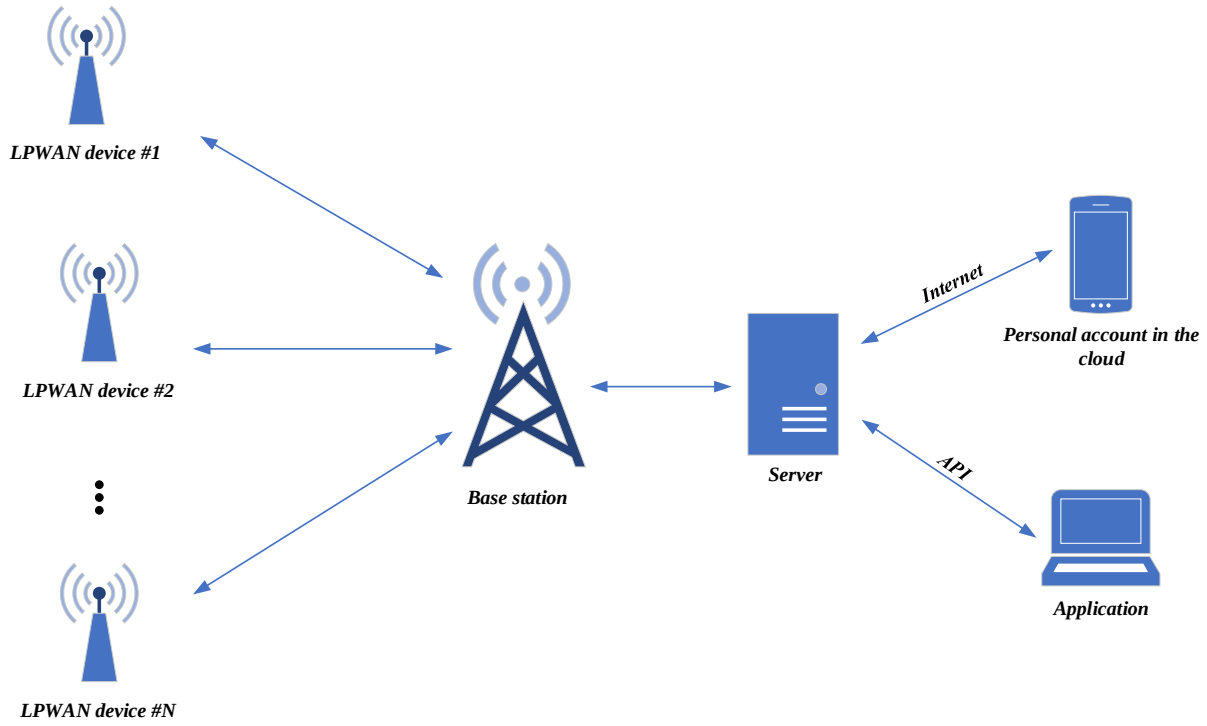


Fig. 4.21. The structure of an LPWAN network using the star topology

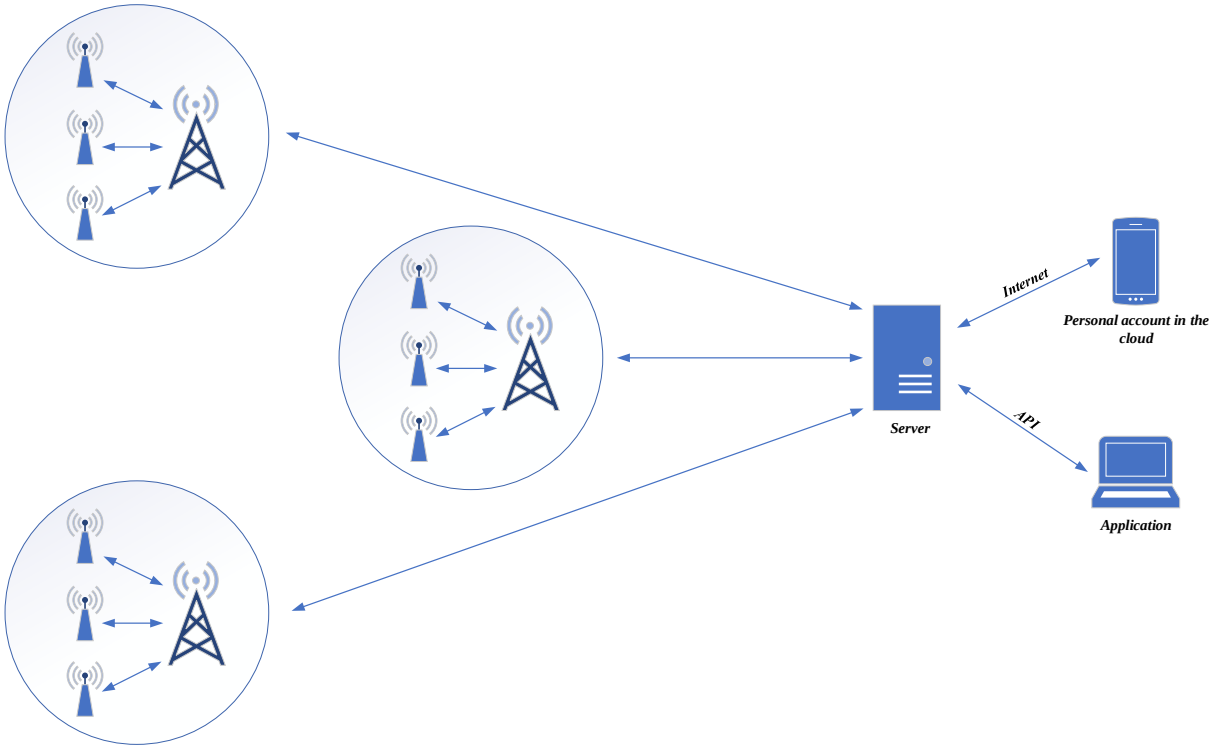


Fig. 4.22. The structure of an LPWAN network based on the ‘star-of-stars’ topology

The following technologies can be used in the hardware and software implementation of LPWAN-based IoT networks: LoRa, NB-IoT and Sigfox. A

comparative description of these technologies is given in Table 4.6 and shown in Fig. 4.23.

Table 4.6. Comparative characteristics of LPWAN technologies

Technology / Characteristic	***LoRa***	***NB-IoT***	***Sigfox***
Modulation method	CSS	QPSK	BPSK
Frequency range	Unlicensed ISM range (Europe – 868 MHz, North America – 915 MHz, Asia – 433 MHz)	Licensed LTE range	Unlicensed ISM range (Europe – 868 MHz, North America – 915 MHz, Asia – 433 MHz)
Bandwidth	125 KHz and 250 KHz	200 KHz	100 Hz
Maximum data transfer rate	50 Kbps	200 Kbps	100 Kbps
Support for bi-directional communication	Yes / half-duplex	Yes / half-duplex	Limited / half-duplex
Maximum number of messages per day	Unlimited	Unlimited	140 (UL), 4 (DL)
Maximum payload volume	243 bytes	1600 bytes	12 bytes (UL), 8 bytes (DL)
Approximate transmission range without data loss	5 km (urban), 20 km (rural)	1 km (urban), 10 km (rural)	10 km (urban), 40 km (rural)
Availability of encryption and authentication	Yes (AES 128 bits)	Yes (AES 128 bits)	Not supported
Support for adaptive data rate	Yes	No	No
Permission for private networks	Yes	No	No
Standardisation	LoRa-Alliance	3GPP	Sigfox in cooperation with ETSI

When implementing DIY (do-it-yourself) projects to create IoT systems and networks, a significant number of modules that are compatible with most microcontroller and microcomputer platforms can be used as wireless data exchange

nodes using LoRa, NB-IoT and Sigfox technologies. The external view of typical LPWAN models of wireless communication modules is shown in Fig. 4.24.

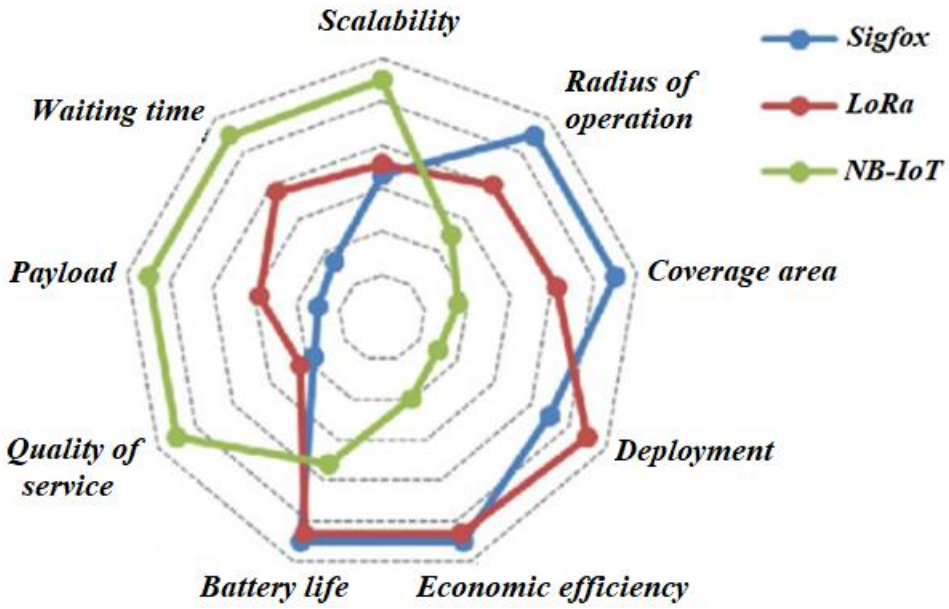


Fig. 4.23.Graphical interpretation of the comparative characteristics of LPWAN technologies

a) LoRa module (model SX1278)

b) NB-IoT module (model SX1278)

c) Sigfox module (model MKR FOX 1200 / ABX00014)

Fig. 4.24. The external view of the serial LPWAN modules

Based on the analysis of the technical and functional characteristics of LoRa, NB-IoT and Sigfox technologies, the range of IoT applications that can be solved using these wireless technologies is quite wide. The most popular areas where hardware and software solutions for IoT systems and networks based on LPWAN technology are dynamically implemented are agriculture, environmental monitoring, smart cities, logistics and others.

4.5 IoT Gateways

In the context of the IoT conceptual framework, gateways are network equipment that operate at the boundary between OT (operational technologies are hardware and software tools for monitoring and controlling real objects and processes) and IT (information technologies are means for generating, aggregating, processing, storing, cyber security, as well as exchanging data and information) (see Fig. 2.15). It is also important to note that the vast majority of IoT gateways implement both primary digital data processing (event processing, filtering and normalisation of data for further transmission to higher levels of IoT systems, as well as online decision support) and intelligent analytics with storage, backup and visualisation.

A significant number of industrial models of IoT gateways are currently known and can be classified according to the following characteristics:

– support and functionality for fog / edge computing: support for a specialised operating system; availability of ready-made applications connected to an IoT gateway for data aggregation and processing at the developer company; the developer has its platforms for implementing user applications; the range of possibilities in terms of programming languages and tools for creating user IoT applications and others;

– the limitations of interaction with network devices: types of supported wired and wireless network protocols; types of supported network interfaces; types of interfaces to external servers and services; support for network self-configuration and others;

– technical parameters: computing power; types and amounts of memory; form

factor and operating conditions.

As previously stated, IoT gateways are more sophisticated than typical network gateways and are designed to perform a range of essential functions, including routing, controlling endpoints, networks and applications, and ensuring the security of devices, networks and applications. A comprehensive enumeration of principal functions exhibited by the majority of commercially available IoT gateway models is shown in Fig. 4.25.

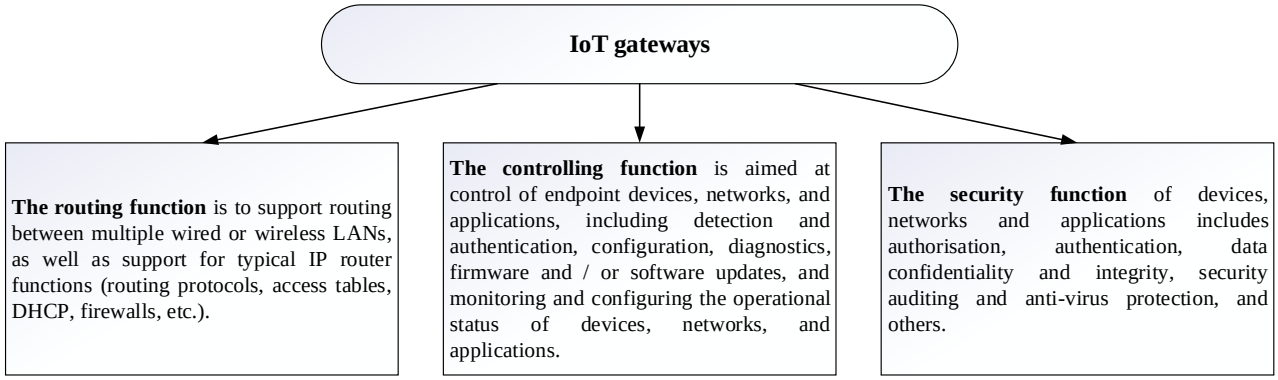


Fig. 4.25 The main functions of IoT gateways

As for now, the network equipment market, including IoT gateways, can be divided into two main segments: consumer and industrial. The main criteria for dividing IoT gateways into these groups are: price, data security, reliability, computing power, application conditions and scalability.

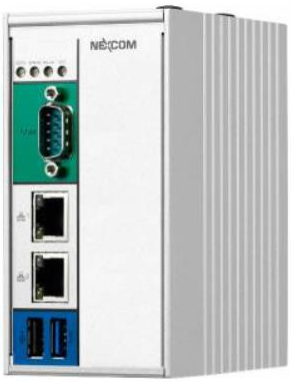
The following subsection provides a generalised overview of the technical and functional capabilities of popular IoT gateway models that have been widely used in the development of IoT systems and networks of various architectures and application areas. The external view of such gateways is shown in Fig. 4.26. As can be seen, the global market leaders in the production of IoT gateways are: Cisco, Dell Technologies, Huawei, Intel IoT, Eurotech and Nexcom. Network devices from these vendors support the full range of functionalities mentioned above and are also manufactured to meet the requirements for high reliability in various conditions, including those that are aggressive and extreme. All of the aforementioned manufacturers offer a range of

universal IoT gateways for utilisation in various industrial areas, as well as highly specialised solutions for certain verticals of the IT services market.

a) IoT gateway by Eurotech (Reliagate 10-12-61 model)

b) IoT gateway based on Intel technologies (UTX-3117 model)

c) IoT gateway by Cisco (ISA-3000 series)



d) IoT gateway by Nexcom (CPS-100-M model)

e) IoT gateway by Huawei (AR-502 series)

f) IoT gateway by Dell (Edge-Gateway-3002 model)

Fig. 4.26. External view of IoT gateways from different manufacturers

IoT gateways by Eurotech. Almost all models of IoT gateways from this manufacturer are developed for industrial applications, including those characterised by aggressive conditions. All models of such gateways are equipped with a significant

number of field buses, as well as wired and wireless data input / output interfaces, such as mobile communication, Fast or Gigabit Ethernet, GPS, ZigBee, Wi-Fi and Bluetooth. Furthermore, Eurotech has developed and implemented its own cloud-based IoT platform, Everyware, which facilitates data control and device administration.

IoT gateway based on Intel technologies. These IoT gateway models are very diverse and developed for a wide range of applications. They can be equipped with Atom, Core, Xeon or Quark processors. The latest generation Atom and Core gateways are capable of delivering high performance, including when working with graphical information and have a wide range of data input / output interfaces. Models IoT gateways based on Intel Xeon are designed to create an infocommunication infrastructure with requirements for increased efficiency and security with the ability to analyse large amounts of data in real-time. IoT gateways based on Intel Quark and powered by Intel Galileo platforms are flexible and low-cost, but relatively low-power solutions for implementing simple computing tasks using embedded devices. Currently, IoT gateways based on Intel technologies are mass-produced by more than a dozen manufacturers. Almost all the models of IoT gateways mentioned above are characterised by significant networking capabilities. These gateways are capable of establishing simultaneous connections to two wired local area networks and several Wi-Fi networks without any disruption to the established connection. A diverse array of supported wireless network interfaces allows such IoT gateways to organise networks based on ZigBee, BLE, Wi-Fi and LPWAN technologies, as well as to connect to cloud services and organise various remote-control scenarios.

IoT gateways by Cisco. To date, Cisco has created a significant number of industrial IoT gateways for various market verticals, including energy, industry, logistics and transport, smart homes and cities, healthcare and others. Cisco's main focus in the Internet of Things is based on the following technological foundations: intelligent data analytics, IT security, support for a wide range of applications, industrial process automation, fog computing and integrated solutions for data transmission to IoT networks. In particular, Cisco serially produces IoT gateways that are compatible with fog and edge computing platforms. This approach allows

customers to create and deploy software directly on Cisco industrial network devices. It is also noteworthy that Cisco specialists have created and implemented support for an open cloud environment to facilitate the deployment of existing and new applications across a range of industries.

IoT gateways by Nexcom. These IoT gateways are characterised by a reliable and ergonomic design that permits their installation in conjunction with programmable logic controllers, sensors and data input / output devices on industrial electrical and telecommunications racks. The principal distinguishing feature of Nexcom's gateways is their focus on achieving high levels of protection against vibrations and blows, as well as the ability to operate in a wide dynamic range of ambient temperatures (from –20 °C to +65 °C). Most of these IoT gateways are capable of aggregating data from networks based on Profibus, Profinet and Ethernet technologies, as well as common wireless communication technologies. They also support cloud-based APIs for interacting with remote servers, such as Microsoft Azure and IBM Bluemix via wireless technologies (such as 3G / 4G and Wi-Fi).

IoT gateways by Huawei. To date, Huawei has developed a wide range of IoT gateways that implement a range of tasks for collecting, transmitting, processing and storing data in IoT networks. Huawei has also created the Huawei Fusion Storage and Fusion Insight cloud platforms, which allow the development and implementation of integrated hardware and software solutions for various applications. Most of Huawei's IoT gateways support a wide range of wireless communication technologies: ZigBee, RF, BLE, Wi-Fi and others. Support for 3G, 4G / LTE and GSM mobile communication technologies is also available. Network traffic routing algorithms can be adaptively configured with different routing scenarios, static routes and support for dynamic protocols (OSPF, RIP, BGP, IS-IS). These IoT gateways also support the conversion of various industry network protocols and the creation of a single network platform. Huawei pays considerable attention to ensuring high design characteristics of the equipment when developing IoT gateways. Most of Huawei's IoT gateway models can operate reliably in highly dusty environments with a wide dynamic range of temperature (from –40 °C to +70 °C) and humidity (from 5 % to 95 %).

IoT gateways by Dell. Dell promotes its IoT gateways (Edge Gateway series) in the industrial networking market as low-cost solutions with increased reliability that implement a range of data collection, peripheral processing and transmission tasks. Almost all models of Dell IoT gateways are designed to meet industrial form-factor requirements. The devices are designed to operate reliably in harsh environments with a wide range of ambient temperature fluctuations (from –30 °C to +70 °C). Furthermore, most Dell IoT gateway models are equipped with a versatile, expandable I/O subsystem that supports interaction with a wide range of interfaces and protocols, such as: Modbus, CANbus, BACNet, LPWAN, ZigBee, BLE, Ethernet and Wi-Fi. These gateways are compatible with the optional Dell Edge Device Manager software, which allows monitoring and control of remote devices.

It is important to note that the above models of IoT gateways from the world's leading manufacturers do not constitute a comprehensive list. The above review and analysis of these models demonstrate the general direction of the development trend, as well as the width and depth of functionality of IoT gateways when creating IoT systems and networks.

Self-assessment Questions on Material Comprehension

1. Define the following terms 'client' and 'server'. Explain the meaning of the term 'client-server architecture'.

2. Give and describe a generalised block diagram of the client-server architecture. Explain the principle of its operation. What functions should be implemented on the client side and which on the server side?

3. Explain the principle of operation of two-, three- and multi-tier client-server architecture.

4. Describe the possible roles of the server in the client-server architecture.

5. Give a block diagram and provide a general description of the service-oriented network architecture.

6. Describe a typical algorithm for creating service-oriented applications.

7. Explain the structure and content of the architecture of multi-tier SOA applications.

8. Define the term ‘network protocol’.

9. Describe the possible levels of network protocols.

10. Describe the functional aim and principle of operation of the following protocols: MQTT, DDS, CoAP, XMPP, SOAP, STOMP and AMQP. Provide a comparative description of these protocols.

11. Define the term ‘wireless technology’. What are the key parameters of wireless technology?

12. Describe wireless technologies by the criterion of coverage area.

13. Explain the principle of operation and functional aim of the following wireless technologies: ZigBee, Bluetooth Low Energy, Wi-Fi, LoRa, Sigfox and NB-IoT.

14. Provide a comparative description of the following wireless LPWAN technologies: LoRa, Sigfox and NB-IoT.

15. Provide structural diagrams and describe the principle of operation LPWAN-networks by topology ‘star’ and ‘star-of-stars’.

16. Describe the principle of operation and functional aim of industrial IoT gateways.

List of Recommended Literature for the Fourth Chapter

1. Kharchenko V.S. (ed.) Internet of Things for Industry and Human Application. In Volumes 1–3. Vol. 1. Fundamentals and Technologies. Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. 605 p.

2. Kharchenko V.S. (ed.). Internet of Things for Industry and Human Application. In Volumes 1–3. Vol. 2. Modelling and Development. Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. 547 p.

3. Zhurakovsky B.Yu., Zeniv I.O. Internet of things technologies. Study guide

[Electronic resource]: education manual for students speciality 126 ‘Information systems and technologies’, specialisation ‘Information support of robotic systems’. Kyiv: KPI named after Igor Sikorskyi, 2021. 271 p. (in Ukrainian).

4. Laktionov I.S., Vovna O.V., Kabanets M.M., Sheina H.O., Getman I.A. Information model of the computer-integrated technology for wireless monitoring of the state of microclimate of industrial agricultural greenhouses. Instrumentation Mesure Metrologie, 2021, Vol. 20 (6). P. 289 – 300.

5. Laktionov I., Diachenko G., Koval V., Yevstratiev M. Computer-Oriented Model for Network Aggregation of Measurement Data in IoT Monitoring of Soil and Climatic Parameters of Agricultural Crop Production Enterprises. Baltic Journal of Modern Computing, 2023, Vol. 11 (3). P. 500–522.

6. QA Test Lab: Client-server architecture. Available at: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (accessed on January 14, 2024).

7. Cross-platform programming and cloud services. Available at: https://victana.lviv.ua/knyhy/konspekty-lektsii/133-kros-platformenne-prohramuvannia-ta-khmarni-servisy?start=10 (accessed on January 16, 2024).

8. MQTT: The Standard for IoT Messaging. Available at: https://mqtt.org/ (accessed on January 17, 2024).

9. Rackley S. Wireless Networking Technology: From Principles to Successful Implementation. Oxford: Elsevier, 2007. 425 p.

CHAPTER 5
DATA AGGREGATION AND PROCESSING TECHNOLOGIES

5.1 Cloud Technologies

Cloud computing represents a model that facilitates ergonomic and ubiquitous access to shared and configurable computing resources (including servers, data storage, applications, IT services, computer networks) provided by a service provider.

The term ‘cloud’ is used to describe an infrastructure of computing services that can be accessed on demand. Clouds are typically large-scale data centres that provide customers with a variety of IT services. Services running in clouds differ from classic local software in their implementation and architecture. Cloud applications are more adaptive and scalable, which means that cloud deployment is very dynamic. A general view of the architectural framework of cloud computing technologies is presented in Fig. 5.1.

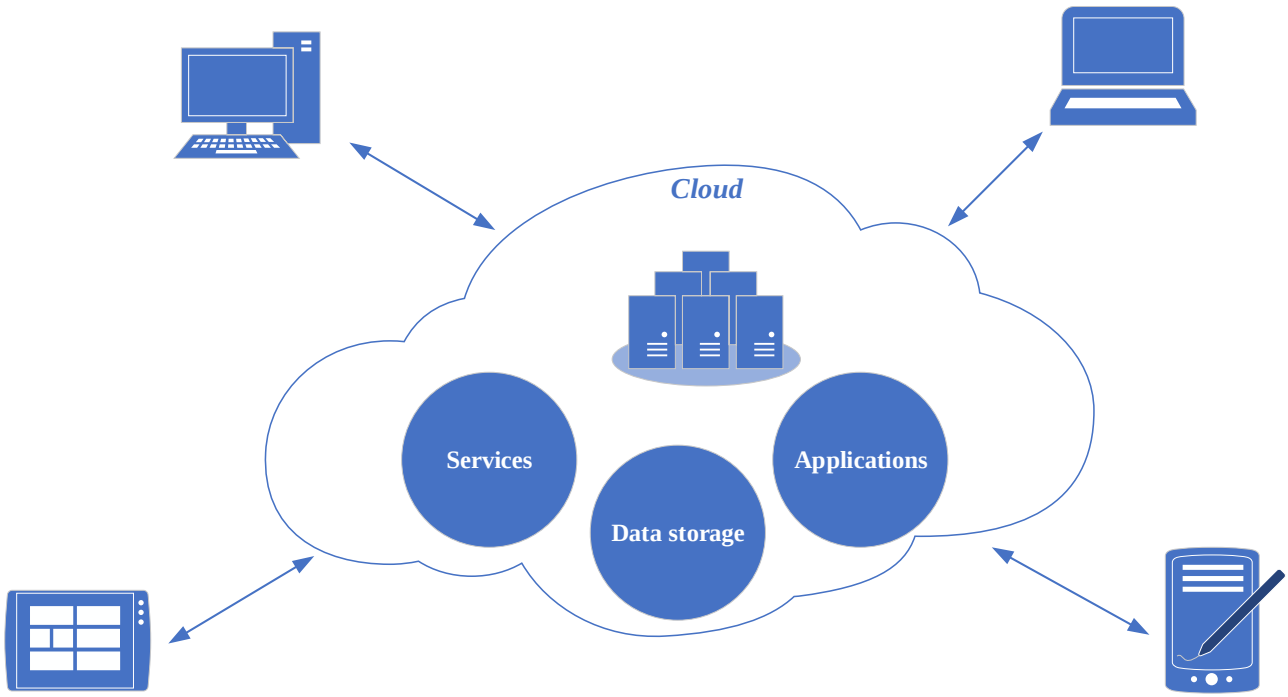


Fig. 5.1. A generalised architecture of cloud computing technologies

The main components of the cloud structure are as follows:

– hardware: servers, client devices and systems, network equipment and storage systems;

– operating systems and system software: virtualisation, automation and resource control tools;

– software that connects distributed resources: software for automated dispatching systems.

The principal advantages of cloud computing are illustrated in Fig. 5.2.

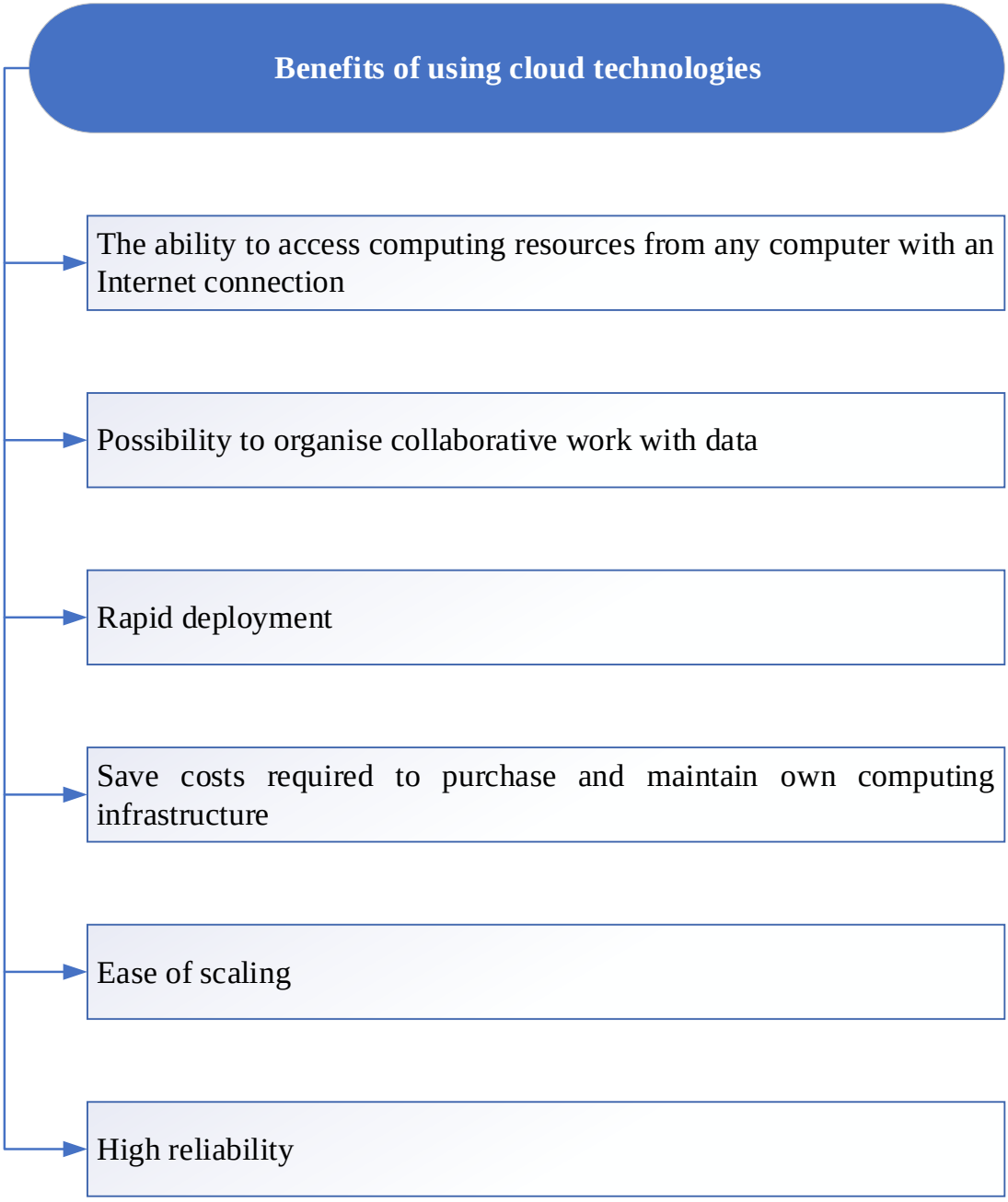


Fig. 5.2. Key benefits of using cloud technologies

So, the key features of cloud services as a modern trend in the development of information technology are the mandatory implementation of the principles of scalability and virtualisation, as illustrated in Fig. 5.3.

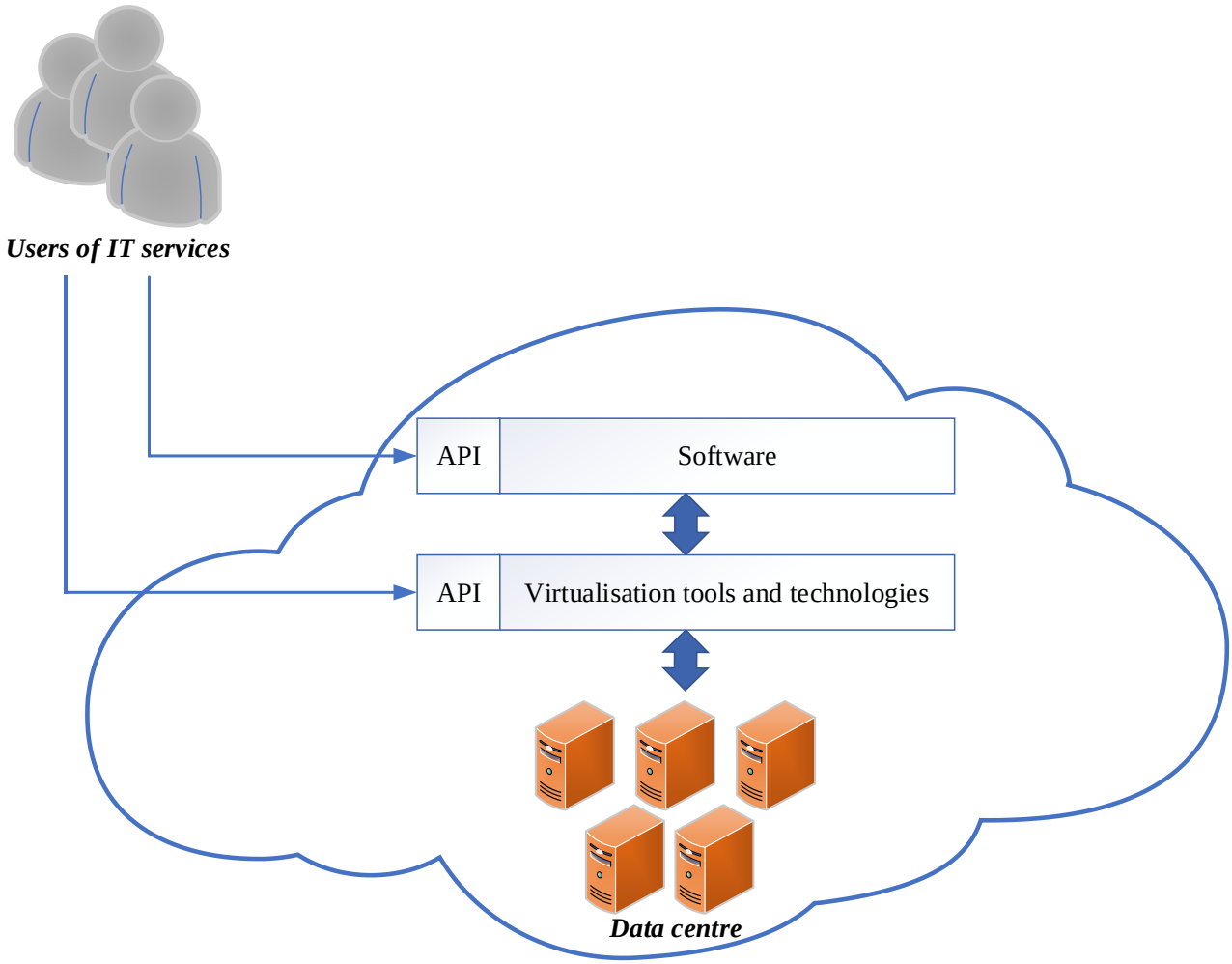


Fig. 5.3. Generalised operation principle of cloud services

In this context, scalability is defined as the capacity to quickly and adaptively reconfigure infocommunication and computing resources in response to fluctuating workload demands. The principles of virtualisation are achieved by meeting the conditions of abstraction and encapsulation. Consequently, the abstraction condition allows for the unification of computing and infocommunication resources, as well as data storage. The fulfilment of the encapsulation condition enables the enhancement of the controllability and security of IoT systems.

The current landscape of cloud services and technologies is vast, comprising

numerous offerings that can be broadly classified into three main categories, as illustrated in Fig. 5.4. The methodology employed in the development of cloud technologies is contingent upon the specific area of work that a company has outsourced, as illustrated in Fig. 5.5.

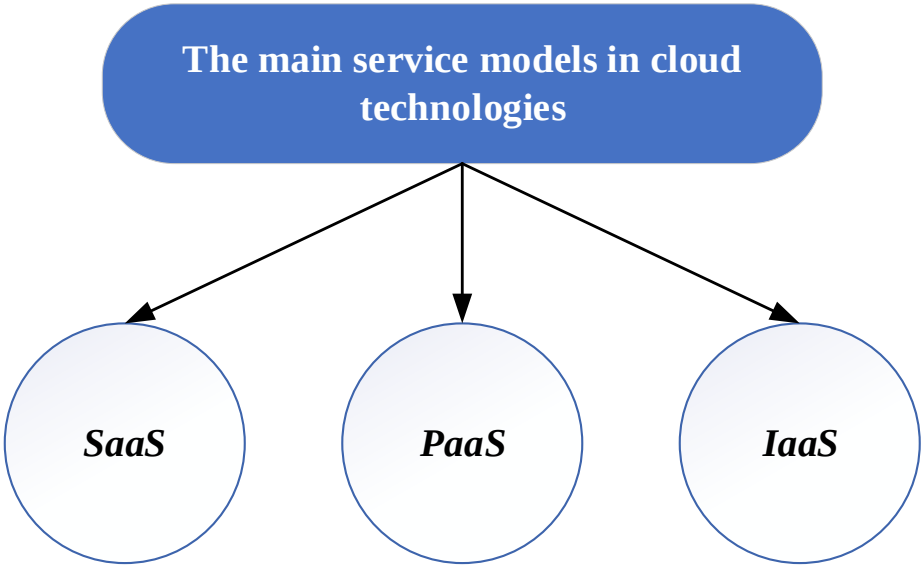


Fig. 5.4. The main service models in cloud technologies

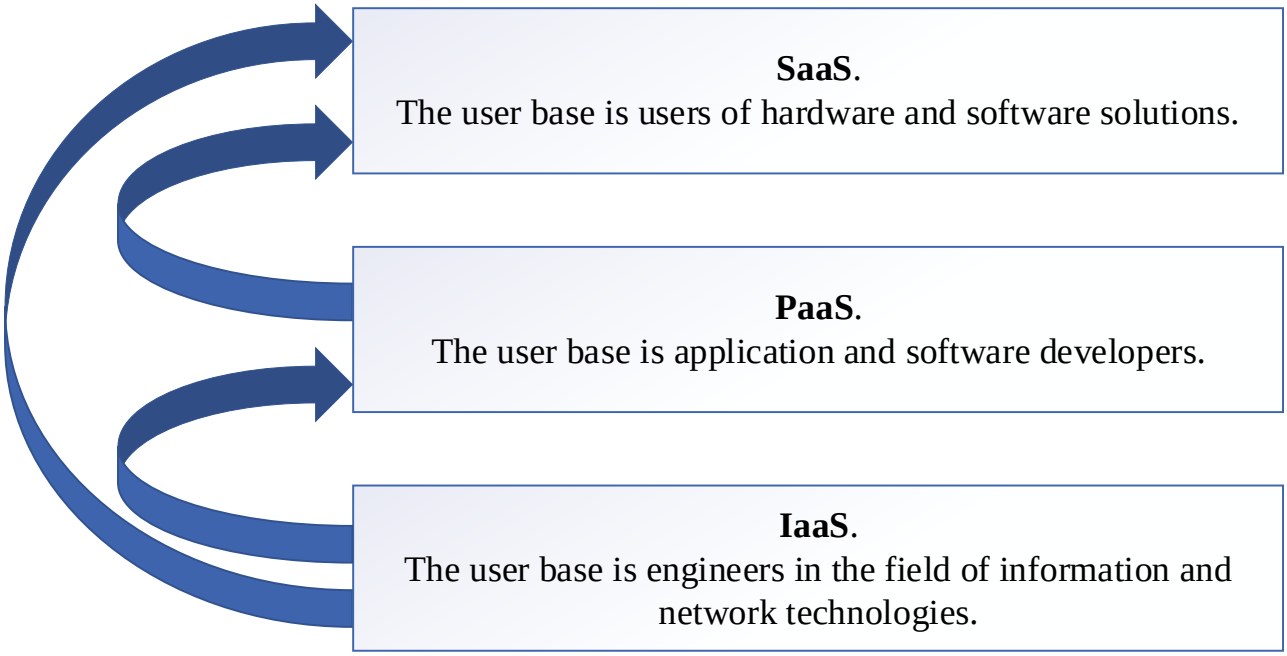


Fig. 5.5. User distribution of cloud technology models

SaaS (Software-as-a-Service). It is a model in which a user is given the

opportunity to use the application software of service providers operating within the cloud infrastructure, accessible via a ‘thin client’ approach across a range of client devices. The main advantages include the fact that the provider solves all technical issues (e.g. updates, performance settings) related to the use of services and applications. The main disadvantages include the following: not all applications and software are available in this format and there are certain restrictions on the customisation of software. One of the most well-known examples of SaaS technology is Google Apps, which includes Google Docs and Google Mail.

PaaS (Platform-as-a-service). The model employs basic lower-level hardware and software provided by the cloud for use. In this case, users are given the opportunity to use the cloud infrastructure to host the basic software and then deploy new or existing applications on it. Such platforms also comprise tools for the creation, testing and deployment of application software. The tasks of monitoring and controlling the real and virtual infrastructure of cloud services are performed by the IT service provider, with the exception of applications developed and installed by users. The key advantages of this model are as follows: high flexibility (there is a potential to synthesise computing systems of any capacity and install all the necessary applications); the ability to use the newest artificial intelligence, big data and other technologies; and the ability to pay for the actual resources used. The principal disadvantages are as follows: the complexity of synthesising the architecture of systems based on virtual components; the basic cost of renting and further using the platform (the level of abstraction between software applications and virtualised infrastructure) is relatively high. Notable examples of this model include Microsoft Azure and Google App Engine.

IaaS (Infrastructure-as-a-service). In this model, users are given the opportunity to utilise the cloud infrastructure to independently control computing resources during the processes of data aggregation, processing, transmission and storage. This approach implies that the IT service provider creates scalable and adaptive hardware services in the cloud, accompanied by the implementation of modifications to software frameworks to facilitate the deployment of client virtual machines. The user is able to

control operating systems, virtual storage systems and installed applications and in some cases, has limited rights to control available network services. The main advantages of this model are as follows: the hardware devices are maintained by the provider; the user has the ability to flexibly adapt the server, operating systems and applications to meet their own needs. The main disadvantage is the high labour intensity of the software configuration and maintenance process.

All three main cloud service models mentioned above are characterised by a nested structure, as shown in Fig. 5.6.

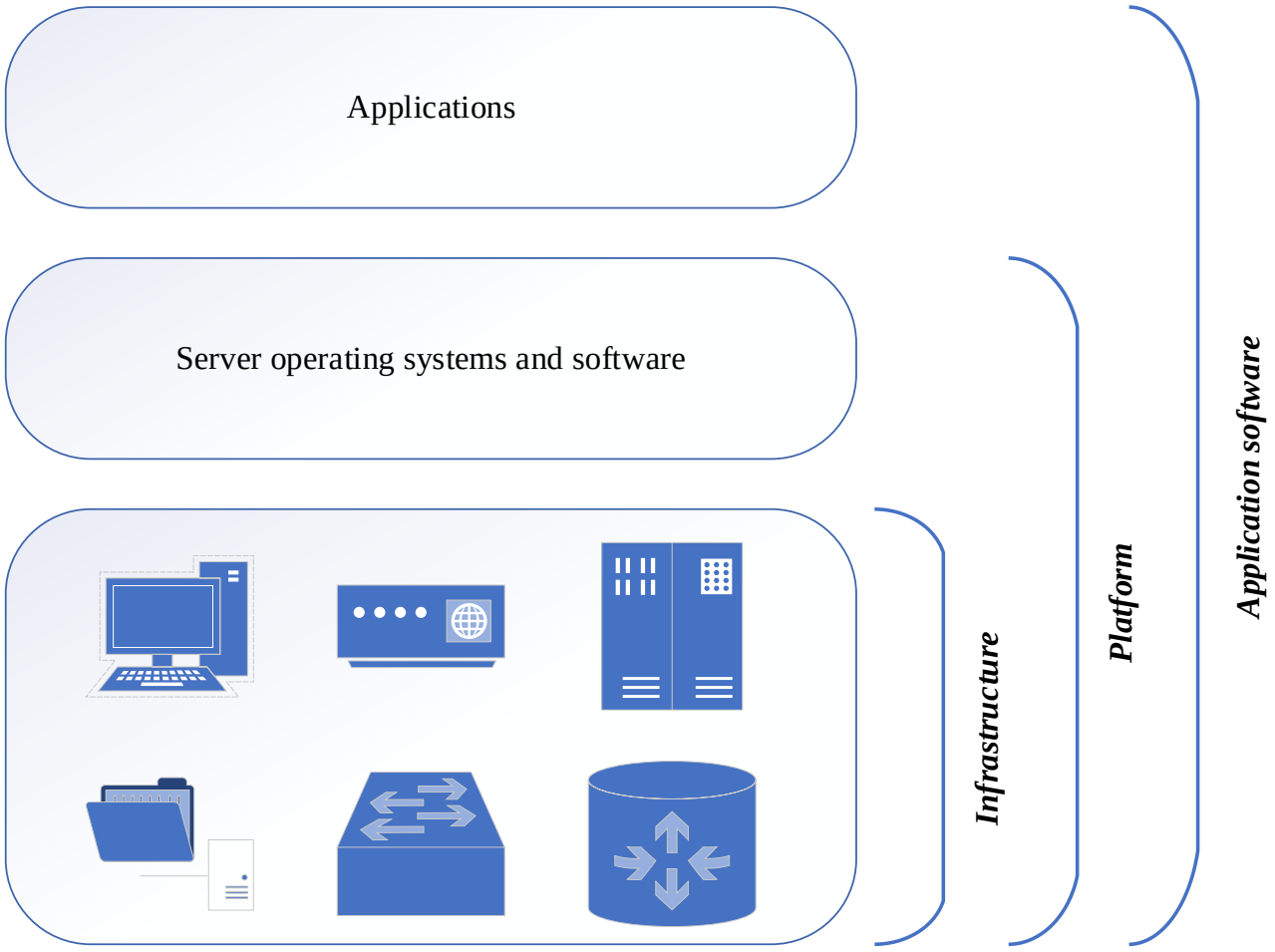


Fig. 5.6. Organisation of the nested structure of cloud service models

It is worth noting that a number of additional approaches to cloud technologies are used when implementing them, as illustrated in Fig. 5.7.

HaaS (Hardware-as-a-Service). In this model, the user gets the opportunity to use the equipment for their own purposes on a lease basis. HaaS is a model that is

analogous to IaaS, with the exception that the user is provided with only the hardware on which to deploy their own software.

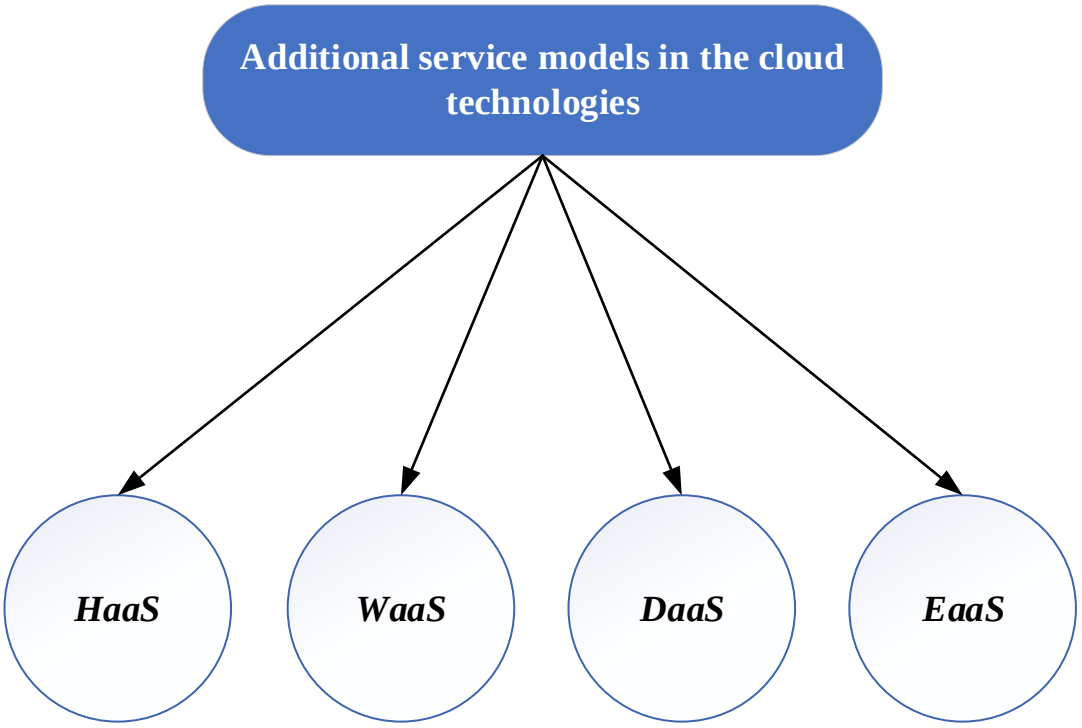


Fig. 5.7. Additional models of cloud technologies

WaaS (Workplace-as-a-Service). In this model, companies use cloud computing technology to organise employee workstations by installing and configuring all the necessary software for the further work of their staff.

DaaS (Data-as-a-Service). This model provides the user with disc space that can be utilised to store large amounts of information.

EaaS (Everything-as-a-Service). This model provides for the integrated provision of IT services, as well as hardware and software resources. Consequently, the EaaS model can be regarded as a generalised concept in relation to all the models previously outlined.

In terms of topology, cloud environments can be classified into three main categories: private, public and hybrid cloud. It is also noteworthy that irrespective of the topological model employed, the frameworks of all clouds should facilitate high rates of deployment efficiency, dynamic scalability and development speed.

Figure 5.8 illustrates the generalised structural diagrams of potential cloud topologies.

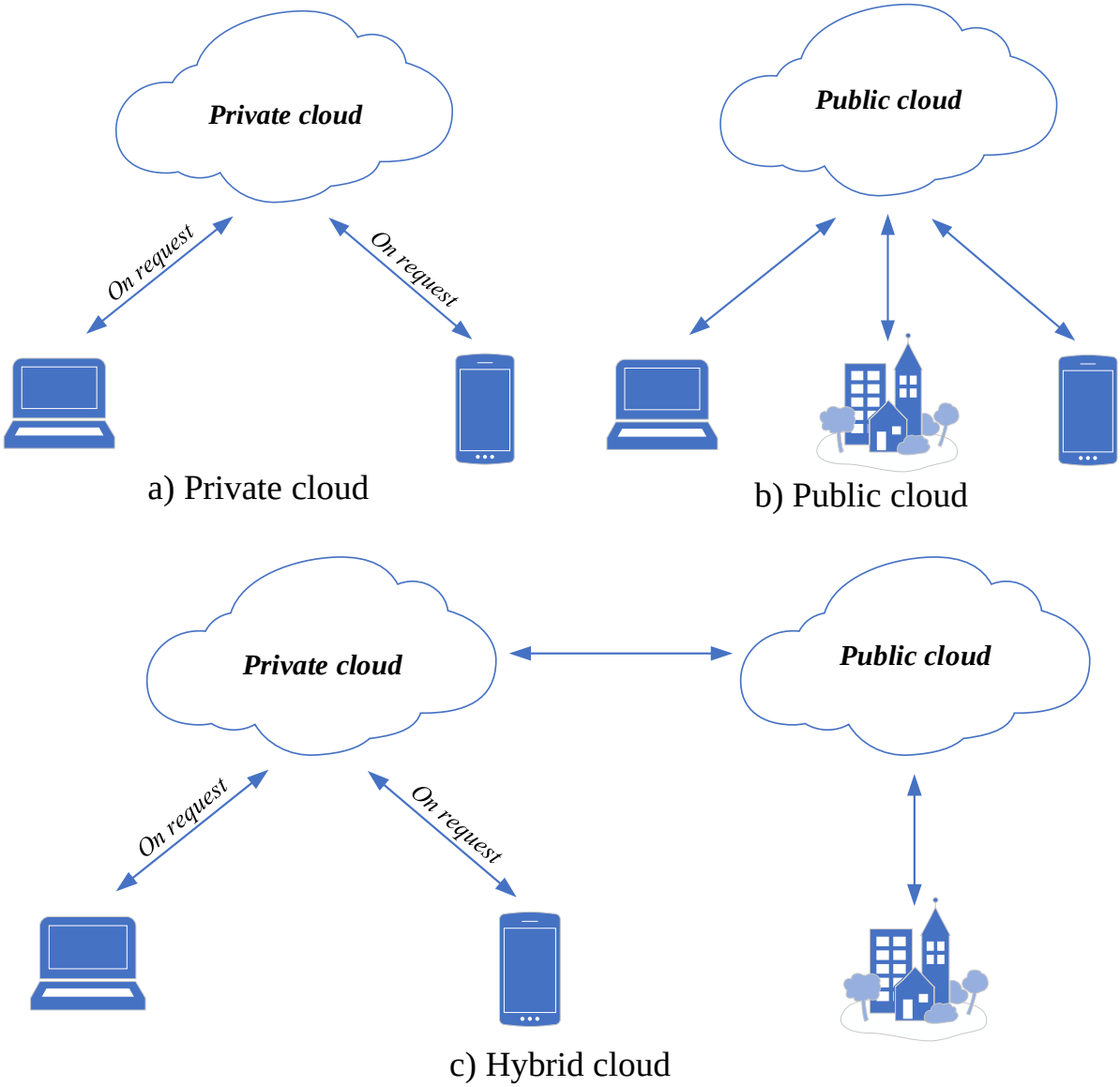


Fig. 5.8. Topological models of cloud services

The core of cloud services is a platform, which is a set of tools and utilities that enable the development, integration and display of cloud applications. A list of the main functional components of cloud service platforms is shown in Fig. 5.9.

Private cloud. This type of cloud service is an infocommunication network infrastructure intended for use by a single organisation, company or department. In this architecture, there is no concept of sharing or combining network and computing

resources outside the private infrastructure of the company (organisation). This approach is appropriate for a number of good reasons, including data security. That is, taking into account the guarantee that data and information are placed exclusively in systems administered by clients. However, in order for such a network organisation to fall under the term ‘cloud’, certain principles of cloud services must be met, namely scalability (adaptive load balancing) and virtualisation. In regard to hardware solutions, private clouds may be either local or created on hardware provided by a third party for the exclusive use of a particular client.

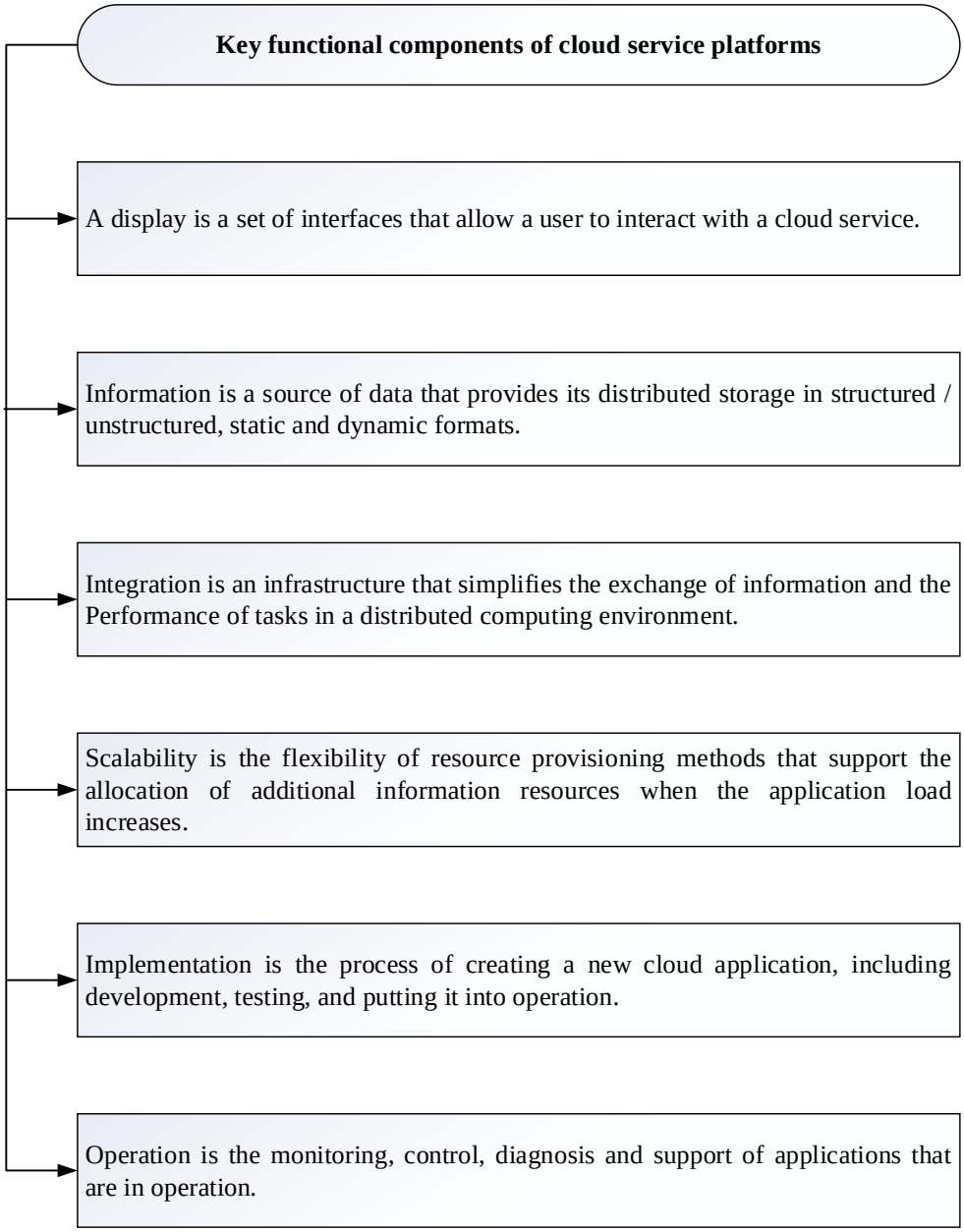


Fig. 5.9. Functional components of cloud platforms

Public cloud. It is an infocommunication network infrastructure developed for free use by a wide range of customers. Physically, such a cloud exists in the jurisdiction of the owner (an IT service provider) and is used and administered by commercial, governmental, scientific and other organisations. The network infrastructure of a public cloud is a set of hardware and software resources that any user can utilise in order to solve a specific range of, in accordance with the terms of an agreement regarding the level of service to be provided.

Hybrid cloud. This architectural model is a combination of two different (public and private) cloud infrastructures which remain unique infrastructure objects. These are connected using standardised or other data transmission technologies and relevant applications. Examples of such an approach include: public clouds used simultaneously or in combination with a private cloud; and short-term use of public cloud resources for the purpose of load balancing between private clouds.

The principal benefits of utilising cloud technologies can be summarised as follows: accessibility, flexibility, reliability and high computing power. The principal disadvantages and constraints of utilising cloud services are as follows: the necessity for a consistent and reliable network connection, the complexity of privacy strategies and the high cost of equipment.

5.2 Fog Technologies

Fog computing is a model that enables data processing and storage in hardware and software environments between endpoints and cloud computing centres. Accordingly, fog computing may be defined as a concept whereby a portion of the data is processed at the level of local networks, rather than exclusively in cloud data centres. Consequently, any hardware or software device that implements computing, networking and data storage tasks may potentially be used as a fog node. This encompasses microcomputers or programmable logic controllers in combination with network modules, IoT gateways, embedded servers and other similar devices. The main aim of fog technologies is to supplement cloud services by performing part of the data

processing and analytics tasks at the network edge, which allows for optimising data processing queues at the cloud server level. Numerous fog computing nodes process data in real time and generate analytical packages of information. This metadata is then transmitted to a central cloud platform where it is analysed to provide useful and / or predictive information.

The main technical and functional characteristics of fog computing technologies are as follows:

– low delay when processing data from end devices;

– the ability to track device locations;

– real-time interaction between network nodes (work with data in motion);

– high compatibility;

– the ability to directly exchange data with mobile devices.

Lists of the main advantages and disadvantages / limitations that are typical for fog technologies are set out in Figs. 5.10 and 5.11.

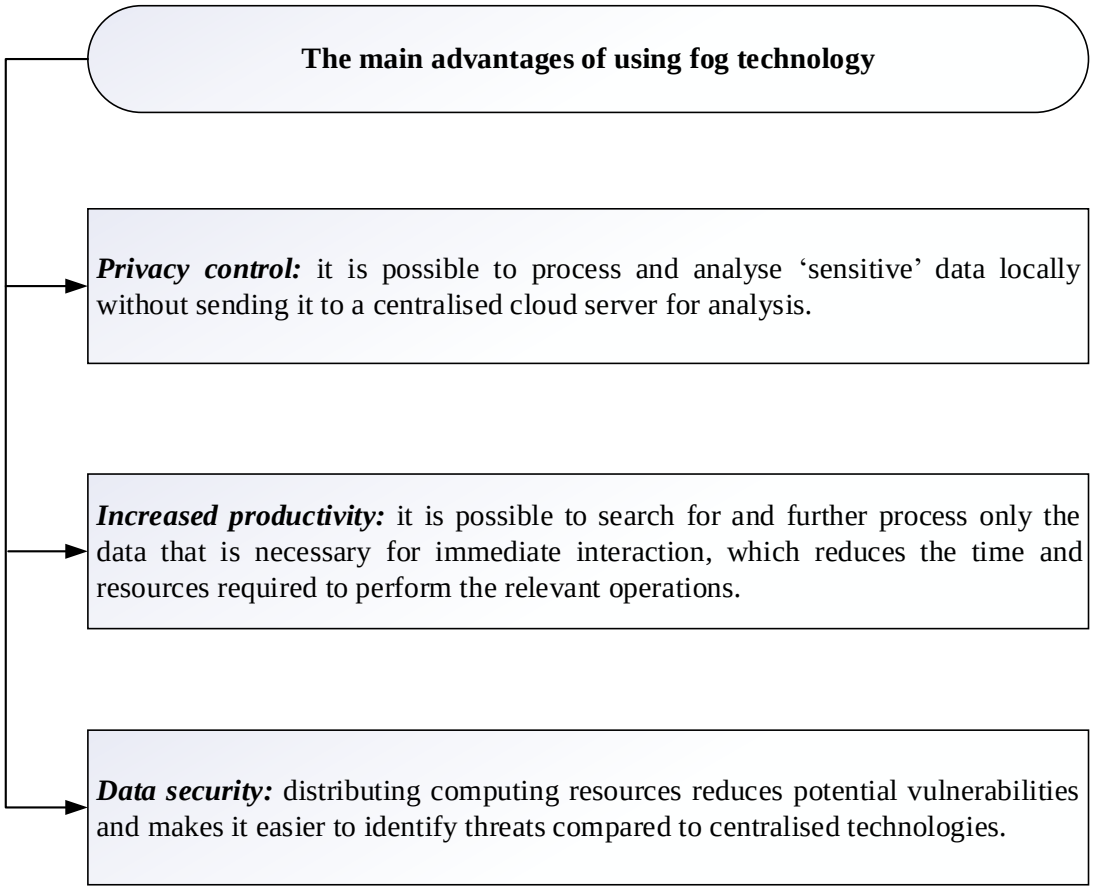


Fig. 5.10. Key advantages of fog technology

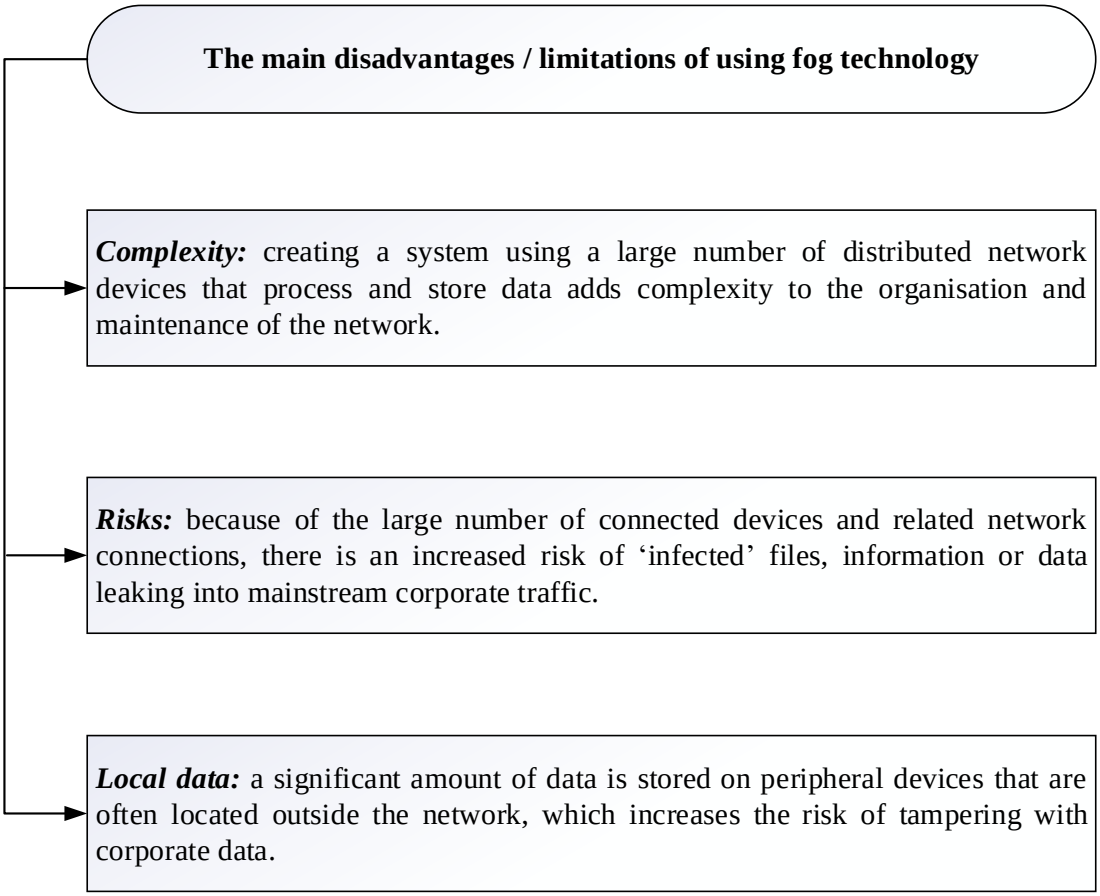


Fig. 5.11. Key disadvantages / limitations of fog technology

In the development of fog network hardware and software solutions, a multitude of potential topologies exists, each characterised by a set of parameters including computing device load, cost, scalability, user interface and others. Fog networks may be relatively modest in scale, comprising IoT gateways with fog computing support and a set of sensors. Such networks may also be more complex, comprising a multi-tier hierarchical structure with disparate data processing and storage capabilities. Furthermore, the majority of developed fog networks are also capable of scaling vertically and horizontally. A block diagram of a typical fog network is presented in Fig. 5.12.

The effectiveness of the implementation of fog computing technologies is contingent upon a number of key factors and characteristics, which can be broadly classified as follows:

– *reducing data volumes:* modern infocommunication technologies, including

IoT systems, are developed to aggregate unstructured data from a large number of real devices (stationary and mobile) and therefore, primary data processing at the network level allows for implementing the principles of data structuring, reducing information volumes and optimising network traffic;

– *number of endpoint devices:* when developing IoT networks, the factor of dynamic changes in the number of simultaneously connected devices should be taken into account and therefore the fog level must be vertically and horizontally scalable;

– *capabilities of fog nodes:* the topology of designing the fog level of IoT networks may vary and therefore, certain nodes of such networks should have the potential to solve machine learning, intelligent data processing, predictive analytics and other tasks;

– *hardware and software reliability:* when developing IoT networks, it is necessary to provide for the possibility of functional nodes failing, which can be implemented by using additional (backup) fog nodes in the overall network organisation.

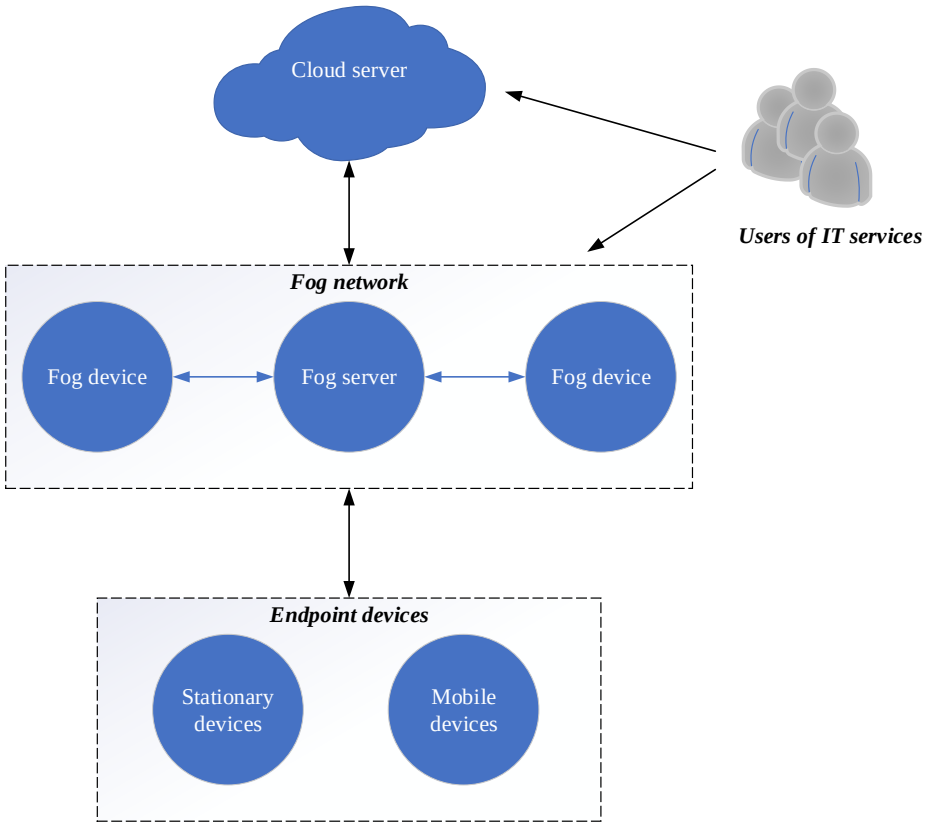


Fig. 5.12. Structural organisation of a typical fog network

Consequently, cloud computing technologies facilitate the processing of substantial quantities of data / information at the network's edge, which has a beneficial effect on the resolution of issues pertaining to information processing latency and data security. The interaction between cloud and fog technologies results in the provision of efficient and reliable data analytics. Therefore, a principal objective in the advancement and deployment of IoT systems is to justify and integrate an optimal network configuration that guarantees seamless and optimised interaction between distributed endpoints, fog and cloud data services.

5.3 Edge (Peripheral) Computing Technologies

Edge computing, which is analogous to fog computing, which was discussed in the preceding subsection, is a distributed technology. Similarly, edge computing can be defined as a technology that facilitates the proximity of computing processes to the locations where measurement data is generated and collected, in a manner analogous to fog computing. As the name implies, edge computing is performed at the ‘edge’ of the Internet of Things (IoT) network. From the perspective of network topology, this implies that the edge computing device is structurally situated in close proximity to or on top of the end nodes of the infocommunication network. This technology has been developed with the intention of interoperating with cloud and fog technologies within the network organisation of IoT systems, as illustrated in Fig. 5.13.

As previously stated, edge computing facilitates the deployment of processing and storage resources in close proximity to the hardware components that generate and collect data. This approach significantly optimises the overall processing time by minimising the tasks involved in transferring data from endpoint devices to centralised data centres and vice versa. This approach optimises data processing by minimising network bandwidth requirements and also ensures low operating costs by allowing the software to be used in remote locations in the event of an unreliable network connection or its absence. Consequently, edge computing is beneficial in information environments that necessitate real-time data processing and minimal time delay. A

typical example of devices capable of implementing edge computing techniques in the context of the IoT conceptual framework is smart sensors. These were discussed in detail in subsection 3.2, entitled ‘Intelligent sensors’, of the textbook.

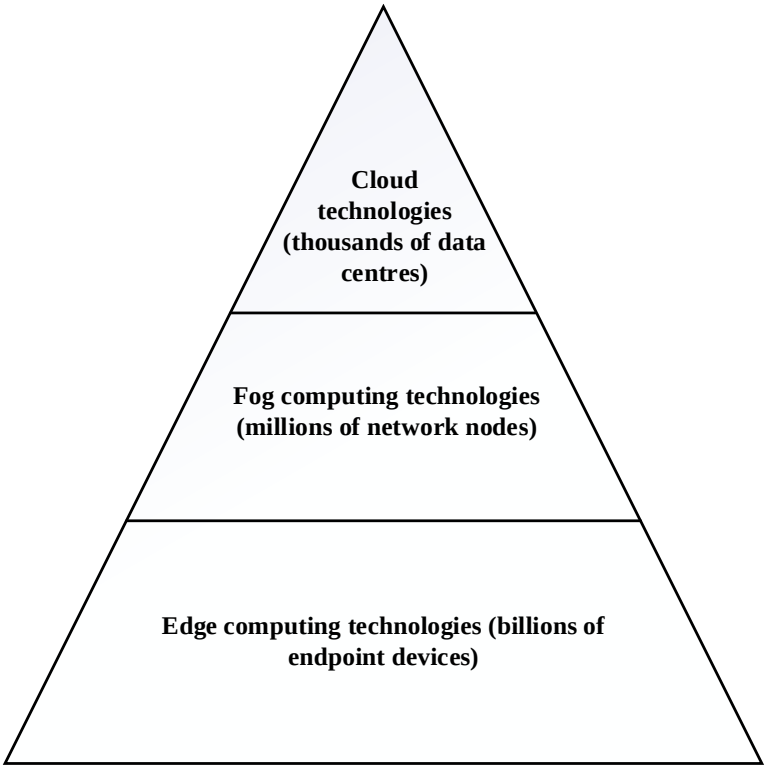


Fig. 5.13. Hierarchical structure of IoT computing technologies

Edge computing technologies are both vertical when analysing IoT systems in a comprehensive understanding of their hierarchical structure and horizontal when analysing the interaction of IoT end devices. This model of network organisation and device interaction includes the following:

– peer-to-peer networks;

– network interaction of edge devices;

– processing of distributed requests for data stored in devices, cloud data centres or other storages;

– distributed control of devices and data;

– ensuring data confidentiality.

Similar to the previously discussed fog technologies, edge computing

technologies possess a number of key technical and functional features that can significantly enhance the overall efficiency of IoT networks through their utilisation. A comprehensive overview of the fundamental attributes of edge computing technologies is presented in Fig. 5.14.

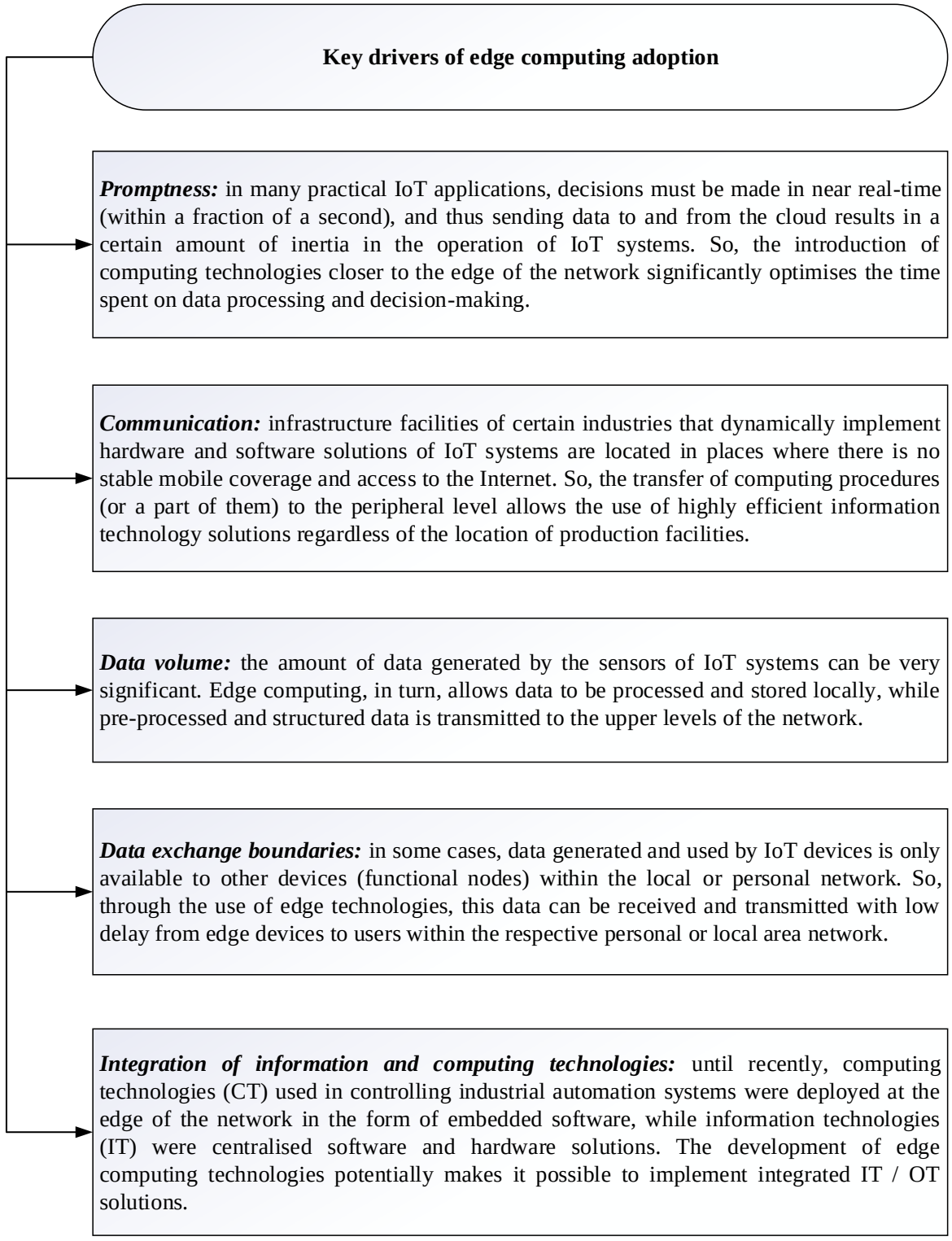


Fig. 5.14. Key principles for implementing edge computing technologies

As edge computing is developed and integrated into IoT systems and networks, it is crucial to prioritise the control of data and information flow. This decision will ultimately influence the level of usability, the quality of network traffic, operational efficiency, and the security and confidentiality of information. The principal factors that facilitate the optimisation of data control processes when utilising edge computing technology are illustrated in Fig. 5.15.

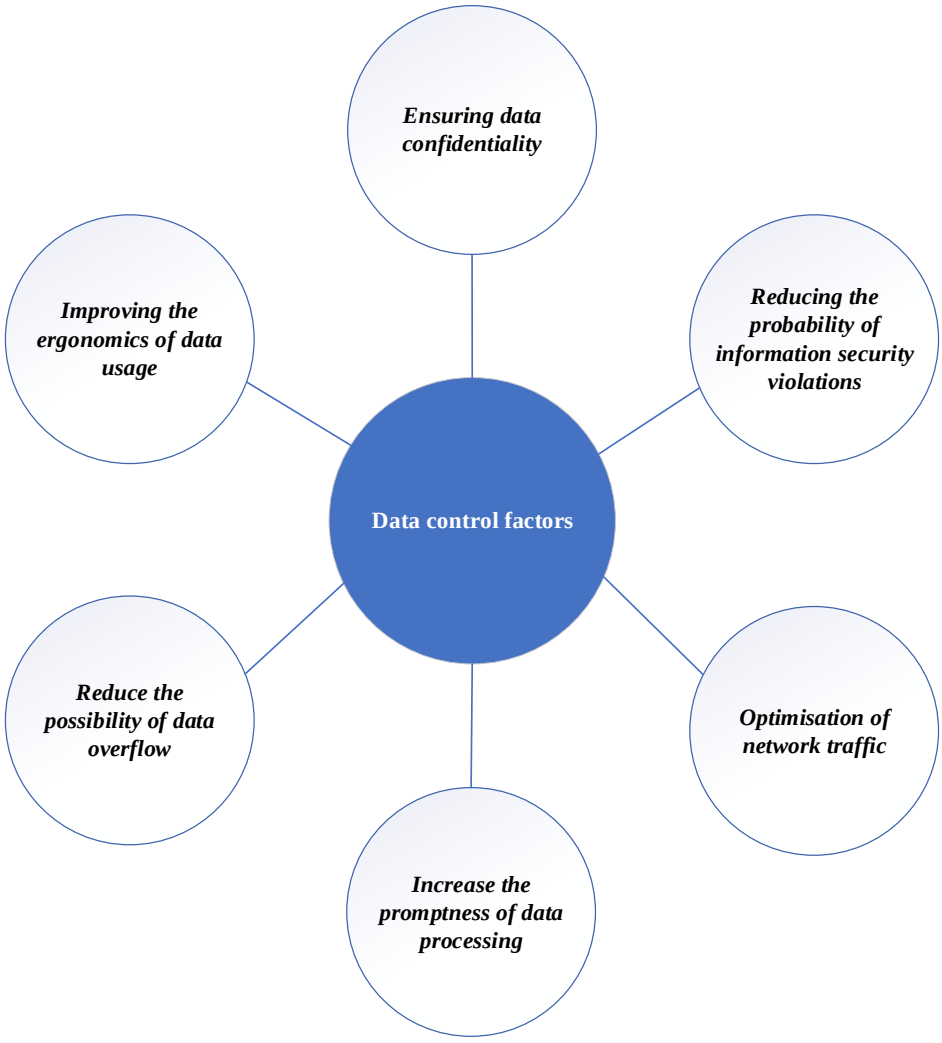


Fig. 5.15. Key drivers of data control in edge computing technologies

It is worthy of mention that edge computing is currently employed extensively in the creation of IoT systems and networks, functioning as an indispensable component of these systems. The ongoing advancement of hardware components at the device level of IoT systems and networks enables the incorporation of edge logic for

the operational processing of measurement data and the formulation of pertinent decisions. This approach enables the generation of control signals for actuators based on the processed data, without exceeding the lower levels of the network organisation of IoT systems and networks. It is also important to note that the challenge of synthesis, as well as the software and hardware implementation of architectural solutions for IoT systems and networks that are controllable, reliable and secure, taking into account the integrated interaction of all hierarchical levels (edge, fog and cloud) of data transformation, remains a significant issue in the present context.

5.4 Comparison of Data Processing Technologies

As previously stated, the principal computing technologies currently employed in the development of IoT systems with diverse architectures and applications are cloud, fog and edge computing. These technologies are not in competition with one another; rather, they are intended to be mutually complementary. The integrated use of the aforementioned computing technologies allows for the collection, processing and analysis of data at different hierarchical levels of IoT systems and networks. This has a positive impact on the overall efficiency, reliability and security of the transformation of large amounts of data and information messages generated by hardware and software of IoT systems and networks.

In general, the interaction of computing technologies is such that distributed edge devices equipped with data processing tools, such as smart sensors, generate large amounts of measurement data that are pre-processed at the device level by endpoint nodes. The pre-processed data is then aggregated, transformed and structured at the level of fog computing network nodes, for example, using IoT gateways. The structured data is then transferred to cloud data centres for long-term storage, deep intelligent analytics and remote access to users in accordance with their rights and roles.

A graphical interpretation of this interaction of IoT computing technologies is shown in Fig. 5.16.

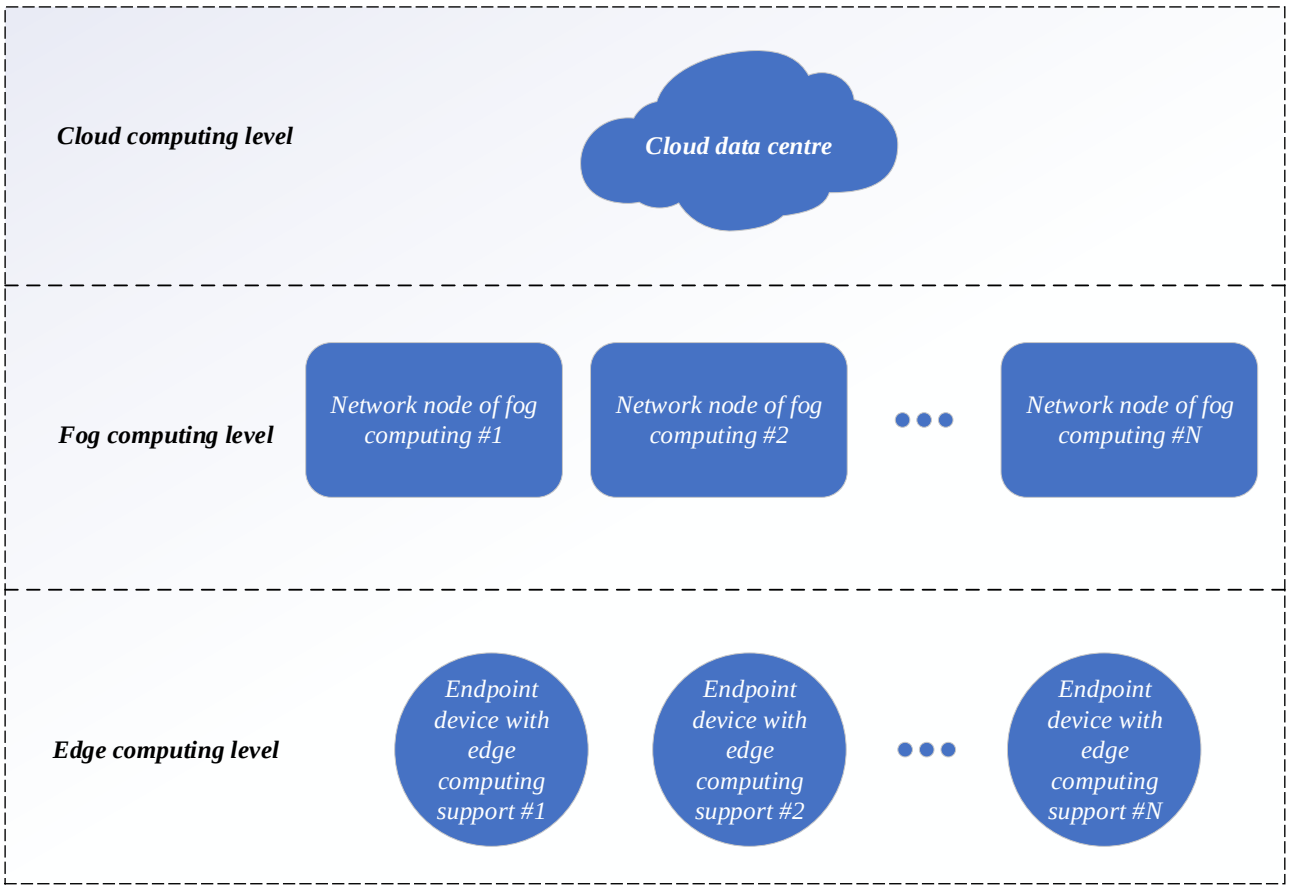


Fig. 5.16. Integrated network organisation of IoT computing technologies

For a detailed understanding of the functional suitability of a particular type of networked computing technology, this subsection of the textbook presents the results of the analysis and logical summary of the main technical and functional capabilities of cloud, fog and edge technologies. The results of the comparative analysis of cloud and fog IoT technologies are presented in Table 5.1.

A review of the information presented in Table 5.1 reveals that the primary distinguishing characteristics of fog and cloud technologies can be attributed to the differing approaches to data collection and processing. In cloud technologies, these procedures are typically implemented at the centre of the network, whereas in fog technologies, they are situated closer to the edge. Consequently, there are limitations imposed on the quantity of generated content, the time delay in data processing, the computing power required for the implementation of complex data transformation and analytics algorithms, as well as the number of users with potential access to information content.

Table 5.1. Comparative characteristics of cloud and fog technologies

Technology / Characteristics	Fog	Cloud
Location of computing resources (storage, data processing)	Network edge	Network centre
Delay level	Low	Varies widely
Access type	Wireless in most cases	Fixed or wireless
Support for mobility	Yes	No
Control type	Distributed / hierarchical (partial control)	Centralised / hierarchical (full control)
Accessibility to services	At the network edge / from user devices	Through the network core
Level of accessibility	High reservation level / high instability	99.99 %
Potentially allowed number of users	Billions	From dozens to hundreds of millions
Main data / content generator	Devices	People / devices
Locations of data / content generation	Everywhere	In a central location
Locations of information / content consumption	Everywhere	On endpoint devices
Virtual software infrastructure	User devices	Central corporate servers

In turn, a comparison of fog and edge (peripheral) computing technologies requires a more thorough analysis of the theoretical and applied foundations of their structure and trends in use than a corresponding comparative analysis of fog and cloud technologies. This is because of the fact that these technologies have a sufficient number of similar and a certain number of detailed distinctive functional and organisational characteristics. The key similarities between cloud and edge computing technologies are shown in Fig. 5.17.

It is also worth noting that in many cases, experts and researchers in the field of information, computer and telecommunication technologies use the terms fog and edge computing synonymously. However, the key difference between fog and edge computing technologies is that edge computing takes place at the edge of the network, in close physical proximity to endpoint devices that collect and / or generate measurement data, while fog computing acts as an 'intermediary' (intermediate layer) between the edge and the cloud. A detailed analysis of the distinctive features of fog and edge computing technologies of IoT systems and networks is given in Table 5.2.

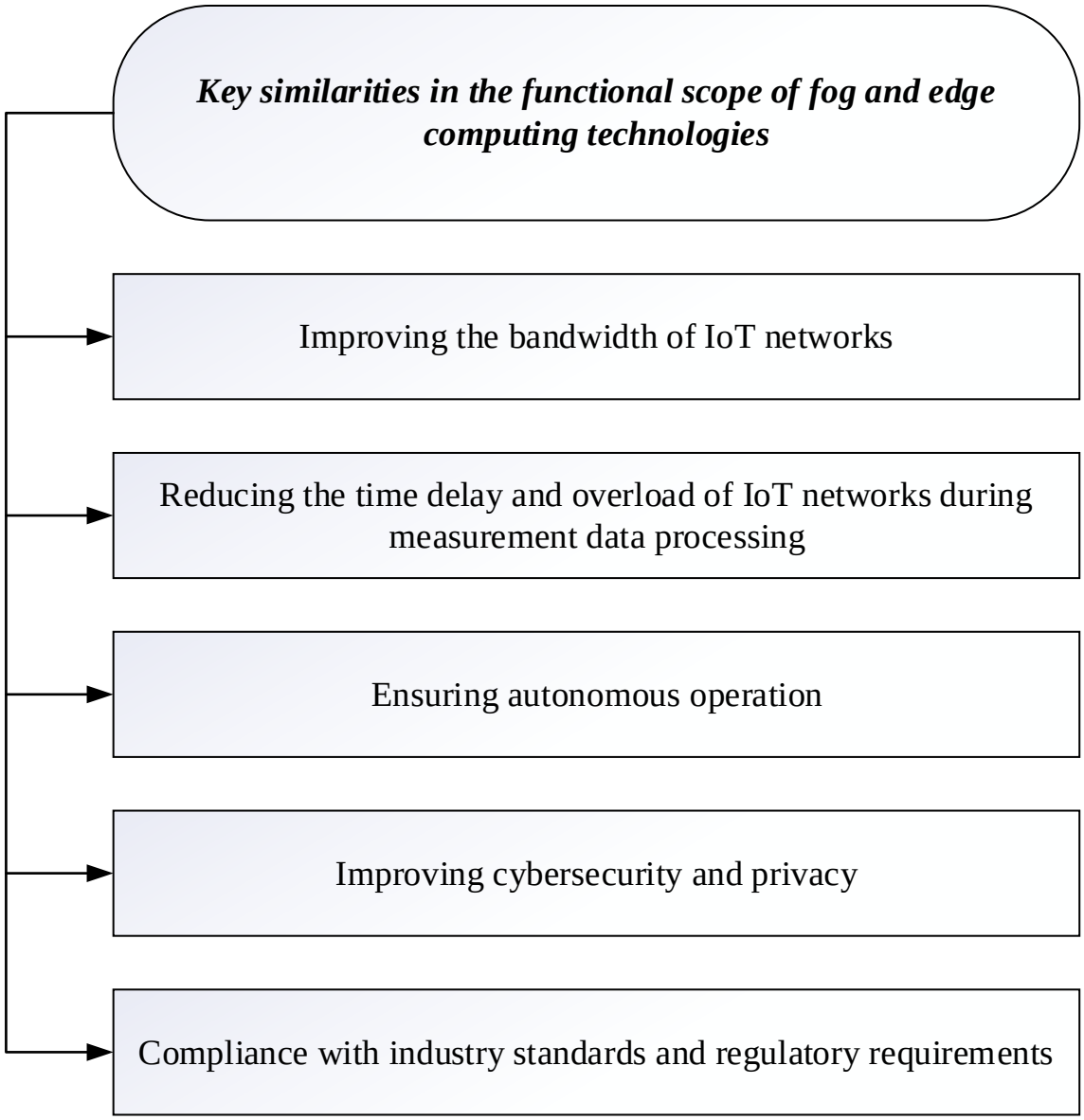


Fig. 5.17. Similarity of functional scope of fog and edge computing technologies in IoT systems

Table 5.2. Key differences between fog and edge computing technologies

Technology / Characteristics	Fog computing	Edge computing
Physical content of the technology	The main aim is to process data as close to the edge of the network as physically possible. Unlike edge computing, fog computing is not implemented by the same device that generates the data.	Defined as a computing architecture that physically brings data processing closer to the data source. In many cases, data collection and processing take place on the same endpoint device.
	Edge computing devices are in most cases the same devices that collect and generate measurement data and fog computing devices are functional nodes that are physically located next to these edge devices.	
Types of applications	Quite often deployed in time-sensitive hardware and software solutions that require a large amount of resource-intensive processing of data aggregated from network devices. Examples of such applications include online analytics and smart grids.	This technology is typically used in less resource-intensive applications due to the limited capabilities of devices that collect and process data. Examples of such applications include: technical monitoring, diagnostics and measurement control of physical processes and objects.
Application area	This technology reduces the load on edge and cloud computers by performing processing tasks on both sides of the network. The fog nodes are physically located next to the edge device, which means that they can be connected to a local network. This computing technology is used in environments where the cloud server is significantly remote to ensure efficient response times and where edge devices are resource-constrained or physically dispersed. By definition, a fog computing device is not capable of detecting or generating data. Consequently, fog computing would not exist without edge tools.	Such computations are typically performed on user endpoints (laptops, smartphones, etc.) or real IoT devices (smart sensors). Edge computing devices are potentially capable of processing measurement data and exchanging data processing results with a cloud service, so edge computing is possible without fog computing.

Continuation of Table 5.2

Principles of data processing and storage	Fog computing devices organise the network connection of peripheral devices, creating a local area network for more efficient data processing and storage. Fog computing improves the efficiency of personal or corporate networks by realising the scale of operations through the use of data from multiple endpoints. This process takes place without the delays and overloads associated with a direct connection to the cloud.	In the case of edge computing, data is processed and stored in the edge device itself. Since edge devices have limited processing and storage capabilities, data can be transferred to fog nodes or cloud centres for further processing.
Economic aspects	This technology is more expensive than edge computing. However, it has a number of advantages: saving bandwidth, computing and storage resources. Fog computing is a cost-effective alternative to large data centres.	Edge computing services are provided by leading vendors such as Microsoft and Amazon. Companies and business organisations can also create their own edge infrastructure. Customised edge setups will be more expensive, whether they are developed from scratch or users subscribe to them.

In conclusion, as evidenced in Table 5.2, fog and edge technologies are developed with the objective of relocating computing power in closer proximity to the data source. This enables the real-time implementation of the application principles of efficient data processing and transmission. Both of these technologies can be employed in almost all industry sectors in the development of innovative hardware and software solutions for IoT networks, which enhances the interaction between users and IoT systems.

5.5 IoT Platforms

An IoT platform is generally a multi-tier technology developed to solve the tasks

of administering, monitoring and controlling events and devices that are connected to an IoT network. In other words, an IoT platform can be defined as an IT service that enables the integration of distributed real devices into a single network. The key characteristics of modern IoT platforms that meet the current requirements of the IT services market include the following:

– scalability: almost all modern IoT platforms allow for varying the number of network devices and make it possible to scale the system to meet individual user needs;

– multifunctionality: the ability to develop algorithms and scenarios for automated control of various types of devices for different application areas on a single platform;

– ergonomics of use: is determined by the convenience and ease of use, as well as the flexibility of source code control and API integration;

– compatibility and integration: defined by the ability to combine devices with different types of software and hardware protocols;

– deployment: determined by the types of server technologies used;

– data security: consists of the administration and control of user access, ownership of processed data and information and information encryption algorithms.

The structure of a basic IoT platform should combine a number of structural and functional components, as shown in Fig. 5.18.

As posited by experts in the field of IoT, a comprehensive IoT platform should facilitate the development of user-created applications. Consequently, such a platform must comprise a greater number of structural and functional components than the basic version (see Fig. 5.18). The authors of IoT Analytics have substantiated the structural and functional provision of a full-fledged platform, as shown in Fig. 5.19.

Further in this subsection, a description of popular IoT platforms that can be used when creating software and hardware solutions for Internet of Things systems and networks is given.

It is worth noting that this list of popular IoT platforms is based on the availability of the platforms for use in projects of various scales and application areas.

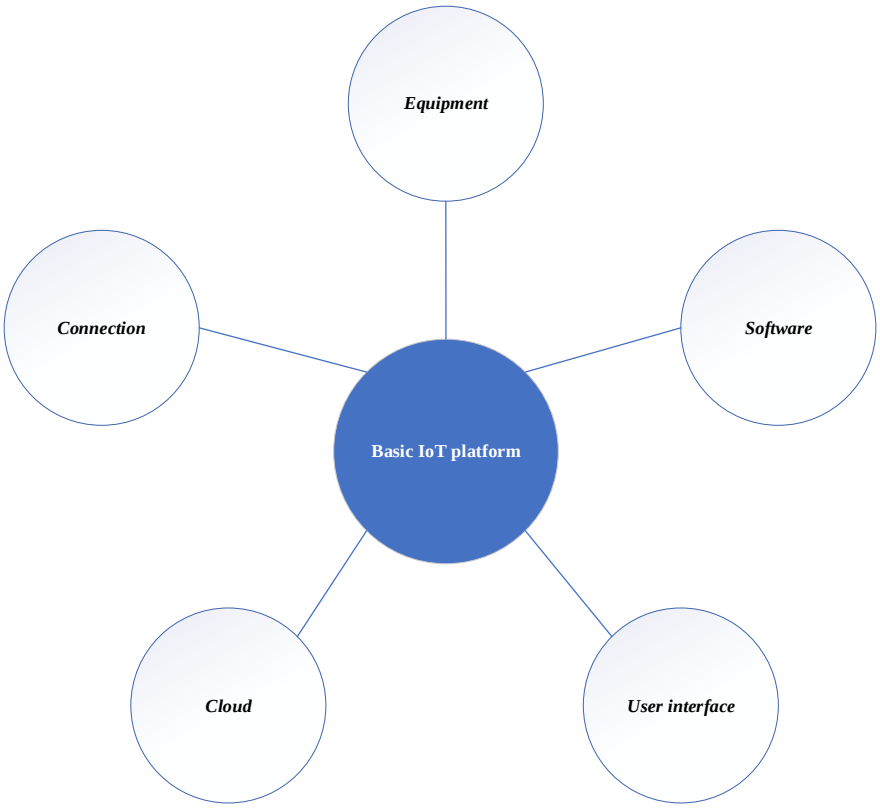


Fig. 5.18. Basic structural and functional provision of IoT platforms

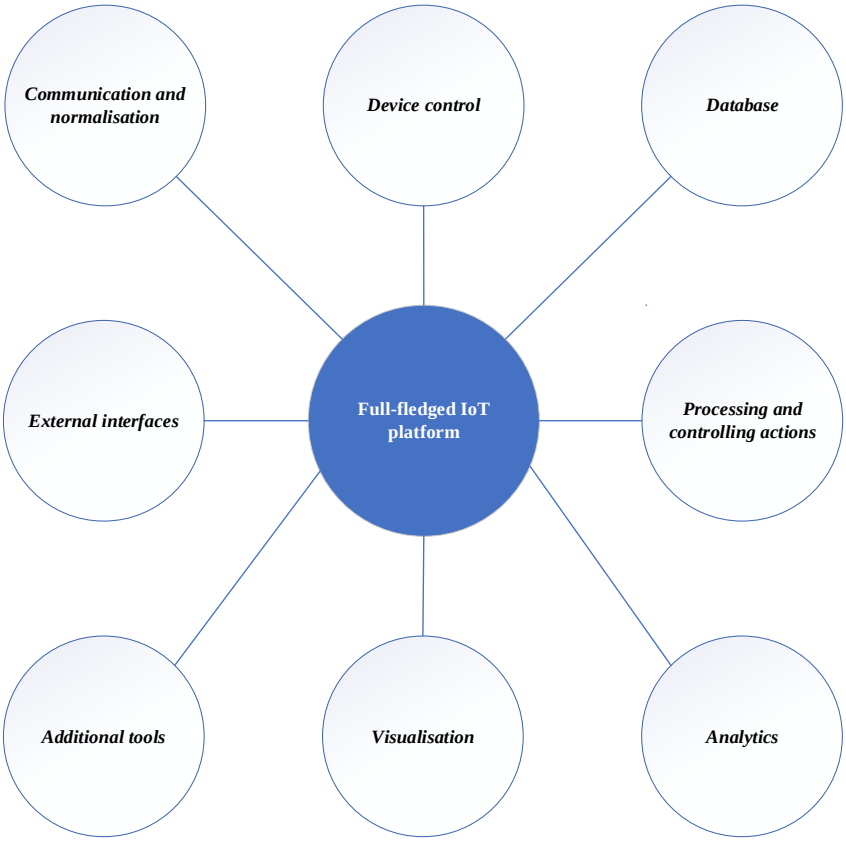


Fig. 5.19. Structural and functional provision of full-fledged IoT platforms

ThingSpeak IoT Analytics. This Internet of Things (IoT) platform enables the

collection, analysis and visualisation of data, including the utilisation of Matlab functionality without the necessity of purchasing a licence, within the Mathworks environment. The IT services of this platform facilitate the implementation of the principles of aggregating and storing measurement data from distributed smart sensors. The ThingSpeak IoT platform is suitable for working with microcontroller (microcomputer) devices, including Arduino, ESP, Raspberry Pi and others. The platform allows the creation of mobile and web applications. The general functionality of this IoT platform is illustrated in Fig. 5.20.

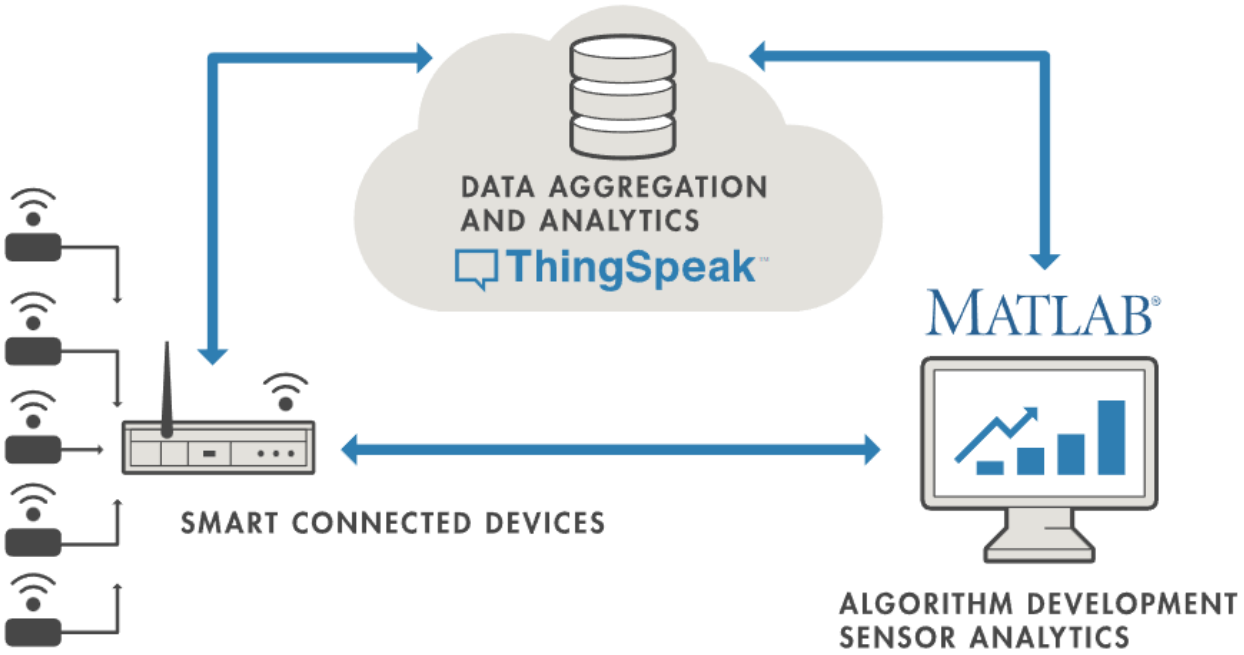


Fig. 5.20. Generalised functionality of the ThingSpeak IoT Analytics platform

The main functionalities of the ThingSpeak IoT Analytics platform are as follows:

– collecting data via private channels;

– providing data via public channels;

– support for Matlab analysis and visualisation capabilities;

– data processing and graphical visualisation of monitoring results in real time;

– the ability to prototype IoT systems without additional web server settings;

– automatic generation of warning signals;

– compatibility with external applications.

Thingsboard. This IoT platform is a 100 % open-source solution that can be used as a cloud environment based on SaaS or PaaS technologies when developing IoT systems and networks. The main aim of this platform is to collect, process and visualise data, as well as to control devices of IoT systems. The Thingsboard IoT platform is compatible with a range of standard network protocols, including MQTT and HTTP, facilitating the interaction of distributed devices in both local and cloud environments. The structural and functional organisation of the Thingsboard IoT platform is shown in Fig. 5.21.

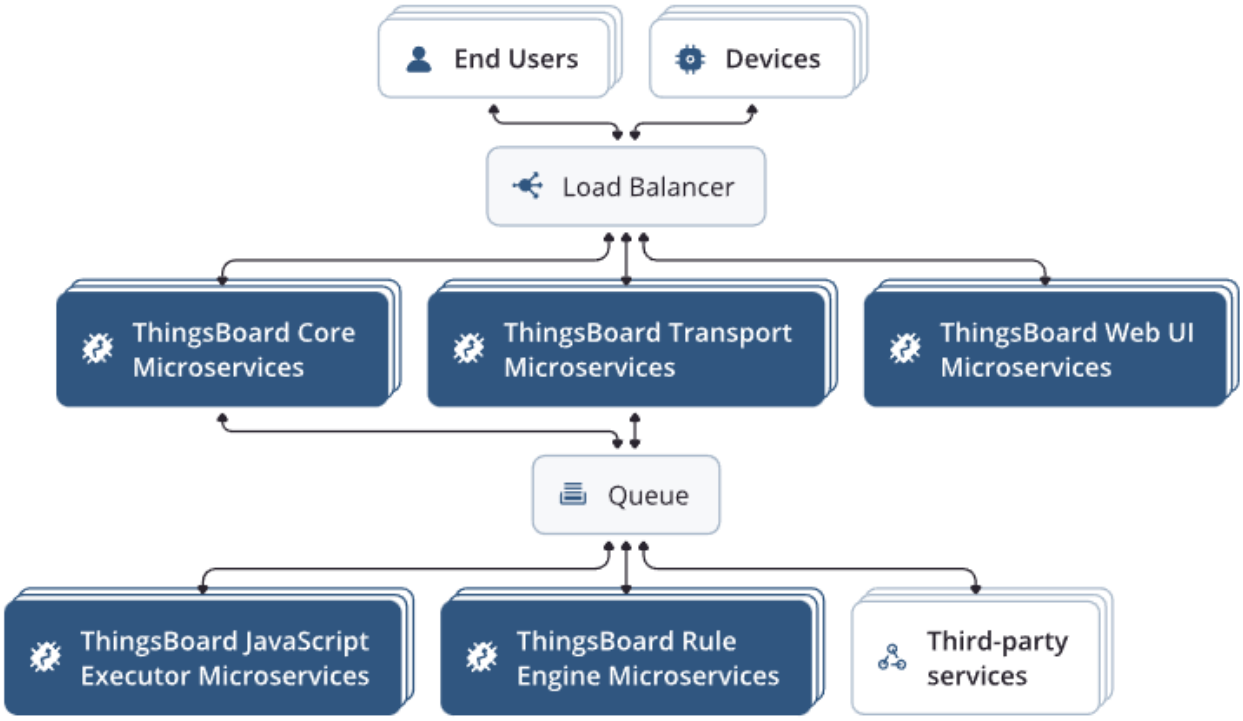


Fig. 5.21. Structural and functional organisation of the Thingsboard IoT platform

The main functionalities of the Thingsboard platform are as follows:

– real-time data collection, analysis and visualisation;

– remote device control;

– support for up to 30 widgets for customised monitoring and control panels;

– support for allowing monitoring and providing attributes of client devices on the server side;

– support for data encryption during transmission via MQTT and HTTP protocols;

– the ability to diagnose network nodes in terms of their performance;

– the possibility of vertical and horizontal scaling of implemented IoT projects.

Microsoft Azure IoT Suite. This IoT platform is developed to be integrated into IoT systems and networks for various industry needs. The main vector of functionality of the Microsoft Azure IoT Suite platform is aimed at solving the tasks of remote and intelligent monitoring, diagnostics and control of smart device network organisations. A generalised functional diagram showing the principle of operation of this IoT platform is shown in Fig. 5.22.

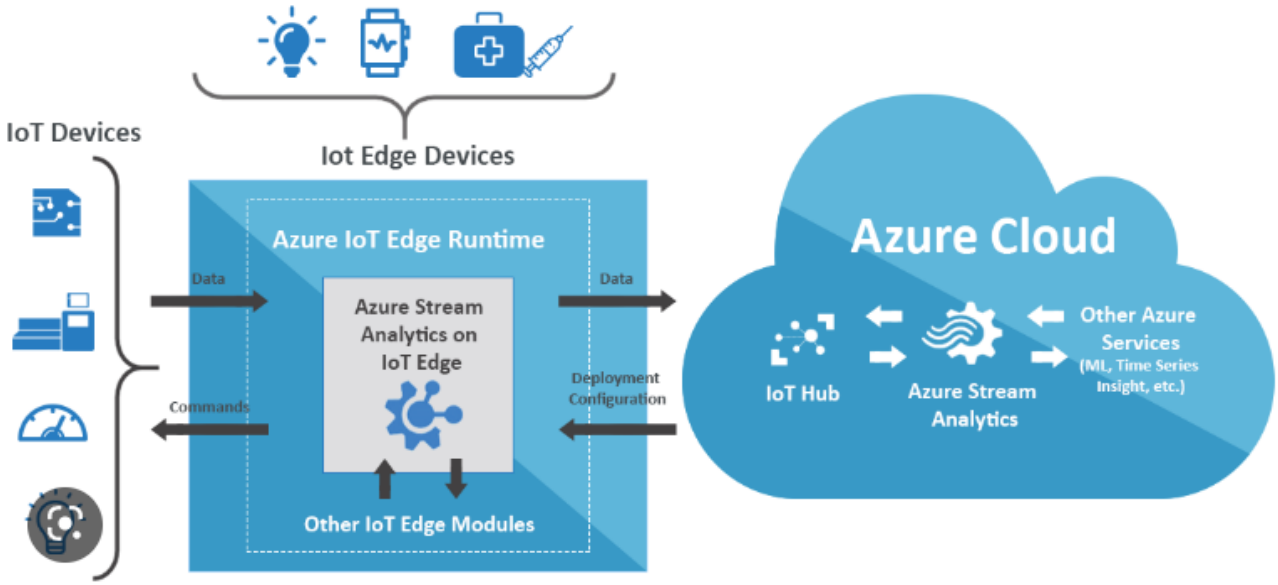


Fig. 5.22. Structural and functional organisation of the Microsoft Azure IoT Suite platform

The main functionalities of the Microsoft Azure IoT Suite platform are as follows:

– the ability to quickly launch the platform using Windows Server and Linux virtual machines;

– support for a significant number of application services and web programs that allow scaling IoT systems to meet the needs of a wide range of application areas;

– support for integration with its own highly reliable cloud data storage;

– aggregation, processing and visualisation of large amounts of measurement data in real time;

– availability of integrated software tools for developing analytical solutions for deep intellectual data transformation.

ThingWorx. This IoT platform is characterised by high levels of dynamic flexibility and adaptability of deployment. When using this IoT platform, data can be accessed from a local, remote or hybrid environment. The developers of this platform have placed a significant emphasis on the Industrial Internet of Things (IIoT), which has been achieved through the utilisation of hardware and software solutions that exhibit enhanced reliability and information security. An illustration of an IoT network architecture utilising the ThingWorx platform is presented in Fig. 5.23.

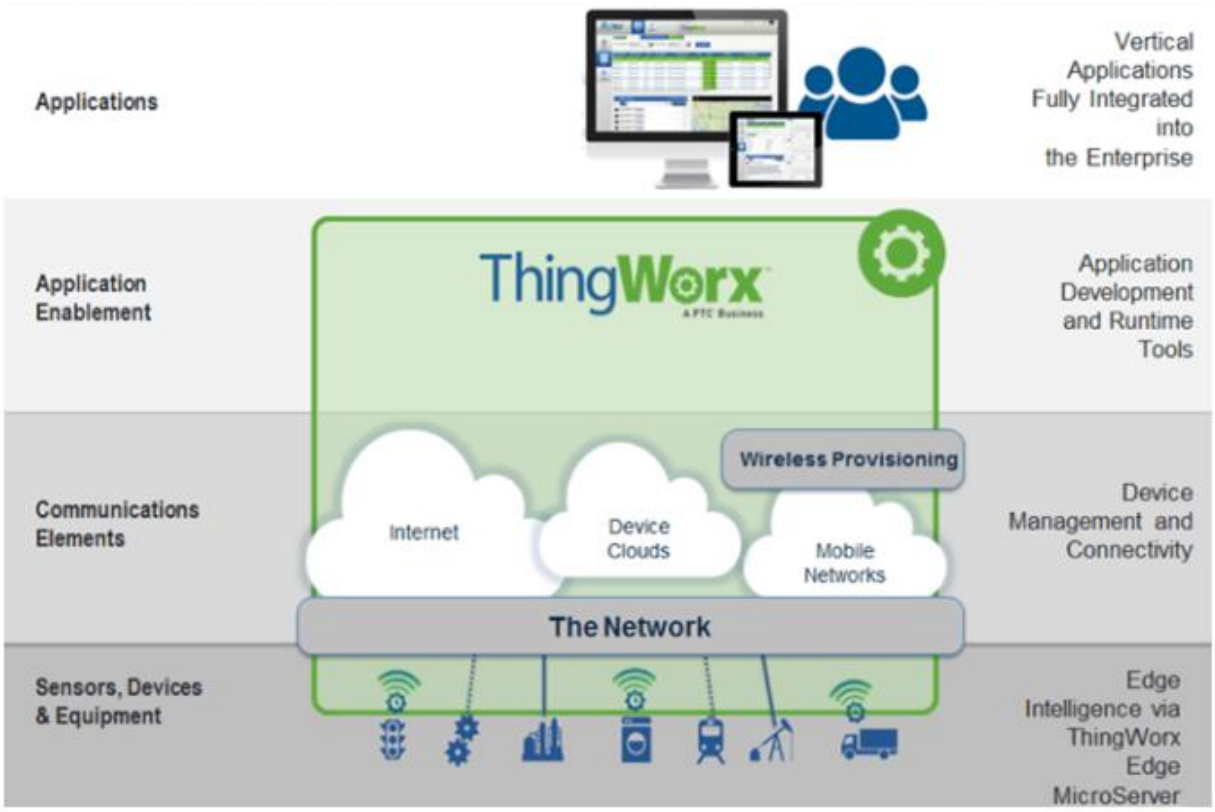


Fig. 5.23. An example of IoT network architecture using the ThingWorx IoT platform

The main functionalities of the ThingWorx IoT platform are similar to those

analysed above and are as follows:

– network organisation of remote devices;

– collecting, transforming, analysing, storing and visualising large amounts of measurement data in real time;

– support for the development of customised IoT applications based on a significant base of integrated models;

– support for a significant number of web services and applications that allow scaling IoT systems to meet the needs of a wide range of application areas;

– implementation of the principles of visibility and control of network devices based on roles;

– high data reliability and security.

IBM Watson IoT. This IoT platform is developed to implement a full cycle of data collection and transformation, providing a wide range of tools and services to optimise the digital transformation of enterprises and business structures. The IBM Watson IoT platform allows aggregating and analysing data from various sensors, devices and technological equipment, as well as provides decision-making support. An example of an IoT system architecture using the IBM Watson IoT platform is illustrated in Fig. 5.24.

The main functionalities of the IBM Watson IoT platform are as follows:

– aggregation of data from distributed devices operating on different hardware and software protocols;

– highly efficient data analytics using artificial intelligence and machine learning algorithms;

– vertical and horizontal adaptability and scalability of software and hardware solutions;

– online operation with ‘data in motion’;

– the ability to implement custom applications using a significant built-in database of tools and models;

– high data reliability and security.

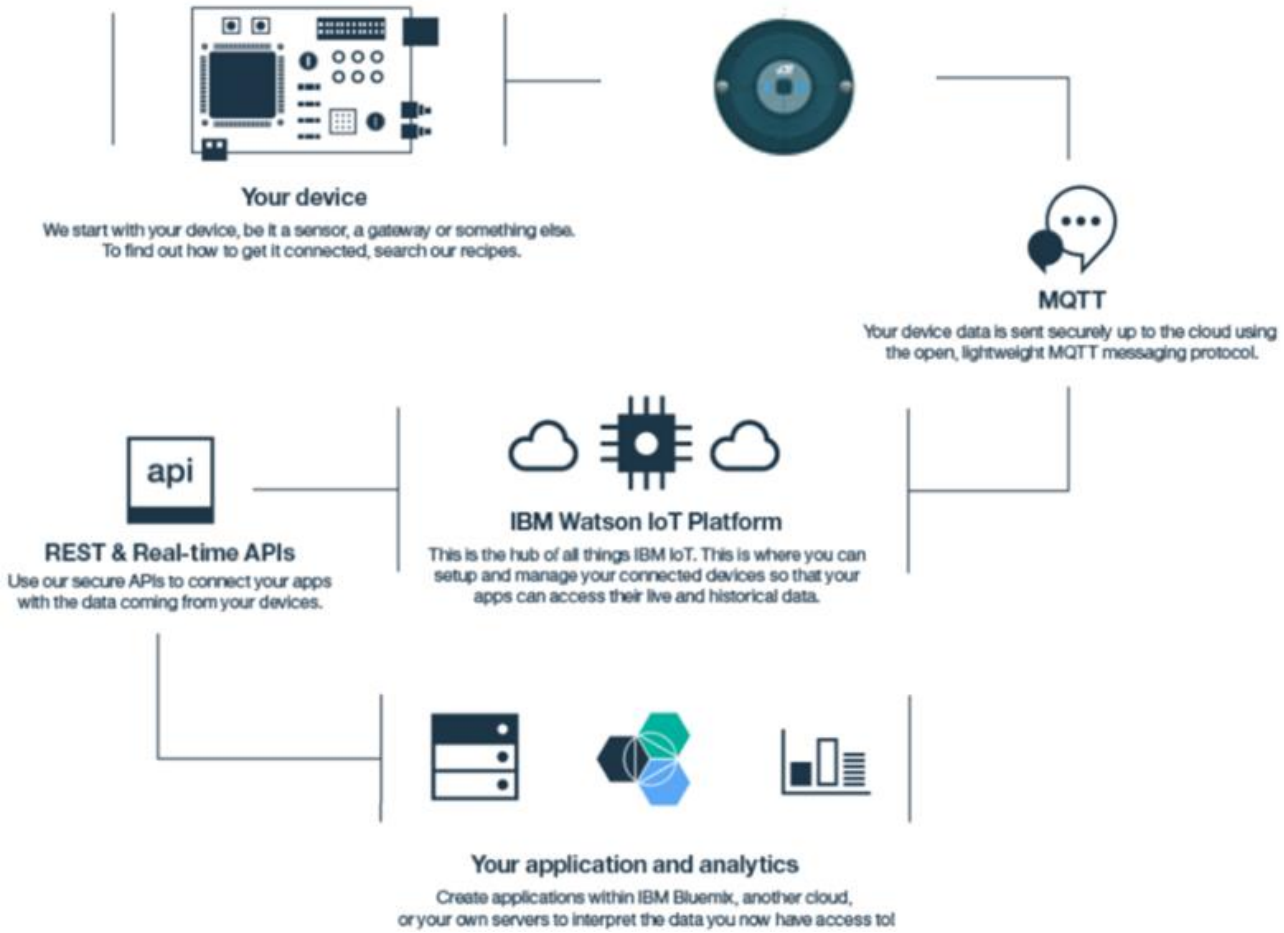


Fig. 5.24. An example of an IoT network architecture using the IBM Watson IoT platform

The developers of this IoT platform also highlight the fact that it has been widely used in applied industries related to the optimisation of operations and resources through the implementation of approaches for highly efficient statistical data analysis and ergonomic means of bilateral data and information exchange.

Google Cloud Platform. This IoT platform is a multi-tier secure infocommunication infrastructure that ensures high operational efficiency. Google Cloud Platform implements predictive maintenance capabilities for production and process equipment, smart city infrastructure and others.

An example of the architecture of an IoT system operating on the Google Cloud Platform is shown in Fig. 5.25.

The main technical and functional capabilities of Google Cloud Platform are as follows:

– highly efficient integrated machine learning and artificial intelligence tools, models and approaches;

– real-time statistical analysis of globally distributed data;

– support for a large number of application services and web programs that allow scaling IoT systems to meet the needs of a wide range of application areas;

– support for a wide range of embedded operating systems;

– high data reliability and security;

– significant computing resources.

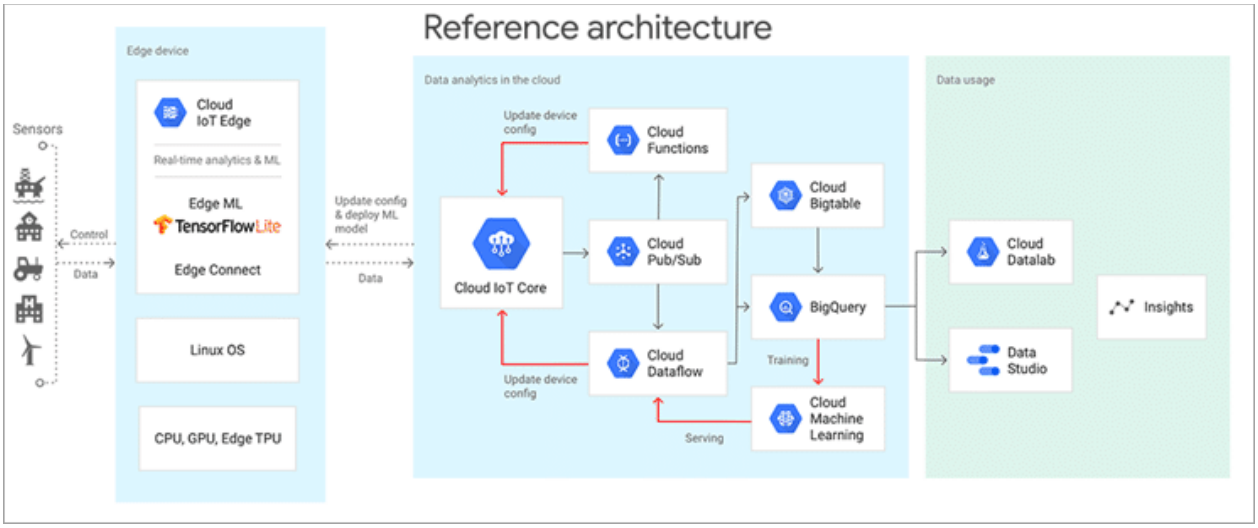


Fig. 5.25. An example of IoT system architecture using Google Cloud Platform

It is once again necessary to emphasise that the IoT platforms described and analysed in this subsection do not constitute a complete list of existing IoT platforms that are currently in use. This list demonstrates the extensive range of pertinent IT solutions that can satisfy the requirements of the information and digital technology market, encompassing both do-it-yourself (DIY) projects and highly reliable industrial Internet of Things (IIoT) solutions.

Self-assessment Questions on Material Comprehension

1. Provide a generalised architecture of cloud computing technologies. Explain

the theoretical and applied principles of cloud services.

2. Explain the functional scope and characteristics of the main service models in cloud computing: SaaS, PaaS and IaaS.

3. Describe the nested structure of cloud computing models.

4. Describe the functional scope and characteristics of additional service models in cloud computing: HaaS, WaaS, DaaS and EaaS.

5. Explain the content of topological models of cloud services: private, public and hybrid clouds.

6. Describe the main functional components of cloud platforms.

7. Explain the principle of operation, structural organisation and main advantages and disadvantages of fog computing technology.

8. Describe the principle of operation, structural organisation and key factors in the implementation of edge computing technology.

9. Provide a detailed comparative description of cloud and fog computing technologies.

10. Provide a detailed comparative description of fog and edge computing technologies. What are the key similarities and differences between them?

11. Define the term ‘IoT platform’ and describe the main technical and functional capabilities of modern Internet of Things platforms.

12. Describe the main functional characteristics of the following IoT platforms: ThingSpeak IoT Analytics, Thingsboard, Microsoft Azure IoT Suite, ThingWorx, IBM Watson IoT and Google Cloud Platform.

List of Recommended Literature for the Fifth Chapter

1. Kharchenko V.S. (ed.) Internet of Things for Industry and Human Application. In Volumes 1–3. Vol. 1. Fundamentals and Technologies. Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. 605 p.

2. Kharchenko V.S. (ed.). Internet of Things for Industry and Human Application. In Volumes 1–3. Vol. 2. Modelling and Development. Ministry of

Education and Science of Ukraine, National Aerospace University KhAI, 2019. 547 p.

3. Zhurakovsky B.Yu., Zeniv I.O. Internet of things technologies. Study guide [Electronic resource]: education manual for students speciality 126 ‘Information systems and technologies’, specialisation ‘Information support of robotic systems’. Kyiv: KPI named after Igor Sikorskyi, 2021. 271 p. (in Ukrainian).

4. Laktionov I.S. Information and measuring equipment and hardware and software for building computerised systems for monitoring the state of microclimate in greenhouses: Thesis of Doctor of Engineering Sciences: 05.13.05 – computer systems and components / Donetsk National Technical University: D 11.052.03. Pokrovsk, 2021. 518 p. (in Ukrainian).

5. Zinchenko O.V. et al. Cloud technologies: a textbook. Kyiv: FOP Guliaeva V.M., 2020. 74 p. (in Ukrainian).

6. Instructor Textbook «Designing & Deploying Cloud Solutions for Small and Medium Business»: Rev. 1.0. Palo Alto: Hewlett-Packard Company, 2013. 893 p.

7. ThingSpeak™. Available at: https://thingspeak.com/ (accessed on January 18, 2024).

8. ThingsBoard Open-source IoT Platform. Available at: https://thingsboard.io/ (accessed on January 18, 2024).

9. ThingWorx IIoT Solutions Platform. Available at: https://www.ptc.com/en/products/thingworx (accessed on January 19, 2024).

10. How to connect your device to IBM Watson Cloud Platform. Available at: https://docs.iotize.com/Technologies/IBMWatson/ (accessed on January 20, 2024).

11. Google Cloud. Available at: https://cloud.google.com/ (accessed on January 20, 2024).

12. Krishnasamya E., Varrettea S., Mucciardi M. Edge Computing: An Overview of Framework and Applications. Brussels: PRACE, 2020. 20 p.

13. Laktionov I.S. et al. Improved Computer-Oriented Method for Processing of Measurement Information on Greenhouse Microclimate. Int. J. Bioautomation, 2019, Vol. 23 (1). P. 71–86.

CHAPTER 6
PRACTICUM

6.1 Description of the CupCarbon IoT Computer Simulation Software

CupCarbon is a specialised software for modelling IoT devices, systems and networks of various architectures and functional aims. The main aim of this software is to design, debug and visualise distributed networks and IoT monitoring algorithms in the context of various educational and research projects. The use of this modelling software allows for explaining clearly the basic theoretical and applied principles and scenarios of wireless sensor network organisations, data exchange protocols and others.

The general concept of the CupCarbon software supports two modelling environments. The first modelling environment allows for developing and exploring mobile scenarios for the monitoring of natural phenomena, smart cities and buildings, as well as more complex technological processes such as traffic flows and the remote control of unmanned aerial vehicles. The second modelling environment is a software product for simulating discrete events in wireless sensor networks, which allows for taking into account the specific features of network protocols and scenarios developed on the basis of the first computer modelling environment. The modelling process in the CupCarbon software is based on the application level of node communication, which allows nodes in IoT networks to be configured dynamically to be able to divide nodes into separate networks or join different networks. This modelling principle allows for understanding the structural and algorithmic organisation of IoT devices, systems and networks. The main network technologies represented in CupCarbon are as follows: ZigBee, LoRa and WiFi. Furthermore, the possibility exists to operate wireless sensor networks based on analogue and digital sensors.

The main window of the CupCarbon simulation software is shown in Fig. 6.1 below.

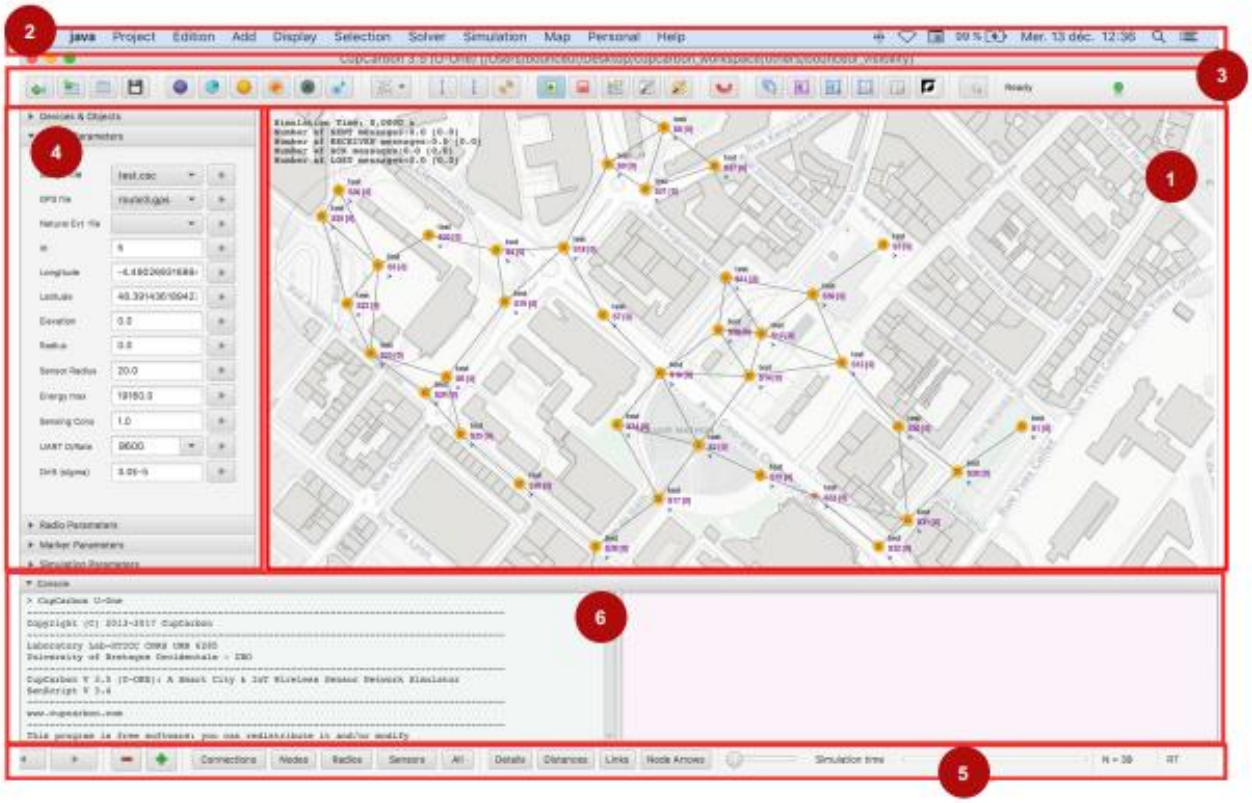


Fig. 6.1. General view of the main window of the CupCarbon workspace

The following notations are shown in Fig. 6.1:

1 – The map;

2 – The main menu (The menu bar);

3 – The toolbar;

4 – The parameter menu;

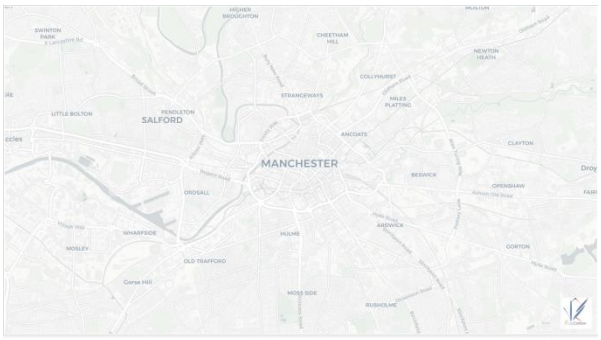
5 – The state bar;

6 – The console.

The functional aim and main characteristics of the structural elements of the CupCarbon software are as follows.

The map. It is the main object of the CupCarbon modelling software. This functional element is the basis of the IoT systems and networks being designed. CupCarbon supports the ability to select different map options (see Fig. 6.2), namely: light open street map; dark open street map; dashed background; small cell white grid background; small cell black grid background; black background; white background; mean gray cell background; mean blue cell background; notebook background; Google

map.



a) Light Open Street Map

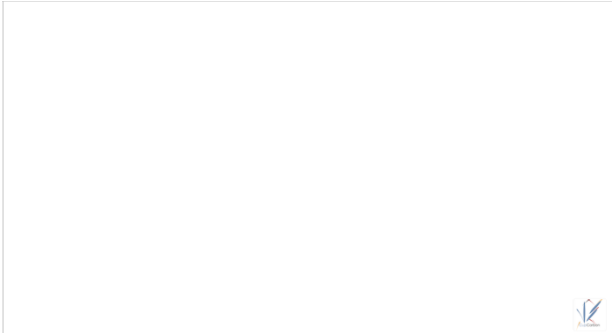
b) Dark Open Street Map

c) Dashed background

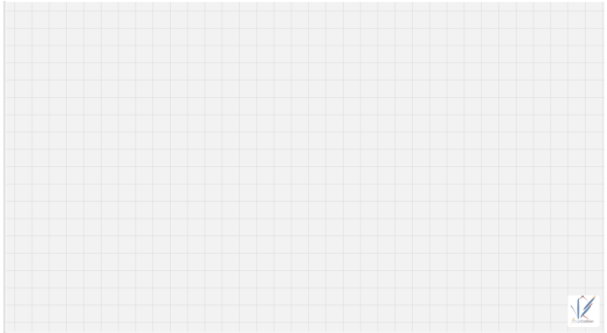
d) Small cell white grid background

e) Small cell black grid background

f) Black background

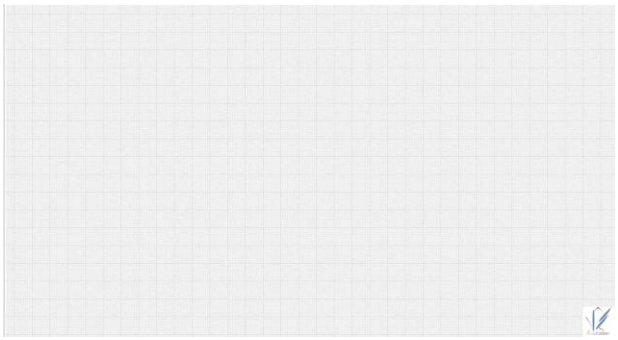


g) White background



h) Mean gray cell background

i) Mean blue cell background



j) Notebook background

k) Google map

Fig. 6.2. Options for available maps in the CupCarbon software

When modelling, the simulation time is also displayed in the upper left part of the map. This parameter is displayed in red during the simulation. This part of the map also displays other information about the number of sent, received and lost information messages. This information board can be hidden and displayed by pressing *ALT+D*.

The menu bar. This component is aimed at selecting the components of creating IoT models and working with the created models of IoT systems and networks. It is shown in Fig. 6.3.

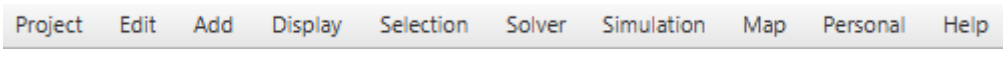


Fig. 6.3. General view of the CupCarbon main menu

The main functional components of the main menu are as follows:

– Project (see Fig. 6.4), which contains the following standard functionality: New project – allows for creating a new project; New project from the current – allows for creating a new project taking into account the settings of an existing project, this

option can be utilised to duplicate projects; Open project – utilised to open existing projects; Open the last project – utilised to open the last open project; Recent projects – allows for opening one of the five most recently opened projects; Save project – allows for saving the project; Reset – utilised to reset completely all project settings, if the project already exists and is not saved, this option will lead to the complete loss of this project; Start Server – utilised to start the project server; Quit – allows for closing and finishing the project.

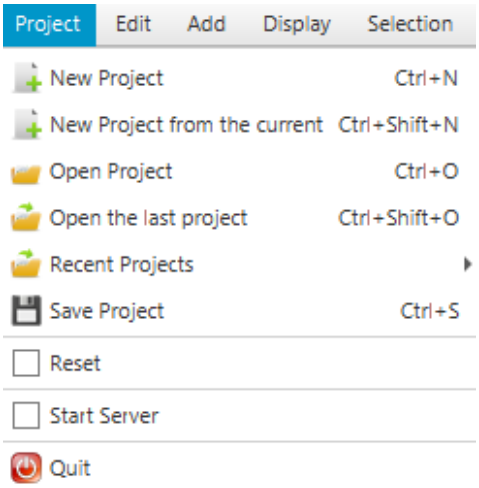


Fig. 6.4. General view of the Project tab of the main menu

– Edit (see Fig. 6.5). It contains the following standard functionality: Undo – undo the action; Redo – view the cancelled action; Duplicate – duplicate selected objects on the map; Delete – delete any selected object on the map.

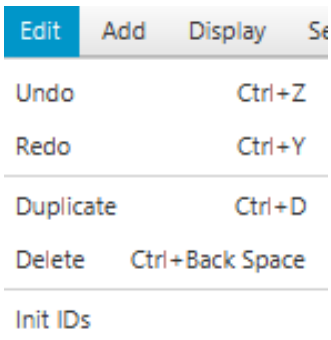


Fig. 6.5. General view of the Edit tab of the main menu

– Add (see Fig. 6.6): Add Sensor Node – add a sensor node (an object that can

detect any event, send and receive data); Add Directional Sensor Node – add a directional sensor node; Add Base Station (Sink) – add a base station node, which is similar in principle to a sensor node, except that it has an infinite battery; Add Gas (Analog Event) – add an analogue sensor node that allows for generating a continuous set of values; Add Mobile – add a mobile node with its route; Add Marker – add a route marker; Add Weather – add a weather data generation module (only one such module can be added to the project); Add Randomly – add a randomly specified number of modules.

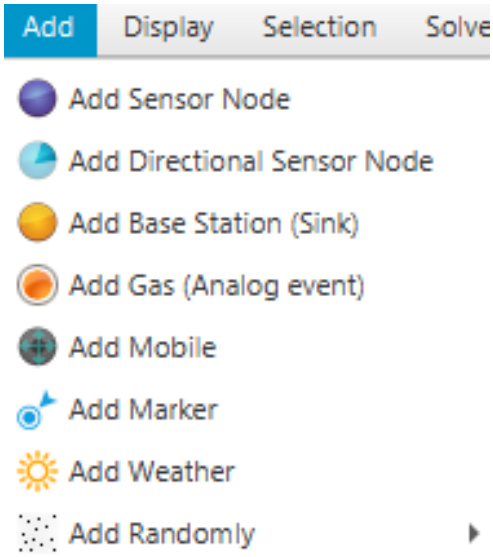


Fig. 6.6. General view of the Add tab of the main menu

– Display (see Fig. 6.7): Display / Hide Details – show / hide the names of the selected sensor nodes; Display / Hide File Name – show / hide the name of the assigned SenScript file for the selected sensor node; Display / Hide Battery / Buffer levels – show / hide the battery and buffer levels of the selected sensor node; Display / Hide Radio Distances – show / hide the distances in metres between the sensors that exchange data; Display / Hide Radio Messages – show / hide the messages being sent; Display / Hide Marker Distances – show / hide distances between markers; Display / Hide Links – show / hide links between communicating sensor nodes; Display / Hide Network Arrows – show / hide arrows between communicating sensor nodes; Display

/ Hide Marker Arrows – show / hide arrows between markers; Display / Hide Buildings – show / hide buildings that have been loaded and / or created; Next Link Color and Previous Link Color – change the colour of links between communicating sensor nodes.

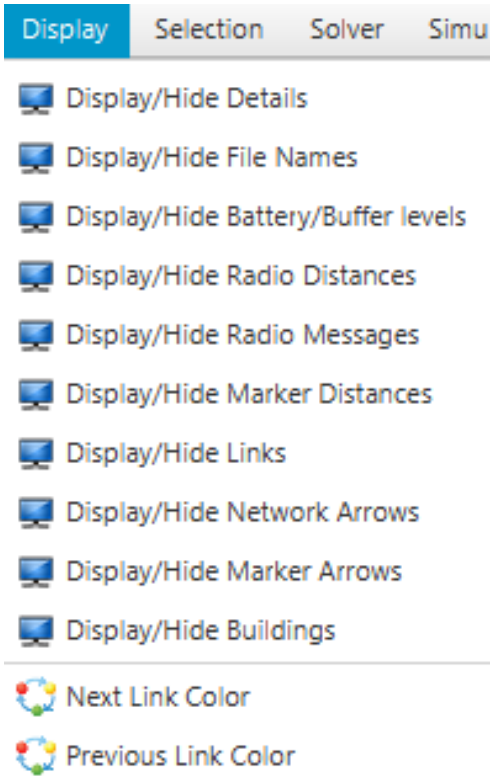


Fig. 6.7. General view of the Display tab of the main menu

– Solver: utilised to select standard algorithms for modelling IoT networks, as shown in Fig. 6.8 below.

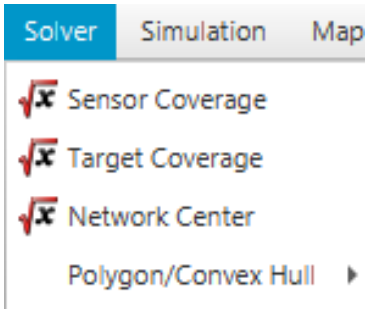


Fig. 6.8. General view of the Solver tab of the main menu

– Selection: utilised to select individual model objects or a group of them, as shown in Fig. 6.9 below.

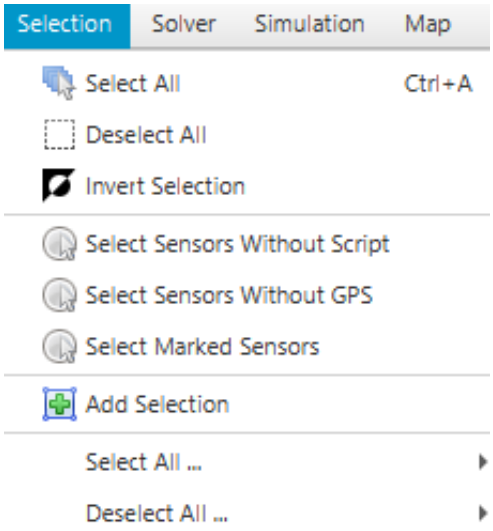


Fig. 6.9. General view of the Selection tab of the main menu

– Simulation (see Fig. 6.10). It contains the following functionality: Run Simulation – start the simulation procedure; Stop Simulation – stop the simulation procedure; Energy Consumption – display the energy consumption graph for the selected sensor nodes after the simulation procedure is completed; SenScript Window – open the SenScript window; Natural Event Generator – generate physical event values; Test Selected Mobility and Events – simulate and check the mobility of the selected model node; Test All Mobilities and Events – simulate the mobility and events of all devices; Test Sensor Nodes Mobilities – simulate the mobility of all sensors; Test Mobiles Mobilities – simulate the mobility of all mobile devices; Stop Testing Mobilities and Events – stop testing mobilities and events; Initialise All – initialise the simulation environment and information displayed during the simulation; Run Fault Injection – generate faulty sensor nodes randomly; Stop Fault Injection – stop the process of generating faulty sensor nodes; Generate Arduino Code – automatically generate the appropriate C++ code for the Arduino microcontroller platforms in accordance with each sensor node based on the SenScript code.

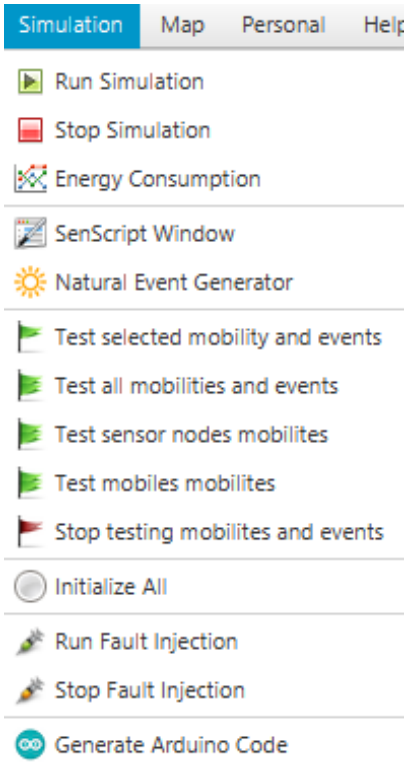


Fig. 6.10. General view of the Simulation tab of the main menu

– Map (see Fig. 6.11): utilised to select the map on which the IoT model will be implemented. A detailed analysis of the maps available for use is shown in Fig. 6.2 above.

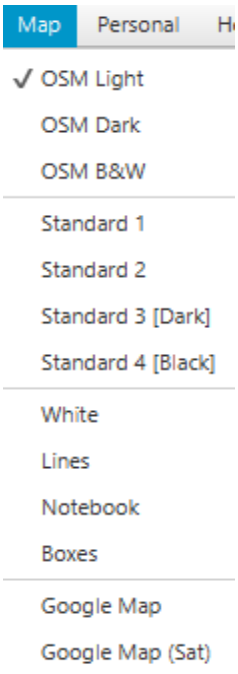


Fig. 6.11. General view of the Map tab of the main menu

– Personal: utilised to create personal SenScript functions using the built-in functionality, as shown in Fig. 6.12.

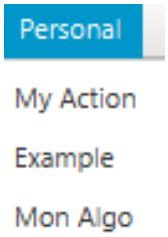


Fig. 6.12. General view of the Personal tab of the main menu

– Help: contains links to reference information and a general information window about the CupCarbon software, as shown in Fig. 6.13.

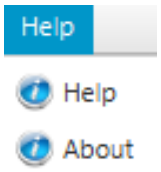


Fig. 6.12. General view of the Help tab of the main menu

Therefore, the above textual description and the corresponding graphical accompaniment (see Figs. 6.3 – 6.13) represent the key functional set of the main menu of the CupCarbon software.

The toolbar is shown in Fig. 6.14 and is utilised to access the main actions that can be implemented in the CupCarbon software.

Fig. 6.14. General view of the CupCarbon toolbar

This toolbar contains seven main functional parts, namely:

– Project part (see Fig. 6.15): allows for creating a new project, opening a project (last project) and saving a project.

Fig. 6.15. General view of the Project part of the toolbar

– Object part (see Fig. 6.16): allows for adding objects to create an IoT model on the map.

Fig. 6.16. General view of the Object part of the toolbar

– Connections part (see Fig. 6.17): allows users to create regular connections or connections based on radio wave propagation between sensor nodes and can also be used to calculate the visibility of sensor nodes in the radio signal transmission mode with the presence of buildings.

Fig. 6.17. General view of the Connections part of the toolbar

– Simulation part (see Fig. 6.18): allows users to start / stop the simulation process, build graphs of energy consumption functions, open SenScript and model event generator windows.

Fig. 6.18. General view of the Simulation part of the toolbar

– Magnetism part (see Figure 6.19): allows users to add objects to an invisible grid. When utilising this functionality, it is recommended to use the following model map setting – Mean Gray Cell Background.

Fig. 6.19. General view of the Magnetism part of the toolbar

– Selection part (see Fig. 6.20): allows users to select all model objects, all sensor nodes, all markers, deselect model elements, or invert the selection.

Fig. 6.19. General view of the Selection part of the toolbar

Therefore, the above textual description and the corresponding graphical accompaniment (see Figs. 6.14 – 6.20) represent the main functional set of the CupCarbon toolbar.

The parameter menu is utilised to set the main parameters and characteristics of the components of the created IoT network in an interactive mode, as shown in Fig. 6.21.

Fig. 6.21. General view of the Parameter menu of CupCarbon

This menu is characterised by the following functionality:

– Network information (see Fig. 6.22): contains certain information about the IoT network, such as the number of sensors, the number of dedicated sensors, the number of isolated sensors and others.

– Devices & Objects (see Fig. 6.23): contains two tabs: the first ‘Devices List’ is a list of devices that are allowed to be selected on the map by their features and

names; the second ‘Selection’ allows users to select / deselect objects by their type. It is also possible to select objects by their addresses or identifiers, which can be entered as a list of numbers in the corresponding text field.

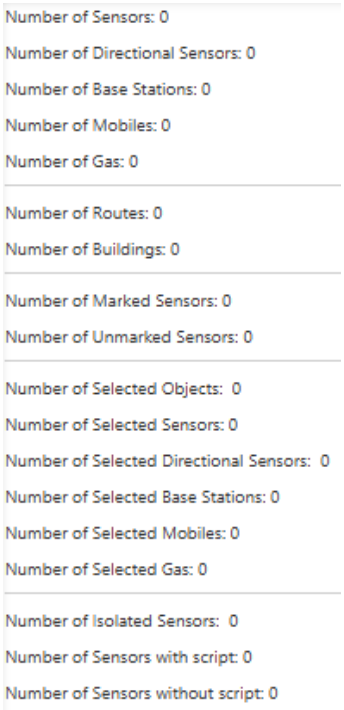
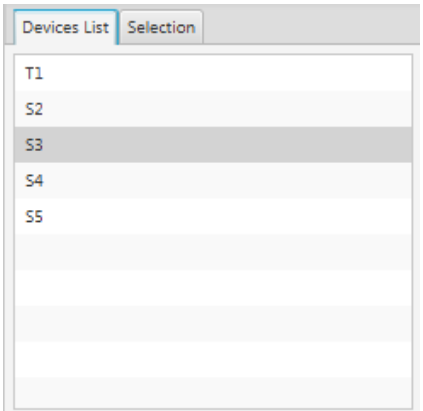
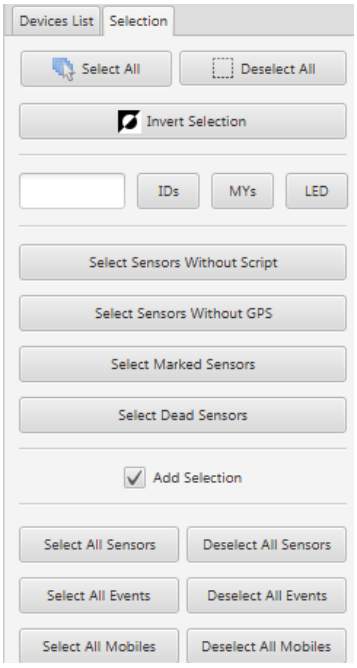


Fig. 6.22. General view of the Network information tab of the Parameter menu



a) Devices List tab



b) Selection tab

Fig. 6.23. General view of the Devices & Objects tab of the Parameter menu

– Device Parameters (see Fig. 6.24): allows users to change the parameters of

the selected objects, namely: Script File – assignment of the SenScript file; GPS File – assignment of the route file; Natural Event File – assignment of the physical event file; Id – setting the identifier; Longitude – assignment of longitude; Latitude – assignment of latitude; Elevation – assignment of the object's height; Radius – setting the range of a wireless detector; Sensor Radius – setting the range for a group of detectors; Energy Max – initial battery energy; Sensing Consumption – energy consumed by the detector; UART D/Rate – the data exchange rate using the serial port; Drift (Sigma) – the clock drift indicator; The Coverage of a Sensing Unit – the coverage area of the sensor module; The Direction (Rotation) of a Sensing Unit – the direction of rotation of the sensor unit.

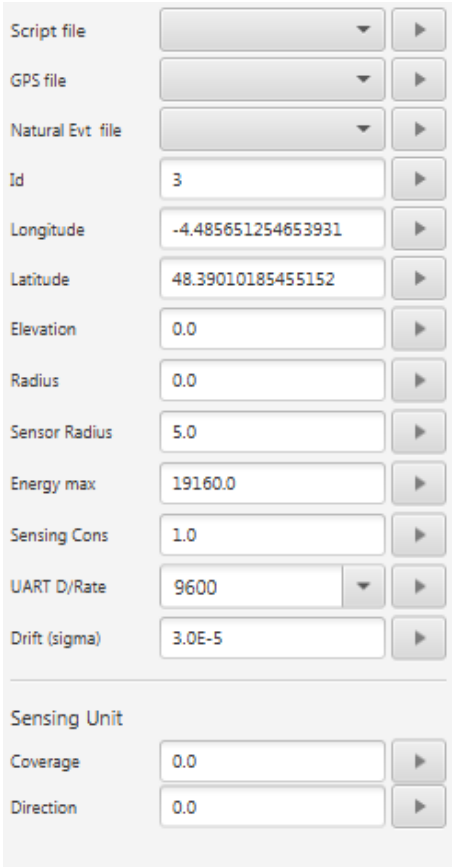
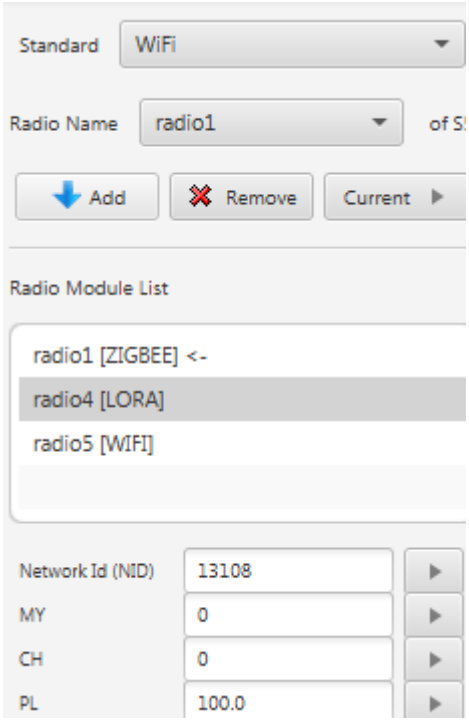


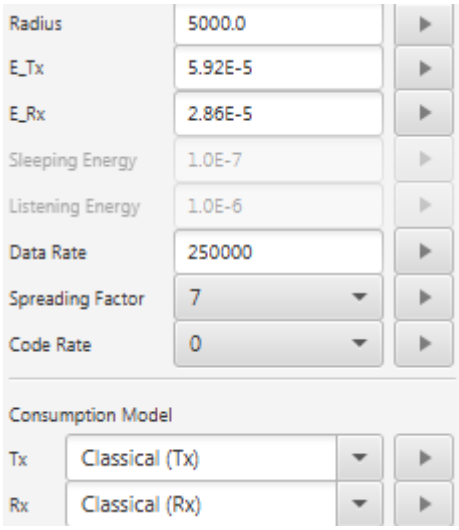
Fig. 6.24. General view of the Device Parameters tab of the Parameter menu

– Radio Parameters (see Fig. 6.25): allows users to configure the radio module parameters for the selected sensor nodes, namely: Standard – selecting the wireless module standard (ZigBee 802.15. 4, WiFi and Lora); Radio Name – selecting the name

of a particular radio module; Add – adding a radio module; Remove – removing a radio module; Current – assigning the current radio module; Radio Module List – list of available radio modules; Network ID – network identifier of the selected radio module; MY – address of the selected radio module; CH – channel of the selected radio module; Radius – range of the selected radio module; E_Tx – power consumption in transmit mode; E_Rx – energy consumption in the receiving mode; Sleeping Energy – energy consumption in the passive mode; Listening Energy – energy consumption in the channel listening mode; Data Rate – data sending / receiving speed parameter (250 Kbit/s by default); Spreading Factor – spreading factor (available only for LoRa radio modules); Consumption Model – consumer model settings.



a) Part 1



b) Part 2

Fig. 6.25. General view of the Radio Parameters tab of the Parameter menu

– Marker Parameters (see Fig. 6.26): allows users to work with the markers of IoT model as follows: Route from Markers – allows users to create a route that is located between two points on the map, which are defined by two corresponding markers; Insert Markers – set markers after the selected ones; Load Buildings – allows users to add polygons to the model that correspond to buildings on the map; File Name

– assign a name to the saved file of the corresponding route; Save – allows users to save the current route; Delete – allows users to delete the file corresponding to the selected route; Selecting Any Route – allows users to select a route from the list; Draw All Routes or a Selected Ones – allows users to display all or selected routes; Hide All Routes – allows users to hide all routes.

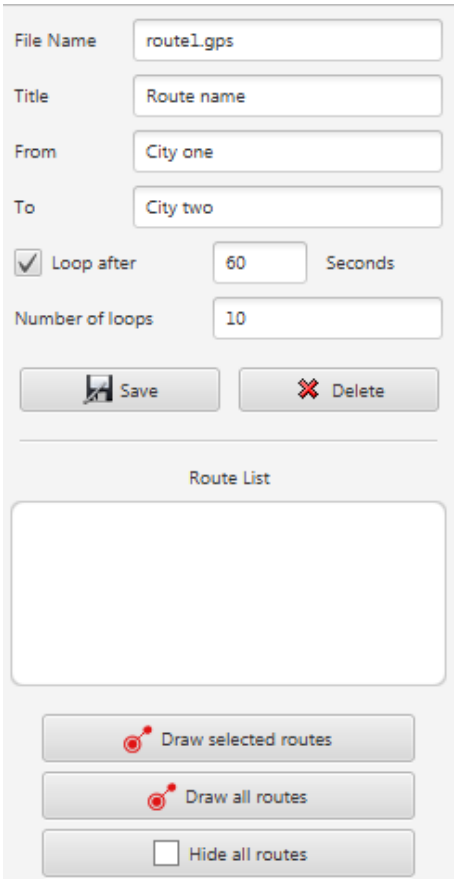


Fig. 6.26. General view of the Marker Parameters tab of the Parameter menu

– Simulation Parameters (see Fig. 6.27): utilised to configure the parameters of the simulation procedure and contains the following configuration tabs: SenScript – opens a window for creating SenScript scripts; Simulation Time – allows users to set a parameter of the simulation duration; Simulation Speed – allows users to set a parameter of the simulation speed; Arrow Speed – allows users to set a parameter of the delay when sending / receiving messages; Run Simulation – allows users to start simulation; Stop Simulation – allows users to stop simulation; Symmetrical Links – allows users to set symmetrical links between events of the sensor nodes; Visibility –

allows users to configure the visibility of the mobile sensor nodes; Log – generates a Log-file of the simulation procedure; Results – allows users to visualise the simulation results in the form of energy consumption graphs; Mobility / Events – this option must be activated if the events and mobility parameters are to be taken into account during the simulation; Clock Drift – this option must be activated if the clock drift parameter is to be taken into account during the simulation; ACK (type) – this option must be enabled if the event acknowledgement parameter is to be taken into account during the simulation; Show – the option to display ACK events during the simulation; MAC Layer – activation of a simplified MAC layer based on the probabilistic nature of events.

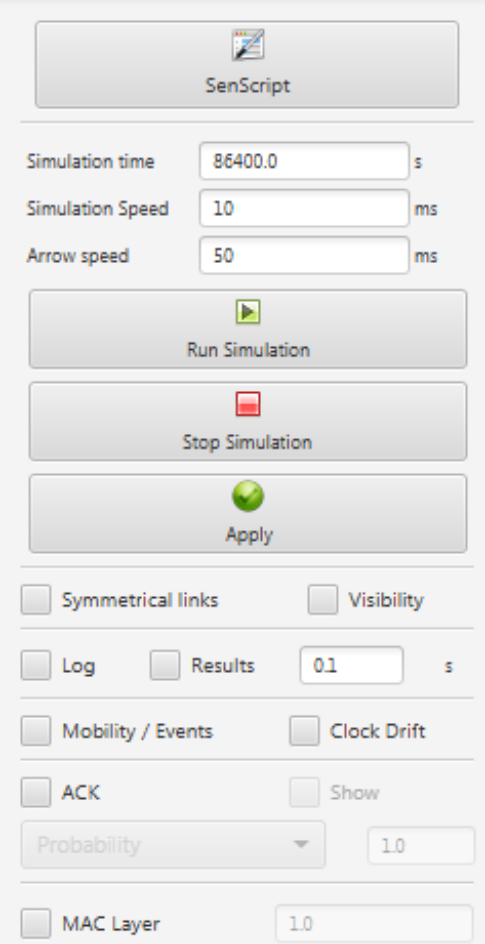


Fig. 6.27. General view of the Simulation Parameters tab of the Parameter menu

Therefore, the above textual description and the corresponding graphical accompaniment (see Figs. 6.21 – 6.27) represent the main functional set of menus for

setting the parameters of the CupCarbon software components.

The state bar is placed at the bottom of the main window of CupCarbon and allows users to display certain information and control the modelling process, as shown in Fig. 6.28.

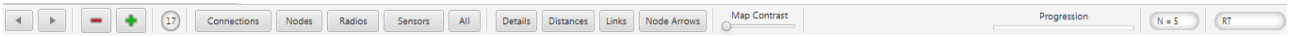


Fig. 6.28. General view of the state bar of the CupCarbon software

The main functional components of the status bar are as follows:

– Status bar navigation buttons, as shown in Fig. 6.29.

Fig. 6.29. Status bar navigation buttons

– Zoom the map as shown in Fig. 6.30.

Fig. 6.30. Map zoom buttons of the status bar

– IoT network visualisation options, as shown in Fig. 6.31.

Fig. 6.31. Functional buttons for visualising the IoT network of the status bar

– The map contrast is shown in Fig. 6.32.

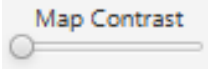


Fig. 6.32. Status bar map contrast adjustment slider

– Visualisation of the progress of the simulation time, the number of sensor nodes and the real simulation time, as shown in Fig. 6.33.

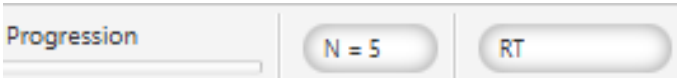


Fig. 6.33. Window for visualising the characteristics of the modelling process

Therefore, the above textual description and the corresponding graphical accompaniment (see Figs. 6.28 – 6.33) represent the main functional set of the CupCarbon state bar.

The console is utilised to display certain information messages that are useful for users when modelling the Internet of Things networks and systems. The console panel (see Fig. 6.34) is divided into two parts: the first one (located on the left) is used to display messages about the modelling progress and it is also possible to display messages from sensor nodes using the 'cprint' command in the SenScript editor; the second one (located on the right) is used to display errors during the modelling in online mode.

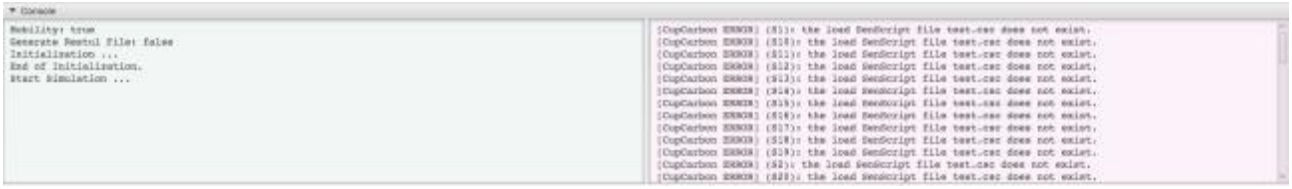


Fig. 6.34. General view of the console window of CupCarbon

Therefore, the console window is utilised by users when testing and validating the developed models of IoT networks and systems to collect and analyse information on the correctness of the modelling process.

6.2 Components for Creating CupCarbon IoT Models

The main types of components for creating IoT network and system models

available to developers in the CupCarbon software include: Sensor Node, Directional Sensor Node, Base Station, Gas Analog Events, Mobile, Marker and Weather.

The Sensor Node is the main object of CupCarbon. It contains the following main parts: a radio module, a sensing element and a battery. In the centre of the sensor node is its name – S, followed by the numerical designation of its identifier. On the right side of the node name is a number in square brackets, which is 0 by default. This number corresponds to the address of this sensor node. If a sensor node is assigned to a corresponding SenScript, its name will be displayed in grey above its name and information messages will be displayed below the name of the corresponding node, as shown in Fig. 6.35.

Fig. 6.35. External view of the Sensor Node

All settings of the sensor node are performed using the component settings menu, as described above.

The Directional Sensor Node is similar to the typical sensor node in its principle of operation, but contains a different type of sensor unit, which is directional. The angle of the cone, which is responsible for the direction and area of signal transmission, can be changed (see Fig. 6.36) using the SenScript editor or the keyboard:

– ‘)’ and ‘(’ – to increase or decrease the range (distance) or ']' and '[' – for similar actions in a more precise mode;

– ‘p’ and ‘o’ – to rotate (left or right) the detector coverage range;

– ‘P’ and ‘O’ – to increase or decrease the detector coverage area.

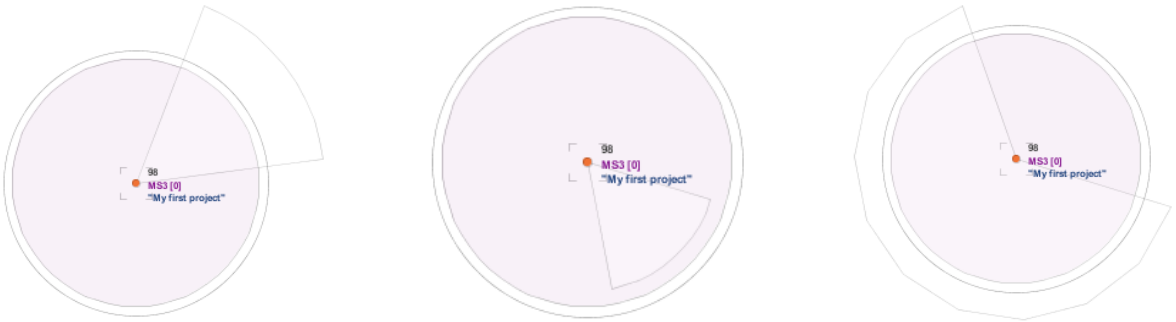


Fig. 6.36. External view of the Directional Sensor Node

All settings of the directional sensor node are performed using the component settings menu, as described above.

The Base Station is similar to the Sensor Node, except that it has an infinite battery life. The external view of the Base Station is shown in Fig. 6.37.

Fig. 6.37. External view of the Base Station

All settings of the directional sensor node are performed using the component settings menu, as described above.

The Gas Analog Events allows for generating analogue values. Its main aim is to simulate random or specified values. The external view of the gas analog events is

shown in Fig. 6.38.

Fig. 6.38. External view of the Gas Analog Events

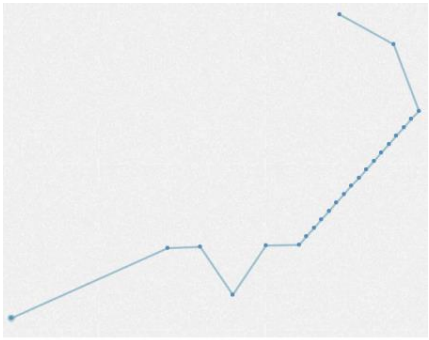
When simulating in the mobile mode, user must first activate the Mobility / Events option in the Simulation Parameters section of the settings menu. All other settings of the Gas Analog Events are made using the component settings menu, as described above.

The Mobile is aimed at simulate moving objects that move along a trajectory defined by markers, as described below. The external view of the mobile is shown in Fig. 6.39.

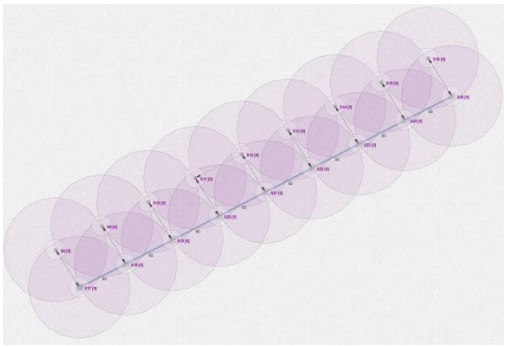
Fig. 6.39. External view of the Mobile

All settings of such a mobile node are performed using the component settings menu, similar to all the above models of IoT network devices.

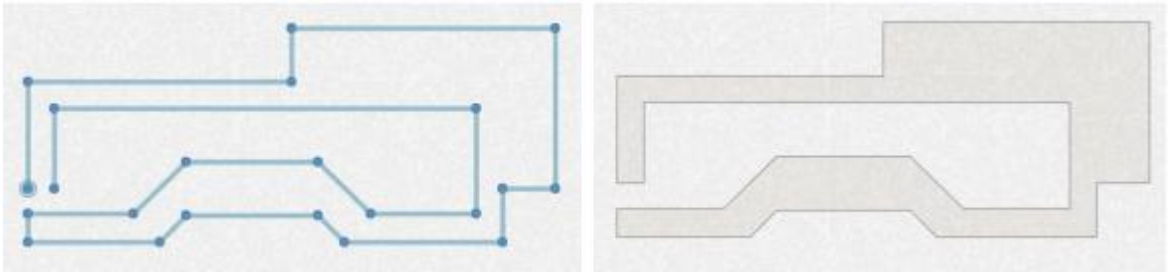
The Marker can be used for many purposes of creating objects of various IoT systems and networks, such as a sensor network, a mobile node or a creating trajectory (see Fig. 6.40).



a) trajectory of the mobile node



b) sensor network



c) buildings

Fig. 6.40. Options for using the Marker

The Weather is utilised for generating any weather parameters, as shown in Fig. 6.41.

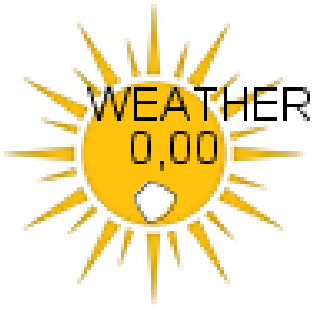


Fig. 6.41. External view of the Weather

It is worth noting that only one such element can be added to a project. Data can be generated using the Event Generator presented above from the simulation part  in the CupCarbon toolbar.

6.3 Function Keys and Buttons of the CupCarbon IoT

The functional purpose of the keyboard keys used to create and configure computer models of IoT networks in the CupCarbon software is shown in Table 6.1.

Table 6.1. Functional assignment of the keyboard keys when working in the CupCarbon software

Key	Assignment
‘a’	– show / hide arrows between sensor nodes that interact with each other;
‘A’	– show / hide link arrows between markers;
‘x’	– show / hide distances in connections between interacting sensor nodes;
‘X’	– show / hide distances in links between markers;
‘w’	– select all sensor nodes → select all markers → deselect;
‘l’	– show / hide connections between interacting sensor nodes;
‘v’	– change the colour of radio links;
‘d’	– show/hide object names and corresponding messages displayed by the print command;
‘n’	– show / hide object file names;
‘r’	– show / hide certain parameters on the map;
‘R’	– show / hide all routes;
Direction keys	– move around the map;
‘Del’	– delete an object;
‘Esc’	– deselect all components.

The functional purpose of the mouse buttons (LMB – left mouse button, RMB – right mouse button) used to create and configure computer models of IoT networks in

the CupCarbon software is shown in Table 6.2.

When working in the CupCarbon IoT modelling software, these function keys and buttons allow users to configure the parameters of both individual components and the model as a whole.

Table 6.2. The functional purpose of the mouse buttons when working in the CupCarbon software

Button / pressing mode	**Assignment**
double-click LMB on the selected sensor node	– open the device settings window;
pressing the RMB on the map	– deselect all components and stop the process of adding objects;
pressing the LMB on the map	– deselect all components or add an object;
drag and drop with the RMB pressed on the map	– draw a rectangle with multiple objects selected.

6.4 Operators, Instructions and Functions of the CupCarbon IoT Scripts

When creating and debugging scenarios and algorithms for the operation of IoT systems and networks in the CupCarbon software, specialised SenScript scripts are used, which are implemented using special operators, commands, functions and instructions, as described below.

– *and (logic)* – logical ‘and’.

Example: and x a b.

– *angle / angle2* **–** forming an angle using edges.

Examples: angle a x1#y1 x2#y2 x3#y3;

angle2 a x1 y1 x2 y2 x3 y3.

– *areadsensor (Analog Read Sensor)* **–** reading values from an analogue sensor.

Example: areadsensor s.

– *atch* **–** change the sensor node channel.

Example: atch 12.

– *atid* **–** change the sensor node ID.

Example: atid 1111.

– *atpl* **–** change the sensor node signal strength, the numerical value is set as a percentage of the nominal value.

Example: atpl 60.

– *atmy* **–** change the 'my' identifier to a numeric value.

Example: atmy 4.

– *atnd* – determine the number of neighbouring sensor nodes.

Example: atnd 5.

– *atnid* – change the network address.

Example: atnid 3333.

– *atget* – get the parameters of the sensor node.

Examples: atget id x;

atget my x;

atget ch x.

– *band* – binary logical 'and'.

Example: band x 1010 1100.

– *battery* **–** setting the battery level of the wireless sensor node in Joules.

Example: battery set 555.

– *bnot* – binary logical ‘no’.

Example: band x 1010.

– *bor* – binary logical ‘or’.

Example: bor x 1010 1100.

– *buffer* – determining the size of the buffer in bits.

Example: buffer x.

– *bxor* – binary logical exclusive disjunction.

Example: bxor x 1010 1100.

– *cbuffer* – clearing the buffer.

Example: cbuffer.

– *charat* – writing a character located at the specified word index to a variable.

Example: charat c "hello" 1.

– *conc* – writing the result of combining characters to a variable.

Example: conc x a b.

– *cprint* – printing the result in the console.

Example: cprint "hello".

– *dec* – decrement operation.

Example: dec x.

– *delay* – delay, the value of which is set in milliseconds.

Example: delay 1000.

– *distance* – setting the Euclidean distance between the current sensor node and the communication node of the sensor with the corresponding id.

Example: distance x id.

– *dreadsensor* – reading values from the digital sensor (*x=1* if the sensor detects an event from the object, *x=0* otherwise).

Example: dreadsensor x.

– *drssi* – converting the value to the Euclidean distance between the current sensor and the last sensor sending a signal to the node.

Example: drssi x.

– *egde* – mark (if *a=1*) or unmark (if *a=0*) the link between the current sensor node and the node with ID x.

Example: edge a x.

– *for end* – loop design.

Example:
```
for i 0 10 1
print "i = " i
delay 1000
set i i+1
end.
```

– *getinfo* – if the mobile device is detected by the sensor, it returns to the variable *p* the information generated on the basis of the mobile device identifier and coordinates in the format: *id#longitude#latitude*.

Example: getinfo p.

– *getpos* – a function that returns to the *pos* variable data on the position of the

detector in the format *#longitude#latitude*.

Example: getpos pos.

– *getpos2* – a function that returns to the variables *x* and *y* the data on the position of the detector in the format *x=longitude* and *y=latitude*.

Example: getpos2 x y.

– *goto* – a function that returns a string of code with the specified number.

Example: goto 5.

– *hash* – a function that assigns the corresponding hash code to a variable.

Example: hash x "hello".

– *if [else] end* – conditional operator.

Example:
```
if (x==1)
mark 1
else mark 0
end.
```

– *inc* – incremental operator.

Example: inc x.

– *int* – an operator for assigning an integer value *a* to the variable *x*.

Example: int x a.

– *kill* – elimination of the current network node with a given probability, for example, due to a low battery.

Example: kill 0.3.

– *led* – setting the colour number *a* for node *x*.

Example: led x a.

– *length* – returning the length of the word to the variable *v*.

Example: length v "hello".

– *loop* – start an infinite loop.

Example: loop.

– *mark* – marking (1) – sets the marking or removes the marking (0) of the sensor node.

Example: mark 1.

– *math* – mathematical operators between variables *x* and *y* (*f*={sqrt, sin, cos, tan, asin, acos, atan, abs, pow, log, log10, exp}).

Example: math tan y 3.14.

– *max* – an assignment of the maximum between *a* and *b* to the variable *x*.

Example: max x a b.

– *min* – an assignment of the minimum between *a* and *b* to the variable *x*.

Example: min x a b.

– *move* – a function that moves the sensor node to a position with latitude (*x*), longitude (*y*) and height (*z*) at a speed (*t*) m/s.

Example: move x y z t.

– *not (logic)* – inversion.

Example: not x a.

– *nth* – writing the i^{th} value of *t* to the variable *v* in the format of a string separated

by &.

Example: nth v i t.

– *or (logic)* – logical ‘or’.

Example: or x a b.

– *pick* – reading the data *x* from the buffer, the read data is not deleted from the buffer.

Example: pick x.

– *printfile* **–** a function to add a text string to the file generated during the simulation in the results directory with the name of the corresponding sensor node.

Example: printfile "hello".

– *print* – a function of displaying information messages, which has several possible formats.

Examples: print "hello" → *result* hello;

print "i = " i → *result* i= value i;

print x → *result* value x.

– *radio* **–** selecting the radio module with the specified name, which means that any command to send data will be executed after the completion of the specified instruction in the corresponding radio module that will send information messages.

Example: radio name_of_radio_module.

– *rand* – a function for generating random values.

Example: rand x.

– *randb* – a function of generating random values in a specified range.

Example: randb x 2 6.

– *rdata* – a function of generating a message *p*, which is formed using the command data with the # delimiter.

Example: rdata p "hello" 5→ *result* p=hello#5.

– *read* **–** assigning a value from the buffer to the variable *x*.

Example: read x.

– *receive* – a function that implements waiting for data to be received from the buffer and assigned to the variable *x*, as well as blocking actions if there is no data in the buffer, additionally, the parameter *t* may be set, which is responsible for the time of waiting for data, if the data is not received within the time *t*, an empty message will be returned.

Example: receive x;

receive x t.

– *rgauss* **–** generating a value based on a random Gaussian distribution.

Example: rgauss x.

– *rmove* – a function of transferring to the next point of the route created using markers (before simulation, the route must be assigned to the sensor node, the next script instruction will be executed in *t* milliseconds, the mobility function must be activated using the tick box in the simulation parameters window).

Example: rmove t.

– *rotate* – a function of rotation by *2x* units of the directional node of the wireless sensor (the specified script instruction will be executed in *t* milliseconds).

Example: rotate x t.

– *route* – a function of changing the route of the sensor node to the route named *route1* after intersection with it.

Example: route route1.

– *rscript* **–** load the *floodint.csc* script when reinitialising software variables.

Example: rscript flooding.

– *sadd* **–** add the value *x* to the stack *t*, which has the format of a string separated by the character &.

Example: sadd x t.

– *script* **–** load the *flooding.csc* script without re-initialising software variables.

Example: script flooding.

– *send* **–** a function of sending information messages in various formats.

Examples: send "hello" 2 → *sending 'hello' to the sensor node with id=2*;

send p 2 → *sending the value of the variable p to the sensor node with id=2*;

send p → *sending the value of the variable p in broadcast mode;*

send p * 3 → *sending the value of the variable p in broadcast mode, except for the detector with id=3;*

send p 0 3 → *sending the value of the variable p to the sensors with the address identifier MY=3;*

send p 0 0 5 → *direct sending of the variable p to the sensors with id equal to 5 (similar to GPRS/3G/4G modes).*

– *set* – a function of assigning a value.

Examples: set x 5 → *x=5*;

set x "hello" → *x=hello*;

set z x*5 → *z=x*5*.

– *simulation* – a function of changing the visualisation speed of the simulation

process.

Example: simulation 50 200.

– *spop* – get the first value from the stack *t* (in the format of a string separated by &) and write it to the variable *v*.

Example: spop v t.

– *stop* – stop the execution of the simulation scenario.

Example: stop.

– *tab* – a function of creating a matrix *t* of dimension *a* by *b*.

Example: tab t a b.

– *tget* – a function of writing to the variable *x* of the matrix *t* of dimension *a* by *b*.

Example: tget x t 2 5.

– *time* – a function of assigning the current simulation time to the variable *x*.

Example: time x.

– *tset* – a function of assigning the value *a* to the element of array *t* with index *i*, *j*.

Example: tset a t i j.

– *vdata* – a function for converting tokens separated by the # symbol into a vector of these tokens named *v*.

Example: vdata v 10#24#3 → *v[0]=10, v[1]=24, v[2]=3*.

– *vec* – a function of creating a vector *v* of dimension *a*.

Example: vec v a.

– *vget* – a function of writing an element of vector *v* with index *i* to the variable *x*.

Example: vget x v i → *x=v[i]*.

– *vset* – a function of assigning the value *x* to the element of vector *v* with index *i*.

Example: vset x v i → *v[i]=x*.

– *wait* – a function of waiting for data to be received from the buffer in *t* ms (if the *t* parameter is not specified, then wait until the data is received from the buffer).

Examples: wait;

wait t.

– *while end* – loop design.

Example:
```
set i 0
while ((i<5) && (i>=0))
print "i = " i
delay 1000
set i i+1
end.
```

– *xor (logic)* **–** a logical function of exclusive disjunction (addition modulo two).

Example: xor x a b.

6.5 Individual Practical Works

6.5.1 Practical Work No. 1. Development and Testing of Basic Algorithms for the Functioning of IoT Network Components

The aim is to reinforce theoretical knowledge and develop practical skills in the

development, testing and analysis of algorithms for the functioning of IoT systems and network components using computer modelling approaches in the CupCarbon software.

Tools and software: personal computer, CupCarbon, MS Office.

The task to be performed is to develop and investigate different functional scenarios for the operation of a wireless sensor node in the CupCarbon environment.

The procedure of the work:

1. Create a new project in the CupCarbon software: Project – New project.

2. Set the map parameters – white background, as shown in Fig. 6.5.1: Map – White.

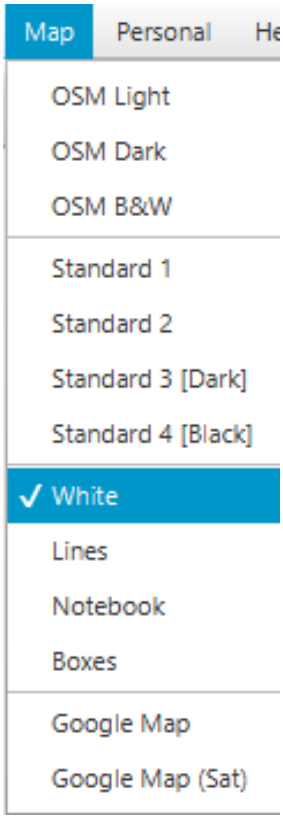
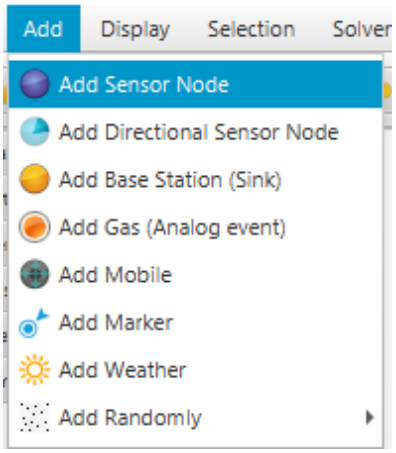


Fig. 6.5.1. Procedure for setting up map parameters

3. Add a new element to the project map – a wireless sensor node: Add – Add sensor node, as shown in Fig. 6.5.2.

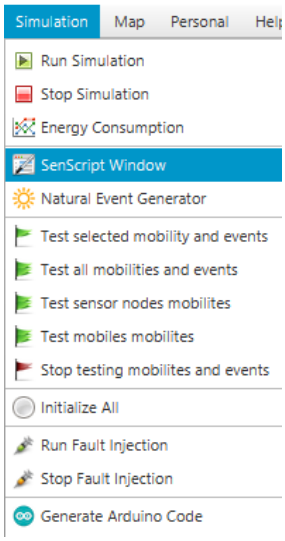


a) The procedure for adding a sensor node

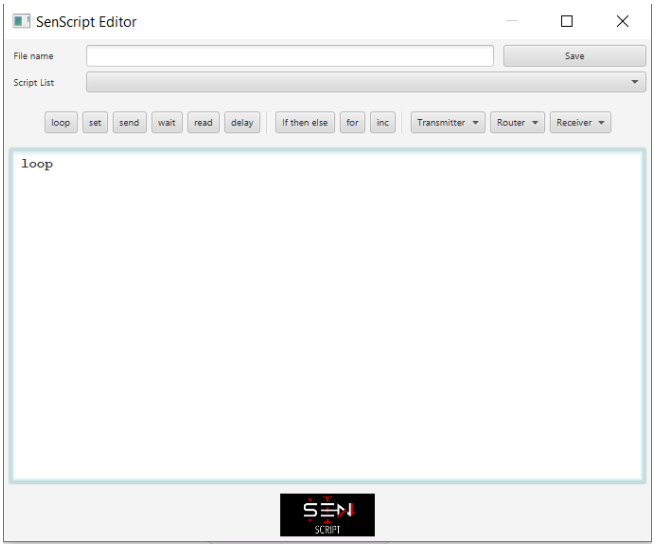
b) The result of adding a sensor node to the map

Fig. 6.5.2. Procedure and result of adding a sensor node to the map

4. Open the window for creating and editing element scripts, as shown in Fig. 6.5.3: Simulation – SenScrip Window.



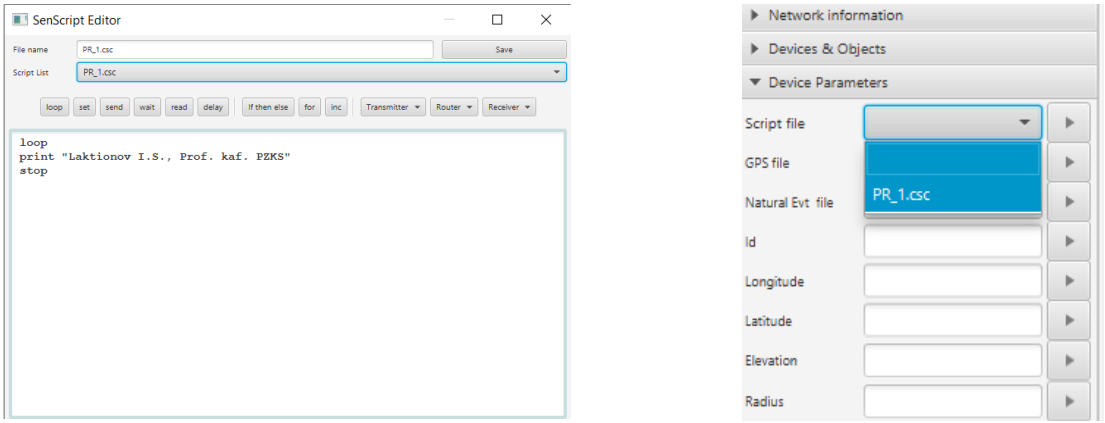
a) The procedure for starting the SenScript Window



b) The result of running the SenScript Window

Fig. 6.5.3. Procedure and result of running the script editor

5. Create and add a script to the sensor node that contains information about the student's name and surname, as well as the group of students performing the practical work, according to the sequence of actions shown in Fig. 6.5.4.



a) Creating a script b) Adding a script

Fig. 6.5.4. Procedure for creating and adding a script to a wireless sensor node

6. Configure the parameters of the modelling process and run the created computer model, as shown in Fig. 6.5.5.

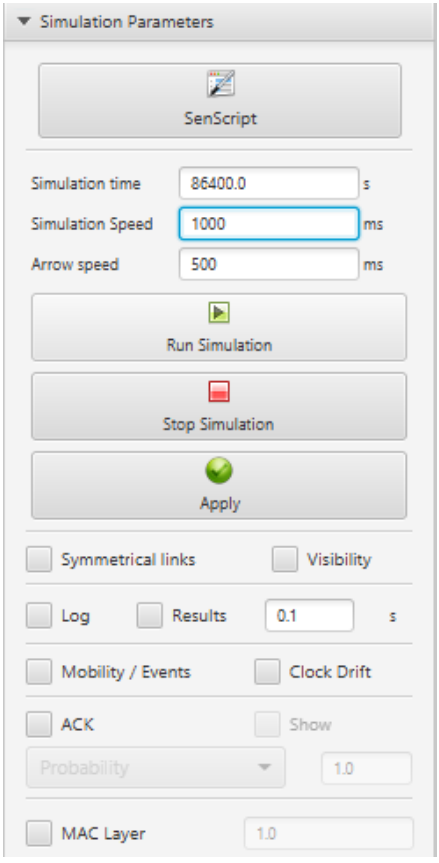


Fig. 6.5.5. Procedure for setting up and running the model

7. Visualise the results of the created model, as shown in Fig. 6.5.6.

Fig. 6.5.6. Test result of the created model

8. Save the created project.

9. Carry out research similar to that described in steps 3 – 8 for performing practical work for scripts that perform the following operations:

– $x=a+b$, where a is the penultimate number of the student card, b is the last number of the student card;

– $x=a*b$, where a is the penultimate number of the student card, b varies from 0 to 15 in increments of 1;

– $x=a$ *and (or)* b, where $a=\{0; 1\}$ and $b=\{0; 1\}$, check all possible combinations (*and* – for even options, *or* – for odd options).

Visualise the test results using the console window, as shown in Fig. 6.5.7.

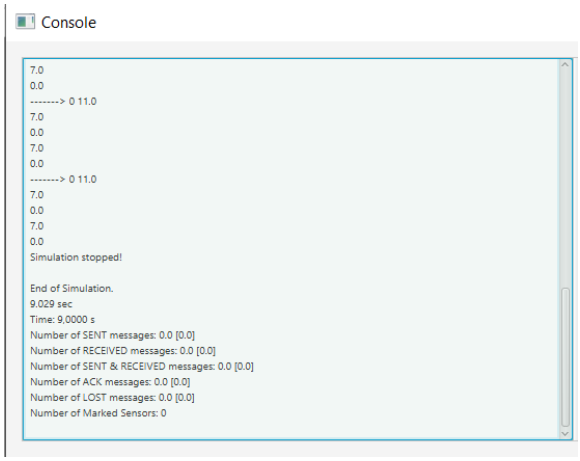


Fig. 6.5.7. An example of visualising results using a console window

10. Analyse the results obtained for the correctness of the created models and

algorithms.

Recommended contents of the report for the practical work:

1. Title page.

2. Aim and objectives of the practical work.

3. The main results obtained during the creation and testing of scripts for the sensor node, according to the relevant research points (text description and graphical visualisation).

4. General conclusions on the practical work, containing the results of critical analysis at the qualitative and quantitative levels.

Review questions:

1. What are the typical components of IoT networks that were investigated in this case study?

2. How is the model map (working field) set up in the CupCarbon environment?

3. Define the terms ‘sensor’ and ‘wireless sensor’. What is the main difference between them?

4. What functions does a wireless sensor node execute?

5. What parameters characterise the model of a wireless sensor node in the CupCarbon software?

6. Explain the sequence of actions when creating and adding scripts to the sensor node.

7. How are the parameters of the modelling process in the CupCarbon software configured?

8. What basic programming statements and functions were utilised when creating and testing scripts? Explain their syntax and reveal their basic aim.

9. What is the aim of the console in the CupCarbon software?

6.5.2 Practical Work No. 2. Development and Testing of Algorithms for the Operation of Wireless Sensor Networks

The aim is to reinforce theoretical knowledge and develop practical skills in the

development, testing and analysis of algorithms for the operation of wireless sensor networks as functional components of IoT systems and networks using computer modelling approaches in the CupCarbon software.

Tools and software: personal computer, CupCarbon, MS Office.

The task to be performed is to develop and investigate models and scenarios of wireless sensor networks functioning in the CupCarbon software, which are different in architecture and application.

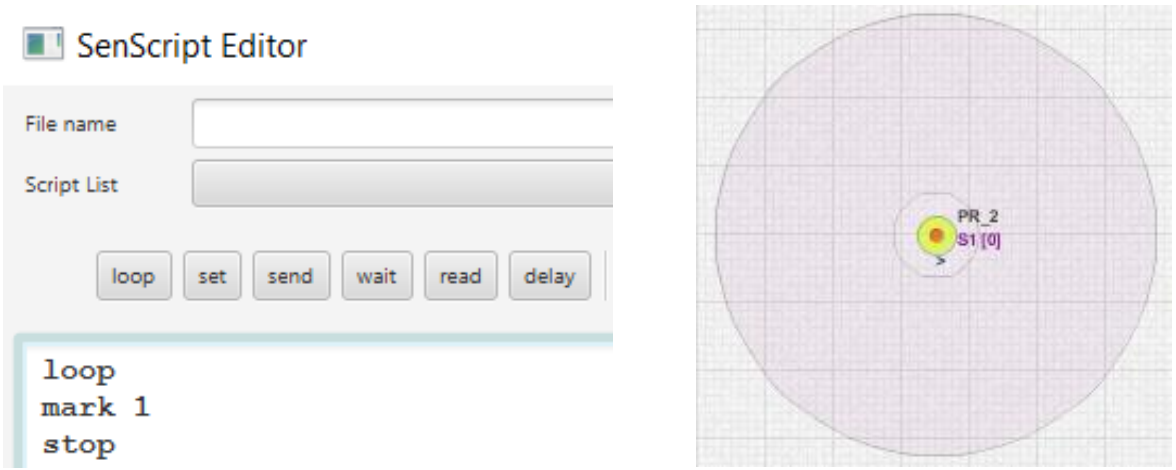
The procedure of the work:

1. Create a new project in the CupCarbon software: Project – New project.

2. Set the map parameters – white background: Map – White.

3. Add a new element to the project map – a wireless sensor node: Add – Add sensor node.

4. Perform the procedure of marking the sensor node using the mark operator, as shown in Fig. 6.5.8. Set the settings of the modelling process identical to those specified in Practical work No. 1.



a) Creating a marking script b) The result of the sensor marking

Fig. 6.5.8. Procedure for marking a sensor node

5. Describe the changes that have occurred to the sensor node compared to the baseline after the marking procedure.

6. Implement the algorithm ‘blinking LED’ (periodic marking of the sensor node) for the sensor node with a period $T=a\cdot 1000$ ms (a is the penultimate number of

the student card, if *a*=0, then *T*=1500 ms). When implementing the corresponding script, it is recommended to use standard operators: *mark* and *delay*.

7. Add a group of sensor nodes to the map in any number from 5 to 8 units, as shown in Fig. 6.5.9.

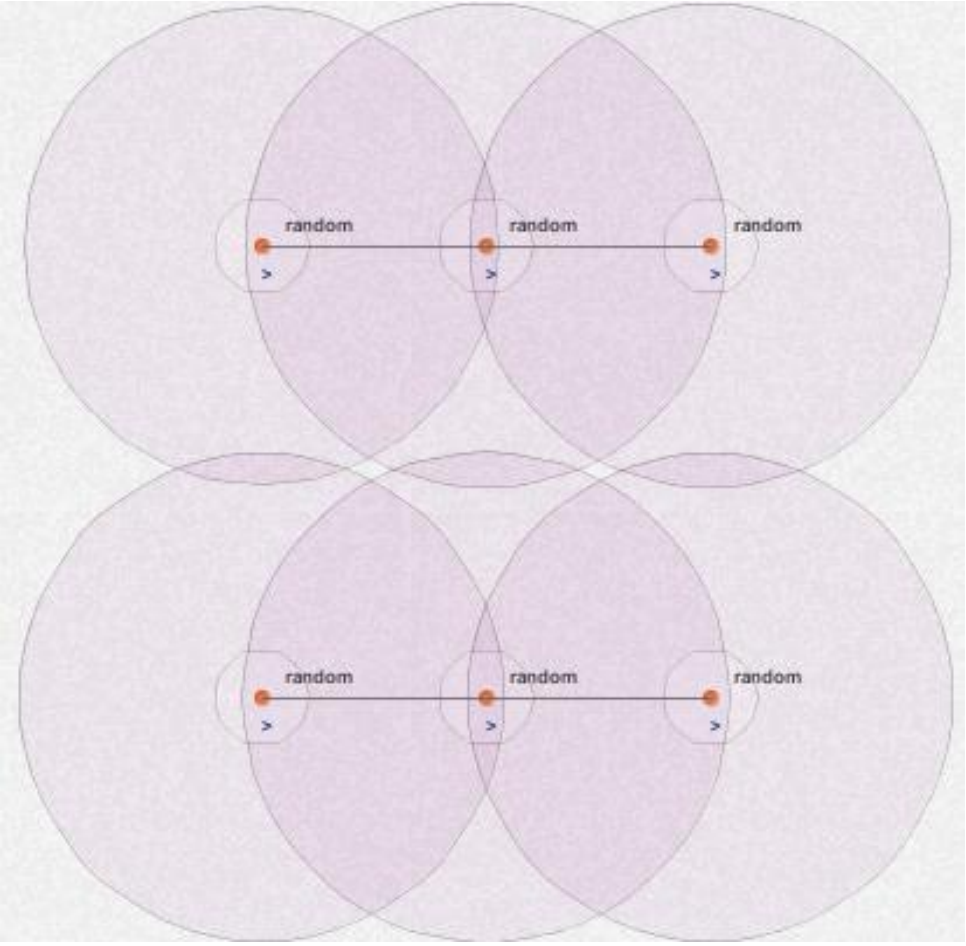


Fig. 6.5.9. An example of a model using a group of sensor nodes

8. Write a script that randomly marks the sensor nodes from the generated group. When implementing the corresponding script, it is recommended to use standard operators: *rand*, *if [else] end*, *mark* and *delay*.

9. Add the developed script to the wireless sensor nodes in the created group.

10. Run the simulation of the implemented script and describe the results. An example of a typical simulation result is shown in Fig. 6.5.10.

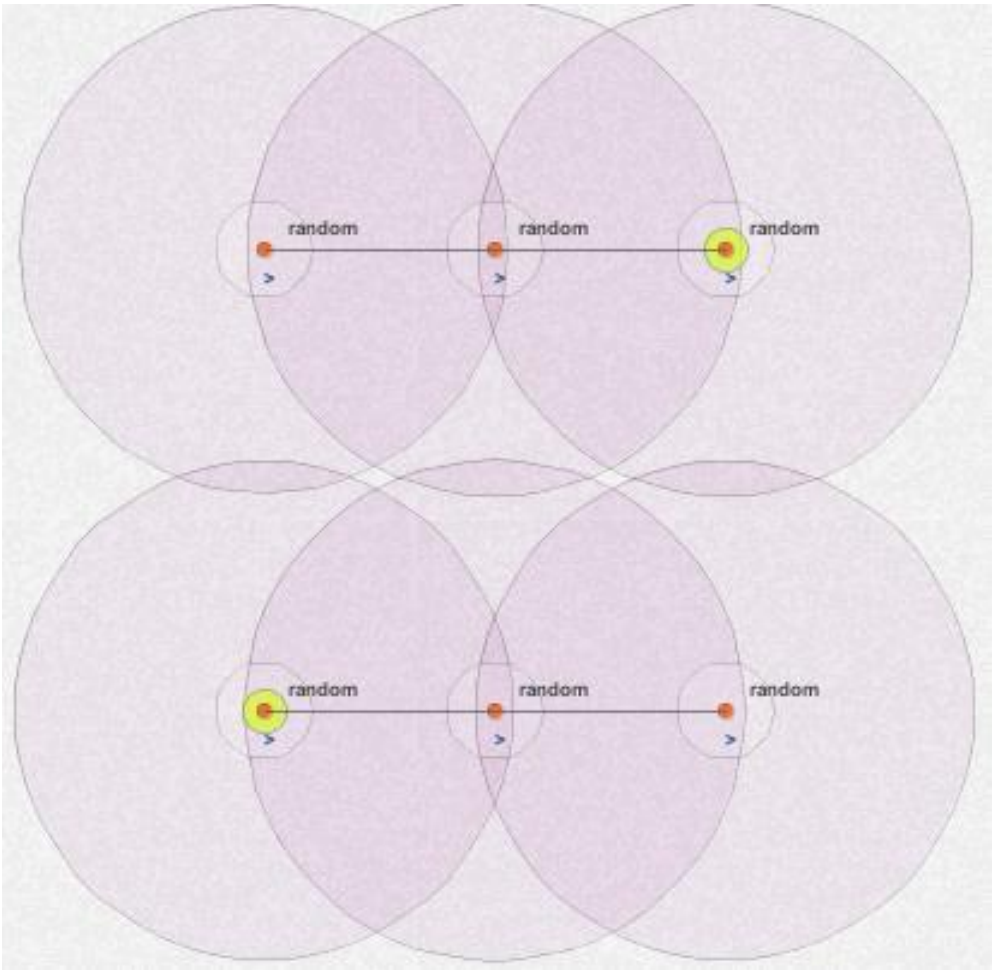


Fig. 6.5.10. Typical view of the modelling result according to steps 7 – 9

11. Analyse the results obtained for the correctness of the created algorithms.

Recommended contents of the report for the practical work:

1. Title page.

2. Aim and objectives of the practical work.

3. The main results obtained during the creation and testing of scripts for the wireless sensor groups and networks, according to the relevant research points (text description and graphical visualisation).

4. General conclusions on the practical work, containing the results of critical analysis at the qualitative and quantitative levels.

Review questions:

1. What scenarios of wireless sensor groups and networks were investigated in this case study?

2. Explain the meaning of the term ‘wireless sensor network’.

3. What are the main programming functions that were utilised when creating and testing scripts? Explain their syntax and reveal the basic purpose.

4. How can a group of sensors be automatically added to the CupCarbon software?

5. Explain the physical meaning of the sensor marking procedure.

6.5.3 Practical Work No. 3. Development and Testing of Models and Algorithms for Data Exchange in Wireless Sensor Networks

The aim is to reinforce theoretical knowledge and develop practical skills in the development, testing and analysis of models and algorithms for sending and receiving information messages in wireless sensor networks using computer modelling approaches in the CupCarbon software.

Tools and software: personal computer, CupCarbon, MS Office.

The task to be performed is to develop and investigate models and algorithms for the exchange of information messages in wireless sensor networks in the CupCarbon software.

The procedure of the work:

1. Create a new project in the CupCarbon software: Project – New project.

2. Set the map parameters – grey background: Map – Standart 1.

3. Add two new elements of the wireless sensor node to the project map: Add – Add sensor node.

4. Place the sensor nodes so that their coverage areas overlap, as shown in Fig. 6.5.11.

5. Write and add to the first wireless sensor node a transmitter script that implements the transmission of the character (the last number of the student card), as shown in Fig. 6.5.12.

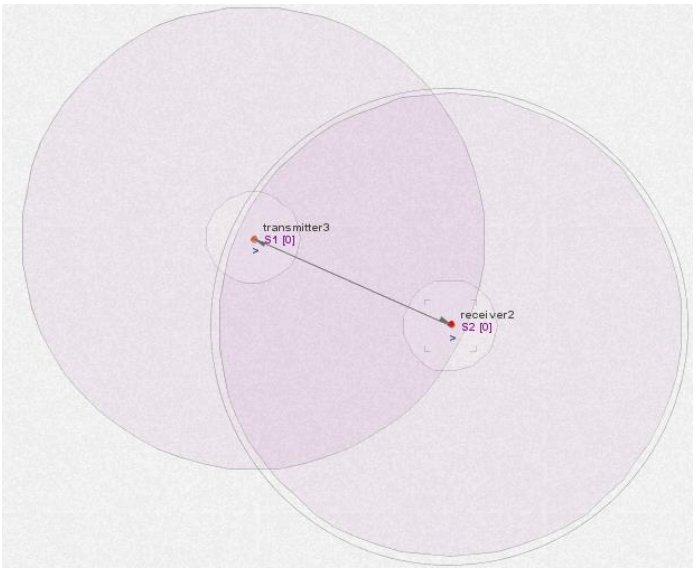


Fig. 6.5.11. An example of a couple (receiver and transmitter) of wireless sensors

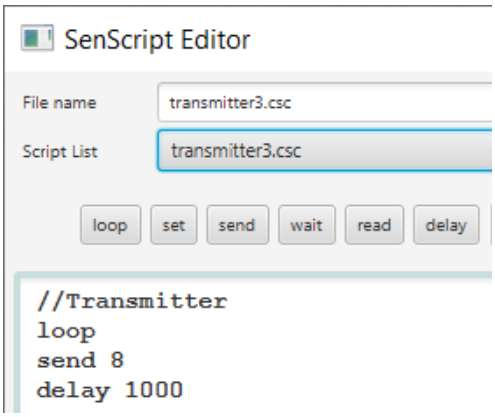


Fig. 6.5.12. An example of a script for the transmitter

6. Write and add to the second wireless sensor a receiver script that implements the reception of the character and marking of the sensor when receiving an information message, as shown in Fig. 6.5.13.

Fig. 6.5.13. An example of a script for the receiver

7. Implement the transmission of text information in the form of a single message – the student's surname.

8. Implement an algorithm for character-by-character transmission of an information message – student's own surname and academic group number with a period of T $T=a\cdot 1000$ ms (a is the penultimate number of the student card, if $a=0$, then $T=1500$ ms). It is recommended to use the *charat* operator together with a loop structure.

9. Implement the algorithm for cyclical transmission of two characters a and b (a is the penultimate number of the student card and b is the last number of the student card, if the numbers a and b are the same, then transmit the character $b+1$). The receiver should have the following functionality: if a is received, the receiver LED turns on, if b is received, the LED turns off.

10. Analyse the results obtained for the correctness of the created models.

Recommended contents of the report for the practical work:

1. Title page.

2. Aim and objectives of the practical work.

3. The main results obtained during the creation and testing of scenarios for transmitting / receiving information messages in wireless sensor networks, according to the relevant research points (text description and graphical visualisation).

4. General conclusions on the practical work, containing the results of critical analysis at the qualitative and quantitative levels.

Review questions:

1. What types of wireless sensor network scenarios were investigated in this case study?

2. What types of data can be transmitted using wireless sensor networks?

3. What basic programming functions and operators were utilised when creating and testing the scripts? Explain their syntax and basic functional purpose.

4. What function is responsible for the period of character transmission? Explain its syntax.

5. What roles can sensor nodes play in wireless networks?

6.5.4 Practical Work No. 4. Development and Testing of Models and Scenarios for Data Routing in Wireless Sensor Networks

The aim is to reinforce theoretical knowledge and develop practical skills in the development, testing and analysis of models and scenarios for routing information messages in wireless sensor networks of various structures using computer modelling approaches in the CupCarbon software.

Tools and software: personal computer, CupCarbon, MS Office.

The task to be performed is to develop and investigate models and scenarios for routing data and information messages in wireless sensor networks of various structures in the CupCarbon software.

The procedure of the work:

1. Create a new project in the CupCarbon software: Project – New project.

2. Set the map parameters – grey background: Map – Standart 1.

3. Add three new elements of the wireless sensor node to the project map: Add – Add sensor node.

4. Place the sensor nodes so that their coverage areas overlap (form a network), as shown in Fig. 6.5.14.

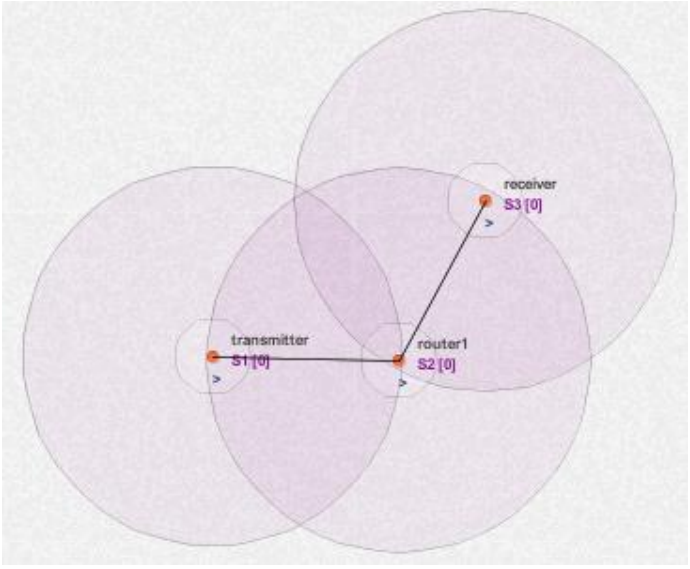


Fig. 6.5.14. An example of a wireless sensor network with an information message router

In such a network, the first sensor node (S1) acts as a transmitter, the second (S2) as a router and the third (S3) as a receiver. That is, the information message routing node must alternately execute the functions of receiving and sending information messages and / or data. The sensor nodes of the receiver and transmitter execute functions similar to those studied in Practical work No. 3 when developing and testing a group consisting of two sensor nodes.

5. Write and add to the first wireless sensor a transmitter script that implements the transmission of the character (the last number of the student card), as shown in Fig. 6.5.15.

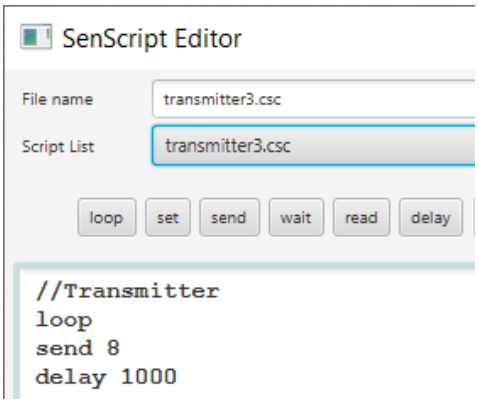


Fig. 6.5.15. An example of a script for the transmitter

6. Write and add to the second wireless sensor a router script that implements the reception of the character and its retransmission to the receiver sensor node, as shown in Fig. 6.5.16.

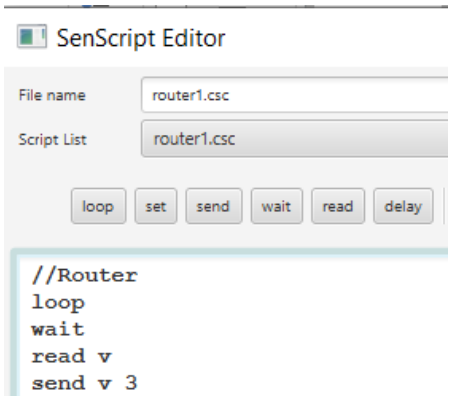


Fig. 6.5.16. An example of a script for the router

7. Write and add to the third wireless sensor a receiver script that implements the reception of the character and marking of the sensor node when receiving an information message, as shown in Fig. 6.5.17.

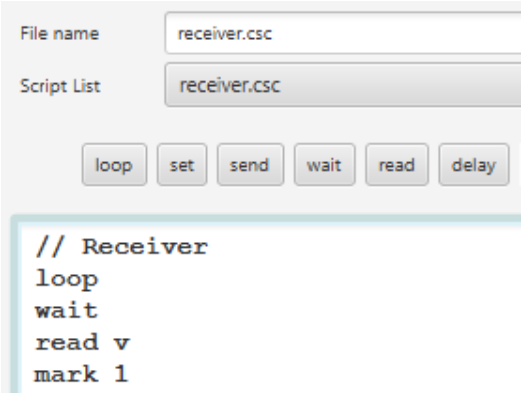


Fig. 6.5.17. An example of a script for the receiver

8. Implement an algorithm for character-by-character transmission and retransmission of an information message – student's name with a period of $T=a\cdot 1000$ ms (a is the penultimate number of the student ID card, if $a=0$, then $T=1500$ ms). The number of routers in the network is assumed to be as follows: four – if the last number of the student ID is 0, 3, 7; five – 1, 5, 8; six – 2, 4, 6, 9. When the last character of the information message is received, the option to mark the receiver must be generated.

9. Analyse the results obtained for the correctness of the created models and algorithms.

Recommended contents of the report for the practical work:

1. Title page.

2. Aim and objectives of the practical work.

3. The main results obtained during the creation and testing of scenarios for routing information messages in wireless sensor networks, according to the relevant research points (text description and graphical visualisation).

4. General conclusions on the practical work, containing the results of critical analysis at the qualitative and quantitative levels.

Review questions:

1. Define the terms ‘measurement data’ and ‘information message’.

2. What are the main functional elements utilised in the creation of wireless sensor networks?

3. What is the purpose of the transmitter, router and receiver in IoT networks?

4. What types of wireless sensor network scenarios were investigated in this case study?

5. What basic programming functions were utilised when creating and testing the scripts? Explain their syntax.

6. How was the algorithm for finding the last character in an information message implemented?

6.5.5 Practical Work No. 5. Development and Testing of Models and Scenarios for Group Data Exchange in IoT Networks

The aim is to reinforce theoretical knowledge and develop practical skills in the development, testing and analysis of models and scenarios for group data and information exchange in wireless IoT networks of various structures using computer modelling approaches in the CupCarbon software.

Tools and software: personal computer, CupCarbon, MS Office.

The task to be performed is to develop and investigate models and scenarios for group data and information exchange in wireless IoT networks of various structures in the CupCarbon software.

The procedure of the work:

1. Create a new project in the CupCarbon software: Project – New project.

2. Set the map parameters – grey background: Map – Standart 1.

3. Add five new elements of the wireless sensor node to the project map: Add – Add sensor node.

4. Place the wireless sensor nodes in such a way that their coverage areas overlap (form a network), as shown in Fig. 6.5.18. In such a network, the first sensor node (S1)

acts as a transmitter and all other nodes (S2 – S5) act as receivers.

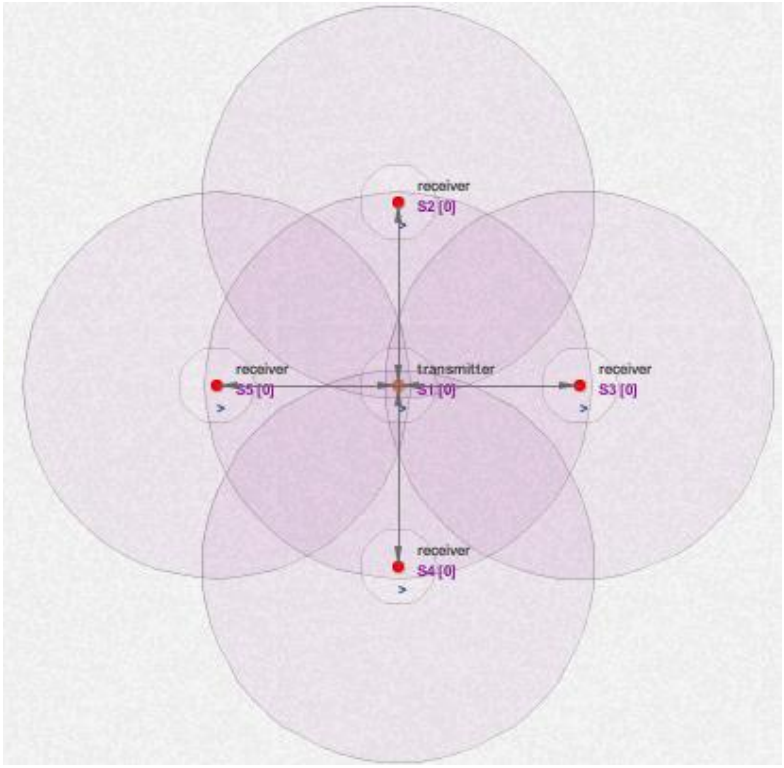


Fig. 6.5.18. An example of a wireless sensor network structure

5. Write and add to the first wireless sensor (S1) a transmitter script that implements the alternate transmission of the characters 1 and 0, as shown in Fig. 6.5.19.

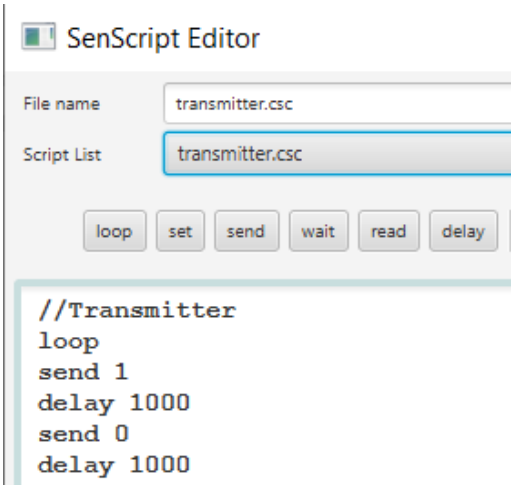


Fig. 6.5.19. An example of a script for the transmitter

6. Write and add to the second to fifth wireless sensors a receiver script that implements the reception of characters and marking (if received 1) / unmarking (if 0 is received), as shown in Fig. 6.5.20.

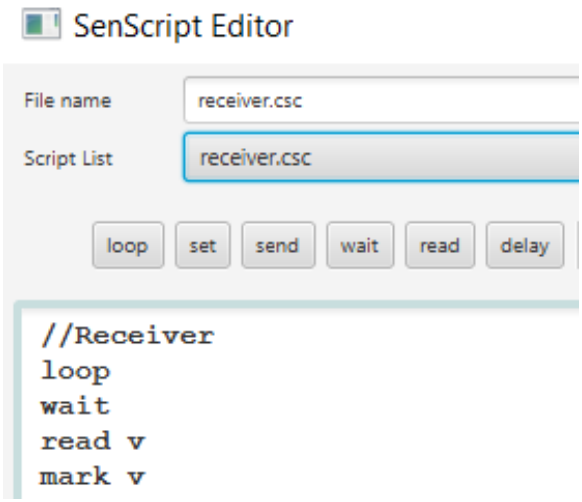


Fig. 6.5.20. An example of a script for the receiver

7. Test the implemented IoT network of wireless sensors for correct operation (simultaneous receipt of information messages by all receiver nodes and marking / unmarking of these sensor nodes).

8. Implement an algorithm for character-by-character transmission of an information message – student's surname with a period $T=a\cdot 1000$ ms (a is the penultimate number of the student ID card, if $a=0$, then $T=1500$ ms). The number of receivers in the network is assumed to be as follows: five – if the last number of the student ID card is 2, 4, 6, 9; six – 0, 3, 8; seven – 1, 5, 7. When the last character is received, the option to mark the receivers must be generated.

9. Implement the algorithm for sending messages described in step 8 for a separate group of receivers from the general network of network receivers.

To do this, in the selected sensor nodes (those that must receive information messages from the transmitter), change the MY address parameter of the sensor nodes to a fixed value (see Fig. 6.5.21, for example, MY=11). Also, when writing a transmitter script, the send function must provide a syntax that takes into account

sending information messages only to receivers with a predefined value of the MY address parameter, as shown in Fig. 6.5.22.

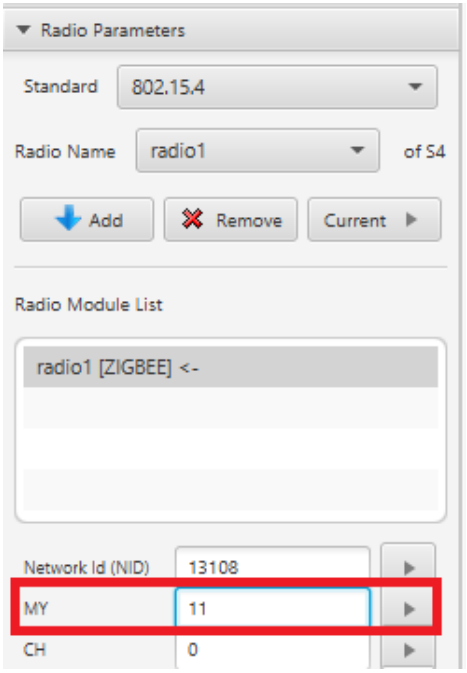


Fig. 6.5.21. Visualisation of the procedure for setting the MY parameter

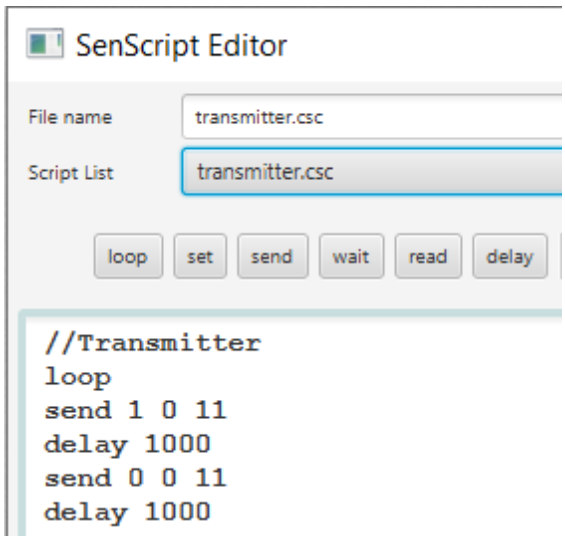


Fig. 6.5.22. An example of a script for an information message transmitter, taking into account the settings of the MY parameter

10. Analyse the results obtained for the correctness of the created models and algorithms.

Recommended contents of the report for the practical work:

1. Title page.

2. Aim and objectives of the practical work.

3. The main results obtained during the creation and testing of scenarios for group exchange of information messages in wireless sensor networks, according to the relevant research points (text description and graphical visualisation).

4. General conclusions on the practical work, containing the results of critical analysis at the qualitative and quantitative levels.

Review questions:

1. What are the main functional elements utilised when creating and testing wireless IoT networks?

2. What characteristic functional features of the information message transmitter during group data exchange were studied?

3. Explain the meaning of the term ‘group exchange of information messages’.

4. What types of functional scenarios of wireless IoT networks were studied in this case study?

5. What basic programming functions were utilised when creating and testing the scripts? Explain their syntax and basic purpose.

6. What is the purpose of the MY identifier when configuring the radio parameters of the network? What general functions of an information and communication network does it affect?

7. How does the syntax of the *send* function change when selectively sending information messages to individual network nodes compared to the basic syntax of this function?

6.5.6 Practical Work No. 6. Development and Testing of Models and Algorithms for the Functioning of Digital Sensors as Part of IoT Networks

The aim is to reinforce theoretical knowledge and develop practical skills in the development, testing and analysis of models and algorithms for digital sensors as functional components of IoT networks of various architectures using computer modelling approaches in the CupCarbon software.

Tools and software: personal computer, CupCarbon, MS Office.

The task to be performed is to develop and investigate operational models and algorithms of digital sensors as functional components of IoT networks of various architectures in the CupCarbon software.

The procedure of the work:

1. Create a new project in the CupCarbon software: Project – New project.

2. Set the map parameters – Notebook: Map – Notebook.

3. Add two new elements of the wireless sensor node to the project map: Add – Add sensor node.

4. Place the wireless sensor nodes in such a way that their coverage areas overlap (form a network), as shown in Fig. 6.5.23. In such a network, the first sensor node (S1) performs the functions of detecting the presence of an object in its coverage area and transmitting the corresponding digital signal and the second (S2) is the receiver of the information messages.

5. Create a route for the mobile object using markers, which must necessarily pass through the coverage area of the sensor node (S1), as shown in Fig. 6.5.23. To do this, you need to place two edge markers (Add Marker) on the map – select them – and press the ‘u’ key several times, which will add intermediate markers. After that, the resulting route (the formed set of markers) must be saved, as shown in Fig. 6.5.24.

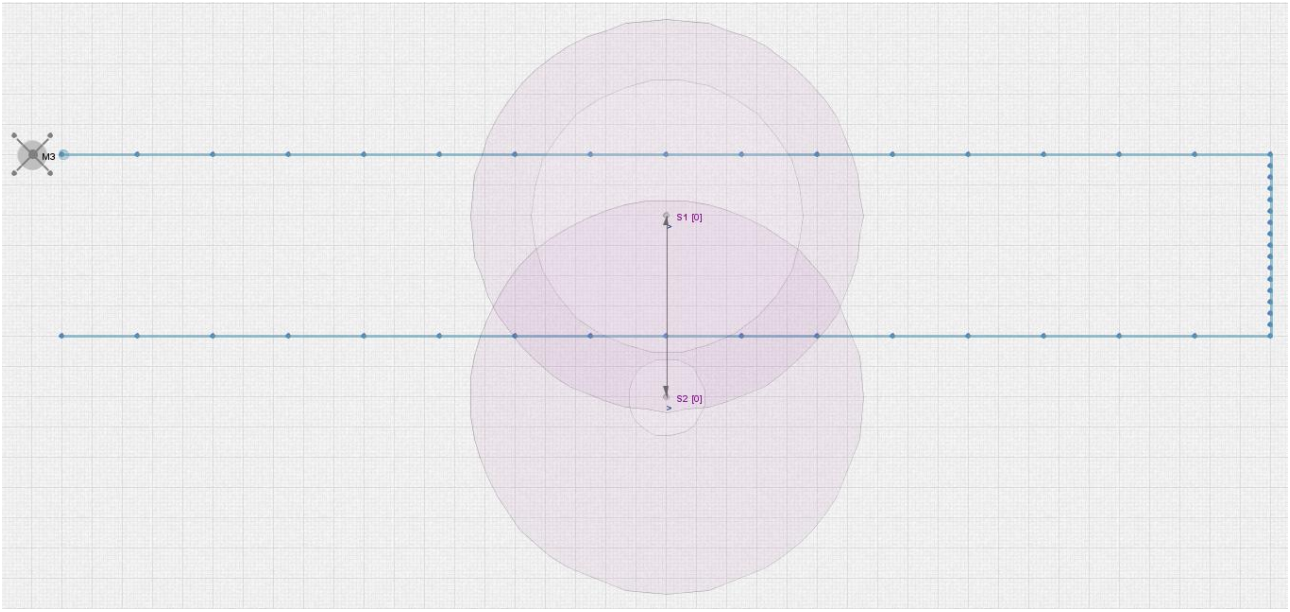


Fig. 6.5.23. An example of the studied model of reading and transmitting a digital signal

6. Add a mobile object to the model (Add Mobile) and synchronise it with the created route, as shown in Fig. 6.5.25. After the mobile object is synchronised with the route, its internal part becomes marked in yellow.

7. Check the parameters of all settings of model objects before testing. A typical view of the model is shown in Fig. 6.5.26.

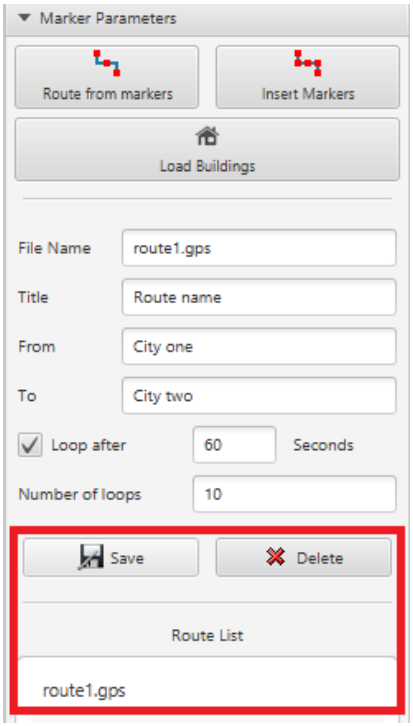


Fig. 6.5.24. The procedure for saving the created route

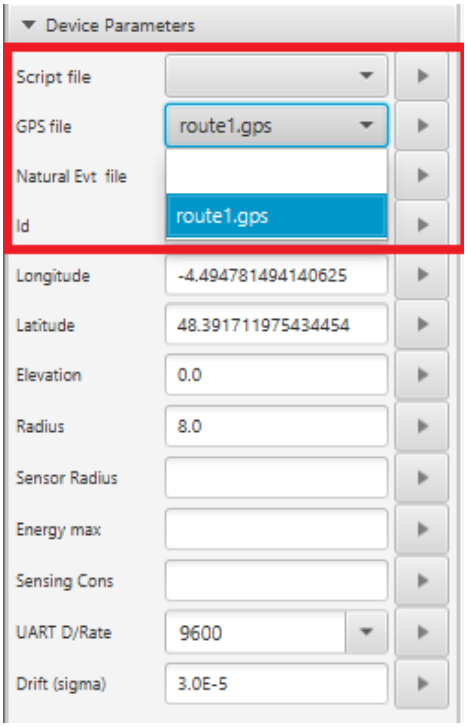


Fig. 6.5.25. Route and mobile object synchronisation procedure

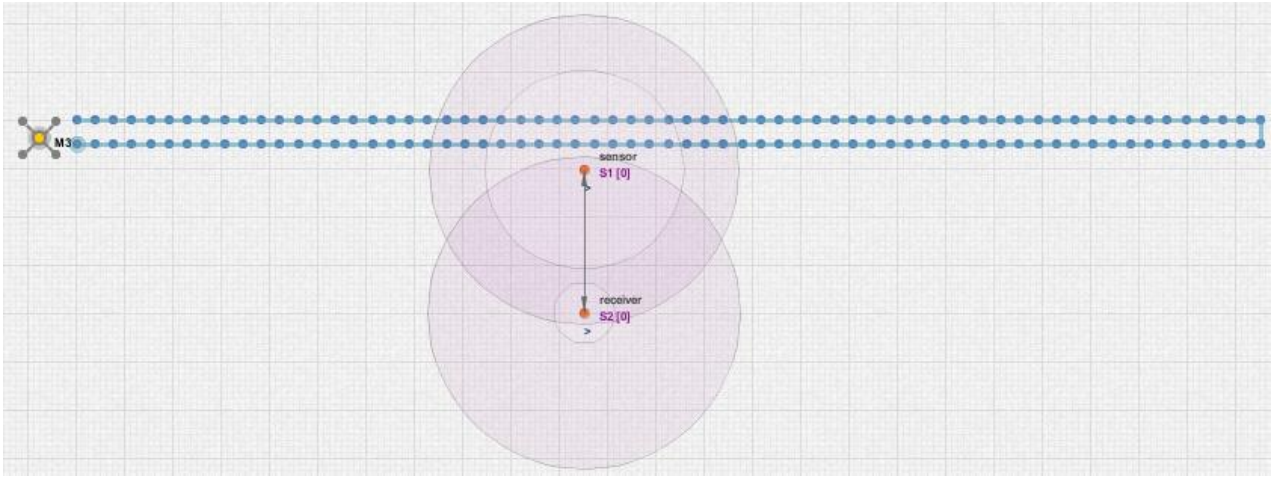


Fig. 6.5.26. Typical view of the studied model

8. Write and add to the first wireless sensor node (S1) a script for reading the signal and transmitting it to node S2, as shown in Fig. 6.5.27.

9. Write and add to the second wireless node (S2) a receiver script that implements the reception of a digital signal, its display and marking (if 1 is received) / unmarking (if 0 is received), as shown in Fig. 6.5.28.

Fig. 6.5.27. An example script for the information signal detector

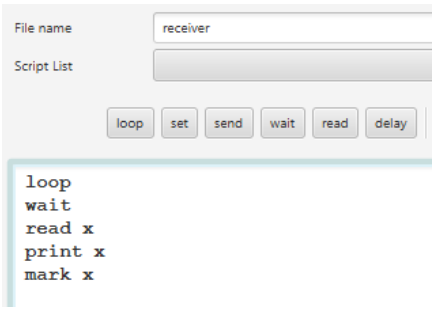


Fig. 6.5.28. An example script for the information signal receiver

10. Test the created model. Before starting the simulation process, the *Mobility / Events* function must be activated in the Simulation Parameters tab. Observe the results in the console window (using the *cprint* function) and draw appropriate conclusions about the correctness of the synthesised model.

11. Modify the created model by adding components of information message routing. The number of routers in the network is assumed to be as follows: three – if the last number of the student card is 2, 6, 9; four – 0, 3, 7; five – 1, 4, 5, 8.

12. Modify the basic version of the model by increasing the number of signal receivers. The number of receivers in the network is assumed to be as follows: four – if the last number of the student card is 1, 4, 5, 8; five – if the last digit is 0, 2, 7; six – if the last digit is 3, 6, 9. Implement the following signal transmission algorithms:

– simultaneously send the digital sensor detection result to all network receivers;

– set the address parameter MY=12 for any three IoT network receivers and send the digital sensor detection signal only to the receivers that have the address parameter MY=12.

13. Analyse the results obtained for the correctness of the created models and algorithms.

Recommended contents of the report for the practical work:

1. Title page.

2. Aim and objectives of the practical work.

3. The main results obtained during the creation and testing of models and algorithms for the operation of digital sensors as functional components of IoT

networks of different architectures, according to the relevant research points (text description and graphical visualisation).

4. General conclusions on the practical work, containing the results of critical analysis at the qualitative and quantitative levels.

Review questions:

1. What are the main components used in the WSN design in this case study? What are their main parameters and characteristics?

2. What are the characteristic functional features of the sensor node during the detection and sending of information signals (data) were investigated?

3. What functional scenarios of wireless IoT networks were investigated in this case study?

4. Explain the meaning of the terms ‘digital signal’ and ‘digital sensor’.

5. What basic programming functions were utilised to create and test the scripts? Explain their syntax.

6. How is a route created for a mobile network node?

7. How is the synchronisation of the route and the mobile node of the IoT network implemented in the CupCarbon software?

6.5.7 Practical Work No. 7. Development and Testing of Analogue Sensor Functioning Algorithms in the Development of IoT Networks

The aim is to reinforce theoretical knowledge and develop practical skills in the development, testing and analysis of operational algorithms for analogue sensors as functional components in the design of IoT systems and networks of various architectures, using computer modelling approaches in the CupCarbon software.

Tools and software: personal computer, CupCarbon, MS Office.

The task to be performed is to develop and investigate algorithms for analogue sensors as functional components of IoT systems and networks of various architectures in the CupCarbon software.

The procedure of the work:

1. Create a new project in the CupCarbon software: Project – New project.

2. Set the map parameters – Notebook: Map – Notebook.

3. Add two new elements of the wireless sensor node to the project map: Add – Add sensor node.

4. Place the wireless sensor nodes in such a way that their coverage areas overlap (form a network), as shown in Fig. 6.5.29. In such a network, the first sensor node (S1) performs the functions of detecting the presence of an object in its coverage area and transmitting the corresponding analogue signal and the second (S2) is the receiver of the information messages.

5. Add an object to the model that simulates the operation of the analogue value generator (Add Gas), as shown in Fig. 6.5.29.

6. For the Gas (Analog event) element to work correctly, it must be synchronised with the Natural Event file (see Fig. 6.5.30), which can be generated automatically, as shown in Fig. 6.5.31. The following parameters should be set when generating this file: the mean value of the random distribution (mean) (calculated using the formula 20+*a*, where *a* is the penultimate number of the student ID card); standard deviation (std) (calculated using the formula *b*/2 with rounding to the nearest whole if *b*=3...9 or *b*+2 if *b*=0...2, where *b* is the last number of the student ID card); period (period) is taken as 1 s; size (size) – 500.

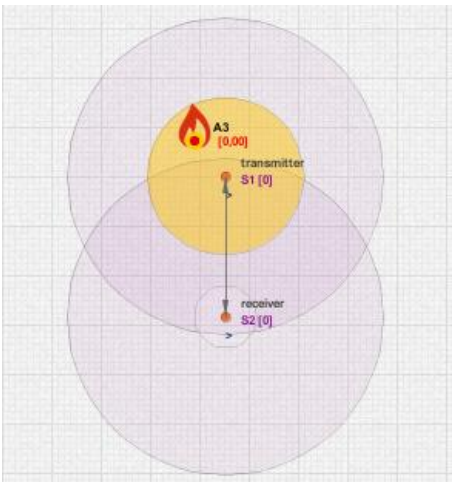


Fig. 6.5.29. An example of the model under study

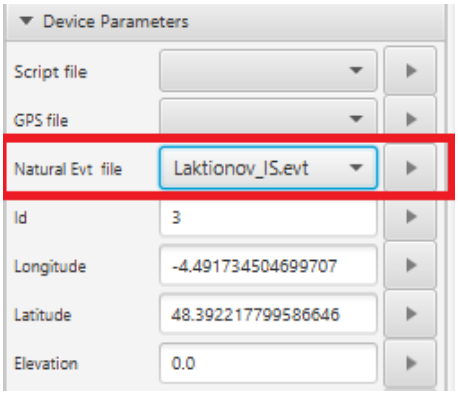


Fig. 6.5.30. The procedure for synchronising the Gas element and the Natural event file

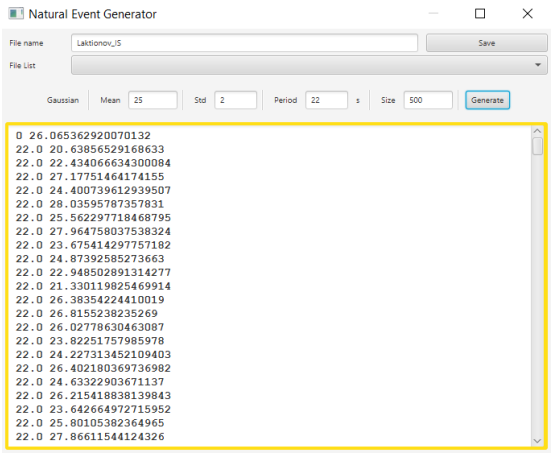


Fig. 6.5.31. The procedure for generating a Natural event file

7. Write and add to the first wireless sensor node (S1) a script for reading the analogue signal and its transmitters to node S2, as shown in Fig. 6.5.32.

8. Write and add to the second wireless sensor node (S2) a receiver script that implements the reception of an analogue signal and its display, as shown in Fig. 6.5.33.

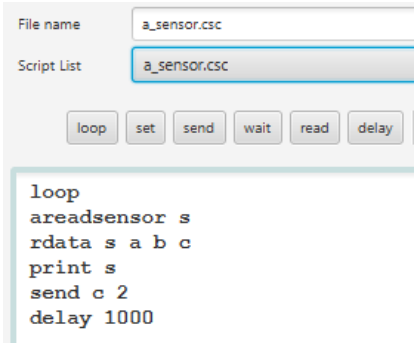


Fig. 6.5.32. An example script for the information signal detector

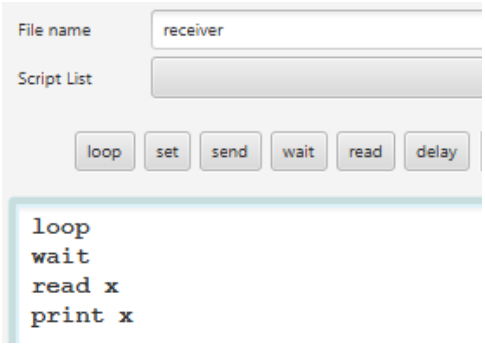


Fig. 6.5.33. An example script for the information signal receiver

9. Test the created model. Before starting the simulation process, the *Mobility / Events* function must be activated in the Simulation Parameters tab. Observe the results and draw appropriate conclusions about the correctness of the synthesised model.

10. Modify the functionality of the signal receiver as follows: add the ability to save the received data and the current simulation time to a file; add the function of marking the receiver node if the received value is greater than the average (the *mean*

parameter in the generated Natural event file) and, accordingly, unmarking if the received value is less than the average.

11. Modify the model obtained in step 10 by adding information message routing components to it. The number of routers in the network is assumed to be as follows: three – if the last number of the student ID card is 1, 5, 8; four – 2, 4, 9; five – 0, 3, 6, 7.

12. Modify the model obtained in step 10 by increasing the number of signal receivers. The number of receivers in the network is assumed to be as follows: four – if the last number of the student card is 0, 2, 7; five – 1, 4, 6; six – 3, 5, 8, 9. Implement the following signal transmission algorithms:

– send the detection result to all network receivers simultaneously;

– set the address parameter MY=11 for any three IoT network receivers and send the analogue sensor detection signal only to the receivers that have the address parameter MY=11.

13. Analyse the results obtained for the correctness of the created models and algorithms.

Recommended contents of the report for the practical work:

1. Title page.

2. Aim and objectives of the practical work.

3. The main results obtained during the creation and testing of models and algorithms for analogue sensors as functional components of IoT networks of different architectures, according to the relevant research points (text description and graphical visualisation).

4. General conclusions on the practical work, containing the results of critical analysis at the qualitative and quantitative levels.

Review questions:

1. What basic components were utilised when creating and testing wireless sensor IoT networks in this case study? What are their main parameters and characteristics?

2. How is the sequence of setting up the correct operation of the Gas (Analog

Event) element implemented?

3. What characteristic features of the Gas (Analog Event) element were investigated?

4. What algorithms of network data exchange in IoT systems were investigated in this case study?

5. Explain the meaning of the terms ‘analogue signal’ and ‘analogue sensor’.

6. What basic programming functions were utilised to create and test scripts? Explain their syntax.

6.5.8 Practical Work No. 8. Research of Typical Models of Radio Modules and Wireless Data Exchange Technologies in the Development of IoT Networks

The aim is to reinforce theoretical knowledge and develop practical skills in testing and analysing basic models of radio modules and wireless technologies, as well as data exchange standards when developing IoT systems and networks of various architectures using computer modelling approaches in the CupCarbon software.

Tools and software: personal computer, CupCarbon, MS Office.

The task to be performed is to study radio module models and wireless technologies, as well as data exchange standards as structural and algorithmic components of IoT systems and networks of various architectures in the CupCarbon software.

The procedure of the work:

1. Create a new project in the CupCarbon software: Project – New project.

2. Set the map parameters – Notebook: Map – Notebook.

3. Add a new element of the wireless sensor node to the project map: Add – Add sensor node.

4. Open the Radio Parameters tab and select the 802.15.4 (Zigbee) standard.

5. Add the second radio communication technology to this sensor node – the LoRa standard and set the name radio2, as shown in Fig. 6.5.34.

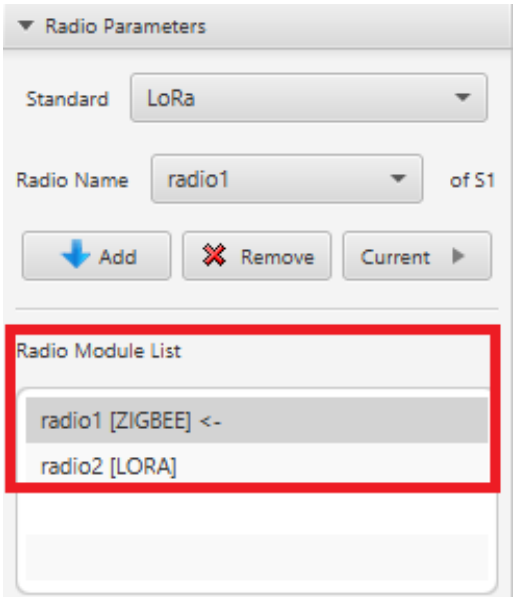


Fig. 6.5.34. Procedure for adding radio technology for a sensor node

6. Add a new element of a wireless sensor node type to the project map and leave its radio parameters unchanged, as shown in Fig. 6.5.35.

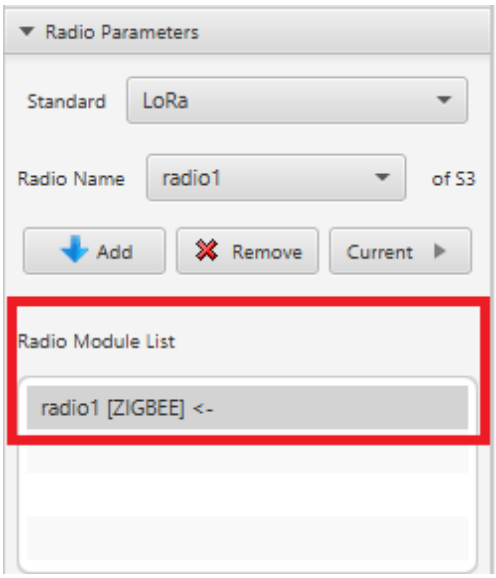


Fig. 6.5.35. Configuring the second sensor node

7. Add a new element of the wireless sensor node type to the project map and make it the main type of radio communication standard LoRa (add the Lora protocol as described in step 5 – click ‘Current’ on LoRa technology – remove Zigbee technology from the list), as shown in Fig. 6.5.36.

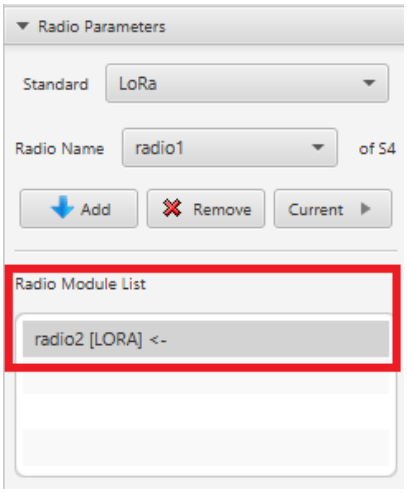


Fig. 6.5.36. Configuring the third sensor node

8. Place the sensor nodes in the structure shown in Fig. 6.5.37.

Fig. 6.5.37. Block diagram of the studied wireless sensor network

9. Add to the sensor node of the transmitter (S1 with LoRa and Zigbee technologies) a script (see Fig. 6.5.38) that implements the alternate transmission of information messages using the LoRa and Zigbee standards.

10. Add a script to the sensor nodes of the receivers (S2 and S3) (see Fig. 6.5.39) that implements the reception of information messages using the Zigbee and LoRa standards, respectively.

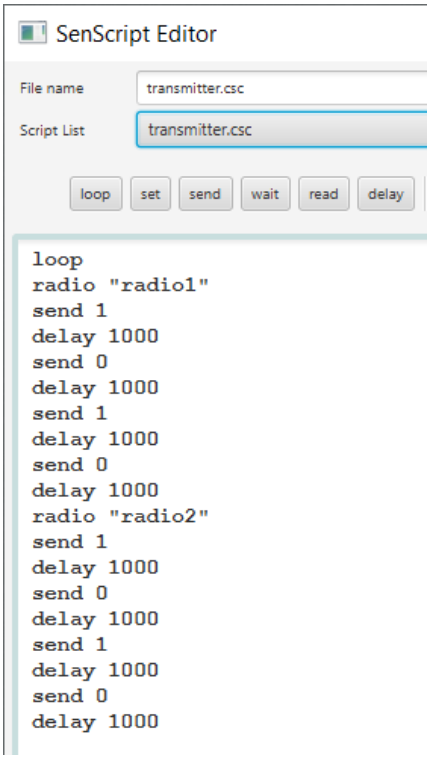


Fig. 6.5.38. An example of a script for the information message transmitter

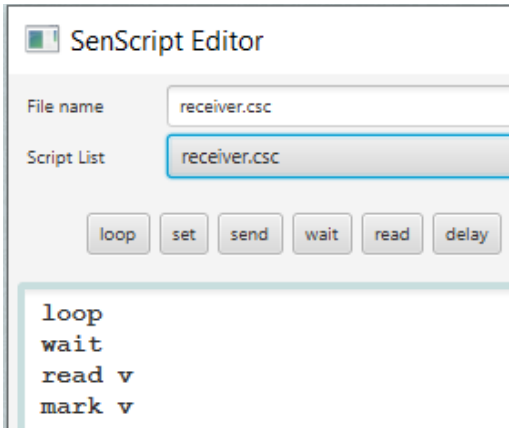


Fig. 6.5.39. An example of a script for the information message receivers

11. Test the created IoT network model. Track the results and draw appropriate conclusions about the correctness of the synthesised model.

12. Modify the functionality of the signal receivers as follows: add the ability to save the received data and the current simulation time to a file. It is recommended to use the functions: *time* and *printfile*.

13. Modify the IoT model of the wireless sensor network obtained in steps 8 – 11, by adding routing components that operate on the same standard as the

corresponding information signal receiver. The number of routers in the network is assumed to be as follows: three – if the last number of the student card is 0, 3, 7; four – 1, 5, 9; five – 2, 4, 6, 8.

14. Modify the IoT network model obtained in steps 8 – 11 by sending an information message with the current coordinates by the transmitter. It is recommended to use the function: *getpos* or *getpos2*.

15. Analyse the results obtained for the correctness of the created models and algorithms.

Recommended contents of the report for the practical work:

1. Title page.

2. Aim and objectives of the practical work.

3. The main results obtained during the testing and analysis of the main models of radio modules and wireless technologies, as well as data exchange standards in the design of IoT systems and networks, according to the relevant research points (text description and graphical visualisation).

4. General conclusions on the practical work, containing the results of critical analysis at the qualitative and quantitative levels.

Review questions:

1. Provide a description of the main parameters and characteristics of the CupCarbon components that were investigated and tested in this case study.

2. Explain the principle of operation of Zigbee technology.

3. Explain the principle of operation of LoRa technology.

4. What algorithms and scenarios of network data exchange in IoT systems were studied in this case study?

5. Explain the functional purpose of wireless data transmission technologies in the development of IoT networks.

6. What basic programming functions were utilised when creating and testing scripts? Explain their syntax.

7. Give examples of industrial applications of Zigbee and LoRa technologies.

8. What are the advantages of Zigbee and LoRa technologies when implemented

in industrial automation compared to classical wired communication technologies?

9. What wireless technologies for data exchange in IoT networks are known besides Zigbee and LoRa? Explain their main functional purpose and basic principle of operation.

List of Recommended Literature for the Sixth Chapter

1. CupCarbon IoT. Available at: https://cupcarbon.com/ (accessed on March 07, 2024).

2. Laktionov I.S. Information and measuring equipment and hardware and software for building computerised systems for monitoring the state of microclimate in greenhouses: Thesis of Doctor of Engineering Sciences: 05.13.05 – computer systems and components / Donetsk National Technical University: D 11.052.03. Pokrovsk, 2021. 518 p. (in Ukrainian).

3. Kharchenko V.S. (ed.) Internet of Things for Industry and Human Application. In Volumes 1–3. Vol. 1. Fundamentals and Technologies. Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. 605 p.

4. Kharchenko V.S. (ed.). Internet of Things for Industry and Human Application. In Volumes 1–3. Vol. 2. Modelling and Development. Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. 547 p.

5. Kharchenko V.S. Internet of Things for Industry and Human Application. In Volumes 1–3. Vol. 3. Assessment and Implementation. Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. 918 p.

6. Laktionov I., Diachenko G., Koval V., Yevstratiev M. Computer-Oriented Model for Network Aggregation of Measurement Data in IoT Monitoring of Soil and Climatic Parameters of Agricultural Crop Production Enterprises. Baltic J. Modern Computing, 2023, Vol. 11 (3). P. 500–522.

REFERENCES

1. Arduino. Available at: https://www.arduino.cc/ (accessed on December 26, 2023).

2. Bravos G., Cabrera A.J., Correa C., Danilovic D., Evangeliou N., et al. Cybersecurity for Industrial Internet of Things: Architecture, Models and Lessons Learned. *IEEE Access*. 2022. Vol. 10. P. 124747–124765.

3. Business workshop: Internet of Things: network architecture and security architecture. Available at: https://www.bizmaster.xyz/2020/12/internet-rechei-merezheva-arkhitektura-ta-arkhitektura-bezpeky.html (accessed on November 16, 2023) (in Ukrainian).

4. Cross-platform programming and cloud services. Available at: https://victana.lviv.ua/knyhy/konspekty-lektsii/133-kros-platformenne-prohramuvannia-ta-khmarni-servisy?start=10 (accessed on January 16, 2024).

5. CupCarbon IoT. Available at: https://cupcarbon.com/ (accessed on March 07, 2024).

6. Dnipro University of Technology: What is a microprocessor, microcontroller and programmable logic controller. Available at: https://elprivod.nmu.org.ua/ua/interesting/what_is_mp_mc_plc.php (accessed on January 04, 2024).

7. Electrical engineering online: Sensors of automation systems. Available at: http://electro-tex.ho.ua/le/m5/5-2 (accessed on December 27, 2023).

8. Embedded Computing Design: Introduction to Web of Things (WoT). Available at: https://embeddedcomputing.com/technology/iot/introduction-to-web-of-things-wot-heres-everything-you-need-to-know (accessed on December 03, 2023).

9. Fortinet: IoT Vulnerabilities: An Overview. Available at: https://www.fortinet.com/resources/cyberglossary/iot-best-practices (accessed on October 25, 2023).

10. Google Cloud. Available at: https://cloud.google.com/ (accessed on January 20, 2024).

11. How to connect your device to IBM Watson Cloud Platform. Available at:

https://docs.iotize.com/Technologies/IBMWatson/ (accessed on January 20, 2024).

12. Huawei: The history of the emergence of the Internet of Things and prospects for development. Available at: https://e.huawei.com/kz/ict-insights (accessed on December 21, 2023).

13. Instructor Textbook «Designing & Deploying Cloud Solutions for Small and Medium Business»: Rev. 1.0. Palo Alto: Hewlett-Packard Comp., 2013. 893 p.

14. IT Enterprise: Internet of Things. Available at: it.ua/knowledge-base/technology-innovation/internet-veschej-internet-of-things-iot (accessed on November 27, 2023).

15. Java Point: IoT Tutorial. Available at: https://www.javatpoint.com/iot-internet-of-things (accessed on December 26, 2023).

16. Kharchenko V.S. (ed.) Internet of Things for Industry and Human Application. In Volumes 1–3. Vol. 1. Fundamentals and Technologies. Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. 605 p.

17. Kharchenko V.S. (ed.). Internet of Things for Industry and Human Application. In Volumes 1–3. Vol. 2. Modelling and Development. Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. 547 p.

18. Kharchenko V.S. Internet of Things for Industry and Human Application. In Volumes 1–3. Vol. 3. Assessment and Implementation. Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. 918 p.

19. Krishnasamya E., Varrettea S., Mucciardi M. Edge Computing: An Overview of Framework and Applications. Brussels: PRACE, 2020. 20 p.

20. Laktionov I.S., Vovna O.V. Lecture notes for the disciplines: ‘Measurement of Electrical Values and Metrological Support of Electronic Systems’, ‘Fundamentals of Metrology and Electronic Measurements’, ‘Measurement of Electrical Values and Metrological Support of Automatic Systems’, ‘Fundamentals of Metrology’. Pokrovsk: SHEI ‘DonNTU’, 2017. 80 p. (in Ukrainian).

21. Laktionov I., Diachenko G., Koval V., Yevstratiev M. Computer-Oriented Model for Network Aggregation of Measurement Data in IoT Monitoring of Soil and

Climatic Parameters of Agricultural Crop Production Enterprises. Baltic Journal of Modern Computing, 2023, Vol. 11 (3). P. 500–522.

22. Laktionov I., Diachenko G., Rutkowska D., Kisiel-Dorohinicki M. An Explainable AI Approach to Agrotechnical Monitoring and Crop Diseases Prediction in Dnipro Region of Ukraine. Journal of Artificial Intelligence and Soft Computing Research, 2023, Vol. 13 (4). P. 247–272.

23. Laktionov I., Rutkowski L., Vovna O., Byrski A., Kabanets M. A novel approach to intelligent monitoring of gas composition and light mode of greenhouse crop growing zone on the basis of fuzzy modelling and human-in-the-loop techniques. Engineering Applications of Artificial Intelligence, 2023, Vol. 126 (Part B), P. 1–21.

24. Laktionov I., Vovna O., Kabanets M. Computer-Oriented Method of Adaptive Monitoring and Control of Temperature and Humidity Mode of Greenhouse Production. Baltic J. Modern Computing, 2023, Vol. 11 (1). P. 202–225.

25. Laktionov I.S. et al. Improved Computer-Oriented Method for Processing of Measurement Information on Greenhouse Microclimate. Int. J. Bioautomation, 2019, Vol. 23 (1). P. 71–86.

26. Laktionov I.S. Information and measuring equipment and hardware and software for building computerised systems for monitoring the state of microclimate in greenhouses: Thesis of Doctor of Engineering Sciences: 05.13.05 – computer systems and components / Donetsk National Technical University: D 11.052.03. Pokrovsk, 2021. 518 p. (in Ukrainian).

27. Laktionov I.S., Lebediev V.A., Laktionova H.A. Methodical instructions for practical work for the disciplines: ‘Fundamentals of monitoring and information processing in technological production’, ‘Microprocessor control and information processing devices in power systems’, ‘Microprocessor technology’ (for full-time and part-time students of all specialities). Pokrovsk: SHEI ‘DonNTU’, 2021. 48 p. (in Ukrainian).

28. Laktionov I.S., Vovna O.V., Kabanets M.M., Getman I.A., Zolotarova O.V. Computer-Integrated Device for Acidity Measurement Monitoring in Greenhouse Conditions with Compensation of Destabilizing Factors. Instrumentation Mesure

Metrologie, 2020, Vol. 19 (4). P. 243 – 253.

29. Laktionov I.S., Vovna O.V., Kabanets M.M., Sheina H.O., Getman I.A. Information model of the computer-integrated technology for wireless monitoring of the state of microclimate of industrial agricultural greenhouses. Instrumentation Mesure Metrologie, 2021, Vol. 20 (6). P. 289 – 300.

30. Lebediev V.A., Laktionov I.S., Vovna O.V., Kabanets M.M., Sahaida P.I., Dobrovolska L.O. Methods of improving technical and functional characteristics of serial budget microprocessor platforms. Journ. Europeen des Systemes Automatises, 2022, Vol. 55 (1). P. 81–88.

31. Links C., Tesla T., Anderton J., Van Hoogstraeten W., Schnaufer D., Warschauer C. Internet of Things: 2nd Special Edition. Hoboken: John Wiley & Sons Inc., 2021. 44 p.

32. Metrology. Terms and definitions: DSTU 2681-94. Effective from 1994-07-26. Kyiv: State Standard of Ukraine, 1994. 68 p. (in Ukrainian).

33. Microsoft: Glossary of IoT terms. Available at: https://learn.microsoft.com/uk-ua/azure/iot/iot-glossary (accessed on November 30, 2023).

34. Moco Smart: Difference between IIoT and IoT. Available at: https://www.mokosmart.com/uk/iiot-vs-iot-technologies/ (accessed on December 15, 2023).

35. MQTT: The Standard for IoT Messaging. Available at: https://mqtt.org/ (accessed on January 17, 2024).

36. Onykiienko Y.O., Ryzhova A.R. Lecture notes ‘Fundamentals of designing Internet of Things systems. The periphery of STM32 microcontrollers: a textbook: electronic network educational edition’. Kyiv: Igor Sikorsky Kyiv Polytechnic Institute, 2022. 127 p. (in Ukrainian).

37. Patel K.K., Patel S.M. Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges. Int. J. of Engineering Science and Computing, 2016, Vol. 6, No. 5. P. 6122 – 6131.

38. Perenio: History of the Internet of Things. Available at: https://perenio.com/blog/the-history-of-the-internet-of-things (accessed on December 23, 2023).

39. Podzharenko V.O., Vasilevsky O.M., Kucheruk V.Y. Processing of measurement results based on the concept of uncertainty: a textbook. Vinnytsia: VNTU, 2008. 128 p. (in Ukrainian).

40. QA Test Lab: Client-server architecture. Available at: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (accessed on January 14, 2024).

41. Rackley S. Wireless Networking Technology: From Principles to Successful Implementation. Oxford: Elsevier, 2007. 425 p.

42. Raspberry Pi 4. Available at: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ (accessed on January 09, 2024).

43. Telesphere: What you need to know about the Internet of Things. Available at: https://www.telesphera.net/blog/iot-likbez.html (accessed on December 26, 2023) (in Ukrainian).

44. ThingsBoard Open-source IoT Platform. Available at: https://thingsboard.io/ (accessed on January 18, 2024).

45. ThingSpeak™. Available at: https://thingspeak.com/ (accessed on January 18, 2024).

46. ThingWorx IIoT Solutions Platform. Available at: https://www.ptc.com/en/products/thingworx (accessed on January 19, 2024).

47. TI40: Industry 4.0 technologies. Lectures (by Oleksandr Pupena). Available at: https://pupenasan.github.io/TI40/%D0%9B%D0%B5%D0%BA%D1%86/intro.html (accessed on November 25, 2023).

48. Tibco: What is Industrial Internet of Things (IIoT). URL: https://www.tibco.com/glossary/what-is-the-internet-of-things-iot (accessed on October 23, 2023).

49. Vovna O., Laktionov I., Andrieieva A., Petelin E., Shtepa O., Laktionova H. Optimised Calibration Method for Analog Parametric Temperature Sensors. Instrumentation Mesure Metrologie, 2019, Vol. 18 (6). P. 517 – 526.

50. Vovna O.V., Laktionov I.S., Koyfman O.O., Stashkevych I.I, Lebediev V.A. Study of Metrological Characteristics of Low-Cost Digital Temperature

Sensors for Greenhouse Conditions. Serbian Journal of Electrical Engineering, 2020, Vol. 17 (1). P.1 – 20.

51. Vovna O.V., Laktionov I.S., Lebediev V.A. Computer-integrated monitoring and control in industrial greenhouses: current results and research prospects: monograph. Pokrovsk: SHEI 'DonNTU', 2020. 255 p. (in Ukrainian).

52. W3: Web Thing Model. Available at: https://www.w3.org/Submission/2015/SUBM-wot-model-20150824/ (accessed on November 12, 2023).

53. Web Thing Model: Specifications and Standards. Available at: https://webofthings.org/standards/ (accessed on December 22, 2023).

54. Y.2060. Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks. Next Generation Networks – Frameworks and Functional Architecture Models / ITU-T. Geneva: International Telecommunication Union, 2013. 22 p.

55. Zhurakovsky B.Yu., Zeniv I.O. Internet of things technologies. Study guide [Electronic resource]: education manual for students speciality 126 'Information systems and technologies', specialisation 'Information support of robotic systems'. Kyiv: KPI named after Igor Sikorskyi, 2021. 271 p. (in Ukrainian).

56. Zinchenko O.V. et al. Cloud technologies: a textbook. Kyiv: FOP Guliaeva V.M., 2020. 74 p. (in Ukrainian).

Educational edition

Laktionov Ivan

Vovna Oleksandr

Diachenko Grygorii

DIGITALIZATION AND INTELLECTUALIZATION OF INDUSTRIAL ECOSYSTEMS. IN 2 PARTS. PART 1: INTERNET OF THINGS

Textbook

Published in the author's edition.

Electronic resource.

Signed for publication 10.09.2024. The author's pages are 18.2.

Prepared for publication at Dnipro University of Technology

Certificate of registration in the State Register DK № 1842 dated 11.06.2004.

49005, Dnipro, av. Dmytra Yavornytskoho, 19