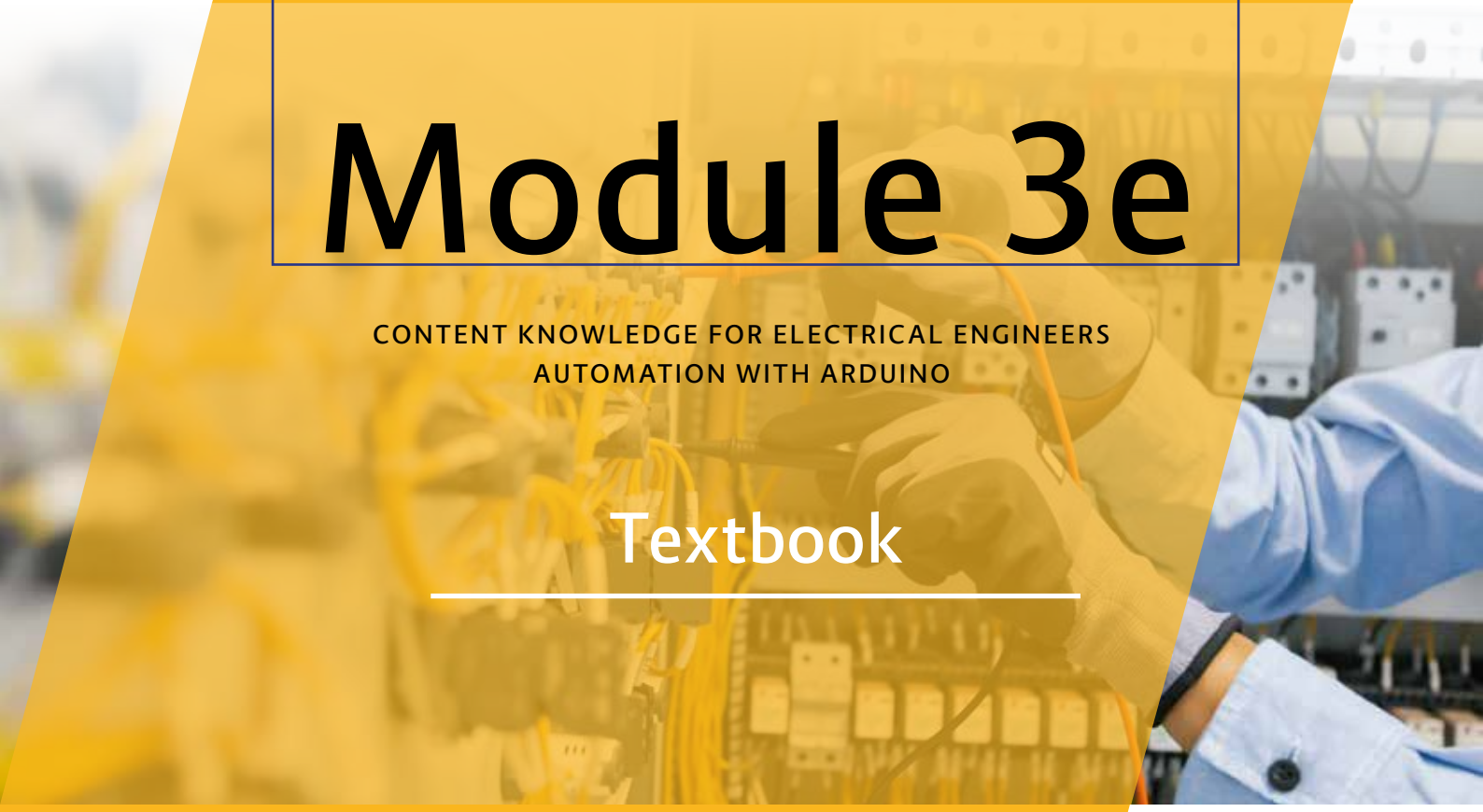
ÜBZO - INTER-COMPANY TRAINING CENTER IN EASTERN BAVARIA 2021

Module 3e

CONTENT KNOWLEDGE FOR ELECTRICAL ENGINEERS
AUTOMATION WITH ARDUINO

Textbook



Education is future

MODULAR
TRAINING
& EDUCATION

IN MECHANICAL
& ELECTRICAL ENGINEERING

TRAINME

VERSION 01

Editing

Author:	Dipl.-Ing. (Univ.) Helmut Windschiegl
Illustrations, Layout:	Dipl.-Ing. (Univ.) Helmut Windschiegl
Cover Photos:	Carina Adam, Helmut Windschiegl
Layout Assistance:	Sebastian Gmeiner
Editor:	Carina Adam
Version:	July 2021

This study book has been developed within the bilateral programme TRAINME (Modular Training and Further Education of South African TVET Lecturers in Mechanical and Electrical Engineering). The programme has been designed by the Inter-Company Training Center in Eastern Bavaria (ÜBZO) and the University of Stuttgart (Dept. BPT) on behalf of the German Federal Ministry of Education and Research in cooperation with the South African Department of Higher Education and Training (DHET).

Powered by: **Supported by:**

Preamble

The field of industrial automation technology is currently developing and changing at a breathtaking pace. It is becoming increasingly important for users and developers to understand automated systems as a whole and to be able to analyse them on the basis of fundamental knowledge.

Take your students into the new world of automation, teaching technologies of the present and the future, building on a solid understanding of basic enabling technologies of sensors, actuators and networking.

This studybook is intended to help bridge the gap between fundamental technical contexts to Industry 4.0 systems. Internet of Things, edge computing, cloud-based and networked systems, and the use of artificial intelligence are just a few buzzwords that will take us directly into the future in this context.

The TRAINME project deliberately focuses on content and technologies that are easily accessible and can be implemented in schools without major financial outlay. For both teachers and students, the introduction to the Arduino development environment opens up a global maker scene with no shortage of projects and project ideas to replicate and improve.

This very versatile microcontroller board is available in many versions and has now also found its way into the world of small industrial controllers with the development and marketing of the Controllino. In addition to the fact that the boards, the sensors, the actuators and all other materials are relatively inexpensive to purchase, it is particularly positive that the development environment (IDE) and numerous software libraries, programming examples and tutorials are freely available on the Internet and can be downloaded free of charge.

Another advantage is that there is a myriad of Internet sites where other users offer advice and valuable tips on how to implement complex projects of their own. It is therefore relatively easy to gain in-depth knowledge and skills in the use of different sensors and the control of actuators based on them. In independent small projects it is possible to understand and reproduce more and more complex applications of control and regulation systems. From a sufficient depth of knowledge, you will even be able to use wireless connection technologies such as Wifi and Bluetooth for component communication.

The fact that a simplified variant of the C++ programming language is used for programming can be seen as a particularly valuable side effect. This generally applicable and easy-to-learn programming environment forms a solid basis for other programming languages such as Python, which is also very widely used for programming complex data structures and in the use of artificial intelligence.

Get involved in a particularly practical and motivating way of teaching important and future-oriented learning content.

The author

Content

Unit 1

Unit 2

Chapter 1

From automation to Industry 4.0

1.1 INTRODUCTION: INDUSTRY 4.0

1.1.1 Changes in production

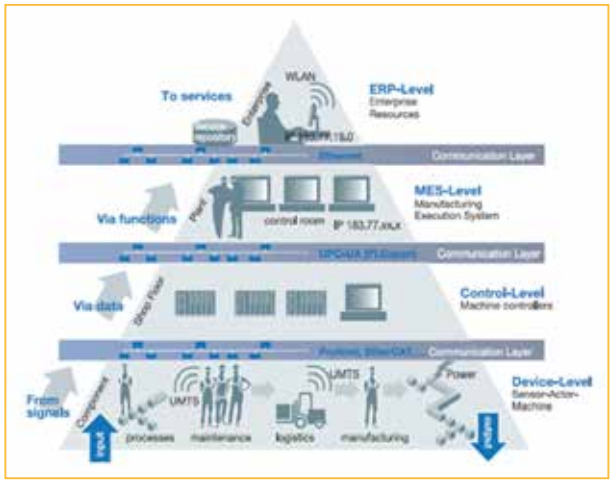
Up to now, the aim in most areas of production was the higher and higher level of automation. More and more complicated systems and machines are trying to substitute human workers in many areas, or at least trying to complement them.

In addition, most known systems of automation are based on classic control technologies. Besides that, the automation pyramid has been the basic structure of automation technology for decades.

Targets in the field of production:

- rising level of automation
- more complex systems and machines
- substitute or complement human workers

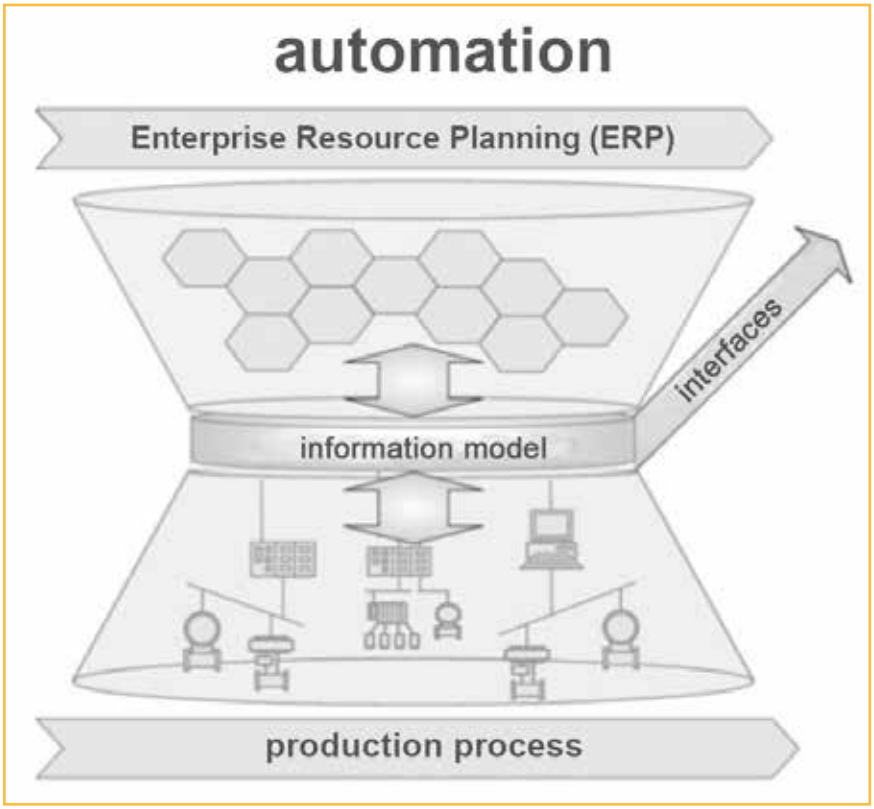
The basis of the new structure is the classic control technology. Up to now, the automation pyramid has served as basic structure of automation technology.



Pyramid of automation

This level-oriented automation pyramid is increasingly being replaced by an automation diabolo, in which the automation levels are dissolved and connected via an information model with the MES (Manufacturing Execution System) –Level.

A manufacturing execution system operates close to the process and acts as the operational layer of a multi-layered manufacturing management system.

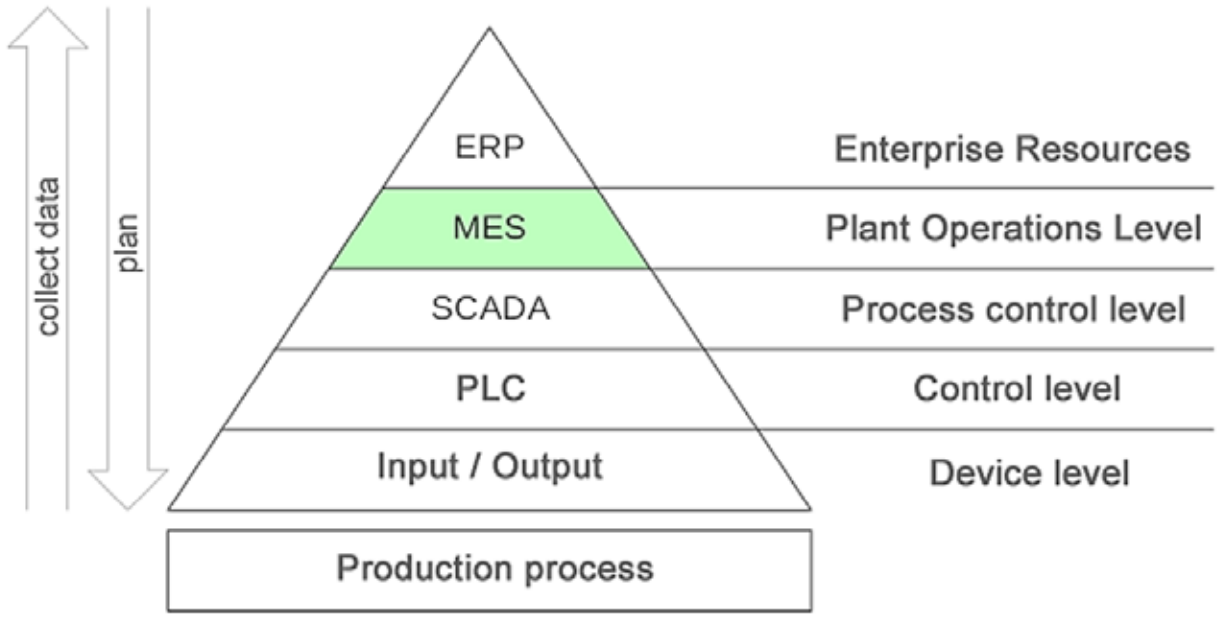


1.1.2 What is an MES?

The term MES - Manufacturing Execution System - usually refers to a complete system, which covers the business-reporting and production-planning levels of an enterprise resource planning (ERP) – and the actual manufacturing or production process at the manufacturing or automation level. MES is originally a process-oriented level of a multi-layered manufacturing management system. It assigns processes and resources to production tasks.

The term “production control” stands for the comprehensive description of this task.

MES map the detailed processes for the respective process of the concrete production task while binding resources to a computer system.



What are the tasks of an MES?
The field of action of an MES can be subdivided into three areas:

1. The operating data refer to the machine. They are always available, regardless of whether the machine is producing or not. Among other things, waiting times, lead times, availability and production figures are recorded. Operational data should essentially contribute to increasing the throughput in the entire technical order processing by rapid repeated optimization.

2. A locking is triggered by the product to be produced, which is located in front of the machine or was fed into it. The purpose of locking is to ensure that all parameters are correctly set by the machine before starting production or machining.

3. Traceability is the end of processing a product. The process parameters and the processed material are stored in this data. They serve for retroactive recognition of production errors or quality defects.

The structures of industrial data capture are also mapped at this level. The best-known method here is probably process control using key figures such as OEE (Overall Equipment Effectiveness).

With this method, data from the production process is collected, bundled, and visualized in real time. The aim is to increase plant availability and reduce throughput times. Consistent application gives them a constantly updated, seamless picture of their production process.

Another important aspect likely to play a role in the future is the traceability of products and pre-products.

The way to an intelligent factory
With the intelligent factory targeted by the Industry 4.0 strategy, it is possible to master complexity, respond quickly to disruptions and increase production efficiency.
The MES distinguishes itself from similarly effective systems for production planning, the so-called ERP systems (Enterprise Resource Planning), by the direct connection to the different systems of process automation and enables management, steering and control of the production in real time.

Objectives of the intelligent factory by appropriate control:

- mastering complexity
- fast reaction to disturbances
- increase efficiency in production

MES is a new perspective in the context of Industry and forces the direct connection to the various systems of process automation, management, control and production control in real time. The emergence of MES structures ultimately leads to the application of Cyber-Physical Systems.

1.1.3 What is a Cyber-Physical System - CPS?

An integral part of Industry 4.0 is formed by so-called Cyber-Physical Systems, called CPS for short. CPS consist of embedded systems (such as part of equipment, products, production equipment and tools) that are properly linked to their environment. As a result, it is possible to directly record physical data with sensors, to use data and services available worldwide, to evaluate and store data and information as desired, to realize efficient provision of information or even to act upon the physical world with corresponding actuators.

Official definition of a Cyber-Physical System
A 2008 publication of Edward A. Lee introduces CPS as follows:

„Cyber-Physical Systems (CPS) are integrations of computation with physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa."

The research agenda "CPS" gives the following explanation:

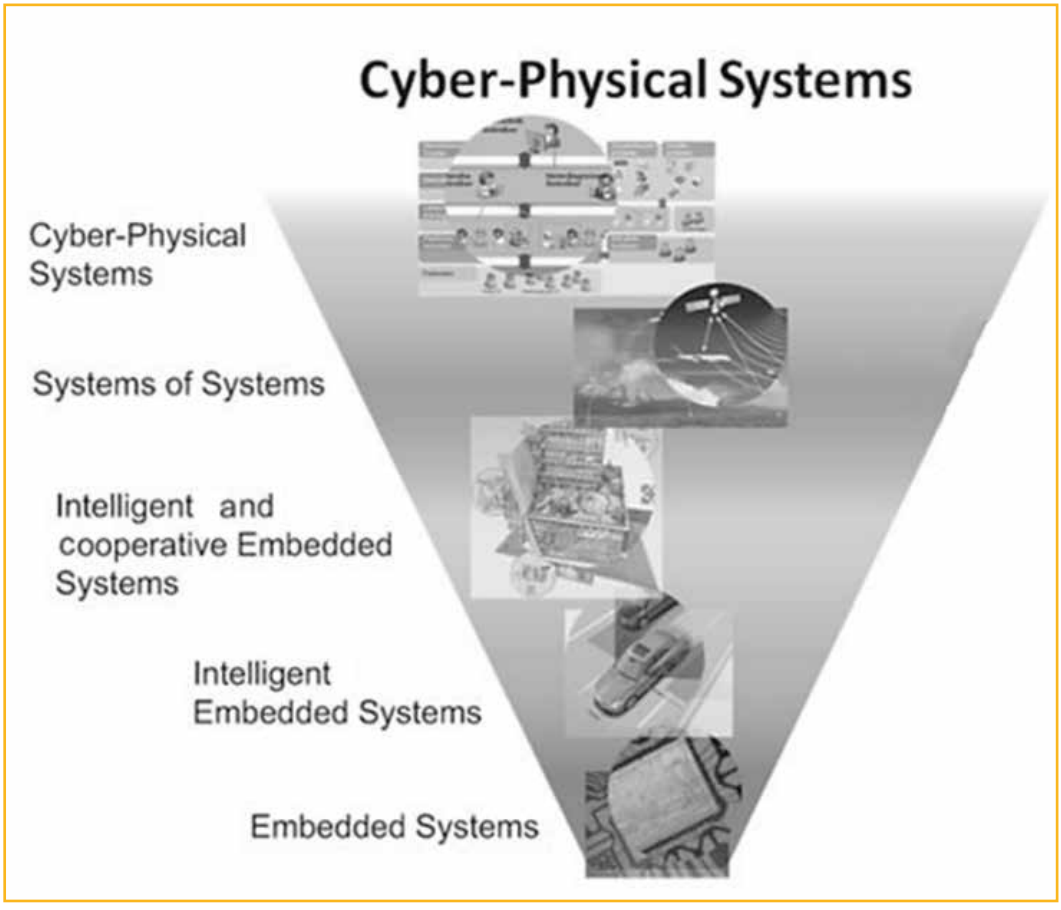
„Cyber-Physical Systems (CPS) are characterized by a combination of real (physical) objects and processes with information-processing (virtual) objects and processes via open, partially global and always interconnected information networks."

A practitioner's perspective

While automation technology only includes the technical process and the dynamics of the processes, the CPS already start at the level of sensors and actuators.

Of particular interest here is the moment from which discrete binary data is available, when sensor and actuator values have become information flows.

However, the CPS levels could also extend beyond the ERP level and integrate different systems from different manufacturers and operators via the Internet or intranet.



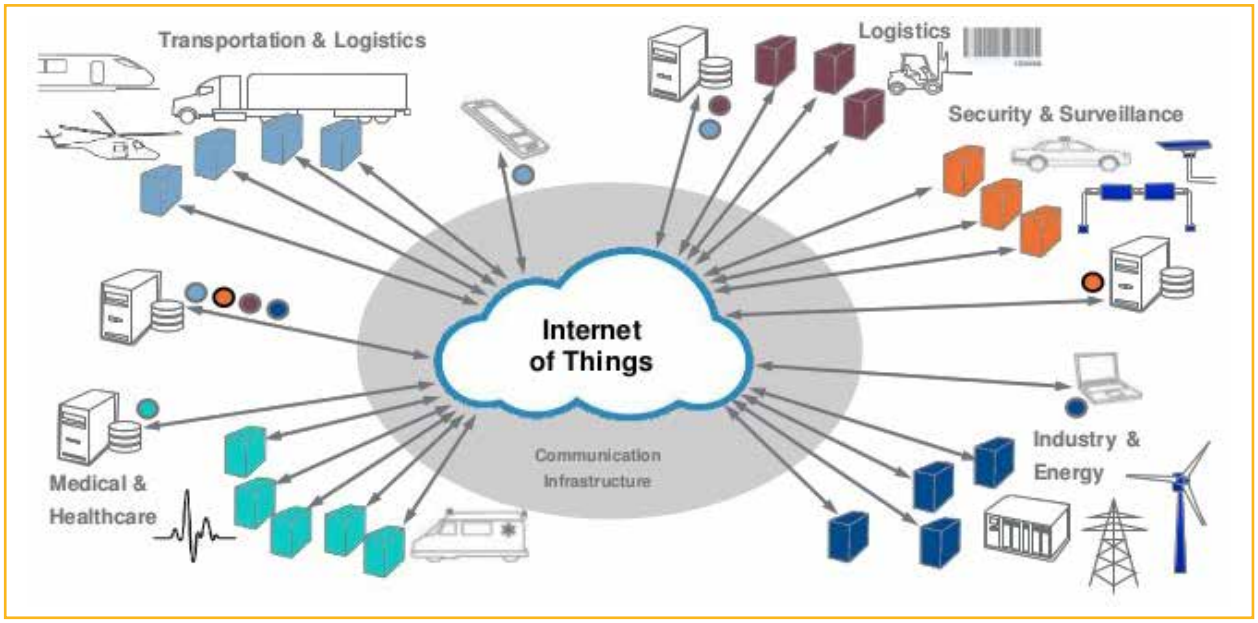
Structure of Cyber-Physical Systems

1.1.4 Basis of smart factories

Now, two crucial questions arise:

1. How can decentralization and autonomy in production go one step further?
and
2. How does the next step towards the so-called Cyber-Physical Production system succeed?

The first approach to a solution is to develop Cyber-Physical Systems that include objects, equipment, buildings, transportation, as well as production facilities, logistics components, and more. These contain embedded systems that are enabled to communicate.

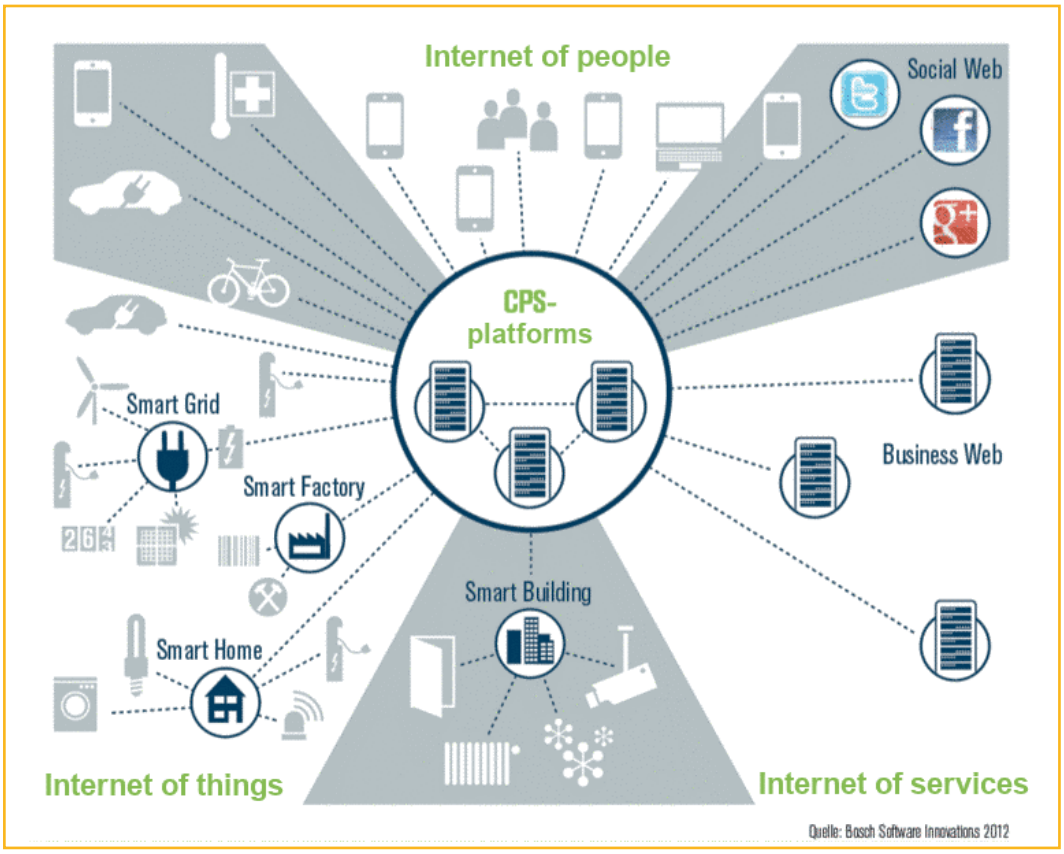


Internet of things

How do Cyber-Physical Systems work?

Cyber-Physical Systems can capture their environment directly with their corresponding sensors, evaluate them with the help of globally available data and services, store them, and can influence the physical world with the help of actuators.

The CPS-platforms form the basis for connecting the different „Internets“:
The Internet of people with **the Internet of things** and **the Internet of services**.



CPS - The Internet of people, the Internet of things, the Internet of services

Technical features of Cyber-Physical Systems

The technical features of Cyber-Physical Systems include, amongst others, the **availability of globally distributed data and services**, which can be accessed by the respective system with high availability, regardless of their location.

A particularly important feature is the **selective access protection** for production and technology data of all partners involved, as well as **data consistency** across all companies involved in the business process in all engineering and equipment phases.

Also needed are **digital networks and interfaces** for communication between the devices of automation equipment, man to plant and plant to plant, and **functional holistic models** for independent analysis, representation, organization and execution of a production process. Additionally, flexible production units that are directly adaptable to changing product requirements, and production units with the capability of troubleshooting and self-healing, including the ability to diagnose, reconfigure and interact with the real and digital world.

And, last but not least, the **provision of necessary data** for configuration, production and order processing.

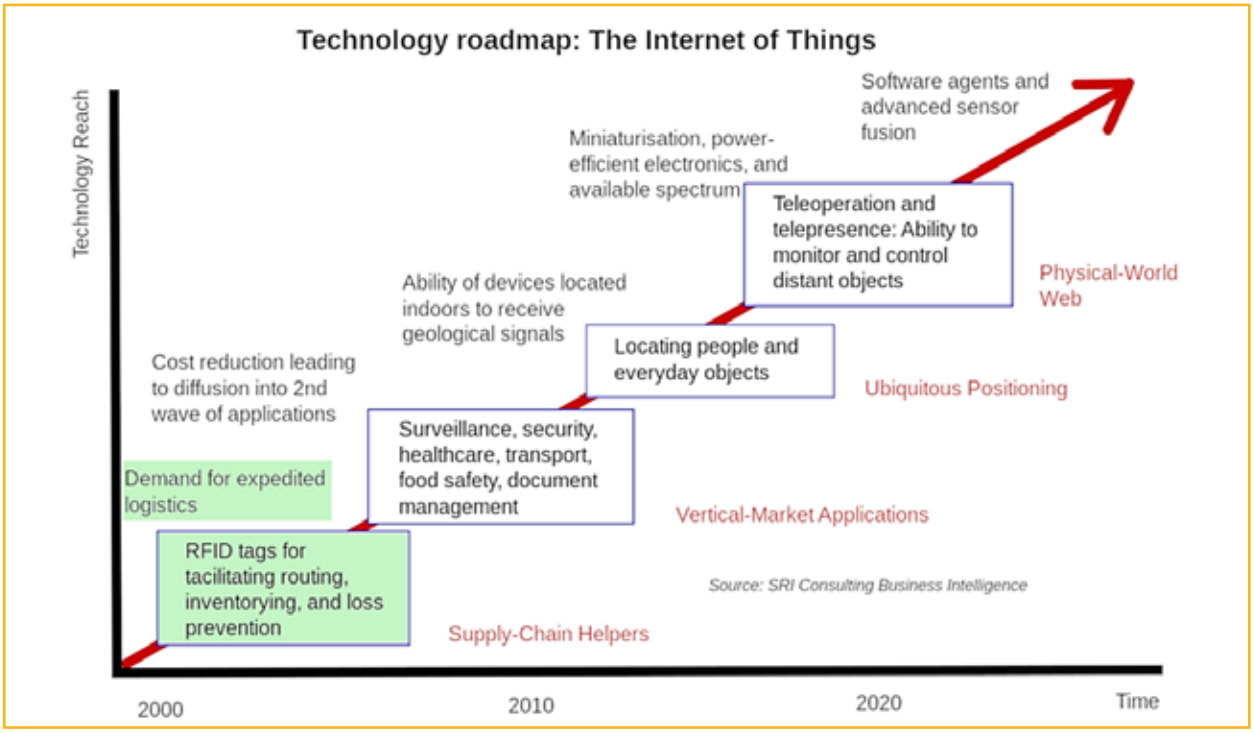
Overview of important technical features of Cyber-Physical Systems

1. availability of globally distributed data and services with location-independent access
2. access protection for production and technology data
3. data consistency across all companies involved in the business process
4. availability of suitable digital networks and interfaces for communication
5. functional holistic models for storing “past data“
6. flexible production units that can respond to changing product requirements
7. production units with the ability to troubleshoot and correct errors independently
8. provision of necessary data for configuration, production and order processing

The technology roadmap of Cyber-Physical Systems

Cyber-Physical Systems are going through different stages of development.

The first stage is still passive.

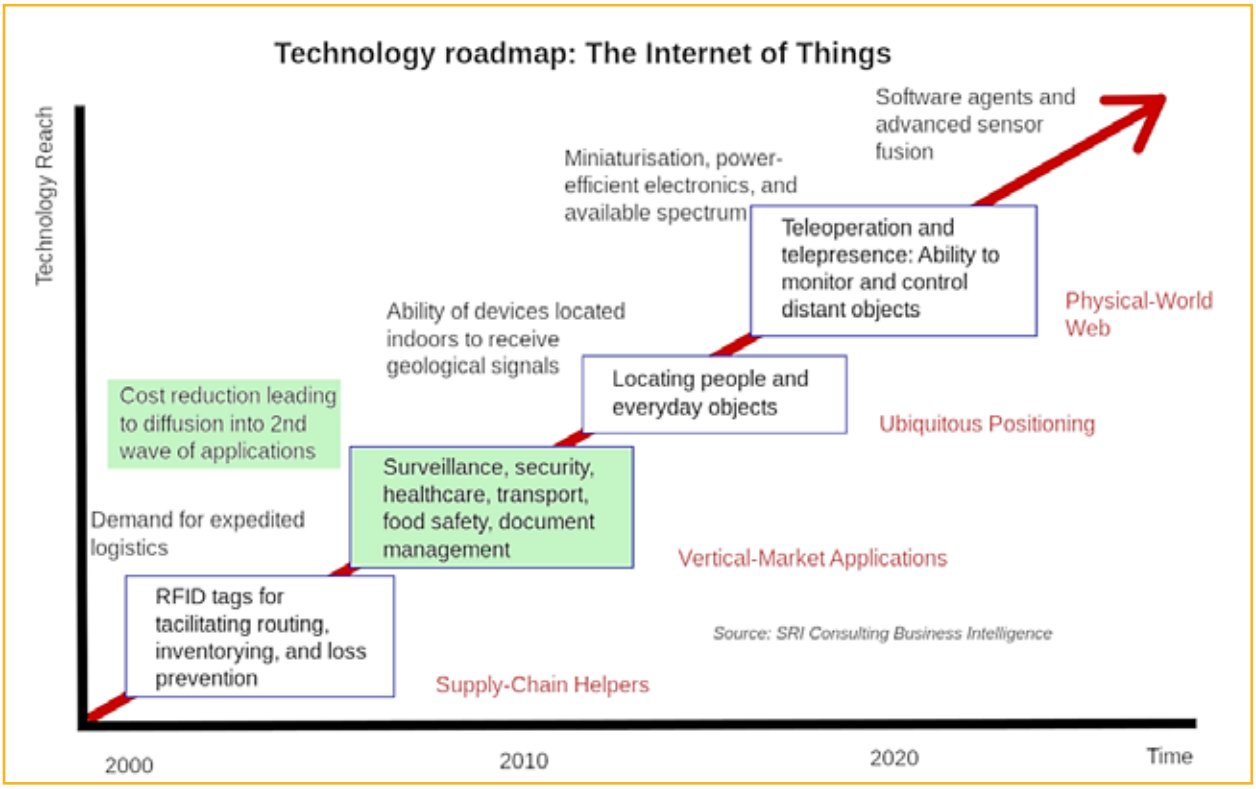


Level 1 of Cyber-Physical Systems

For example, RFID chips or smart speaker systems merely provide unambiguous identification respectively intrusion-free voice control and speech recognition, but the intelligence of the system can only be provided by centralized services.

The system itself is not yet intelligent and still has no memory or evaluation options.

The second step extends the tasks of the sensors and actuators.



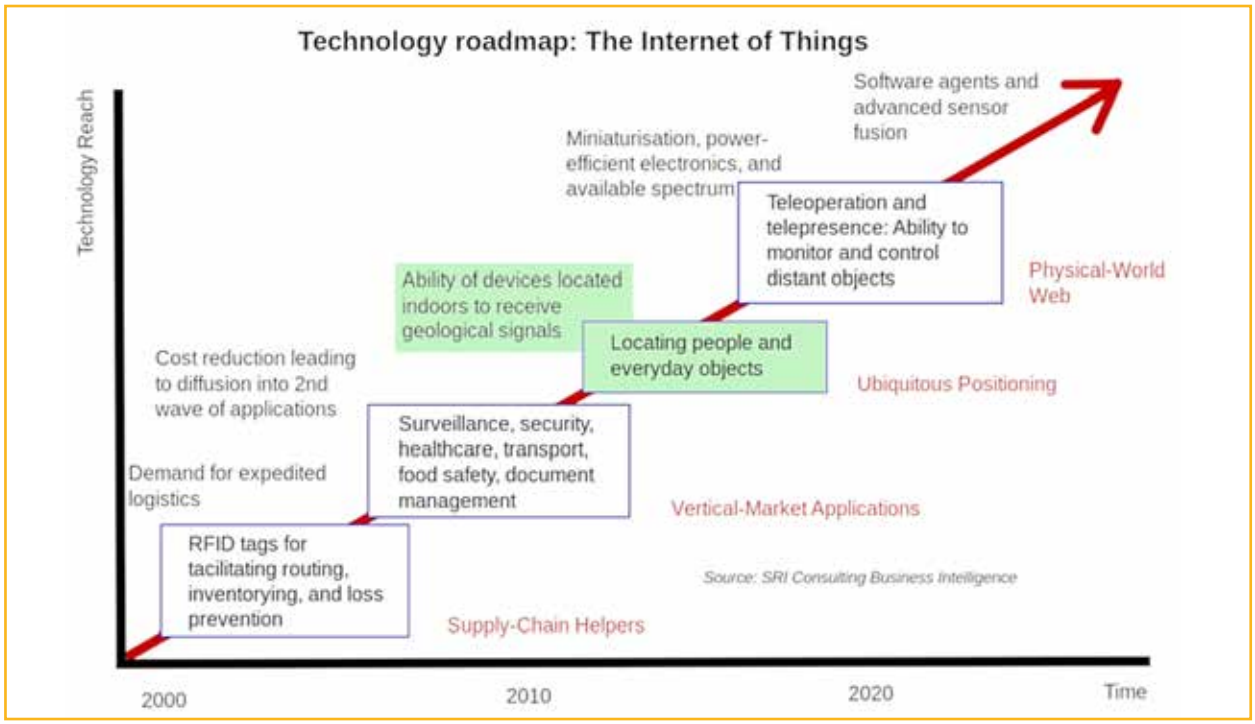
Level 2 of Cyber-Physical Systems

The next step is towards active sensors and actuators, which still have a well-defined and relatively small range of functions. The capabilities of the peripheral components (sensors and actuators) are expanding towards specialized tasks such as the pre-processing of camera images regarding the position and location of objects in an optical system or the compression and classification of huge amounts of data.

Level three: Intelligent connection of the systems

These are the intelligent, network-capable systems that consist of several actuators and sensors. They already have their own intelligence, a corresponding interface, and they make contact with other systems themselves.

In an extended sense, closed communication systems are connected with open systems at this level. Closed networks are understood to be networks set up and operated exclusively for the automation function (e.g., fieldbuses), while open networks originally or additionally also serve other communication tasks (e.g., Internet).



Level 3 of Cyber-Physical Systems

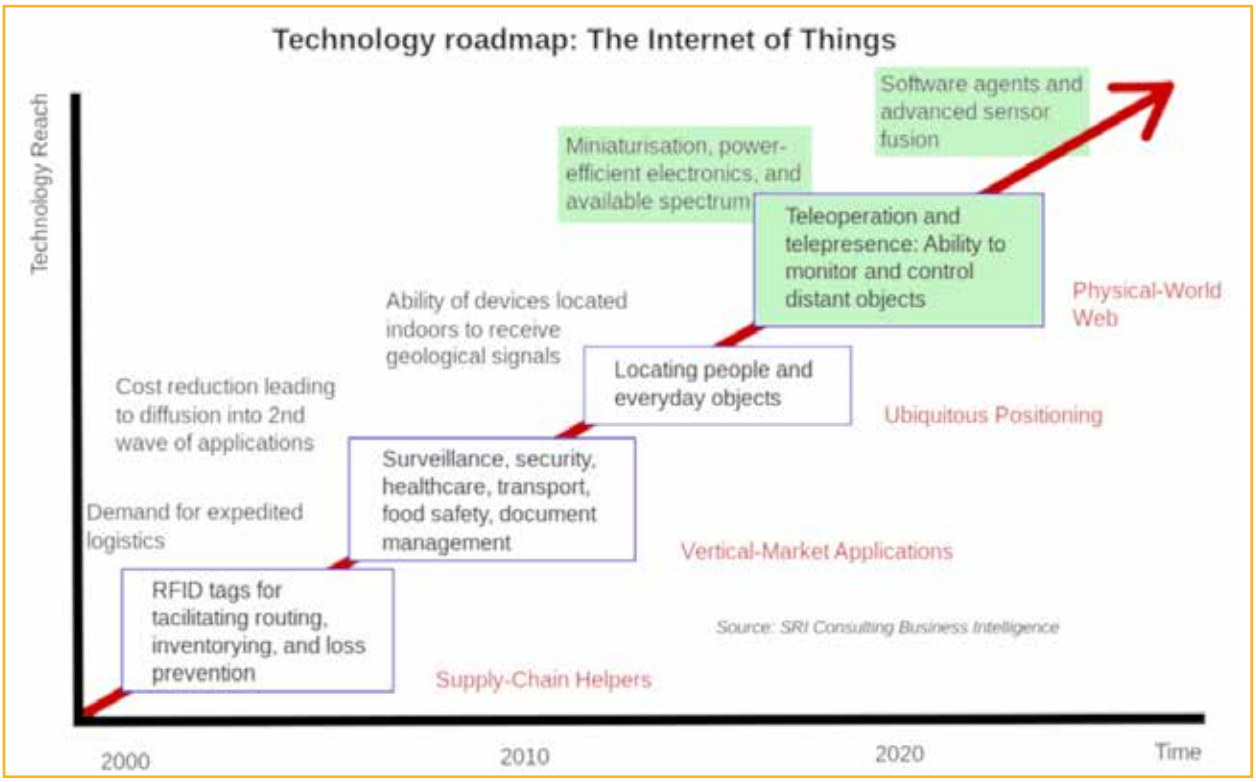
Fourth level: The "system of systems"

In the fourth stage of development, Cyber-Physical Systems can combine their individual capabilities intelligently and independently. This enables them to develop completely new capabilities and offer their own services.

In manufacturing companies, this means that in the future the areas of responsibility for production will increasingly rarely be bundled centrally and structured hierarchically, but will instead be shifted peripherally to specialized units. That will ultimately lead to much more intensive networking and closer cooperation between different companies. The classic company structure will increasingly give way to flexible and decentralized forms of organization.

Self-organizing, networked production plants recognize and configure their components and tools. Via the Internet, they independently request required process parameters from the manufacturer or from comparable plants from other users. Decentralized local memories in production facilities or products collect, store, evaluate and distribute detailed information about product creation, production processes and the product life cycle. As a final consequence, these embedded systems equipped with adequate sensor technology are capable of recording their environment and the resulting data. This data is then analyzed with the help of data already available worldwide and stored in a cloud. At best, with the support of actuators, they are also able to influence the physical world in such a way that unplanned events such as a machine failure can be reacted to immediately.

The result is an autonomous, decentralized system development.



Level 4 of Cyber-Physical Systems

1.1.5 Technical potentials through the introduction of CPS in production

By networking and arbitrarily arranging and classifying data, information, and services, it will be possible for the following potentials to become more efficient in the future:

- acquisition and processing of data of all levels of detail (including their availability and quality) at any level and subsystem of the automation pyramid
- provision of new services that evaluate these data in relation to each other in innovative ways and translate them into decisions or their preparation
- cross-discipline, end-to-end exchange of information, e.g. on product/process and production statuses with all necessary participants in the value chain
- End-to-end engineering with the ability to access all relevant information, including the design, when modifying subsystems
- dynamic integration of services and service providers
- function modules, consisting of hardware and software, which can be used as a functional unit for better structuring and documentation of an automation system
- adaptive, changeable configuration or partial self-organization of the production processes and automation systems
- faster adaptation to new market conditions and product variants via flexible replacement of individual components and plant parts
- assistance, explainability and transparency for the human user about product and plant life cyc-

le and processes
- context-dependent decisions depending on the current or predicted status of the product, the processes, the plant and the operation
- the product to be manufactured becomes an automation component itself; it communicates with the plant that produces it
- automatic optimization of systems and personnel capacities, e.g. with regard to costs, resource consumption or throughput
- self-diagnosis of components and systems with the aim of minimizing maintenance costs, downtimes and wear
- and many more

1.1.6 Theses and fields of action for the successful introduction of Cyber-Physical Systems in production

1. Automation is the leading discipline for the realization of CPS in production

The introduction of CPS in production technology and the realization of the manufacturing facilities of „Industry 4.0" will not happen in one fell swoop and on a greenfield site. Both must be developed evolutionarily from the existing automation solutions and consider the diverse boundary conditions. The competence of automation, among other things, in the targeted engineering of systems must therefore be integrated into all developments of CPS in the production area.

2. The degree of automation will continue to increase with CPS

The degree of automation achievable in the future will increase significantly as a result of open and global networking and will increase the virtual presence of automated components. The use of extensively available data, information and services will enable more functionalities and greater flexibility in plant configuration. Conversely, new products will also require a higher degree of automation and their efficient production will require Cyber-Physical Production Systems (CPPS).

3. Security and safety are critical success factors for the realization of CPPSs

Data, information and communication security is the most critical success factor for the realization and introduction of Cyber-Physical Production Systems (CPPS). The massive additional networking creates a new quality of security threats for industrial production with potential impact on aspects of functional safety. Appropriate security architectures, protection measures and validation methods are urgently needed. If this does not succeed, or if negative examples become known, the success of CPPS will very quickly be called into question.

4. Clarification of terms and standardization must take place now
The disciplines of production, information technology and automation must work together for the successful introduction of CPS in the manufacturing industry. This requires a uniform terminology. Technical regulation and standardization are needed at an early stage so that proprietary solutions do not prevail.

5. The controllability of CPS requires new methodological approaches for planning, development, and operation
CPS as presented will introduce an unprecedented level of complexity and heterogeneity into automation. To manage these, new approaches are needed for requirements management, model-based design, and optimization of complex, distributed, time-varying systems, among others. A cross-domain modelling approach is required.

6. People must be at the centre of the introduction and use of CPSs
The complexity of future CPSs must be manageable for people, i.e. in particular comprehensible. People must be enabled to „go along" with this development. Appropriate training, qualification and continuing education programmes must be developed for engineers and for people working in production.

7. New business models become possible through CPS
Due to the free arrangement of data, information and services in the CPS, new opportunities arise for business models in production (e.g. „shared economy"). These must be sought and explored with a view to increasing efficiency and value creation in production.

8. The transformation of production through CPS requires knowledge and experience transfer as well as consultation
Due to the evolutionary process of realizing „Industry 4.0", the concrete benefits for each individual company cannot yet be estimated today. That is why it is particularly important for small and medium-sized enterprises to have platforms and points of contact for the exchange of knowledge and experience, for consultations and for the qualification of their employees.

9. Research is required in all of the preceding fields of action
The necessary technical research activities must involve all the disciplines involved, especially automation, computer science and production.

All these aspects show the opportunities, risks and requirements of a heterogeneous production landscape and should be taken into account, if possible, in the design of a reference architecture as well as in the standardization activities. Particular attention should be paid to IT security and functional safety, i.e., plant safety and occupational safety.

1.1.7 Importance of microelectronics for Industry 4.0

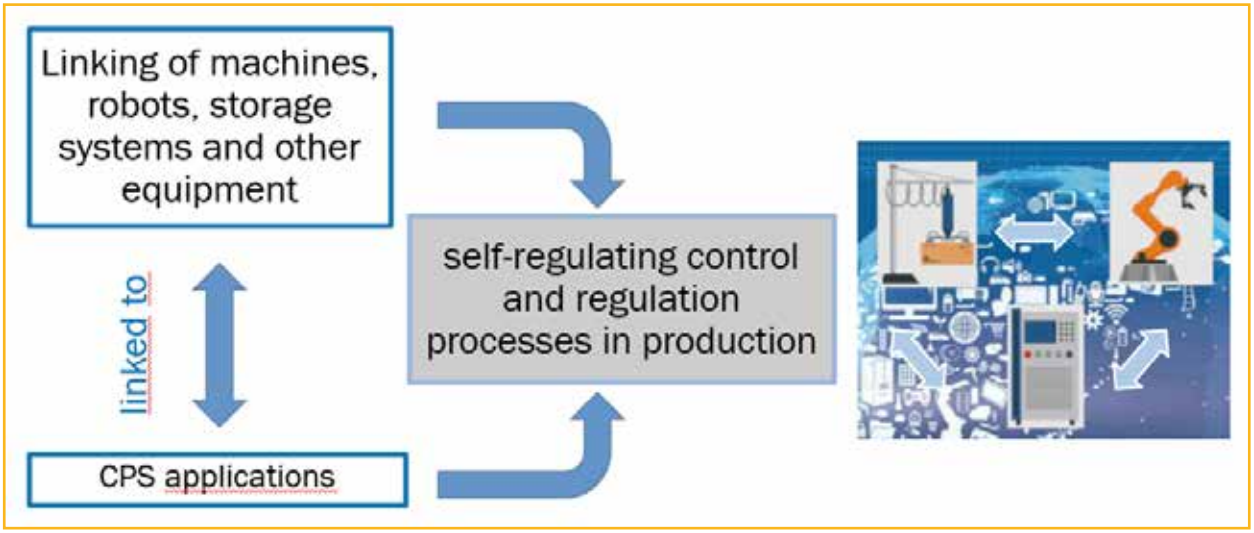
Already today, about 90 percent of all industrial production processes are supported by „information and communication technology“, or ICT for short.

During the migration to Industry 4.0, this trend towards the omnipresence of ICT continues. Autonomous, high-performance microcomputers (embedded systems) are increasingly linked wirelessly with each other and with the Internet. The physical world and the virtual world initially merge to form Cyber-Physical Systems and, in a further step, Cyber-Physical Production Systems (CPPS). CPS applications and the new Internet protocol IPv6 and 5G will enable the extensive networking of resources, information, objects and people via the Internet of Things and Services.

ICT (information and communication technology) is important for all processes:

- 90 percent of all industrial production processes are already supported by ICT
- the migration to Industry 4.0 calls for an increased use of ICT
- more and more powerful minicomputers are linked with each other and with the Internet
- the physical and virtual world (cyber space) merge into Cyber-Physical Systems (CPS)
- new protocols (e. g. IPv6 and 5G) accelerate this networking

Networking is Industry 4.0
By linking machines, robots, storage systems and other resources with CPS applications, they become intelligent resources in production.



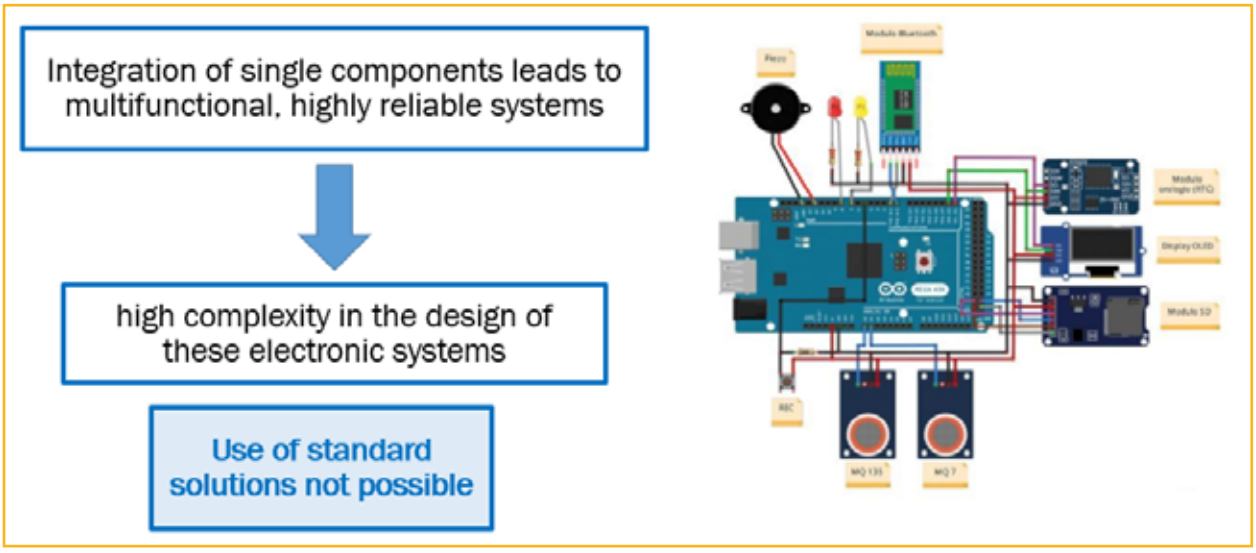
Networking is Industry 4.0

Depending on the purpose, these intelligent resources are equipped with sensors, microcontrollers and power electronics, which - to a certain extent - form the senses, brain and muscles of the intelligent systems.

When intelligent resources are connected to fast and secure IT systems by means of modern communication electronics, they can identify and communicate with each other. This enables decentralized self-regulating control and regulation processes in production.

Necessity of system integration
Only by integrating the individual components such as sensor / actuator, processor, memory, radio module, antenna, controller and power supply into a complete system, multifunctional, energy-efficient and highly reliable systems can be provided as the basis for Industry 4.0.

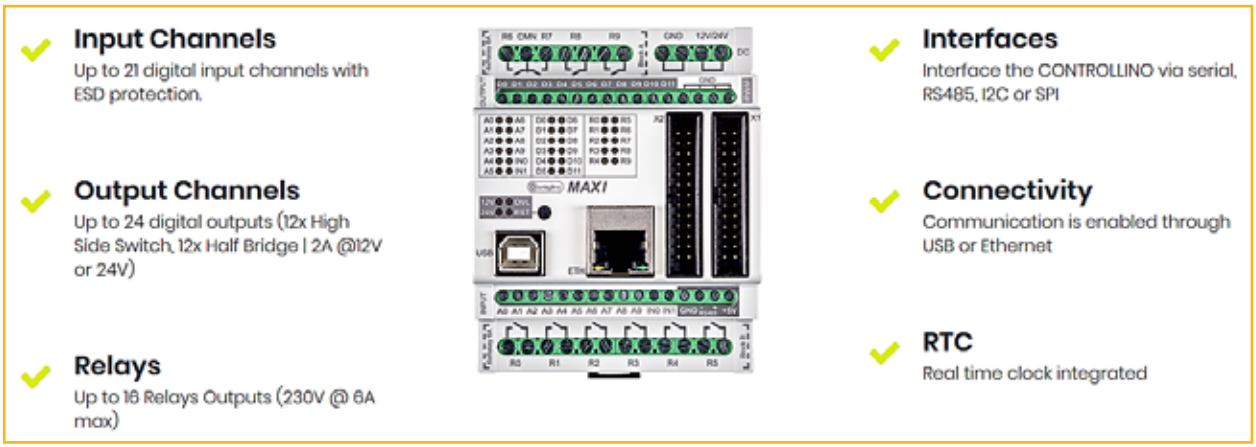
Because of system requirements that are very high in many cases, the complexity in the design of electronic systems is also increasingly significant. Standard solutions can therefore no longer be used in many cases.



System integration as a solution

Another important point is the development of embedded systems towards Cyber-Physical Systems through intelligent autonomous sensor systems and adapted industrial applications. Therefore, sensor functions, evaluation electronics, antennas for data transmission, and, above all, the energy supply are to be integrated with the aid of suitable power electronics in a very small space.

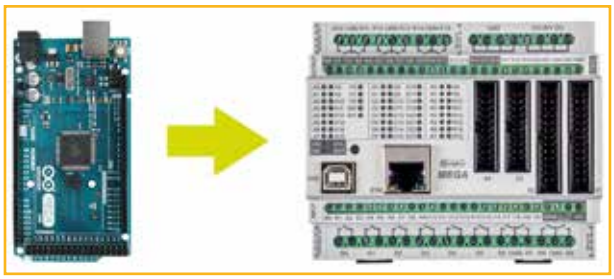
It also requires high-quality software, working in real time. Only in this way devices and objects can be linked, real states can be requested across application limits, and processes can be controlled based on this information.



Example of a small industrial control

Meaning of Microcontroller Systems in CPS Environments

In the meantime, the world of microcontrollers, which has hitherto been reserved for hobbyists and specialists, is increasingly penetrating into the industrial sector and gaining importance for the realization of embedded systems in the sense of the development towards Industry 4.0. Decisively for this is their flexibility in programming and the variety of existing communication options. For example, the Controllino, as an industrial version of the very popular Arduino microcontroller system, builds a bridge to applications in the industry.



Arduino - Industrial Automation

The Controllino was developed under consideration of the highest industrial and electronic safety standards. That makes it perfect not only for use in prototyping scenarios, but also in end products. It offers up to 61 input and output channels and relays with a voltage of 5 V to 230 V at up to 16 A. Thus, the Controllino can be used for a virtually unlimited range of automation and control applications.

100% compatibility with the Arduino system saves time and money on training and project development.

Other solutions with similar approaches but with much more performance are based on the Raspberry Pi platform. Here, too, the focus is on open source, modularity and low price.

Chapter 2

Introduction to automation

2.1 BASIS OF AUTOMATION

If you say in everyday life that something happens automatically, you can be sure that this procedure will work without the need to interfere. This also applies to a technical environment.

"Automatic" means:

- there is no need to interfere
- the process runs largely independently and without any intervention.

The Electrotechnical Dictionary DIN IEC 60050-351 defines the term „automatic" as follows:

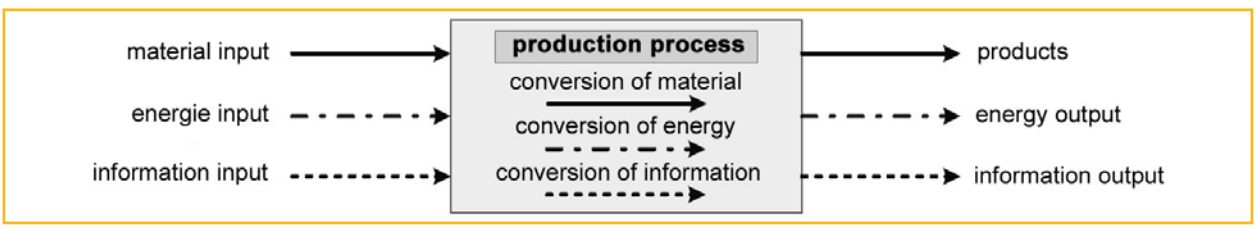
„A process or device that operates under specified conditions without human interventions is called automatic."

2.1.1 Manufacturing Automation

The term „manufacturing" includes all processes that cause the production of goods and energy. It therefore includes all machining and assembly processes and the related secondary activities such as transport operations.

Manufacturing = Production of goods and transformation of energy

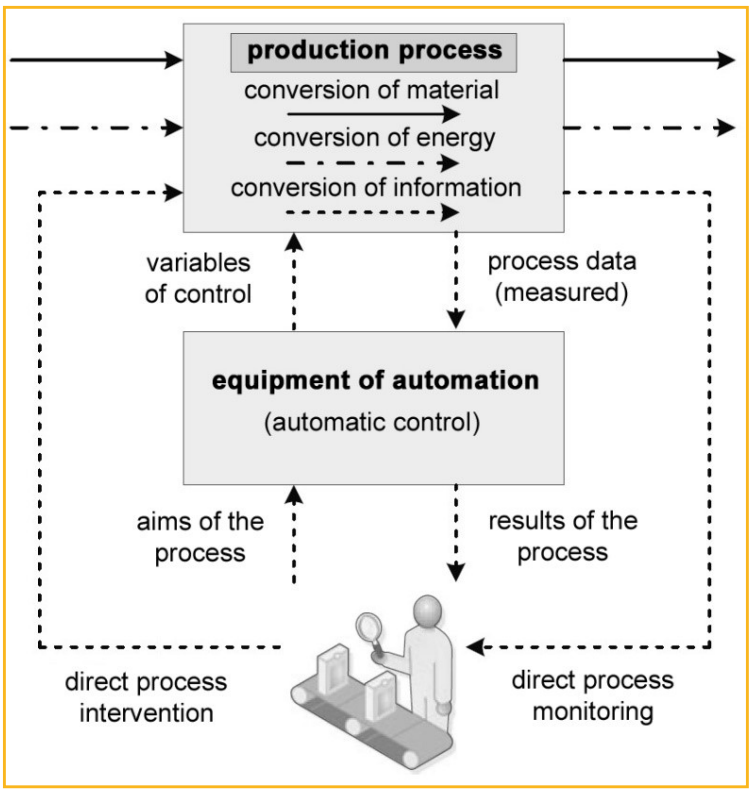
Including machining and assembly operations as well as all necessary secondary activities.



Process of manufacturing

A process takes place in which a material or product is transformed with the aid of energy and information.

If these processes are to function according to the definition of automation without human intervention, the result is the automated production process according to this diagram.



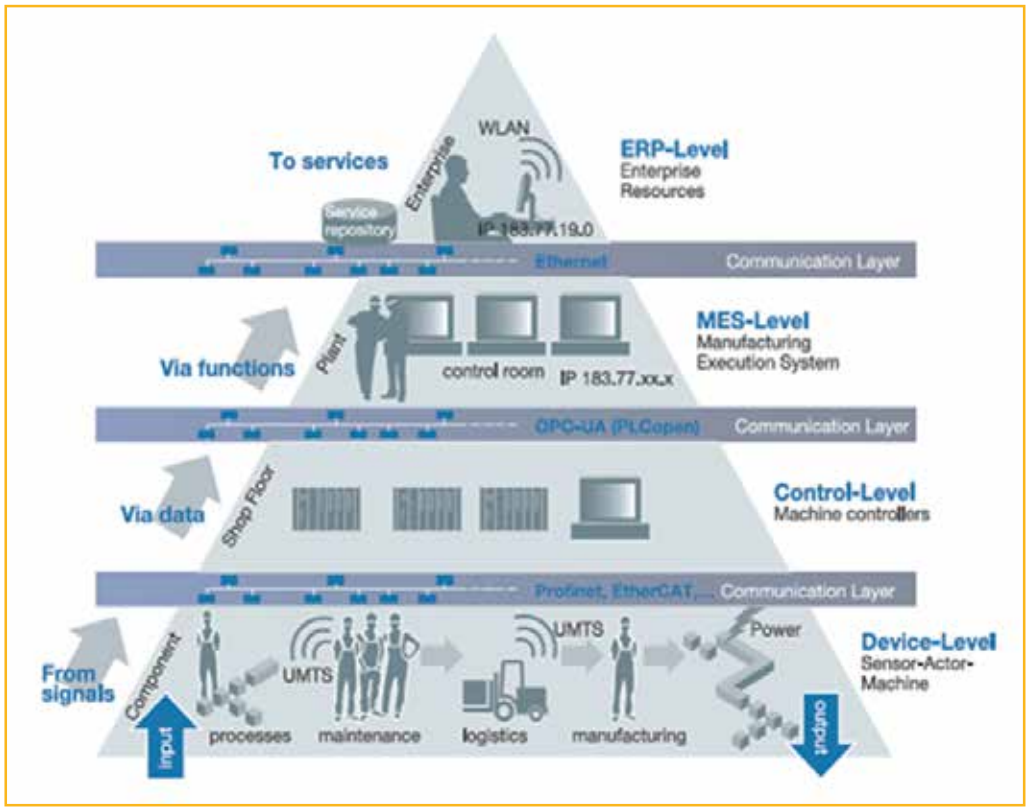
Process of manufacture and the role of the operator

Important functions of automation are summarized under the terms „control“ and „regulate“. Especially the measuring of the process results and reacting with process interventions play a special role. The aim is to achieve the defined process targets as precisely as possible by adapting the set values.

2.1.2 Automation pyramid

The automation pyramid serves for the classification of techniques and systems in control technology and represents the various levels in industrial production. Each level has its own task in production.

However, the lines between the levels are thin and can vary, depending on the operational situation. Depending on the task of the level, specific techniques of analog as well as digital data transmission and processing have developed.



Pyramid of automation

The automation pyramid:

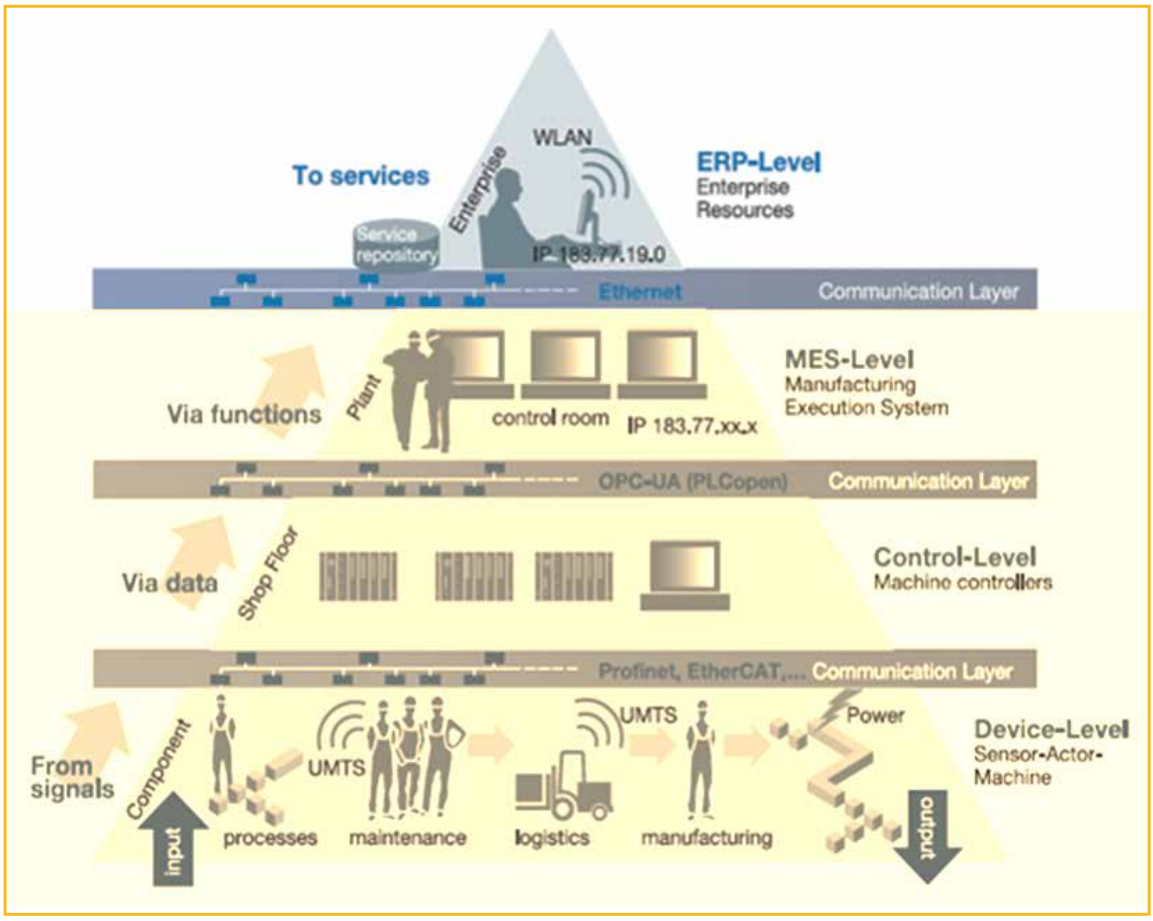
- serves for the classification of techniques and systems in control technology
- represents the different levels in industrial manufacturing
- distributes the special tasks in production on different levels
- for each level, specific techniques of analog and digital data transmission and processing have evolved

2.1.3 Levels of Automation

The current view of automation structures divides them into four levels, each with very different tasks.

Enterprise management level (ERP-Level)
At the enterprise management level, all processes that secure the company's survival in the market are executed. Tasks such as market analysis, corporate management, strategic personnel, investment and production planning are realized.

For these tasks Enterprise Resource Planning Systems (ERP) are used.



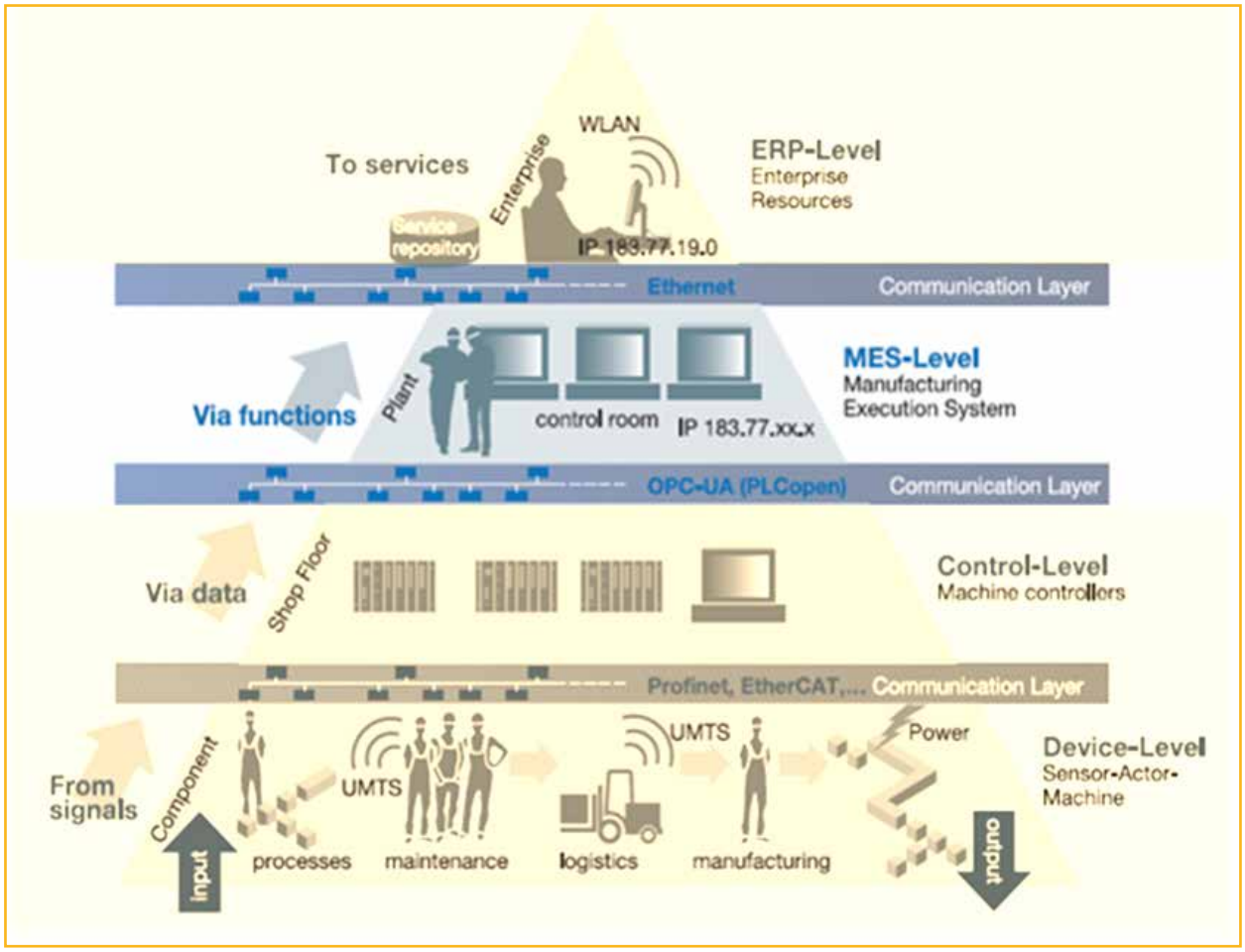
Pyramid of automation - ERP-Level

Tasks of the enterprise management level:

- market analysis
- corporate management
- strategic personnel planning
- investment planning
- production planning

Plant Operations Level (or MES-Level)
The plant control level includes for example processes of administration and processing of delivery orders, production planning, deadline monitoring and cost analysis, processes that ensure the daily operation of the company.

Also for these tasks there are software solutions such as the Manufacturing Execution System (MES).



Pyramid of automation - MES-Level or Plant Operations Level

Tasks of the plant operations level:

- processes of administration
- processing of delivery orders
- production planning
- deadline monitoring
- cost analysis
- processes of daily operation

The transition from the plant management level to the production management level is blurred. In the automation pyramid, no separate level is set up for this purpose. In the production management level, the short-term production planning, such as the operational planning of machinery, equipment and personnel is ensured.

Software solutions for this level are so-called Supervisory Control and Data Acquisition (SCADA)-Systems.

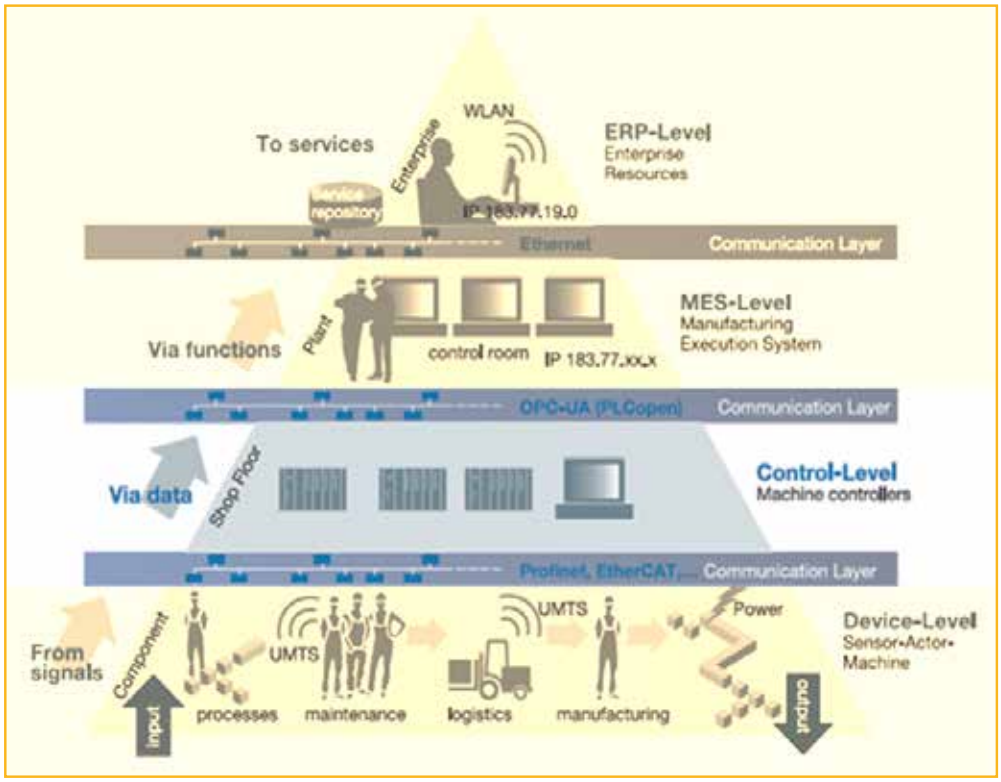
Tasks of the production management level:

- short-term production planning
- resource scheduling of machines and plants
- staff deployment planning

Process control level

Depending on the size of the plant, the process control level can be subdivided into other levels, from the plant level through the group level to the single control level. They control and regulate the production processes and their monitoring and can also realize a connection of the individual manufacturing cells.

Since the processes take place within a production area or cell, this level is also called the cell level from the point of view of the production facilities.



Pyramid of automation - Process control level

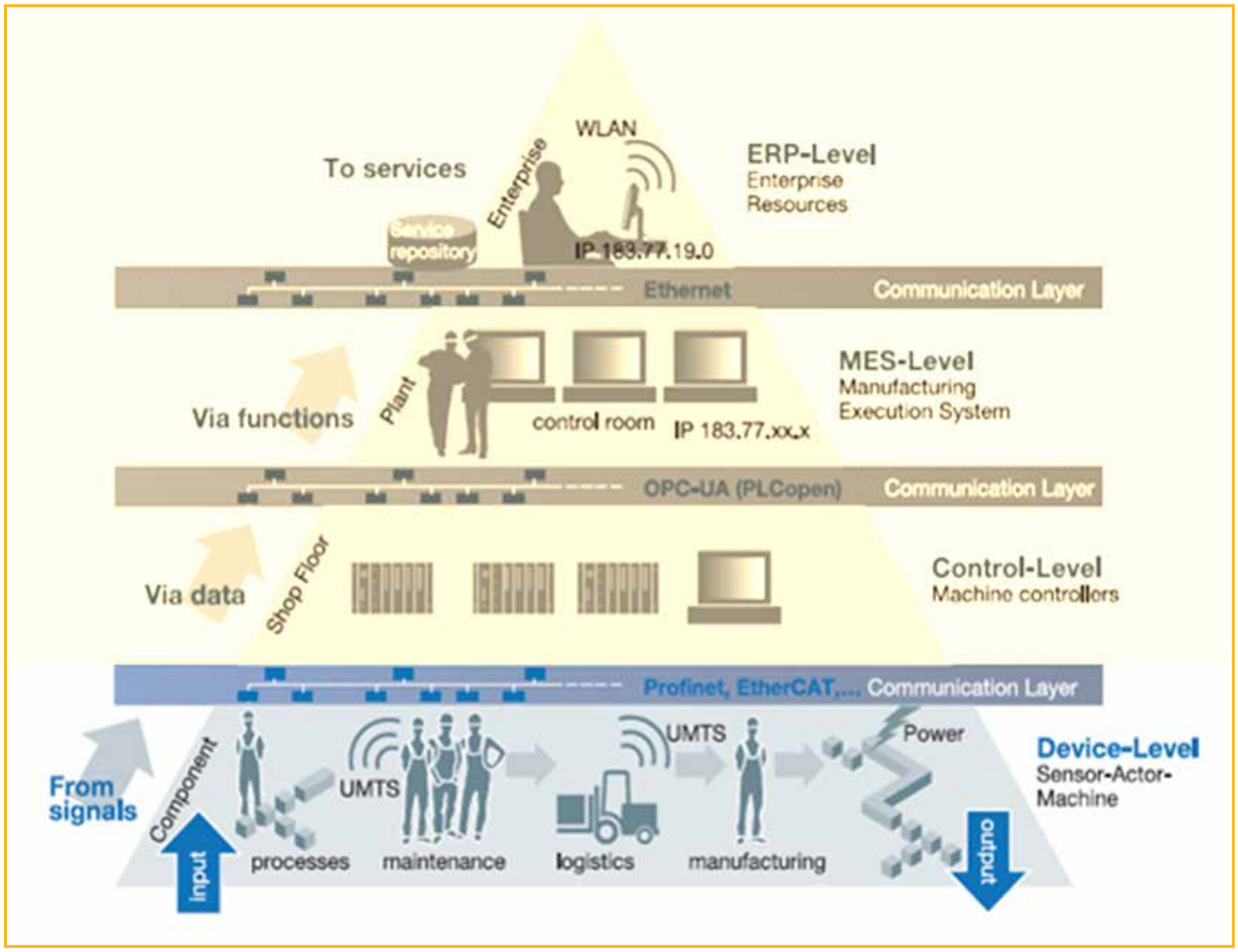
Tasks of the process control level:

- can be subdivided again depending on the size of the plant (e.g. in plant level, group level and individual level)
- it controls and regulates the production processes it monitors the production processes
- allows a connection of the individual manufacturing cells

Device level

The device level contains the actuators and sensors as well as display devices.

Here, the data are recorded and prepared and the reactions are carried out according to the evaluated information. It connects the process with the superior controls. To connect to the higher-level systems, the data volumes must be transmitted within short reaction times.



Pyramid of automation - Device Level

Tasks of the device level:

- consists mainly of actuators, sensors and display devices
- serves for the collecting and processing of data
- reacts according to the evaluated information
- connects the process with the superior controls

Assessment of the automation levels

upper levels	• complex computer systems in networks with large dimensions and many subscribers • processing large amounts of data • processing times (reaction times) are not critical
lower levels	• low network coverage and low subscriber numbers • small amounts of data must be processed at high speed • there are real-time requirements

2.1.4 Requirements for hardware and software

Each functional level has its own requirements for the hardware and software. The tasks of the device level and the process control level require real-time capable systems and standard software (e.g. modular function blocks) for the basic process control functions.

In PLC-based process control systems, powerful PCs are connected via a bus system to one or more programmable logic controllers (PLCs). The monitoring, control and regulation tasks are implemented in the PLC. Systems with a very large number of control and regulating stations can be formed.

Device level and process control level

=> real-time capable systems and standard software for basic process control functions

Process control systems

=> high-performance computers are connected via a bus system with one or more programmable logic controllers (PLCs)

Programmable logic controller (PLC)

=> implementation of monitoring-, regulation- and control tasks, mapping of systems with a very large number of control- and regulating stations

The general rule applies:

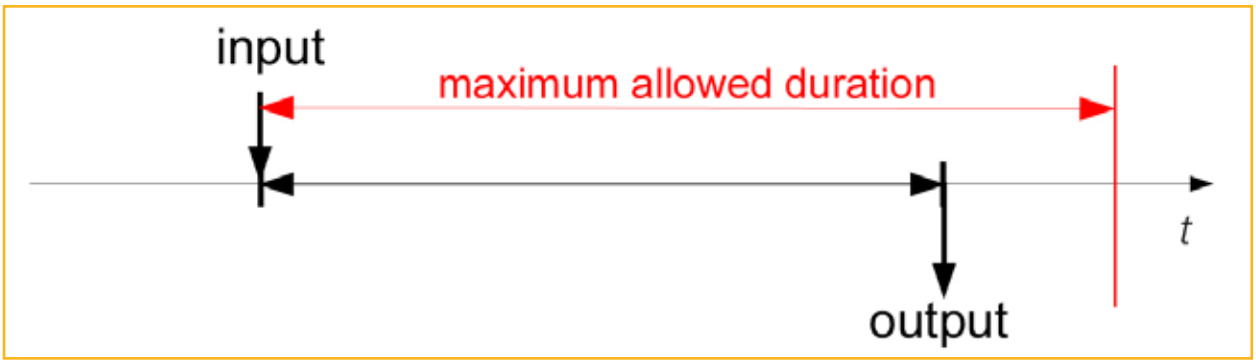
The lower the level, the more important the real-time capability of the controller!

2.1.5 Requirements for hardware and software

The term real-time says something about a system's ability to respond to an event within a given timeframe. The term does not say anything about the speed or processing power of a system.

Real-time systems are defined as „systems for direct control and execution of processes", that have to meet the demanded quantitative real-time requirements. These are used in various technical fields, for example in process control technology, in motor control systems, in satellite systems engineering, in signal and point control systems, in robotics and many other areas.

In controlled systems, there is often the requirement that a result is guaranteed to be calculated within a previously defined time interval. The size of the time interval does not matter.



Real-time capability

Requirements for real-time operation

According to DIN IEC 60050-351 7-2013 (draft), real-time capability is the ability of a process control system to keep the computational processes constantly running, so that they can react to events in the course of a technical process within a given time interval.

> In real-time operation, the cycle time must be so short that changing the signal states at the inputs will result in a timely response at the outputs.

Response time

The response time is the time between the change of an input signal and the reaction by changing the output signal. The response time is significantly influenced by the cycle time of the PLC and by the signal type of the input or output (analog or digital) and the components involved on the input or output side.

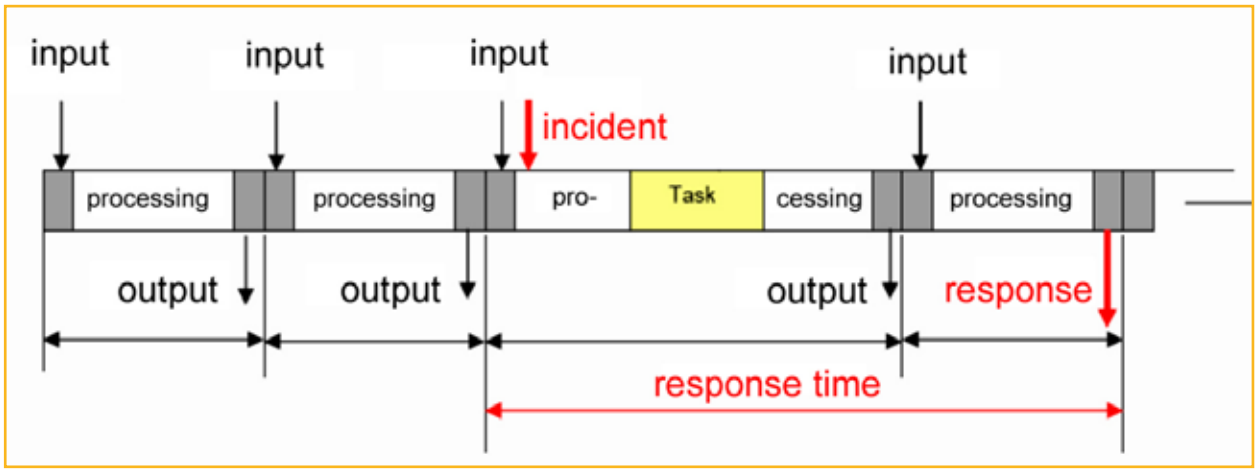


Diagram: Response time of a PLC

In the worst case, the response time to a binary event is the sum of twice the maximum expected cycle time, the delay time of the binary input and the reaction time of the actuator.

Response time

=

delay input

+

2 • cycle time of the PLC

+

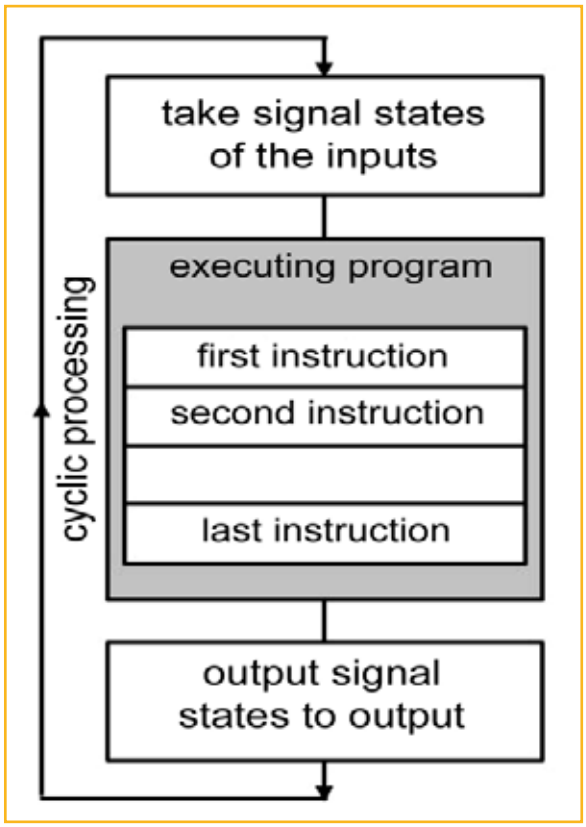
response time of the output

Worst case calculation for the response time

The delay for a binary input is usually about 2-5 milliseconds, for analog signals - if they still need to be converted -, it is much longer. The response time of an actuator depends essentially on its nature. Typical times for a solenoid valve, for example, are about 20-30 milliseconds.

Cycle time

The cycle time of a PLC is the time for processing a cycle including all communication tasks. It depends on the processing time for the programme, that is, on the number of instructions and on the processing power of the central unit.



Cycle time of a PLC

cycle time
=
Intake of signal states of the inputs
+
Execution of the programme
+
dumping signal states to outputs

Calculation scheme for the cycle time

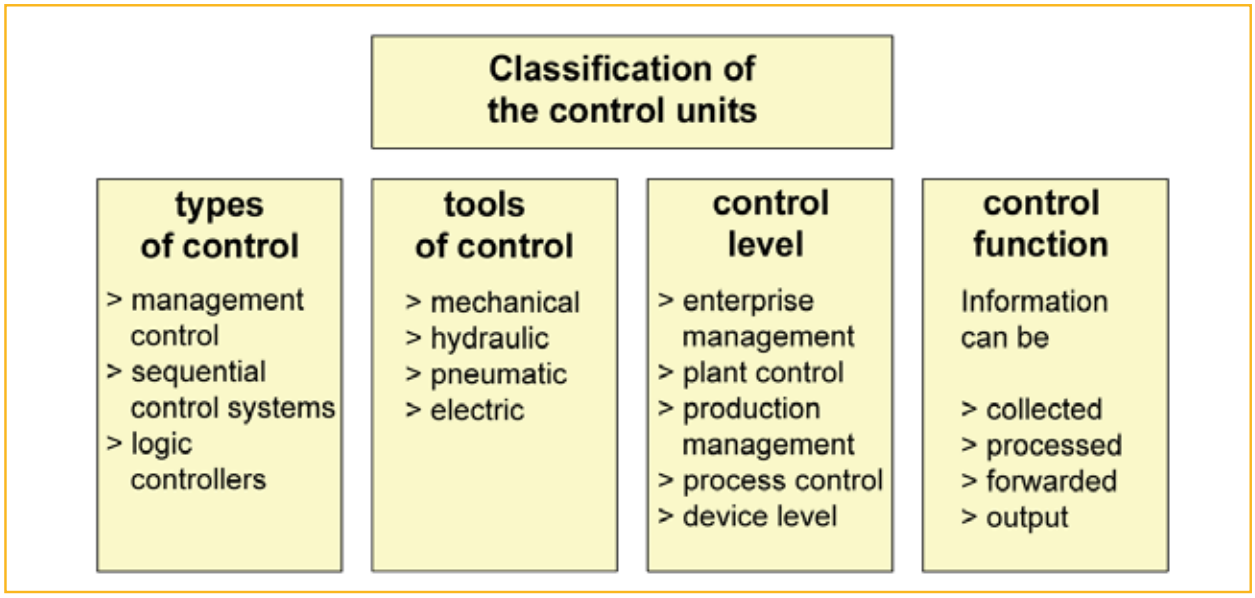
Higher priority tasks can interrupt the cycle and thus extend the cycle time. The cycle time is normally monitored. If it exceeds an adjustable cycle monitoring time (typically 150 ms), the CPU will stop.

The manufacturers provide values for calculating the computing time.

2.1.6 The wide field of control units

The classification of controls can be made according to different criteria.

Among the main classification criteria are the types of control, the tools of control, the control levels and the control functions.

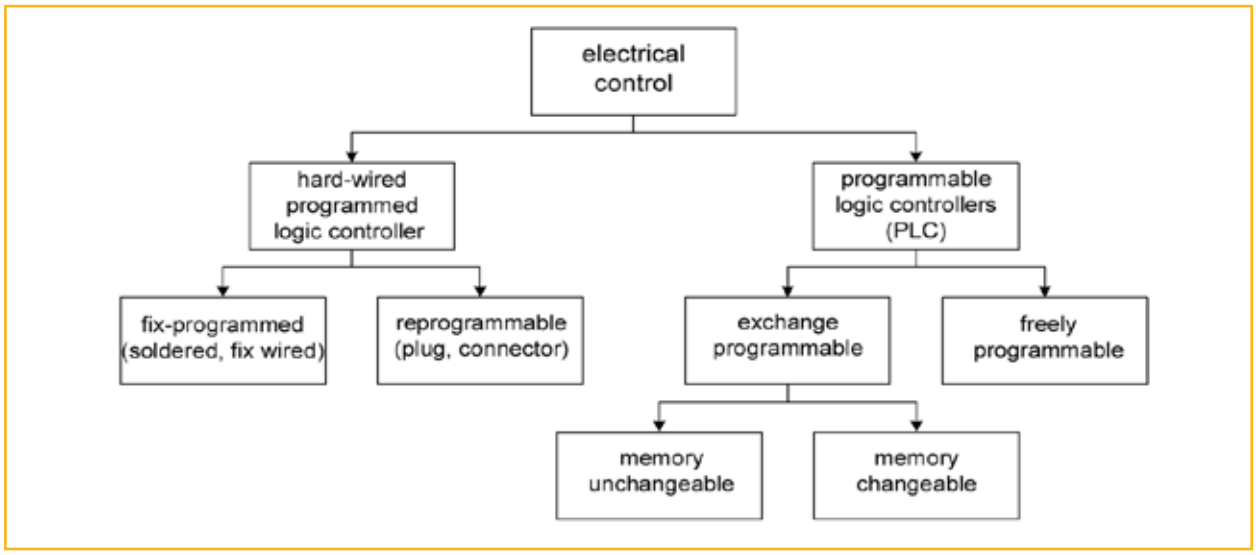


Classification of the control units

The types of control are subdivided into management control, sequential control systems and logic controllers. In the tools of control, you distinguish mechanical, hydraulic, pneumatic and electrical effects.

The control levels are mapped by the already discussed levels of the automation pyramid, and the control functions mainly relate to the way information is handled. Information can be collected, processed or forwarded and output.

Classification of electrical controls
The electrical controls can be divided into two main areas, the hard-wired programmed logic controller and the programmable logic controllers (PLC).



Classification of electrical controls

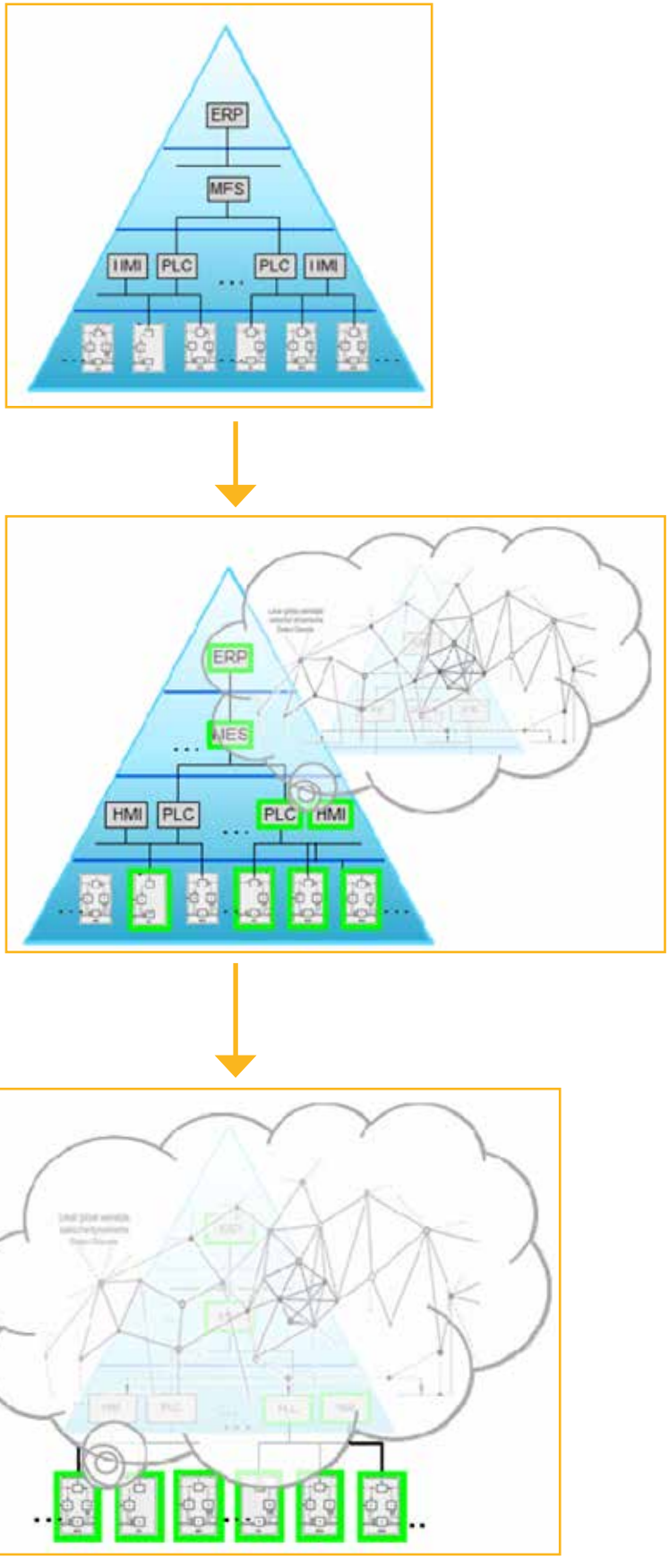
The far more important area in automation technology is the field of programmable logic controllers (PLC) and especially the freely programmable controllers. They are universally applicable and quickly adaptable.

2.2 THE PATH FROM THE AUTOMATION PYRAMID TO SELFORGANIZING STRUCTURES

If one assumes the unstoppable transformation from automated systems to cyber-physical production systems (CPPS), then the use of small, powerful, and communication-based units at the field level is the logical consequence. In CPPS, data, services and functions are held, retrieved, and executed where it is most advantageous in terms of flexible, efficient development (incl. design and engineering) and production.

This will no longer necessarily be at the classic automation levels. For example, process data could also be obtained via services in a so-called „automation cloud“ instead of via sensors at the field level. This gives the data-collecting structure a much wider reach.

This leads to the hypothesis that the automation pyramid, which still predominantly exists today, will be gradually dissolved by the introduction of networked, decentralized systems and that the various levels will no longer exist both for the structure of the hardware and networking and for information processing and engineering. Services, data and hardware components can be distributed to any nodes of the emerging network and thus form abstract functional modules from which the automation system is built.



Change in automation structures

2.2.1 Rethinking data structures and processing

In addition to the functional structure of an automation system, the classic automation pyramid shows the compression of data and information in the individual nodes. With CPPS, the automation pyramid is gradually abstracted to its functional structure by the possibility of using and providing decentralized services in the various nodes.

These new structures have also given rise to new terms such as edge computing, edge devices, embedded system and so on.

What is Edge Computing?
In computing, the term **„edge“** refers to the outermost edge of a technical information network, where the virtual world of a computer network meets the real world. In most cases, the term „edge“ implies that the network is the Internet.

The term **„edge computing“** is inextricably linked with the term **„cloud computing“**. The term **„cloud“** refers to the totality of central computing units distributed globally over the Internet. These computing units, also known as servers, do not themselves interact directly with the real world, but are connected to each other via the network and to the devices at the edge. These devices are called **„edge devices“**.

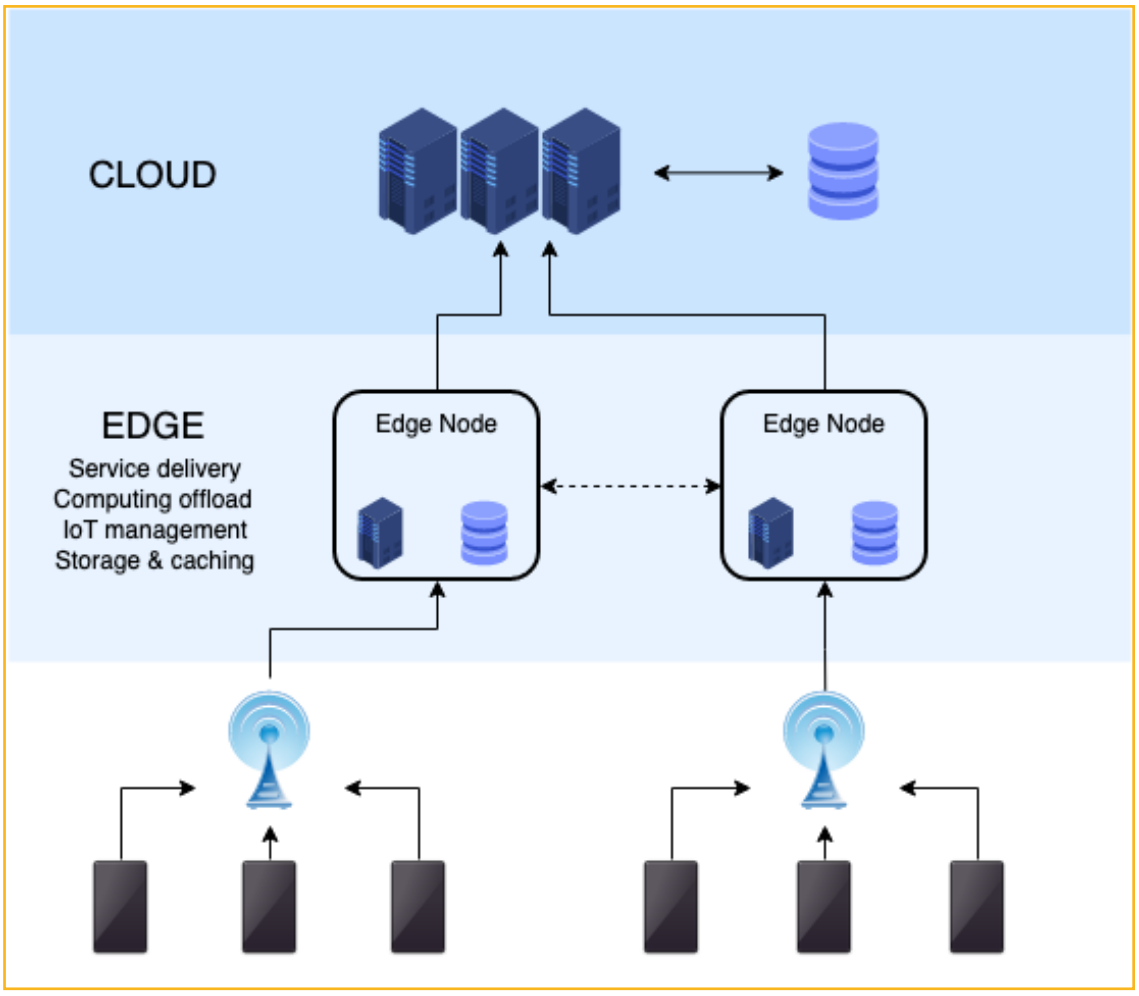
Examples of typical edge devices:

Any type of networked intelligent system equipped with microcontrollers, sensors and actuators can be considered an edge device. These systems are usually referred to as **„Smart City“**, **„Smart Grid“**, **„Smart Building“** and **„Industrial IoT“**.

Platforms for industrial edge applications already known from the Maker scene are increasingly gaining acceptance. The single board computer **“Raspberry PI“**, for example, represents a full-fledged Linux-based Internet node. For simpler applications, a board by Arduino also works. Both platforms are also often used in prototypes for commercial applications. Industrial devices with PLC approvals and DIN rail mounting are already available on the market for both platforms.

How does „edge computing“ work?
Sensors and actuators are located at the „edge“. Sensors capture the real world. Actuators manipulate it. Analog measurement data is converted into digital parameters at the edge and digital manipulated variables are converted into analogue output signals.

Edge computing means that captured data is not first forwarded for central processing on a server but is already processed on the spot. In this way, a microprocessor can derive appropriate actions directly from the acquired measurement data and control the actuators.



Structure of edge computing

Examples of near-local reactions by the „Edge“ are:

- A temperature sensor detects the room temperature and then switches a heater on or off - depending on the control algorithm and target temperature.
- An angle or distance sensor detects a position and controls a motor accordingly.

However, this small-scale event-based control does not mean that the data collected is not transmitted to higher levels or to the cloud, or that the nature of the response is altered by higher-level structures. The „Edge“ reports the system‘s values and responses to the cloud and receives modified and optimized control algorithms from the cloud as needed.

How are „embedded computing“ and “artificial intelligence” (AI) related?

The term „edge computing“ has only become established in recent years with the increased application of “artificial intelligence” (AI) or neural networks.

The wide availability of data and computing power in the „cloud“ has led to great advances in the field of artificial intelligence and machine learning.

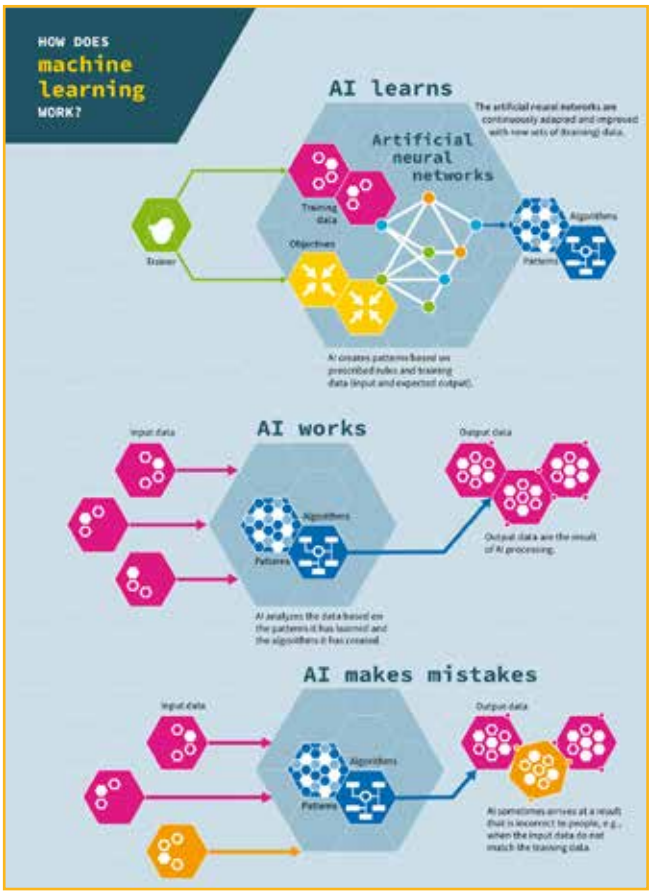
One can now use the almost unlimited data processing capacity available in the cloud to train ever more extensive artificial neural networks for demanding tasks.

These machine-learned algorithms have led to breakthroughs in a wide variety of fields.

These include:

- Natural language recognition and processing
- Text recognition and text generation
- Object classification and tracking with imaging sensors
- Recognition of patterns and prediction of trends in large data sets
- Authentication of individuals based on biometric characteristics

Training for a specific task is time-consuming, even on powerful servers, and can sometimes take days or weeks. At first glance, it seems obvious to locate artificial intelligence (AI) exclusively in the cloud.

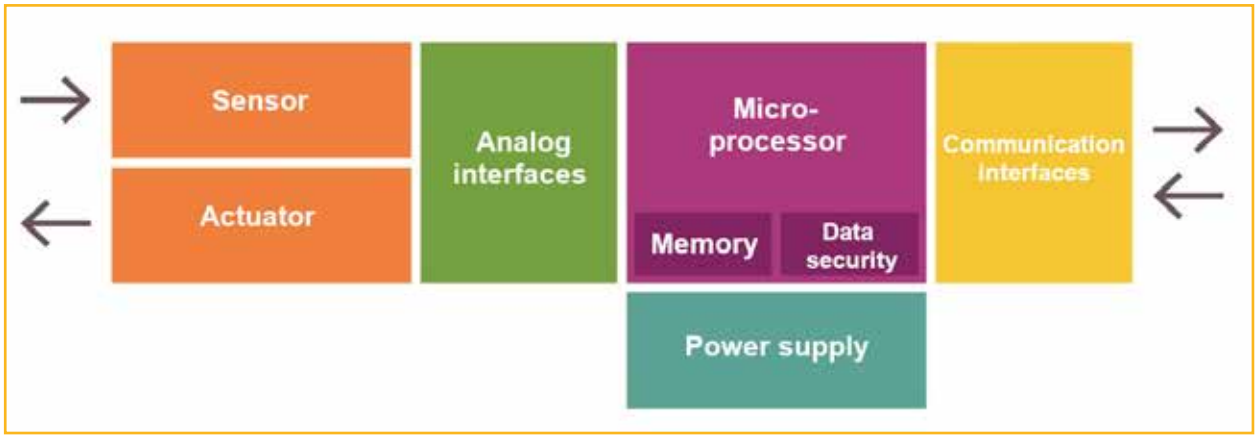


Artificial intelligence - How does machine learning work

However, the pure execution, so-called „inference“, of trained neural networks requires much less computing power than the training itself. This fact, together with modern hardware-friendly network architectures and tools for the efficient mapping of such networks on embedded systems, allows many artificial intelligence applications to be executed directly at the edge.

What is an „embedded system“?

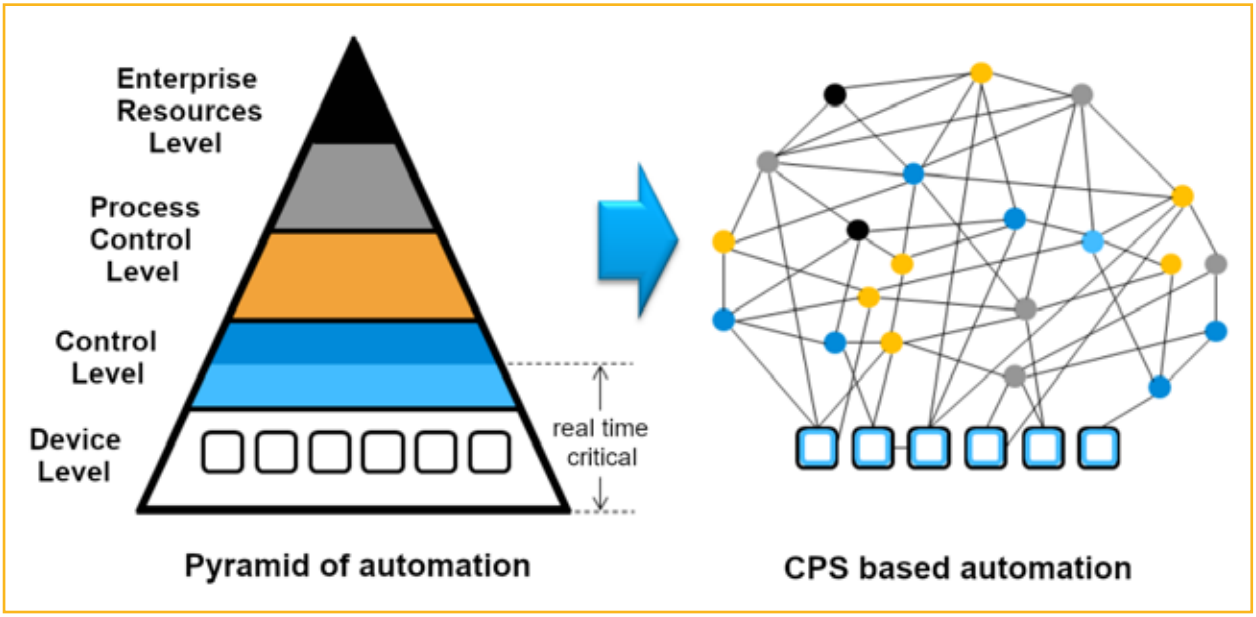
The activities described above take place in „embedded systems“. An „embedded system“ is usually an integrated system consisting of sensors, actuators, analog/digital conversion and a processor. In the form of an edge device, the embedded system also has a communication or network interface to the cloud. This can be used to transmit recorded or locally preprocessed data to central servers or to receive commands and configuration data.



Embedded system

Final phase of restructuring

Real-time-critical open and closed-loop control systems will initially remain close to the process at the field level. In the future, however, it is also conceivable that real-time-relevant requirements can be met in a distributed manner by new architectures in the CPPS.



Old and new structur of automation

Conclusion

All these considerations result in an increasing importance of small processing structures, which allow more and more computing power or functions on smaller and smaller space at rapidly decreasing prices. This trend is accelerating the shift toward small, independent control units with communication capabilities.

A Raspberry Pi model microcomputer already has many times the computing power of earlier mainframe computers, and the development continues.
This fact and the changes in automation structures described above will inexorably accelerate the shift to small controllers with communication capabilities.

Chapter 3

Microcontroller - The key to automation

3.1 INTRODUCTION MICROCONTROLLER

3.1.1 What is a microcontroller?

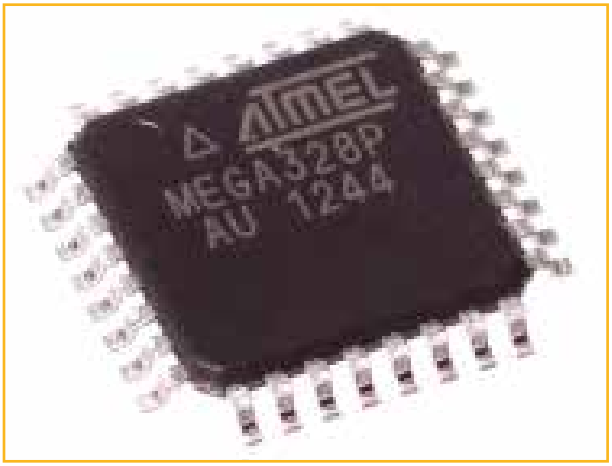
Our everyday devices contain many electronic and logical components that have a pronounced black box character. This includes signal receiving components such as sensors, and executing components such as actuators. Between these components there is a unit which processes the incoming signals, passes them on to the executing components and thus controls them. A microcontroller is such a unit. It works like a small computer that can record sensor data, process it and output it via actuators.By programming certain sequences and commands, the microcontroller can execute instructions. It acts according to predefined instructions.

What does a programmable object need in order to be able to implement these instructions?

A computer has a computing unit, a memory, various interfaces (for example a USB port), a controller and various converters. To operate a microcontroller, a similar periphery must be available. This is usually provided on a small circuit board, called a microcontroller board. Therefore, a microcontroller board can be considered as a very small computer.

Profile of a microcontroller:

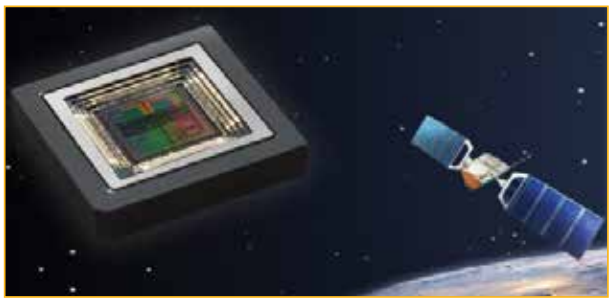
- small computer, integrated into a single chip: CPU, RAM, EEPROM and other peripherals in one unit
- excellent for embedded applications
- low cost and low power consumption
- can operate at low frequency from a few MHz up to several tens of MHz clock rate
- can work as a standalone device (on a board)
- most of them only need power supply and a clock generator to run



Microcontroller Atmel Mega 328p

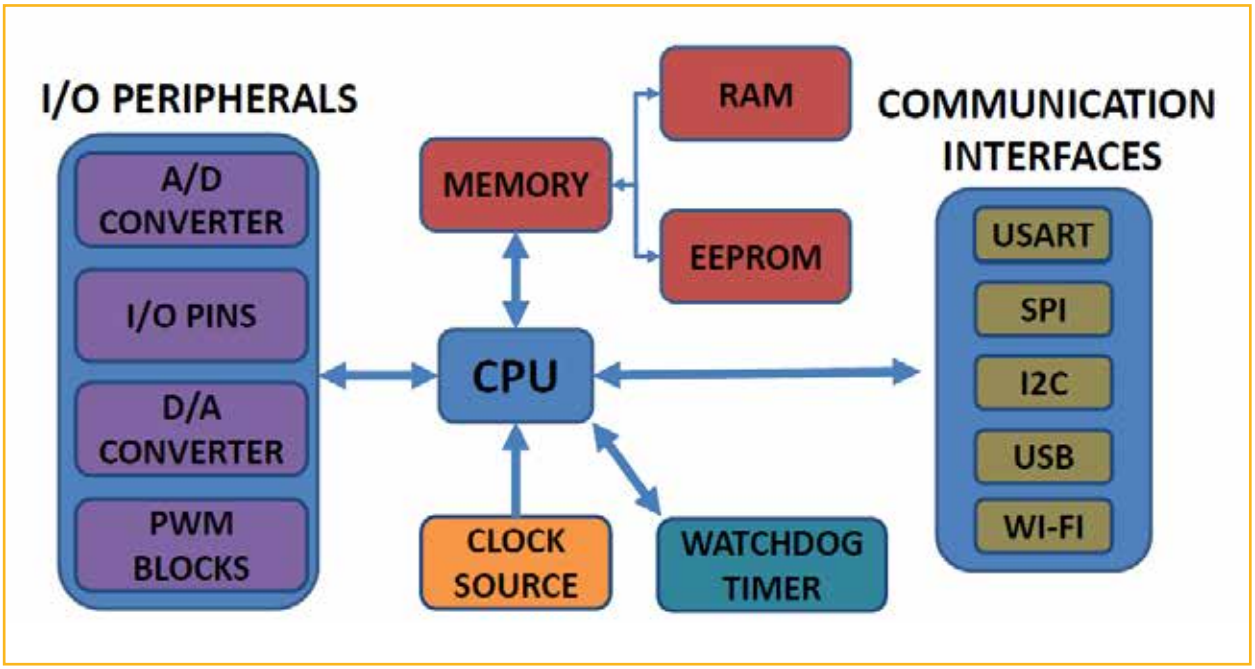
What are they used for?

- data control
- data acquisition systems
- power control and monitoring
- motor control applications
- smartphones
- automotive industry
- space applications
- etc ...



Use of Microcontrollers

Microcontroller Architecture



Simplified Architecture of a Microcontroller

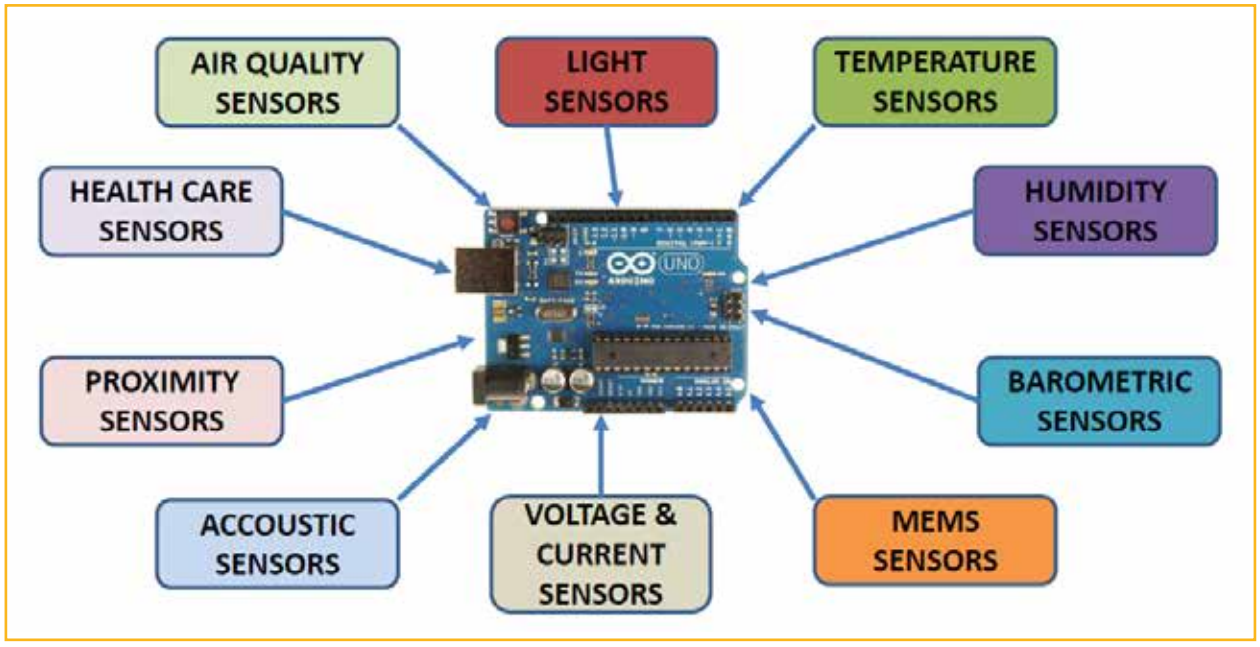
3.1.2 Arduino

What is Arduino actually?

Arduino is a microcontroller. But this is not quite correct, because a microcontroller looks like the one in the picture above. Confusing? It is because of the name. Arduino is an Italian company that manufactures boards as mentioned in the chapter before. Boards called Arduino Uno, Nano, Mega, etc. The most common board is UNO. Which, however, is being replaced more and more by the Mega Board.

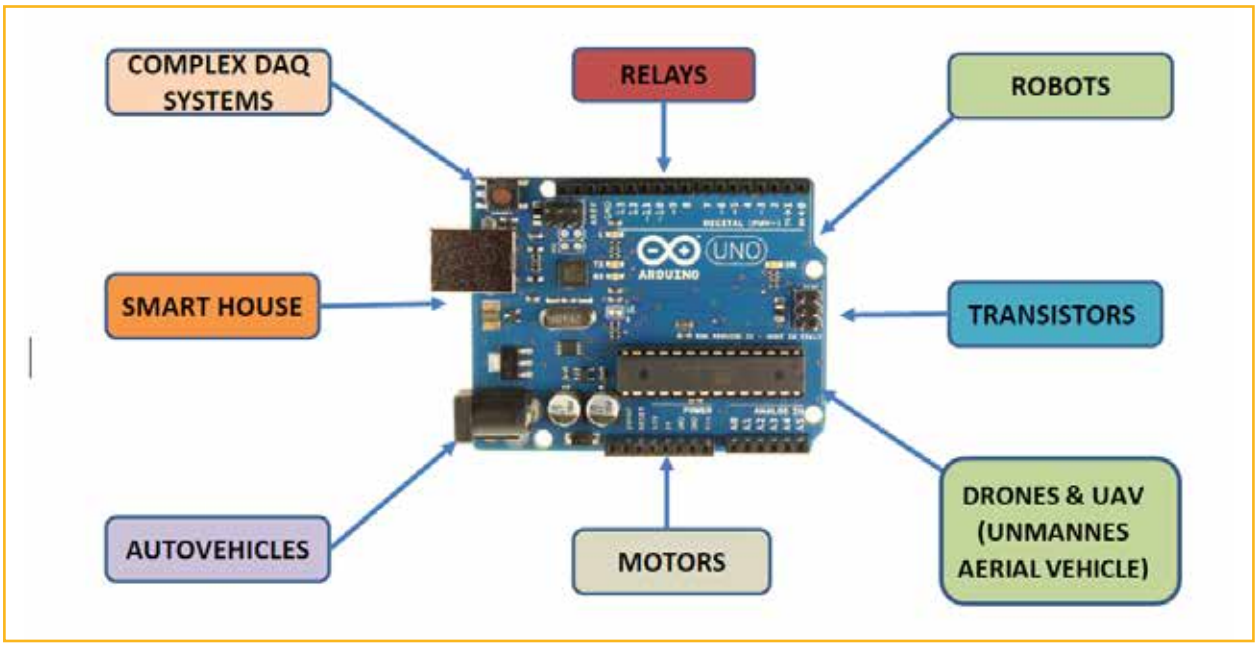
Accordingly, the Arduino is not a microcontroller, but a combination of microcontrollers and some other components on one board. Microcontrollers are basically tiny computers that can perform simple tasks. They are programmable by software. Arduino uses a series of microcontrollers called ATmega AVR, manufactured by the Atmel company. The best thing about Arduino is that we do not have to use assembly language to program these microcontrollers.

Basically an Arduino is a circuit board with many electronic parts around the actual microcontroller. On the edge of the board there are many pins that make it possible to connect different components such as switches, LED's, ultrasonic sensors, temperature sensors, displays, stepper, etc ...



Arduino with examples for sensors

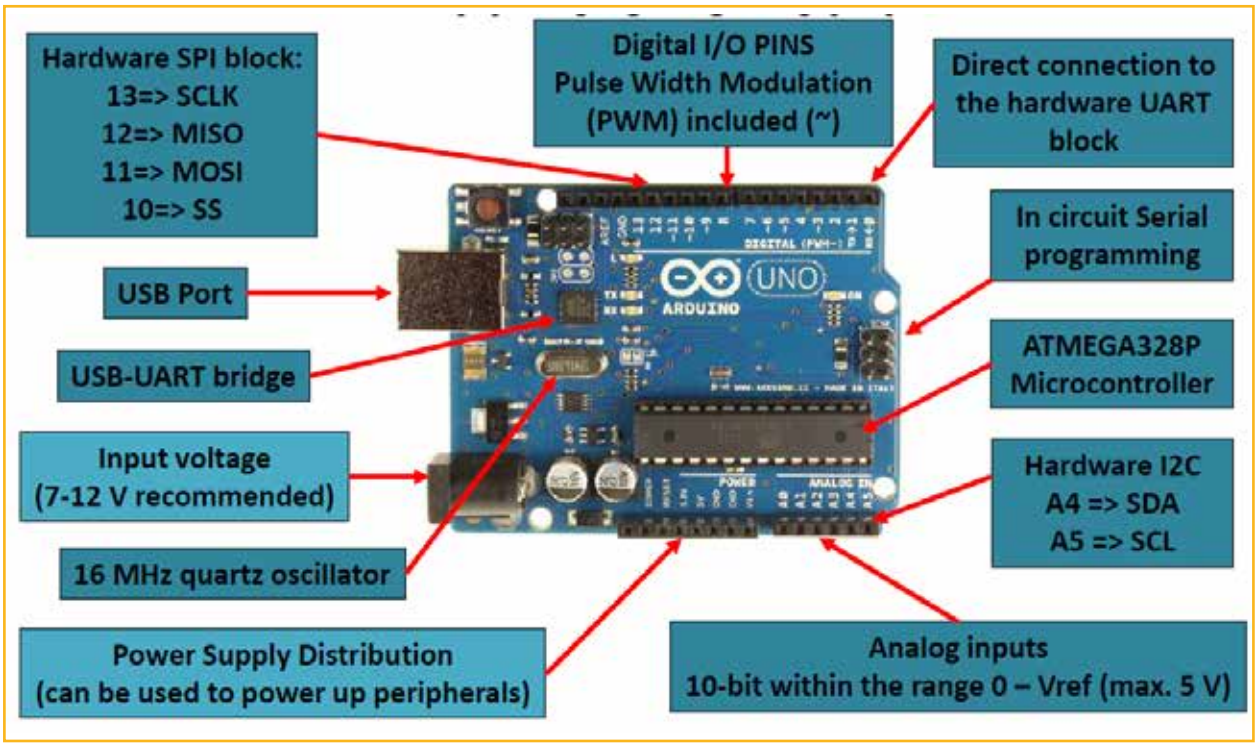
Arduino as an application control



Arduino with examples for actuators

3.1.3 Hardware

In the following picture, the positions of the individual components and connections on the board are marked.



Typical connections and components of an Arduino

Overview of the most important connections

USB Port

- Connection interface for the USB cable
- This interface is used to connect the card to the computer for programming

Input voltage / External power supply

- This connector is used to connect a power supply or battery packs

Power Supply Distribution

- Can be used to power up peripherals
- Often sensors require 3.3 V or 5 V power supply

Various bus connections

- Universal Asynchronous Receiver Transmitter (UART)
  A UART interface is used to send and receive data via a data line and forms the standard of serial interfaces on PCs and microcontrollers. The interface is also very common in the industrial sector with various interfaces, e.g. RS-232 or EIA-485
- Serial Peripheral Interface (SPI)
  Standard for a synchronous serial data bus with which digital circuits can be interconnected according to the master-slave principle
- Inter-Integrated Circuit (I2C)
  Mainly used internally in devices for communication between different circuit parts, e.g. between a controller and peripheral ICs

Digital I/O Pins

- Digital signals can be input and output here. Some connections also allow the output of pulse width modulated values

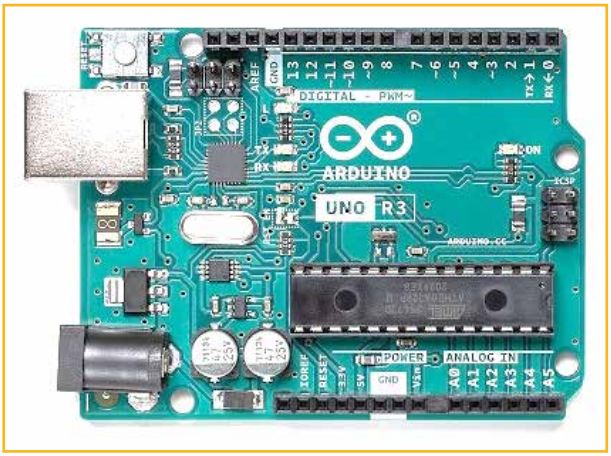
Analog Inputs

- Analog signals can be read in here and converted into 10-bit values.

Technical data of typical Arduino boards

The **Arduino UNO** is the best board to get started with electronics and coding. If this is your first experience tinkering with an automation platform, the UNO is the most robust board to start playing with. And the UNO is the most used and documented board of the whole Arduino family.

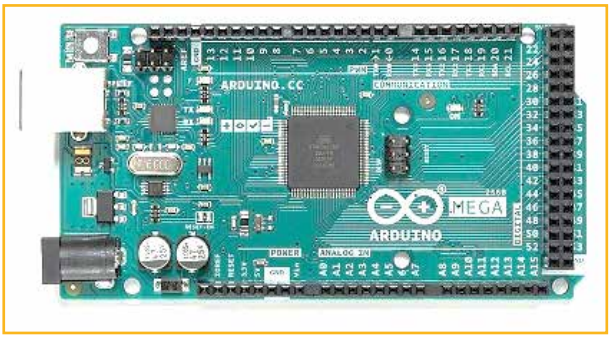
Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller, simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. You can tinker with your Uno without worrying too much about doing something wrong, worst case scenario you can replace the chip at little cost and start over again.



Arduino UNO R3

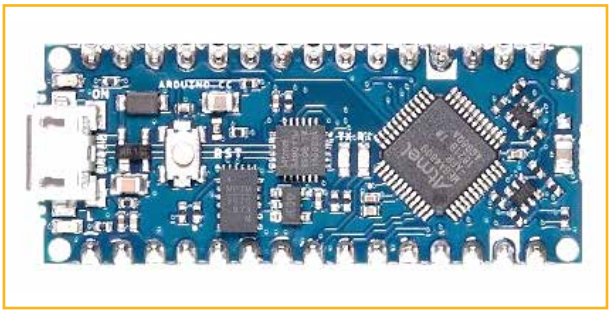
The **Arduino Mega 2560** is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno.

The Mega 2560 is an update to the Arduino Mega, which it replaces.



Arduino Mega 2560 Rev3

The **Arduino Nano Every** is Arduino's 5V compatible board in the smallest available form factor (45x18mm).



Arduino Nano Every

The Arduino Nano is the preferred board for many projects requiring a small and easy to use microcontroller board. The low price makes the Nano Every particularly suited for wearable inventions, low cost robotics, electronic musical instruments, and general use to control smaller parts of a larger projects.

The Arduino Nano Every is an evolution of the traditional Arduino Nano board, but features a lot more powerful processor, the ATMega4809. This will allow you to make larger programs than with the Arduino Uno (it has 50% more program memory), and with a lot more variables (the RAM is 200% bigger).

Technical data at a glance

	Arduino UNO	**Arduino Mega 2560**	**Arduino Nano Every**
Microcontroller	ATmega328P	ATmega2560	ATMega4809
Operating Voltage	5 V	5 V	5 V
Input Voltage (recommended)	7 - 12 V	7 - 12 V	7 - 12 V
Input Voltage (limit)	6 - 20 V	6 - 20 V	7 - 21 V
Digital I/O Pins	14 (of which 6 provide PWM output)	54 (of which 15 provide PWM output)	20 (of which 5 provide PWM output)
Analog Input Pins	6 (ADC 10 bit)	16 (ADC 10 bit)	6 (ADC 10 bit)
DC Current per I/O Pin	20 mA	20 mA	20 mA
DC Current for 3.3V Pin	50 mA	50 mA	50 mA
CPU Flash Memory	32 KB of which 0,5 KB used by bootloader	256 KB of which 8 KB used by bootloader	48 KB
SRAM	2 KB	8 KB	6 KB
EEPROM	1 KB	4 KB	256 Byte
Clock Speed	16 MHz	16 MHz	20 MHz

For all data please also have a look at the individual data sheets.

In addition to these motherboards, there are countless special boards for data transmission via cable and Wifi, for connecting to the IoT, and so on.

Also browse the full range of official Arduino products, including boards, modules (a smaller form factor of classic boards), shields (items that can be plugged onto a board to give it additional functionality), and kits.

www.arduino.cc/en/Main/Products

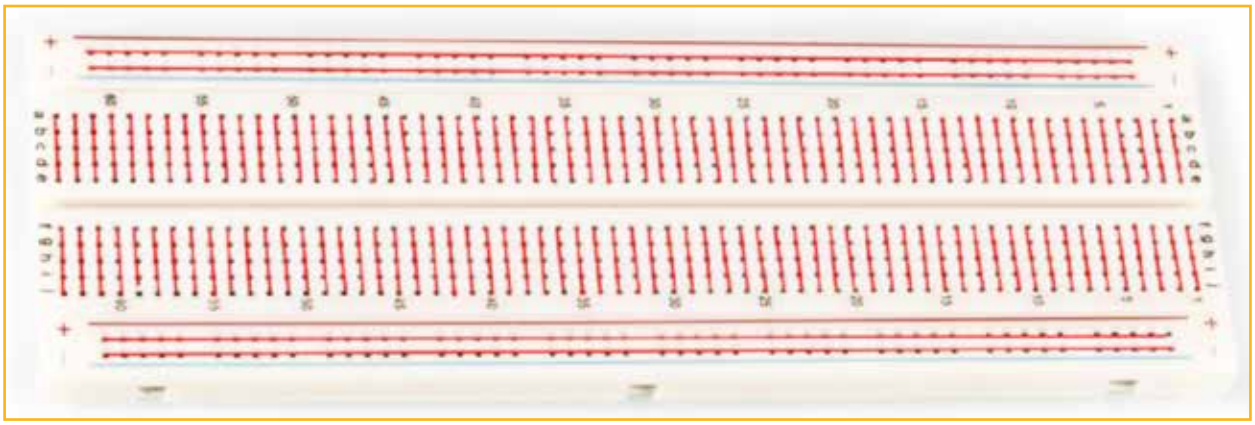
3.1.4 Description of typical equipment

In addition to sensors and actuators, resistors, LEDs, cables and other useful electronic components, you will need a breadboard for a quick and flexible start. Also, some basic tools like pliers, side cutters, cable knives etc. are useful for creating experimental setups.

Breadboard
Breadboards are indispensable tools when learning electronics or prototyping new designs and are thus frequently used in the Arduino environment.

The breadboard provides several sockets where the components can simply be plugged into. This allows electrical circuits to be built and realized. Therefore, it is possible to connect many components and cables with each other without soldering or screwing them together. Some sockets are horizontally conductively connected with each other, others are vertically conductively connected with each other.

This image below shows in colour which contacts are connected.



Breadboard with conductive connections drawn in

Separate circuit boards, which are available in various designs and sizes, are suitable for implementing permanent and larger projects. These can be individually designed and then simply connected to the microcontroller board. Working with the boards requires suitable tools (e.g. soldering station with soldering material) and accurate and thorough soldering.

Jumper cable

When using the plug-in board, connecting cables (Jumper cables) are suitable for plugging conductive connections.

Jumper cable

Various useful electronic tools

Precision mechanics round nose pliers / Electronics side cutter / Knife

Additional different electrical and electronic components will be discussed further in the following chapters.

Additional different electrical and electronic components will be discussed further in the following chapters.

3.1.5 Summary

Arduino is an open-source electronic prototyping base for easy-to-use hardware and software in the field of microcontrolling. It is suitable for realising fascinating projects in a short time. Many of them can be found on the Internet under the keyword „Arduino“. It is mostly used by artists, designers, or tinkers to realize creative ideas.

But Arduino is also increasingly used by universities and schools to enable an interesting and simple entrance into the world of micro controlling and automated systems.

Notes

Chapter 1

Introduction programming

1.1 PROGRAMMING BASICS

1.1.1 Processing

The “Input – Process – Output” (IPO) model, or input-process-output pattern, is a widely used approach in systems analysis and software engineering for describing the structure of an information processing program or of another process. Many introductory programming and systems analysis texts introduce this as the most basic structure for describing a process.

A computer program is useful for another type of process that uses the input-process-output model: It receives input from a user or the process environment via sensors, performs calculations and decisions with the input, and returns the results. Technically, the system separates itself from the environment so that both inputs and outputs are defined as a unified mechanism.

The system divides the work into three categories:

Input	A requirement from the environment
Process	A computation based on the requirement
Output	A provision for the environment

This model can be used to describe both production processes and data technology processes.

1.1.2 Basics: Program design

A computer program is a collection of instructions that can be executed by a computer to perform a specific task.

A computer program is usually written by a computer programmer in a programming language. From the program in its human-readable form of source code, a compiler or assembler can derive a machine code—a form consisting of instructions that the computer can directly execute. Alternatively, a computer program can be executed with the aid of an interpreter.

→ **The microcontroller on the Arduino board also works according to the IPO principle.**

IPO model

IPO processing with a microcontroller:

Input

Input can be made via sensors or switches (e.g. pushbuttons). Depending on their design, sensors are able to measure various variables of the outside world.

Examples: Temperature, brightness, humidity, pressure, etc.

Processing

Processing takes place within the microcontroller in the so-called „controller“. It processes the input based on the programming in the form of the commands entered. The processing takes place in the microcontroller through the programmed sketch.

Output

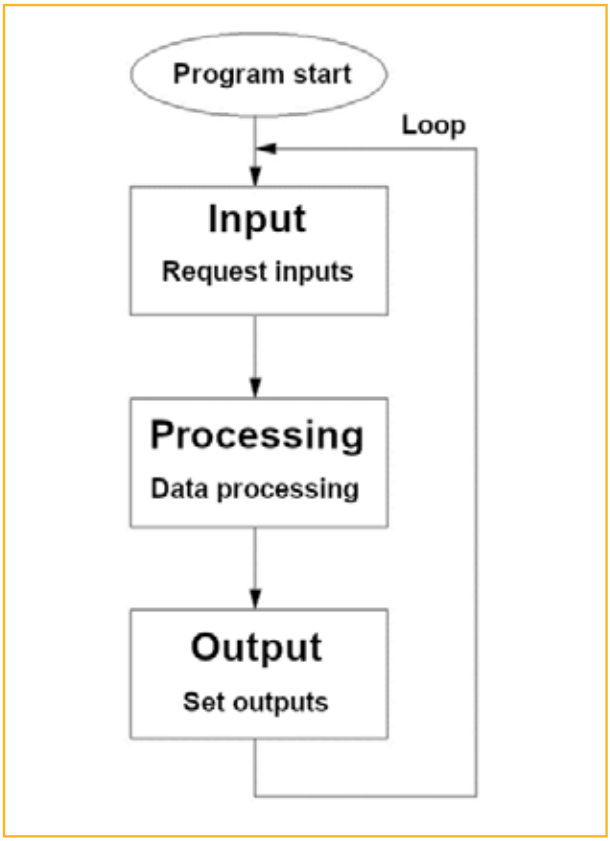
The processed input signals are output to the environment as a voltage signal via the output. This output can be used to actuate an actuator.

Examples: Light (e.g. LED), motor, buzzer, valve, etc.

Structure of a program

A program proceeds from instruction to instruction. This means that the program tells the controller what to do and in which order. The source code should be designed as simple and well-structured as possible to make the source code as comprehensible as possible.

In sequential programming, the same code, which consists of individual instructions, is always run through in a loop. The sequence corresponds to the IPO principle (Input - Processing - Output).



Simple structure of an IPO model

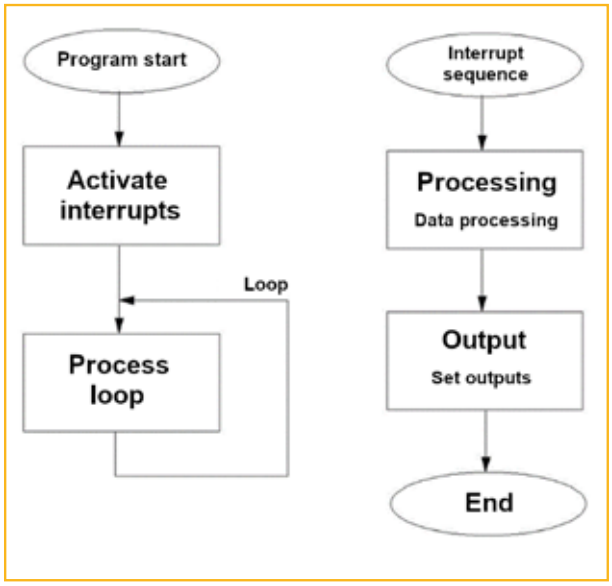
Interrupt controlled program flow

An interrupt is a temporary interruption of a running program in order to process another, usually short, but time-critical process.

Structure of a program

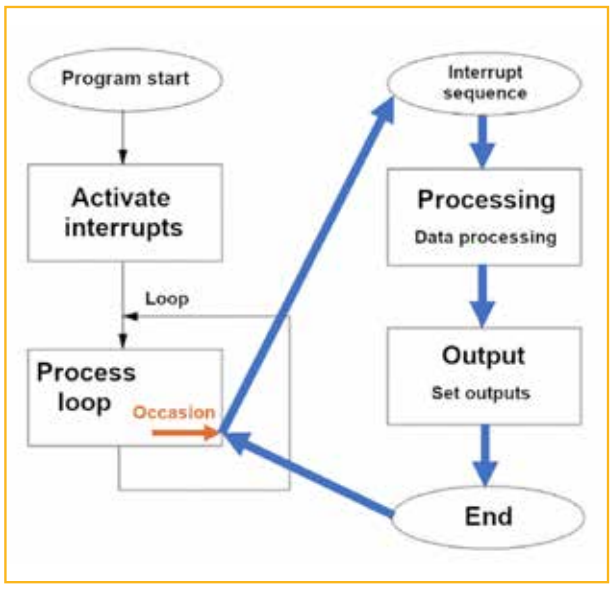
A program proceeds from instruction to instruction. This means that the program tells the controller what to do and in which order. The source code should be designed as simple and well-structured as possible to make the source code as comprehensible as possible.

In sequential programming, the same code, which consists of individual instructions, is always run through in a loop. The sequence corresponds to the IPO principle (Input - Processing - Output).



Interrupt controlled program flow

What happens when an interrupt is reported?



Sequence with an interrupt

The triggering event is called an Interrupt Request (IRQ). After this request, the processor executes an interrupt routine (Interrupt Service Routine - ISR). The interrupted program is then continued where it was interrupted.

Interrupts can be triggered hardware-wise by pushbuttons or sensors, or software-wise by a specific calculation result.

Polling versus interrupt

Besides interrupts, there is only the technique of programmed **cyclic polling** to get the status of input/output devices, processes or other.

Advantages of interrupt compared to polling:

The method of polling is simpler and does not require additional hardware, but it is much more inefficient than working with interrupts because it uses the processor very frequently.

In addition, the response speed in polling depends on how much time elapses between queries, this can be critical in situations that require an immediate response.

1.2 THE WORLD OF PROGRAMMING LANGUAGES

1.2.1 Short introduction: Programming languages

There are countless programming languages with very different tasks and areas of application.

A programming language is a formal language for formulating data structures and algorithms, i.e. calculation rules that can be executed by a computer. They are composed of instructions according to a given pattern, the so-called syntax.

```
1
2  // Source code example in C++
3
4  #include <cstdlib>
5  #include <iostream>
6
7  using namespace std;
8
9  int main(int argc, char *argv[])
10 {
11     int   age; // Variable Typ Integer
12
13     cout << "How old are you?"     ;
14     cin >>   age;
15     cout << "You are "   <<  age  <<  " years old"  << endl;
16     getc();
17     return 0;
18 }
19
```

Example of a source code

While the first programming languages were still directly oriented to the characteristics of the respective computers (machine language), one uses today mostly problem-oriented languages. These higher programming languages have a more abstract way of expression that is more easily understandable for humans.

Programs written in these languages can be automatically translated into machine language, which is executed by a processor. Increasingly, visual programming languages are also being used to make programming languages more accessible.

1.2.2 Programming microcontrollers

Microcontrollers were and still are often programmed in machine language or in “assembler”.

The assembler language is a programming language that is oriented towards the instruction set of a specific processor architecture. Assembler languages are therefore called machine-oriented programming languages and the successor of direct programming with numeric codes.

In contrast, in higher programming languages a so-called compiler translates more abstract (more complex, not limited to the processor instruction set) instructions into the machine code of the given target architecture or into an intermediate language.

Example of a short program:

The following program in the C programming language calculates the sum of the numbers a=2 and b=3 and returns the result c to the caller.

```c
int main() {
    int a = 2;
    int b = 3;
    int c = a + b;
    return c;
}
```

Short program written in C

Compiling this program may result in the following machine code in hexadecimal notation:

machine code (hexadecimal)	associated assembler code	associated C-code
55 48 89 E5	`push rbp` `mov rbp, rsp`	`int main() {`
C7 45 FC 02	`mov DWORD PTR [rbp-4], 2`	`int a = 2;`
C7 45 F8 03	`mov DWORD PTR [rbp-8], 3`	`int b = 3;`

8B 45 F8
8B 55 FC
01 D0
89 45 F4
	`mov eax, DWORD PTR [rbp-8]`
	`mov edx, DWORD PTR [rbp-4]`
	`add eax, edx`
	`mov DWORD PTR [rbp-12], eax`
		`int c = a + b;`

8B 45 F4	`mov eax, DWORD PTR [rbp-12]`	`return c;`
5D C3	`pop rbp` `ret`	`}`

Example of a source code

1.2.3 The programming language “C”

“C” is an easy to learn programming language and suitable for numerous microcontroller systems. A freely available C compiler exists for almost all microcontrollers. The system kernels of many operating systems are programmed in C.

Explanation of “C”

C is a universal programming language in which it is also possible to program close to the hardware, originally intended as a kind of machine-independent macro assembler. Besides assembler, C is the most commonly used programming language for microcontrollers, but is also frequently used outside this field, for example in application and system programming.

Arduino understands C

However, the C of Arduino is kept much simpler than that of a professional C compiler and thus saves the user a lot of work. Above all, one does not have to worry about the complicated hardware routines with Arduino, since they are already integrated as fixed commands in the development environment.

This will be described in more detail later.

The logo of C Programming Language

Explanation: Program library

In programming, a program library (library or lib for short) refers to a collection of subprograms/routines that offer solutions to recurring problems.

In contrast to programs, libraries are not independently executable units, but they contain auxiliary modules that are requested by programs.

These libraries contain, for example, functions for providing communication between connected hardware and the microcontroller.

Possible hardware applications are e.g.:

- Querying keyboards or a key matrix
- Operating interfaces (USB, Ethernet)
- Addressing sensors or actuators
- Use of various displays

1.3 IMPORTANCE OF A GOOD PROGRAMMING STYLE

1.3.1 Aim of a defined programming style

In programming, code conventions or coding standards are understood to mean the creation of source code according to certain predefined rules. It is regarded as a partial aspect of software quality, which is to support in particular the understandability and maintainability of software. These are recognized criteria for software quality ISO/IEC 25000.

The purpose of a defined programming style:

- facilitating the work of all team members involved in a programming project
- improving the readability and comprehensibility of the program source code
- simplification of software maintenance
- elimination of avoidable sources of errors in programs

What is meant by the term „programming style“?
A good programming style is understood as rules according to which the programmer designs the code.

The following aspects are to be taken up:

- clear source code formatting
- clear naming conventions
- modularization of the software concerning reusability and maintenance
- concise, but meaningful documentation of the code
- robustness by error handling

What is a good programming style?
Good programming style is always subjective. There is no fixed definition of what counts as good programming style. Within a team, certain conventions should be agreed upon. These should then also be used consistently.

Various elements to be defined:

- indentation style (space or tab)
- maximum line length (e. g. 80 characters per line)
- use of common naming conventions
- consistent use of „left-hand“ or „right-hand“ comparisons
- clear guidelines for documentation

1.3.2 Useful agreements for a good programming style

Indentation style

There are countless indentation styles, which are rarely restricted by the given syntax. Everyone has a preferred style, nevertheless you should agree on a style in the team.

Space vs. tabulator
Both methods have their advantages:

Blank space:	Constant representation independent of display options.
Tab:	Each programmer can choose the indentation depth.

The following applies: Variants should not be mixed!

Comments

Comments should be used extensively and are essential for team projects.
But there are „guidelines“ for comments as well:

- The code should not be explained or repeated.
- The intention should be clear.
- Comments can be used as placeholders.
- Practical for debugging („commenting out“).

Naming conventions

In most cases the „Hungarian notation“ is used. Hungarian notation is a naming convention used by programmers to choose identifiers for variables and constants, functions and methods, and other objects.

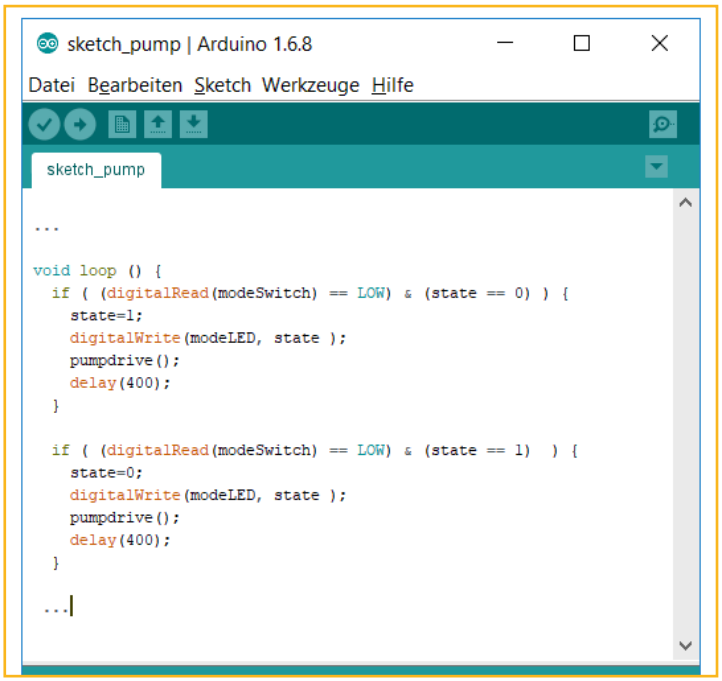
Typical naming conventions for “C” are:

- Constants are declared in uppercase.
- Functions, type definitions, and variables: Lowercase
- Identifiers with leading underscore should not be used
- Avoid names that differ in only one letter.
- Avoid names that look similar (lowercase L, uppercase I, 1).

Programming example without comments

In the present case, you see examples of conventions of naming, bracketing, and structuring indentation.

Also the principle of „right-hand-comparisons“ was used. (cf. „state=1“)



Example programming style

Conclusion programming style

There are different styles.

- Every programmer has his own conception of good programming style.
- It is important to use this style consistently for oneself.
- A consistent style should be agreed upon and be used within the team.
- Programming styles do not impose rules, but rather provide guidelines.
- However, these guidelines can be circumvented in individual cases, if well justified.

Normally, the programmer receives support from the editor.

1.4 STEPS OF PROGRAM DEVELOPMENT

1.4.1 Structured approach to solving a programming task

No matter with which programming language you want to develop a program for the computer, the following steps are useful, if not necessary, in any case:

1. Problem analysis
The first thing to do is to clarify the problem. You cannot solve a problem that you do not understand. There is a difference between the problem and the problem you think you are solving. It is easy to read the first few lines of a problem and assume the rest because it is similar to something you have seen before.

Very often it is advantageous to explain the problem to others to see if they understand the explanation as you have understood it yourself. You don‘t want to find out halfway through the solution that you have misunderstood the problem.

It pays to take more time at the beginning. The better you understand the problem, the easier it will be to solve.

2. Data analysis
It must be determined which data are to be output and which data are to be entered for this. It must be clarified which data types the program is to use.

What are the inputs?
What kinds of inputs will go into this problem?
What are the outputs?
What will this function return?

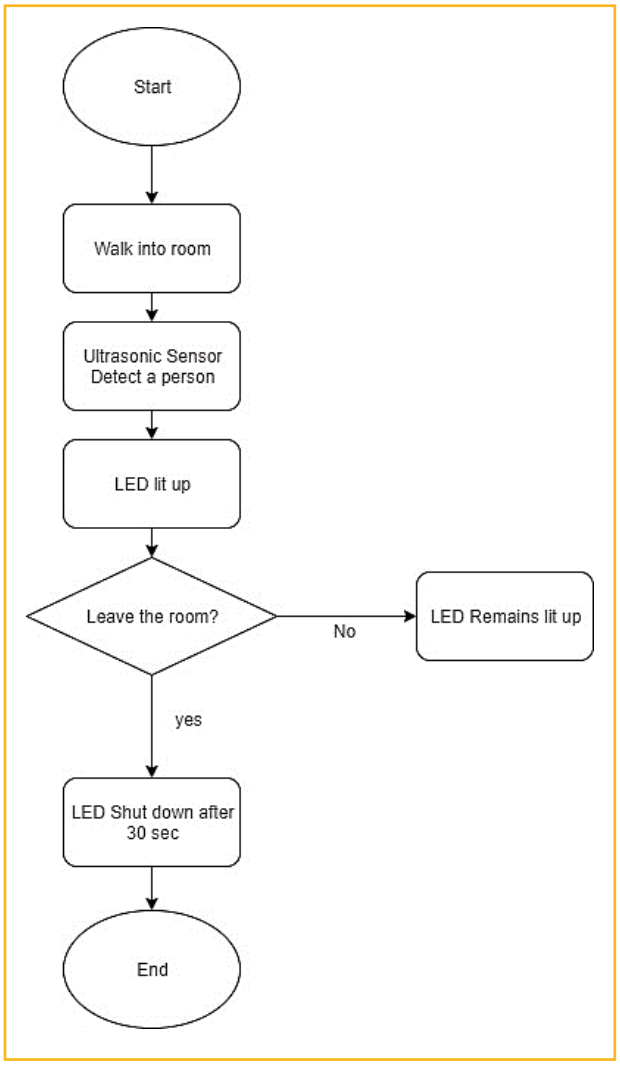
3. Work through the problem manually
Next, design a plan for how you will solve the problem. Take a piece of paper and work through the problem manually. Look for patterns and see if there is anything you can generalize. See if you can reduce any steps or if you are repeating any steps.

4. Write pseudocode

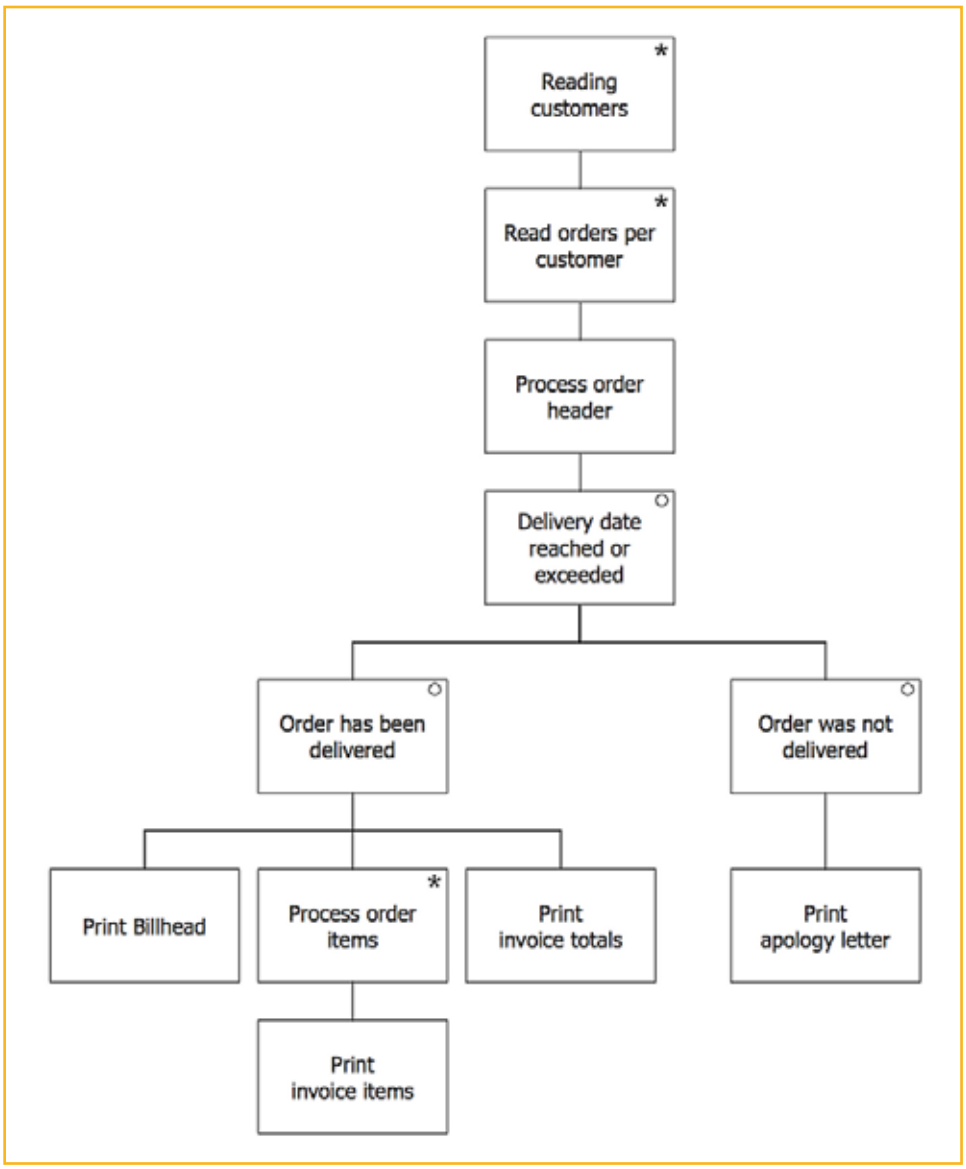
Even after you've worked out the general steps, writing out pseudocode that you can translate into code will help defining the structure of your code and make coding a lot easier. Write pseudocode line by line. You can do this either on paper or as comments in your code editor. If you're starting and find blank screens daunting or distracting, I recommend doing it on paper.

Pseudocode is a description of the steps of an algorithm in simple language. In other words: Your pseudocode is your step-by-step plan for how to solve the problem. Write down the steps you need to take to solve the problem.

The problem must be broken down into tiny, logically sequential steps. It must be clear beforehand what is to be done, under which condition, and when, in order to achieve the goal. It is recommended to make the algorithm visually descriptive. For this purpose, you can use pseudocode, a program flow chart or a structure diagram.



Example of a flowchart



5. Translate pseudocode into code

When you have your pseudocode ready, translate each line into real code in the language you are working on. If you wrote it out on paper, type this up as comments in your code editor. Then replace each line in your pseudocode.

Programs are a precisely defined collection of operations to be performed by the computer, written in a specific syntax. Normally, programs are written in a programming language consisting of a collection of commands and instructions understandable to humans. Only later is the program converted into a form that can be processed on the computer.

In order to convert the program code into the machine language which can be understood by the computer alone, there are two possibilities:

Either it is **interpreted** (translated) line by line by software during the execution of the steps or it is **compiled** completely by the programming software.

The machine code produced during compilation is combined into either one or more files to form a final machine-executable program to be transferred to the processor‘s program memory.

6. Simplify and optimize your code

You’ve probably noticed by now that simplifying and optimizing are recurring themes. To simplify and optimize your code, you may need to work through the structure of your solution set a few times to find ways to further simplify and optimize the code. The goals will depend on your team’s style or your personal preference.

Here are some questions to keep in mind:

- What are your goals for simplifying and optimizing?
- How else can you make the code more readable?
- Are there any more extra steps you can take out?
- Are there any variables or functions you ended up not even needing or using?
- Are you repeating some steps a lot? See if you can define them in another function.
- Are there better ways to handle edge cases?

7. Program test and debug

The program is tested from two directions. First, the syntax must be checked so that the machine code can be executed at all. This check is usually performed by the compiler or the interpreter. In the second test phase the logic is checked. Logical errors can have dangerous consequences. For the logic test, the program must be tested under verifiable conditions with previously defined data. As many program sequences as possible must be recorded.

This step should be carried out throughout the process. Debugging throughout will help you catch any syntax errors or gaps in logic sooner rather than later. Take advantage of your integrated development environment (IDE) and debugger. When you encounter errors, trace the code line by line to see if anything didn‘t go as expected.

Here are some techniques to use:

- Check the console to see what the error message says. Sometimes it gives a line number to check. This very often gives clues as to where to start, although sometimes the problem is not in that line at all.
- Mark parts or lines of code as comments so that they are not translated when translating to machine code. With this method it is possible to see how the code behaves towards these changes.
- Modify the sample data if there are scenarios that were not thought of and check if the code still works.

8. Program documentation – Write useful comments

You may not always remember what every single line meant a month later. And someone else working on your code may not know either. That's why it's important to write useful comments to avoid problems and save time later on if you need to come back to it.

Try to write short high-level comments that help you understand what's going on when it's not obvious. This is very useful when working on more complex problems. It helps to understand what a particular function is doing and why. By using unique variable names, function names, and comments, you and others should be able to understand:

What is this code for?
and
What is it doing?

Chapter 2

Introduction: Programming Arduino

2.1 SOFTWARE FOR PROGRAMMING

2.1.1 What is “Arduino IDE”?

Before working with the Arduino, it must be connected to the PC and programmed. The Arduino is only as “smart“ as the person operating it. It cannot think for itself, instead it processes commands in sequence. A sequence of such commands is called a program or sketch.

The software that is used to program the microcontroller called Arduino IDE, is open-source-software and can be downloaded for free on www.arduino.cc. With this “Arduino software” you can write little programmes which the microcontroller should perform. In the end the sketches are transferred to the microcontroller by USB cable.

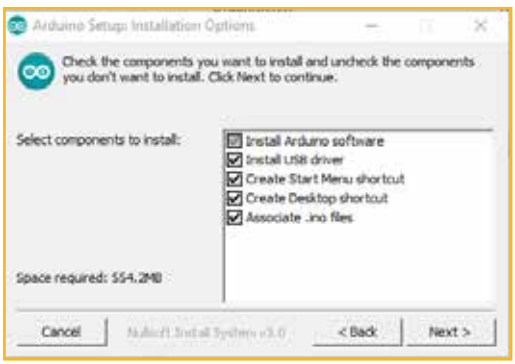
Download Arduino IDE for free on: www.arduino.cc

2.1.2 Installation and setup of the Arduino software

Download the latest version of Arduino software (as explained above) and install it on the computer.

The microcontroller is NOT connected to the PC during the installation!

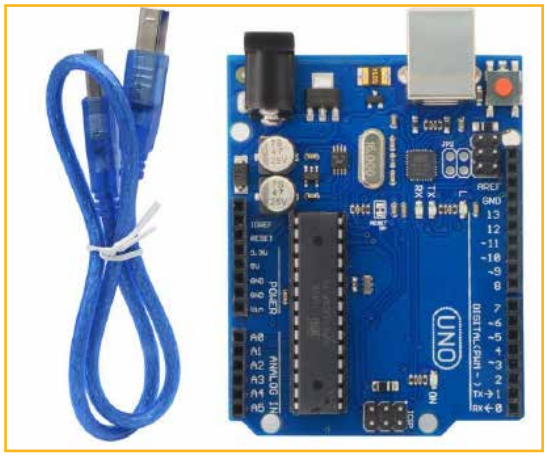
After that you open the software file and start the programme named arduino.exe. Allow the driver installation process.



The process will extract and install all the required files to execute the Arduino Software (IDE) properly.

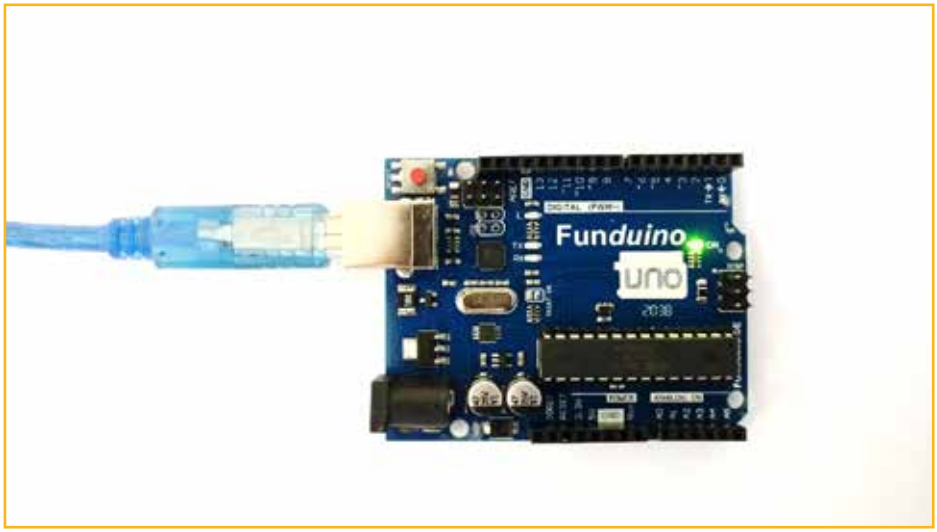
2.1.3 Connecting the Arduino

In this tutorial an Uno R3 is used, but the software also allows the programming of other Arduino boards. You also need a standard USB cable (A plug to B plug): The kind you would connect to a USB printer, for example.



USB Cable and Arduino Uno R3

The USB connection to the PC is necessary to program the board and not only to supply it with power. The Uno and the Mega automatically draw power either from the USB or from an external power supply. Connect the board to your computer using the USB cable. The green power LED (labeled ON) should light up.



Connected Arduino

2.1.4 Subsequent installation of the USB driver

If for some reason the USB driver was not installed during the installation of the software, then you will receive an error message when connecting or attempting to transfer a program to the Arduinos.

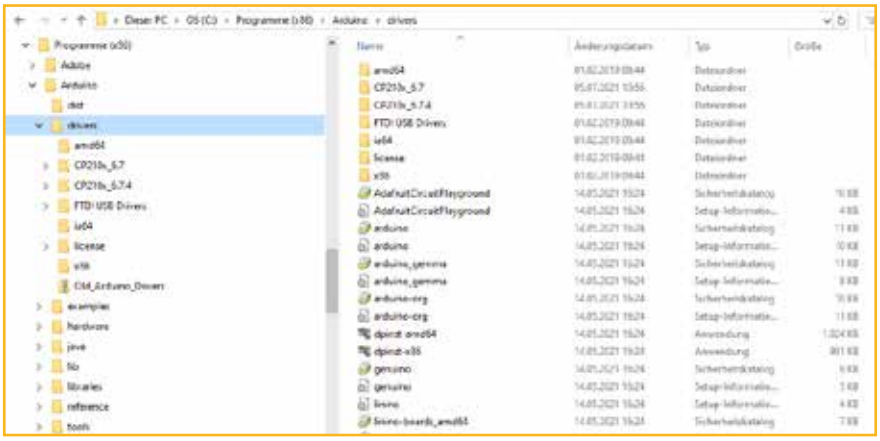
You can control this in advance:

At the control panel of the Computer you can find the “Device manager”. If the board has been installed successfully, it should appear here.

Computer Settings => Bluetooth and other devices

Here you can also see to which interface (COM) the Arduino is connected, in our case COM4.

When the installation has failed, there is either nothing special to find or you will find an unknown USB device with a yellow exclamation mark. In this case: Click on the unknown device and choose “update USB driver”. Now you can start over with the manual installation. You might choose the driver by your own to install it. It can be found in the Arduino file under “Drivers”.

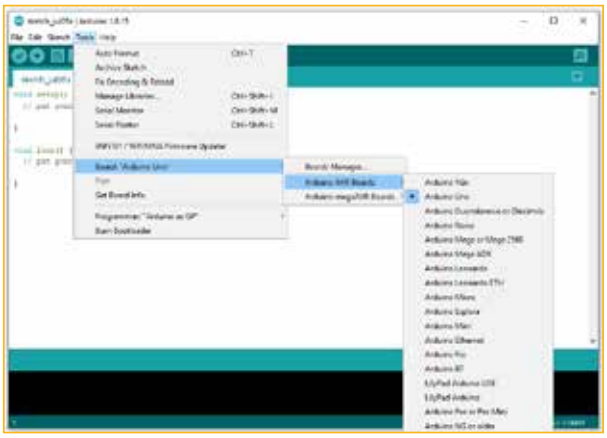


Place where the drivers can be found

2.1.5 Important programme settings

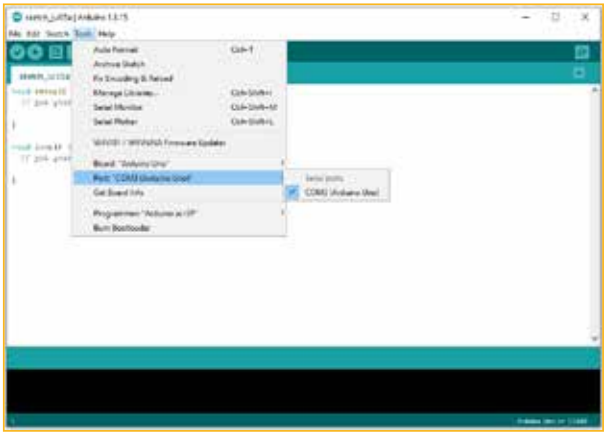
Before the first programme can be transferred to the Arduino, two settings must be made on the software.

1. You'll need to select the entry in the **Tools => Board menu** that corresponds to your Arduino board.



Selection of the right board

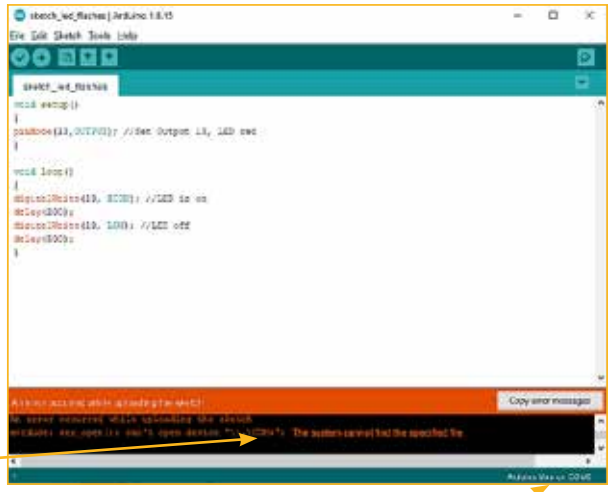
2. Selection of the correct serial port (USB port)
In the menu Tools => Port => Serial ports select the port for the serial device of the card. Very often this is COM3 or higher (COM1 and COM2 are usually reserved for serial hardware ports). To find out, you can unplug your board and open the menu again, the entry that disappears should be the Arduino board. Reconnect the board and select the correct serial port. Or you look up in the system settings of Windows.



Selection of the right port

If incorrect settings have been made here, IDE will issue an error message after the upload attempt.

Example of an error message when the COM interface is set incorrectly.



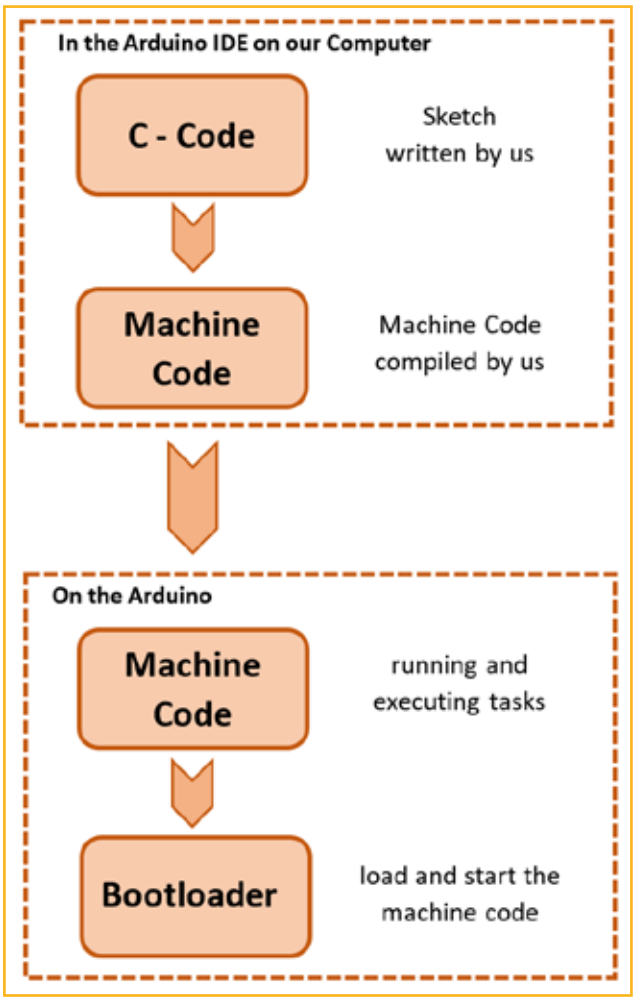
COM 4 is set

Arduino is connected to COM 5

2.2 USING ARDUINO IDE 2

2.2.1 How Arduino IDE 2 works

To understand how an Arduino board and programming work, we need to understand the global software architecture and the rough flow of software development.



Workflow between IDE and Arduino-Board

The central unit on the Arduino board is the processor. This processor is the place that contains the entire program that we design and that works the way we want it to.

In a new Arduino board, the processor is pre-baked. That is, the board has already been prepared by thoughtful people and programmed in a first step to make our lives easier. The program that is already included in the chipset is called **Bootloader**. Basically, it takes care of the very first moment of a processor‘s awakening when it is supplied with power.

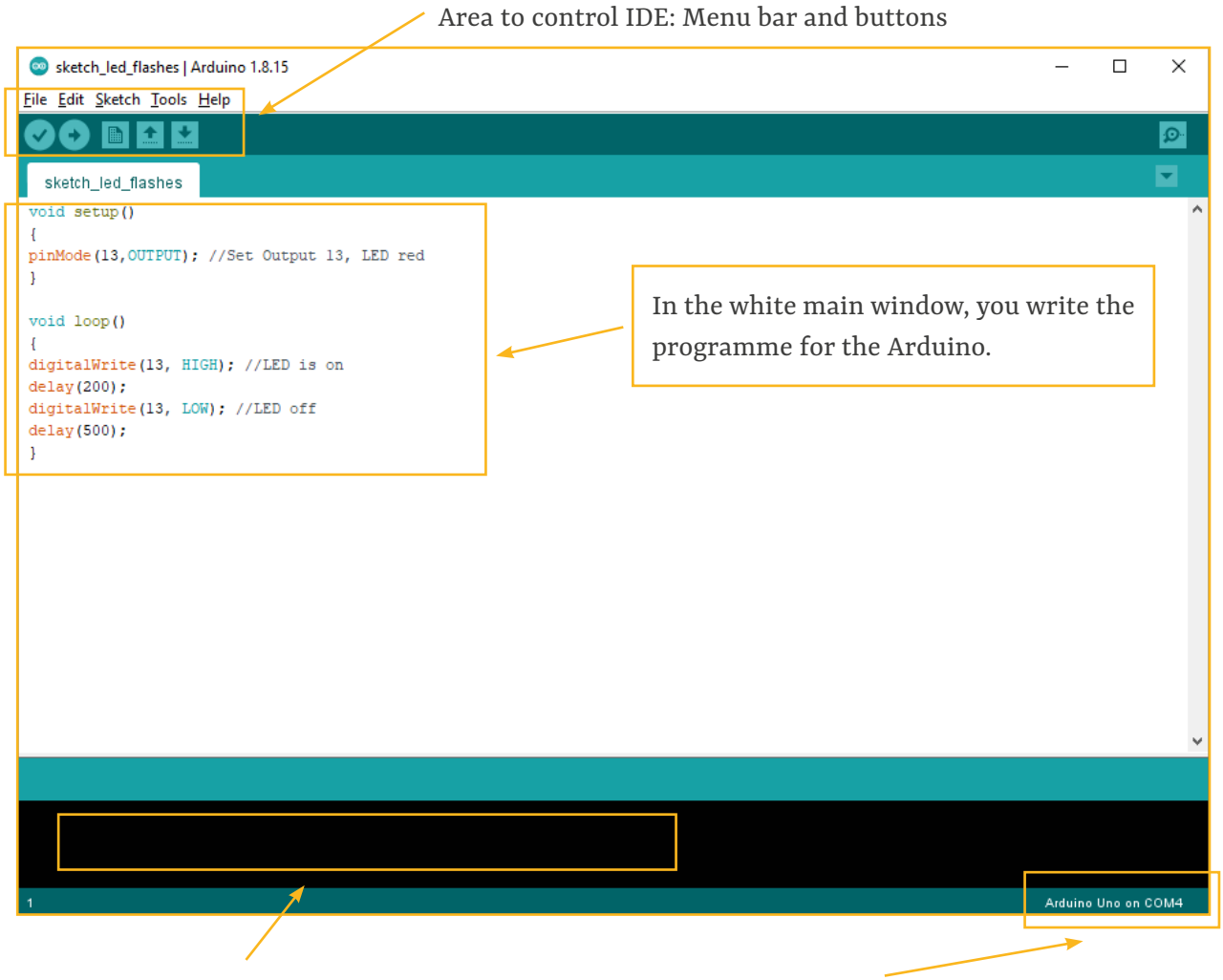
Bootloader

The bootloader is the software that makes the hardware work. It is necessary to place the user's software in the correct memory areas. It plays the main role in loading and starting our compiled program.

During normal use of the board we cannot overwrite the bootloader, but in principle it is possible to change or exchange the bootloader.

2.2.2 Important messages, buttons and functions of the IDE

The programming interface is divided into different areas:

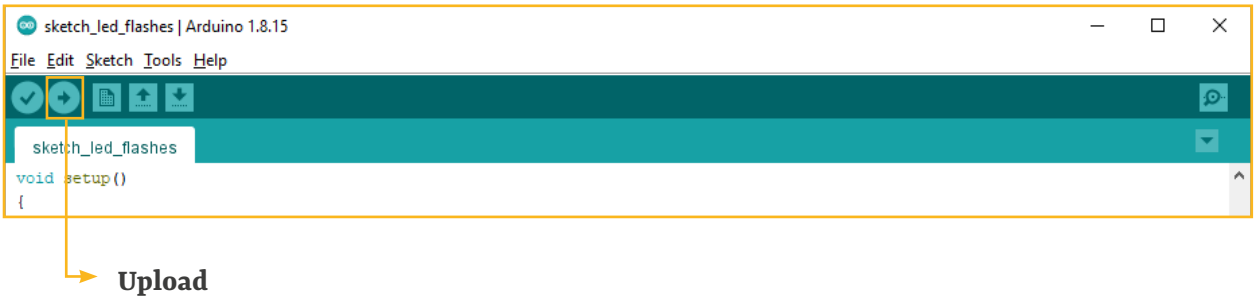


In the top line of the Arduino IDE (Integrated Development Environment) there are six important buttons.

Command "Verify"

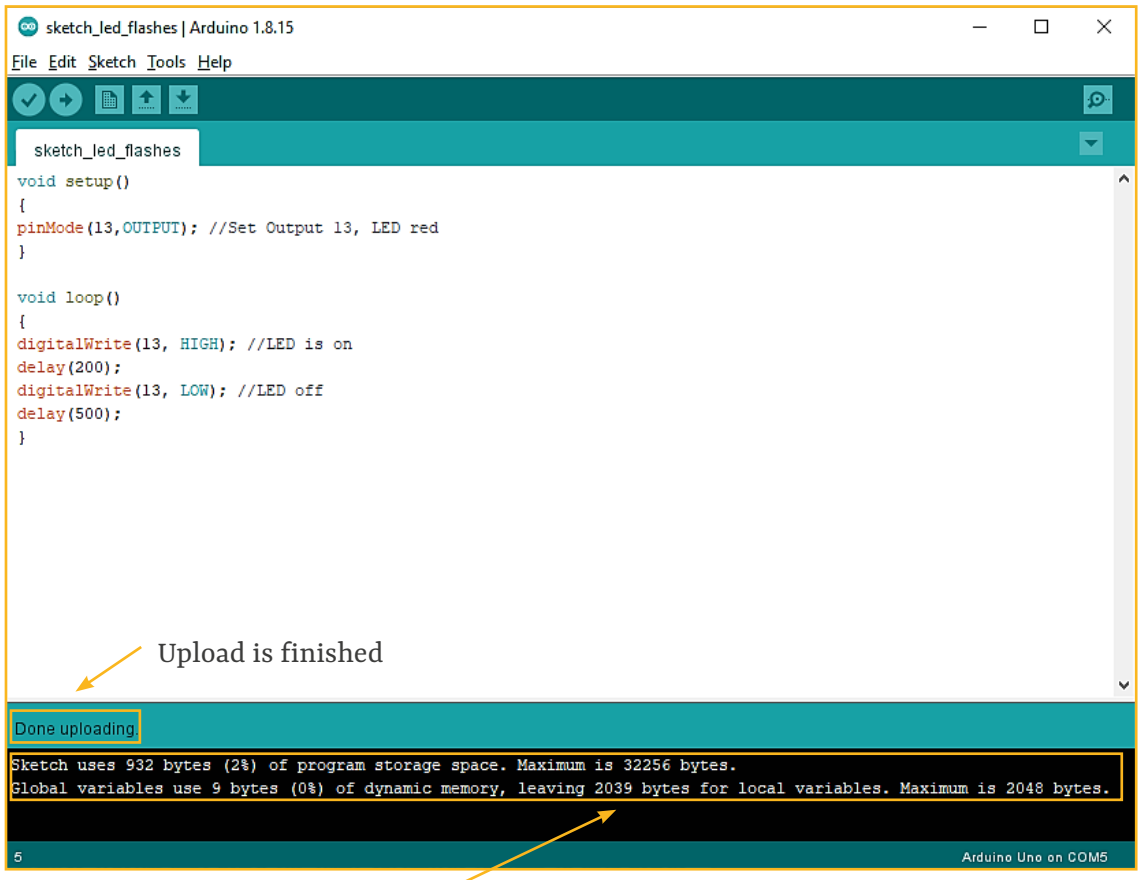
Verify
You need this button to compile the sketch. The IDE software checks the program for syntax errors and translates it into machine code.

Command "Upload"



Upload
This is the command to upload the sketch to the microcontroller's memory. When you use this command, the IDE first translates the program written in "C" into machine code and then, if there is no error in the syntax, immediately transfers it to the board. Use this button if you are sure that the program is error free.

If the upload went correctly, IDE returns a status message.



This program uses 2% of the total program memory available on the Arduino Uno (maximum 32256 bytes) and 9 bytes of dynamic memory (maximum 2048 bytes).

Command "New"

New
Opens a new sketch. The default name is the current date with the addition „a", „b", „c" and so on.

Command "Open"

Open
Command to open a saved sketch.

Command "Save"

Save
Saves the sketch under the current name.

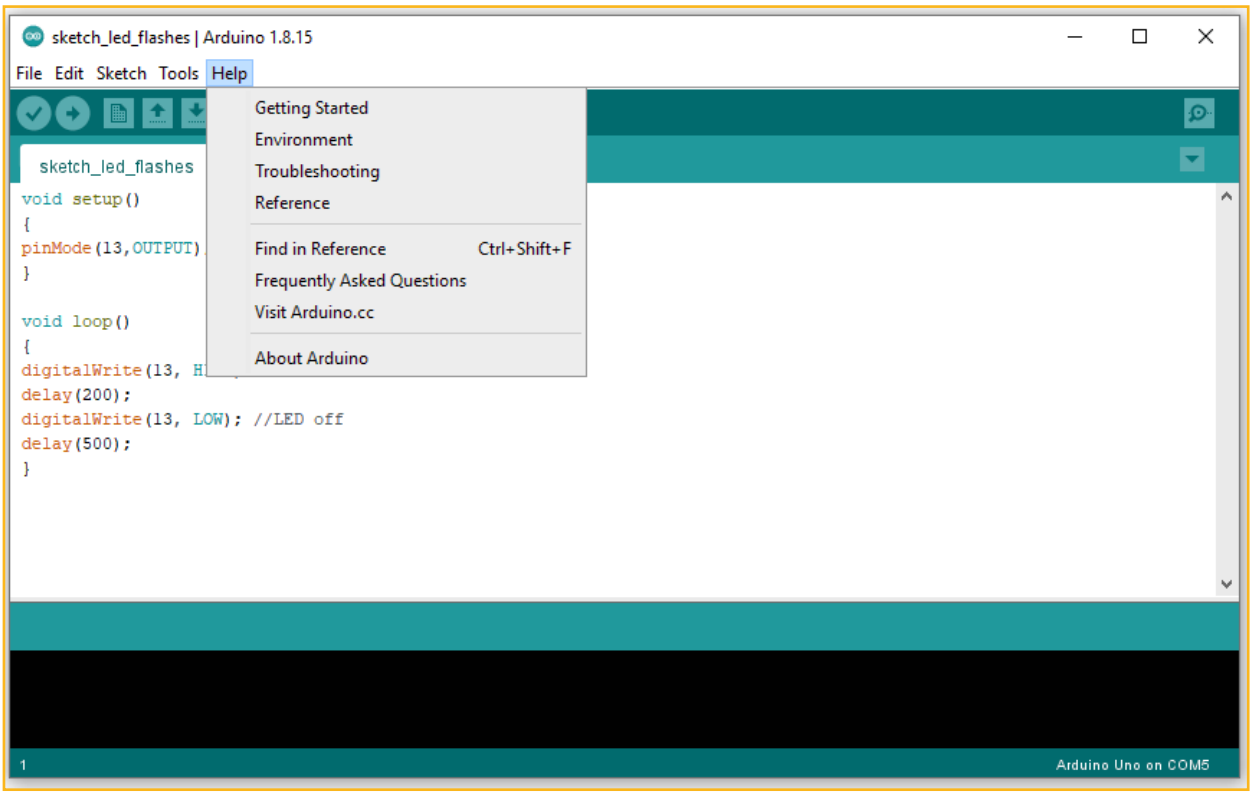
Command "Serial Monitor"

Serial Monitor
This button opens a new screen where the Arduino can send information back to the computer.
The Serial Monitor is an essential tool when creating projects with Arduino. It can be used as a debugging tool, testing out concepts or to communicate directly with the Arduino board.
More about this later.

All actions can also be executed via the menu or via corresponding short cuts. Many other useful commands can be found in the menus. Most of them should be known from the usual editors. All important and useful sub-items will be discussed in the following explanations if necessary.

Menu "Help"

Very comprehensive information can also be obtained via the „Help" menu.



Most of the sub-items in this menu refer to corresponding information on the website www.arduino.cc. For example, under „Getting Started" you can get a detailed explanation of how to get the IDE software running and how the first steps to transfer a sketch to the Arduino board work. In addition, you will learn other important details about working with the Arduino.

2.2.3 Adding libraries

What is a library?

In programming, a program library is a collection of subroutines/routines that offer solutions to problems that are thematically related. Libraries are, in contrast to programmes, no independently executable units, but they contain auxiliary modules, which are requested by programmes.

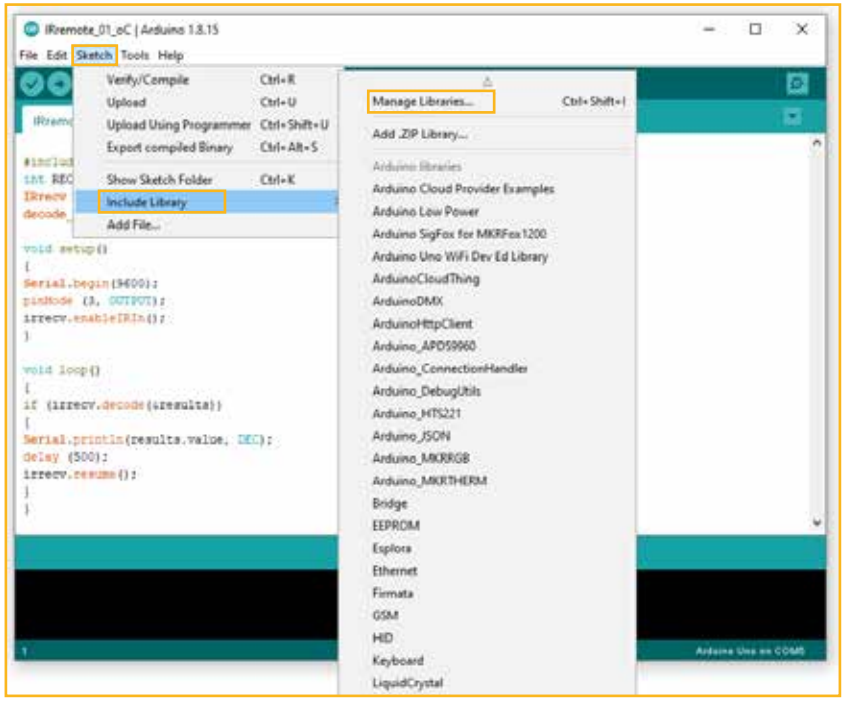
Libraries comprise a collection of usually precompiled, reusable routines that a programmer can call when writing a code so that he does not have to write this routine again. Often, they can be assigned to or are required for concrete interfaces or concrete hardware. The use of program libraries facilitates fast integration of different hardware components and simplifies programming.

For example, the built-in LiquidCrystal library makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download. The built-in libraries and some of these additional libraries are listed in the reference. To use the additional libraries, you will need to install them.

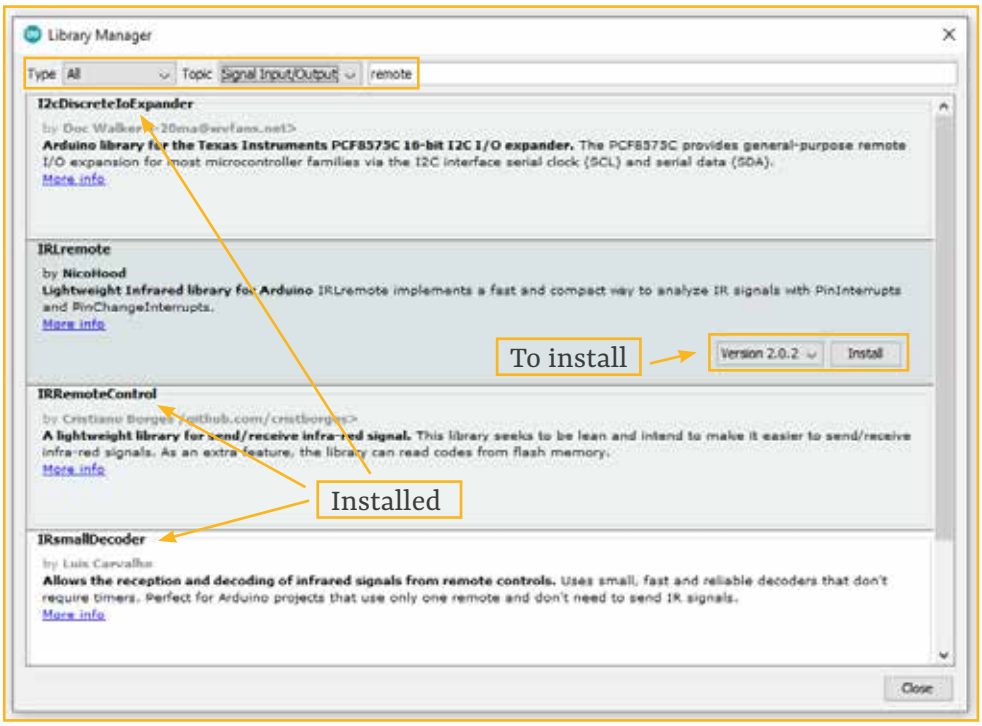
How to integrate libraries into the IDE

First option: Using the Library Manager

To install a new library into your Arduino IDE you can use the Library Manager. Open the IDE and click to the „Sketch“ menu and then “Include Library > Manage Libraries”.



Then the library manager opens, and you will find a list of libraries that are already installed or ready for installation. In this example we want to install a library for an IR remote control. To narrow down the scope, we set the „Topic“ to „Signal Input/Output“ and enter the term „remote“ into the search field. A list of available libraries will appear. Scroll through the list to find the right one, click on it and select the version of the library you want to install. Sometimes only one version of the library is available.

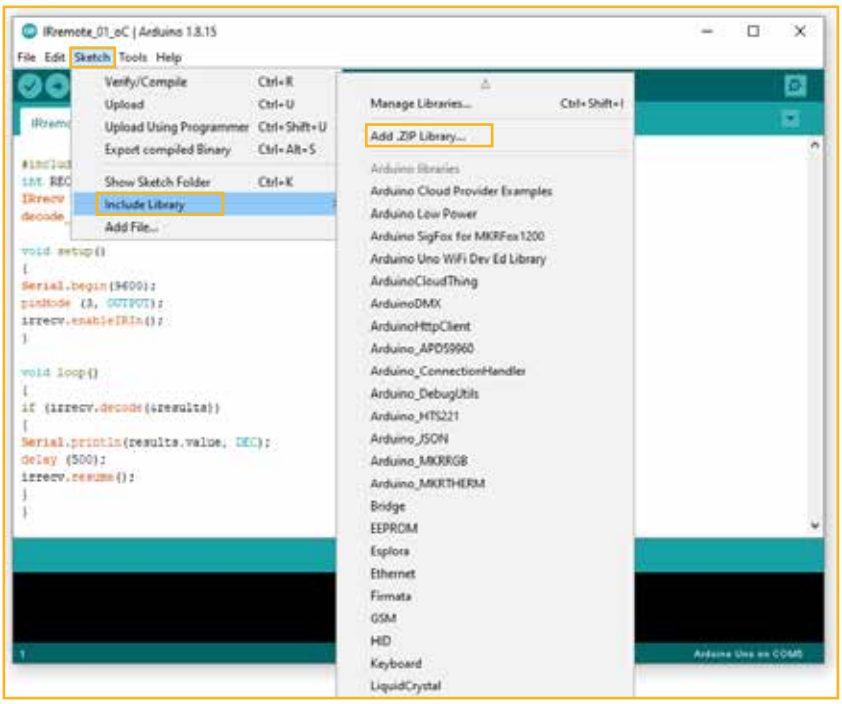


Finally click on install and wait for the IDE to install the new library.

Second option: Importing a .zip Library

Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. This folder contains a **.cpp file**, an **.h file**, and often a **keywords.txt file**, a folder with examples, and other files required for the library. Do not unpack the downloaded library but leave it as it is.

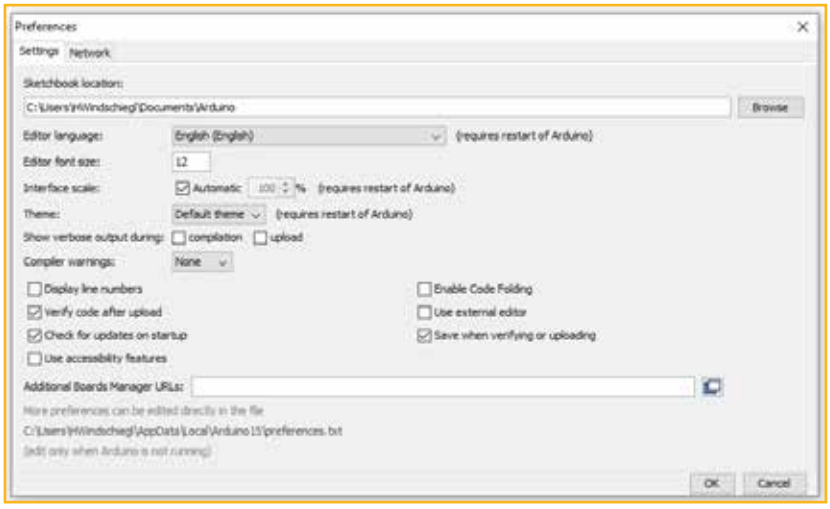
In the Arduino IDE, navigate to **“Sketch > Include Library > Add .ZIP Library”**. At the top of the drop-down list, select the option to **„Add .ZIP Library“**.



You will be prompted to select the library you would like to add. Navigate to the .zip file's location and open it.

Third option: Manual installation

When you want to add a library manually, you need to download it as a ZIP file, expand it and put in the proper directory. The ZIP file contains all you need, including usage examples if the author has provided them. The library manager is designed to install this ZIP file automatically as explained in the former chapter, but there are cases where you may want to perform the installation process manually and put the library in the libraries folder of your sketchbook by yourself.
You can find or change the location of your sketchbook folder at **"File > Preferences > Sketchbook location"**.



To consider

Arduino libraries are managed in three different places: inside the IDE installation folder, inside the core folder and in the libraries folder inside your sketchbook. The way libraries are chosen during compilation is designed to allow the update of libraries present in the distribution. This means that placing a library in the "libraries" folder in your sketchbook overrides the other libraries versions.

The same happens for the libraries present in additional cores installations. It is also important to note that the version of the library you put in your sketchbook may be lower than the one in the distribution or core folders, nevertheless it will be the one used during compilation. When you select a specific core for your board, the libraries present in the core's folder are used instead of the same libraries present in the IDE distribution folder.

Last, but not least important is the way the Arduino Software (IDE) upgrades itself: all the files in Programs/Arduino (or the folder where you installed the IDE) are deleted and a new folder is created with fresh content.

This is why we recommend that you only install libraries to the sketchbook folder so they are not deleted during the Arduino IDE update process.

More about the use of libraries in programming later.

2.3 PROGRAMMING WITH ARDUINO IDE

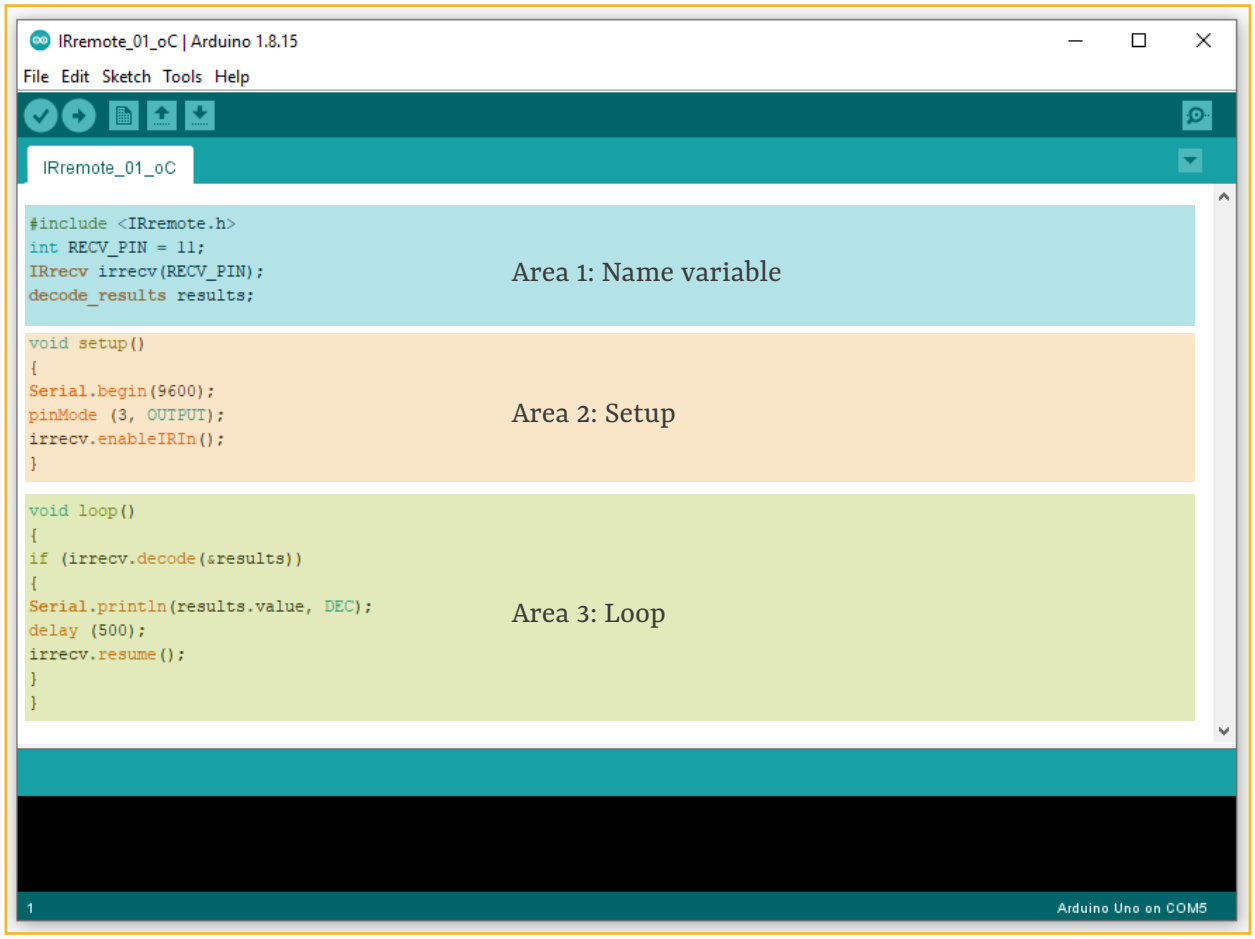
The Arduino is programmed in a syntax based on the programming language „C“.

C is a relatively „simple“ language. This characterization is not pejorative; it simply means that C works with the same kind of structures as most computers. It is often defined as a general-purpose programming language and is in fact one of the most widely used languages of all time. So you will learn a language that will open your access to global and universal programming concepts.

A very comprehensive work about programming the Arduino in C can be found on the website www.itbook.store. Search for the author Julien Bayle or for the title C Programming for Arduino. The book is a Free eBook and can be downloaded for free.

2.3.1 Structure of a sketch

Each program written for the Arduino, called a sketch, consists of three parts.



Area 1: Name variable

In the first area elements of the program are named. For example, variables are set there, or program libraries are loaded.

This part is not necessary for every sketch.

Area 2: Setup

This part of the program is run through once by the microcontroller and is required for each sketch, even if no entries are made in this area.

In the setup, for example, it is defined which pin on the microcontroller board is an output or an input. Defined as output, a voltage can be given out at the respective pin, for example to let an LED light up or to control an actor. Defined as input, voltages (e.g. voltage values of a sensor) can be read in at the pin.

Serial communication can also be started here.

Area 3: Loop

This loop part is the main program. It is repeated continuously by the board and can therefore also be regarded as the central part of the sketch. The microcontroller processes the sketch once completely to the end and then starts again at the beginning of the loop part.

This segment runs in an endless loop and can only be stopped by a reset or an interruption of the power supply.

2.3.2 Short introduction to important syntactical rules

A computer program is a sequence of statements consisting of declarations and instructions, satisfying the rules of a particular programming language, to perform or solve certain functions or tasks or problems with the aid of a computer.

When writing the instructions, certain rules must be followed so that the computer or the microcontroller can execute them. These rules are also called the syntax of a language.

Here are some important rules for programming the Arduino:

Functions
All programming languages provide an extensive collection of functions. A function is a block of program code that has a name and a set of instructions that are executed when the function is called. As part of learning a programming language, it is useful to look for and use new functions.

Example of a “Time-function”:

```
delay()
```

This call interrupts the program for the time (in milliseconds) specified as parameter. The number 1000 stands for 1000 milliseconds which results in a program interruption of one second. For example **delay(1000)** stands for an interruption for one second.

> Further functions can be found in the document "Arduino - List of Commands" or on the website "www.arduino.cc" under the item "reference" (www.arduino.cc/reference/en/). There, all possible programme functions, programme structures, operators, data types and so on are listed.

Curly brackets – {}

Curly brackets define the beginning and end of function and initiation blocks.

Example “void loop”:

```
void loop()
{
instructions;
}
```

An **opening curly bracket {** must always be followed by a closing curly bracket **}**. The number of brackets must always be balanced. Unbalanced parentheses often lead to cryptic opaque errors in the compiler, which are sometimes hard to retrieve, especially in large programs.

The Arduino IDE helps you to check the equality of the curly brackets. For this you simply have to select a bracket or click shortly after and the logically connected counterpart will be highlighted.

Semicolon – ;

A semicolon must be used at the end of a statement and is used to separate the elements of a programme.

Forgetting a semicolon at the end of a line leads to an error in the compiler. The error description can be very clear and point directly to the missing semicolon, but this does not have to be the case. Therefore, if an obscure or seemingly illogical error is reported, the first thing that should be done is to look for missing semicolons near the reported error and add them.

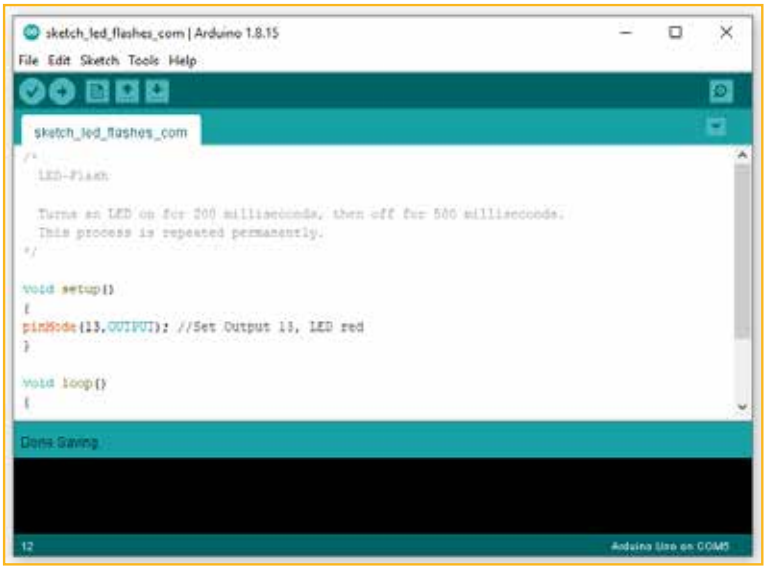
Single line comments – //

Simple one-line comments are defined with a // at the beginning of the line and end with the end of the line. They are ignored by the IDE and are not translated into machine code. Therefore they do not consume any memory.

Single-line comments are often used after instructions to convey more information about the instruction used or to record notes for future changes to the code.

Block comments – /* ... */

Block comments or multiline comments are text areas that are ignored by the program. They are used for longer descriptions or comments and help other authors to understand parts of the program. They start with **/*** and end with ***/** and can span several lines.



Because comments are ignored by the program and therefore do not consume memory they should be used generously.

> Block comments can also be used to invalidate entire programme sections for debugging. This way you can „turn off“ suspicious, possibly buggy programme parts without deleting them.

Variables

A variable is the designation of a numeric value with a name and memory location for later use in a programme.

As the name implies, the value of the variable can be changed continuously, in contrast to constants whose value remains constant during programme execution. A variable must be declared and can optionally be given a value.

Practical use of variables:

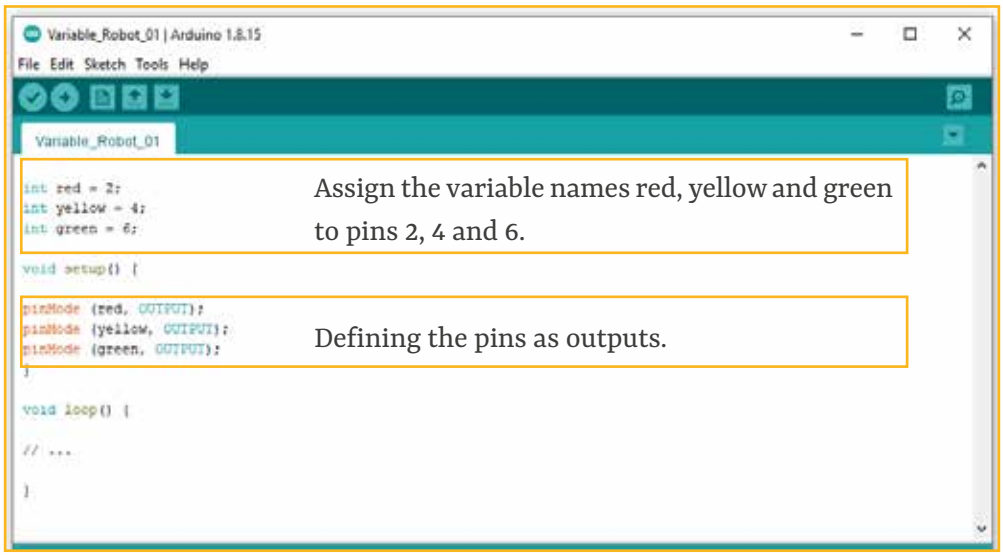
Each pin of the Arduino has a specific component in later projects. For example, if several LEDs are connected to different pins, it is easy to lose the overview when writing a sketch. This can be avoided by specifying certain variables. To make the sketch as clear as possible, names can be assigned to the pins. The names should describe the connected component or their function. The distribution of names to the pins is called declaration.

Normally, when writing a sketch, this name distribution is done before the setup area in the so-called declaration and initialization area. During further programming then only the name is used instead of the pin number. We are talking about „global variables“ here.

Declaring a variable means defining its type such as int, long, float, etc., assigning a name and optionally assigning an initial value.

In computer languages, a variable must always correspond to a predetermined fixed type. In addition to simple number formats such as “integer” (whole numbers in the range from -32768 to +32767) and floating-point numbers “float” (value range from -3.4028235E+38 and up to +3.4028235E+38), there are also data types for storing a character value, e.g. "char" or for storing a text string, e.g. "string".

For a complete overview, please refer to the references under the heading "Variables ":
https://www.arduino.cc/reference/en/



For pin 2, the variable name „red“ is assigned in the above example. The **„int“ command** is used to define the variable name „red“ as an integer number and to connect it to pin 2. During programming, only the name is used instead of the pin number. If, after this initialization phase, the name „red“ appears in a command when the sketch is processed, pin 2 is always addressed.

In the setup area the variable „red“, i.e. pin 2, is then defined as output with the help of the command „pinMode“.

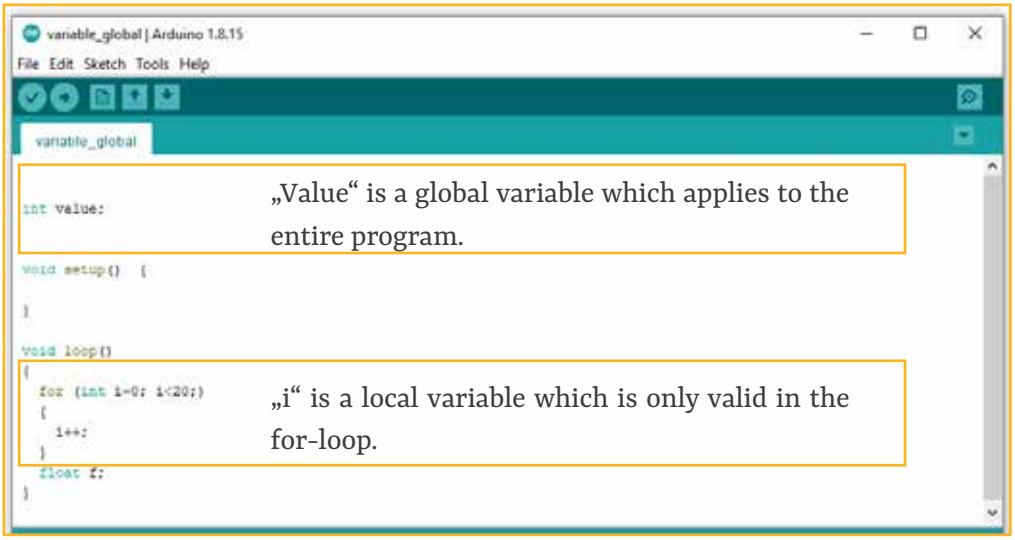
Scope of variables

Two types of variables can be distinguished, the global variables and the local variables. If the variable is defined in the setup area, then it is a global variable. This variable is seen by every function and every instruction of the programme to be able to access its content or value.
If, on the other hand, a variable is declared within a statement block, e.g. in a loop, it is called a local variable and is only available in the area in which it was defined. It is only visible and usable within

this function.

Therefore, it is possible for two or more variables with the same name to contain different values in different parts of the program. By ensuring that only the function itself has access to its own variables, the program is simplified and one reduces the risk of errors.

The following example shows the different variables and illustrates their scopes:



Arithmetic

Arithmetic operators include addition, subtraction, multiplication and division. They return the sum, difference, product or quotient of two operators.

Examples:

```
x = x + 12;
result1 = input1 + input2;
```

and so on.

Other mathematical functions such as root function, sine functions, absolute value determination, etc. are also available. (see also: www.arduino.cc/reference/en/)

The operation is performed while preserving the data types. For example, 9 / 4 becomes 2 and not 2.25 because 9 and 4 are integers and do not support decimal places. This also means that the operation can overflow if the result is larger than the data type allows.

If the operands have different data types, the larger type is used. For example, if one of the operands has the data type „float“ and the other „int“, floating point math is used for the calculation. For calculations that result in fractions, float variables should always be used. However, the use of float variables entails large memory requirements and slow speed of the calculations.

Other operators are comparative operators (e.g. x < y => x is less than y) and logical operators (e.g. AND, OR, NOT, etc.).

Sequence control

Control structures are used to control the program flow. A control structure offers the possibility to execute program parts only under certain conditions or to repeat program parts. The most important control elements in programming are selection (decisions) and the loops. In addition, there are some other control statements. The best-known control structure is the “if-else loop”.

if ... else

„if... else“ allows to make an „either ... or“ decision. For example, you want to check a digital input and execute different instructions in case of „HIGH“ than in case of „LOW“.

Example:

```
if (inputPin == HIGH)
{
digitalWrite(LED_red, HIGH);       //red LED is on
}
else
{
digitalWrite(LED_green, HIGH);       //green LED is on
}
```

The description of further control structures and programming examples (e.g. „for“, „while“, „do ... while“, „else“, etc.) can also be found at https://www.arduino.cc/reference/en/.

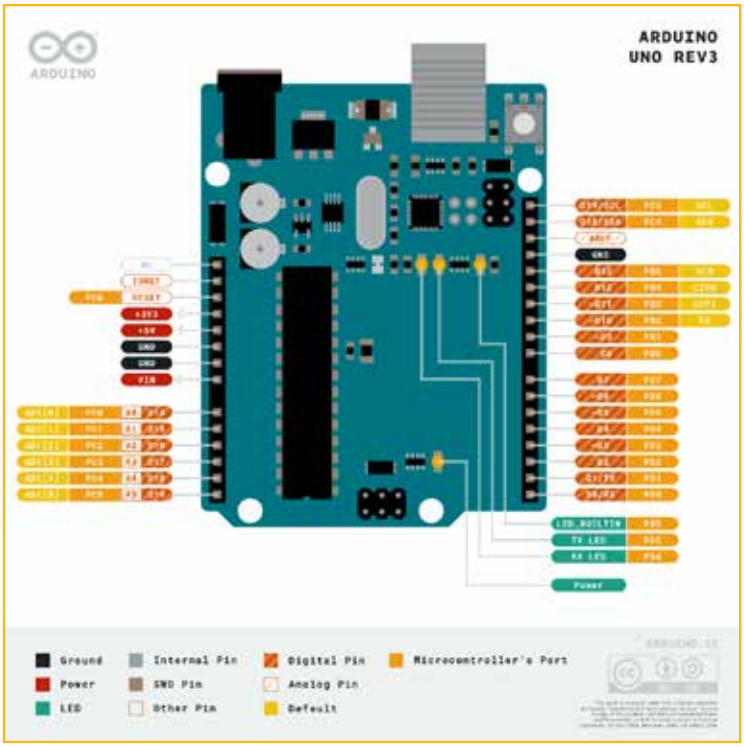
Input – Output

The pins of the Arduino must be defined in „void setup()“. A distinction is made between input / output or analog / digital.

```
pinMode(7, OUTPUT);       // sets pin 7 as an output
```

Not all pins can be set as analog outputs or inputs. As already shown in chapter 3, each Arduino board has a different number of specifiable inputs and outputs.

The Arduino UNO R3 board (see figure) has a maximum of 20 digital pins, 6 of which can act as analog inputs and 6 more as analog outputs via pulse width modulation (PWM). The analog input pins can also be used as digital pins (A0, A1, etc.).



The „void setup()“ command configures the specified pin to behave like either an input or an output. For details on pin functionality, see the Digital Pins page (www.arduino.cc/en/Tutorial/Foundations/DigitalPins).

The digital pins of the Arduino are inputs by default, so they don‘t need to be set as inputs with „pinMode()“. Pins configured as inputs have a high impedance state.

Pins configured as an output are in a low impedance state and can be loaded with a maximum of 40mA current from connected elements and circuits. This is usually enough to light up an LED (don‘t forget the series resistor!), but not enough to power most relays, solenoids or motors. Short circuits at the Arduino pins as well as too high currents can destroy the output pin or even the whole chipset. For this reason it is a good idea to connect an output pin with external elements in series with a 470 or 1k resistor. If in doubt, always read the corresponding data sheet carefully beforehand.

Reading inputting and outputting digital signals

The values applied to the pins are read in via corresponding commands and stored in the memory locations provided for this purpose.

The function „digitalRead()“ reads the value from a specified digital pin and receives either the value HIGH (1 if 5V is applied) or LOW (0 if 0V is applied). It is usual to store the read value in a suitable variable.

```
Pin7 = digitalRead(7);        // sets the value of the variable „Pin7“
                              // equal to that of the input pin
```

With the help of the function „digitalWrite()“ you write a HIGH or LOW value to a digital pin. If the pin is set as output pin and assigned with „HIGH“, the voltage is raised to the value 5V (or 3.3V), 0V for LOW.

Caution:
Always observe the maximum current carrying capacity of 40mA at output pins.

Reading and output of analog signals

The analog inputs of the Arduino are mainly used to read analog sensors. If, for example, a photo resistor is used as a sensor, which changes its resistance depending on the brightness, this sensor transfers a voltage of different value to the input pin with the help of a suitable circuit. This value is converted by the microcontroller and can then be related to a real brightness.

The assignment takes place via an A/D converter. It converts the analog signal into a digital one, the input signal becomes a bit combination which is stored in a memory and can be further processed by the processor.

Caution:
When working with external voltage sources, it is important to note the voltage for which the connected sensors are designed. Information on this can be found in the data sheets of the sensors. These also contain the most important key data (for example brightness in relation to the resistance) of the sensor. The maximum applied voltage (5V or 3.3V) must also not be exceeded.

The command **„analogRead()“** reads the value from the specified analog pin. Since the Arduino works with a 10-bit system, the resolution of the measured signals is not very fine. Integer values between 0 and 1023 are available to map the voltage range between 0V and +5V which corresponds to a resolution of about 4.9mV (exactly 4.833mV).

The digital signal is thus divided in steps of 4.833mV. An external signal must therefore change its value by at least 4.833mV to cause a signal change in the Arduino. More details can be found in the literature under the topic **„Analog-Digital-Conversion“**.

With most boards, the input width can be changed via the function **„analogReference()“**. So it is possible to achieve a larger resolution over a smaller value range.

The **„analogWrite()“** function writes pseudo-analog values to an output pin using hardware-based pulse width modulation (PWM). A value of 0 generates a steady voltage of 0 volts on a specified pin, a value of 255 generates a steady voltage of 5 volts. All values between 0 and 255 are possible. Since this analog value output is pulse width modulation, the pin changes very quickly back and forth between 0V and 5V for all intermediate values. The higher the required value is, the longer the pin HIGH is switched to 5V. At a value of 64, for example, the pin is set to 0V three quarters of the time and to 5V one quarter of the time.

Example:

```
analogWrite (7, 127);        // outputs a clocked voltage with the
                             // reference value of 2.5V at pin 7
```

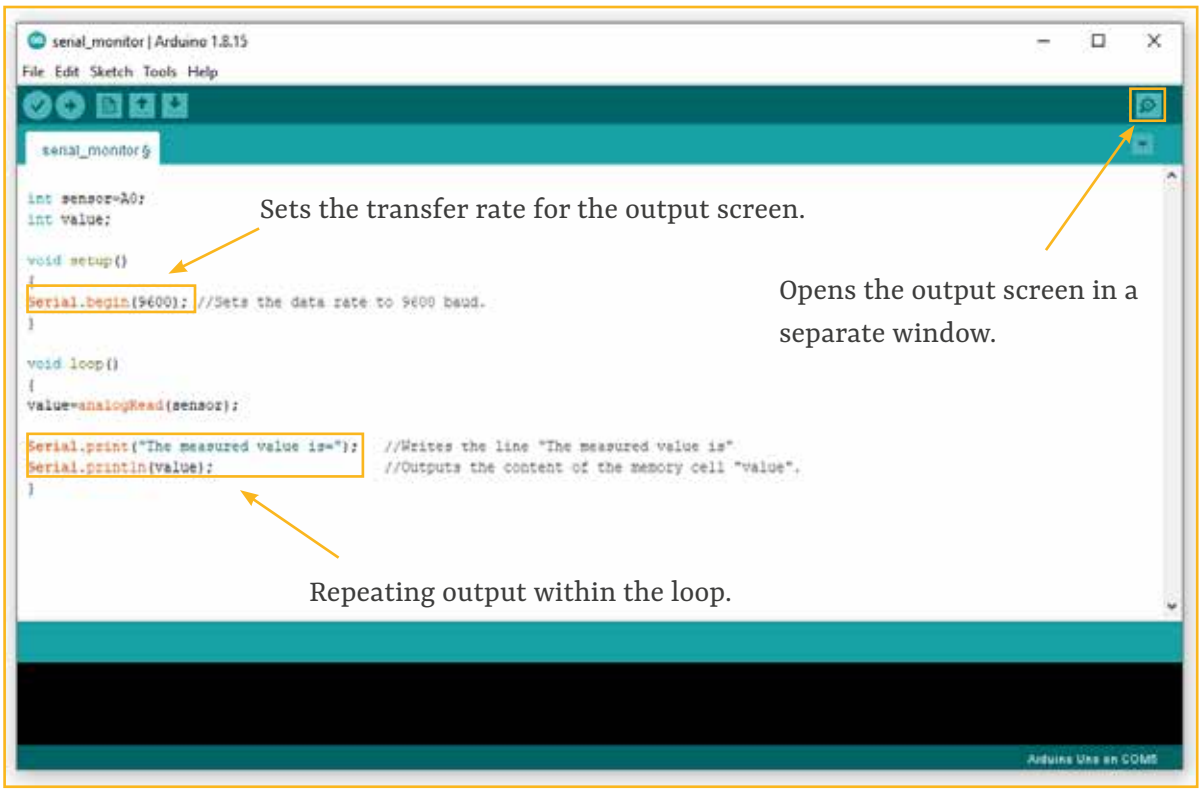
More about pulse width modulation (PWM) can be found at www.arduino.cc/en/Tutorial/Foundations/PWM.

Serial communication (Serial Monitor)

Finally, a very important function for monitoring the progress of the program should be briefly described.

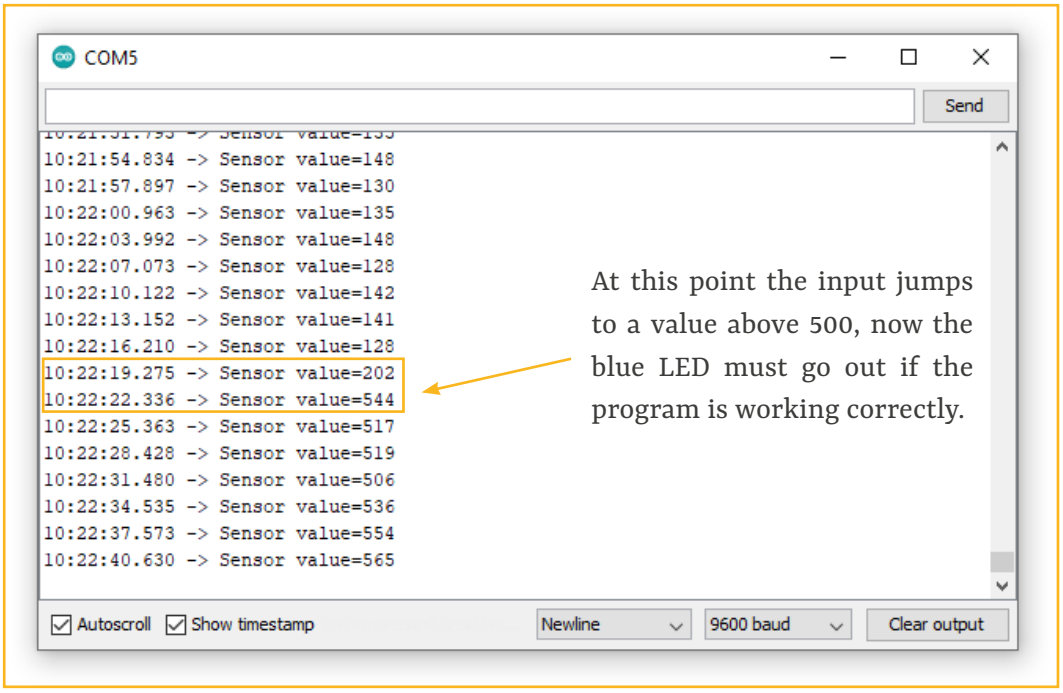
With the help of the „Serial Monitor“ it is possible to observe the values stored in one or more memory cells. If a sensor is connected to a pin, for example, it can be useful to display the measured values to get a feeling for the value range of the sensor. This knowledge can be used, for example, to set switching thresholds above certain values or to detect programming or circuit errors.

Example of preparation of the serial monitor:



The following program sequence shows the reading of a brightness sensor (photoresistor - LDR) connected to pin A0. The current value is stored under the variable **„sensorvalue“**. This value is read out by the command **Serial.println(sensorvalue)** every three seconds (delay(3000)) and passed to the serial monitor. In addition, the microcontroller switches on a blue LED if the value falls below 350.

By observing the brightness value via the serial monitor, you can check the safe programme function and possibly find a suitable threshold value for the switching process.



This programme sequence passes the data to the connected computer via the serial port so that the stored value can be output as human readable ASCII text. The Serial.print(„Sensor value=“) command writes the text Sensor value= before outputting the value. The Serial.println command causes a line continuation by the addition „ln“.

Further variations and functions for communication can be found at www.arduino.cc/reference/de/language/functions/communication/serial/

2.4 PREVIEW

Programming can only be learned and improved through constant practice and trial and error. By practicing on circuits, you will also get to know more and more new electronic components. Programming short sequences and building small experimental circuits will continuously expand your knowledge and you will increasingly venture into larger automation projects.

A very comprehensive work on programming and the commands of the Arduino in C can be found on the website https://www.ebooks.com. Search for the author Julien Bayle or for the title „C Programming for Arduino“. The book is a Free eBook and can be downloaded for free.

Many tutorials, videos and interesting projects based on the Arduino can also be found on the internet.

Just poke around the pages:

https://www.arduino.cc/education
https://create.arduino.cc/projecthub
https://www.thingiverse.com (Search for Arduino)
https://howtomechatronics.com/arduino-projects/
https://www.tinkercad.com

Further in-depth basics and useful procedures are presented in practical exercises (worksheets).

Have fun entering the world of automation.

Notes

Sources

Unit 1

Vogel-Heuser B, Diedrich C, Broy M, 2013, Anforderungen an CPS aus Sicht der Automatisierungstechnik (at) 61(10): 669–676. doi:10.1524/auto.2013.0061

Wikipedia MES: Author: UlrichAAB, Import from de.wikipedia: https://de.wikipedia.org/wiki/Manufacturing_Execution_System
(version: 18.04.2019)

E. A. Lee, "Cyber Physical Systems: Design Challenges", Technical Report No. UCB/EECS-2008-8
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-8.html

„Integrierte Forschungsagenda Cyber-Physical Systems", Acatech 2012
http://www.acatech.de

Amar InfoTech
https://www.amarinfotech.com/iot-logistics-assets-solutions.html (version: 26.03.2019)

Internet der Dinge und Dienste, Bosch Software Innovations 2012

SRI Consulting Business Intelligence – National Intelligent Council (2008). Disruptive Civil Technologies: Six Technologies with Potential Impacts on US Interests out to 2025, Conference Report.

www.create.arduino.cc
https://create.arduino.cc/projecthub/ (version: 26.03.2019)

CONELCOM GmbH, Innsbruck, Austria www.controllino.biz:
https://www.controllino.biz (version: 26.04.2019)

Cyber-Physical Systems: Chancen und Nutzen aus Sicht der Automation, 2013
VDIIVDE-Gesellschaft Mess- und Automatisierungstechnik

Automation 2020: Bedeutung und Entwicklung der Automation bis zum Jahr 2020, 2012
VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik

Informationssicherheit in der industriellen Automatisierung, Richtlinie VDI/VDE 2182, VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik

Infineon Technologies AG
https://www.infineon.com/cms/de/discoveries/edge-computing (version: 26.07.2021)

www.wikipedia.org
https://en.wikipedia.org/wiki/Edge_computing (version: 28.07.2021)
Image credits: „Artificial intelligence – How does machine learning work?“
von Erfurth Kluger Infografiken GbR and iRights.Lab, lizenziert unter CC BY-SA 4.0 INTERNATIONAL, Herausgeber/-in Siemens Stiftung
(version: 28.07.2021)

Wikipedia ATmega328:
https://en.wikipedia.org/wiki/ATmega328 (version: 18.04.2019)

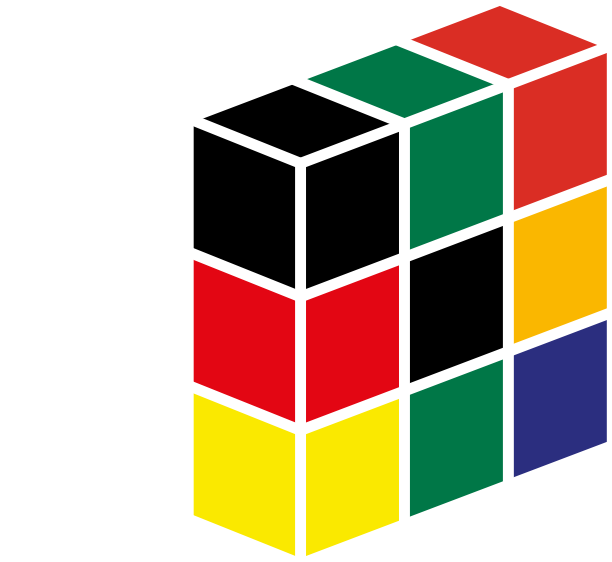
Heinrich B, Linke P, Glöckler M, 2015, Grundlagen der Automatisierung – Sensorik, Regelung, Steuerung, Springer Vieweg, doi: 10.1007/978-3-658-05961-3

Unit 2

Wikipedia IPO model: https://en.wikipedia.org/wiki/IPO_model
(version: 29.07.2021)

Wikipedia Computer program: https://en.wikipedia.org/wiki/Computer_program
(version: 29.07.2021)

Arduino.cc: https://www.arduino.cc/



TRAINME

This Guide is developed within the bilateral programme TRAINME in order to promote in-service lecturer in the fields of mechanical and electrical engineering at South African TVET colleges. The programme is designed by Inter-Company Training Center in Easter Bavaria (ÜBZO) and University of Stuttgart on behalf of the German Federal Ministry of Education and Research and DLR in cooperation with the South African Department of Higher Education (DHET).