Application No. 2024-2-HU01-KA210-VET-000271388, Application of Industry 4.0 technologies in the production and operation of smart homes

Content

1. Project presentation

1.1. Introduction – The project and the educational environment

This manual is the professional and methodological documentation of an international collaborative educational project based on Industry 4.0 and IoT technologies. The project focuses on a smart house/smart farm model, the development of which will involve students from digital design and additive manufacturing to electronics integration and software development, to the creation of a complex, web-based building management system.

The project started with a commercially available training kit (Smart Farm Kit), which provided a good foundation for learning about sensors, actuators and IoT-based control. However, the development process did not stop at the application of factory solutions: the participants performed critical analysis, identified system limitations, and then responded to these by developing their own scalable system architecture, following an industrial model.

The aim of the manual is not only to document a specific project implementation, but also to present an open, adaptable educational model that can be followed and further developed by other institutions and student groups, even without the use of factory-made elements.

1.2. Purpose and role of the manual

The aim of this manual is to provide comprehensive professional and methodological guidance for an educational project that supports the development of students' digital, technical and engineering competencies through the practical application of Industry 4.0 and IoT technologies. The document does not only provide ready-made solutions, but also presents a learning and development process in which problem solving, analysis and iteration play a prominent role.

The manual's approach is that education is not based on passive knowledge transfer, but on active creative activity. During the project, students are not just users, but designers, implementers and developers of a complex technical system. The document supports this approach throughout, from design, production and programming to system integration.

The primary function of the manual is to be used as a teaching aid. Its content is suitable for:

- serve as a basis for classroom and extracurricular project work,
- fit into vocational, talent management or project-based training programs,
- provide methodological support for teachers in the implementation of complex technical and IT projects.

The methodological approach focuses on project-based and discovery learning. Students gradually progress from simpler subtasks to complex system understanding in sequential steps. Accordingly, the manual does not provide a single “correct”

It not only records a solution, but also presents alternatives, decision points and development opportunities.

The document is relevant for several target groups:

- For students: the manual presents the development processes with a student-friendly logic, providing the opportunity for independent experimentation, making mistakes, and redesigning. During the project, students experience the basics of engineering thinking: they define a problem, design a solution, test, evaluate, and refine it.
- For educators: the manual supports lesson organization, differentiated assignment, and the management of heterogeneous groups. It provides assistance in how to involve students with different backgrounds in a common project, and how to process technical content in a pedagogically structured yet flexible way.
- For institutions: the project presents an adaptable model that can be integrated into local curricula, project weeks or international collaborations. The open nature allows institutions to adapt the implementation to their own equipment, infrastructure and training profile.

1.3. Brief introduction and objectives of the project

Connection to technical and secondary IT training: the project originally started in a technical environment, where students of industrial IT, mechatronics and related technical training already have basic technological knowledge. However, during the international cooperation, a theoretical high school also joined the program, whose students had no prior technical experience. This situation represented both a challenge and a pedagogical opportunity.

Advantages:

- The collaboration of students with different backgrounds strengthened explanatory and systematic thinking.
- The technical school students took on the role of mentors, which deepened their own knowledge.
- For the students of the theoretical lyceum, the project formed a bridge between abstract IT knowledge and physical systems.

Possible difficulties:

- The lack of basic technical knowledge may have slowed down initial progress.
- Increased emphasis had to be placed on the step-by-step introduction of basic concepts (electronics, sensors, control).
- The role of the instructor has become more facilitative and supportive.

Based on the experiences of the project, it can be stated that the presented model is not only functional in a technical school environment. With an appropriate methodological structure and differentiation, it can also be successfully applied in secondary, non-technical institutions.

1.4. The basic idea of the "smart house / smart farm" project

The basic idea of the project is to create a physical and digital model that tangibly illustrates the operation of smart systems, while presenting the application of modern technologies through understandable and real-world problems for students. The smart house / smart farm model creates an opportunity for technical, IT and environmental aspects to appear as a unified system in education.

The model is suitable for simultaneously presenting:

- the role of sensors and actuators in monitoring and controlling environmental conditions,
- the process of data collection, data processing and feedback,
- the logic of control and supervision in local and remote environments,
- web-based, remote access options as a tool for energy and resource-efficient operation.

In the project, the smart house and the smart farm appear not only as a technological demonstration, but also as a carrier of an environmentally conscious approach. Through sensor-supported measurement and automated intervention, students learn how intelligent control can contribute to the optimization of energy consumption, the conscious use of environmental resources, and sustainable operation (e.g., lighting, ventilation, irrigation control).

The model is not a final product, but a consciously open, continuously evolving learning tool that supports the development of systems thinking through cycles of design, experimentation, evaluation and redesign. This approach helps students not only master technical solutions, but also understand their environmental, economic and social contexts.

1.5. Integrated Industry 4.0 educational architecture and international development environment

The project follows the principles of the Industry 4.0 approach:

- distributed system architecture,
- data-driven operation,
- use of networked devices,
- using standard communication protocols.

The use of IoT technologies allows students to not work with isolated devices, but to interpret hardware and software components as part of a coherent system.

The Smart Farm Kit was deliberately designed as a starting point. One of its key elements was to analyze the factory system, identify its limitations, and then respond to these by developing its own scalable system architecture that follows an industrial pattern. This process highlighted the fact that critical thinking and iterative development are the foundations of technological progress.

The approach is entirely project-based. Students do not follow predefined steps, but rather work through problems, make their own decisions, and analyze their consequences. During the learning process, mistakes are not seen as failures, but as a natural part of development.

Physical models, immediate feedback, and visible results naturally increase student motivation. Gamification elements such as challenges, milestones, and shared goals appear throughout the project to structure and maintain interest.

It develops digital competencies in a complex way: programming, database management, network knowledge, digital design and documentation are all present. It also strengthens engineering thinking, systems thinking, and the recognition of relationships between functional units.

The project was implemented in international cooperation, which provided significant added value in terms of competence development and pedagogical content in addition to the professional content. The participating institutions operate in different countries, educational systems and institutional profiles, so the joint work created a real collaborative environment that well models future labor market situations.

Language differences were prominent during the joint activities. In many cases, professional communication took place in English, which improved the students' foreign language competences and made them aware of the importance of precise, clear technical communication.

Multicultural collaboration became a natural part of the work process. The joint work of students with different educational backgrounds and problem-solving strategies strengthened flexibility, adaptability and professional dialogue.

Its operation reflected the expectations of multinational companies in several aspects: communication in foreign languages, joint documentation management, teamwork and responsibility in complex systems. The experiences gained contributed to the students being able to interpret their technical knowledge in a broader, international context.

2. Project structure and implementation logic

2.1. Project phases and their interdependence

The project was implemented in stages that were consciously built on each other. This structure was not only a technical necessity, but also a pedagogical decision: the aim was for students to gradually, building on their own experiences, reach independent development tasks. The phasing allowed for the involvement of students with different backgrounds, as well as for reflection and evaluation related to each step.

In the first phase, the students encountered a ready-made, working system. The goal was to learn the basic concepts, operational relationships, and system logic. The emphasis was not on “copying” the solution, but on understanding: what the system does, why it does it, what elements it consists of, and how they are connected to each other.

This stage proved to be particularly important for students with less technical backgrounds, as it created a common starting point and lowered the barrier to entry for later, more complex tasks.

The second phase, the conscious analysis phase, followed the familiarization phase. The students not only used the system, but also examined its operational limits, strengths and shortcomings. This step played a key role in understanding that a technical solution is always the result of compromises and is optimized for specific purposes.

Through critical evaluation, they learned to formulate their own development needs and recognize when it is appropriate to further develop or rethink an existing system. This way of thinking is directly related to the principles of engineering and IT problem solving.

In the third phase of the project, the students began their own development work, building on their analytical experiences. Here, the process was no longer guided by predetermined steps, but by jointly made decisions, iterative planning, and continuous testing.

At this stage, the advantage of the project approach became truly visible: the students experienced that a complex system is not "completed" all at once, but takes shape gradually, and every decision has an impact on the entire operation.

2.2. Participants and roles

A defining feature of the project was the conscious division of roles among the participants. Students, teachers and institutions worked together in different but mutually reinforcing roles, which contributed to the flexibility and sustainability of the project.

During their participation, the students were not passive recipients, but active developers. They collaborated in planning, interpreting problems, testing and evaluating solutions. During the distribution of subtasks, they experienced the importance of taking responsibility and how their work fits into a more complex system. This role strengthened independence, cooperation and long-term thinking.

The role of the instructors was primarily supportive and guiding. The emphasis was on guidance, questioning, and structuring the learning process, rather than on providing ready-made solutions.

This approach allowed students to find solutions through their own experiences, while providing them with a solid professional and pedagogical background.

International cooperation resulted in mixed learning groups, in which students with different educational backgrounds and experiences worked together. This situation strengthened communication skills, adaptability and joint problem-solving.

Mixed teamwork accompanied the entire project and created real collaborative situations that went beyond the traditional school framework.

2.3. Learning outcomes and competencies to be developed

During the design, it was a conscious goal to not narrow the learning outcomes to a single competency area. The development process supported the development of technical, cognitive and social skills at the same time.

The project contributed to the conscious use of digital tools, structured work and creation in a digital environment. Students experienced how a complex system becomes understandable and manageable.

They encountered complex problems that required multiple steps, planning, and feedback to solve. This strengthened systems thinking, the recognition of cause-and-effect relationships, and the ability to plan for the long term.

Documenting and presenting the work completed was an integral part of the project. Students learned how to transparently record a development process and present it in an understandable form. The resulting portfolios also enable the long-term utilization of learning outcomes.

3. Detailed presentation and processing of the Smart Farm Kit The

3.1. role of the Smart Farm Kit in the learning process

The Smart Farm Kit is a modular IoT kit designed for educational purposes, demonstrating the basic principles of smart systems through a tangible model. The kit aims to expose students to a real, working system before they start developing their own solutions.

In the project, the kit is not presented as a final product, but as a learning tool. During processing, the emphasis is on understanding the operation of the elements, mapping the logic of the system, and laying the foundation for later development.

3.2. Main components of the Smart Farm Kit

The Smart Farm Kit's electronic components are designed to demonstrate a complex IoT system, including multiple input sensors, control modules, and output actuators. The list is based on the manufacturer's source and reliable product descriptions, and covers the actual devices used during the kit's learning program.

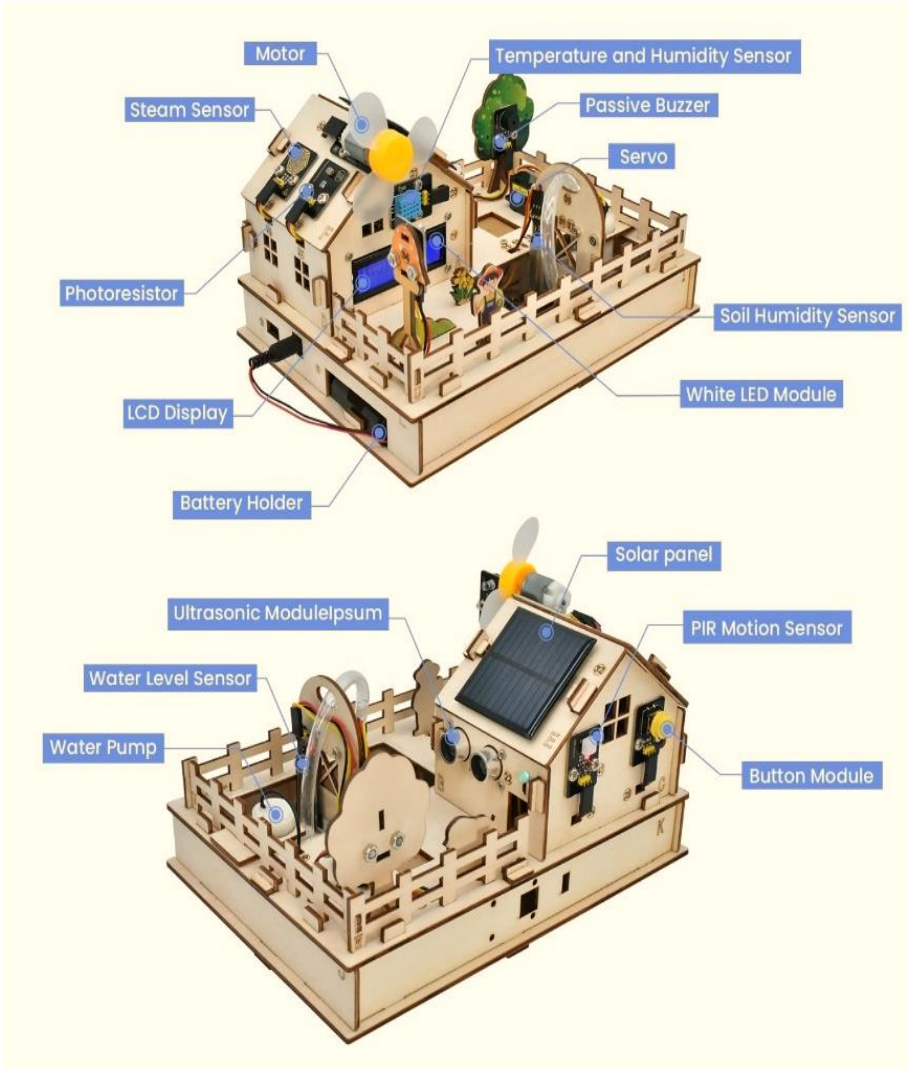


Figure 1: Main parts of the Smart Farm Kif (Source:https://www.keyestudio.com)

3.3. Controller and base modules

- ESP32 microcontroller:
  - he Structure: Microcontroller integrated with Wi-Fi and Bluetooth module
  - he Function: The “brain” of the kit, handling sensor data, logic, and communication.
  - he Educational significance: Study of network communication, parallel event handling, and I/O logic.

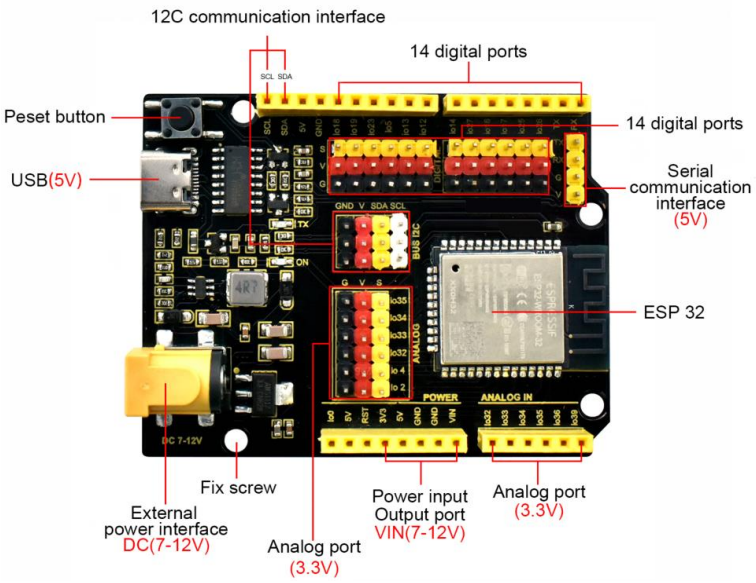


Figure 2: ESP32 microcontroller (Source: https://docs.keyestudio.com)

- Relay module:
  - he Construction: Opto-isolated relay with 5 V control
  - he Function: Switching devices with higher consumption (e.g. pump).
  - he Educational goal: Isolation and safe control of actuators.

Figure 3: Relay module (Source: https://docs.keyestudio.com)

3.4. Input elements – Sensors

- Photoresistor: he

  Construction: photosensitive semiconductor resistor

  - he Operation: the resistance changes depending on the light intensity (analog signal).
  - he Educational purpose: measuring light conditions, handling analog inputs.

Figure 4: Photoresistor (Source: https://docs.keyestudio.com)

- Soil moisture sensor: he

  Structure: two electrodes and transducer

  he Operation: water in the soil reduces resistance → higher signal value. Educational goal:

  he analog measurement techniques and interpretation of environmental data.

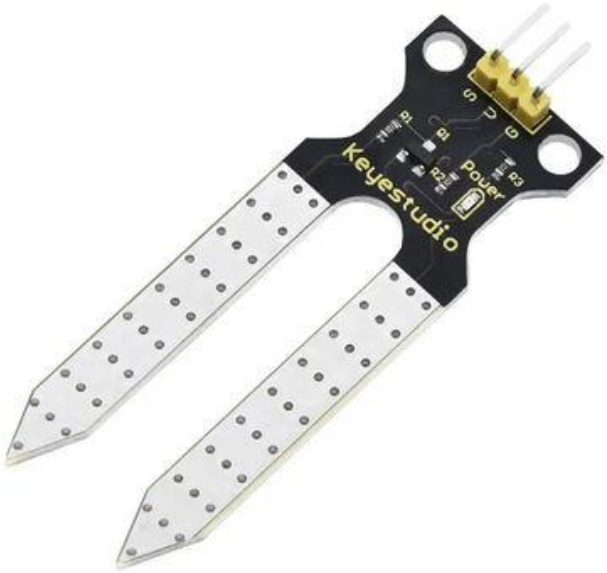


Figure 5: Soil moisture sensor (Source: https://docs.keyestudio.com)

- Water level sensor:

  he Structure: multi-electrode contact

  he Operation: closing/opening signal based on liquid level.

  he Educational purpose: digital event-driven measurement.

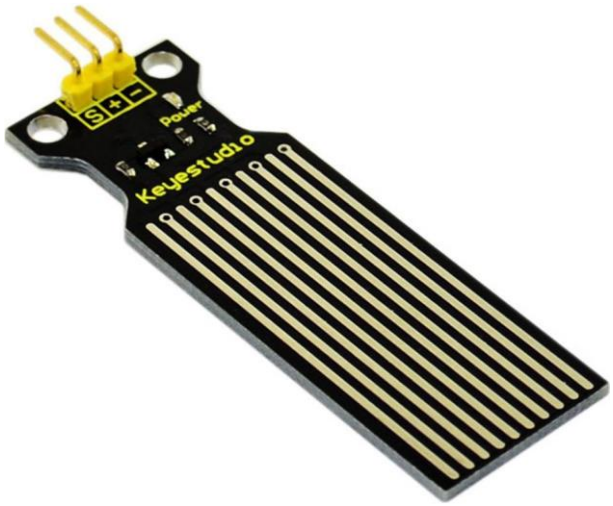


Figure 6: Water level sensor (Source: https://docs.keyestudio.com)

- DHT11 temperature and humidity sensor: he

  Structure: digital sensor integrated into the unit

  he Function: calculates temperature and humidity with an integrated algorithm. Educational

  he purpose: digital data communication and measurement of the physical environment.

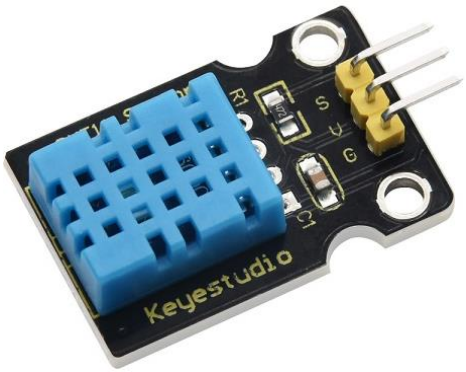


Figure 7: DHT11 temperature and humidity sensor (Source: https://docs.keyestudio.com)

- SR01 V3 ultrasonic distance sensor: he

  Structure: transmitter and receiver piezo elements.

  he Operation: ultrasonic pulse → reflection → distance calculation from time measurement

  he Educational goal: understanding physical waves and time-based measurement.

Figure 8: SR01 V3 Ultrasonic Distance Sensor (Source: https://docs.keyestudio.com)

- PIR motion sensor:

he Construction: passive infrared sensor
he Function: Detects sudden temperature changes in the field of view.
he Educational objective: Event-driven logic and real-time sensing.

Figure 9: PIR motion sensor (Source: https://docs.keyestudio.com)

- Passive buzzer: he

  Construction: piezo speaker
  he Operation: responds to voltage pulses by emitting sound Educational objective:
  he connection between output modules and practical signaling system.

3.5. Output elements – Actuators and displays

- White LED module: he

  Construction: pre-mounted LED
  he Operation: digital output controls lighting Educational
  he purpose: visual feedback for simple logic.

Figure 10: White LED module (Source: https://docs.keyestudio.com)

- SG90 9G servo motor: he

  Structure: micro servo motor with position feedback
  he Operation: position control based on PWM signal

heEducational goal: precise control of actuators and mechanical effects.

Figure 11: SG90 9G servo motor (Source: https://docs.keyestudio.com)

- DC motor/fan: he
  Construction: DC motor with small fan
  he Function: output device that serves to move ambient air Educational
  he purpose: motor control and performance monitoring.

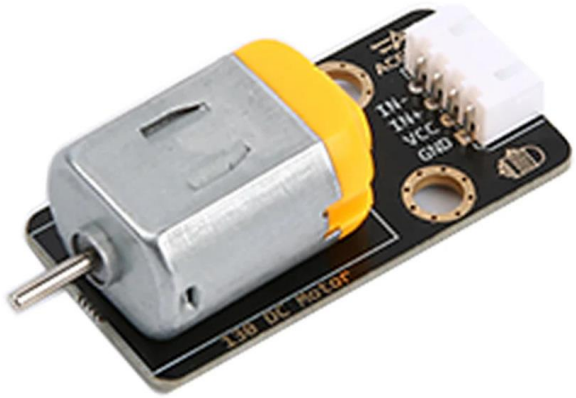


Figure 12: DC motor/fan (Source: https://docs.keyestudio.com)

- I2C 1602 LCD display:
  he Structure: 2 lines × 16 characters, I2C communication
  he Function: display character data from microcontroller messages
  he Educational goal: learn data display and communication protocols.

Figure 13: I2C 1602 LCD display (Source: https://docs.keyestudio.com)

- Solar panel:
  - he Structure: small photovoltaic panel Operation:
  - he photovoltaic effect → electricity Educational goal:
  - he renewable energy sources in IoT systems.

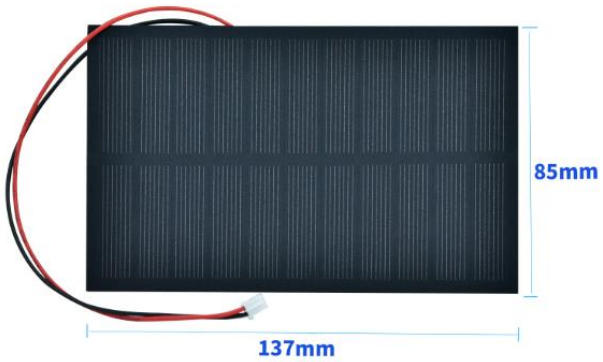


Figure 14: Solar panel (Source: https://docs.keyestudio.com)

- Water pump:
  - he Construction: small DC pump
  - he Operation: liquid transfer with relay control
  - he Educational goal: understanding actuator control and mechanical work.

Figure 15: Water pump (Source: https://docs.keyestudio.com)

3.6. Accessories and mounting tools

- transparent acrylic elements and wood chip covering
- power cables, DuPont wires, screws, fasteners
- battery carrier or battery holder
- USB cable for communication and power supply

3.7. The role of components in education

The 17 electronic components included in the kit cover the introductory IoT and automation curriculum well:

- sensors provide signal transmission from the physical environment,
- the control unit interprets and decides,
- actors display or intervene in the environment.

This structure provides a systematic overview of how elements work together in a real IoT system and how they can be connected with programming, measurement logic, and control.

Assembling the Smart Farm Kit is the first step in the project, where students work with a complex, multi-component physical system. The factory documentation guides the user step by step through the assembly of mechanical and electronic elements, which can also be interpreted as a conscious learning process.

Assembly is not merely a technical task, but rather an experience that lays the foundation for later independent design and manufacturing work.

3.8. The role of the factory assembly instructions

The Smart Farm Kit factory manual serves as the primary reference during assembly. The manual:

- determines the sequence of each assembly step,
- shows the correct position and fastening of the parts,
- helps to safely connect electronic components,
- supports understanding with visual illustrations.

While using the guide, students will experience that putting together an engineering system is not improvisation, but a structured, documentation-based process.

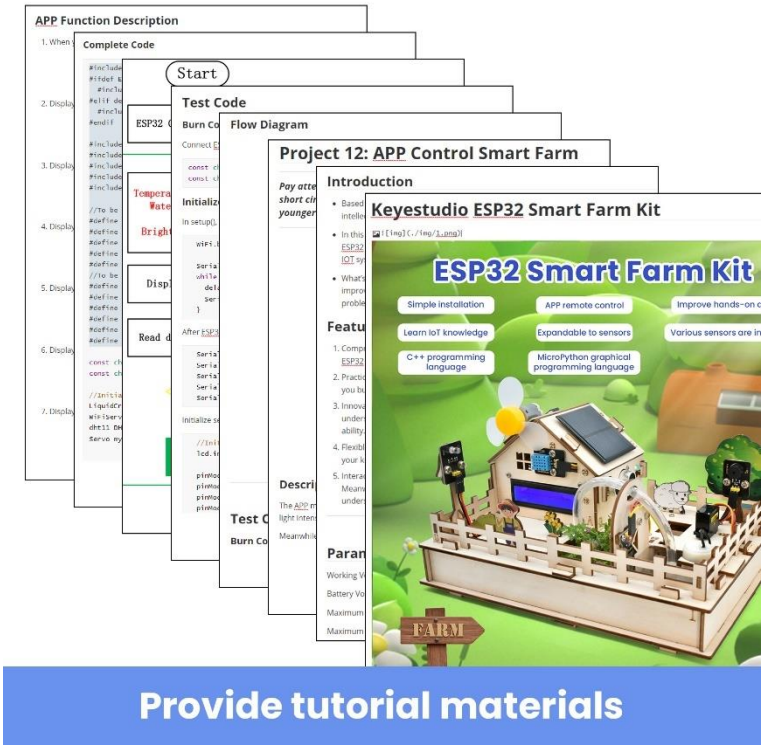


Figure 16: Factory assembly instructions (Source: https://www.keyestudio.com)

3.9. Mechanical and electronic assembly and modular design

During assembly, students will be introduced to the modular mechanical design of the kit. Each element has pre-designed mounting points based on standard connection distances, allowing for the replacement and relocation of components.

This modular approach conveys an important engineering principle: the expandability and maintainability of the system are decided at the physical design level.

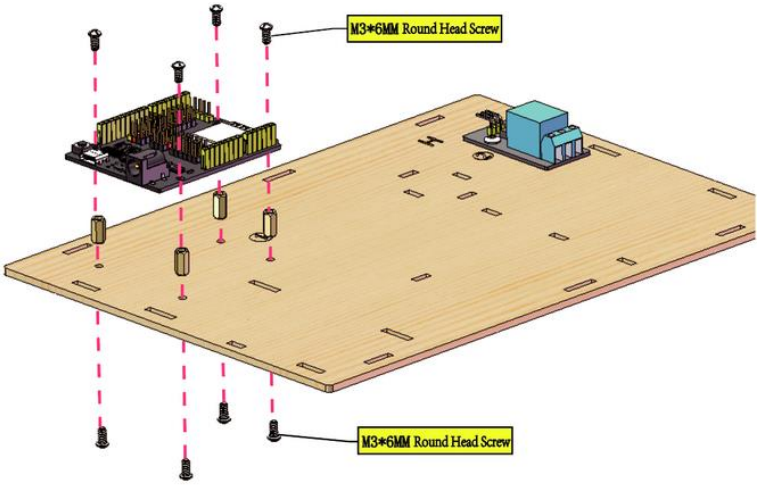


Figure 17: Smart home motherboard (Source: https://docs.keyestudio.com)

During electronics assembly, students identify the connection points of sensors and actuators, learn about the role of power supply and signal wires, and follow factory wiring diagrams.

During the connections, special attention is paid to the correct handling of polarity, the distinction between digital and analog inputs, and the mechanical protection and clarity of the wires.

This step lays the foundation for later troubleshooting and system diagnostic thinking.

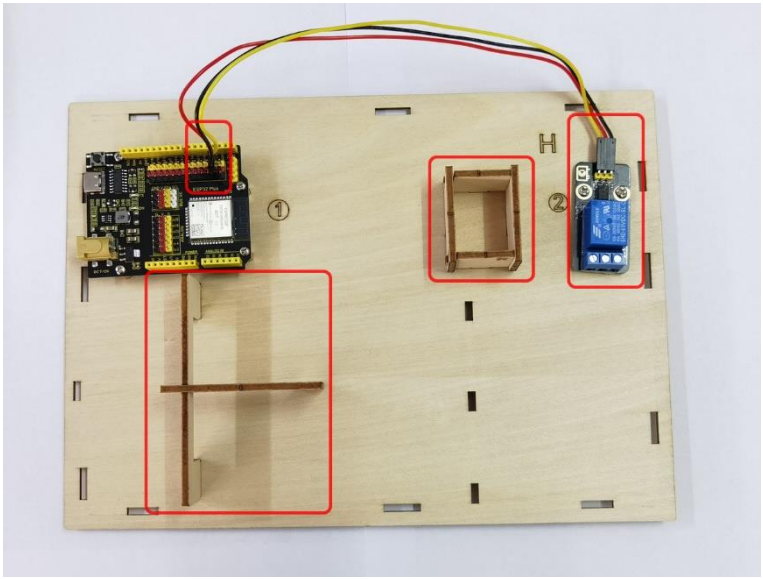


Figure 18: Smart home motherboard connection points (Source: https://docs.keyestudio.com)

Assembly is not a one-time, closed process, but an iterative activity. After each stage of assembly, students:

- check the mechanical stability,
- they test the electrical connections,
- The operation of each element is validated with short test programs.

This approach prevents later complex errors and draws attention to the importance of gradual verification.

3.10. Engineering considerations based on assembly experience

During the assembly process, several engineering aspects become observable that can be directly utilized in the subsequent creation of self-designed elements:

- Geometric fit, the accuracy of the dimensions and mounting points of the components fundamentally affect the assembly.
- Designing cabling routes and arranging wires affects transparency, maintainability and aesthetics.
- Accessibility and serviceability, the placement of sensors and controls affect the possibility of later modifications and expansions.

- In terms of modularity and interchangeability, the factory solution highlights how a system can be built in such a way that individual elements can be operated independently.

These experiences make students aware that physical design and functional operation form an inseparable unit.

As the assembly phase concludes, students are no longer just users, but also analysts of the system. The experiences gained here directly prepare them for:

- the subsequent geometric measurements,
- your own 3D design tasks,
- the design of individual fastenings and housing elements,
- further development or replacement of factory solutions.

Assembling the Smart Farm Kit is therefore not an end point, but a starting point for a higher-level, independent engineering creative process.

3.11. Programming the Smart Farm Kit Block-

3.12. based programming environment

Block-based programming plays an important role in understanding systems and in the transition to other programming models. The Smart Farm Kit's graphical programming environment (KidsBlock) follows Scratch-based logic. Programming is done by fitting visual blocks together, which are backed by real, executable control instructions. This allows students to avoid having to deal with the syntactic details of programming languages while controlling the actual operation of the system.

From an educational perspective, this environment serves several mutually reinforcing functions:

- It separates logical thinking from linguistic difficulties, so students can focus on control processes, cause-and-effect relationships, and state changes.
- It provides immediate feedback, as the operation of the physical system can be directly observed after modifying the program.
- It supports iterative experimentation, the natural learning cycle of trial-error-improvement.
- It prepares you for later textual programming, as basic structures such as event handling, condition testing, loops, and state management appear behind the blocks.

Block-based programming in this project is not an end point, but a consciously chosen entry level on a multi-step programming learning path.

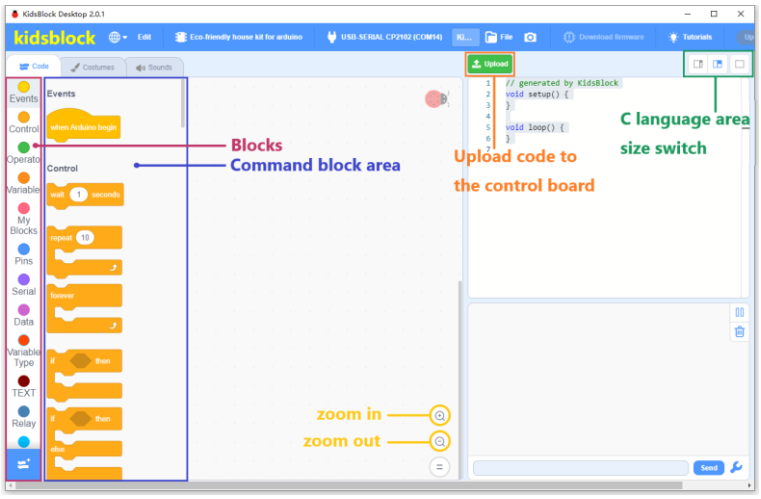


Figure 19: Block-based programming interface, overview of main block categories (Source: https://docs.keyestudio.com)

3.13. Connecting to text-based programming environments

The underlying control unit of the Smart Farm Kit, the ESP32 microcontroller, is not only programmable in a graphical environment. The project also provides the opportunity to implement the same functions in a text-based programming language, such as an Arduino-based environment.

The Arduino environment:

- It uses C/C++ based syntax,
- allows low-level hardware management,
- It gives you direct insight into the program's execution, variables, and memory usage.

In pedagogical terms, this shift:

- helps you understand that there is specific code running behind block-based elements,
- develops the ability to abstract (block ↔ line of code correspondence),
- prepares you for the use of more professional development environments.

During implementation, block-based and text-based programming appear as complementary tools rather than mutually exclusive.

The educational value of the Smart Farm project is further enhanced by the fact that the control logic learned by the students is closely related to the programming principles used in industrial automation.

Professional opportunities are further expanded by the open source OpenPLC software platform, which enables IEC 61131-3 programming not only on classic PLCs, but also on Arduino and ESP32-based controllers. The OpenPLC environment provides full IEC 61131-3 language support, thus enabling the practice of industrial control logic on low-cost hardware, and supports industrial communication protocols, in particular the Modbus and Modbus TCP standards.

This approach bridges the gap between educational microcontroller development and real-world industrial automation systems. Students can learn about the

with industrial programming, which is technologically simpler but fully compatible with professional systems in its approach.

3.14. Link to later stages of the project

In the further phases of the Smart Farm project, industrial communication will play a key role. Accordingly:

- the use of the Modbus / Modbus TCP protocol forms the basis of system integration,
- Arduino-based, C language programming is still used in the microcontroller control,
- The block-based and IEC 61131-3 approach serves primarily as a thinking and design framework.

This decision ensures that students learn about industry standards and expectations while working in a flexible, education-friendly development environment, and prepare for later work with PLC-based or industrial IoT systems.

The Smart Farm project uses block-based programming not in isolation, but embedded in a broader industrial context. The introduction of the IEC 61131-3 standard, the OpenPLC platform/ project and Modbus-based communication allows students to acquire a systems approach at an intermediate level that directly matches the expectations of the modern, multinational industrial environment, while maintaining the safe, gradual and motivating nature of the education.

3.15. Project 1 – Lighting System

Lighting control is one of the simplest, yet educationally important sample projects of the Smart Farm Kit. In this task, the digital input → decision logic → digital output control chain appears for the first time as a complete, closed unit. The goal of the project is not to understand lighting as a function, but to understand the basic logic of automation.

The push button as a digital input

A pushbutton is the simplest digital input device that can transmit two distinct states to the controller:

- not pressed state → logical LOW (0),
- pressed state → logical HIGH (1).

For the controller (ESP32), the push button is not a "physical object", but a digital signal that appears on a specific input pin. The state of the input is typically stabilized by an internal or external pull-up or pull-down resistor (pull-up/pull-down), so the controller senses a clear HIGH or LOW value at all times.

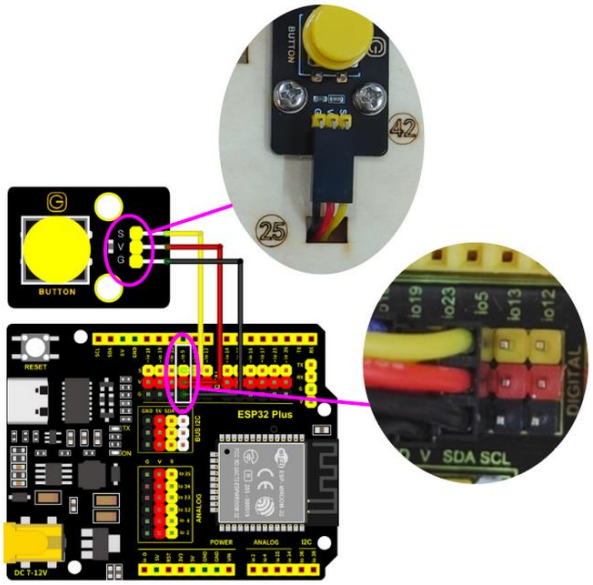


Figure 20: Pushbutton wiring diagram as digital input (ESP32 GPIO) (Source: https://docs.keyestudio.com)

The push button is excellent for introducing the concept of a digital signal, as there is no "intermediate" value, the state can always be interpreted clearly, and the information does not come from the magnitude of the value, but from the change in state.

This approach is in sharp contrast to analog sensors, where the signal is continuous and requires numerical interpretation. Understanding the difference between the two is essential in learning automation systems.

The push button is just one of many digital inputs. During the project – or as a further development of it – the following devices also operate on a similar principle:

- Reed relay (magnetic contact): closes or opens a circuit under the influence of a magnetic field; typical application is door and window detection.
- Limit switch: used to detect the limit position of mechanical movement.
- PIR motion sensor digital output: gives a HIGH signal when motion is detected, otherwise remains in the LOW state.
- Digital relay contact feedback: a common form of feedback in industrial systems.

The common feature of these devices is that they provide status information, not a measured quantity.

The educational significance of using digital inputs – especially push buttons – is that it develops several fundamental thinking patterns:

- event-driven thinking (something happens → reaction follows),
- the basic concept of the human-machine interface,
- the abstraction, during which physical action is transformed into digital information,
- understanding of later state machine and control logics.

At this point, students first understand that an automated system does not "constantly monitor" but responds to events, which is a fundamental principle of industrial controls.

Digital output – LED as controlled actuator

The LED (Light Emitting Diode) is one of the simplest digital output devices in the Smart Farm Kit, providing a visual representation of the controller's decisions. It operates in two states:

- output LOW → LED off,
- output HIGH → LED on.

The digital output pin of the controller (ESP32) drives the LED typically through a current-limiting resistor, which protects the component from overcurrent. Here, students first encounter the practical fact that the controller cannot directly drive any load, but must be designed taking into account the electronic environment.

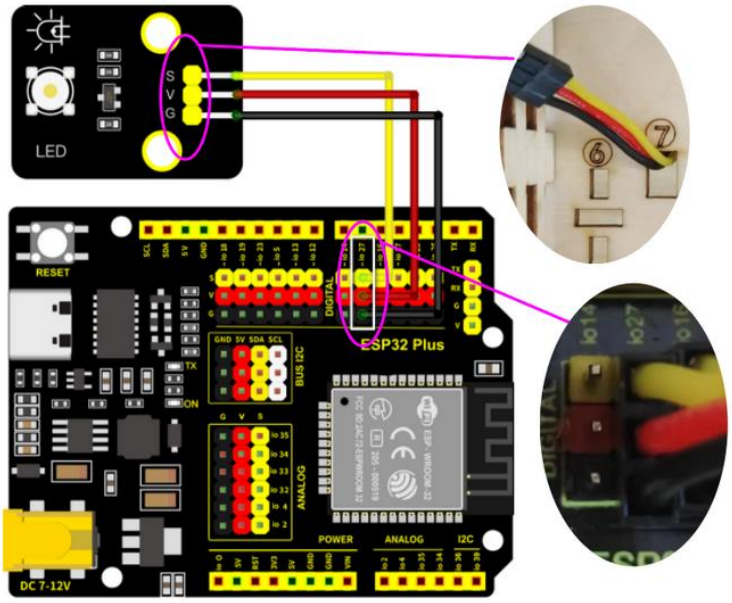


Figure 21: LED wiring diagram for digital output (ESP32 GPIO + current limiting resistor) (Source: https://docs.keyestudio.com)

The concept of digital output is fundamentally different from inputs. While inputs provide information about the system, outputs represent active intervention, as the system not only senses but also affects its environment.

The LED at this stage is not for lighting purposes but rather as a status indicator. It becomes clear to students that the state of the LED is a representation of the internal logic of the program, as the output is always the result of a decision process. Physical feedback plays a key role in understanding and debugging.

When controlling the LED, students experience the entire control chain for the first time:

1. input event (e.g. pressing a button),
2. processing / decision (condition check, status check),
3. output intervention (turning LED on or off).

This chain serves as an abstract model for later, more complex systems (motor control, relaying, automation).

From an educational perspective, LED as a digital output prepares the understanding of the following devices:

- relay modules (switching higher power devices),
- buzzer / sounders,
- traffic lights (red–yellow–green),
- control lines for digital displays.

Students will thus realize that the LED is just a simplified model, behind which lies the same control principle as in the case of industrial actuators. Its application develops:

- cause-and-effect thinking,
- the systems approach (input → processing → output),
- interpretation of visual feedback,
- the ability to abstract, which later allows us to abandon physical devices and understand simulated systems.

At this point, students understand that an automated system "does not think" but reacts according to predefined rules, and that the output in each case is a consequence of the logical state of the program.

Lighting control – connecting input and output

The next step in the project is to connect the pushbutton and the LED with a simple but complete control logic. This task is the first to present the input - processing - output thinking model as a unified system.

The lighting control logic model is made up of the following elements:

- event: change in state of the push button,
- decision: checking the current state of the LED,
- intervention: turn the LED on or off.

This structure already describes a complete automation cycle, even if the implementation is technically simple.

Students should be made aware that the task itself does not require a microcontroller. A pushbutton and an LED can be connected as a simple hardwired circuit, where the pushbutton directly closes the circuit and the LED immediately responds to the physical switching. However, this is not automation, but direct, hardware operation.

The purpose of involving the microcontroller is not to “solve” the task, but to model the entire chain of effects, i.e. digitizing a physical event, making logical decisions in software, and executing programmed interventions. This distinction is key in developing students’ understanding of the system.

Programming elements (in a block-based environment)

During block-based programming, the following elements appear:

- event trigger blocks (button press or state change),
- digital output control blocks (LED on/off),
- optional state variable that stores the current state of the LED.

The introduction of the variable is not a technical constraint, but a conscious design decision that separates the event (what happened) and the state (what is the current state of the system).

Figure 22: Complete structure of a lighting control block program, with variable usage (Source: https://docs.keyestudio.com)

During the task, students quickly realize that the controller does not automatically "remember" previous states, but that preserving the state is a programming responsibility, meaning that the event and the state are concepts that must be treated separately.

This experience is a direct preparation for multi-state automation, timed controls, and networked, asynchronous systems.

Educational reflection

Despite its simplicity, the lighting control project has outstanding educational value. Students experience that digital systems do not think, but execute, meaning that the operation of the system depends entirely on the designed logic, so even a seemingly trivial task is based on conscious engineering decisions.

Comparing a hardwired solution and a controller-based implementation helps students understand that automation is not about the tools, but about the mindset.

This part of the project creates a solid foundation for understanding later sensor-based, condition-based, and networked control systems, and provides a natural transition to more complex IoT solutions.

3.16. Project 2 – Light Control System

This project introduces the measurement and processing of analog quantities for the first time, which is a fundamental difference from previous tasks based solely on digital logic. Here, students no longer work with two-state signals, but interpret and transform continuously changing physical quantities into control decisions.

Light sensor (LDR) operation – analog input

An LDR (Light Dependent Resistor) is a semiconductor-based resistor whose resistance value varies depending on the intensity of the light falling on it:

- strong lighting → lower resistance,
- low light → higher resistance.

This change is continuous, so it is analog in nature. The ESP32 does not "see" the brightness directly, but measures the voltage appearing on the voltage divider circuit formed with the LDR on its analog input.

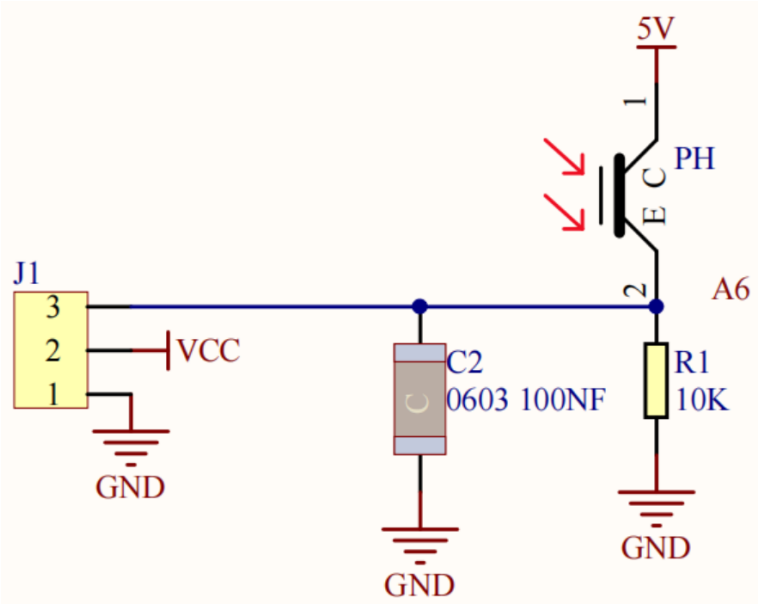


Figure 23: LDR characteristic and voltage divider connection diagram (ESP32 analog input) (Source: https://docs.keyestudio.com)

The analog input:

- does not give a HIGH or LOW value,
- but a measurement result within a numerical range (e.g. 0–4095),
- which is proportional to the intensity of the light.

The students participating in the project are faced with the fact that the measurement value is not absolute, but context-dependent, the accuracy of the sensor and electronics is limited, and the interpretation of the measured data always requires context.

Decision process and intervention – analog processing The

simplest logical structure of a light-controlled system:

1. light intensity measurement (analog input),
2. comparison with a limit value,
3. perform an intervention (turn the lighting on or off).

This model still produces a digital decision based on an analog input.

Figure 24: Conditional control block structure for analog input (Source: https://docs.keyestudio.com)

The educational significance of the analog and digital transition

This step emphasizes the digitization of the analog signal, the problem of choosing the limit value, and the handling of noisy, fluctuating measurement data.

Students will experience that too low or too high a limit can cause unstable operation, so hysteresis or filtering may be necessary, thus establishing that engineering decisions affect system behavior.

Analog output – PWM-based dimming

At the next level of the project, the intervention is no longer binary, but continuous. The brightness of the LED is not just turned on and off, but is regulated proportionally.

PWM (Pulse Width Modulation) is a technique that allows us to achieve an analog effect with a digital output:

- the output turns on and off quickly,
- the proportion of time on (duty cycle) determines the average performance,
- The brightness of the LED is proportional to the fill factor.

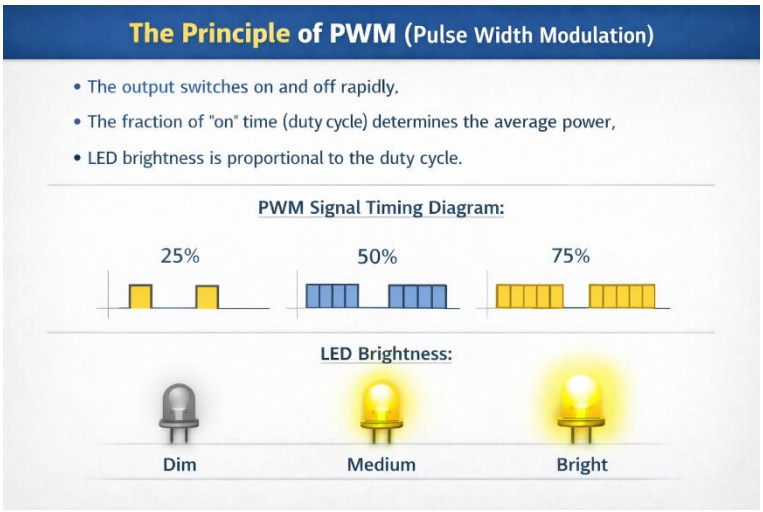


Figure 25: PWM signal timing diagram and LED brightness change (Source: AI generated)

In this project, students will create a proportional relationship:

- input: measured light intensity (analog value),
- processing: scaling, transformation,
- output: LED brightness controlled by PWM signal.

This approach already reflects a process control perspective and is a direct precursor to industrial regulations.

Educational reflection

The light-controlled system project plays a key role in developing students' thinking:

- distinguish between digital and analog quantities,
- understand that the real world is continuous and controllers are discrete,
- recognise the principle of proportional regulation,
- They experience that “better” operation requires more planning.

This part forms a natural bridge to later temperature control, motor control, and energy optimization systems, and is a significant step in deepening engineering thinking.

3.17. Project 3 – Distance Sensing and Automatic Intervention (Smart Feeding System)

The Smart Feeding System is an integrated project in which students combine a time-based sensor and a mechanical actuator into a working automation. It models a simple yet real-life situation: sensing the proximity of an object, a mechanical action is performed.

This fully covers the basic operational chain of automated systems:

The functional purpose of the Smart Feeding System is that the system implements the operation of an automatic feeder that detects the approach of an object (e.g. hand), then makes a decision based on the measured distance and performs a mechanical movement (e.g. opening/closing a lid) using a servo motor.

The function is simple to understand, but at the same time requires complex technical and engineering thinking.

Ultrasonic distance sensor

Its task is to determine the distance to an object using time measurement. The sensor's operation is based on digital control (trigger/echo), and the distance value is calculated.

Educational significance:

- learning about a new measurement principle,
- separation of raw measurement data and interpreted information.

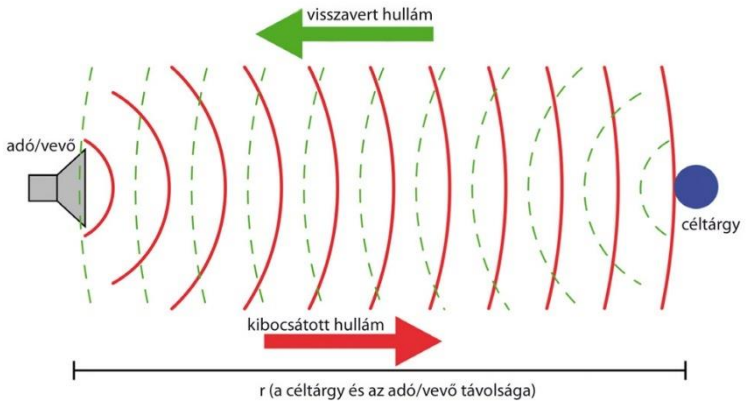


Figure 26: Schematic diagram of ultrasonic distance measurement (Source: https://docs.keyestudio.com)

Servo motor as a mechanical actuator

Servo motor is the control decision physical for its implementation answer.
It can be controlled with a PWM signal, adjusted to specific angular positions, and provides precise and repeatable movement.

Educational significance:

- connecting electronics and mechanics,
- experiencing the direct physical effect of control.

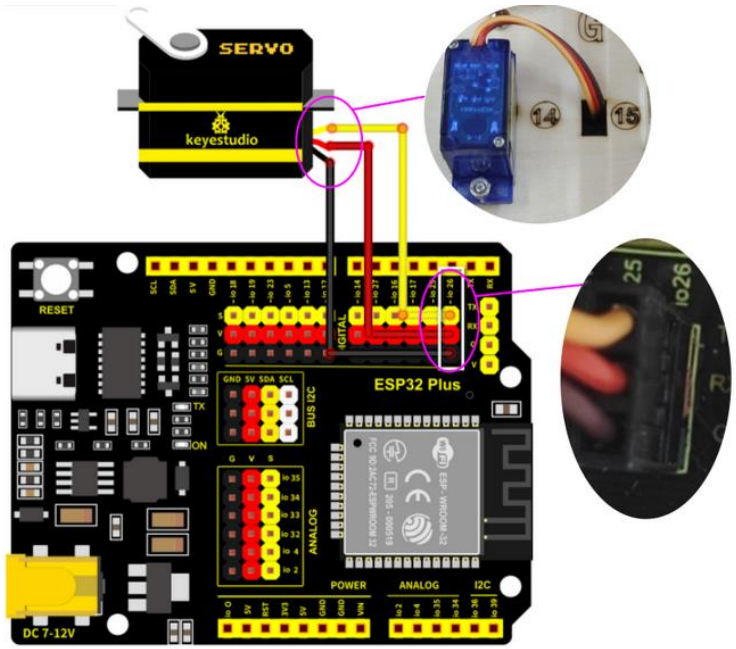


Figure 27: Servo motor structure and control principle (Source: https://docs.keyestudio.com)

Principle of ultrasonic distance measurement

The ultrasonic measurement process:

1. the sensor emits a sound pulse
2. the sound reflects off the object
3. the controller measures the elapsed time
4. the distance can be determined by calculation

This is not an analog voltage measurement, but a time-based measurement based on digital signals, which requires a computational step.

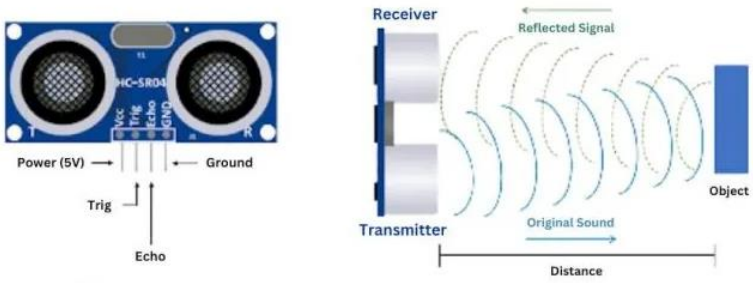


Figure 28: Ultrasonic measurement process (Source: https://docs.keyestudio.com)

To aid the process of processing sensor data and decision logic, the measured distance forms the basis for the control decision. The logic process:

1. measuring distance
2. comparison with limit value
3. making a decision
4. initiating an intervention

This structure already describes a completely automated process.

Figure 29: Processing distance data on a block basis (Source: https://docs.keyestudio.com)

Servo motor control

The servo motor is controlled with a PWM signal, where the pulse width determines the angular position and the control does not control power, but position.

This is a significant difference compared to PWM dimming of LEDs.

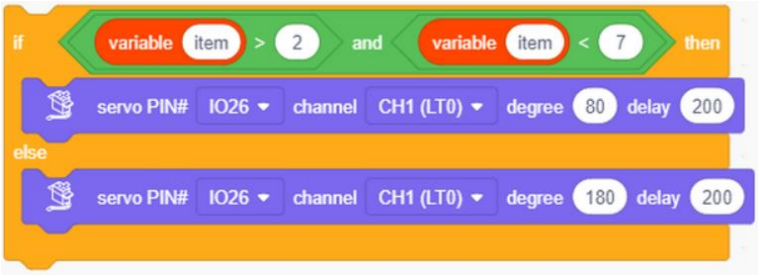


Figure 30: Relationship between PWM signal and servo position (Source: https://docs.keyestudio.com)

Smart Feeding System – integrated operation

The operation of the Smart Feeding System forms a unified logical whole: if the measured distance is within a threshold value, the system makes a decision, the servo motor moves and a mechanical operation occurs.

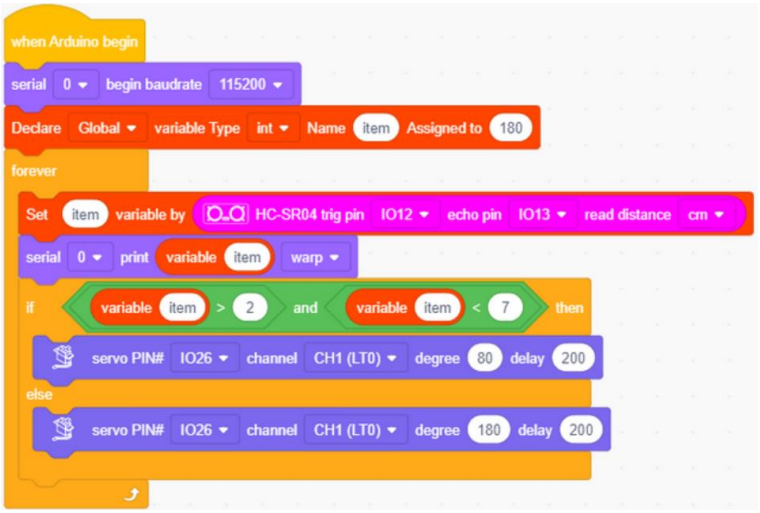


Figure 31: Smart Feeding System complete block program (Source: https://docs.keyestudio.com)

Educational reflection

The Smart Feeding System project is particularly suitable for students to:

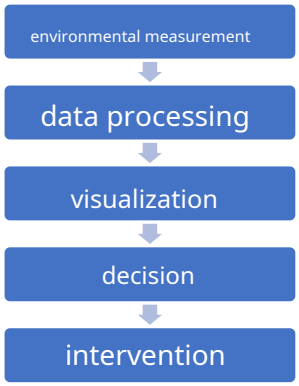
- see the operation of the control at a system level,
- understand the specifics of time-based measurement,
- recognize the relationship between decision logic and mechanical effect,
- develop conscious engineering thinking.

The project clearly shows that the functioning of automated systems lies not in the individual elements, but in their interconnection and the designed logic.

3.18. Project 4 – Temperature Control System

The Temperature Control System models a classic control task that is fundamental in industrial automation and building management systems. The project focuses on the continuous measurement of an environmental quantity, its display, and then the automatic control of an intervention device.

The entire operational chain of the project:



This structure is easy for students to understand, while also reflecting real industrial regulatory logic.

Temperature and humidity sensor (DHT sensor)

The DHT sensor is a digital environmental sensor that measures temperature and relative humidity and transmits the data to the controller in the form of a digital data packet.

Electronic features:

- communicates over a single data line,
- the measurement result is not an analog voltage, but a processed digital value,
- the sensor uses an internal timing and control mechanism.

Educational significance:

- separates analog measurement from digital data communication,
- points out that some of the sensors already provide “processed” data,
- introduces the concept of sampling and update time.

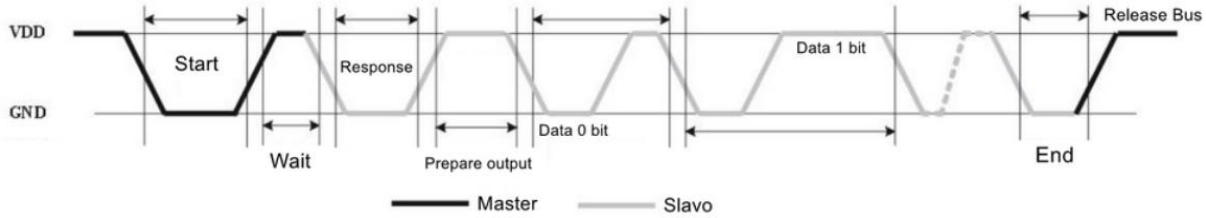


Figure 32: DHT sensor operating principle and data transmission process (Source: https://docs.keyestudio.com)

LCD 1602 display – display of measurement data

An important element of the temperature control system is user feedback. This role is fulfilled by the LCD 1602 display, which displays the measured data in numerical form and allows continuous monitoring of the current status of the system.

Functional role:

- temperature and humidity display,
- operating status feedback (e.g. fan active/inactive),
- display diagnostic information.

Educational significance:

- emphasizes the separation of measurement and display,
- demonstrates that the operation of the system is not "invisible",
- supports debugging and system understanding.

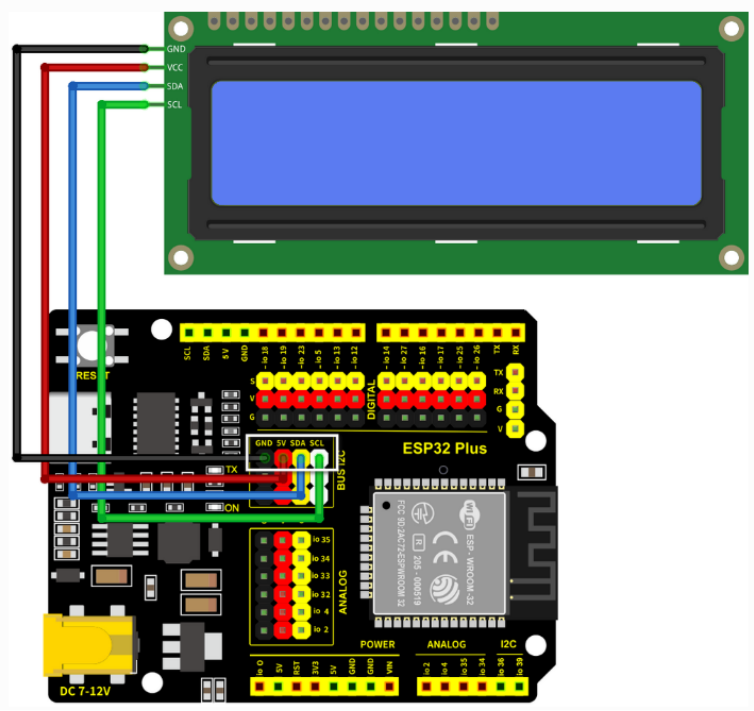


Figure 33: LCD 1602 wiring diagram (Source: https://docs.keyestudio.com)

Fan as an intervention element

The fan is an active actuator in the system, which is used to change the environmental conditions. The controller switches the fan directly or via a driver via a digital output.

Control logic:

- if the measured temperature exceeds a set limit value → the fan turns on,
- if the temperature drops below the limit value → the fan switches off.

Educational significance:

- demonstrates simple two-state control,
- understand the advantages and limitations of limit-based control,
- prepares for the later understanding of hysteresis and PID control.

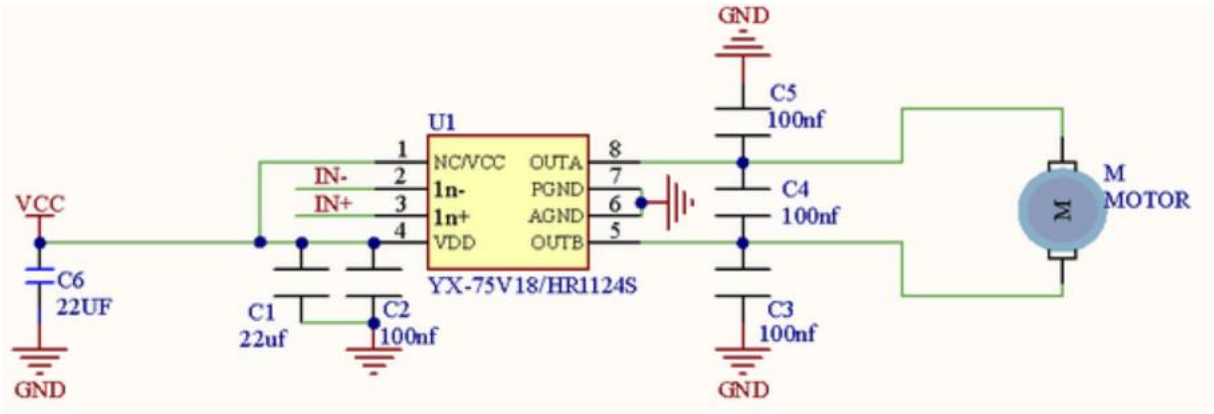


Figure 34: Fan control wiring diagram (Source: https://docs.keyestudio.com)

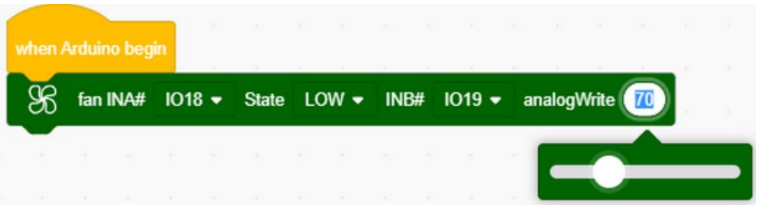


Figure 35: Fan control block logic diagram (Source: https://docs.keyestudio.com)

Integrated operation of the Temperature Control System

The entire Temperature Control System project operates according to the following logic:

1. the DHT sensor performs a measurement
2. the data is processed
3. the measured value is displayed on the LCD display
4. the controller compares the value with the limit value
5. the fan status changes accordingly

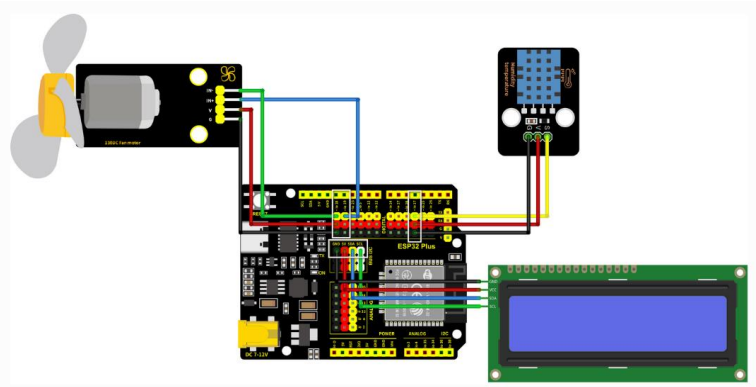


Figure 36: Temperature Control System wiring diagram (Source: https://docs.keyestudio.com)

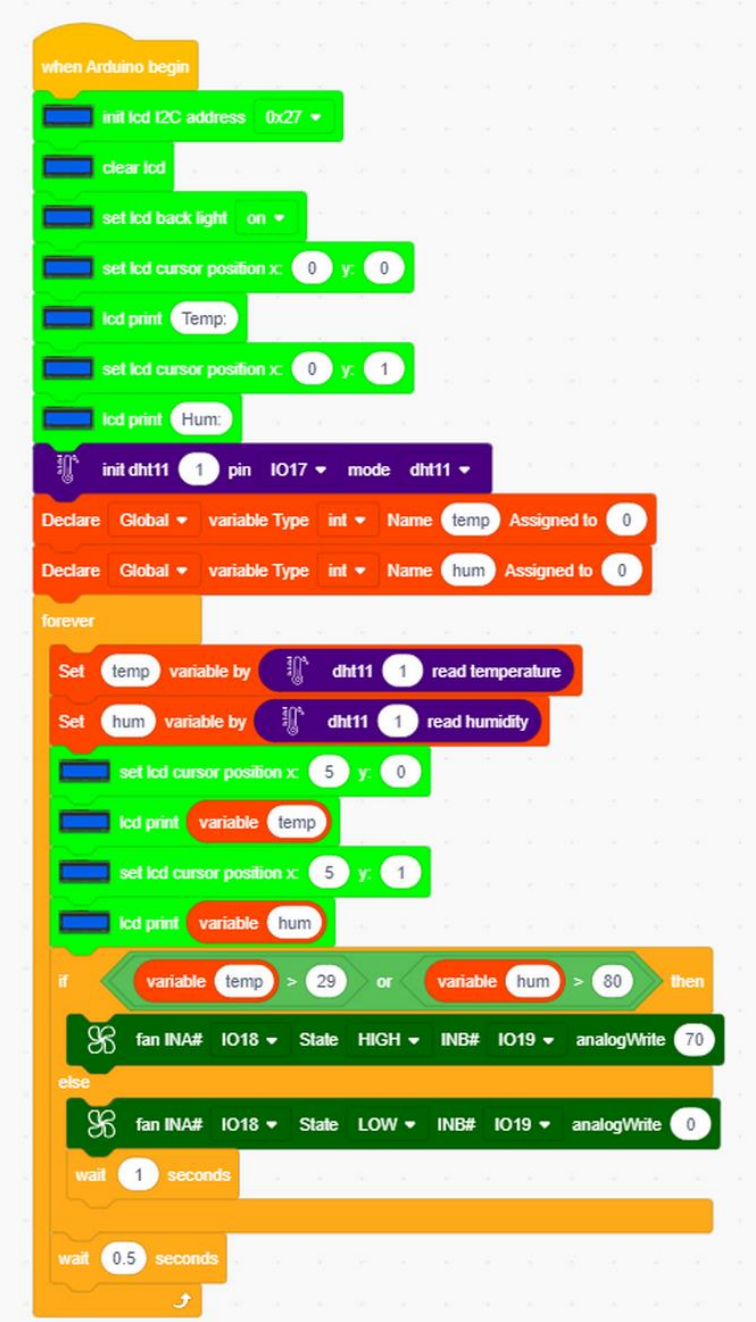


Figure 37: Complete block program of a temperature control system (Source: https://docs.keyestudio.com)

This structure is already a simplified model of a real regulatory system.

Educational reflection

The Temperature Control System project provides students with the opportunity to:

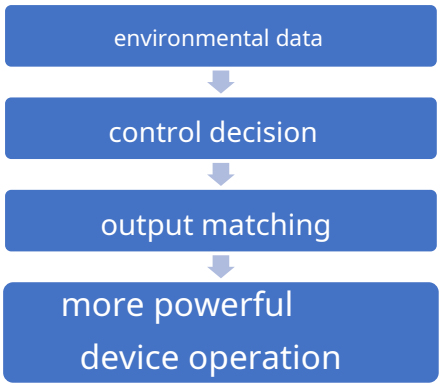
- understand the role of environmental data in automation,
- distinguish between measurement, visualization and intervention,
- recognize the operation and limitations of limit-based regulation,
- think systematically about a seemingly simple task.

This fits well with the smart home and smart farm concept and creates a solid foundation for understanding later, more complex control and networked solutions.

3.19. Project 5 – Automatic Irrigation System

The Auto-Irrigation System project first highlights the engineering problem that the electrical capabilities of the control unit are insufficient to directly drive the actuator. The goal of the task is to create a model of an automated irrigation system in which a safe, industrial-style interface between decision-making and physical execution is achieved.

The project's operational chain:



This structure is fundamental to all real industrial automation systems.

Limitations of microcontroller outputs

The Smart Farm Kit controller is a microcontroller-based system whose digital outputs operate at low voltage levels (typically 3.3 V or 5 V DC) and can only deliver small currents (on the order of a few tens of mA).

This is sufficient for controlling LEDs, smaller electronic circuits, or logic signals.

However, it is not sufficient for directly operating motors, pumps, valves, or mains voltage devices (e.g. 230 V AC).

From an educational perspective, this is a key point: students recognize that control and performance are not the same concept.

Voltage levels and power differences

Automatic irrigation is a good example of multiple voltage levels coexisting in a system:

- control side: low voltage (5 V DC),
- actuator side: higher voltage or power (e.g. 12 V DC, 24 V DC, up to 230 V AC).

During the project, emphasis is placed on why a motor cannot be connected directly to the microcontroller output, why a power matching element is necessary, and how to safely separate the two worlds.

Relay module as a power matching element

The central element of the Auto-Irrigation System is the relay module, which allows a low-power signal from the microcontroller to control a higher-power circuit.

Based on the operating principle of the relay, the controller switches a coil, the magnetic field of the coil mechanically closes or opens contacts, and the contacts operate in a separate circuit.

This implements galvanic isolation, which protects the control electronics, increases the operational reliability of the system, and complies with the principles of industrial automation.

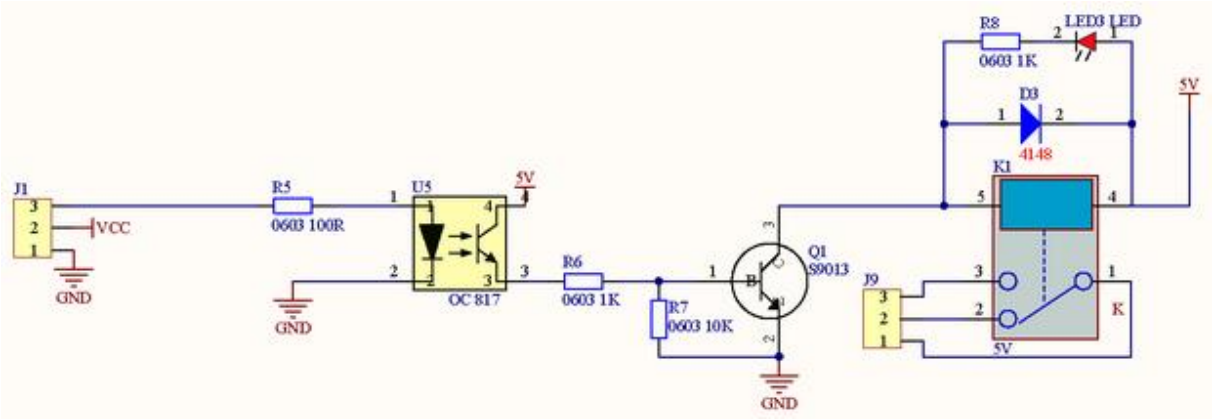


Figure 38: Relay module contacts and operating logic (Source: https://docs.keyestudio.com)

Industrial Parallel – PLC Output Solutions

The use of relays is not a "hobby solution", but an industrial principle. The outputs of programmable logic controllers (PLCs) are also typically:

- relay outputs,
- transistor (PNP/NPN) outputs,
- less often triac outputs for alternating current.

Advantages of relay PLC outputs:

- wide voltage range,
- Switching AC and DC loads,
- electrical isolation.

The Smart Farm project models an industrial mindset at this point, even if the tool used is for educational purposes.

Control logic of the Auto-Irrigation System

The system's operation is simple, but it achieves full automation:

1. the controller evaluates the input conditions
2. compares them with a limit value,
3. decides on the need for irrigation
4. activates the relay via digital output
5. the relay switches the irrigation unit

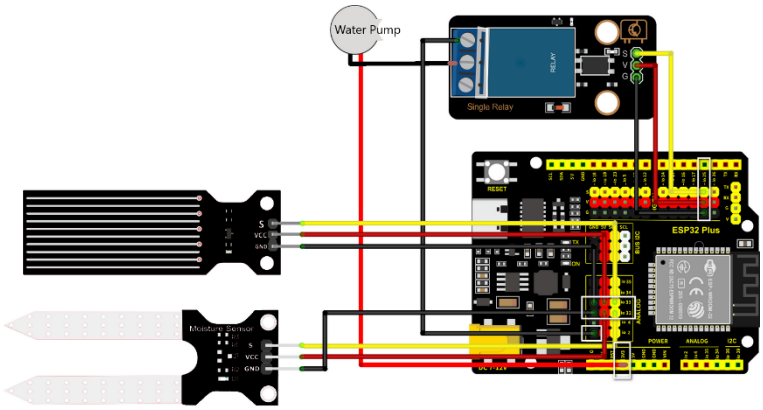


Figure 39: Connecting the Auto-Irrigation System (Source: https://docs.keyestudio.com)

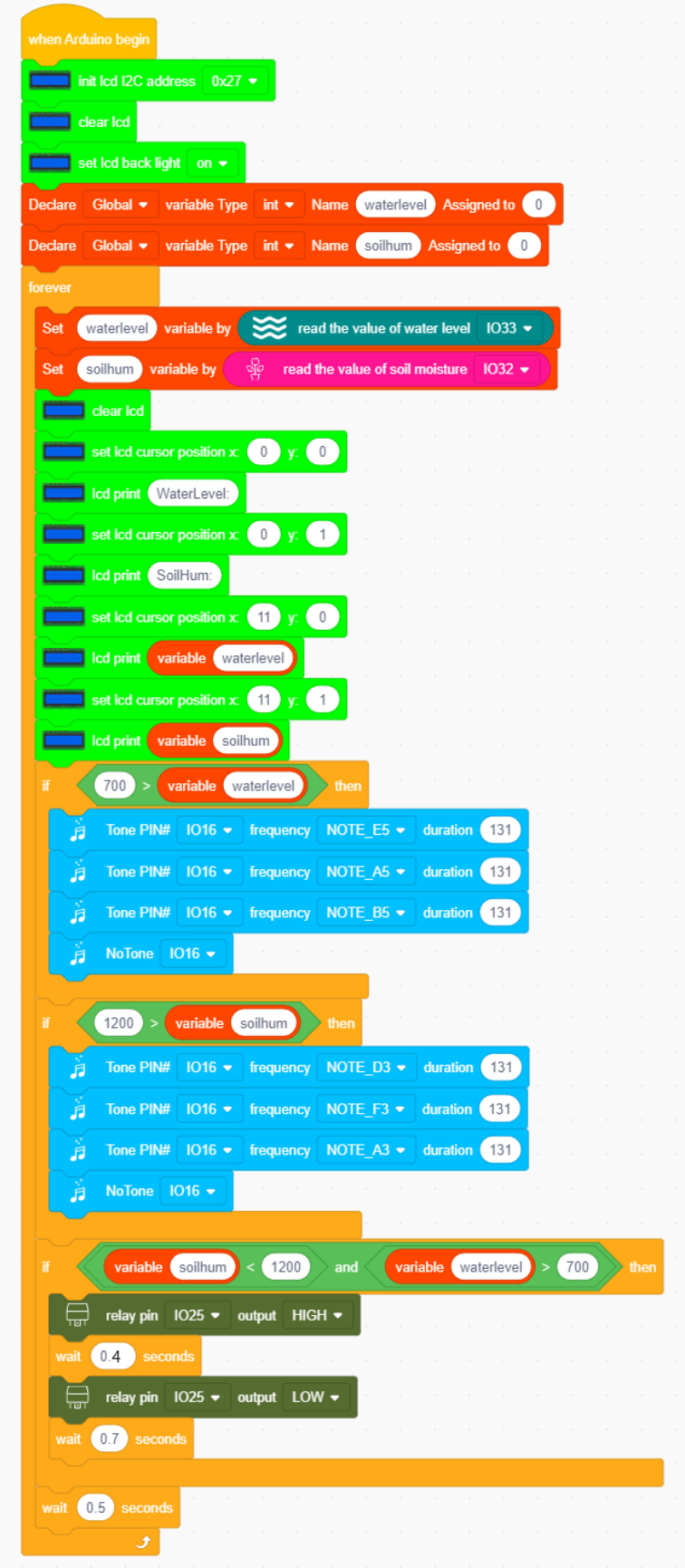


Figure 40: Auto-Irrigation System block-based control (Source: https://docs.keyestudio.com)

This structure can be directly mapped to industrial process control cycles.

Educational and engineering lessons

During the automatic irrigation project, students will:

- understand the physical limitations of microcontrollers,
- recognize the need for performance alignment,
- encounter the concept of galvanic isolation,
- They use solutions analogous to industrial automation principles.

The project is particularly strong in raising awareness of the physical consequences of control decisions, preparing for PLC-based thinking, and laying the foundation for understanding later systems that are networked and use industrial communication.

This chapter forms a natural transition between the world of simple IoT automation and professional industrial control systems.

3.20. Project 6 – WiFi-controlled Smart Farm system

The WiFi-controlled Smart Farm project represents the transition from locally operating automation to networked, remotely monitored systems. At this stage, students no longer just control sensors and actuators, but implement a complete IoT-based monitoring model.

The central idea of the project is that the system:

- connects to the network,
- provides data to a remote client,
- It can be controlled via HMI web interface.

This functionality goes beyond simple embedded systems and brings you closer to real industrial and smart device solutions.

The role of WiFi connection in the IoT system

The Smart Farm Kit ESP32 controller has a built-in WiFi module, which allows direct connection to a wireless network. WiFi connection in this project is not just a technical addition, but a systemic change of perspective: control and monitoring leave the local, wired environment and enter a network context.

By connecting to the network, the system does not operate as an isolated controller, but sensor data can be accessed remotely via a browser, as the interventions are not related to direct physical input, but to network requests.

From an educational perspective, this is where the concept of the Internet of Things (IoT) becomes tangible: the device does not exist in itself, but is an active node in a networked system.

While establishing a WiFi connection, students will be introduced to basic networking concepts such as Access Point. The ESP32 connects to an existing WiFi network (e.g. a school or home router) which acts as an access point, so they understand that the controller connects as a client and the network provides the medium for communication.

An ESP32 connected to the network receives a unique IP address that identifies the device on the network, allows access from a browser, and is the basis for subsequent client management system communication.

This is the first time students experience that a physical device has an “address” on the network, similar to other computers or surveillance systems.

During the project, it becomes clear that the ESP32, the user's computer or mobile device are located on the same network, meaning that communication is only possible if these devices "see each other" at the network level. Control and data retrieval takes place in the form of network messages.

Educational significance

During the introduction of WiFi connection, students:

- they master basic network thinking,
- understand the role of IP-based access in modern systems,
- recognize the difference between local and network control,
- experience one of the basic operating conditions of IoT systems.

This knowledge directly prepares you for:

- understanding the web management interface,
- the processing of subsequent client-supervision system architectures,
- and the adoption of a network approach to industrial and distributed systems.

Embedded web monitoring system operation

In the factory Smart Farm solution, the web management interface is implemented using a web management system running directly on the ESP32. This means that the controller itself provides the website, so there is no external management system or computer in the system, but the client (browser) connects directly to the ESP32.

This solution is extremely intuitive from an educational perspective because it appears within a single tool:

- data collection,
- the processing,
- the display,
- the control.

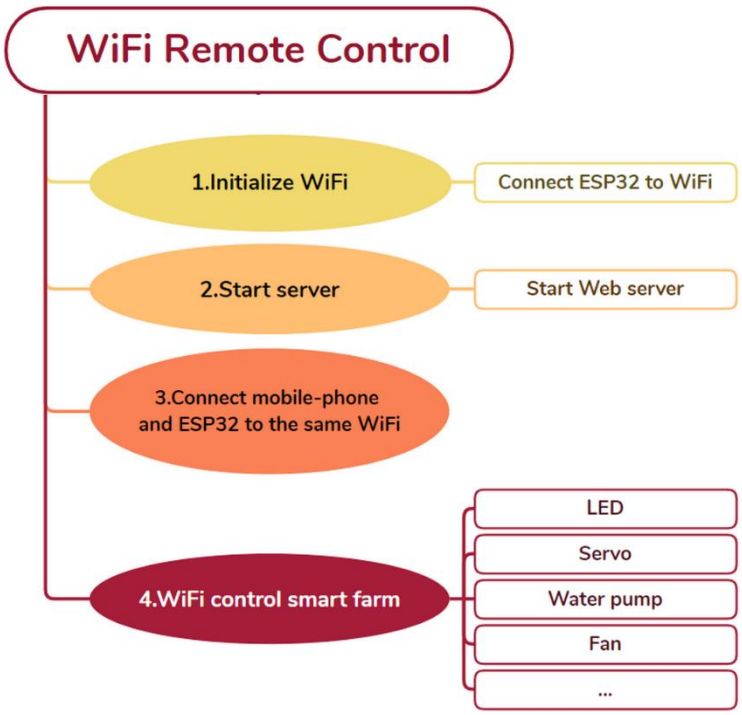


Figure 41: Communication model of a web monitoring system running on ESP32 (Source: https://docs.keyestudio.com)

The web management page is the system's "window" to the user. Based on the factory tutorial, the interface typically implements the following functions:

- display current sensor data,
- feedback of output status,
- providing simple controls (buttons, switches).

The interface is not strong in its graphical complexity, but in its ability to reflect the state of the physical system in real time.

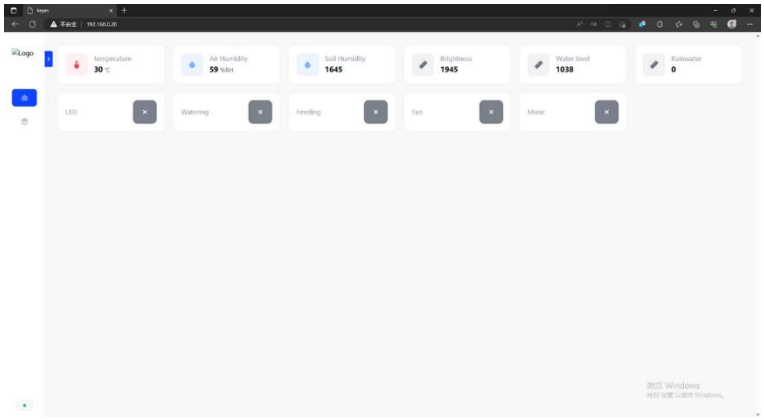


Figure 42: Smart Farm web management page appearance (Source: https://docs.keyestudio.com)

An important educational element is to recognize that a web monitoring system does not run on unlimited resources. In the case of ESP32, students are faced with the following factors:

- limited memory,
- limited storage space,
- managing a limited number of simultaneous connections,

- using simple HTML structures.

This results in the website's functionality being deliberately simple, so the amount of data displayed is limited, meaning that the ESP32 is not suitable for running complex web applications.

This realization is a key engineering lesson: all systems are built around compromises.

The control elements displayed on the HMI web interface allow the user to send a command to the system, the controller interprets this as a digital instruction, and the actuators respond to the signal coming from the network.

This operation demonstrates the basic principle of modern IoT systems, where the user is not physically present, yet control is immediate and feedback-driven.

During the WiFi-controlled Smart Farm project, students will:

- understand the essence of network-based control,
- recognize the resource limitations of embedded systems,
- experience the basics of the client-supervision system model,
- they separate control and display tasks.

The strength of the project is that:

- provides a real IoT experience with minimal device requirements,
- clearly points out the limits of factory solutions,
- It provides a natural preparation for understanding later, external, systematic, industrial architectures.

This chapter concludes the mapping of the possibilities of the factory Smart Farm system and also lays the foundation for the development of your own system architecture, separated by resources and following an industrial pattern.

3.21. Summary pedagogical interpretation

The Smart Farm project part, based on block-based programming, does not appear as a stand-alone learning objective, but as a consciously planned pedagogical transition throughout the project. In this phase, students acquire basic concepts, relationships, and operating principles that are later essential for understanding and developing more complex, industrial-type systems.

To promote systems thinking, students consistently work with the same basic model throughout the project:

This chain of influence appears in every project part:

- digital inputs (e.g. pushbutton, switch),
- analog inputs (e.g. LDR, potentiometer),
- time-based measurements (ultrasonic sensor),
- digital and PWM-based outputs (LED, fan, servo motor),
- relayed power switching (irrigation system),
- network communication and web control (WiFi).

Students gradually realize that individual elements cannot be interpreted in isolation, but are functional parts of a larger system.

The connection between analog and digital worlds

A prominent pedagogical outcome is that students can make a clear distinction between:

- digital signals (bi-state, event-driven),
- analog quantities (continuous values),
- the role of analog-to-digital conversion,
- between limit values, proportional control and PWM modulation.

The concept of measurement noise, the importance of choosing the decision threshold, and the "measurement ≠ information" approach appear during light control, temperature control, and fan control.

Due to the understanding of intervention and power management, when using the relay module and motorized actuators, students are faced with the fact that the outputs of microcontrollers are capable of limited current and power delivery, therefore control and power transmission are separated, so that low-voltage logic signals (3.3–5 V DC) can control higher-power circuits (e.g. 12 V DC, 230 V AC).

This creates a direct parallel to the operation of industrial controls, where PLCs control external actuators through relay, transistor, or semiconductor outputs.

Network thinking and IoT perspective

In the WiFi-driven project, students will experience:

- the basics of the client-supervision system model,
- the operation of the embedded web monitoring system,
- the impact of physical resource constraints (memory, storage) on system design,
- the possibilities and limitations of remote monitoring and control.

This experience prepares us to recognize that in an industrial environment, tasks are often separated (field node controller – central monitoring – backend services).

Block-based programming helps prepare for a change in programming approach, which in this context:

- decouples control logic from syntactic difficulties,
- visualize state management, events and conditions,
- can be easily adapted to industrial programming paradigms (e.g. IEC 61131-3, function blocks, state machines).

Students are not "learning Scratch", but rather learning control thinking, which can later be applied in C-based microcontroller programming, industrial PLCs, Modbus-based communication, and distributed system architectures.

Summary

This project phase creates a stable foundation for the development of self-designed hardware elements, the transition to text-based programming, the understanding of industrial-oriented, distributed architectures, and the subsequent development of systems with a monitoring client-field node, backend-frontend structure.

The block-based environment is therefore not a terminal station, but a consciously constructed learning ladder that gradually leads students towards addressing real engineering and industrial problems.

3.22. Practical implementation of the chapter

Assembling and commissioning the Smart Farm Kit was the first truly “hands-on” experience for the students. Observing the factory kit in operation gave a quick sense of success, but it soon became clear that behind the seemingly simple automations lie complex logical and electronic connections.

Processing the factory documentation, especially interpreting foreign language descriptions, was a serious challenge. During the partial Hungarian translation and educational adaptation, the students actively engaged in the content interpretation, which significantly deepened their understanding. Block-based programming seemed easy at first, but state management and condition-based control required more serious thinking.

4. Digital design of your own smart home / smart farm model (Fusion 360)

This chapter is a defining milestone of the project: it is here that the previously learned sensor, control and IoT logic becomes a concrete, tangible physical structure. At this stage, students move from the world of electronics and software thinking to the field of engineering design, where every decision has geometric, assembly and manufacturing technological consequences.

The goal of this chapter is not simply to create a 3D model, but to understand that a physical model intended for 3D printing is a supporting structure: it must provide space for the controller, sensors, actuators, and cabling, while supporting assembly, measurement activities, and later modifications. The model is in this sense a learning tool, not a final product.

The learning path is deliberately structured in several steps. The digital design is preceded by a physical concept phase, which analyzes the factory design of the Smart Farm Kit,

It is based on identifying functional elements and recognizing the limitations of additive manufacturing. This is followed by learning about the Fusion 360 design environment, then developing a parametric, component-based digital model, all the way to a production-ready state.

During the process, students experience:

- Good design doesn't start with software,
- Form is always a consequence of function,
- parametric thinking enables iteration and redesign,
- The digital model acts as a “virtual prototype” to help prevent manufacturing and assembly problems.

This part thus forms a bridge between electronic-informatics system understanding and physical implementation, and provides a solid foundation for the later stages of the project dealing with 3D printing and system integration.

4.1. Starting point: analysis of the physical design of the Smart Farm Kit

The first step in the digital design process is not to use CAD software, but to consciously analyze existing physical solutions. The Smart Farm Kit serves as a reference system in this project: a starting point without which it is impossible to create a well-functioning, educationally optimized model of your own.

The physical design of the Smart Farm Kit is presented not as a model to be copied, but as an example to be analyzed. The students' goal is to explore:

- which elements carry an actual function,
- how electronic components are connected to physical space,
- which solutions serve educational purposes and which were developed for production technology or aesthetic considerations.

This approach establishes the basic principle of function-driven design at this early stage: form is a consequence of function, not its starting point. Students learn to critically examine ready-made solutions and distinguish necessary elements from contextual or optional details.

The basis for conceptualization is to get to know the real dimensions. Students examine factory elements with simple tools that are accessible to everyone. Using a caliper, they determine the surrounding dimensions, then make proportional sketches on graph paper and record the locations of openings, mounting points and sensor positions.

From a pedagogical point of view, it is important that standard technical drawings are not produced in this phase, as the primary goal is not formal accuracy, but rather an understanding of spatial relationships and proportions.

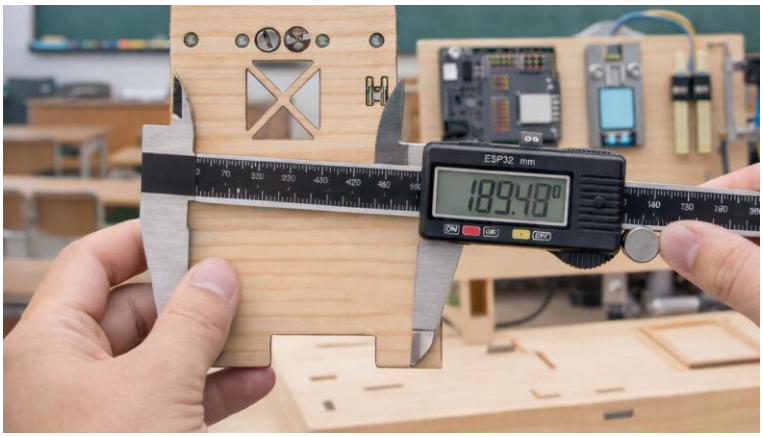


Figure 43: Measuring the physical elements of Smart Farm (Source: AI generated)

This step directly prepares for later parametric thinking while reinforcing the connection between physical reality and abstract design.

4.2. 3D Printing Limitations and Design Implications

While analyzing factory components, students will be confronted with the practical limitations of additive manufacturing. They will recognize that some geometric details cannot be reproduced with desktop 3D printing, as thin walls, closed cavities, and undercuts are problematic, requiring conscious simplification rather than full geometric fidelity.

This realization introduces the design for manufacturing approach and highlights that design decisions are always related to the chosen manufacturing technology.

The next step in physical analysis is conscious decision-making: what to keep and what to leave out when developing your own model.

Students jointly determine which elements are needed to place sensors, actuators and controllers, where to provide attachment points for them and which parts can be omitted without compromising the learning objectives.

Decisions are always based on function and trainability, not formal similarity to the factory solution. The goal of conceptualization is to create a physical structure that:

- does not contain unnecessary blank spaces,
- material-saving and fast to print,
- easy to install and modify,
- provides a transparent educational environment.

This step has a direct impact on print time, material usage, and the speed of iteration cycles.

4.3. Physical concept as system description

At this stage, students no longer interpret the model as a “house” or “box,” but as a system of points for placing sensors, a structure providing a range of motion for actuators, and a carrier for control and cabling.

This approach helps ensure that the design does not drown in formal details too early, but remains interpretable at a systemic level.

Conceptual preparation for parametric design

Although this chapter does not yet introduce the use of Fusion 360, the basic idea of parametric design is introduced. Dimensions are not isolated numbers, but interrelated parameters, where a modification can affect multiple elements and where the goal of design is adaptability and re-designability.

This mindset forms a direct transition to the next part, where the physical concept is transformed into a digital, parametric model.

Pedagogical summary

This preparatory and conceptualization phase ensures that subsequent Fusion 360-based design is not a mere software usage exercise, but a digital representation of conscious engineering decisions.

Students will experience that good design begins before the computer is turned on, that simplification and abstraction are engineering virtues, and that the physical and digital worlds form a close unity.

4.4. Fusion 360 design environment – basics and approach

In this stage, students no longer talk about digital design at a theoretical level, but experience through concrete operations how a physical concept becomes mappable with a digital, modifiable model.

Students will learn the basics of Fusion 360 and its use using Autodesk's official publication, "Autodesk Fusion 360 Training: The Future of Making Things Attendee Guide."

Based on the processed educational material, the learning process is not based on learning function lists, but on developing basic design routines, for example:

- how do we launch a new model,
- how we navigate in 3D space,
- how to create simple but stable geometry.

The pedagogical goal is for students to recognize that CAD design is an activity, not the creation of a visual image.

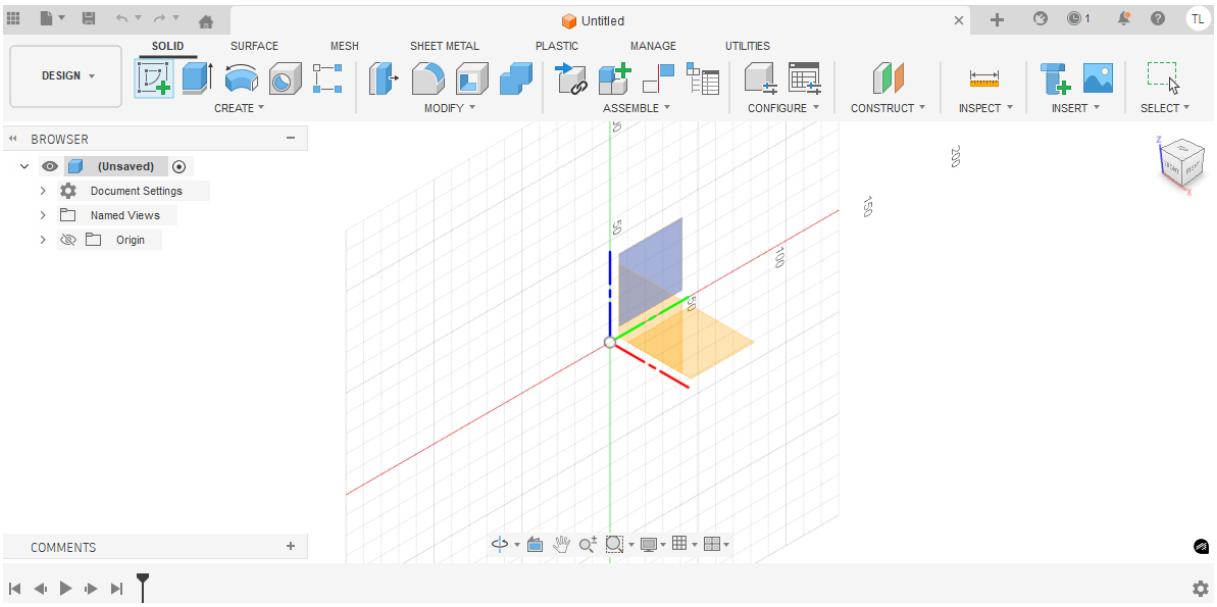


Figure 44: Fusion 360 design environment – basics and approach (Source: Own editing)

Fusion 360 is presented in education as a “thinking surface.” The document emphasizes that even at a beginner level, a full design cycle can be worked through, creating the model, modifying it, reverting to previous states, and observing the impact of changes. This approach introduces them to the world of parametric modeling and story-based design, where the model is not a static object, but the result of a process.

Specific student activities:

- Create a New Design
- Save to project (understanding cloud-based operation)
- Revert to a previous state in the Timeline

Key terms from the document:

- Parametric Modeling
- History-Based Design
- Non-destructive Editing

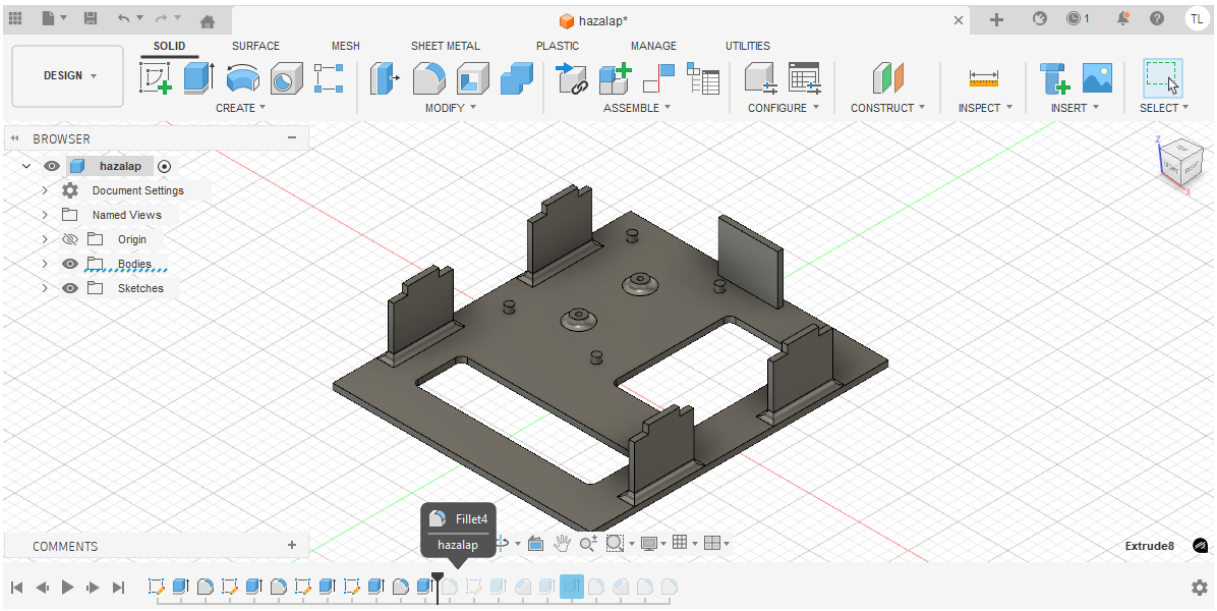


Figure 45: Creating a new Design – smart home base (Source: Own editing)

Pedagogical focus: students experience that the history of the model is just as important as its current state.

The Autodesk Fusion 360 Training material introduces the interface specifically for beginners, based on usage logic rather than functions.

4.5. Canvas – mastering spatial movement

The development of spatial thinking is given a special role. Students actively use view manipulation operations – rotating, zooming, moving – as well as basic views to confidently navigate in space. This skill is a prerequisite for later creating accurate and thoughtful models.

Student exercises:

- model rotation (Orbit),
- Zoom in/out,
- Pan,
- using basic views (Top, Front, Right).

Pedagogical goal: to develop spatial thinking even before drawing.

Browser – the “internal map” of the model

The "internal map" of the system, the Browser, plays an important role in understanding the structure of the model. Students learn to manage sketches, bodies, and components, and realize that a well-structured model is not only more transparent, but also easier to modify later.

Student activities:

- Hide/show Sketch,
- Separation of bodies,

- naming elements.

Key terms:

- Sketch Folder
- Bodies Folder
- Components

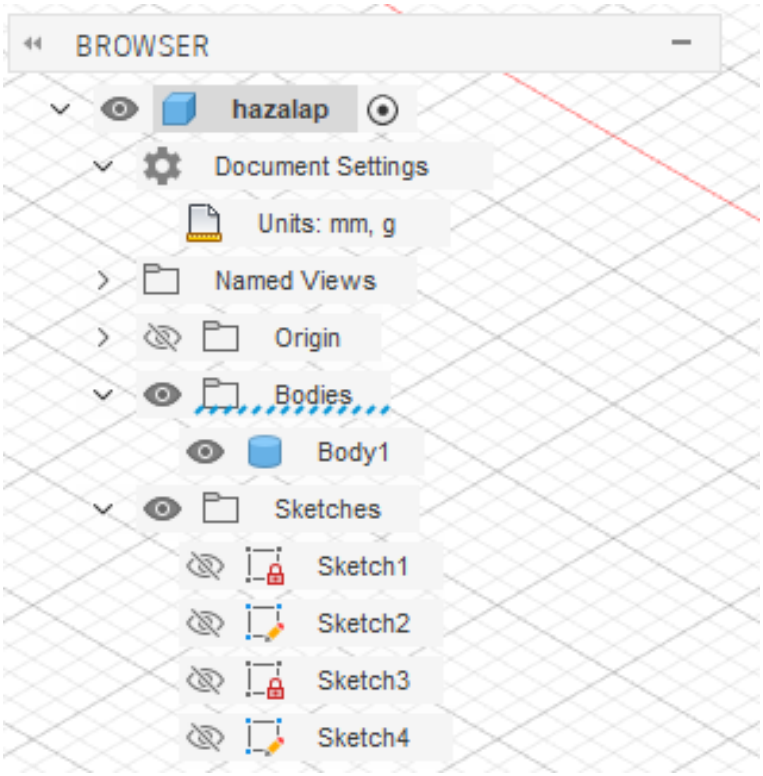


Figure 46: "Internal map" of the model (Source: Own editing)

In parallel, the use of Timeline makes the causal system of the design process visible: modifying an earlier step affects the entire model, which helps develop systems thinking.

The document makes strong use of Timeline as a learning tool.

Student practice:

- modify an Extrude step,
- observing consequences on the entire model.

Keyword: Design History

Figure 47: Using Timeline (Source: Own edit)

4.6. Basic geometric and design concepts – in a tangible way

Sketch – not a drawing, but a set of rules

It is important to understand the concept of Sketch, which is not a drawing but a set of rules. Students experience that a sketch is not a simple drawing, but a mathematically defined system. The use of dimensions and geometric constraints ensures that the model is stable and unambiguous. The problems of "incompletely defined" sketches quickly become visible, so students understand the importance of accurate design from their own experience.

According to Autodesk Fusion 360 Training, the most common mistake made by beginning designers is an undefined sketch.

Student practice:

- drawing a rectangle,
- adding sizing,
- application of constraints.

Key terms:

- Fully Constrained Sketch
- Degrees of Freedom
- Geometric Constraints

Pedagogical insight: what is not defined will cause problems later.

Body and Component – when to use which?

The distinction between solids and components also emerges as an important engineering decision. Students recognize that while a solid is suitable for quick design, the use of components requires structural thinking, especially for more complex models. This distinction also prepares them for later assembly tasks.

The document shows through practical examples:

- Body = fast shaping,
- Component = structural thinking.

Student decision situation:

- when one body is enough,
- when a separate component is justified.

Key concept: Create Component from Body

Parameters as design tools

During the introduction to parametric design, students experience that modifying a single dimension can affect the entire model. This represents not only a technical but also a change in perspective: the model is no longer a fixed form, but a changeable system that adapts to design needs.

Based on Autodesk Fusion 360 Training, students will try:

- changing a size,
- the transformation of the entire model.

Key terms:

- Parameters
- Change Parameters Dialog

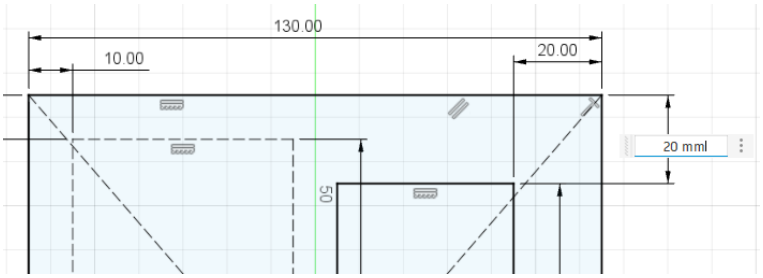


Figure 48: Using parameters (Source: My own edit)

Body modeling – basic operations with a sure hand

Basic solid modeling operations, such as extruding, rounding, or repeating, are not presented in isolation, but are always linked to specific design situations. This way, students do not learn commands, but understand their role and consequences for the model as a whole.

Specific, teachable operations based on Autodesk Fusion 360 Training:

- Extrude (Join / Cut),
- Fillet and Chamfer,
- Pattern (Linear repetition).

Key terms:

- Join vs. Cut
- Edge Fillet
- Rectangular Pattern

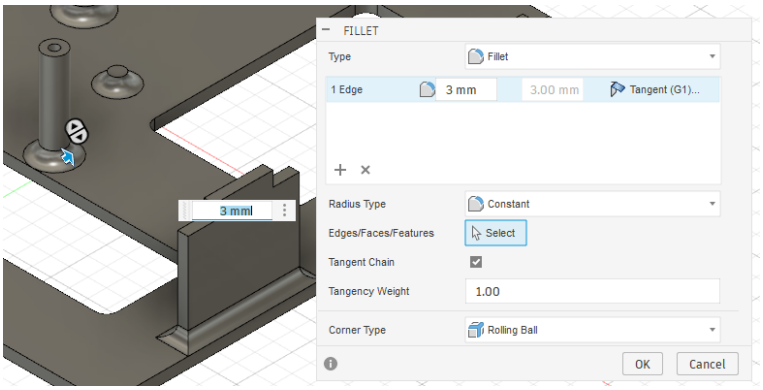


Figure 49: Body modeling (Source: Own editing)

4.7. Preparing for 3D printing – final practical focus

Pages 62–65 of Autodesk Fusion 360 Training focus specifically on the “finished model” state.

In the final stage of the design process, the focus shifts to the physical feasibility of the digital model. Students learn that a well-designed model is not necessarily suitable for automatic manufacturing, so special attention must be paid to the requirements of 3D printing.

When checking the model, the primary consideration is to ensure closed geometry, as printing is only possible with clearly defined volumes. In addition, wall thickness, detail size, and geometry stability must be examined. Students will experience how design decisions directly impact the quality and success of manufacturing.

Exporting the model introduces the formats required for manufacturing (such as STL or 3MF) and the process of creating a printable data file from the CAD model. This step closes the design cycle and also provides feedback: design errors and deficiencies become immediately visible during the creation of the physical model.

Student checklist:

- closed body (watertight),
- appropriate wall thickness,
- export in STL/3MF format.

Key terms:

- Mesh Preview
- Export for Manufacturing

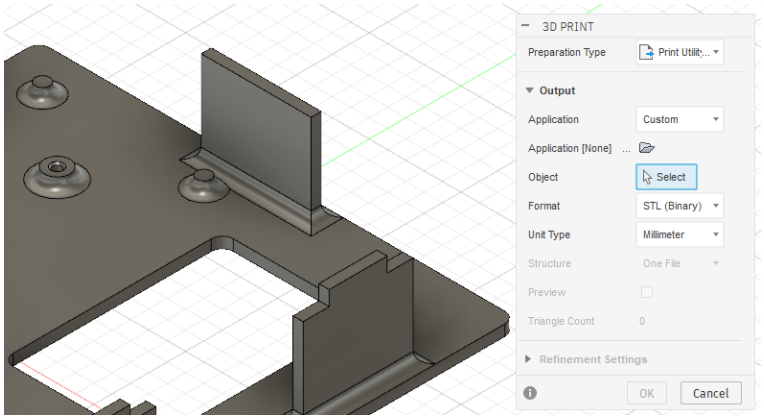


Figure 50: Preparation for 3D printing (Source: Own editing)

Didactic summary

This stage is particularly important from a didactic point of view, as it becomes clear to students that digital design is not an activity in itself, but the preparation of systems to be implemented in the real physical world.

By the end of the entire process, students will not only create a model, but also understand the full cycle of design: from idea to parametric model to manufacturable form. This experience establishes the engineering approach in which design and implementation form an inseparable unit.

In this part of the project implementation, students do not learn to use a software program, but rather begin to design, build their first digital model step by step, and realize that CAD is the external representation of thinking.

This creates a solid foundation for the next phase, where the digital model becomes a real physical object through 3D printing.

4.8. Results of the digital design process – presentation and interpretation of student models

This subsection presents the digital models created during the project. These elements are not prefabricated samples, but 3D models that the students created based on previously learned design principles and with teacher guidance.

During the creation of the models, the students consciously applied the principle of function-driven design, parametric thinking, the aspects of modularity and assembly, and the geometric limitations of additive manufacturing (3D printing).

From a pedagogical perspective, these models are tangible imprints of the learning process: they show how abstract design principles manifest in concrete physical forms.

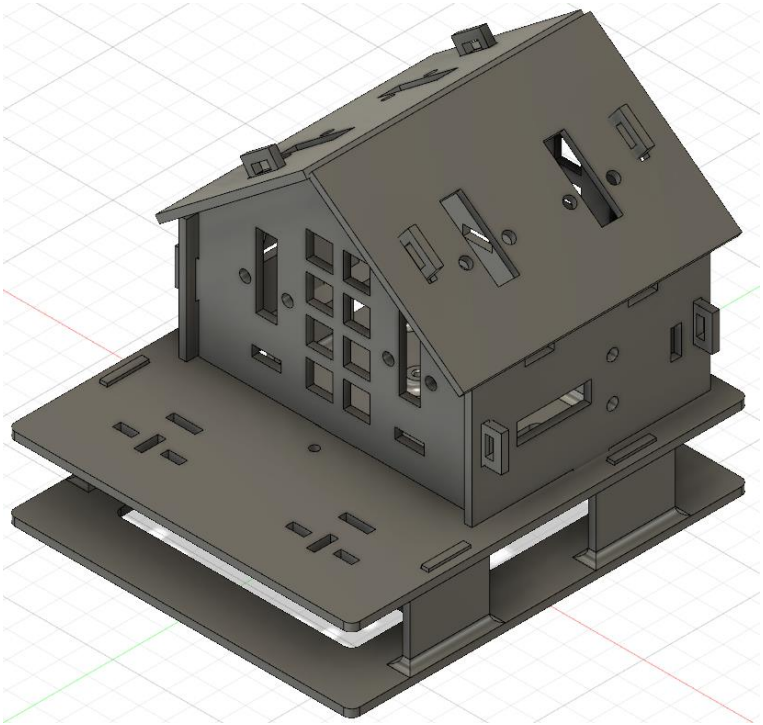


Figure 51: Smart home assembly model (Source: Own editing)

Together, the STL files created by the students form the physical basis of a modular smart home/smart farm model. The design goal was not to create an aesthetically final object, but to create a supporting structure that:

- suitable for accommodating electronic components,
- supports the recording of sensors and actuators,
- allows assembly, measurement and modification,
- meets the requirements of desktop 3D printing.

It thus became clear to the students that the digital model is not an independent goal, but a physical framework for electronics and control projects.

Baseplate and lower structural elements – student design decisions

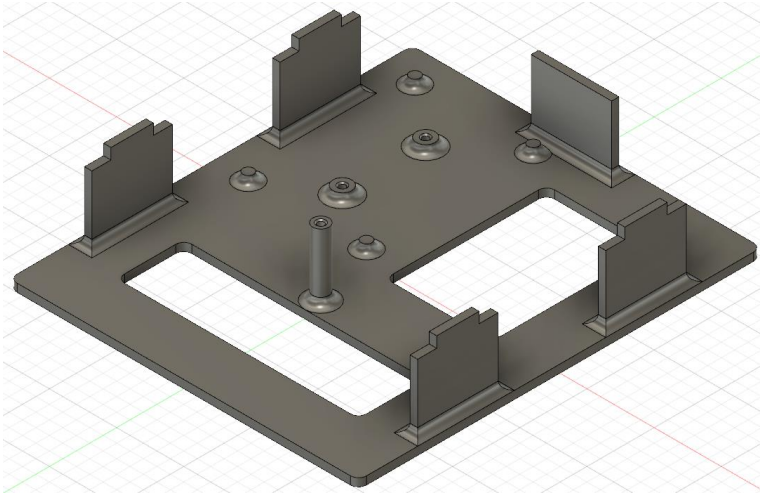


Figure 52: Base plate and lower structural elements (Source: Own editing)

When designing the base plate and the bottom cover, the students applied the engineering aspects they had learned earlier.

Decisions made in the design process:

- creating a stable, flat support surface,
- preparation for mounting electronic modules,
- ensuring orderly cable routing.

Didactic significance: students experienced that the quality of the basic structure determines the usability of the entire system, and that even seemingly "simple" elements have conscious engineering thinking behind them.

Side panels – function and accessibility

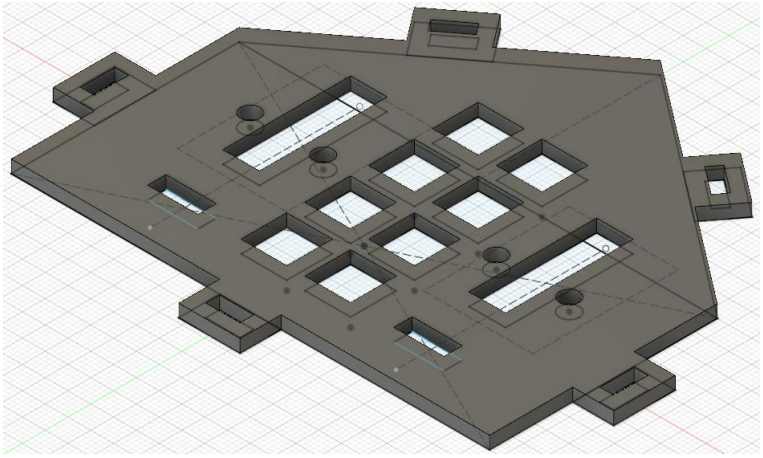


Figure 53: Side panels (Source: Own edit)

When designing the side panels, the students consciously aimed to ensure that the model was not a closed enclosure, but a "readable" structure that supported educational goals.

The design considerations included ensuring the location of sensors and displays, access for assembly and measurement, and visual transparency of the interior.

Pedagogical lesson: the students recognized that the cover of the educational model does not hide, but makes the operation understandable.

Roof elements – balance of protection and dismountability

When designing the roof elements, the students used solutions that protect the electronics while not hindering disassembly and supporting later modifications.

This design task clearly highlighted that “closure” does not mean finality, but rather a conscious compromise between protection and accessibility.

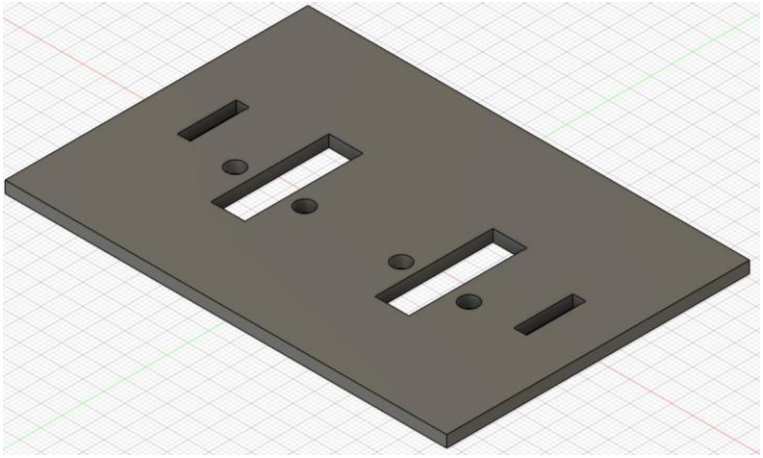


Figure 54: Roof element (Source: Own editing)

4.9. Summary of learning outcomes based on completed models

The models created during the project provide students with the opportunity to look back on their own design decisions and recognize their consequences.

Key concepts captured through the models:

- parametric design,
- modularity,
- installation,
- design for manufacturing,
- iterative development.

This section makes it clear that students have not only learned software usage, but also an engineering approach, which will be completed in the following chapters with 3D printing and physical assembly.

4.10. Practical implementation of the chapter

It was at this stage that it became really clear how difficult it was for students to accept that digital design is based on an accurate understanding of physical reality. At first, the analysis of measurements, proportional sketches and spatial relationships seemed slow and “less spectacular” to them, but later they themselves realized that inaccuracies in the model and in the print immediately reflected back.

When introducing Autodesk Fusion, it was often necessary to deliberately slow down the process. Students wanted to draw while having to learn parametric thinking. The turning point was when they understood from their own experience that a poorly structured sketch later causes serious problems, while a well-thought-out structure results in a stable, easily modifiable model.

When designing our own model, real engineering decisions were made: what to simplify, where to provide access, what counts as a real function. Overcrowding was common, which had to be contained together. From a teacher's perspective, one of the biggest results was that the students' thinking changed: they began to view the model not as an object, but as a functioning system.

5. 3D printing and manufacturing experience

A key element of the project was the physical implementation of the digital design processes. The use of 3D printing provided the opportunity for students to gain direct experience of the connections between computer modeling and real-world manufacturing, and to understand the importance of the principles of design for manufacturing. 3D printing, which is popular today and has a constantly growing range of applications, was in many ways an ideal choice for the physical implementation of this phase of the project.

During the production phase, it became clear that solutions that seemed right in the digital space often needed modification during physical implementation. This experience was particularly valuable from an educational perspective, as students encountered problems, their analysis and solution not as isolated tasks, but as part of a coherent development process.

5.1. Introduction to the 3D printer and Bambu Studio

5.2. Brief introduction to the process used

The additive manufacturing process used in the project was FDM (Fused Deposition Modeling) technology, which is one of the most widespread and easily accessible 3D printing methods today. The essence of the process is that the printer melts a continuous plastic thread (filament) and then deposits it layer by layer along predetermined paths.

FDM technology is particularly well suited to educational environments, as the steps of the manufacturing process can be easily tracked and the effects of changing printing parameters can be immediately seen. This has created an opportunity for students to not only observe the final result, but also to understand and interpret the entire manufacturing process.

5.3. Hardware structure and operation

Print jobs are sent to a**Bambu Lab X1 Carbon**We implemented the project with a closed-type desktop 3D printer. The device has an advanced sensor system, automatic table leveling and a high-precision motion system, which together ensure stable and reliable operation. We chose this printer during the implementation of the project because we were looking for a device among the alternatives available on the market that could not only produce objects in high quality, with numerous calibration and customization options, but also do it in a time-efficient manner. Using the printer required significantly less background calibration processes than other devices, which allowed us to spend additional time on more important areas of the project. We also selected a Prusa i3 MK3S+ printer as an alternative, but we ultimately decided on the X1 Carbon due to its easier handling and professional design.

Figure 55: Bambu Lab X1 Carbon 3D printer (Source: https://www.3djake.hu/bambu-lab/x1c)

The enclosed printing space played a particularly important role in the project, as it reduced the occurrence of printing errors by minimizing environmental influences (temperature, air movement). This allowed the students to focus on the technological context and not let the problems arising from environmental instability dominate their experience.

5.4. The AMS filament feeding system

Figure 56: AMS filament dispenser (Source: https://www.3djake.hu/bambu-lab/x1c)

The AMS (Automatic Material System) automatic filament feeding system connected to the printer is capable of handling multiple rolls of material at the same time. Although the project mainly produced parts made of one color and one type of material, using the AMS system provided important technological knowledge. It also facilitated the production process, as there was no need to remove and insert the filament into the printer when replacing empty rolls or changing colors.

The students learned about the importance of material handling and the condition of raw materials, with particular attention to the effect of moisture and the importance of storage, highlighting the passive and active AMS

filament drying devices. In addition, the operation of AMS provided a good starting point for the later introduction of multi-material and multi-color manufacturing capabilities.

5.5. Slicing principles

The conversion of digital models into manufacturable forms is**Bamboo Studio**We did this using slicing software. Bambu Studio is the printer manufacturer's (Bambu Lab) own, open source software, which is based on the PrusaSlicer slicing program and is available for all major desktop operating systems.

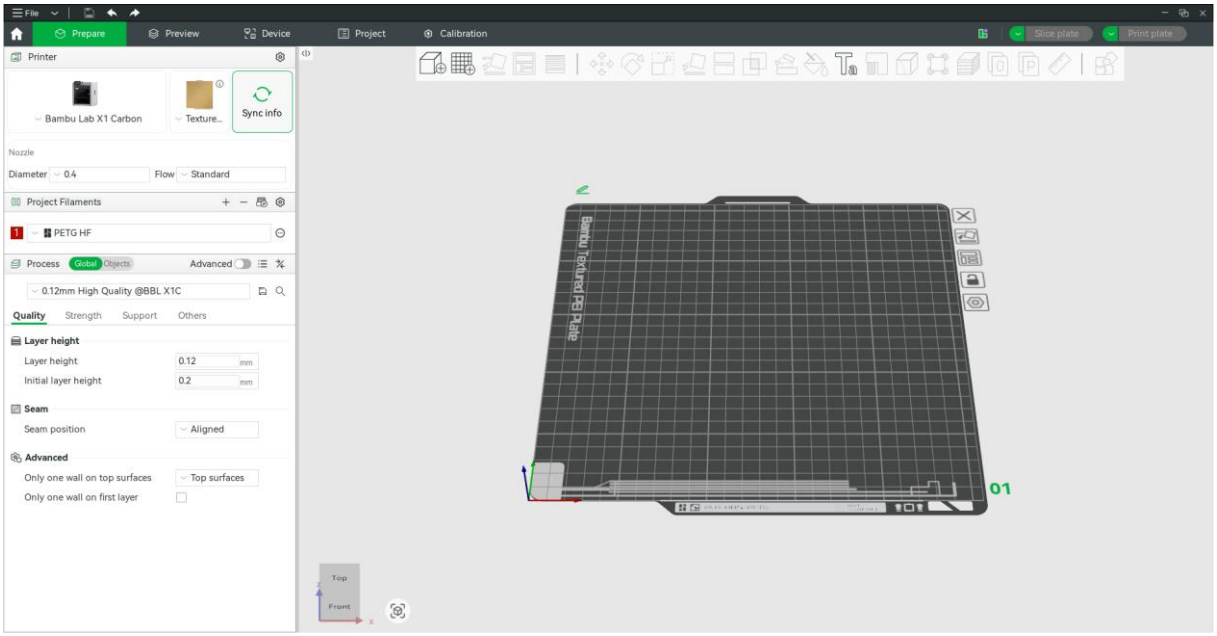


Figure 57: Bambu Studio slicing software (Source: Own edit)

During slicing, the students were confronted with the fact that geometry alone is not enough for successful printing; proper parameterization is at least as important.

By modifying the slicing settings, the relationship between print time, material consumption and structural stability became apparent. This experience helped the students to understand the manufacturing process as a complex system, rather than a series of isolated steps. During the slicing processes, the default print profiles integrated into the software provided a good starting point, which not only addressed the print quality, but also the optimal configuration resulting from the material used. With their help, the difference between the individual settings became even more clear.

In addition to specifying the printing parameters, the orientation of the printed models, color settings, and tray preparation were equally important. Special attention had to be paid to optimal adhesion of the models and the need for supports.

5.6. Materials (PLA, PETG, ABS)

During the project, we examined the properties of several commonly used 3D printing materials, however, for actual production**PETG**We chose the material. The decision was based on both professional and pedagogical considerations.

PETG provides a good balance between ease of printing and mechanical resistance, while being less prone to warping than ABS. This was especially beneficial for functional parts with longer print times. This allowed students to experience the importance of material selection in a real engineering decision-making situation.

We would like to note that during the implementation of the project we also had PLA filament available, which is considered the default material choice for most 3D printing projects, but due to its durability and low resistance to physical impacts, we discarded it and used it only for prototyping and testing purposes. In addition to the three types of materials mentioned, we did not investigate the use of other types of filament (e.g. TPU).

5.7. Steps in the printing process

The practical implementation of 3D printing was carried out along structured, interconnected steps. The conscious structure of the process allowed the students to interpret the individual phases not as isolated operations, but as part of a coherent production system. During the printing steps, special emphasis was placed on preparation, process monitoring and evaluation of the completed parts.

5.8. Model preparation and verification

In each case, printing was preceded by detailed model preparation, which aimed to prevent manufacturing errors and use resources efficiently. Students checked the geometric correctness of the digital models, with particular attention to closed bodies, wall thicknesses and critical mating surfaces.

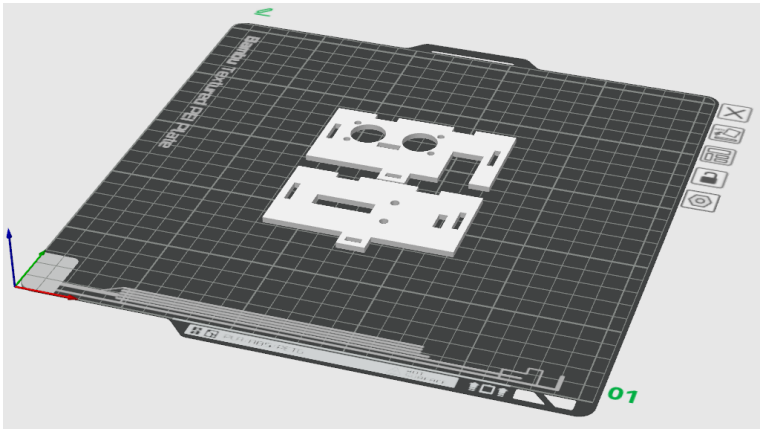


Figure 58: Model preparation and verification
(Source: My own edit)

During the preparation, the choice of printing orientation also played a prominent role. The students experienced that the placement of the part in the printing space has a direct impact on

There is a need for surface quality, mechanical strength and the amount of support material and support required. This decision-making situation facilitated the development of spatial perception and the conscious application of production technology aspects. It goes without saying that with the optimal arrangement of the elements, the students had the opportunity to start more optimized printing processes, since they could place several models on one tray.

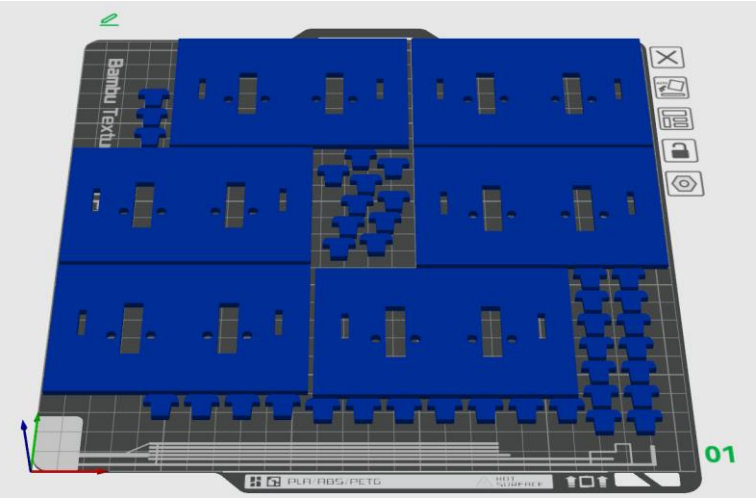


Figure 59: Printing optimization
(Source: Own editing)

At the end of the verification process, the models were validated before slicing, which confirmed the view that early detection of errors is significantly more effective than their subsequent correction.

5.9. Printing and post-production

At the beginning of the printing process, the students paid close attention to the proper adhesion of the first layer, which is one of the most critical factors for the success of the entire production. The X1 Carbon printer has a built-in AI assistant that stops printing for a short time after the first layer is placed and then inspects it via the built-in camera. If it detects a problem, it indicates it to the user via the slicing program (or the separately installable Bambu Handy mobile application). After starting the print, we considered it more informative if the participating students analyzed the quality of the print with their own eyes. Observing the first layer gave the students the opportunity to receive immediate feedback on the correctness of the preparation and settings.

Figure 60. The printer at work

Continuous monitoring of the process during longer printing jobs was also an important learning element. The students experienced that printing is not a completely autonomous operation, but a manufacturing process that may require intervention or redesign. This is the case, for example, if an object moves out of place or the filament runs out of the AMS dispenser. The students also had the opportunity to see how these situations are handled.

After printing, the finished parts were post-processed. This included handling surface irregularities and refining joint points. During post-processes, it became clear that 3D printing sometimes results in a semi-finished product that requires additional manual work for functional use. This increases with the complexity of the models and the amount of supports.

Figure 61: Results of incorrect printing (Source: Own editing)

5.10. Quality control and bug fixing

The finished parts were subjected to detailed quality control in all cases. The inspection not only covered aesthetic aspects, but primarily assessed functional suitability and assembly.

The students compared the dimensions of the printed parts with the digital model data and analyzed the causes of any discrepancies. When identifying errors, they also took into account the effects of slicing settings, material characteristics, and the printing environment. Thus, error correction was not an isolated intervention, but rather a feedback loop throughout the entire manufacturing process.

This approach fostered the development of analytical thinking and laid the foundation for the professionally justified execution of subsequent design iterations.

5.11. Sizing and fitting issues

One of the most important lessons learned from the project was the practical importance of sizing and fitting issues. The students experienced firsthand that nominal dimensions used in digital design do not always result in proper fit during manufacturing, therefore conscious management of tolerances is essential.

5.12. Handling manufacturing inaccuracies

Inaccuracies during 3D printing were the result of a combination of factors, such as material shrinkage, layer structure characteristics, and printing orientation. These phenomena made the limitations of manufacturing technology tangible for the students and necessitated minimal adjustments to the sizing of the models.

To address manufacturing variances, students learned about the use of fit-up gaps and that moving or interlocking parts require a different design approach than solid parts. This approach helped deepen their technical thinking.

5.13. Design iterations

Solving sizing problems often required multiple design and manufacturing cycles. It was also necessary to deal with cases where the parts did not fit together due to design errors rather than the printing process. Based on their experiences, the students modified the models and reprinted the affected parts, thus receiving feedback on the correctness of their decisions within a short time.

The advantage of parametric design was particularly evident during the iterative process, as the ability to quickly modify and re-manufacture dimensions allowed for efficient experimentation. This method helped students think in terms of a flexible development process rather than static solutions.

5.14. The educational significance of processes, student experiences and lessons learned

The handling of sizing and fitting problems had outstanding educational value, as the students were confronted with real engineering situations. Their independent thinking and responsible decision-making developed through analyzing problems, recognizing errors, and developing solutions.

It became clear to the students that design and manufacturing form a close unit, and neither can be understood without the other. The experiences contributed to the project not only imparting technological knowledge, but also developing an approach that can be utilized in the long term.

In addition, the experience gained in 3D printing offers additional knowledge transfer opportunities, which can be used when learning other subjects (mathematics, physics, technical subjects), as well as even in a home environment, when operating your own 3D printer.

Figure 62. Printed, assembled housing

6. Integrated control system – conceptual foundations The

6.1. purpose of the system design

A conscious design decision was made in the electronics-software phase of the project: the system is not a monolithic device based on a single microcontroller, but is implemented in the form of a distributed control architecture.

It is important to emphasize that this is not due to the "weakness" of the ESP32. The ESP32 is an excellent field controller and data acquisition unit: it provides stable digital I/O management, has multiple communication interfaces (I^2C, SPI, UART, WiFi), and is also suitable for running independent control logic. If the number of GPIOs is not sufficient, in accordance with industry practice, I/O expanders (e.g. I^2C port expanders) or additional nodes can be used — this in itself points towards distributed systems.

The real limitation during development was not in the area of I/O or communication, but in the implementation of monitoring and display (HMI) functions. The ESP32:

- has limited RAM and flash storage space,
- optimized for running non-web user interfaces, database management, and persistent logging,
- When displaying complex graphics and serving multiple clients, it quickly reaches resource limits.

In practice, this meant that although the ESP32 was able to perform field tasks stably (sensors, actuators, local control), the system's monitoring layer justified the inclusion of a device with greater computing capacity.

The resulting architecture divides the system into clearly distinct layers:

Level	Function	Device
Field level	Sensor data collection, actuator control, local logic	ESP32
Control / data management level	Cyclic processing, logical decisions, Modbus master role	Raspberry Pi / PC (Modbus)
Supervision level (HMI)	Web display, user control, data logging	Raspberry Pi / PC (Flask based server)

In this model, ESP 32 acts as a field node and, following an industrial pattern, provides data to the higher-level system via a standard protocol (Modbus TCP). It is important to note that in a purely monolithic, single-device system, this communication standard would not be necessary — but the distributed architecture makes it justified.

The goal of the project was therefore not merely to create a working smart farm model, but to demonstrate that:

- how a microcontroller becomes a field node,
- how a supervisory system-side logic and HMI layer is organized above it,
- and how the layered, networked architecture of industrial automation can be modeled in an educational environment.

This approach brings students closer to the operation of real industrial systems, while the system can still be easily expanded with new sensors, nodes, and services.

The concept of a distributed system

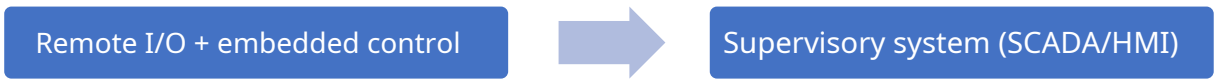
The control model developed in the project consciously follows a two-level, layered architecture. This structure was not only a technical decision, but also a step in terms of approach from the monolithic, "everything on one microcontroller" approach towards the structure typical of industrial systems.

The lower level is the field level, where the sensors and actuators are directly handled. This role is played by the ESP32 in the system. Its tasks include handling digital and analog signals, collecting sensor data, basic preprocessing, and operating physical outputs (e.g. LED, relay, servo). This level is where the direct connection to the real physical process takes place.

The upper layer is the supervisory level, which in the project is implemented by a Raspberry Pi (or PC) based system. An important difference is that at this level, the central control logic does not run, but a digital status image of the system is displayed. The Pi collects data from field-level devices (ESP32), logs it and displays it on an HMI web interface, and provides the possibility of remote intervention.

The actual control is done at the field level, on the ESP32. I/O handling, sensor data processing and basic operating logic run locally, so the system remains operational even in offline mode. Failure of the monitoring layer does not stop the physical process, only the display and remote access are affected.

This structure can thus be more accurately compared to the following industrial model:



The equivalents in the project are:

- ESP32 as an intelligent field unit that
  - he Performs I/O handling
  - he collects sensor data
  - he runs local control logic
  - he Communicates as a Modbus supervisory system
- Raspberry Pi + Python + Flask as a management layer, which
  - he collects and displays data
  - he reflecting the system status

he allows operator intervention but is not
he critical to the operation of the process

In the future, additional features can be implemented at the Raspberry Pi level, such as:

- data logging and long-term trend analysis
- graphic display
- user authorization management
- notifications, alarm logs
- remote configuration interface

These functions should be of a supervisory and informational nature only. It is a design principle that additions running on the Raspberry Pi should not affect the basic functionality or safety of the system. Critical control logic and basic state management remain at the field level.

6.2. Technical advantages of distributed architecture

One of the keys to the system is functional separation, which directly increases reliability and scalability. The ESP32 remains at the field level: sensors, inputs, outputs, and basic control logic run locally, so the operation of the physical process does not depend on a continuous network connection. This results in more deterministic I/O behavior and more stable operation.

The upper layer is responsible for monitoring, data management and visualization, not critical control. This layered architecture follows an industrial pattern, where the field device, the control layer and the monitoring system operate in separate roles.

A significant advantage of the architecture is scalability: new ESP-based nodes can be added to the network, additional sensors can be integrated, and the system can be functionally expanded without burdening the resources of a single device. This horizontal expansion is a fundamental feature of distributed systems.

From a reliability perspective, the principle is that the field level remains autonomous: a failure of the supervisory system or network does not stop local operations, only monitoring and remote intervention are affected. This directly corresponds to the practice of industrial automation.

6.3. Communication model and approach

The connection between the layers is based on Modbus TCP. This implements a state-based data exchange with a standard register structure. Sensor data and control signals do not move in the form of specific “requests and responses”, but as states accessible at well-defined addresses. This differs from the typical HTTP-based, transactional approach and is closer to industrial communication models, where the controller cyclically reads the field-level states and then writes the output registers accordingly.

System framework – logical structure

At a high level, the system data path can be described as follows:

Sensors (button, DHT, etc.) and actuators in the physical world are connected to the ESP32. The ESP32 acts as a Modbus supervisory system and makes the status of the inputs and outputs available via registers. Via the WiFi-based TCP network, the system running at the supervisory level – Raspberry Pi or PC – reads and writes this data cyclically as a Modbus client. The processed states and measured values are then displayed on the HMI web interface, where visualization and remote control are also implemented.

In this model, the ESP is not a central controller, but an intelligent field I/O unit, while the Raspberry Pi serves as the logic and monitoring center.

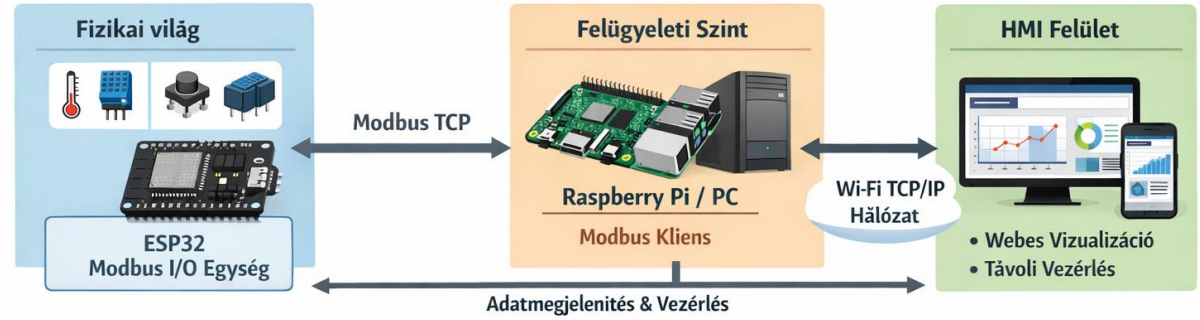


Figure 63: System framework (Source: AI generated)

Educational significance

The applied architecture is not only a technical solution, but also a tool for shaping attitudes. Students understand the reason for the separation of the field and supervisory levels, the role of resource allocation, and how the layered structure of an industrial automation system is built. This provides a direct basis for understanding PLC programming, HMI design, and SCADA systems, because the model used in the project is a simplified but realistic representation of these.

In the next section, we present the specific software implementation of the architecture: the ESP field program, the cyclic processing on the monitoring side, and the operation of the web interface.

6.4. Field control node – stand-alone sample unit

The lower level of the project's electronics system is represented by a self-contained field node, based on an ESP32 microcontroller. This unit is directly connected to the physical world: it reads sensors, controls actuators, and provides local feedback. The configuration presented is a deliberately simple, educational sample that illustrates the basic logic of the system, and does not represent a final build.

The node manages the following elements:

- a push button that functions as a digital input,
- a digital output LED,
- a DHT11 temperature and humidity sensor,
- a 16×2 character LCD display connected to an I^2C bus.

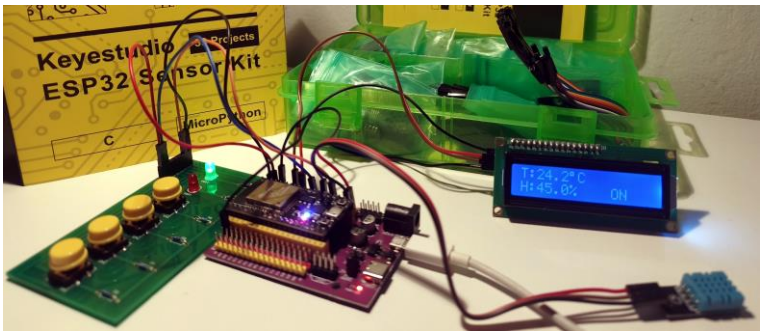


Figure 64: Test environment (Source: Own editing)

The essence of the operation is that the device performs its basic tasks completely independently, without a network connection. The user can intervene in the system state (LED on/off) with the physical button, while environmental data is continuously measured and displayed on the display. This approach models the industrial principle that the field level should be able to operate autonomously, independently of the monitoring system.

The presented program thus demonstrates the behavior of an intelligent I/O unit. The structure is modular: new sensors, output devices or displays can be easily added using additional GPIOs or communication buses. Later, this node can also be integrated with higher-level systems (PLC, HMI, SCADA), but in this chapter we focus exclusively on managing local resources.

6.5. Program structure broken down into functional units

Libraries and hardware definitions

```
// ================= KÖNYVTÁRAK BETÖLTÉSE =================
// Ebben a szakaszban történik a felhasznált perifériákhoz szükséges könyvtárak betöltése.
// DHT könyvtár a szenzor kommunikációját valósítja meg.
// A Wire és hd44780_I2Cexp az I²C LCD vezérléséhez szükséges.
#include <DHT.h>
#include <Wire.h>
#include <hd44780.h>
#include <hd44780ioClass/hd44780_I2Cexp.h>


// ================= HARDVER KIOSZTÁS =================
//A #define direktívák rögzítik a fizikai bekötés és a program közötti kapcsolatot, így a hardverkonfiguráció egyértelműen dokumentált és könnyen módosítható.

#define BTN 17      // Fizikai gomb bemenet
#define LED 2       // LED kimenet
#define DHTPIN 4    // DHT adatvonal
#define DHTTYPE DHT11 //DHT típus megadása
```

In this stage, the libraries required for the peripherals used are loaded.

- The DHT library implements sensor communication.
- Wire and hd44780_I2Cexp are required to control the I²C LCD.

#define directives record the relationship between the physical connection and the program, so the hardware configuration is clearly documented and can be easily modified.

Objects and global variables

```
//Itt jönnek létre a szenzor- és kijelzőobjektumok. Ezek a program teljes futása alatt elérhetők, így bármelyik ciklusban használhatók.
// Szenzor objektum
DHT dht(DHTPIN, DHTTYPE);
// I2C LCD objektum
hd44780_I2Cexp lcd;
```

This is where the sensor and display objects are created. They are available throughout the entire program run, so they can be used in any cycle.

Button debounce handling

```
// ================= GOMB PRELLEGÉS SZŰRÉS =================
// A mechanikus gombok érintkezői záráskor rövid ideig „rezegnek” (prellegnek), ami több hamis jelváltást okozhat.
// Ez a logika időalapú szűréssel biztosítja, hogy csak a stabil állapotváltás legyen érvényes.

bool lastReading = false;      // Legutóbbi nyers bemenet
bool stableState = false;      // Szűrt, stabil állapot
unsigned long lastChangeTime = 0;
const unsigned long debounceTime = 50;  // 50 ms stabilitási küszöb
```

The contacts of mechanical buttons “vibrate” (bounce) for a short time when closed, which can cause multiple false signal transitions. This logic uses time-based filtering to ensure that only stable state transitions are valid.

LED status and measurement variables

```
// A LED állapota külön változóban tárolódik. A DHT mérések időzítéséhez és az aktuális értékek tárolásához is globális változók szükségesek.
bool ledState = false;
unsigned long lastDHTread = 0; // Utolsó olvasás időpontja
float temp = 0;                // Hőmérséklet cache
float hum = 0;                 // Páratartalom cache
```

The LED state is stored in a separate variable. Global variables are also required for timing DHT measurements and storing current values.

Initialization – setup()

```
void setup() {

  // Hardver inicializálás
  pinMode(BTN, INPUT);
  pinMode(LED, OUTPUT);

  // DHT inicializálás
  dht.begin();

  // I2C inicializálás ESP32 alapértelmezett lábakkal
  Wire.begin(21,22);

  // LCD inicializálás
  lcd.begin(16,2);
  lcd.backlight();

  lcd.setCursor(0,0);
  lcd.print("Smart Farm Node");
  delay(2000);
  lcd.clear();
```

setup() runs once at startup. Here's what happens:

- setting the GPIO directions,
- initialization of the sensor and the I^2C bus,
- starting the LCD.

Button management and LED switching

This block implements the LED switching on the press edge. The LED only changes when the button is pressed down steadily.

```
//Ez a blokk valósítja meg a lenyomási élre történő LED-váltást.
//A LED csak akkor változik, amikor a gomb stabilan lenyomott állapotba kerül.
// Gomb olvasása
bool reading = digitalRead(BTN);

// Prellegés detektálása
if (reading != lastReading) {
  lastChangeTime = millis();
}

// Stabil állapot vizsgálata
if ((millis() - lastChangeTime) > debounceTime) {

  // Állapotváltozás kezelése
  if (reading != stableState) {
    stableState = reading;

    // Lenyomás detektálás
    if (stableState == true) {
      ledState = !ledState;         // LED kapcsolás
    }
  }
}

lastReading = reading;
```

DHT11 periodic measurement

```
// DHT szenzor olvasás időzítetten
//A szenzor csak 2 másodpercenként kerül kiolvasásra, ami megfelel a DHT11 időzítési követelményeinek.
if(millis() - lastDHTread > 2000){
  lastDHTread = millis();

  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // Hibás olvasás kizárása
  if(!isnan(h) && !isnan(t)){
    temp = t;
    hum = h;
  }
}
```

The sensor is only read every 2 seconds, which meets the timing requirements of the DHT11.

LCD display

```
// LCD kijelzés
//A kijelző a mért adatokat és a LED állapotát jeleníti meg, így a rendszer állapota hálózat nélkül is
ellenőrizhető.
lcd.setCursor(0,0);
lcd.print("T:");
lcd.print(temp,1);
lcd.print((char)223);
lcd.print("C ");

lcd.setCursor(0,1);
lcd.print("H:");
lcd.print(hum,1);
lcd.print("% ");

lcd.setCursor(11,1);
lcd.print(ledState ? "ON " : "OFF");
```

The display shows the measured data and LED status, so the system status can be checked even without a network.

Summary

This program demonstrates the operation of an autonomous field control unit:

- physical input processing
- actuator control
- sensor data collection
- local visualization

The code structure separates individual functions well, which makes it easier to later expand and connect to higher-level systems.

6.6. Field control node with network expansion – WiFi + Modbus TCP

In this version, the previously introduced standalone field controller is supplemented with network capabilities.

The local operation (button → LED, DHT measurement, LCD display) remains unchanged, but the device is now able to:

- Connect to a WiFi network
- Act as a Modbus TCP monitoring system
- make the status of physical inputs and outputs available through registers
- for remote reading and intervention by an external system (PLC / monitoring software)

It is important that the logic does not move out of the field device: local control remains autonomous, the network is only used for status sharing and remote intervention.

Program structure according to functional units

Libraries and hardware definitions

```
#include <WiFi.h>          // ESP32 WiFi stack
#include <ModbusTCP.h>     // Modbus TCP szerver kezelése
#include <DHT.h>           // DHT szenzor magas szintű kezelése (protokollt elrejti)
#include <Wire.h>          // I2C kommunikáció (LCD miatt)
#include <hd44780.h>       // LCD vezérlő
#include <hd44780ioClass/hd44780_I2Cexp.h> // I2C LCD expander kezelés

ModbusTCP mb;  // Az ESP Modbus TCP szerverének létrehozása.


// ================= HARDVER KIOSZTÁS =================
// GPIO-k definiálása hardver absztrakcióként

#define BTN 17         // Fizikai gomb bemenet
#define LED 2          // Fő LED kimenet (vezérelt fogyasztó)
#define WIFI_LED 5     // WiFi státusz visszajelző LED (terepi hibajelzés)
#define DHTPIN 4       // DHT adatvonal
#define DHTTYPE DHT11 // Szenzor típusa

// Szenzor objektum létrehozása
DHT dht(DHTPIN, DHTTYPE);

// I2C LCD objektum létrehozása (automatikus I2C címfelismerés)
hd44780_I2Cexp lcd;
```

Network parameterization

```
// ================= WIFI HÁLÓZATI PARAMÉTEREK =================
// Ezek konfigurálják a hálózati kapcsolatot, valamint biztosítják,
// hogy az ESP mindig ugyanazon IP címen legyen elérhető.

const char* ssid = "plc";
const char* pass = "12345678";

IPAddress local_IP(192, 168, 137, 200);
IPAddress gateway(192, 168, 137, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress primaryDNS(8,8,8,8);
IPAddress secondaryDNS(8,8,4,4);


// ================= WIFI ÚJRACSATLAKOZÁSI LOGIKA =================
// A rendszer nem blokkoló reconnect stratégiát használ.
// Ha a kapcsolat megszakad, a program nem áll meg,
// hanem meghatározott időközönként újrapróbál csatlakozni.

unsigned long lastReconnectAttempt = 0;         // Utolsó reconnect próbálkozás időpontja
const unsigned long reconnectInterval = 300000; // 5 perc újrapróbálkozási ciklus
bool wifiConnected = false;                     // Hálózati állapot logikai jelző

// WiFi csatlakozási rutin
// Ez a függvény bontja az előző kapcsolatot,
// majd új kapcsolatot kezdeményez a megadott SSID-re.
void connectToWiFi() {
  WiFi.disconnect(true);
  delay(100);                // Rövid stabilizációs várakozás
  WiFi.begin(ssid, pass);
}


// ================= WIFI HIBA LED IDŐZÍTÉS =================
// Amennyiben nincs WiFi kapcsolat,
// a WIFI_LED másodperces periódussal villog.
// A megvalósítás millis() alapú, tehát nem blokkolja a fő ciklust.

unsigned long lastBlinkTime = 0;     // Utolsó LED állapotváltás
const unsigned long blinkInterval = 1000; // 1 másodperces villogási periódus
bool wifiLedState = false;           // WiFi LED aktuális állapota
```

Button debounce handling, LED status and measurement variables

```
// ================= BELSŐ ÁLLAPOT RÉTEG =================
// A rendszer egyetlen "igazságforrást" használ a LED állapotára.
// A fizikai gomb, a Modbus és a fizikai LED is ezt az állapotot követi.

bool ledState = false;   // A LED központi logikai állapota


// ================= GOMB PRELL SZŰRÉS =================
// Mechanikus gomb érintkezése záráskor pattog (prell).
// A stabil állapot detektálás millis alapú időszűréssel történik.

bool lastReading = false;       // Utolsó nyers bemeneti érték
bool stableState = false;       // Szűrt, stabil állapot
unsigned long lastChangeTime = 0;
const unsigned long debounceTime = 50;  // 50 ms stabilitási küszöb


// ================= DHT MINTAVÉTELI IDŐZÍTÉS =================
// A DHT szenzor olvasása nem történhet túl gyakran,
// ezért 2 másodperces mintavételezési ciklust alkalmazunk.

unsigned long lastDHTread = 0;
float temp = 0;
float hum = 0;
```

Initialization

```
// ====================== SETUP ======================
// ===================================================
void setup() {

  Serial.begin(115200);

  // Hardver inicializálás
  pinMode(BTN, INPUT);
  pinMode(LED, OUTPUT);
  pinMode(WIFI_LED, OUTPUT);

  // ===== Stabil statikus IP konfiguráció =====
  // Az ESP32 WiFi stack sajátossága miatt:
  // - Station mód explicit beállítása szükséges
  // - DHCP cache törlése
  // - Statikus IP konfigurálása csatlakozás előtt

  WiFi.mode(WIFI_STA);
  WiFi.persistent(false);
  WiFi.disconnect(true);
  delay(100);

  WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS);
  WiFi.begin(ssid, pass);

  // Induláskori maximum 10 másodperces várakozás.
  // Ha ezalatt nem jön létre kapcsolat,
  // a program offline módban folytatódik.
  unsigned long startAttemptTime = millis();
  while (WiFi.status() != WL_CONNECTED &&
         millis() - startAttemptTime < 10000) {
    delay(500);
  }

  wifiConnected = (WiFi.status() == WL_CONNECTED);

  // Szenzor és perifériák inicializálása
  dht.begin();
  Wire.begin(21,22);

  lcd.begin(16,2);
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Smart Farm Node");
  delay(2000);
  lcd.clear();
```

Modbus register allocation

```
  // ================= MODBUS REGISZTERKIOSZTÁS =================
  // Coil 0  → LED állapot
  // Ists 0  → Gomb állapot
  // Hreg 0  → Hőmérséklet (x10 skálázva)
  // Hreg 1  → Páratartalom (x10 skálázva)

  mb.server();
  mb.addCoil(0);
  mb.addIsts(0);
  mb.addHreg(0);
  mb.addHreg(1);

  // Coil inicializálása a belső logikai állapotból
  mb.Coil(0, ledState);
}
```

WIFI monitoring, Modbus service with active connection

```
void loop() {

  // ===== WIFI FELÜGYELET =====
  // Nem blokkoló hálózati állapotellenőrzés.
  // Ha nincs kapcsolat, 5 percenként újrapróbál.

  if (WiFi.status() != WL_CONNECTED) {

    wifiConnected = false;

    if (millis() - lastReconnectAttempt >= reconnectInterval) {
      lastReconnectAttempt = millis();
      connectToWiFi();
    }
  }
  else {
    wifiConnected = true;
  }

  // ===== WIFI HIBA LED KEZELÉS =====
  // Offline állapotban 1 Hz villogás.
  // Online állapotban LED kikapcsolva.

  if (!wifiConnected) {

    if (millis() - lastBlinkTime >= blinkInterval) {
      lastBlinkTime = millis();
      wifiLedState = !wifiLedState;
      digitalWrite(WIFI_LED, wifiLedState);
    }
  }
  else {
    digitalWrite(WIFI_LED, LOW);
    wifiLedState = false;
  }

  // ===== MODBUS KISZOLGÁLÁS =====
  // Csak aktív hálózat esetén kezel TCP kéréseket.
  if (wifiConnected) {
    mb.task();
  }

  // ================================================
  // ===== MODBUS → BELSŐ ÁLLAPOT SZINKRON =====
  // ================================================
  // Ha külső rendszer írja a Coil 0-t,
  // a belső állapot frissül.

  bool coilState = mb.Coil(0);

  if (coilState != ledState) {
    ledState = coilState;
  }
```

Button reading, LED status control and remote monitoring

```
// ================================================
// ===== GOMB → BELSŐ ÁLLAPOT =====
// ================================================
// Fizikai gombnyomás esetén a belső állapot toggolódik,
// majd a Coil regiszter tükrözi azt.

bool reading = digitalRead(BTN);

if (reading != lastReading) {
  lastChangeTime = millis();
}

if ((millis() - lastChangeTime) > debounceTime) {

  if (reading != stableState) {
    stableState = reading;

    if (stableState == true) {
      ledState = !ledState;
      mb.Coil(0, ledState);
    }
  }
}

lastReading = reading;

// ================================================
// ===== FIZIKAI LED VEZÉRLÉS =====
// ================================================
// A fizikai kimenet mindig a belső állapotot követi.

digitalWrite(LED, ledState);
```

Periodic measurement, remote monitoring of DHT11, button status reporting via Modbus

```
// ================================================
// ===== DHT IDŐZÍTETT OLVASÁS =====
// ================================================

if (millis() - lastDHTread > 2000) {

  lastDHTread = millis();

  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (!isnan(h) && !isnan(t)) {
    mb.Hreg(0, (int)(t * 10));
    mb.Hreg(1, (int)(h * 10));
    temp = t;
    hum = h;
  }
}

// Gomb státusz jelentése Modbuson
mb.Ists(0, stableState);
```

LCD display

```
273     // ================================================
274     // ===== LCD KIJELZÉS =====
275     // ================================================
276     // Első sor: Hőmérséklet + Hálózati állapot
277     // Második sor: Páratartalom + LED állapot
278
279     lcd.setCursor(0,0);
280     lcd.print("T:");
281     lcd.print(temp,1);
282     lcd.print((char)223);
283     lcd.print("C ");
284     lcd.print(wifiConnected ? "N-ON " : "N-OFF");
285
286     lcd.setCursor(0,1);
287     lcd.print("H:");
288     lcd.print(hum,1);
289     lcd.print("% ");
290     lcd.print(ledState ? "L-ON " : "L-OFF");
291
292     delay(10);  // CPU terhelés csökkentése
293 }
```

System block diagram

The block diagram illustrates the layered, PLC-like architecture of the ESP32-based firmware. At the top level are the WiFi interface and ModbusTCP communication, which are solely responsible for network data exchange and register space management. The center of the system is the State Layer, which stores all internal state variables – such as LED status, measured temperature, humidity, and WiFi status – as a single source of truth.

The Input Layer handles the physical inputs, while the Output Layer handles the outputs, and both rely directly on the internal state layer. The control remains entirely on the ESP32, i.e. at the field level, so the system operates autonomously, regardless of the network. The Modbus and supervision layers only reflect the internal state as a digital shadow, but do not carry the control logic. This structure ensures deterministic, offline operation and the robust architecture typical of industrial systems.

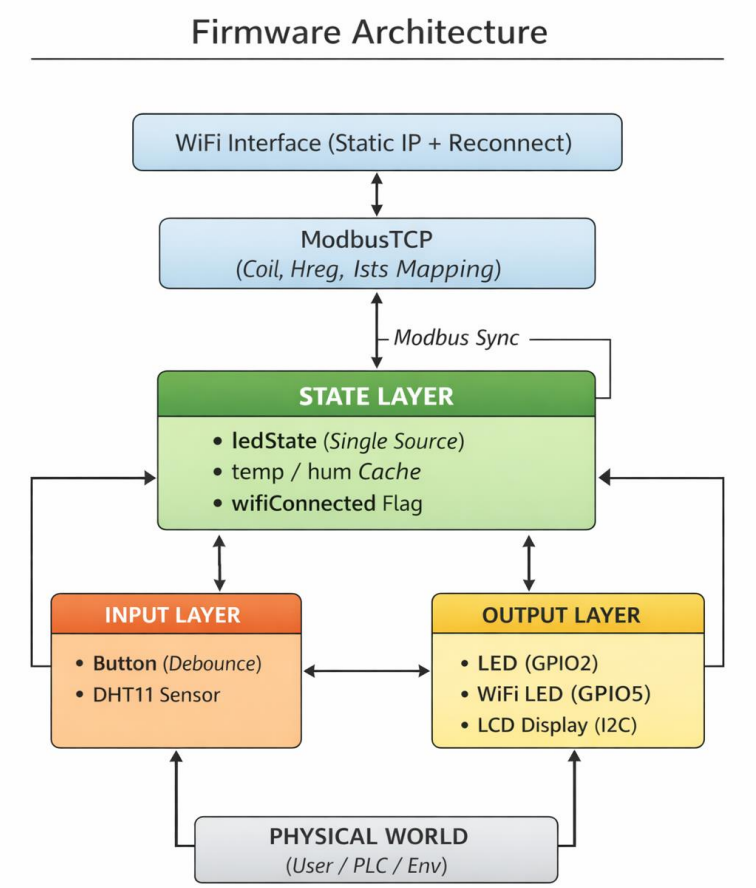


Figure 65: System block diagram (Source: AI generated)

6.7. Supervision level – Python based HMI / supervision cycle

The field ESP32 node makes the status of sensors and actuators available via Modbus registers. The upper level is not responsible for taking over physical control, but for digitally mapping and monitoring the system.

This layer in the project is a Python-based, cyclical application, which in terms of its function is an HMI/SCADA-type monitoring system.

Tasks:

- cyclic query of field data
- storing and processing states
- providing remote intervention options
- web display service

Figure 66: Python-based HMI (Source: Own edit)

6.8. Preparing and deploying a Raspberry Pi-based monitoring system

The next step was to run the software monitoring logic on a standalone, low-power hardware that can also be used in industrial environments. For this purpose**Raspberry Pi** We used a Raspberry Pi based computer, which provides an ideal platform for Python-based applications that include both network communication and a web interface. The Raspberry Pi is a relatively high-performance computer with a good price-performance ratio, which can be flexibly adapted to a variety of applications. By default, it runs a Debian-based distribution, Raspberry OS, which fully met the requirements of the project.

Figure 67: Raspberry Pi (Source: https://malnapc.hu/raspberry-pi-5-4gb)

The Raspberry Pi in this architecture**as a monitoring and visualization layer**It functions as a controller that communicates with the ESP32-based subsystem via Modbus TCP protocol, while providing a web-based HMI interface for users.

Operating system and base system installation

The device preparation began with the installation of a stable, long-term supported Linux-based operating system. Although it is possible to install custom operating systems, the Raspberry Pi OS provided the appropriate hardware support, system stability, and easy access to the Python development environment. The system was installed using the official Raspberry Pi Imager program, which, after specifying basic configuration settings, provided the students with a ready-made desktop environment for further work.

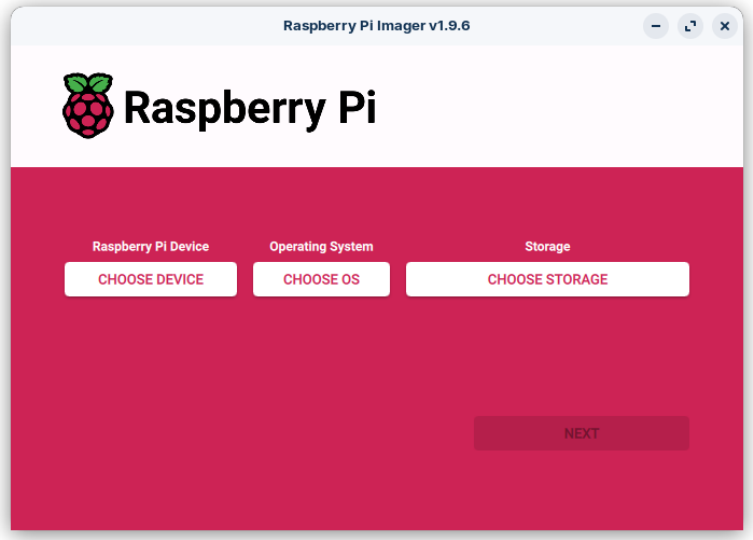


Figure 68: Installing Raspberry Pi OS (Source: Own edit)

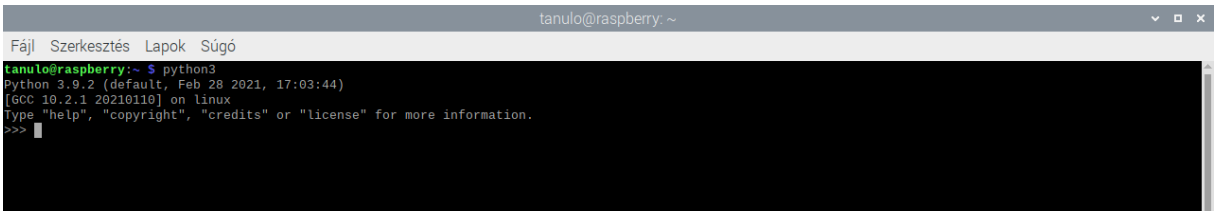
During system installation, the following occurred:

- configure the default user account,
- system update,
- and checking the network connection.

Reliable network operation was of paramount importance, as the entire monitoring logic is based on IP-based communication.

Preparing the Python runtime and dependencies

Since the monitoring application is written in Python, a unified and maintainable Python environment was created on the Raspberry Pi. The system includes Python 3



Version 1.

During the preparation of the environment, emphasis was placed on:

- using the Python package manager (PIP),
- installation of the necessary external libraries (web server, Modbus client),
- and avoiding version conflicts.

```
tanulo@raspberry:~ $ pip install flask
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: flask in /usr/lib/python3/dist-packages (1.1.2)
tanulo@raspberry:~ $ pip install pymodbus
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting pymodbus
  Downloading https://www.piwheels.org/simple/pymodbus/pymodbus-3.8.6-py3-none-any.whl (164 kB)
     |████████████████████████████████| 164 kB 938 kB/s
Installing collected packages: pymodbus
  WARNING: The script pymodbus.simulator is installed in '/home/tanulo/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed pymodbus-3.8.6
tanulo@raspberry:~ $
```

This step drew students' attention to the reproducibility and maintainability of software systems.

Network configuration and Modbus TCP connection

The basic requirement for the system to work is a stable Modbus TCP connection between the Raspberry Pi and the ESP32. To achieve this, the Raspberry Pi operated with a fixed IP address in a predictable network environment, so that the monitoring application could always reach the subsystem.

Students understood that the reliability of network communication is crucial for a surveillance system.

The Modbus connection in this context was not used for low-level control, but for status querying and command transmission, which fits well with the concept of the software PLC cycle.

Application launch and parallel operation management

When the monitoring program is started, the logic for performing cyclic status updates runs on a separate thread, while the Flask-based web server handles user requests. This solution ensures that serving the web interface does not block the system's monitoring functions.

The Raspberry Pi's resources proved to be sufficient:

- for continuous Modbus communication,
- for cyclic state processing,
- and to serve the web HMI.

This operating model well demonstrated to students the difference between parallel processing and non-real-time but deterministic operation.

Education and system integration experience

The development of the Raspberry Pi-based runtime environment was integral to the previous stages of the project. The students saw a complete system in action, in which sensors and actuators appear at the physical level, communication is via industrial protocols, and monitoring and control are implemented using software tools.

Furthermore, the Debian-based operating system gave students the opportunity to deepen their knowledge of Linux. Since the configuration of these devices in an industrial environment is not necessarily done through a graphical interface, we aimed to have students work as much as possible through a command-line interface.

However, during the practical implementation, the students and I reached the hardware limitations of the Raspberry Pi. When the entire student team connected to the web servers at the same time, there was a slowdown in data processing. In addition, the additional load would have required the installation of active cooling, as the chip PC rebooted several times due to overheating. Although it met the needs of the project, it served as an important lesson during the implementation that in an industrial-level system, planning the resources of the server device in advance and assessing the expected load should be a priority.

6.9. The role of the Python client

The ESP Modbus monitoring system, the Python program, aims to implement the monitoring logic and digital "shadow".

Connecting to ESP

```python
# ================= FLASK + MODBUS TCP PLC RÉTEG =================
# Ez a program egy "szoftveres PLC ciklust" valósít meg.
# Feladata:
#   • ESP32 Modbus eszköz lekérdezése
#   • LED vezérlési logika kezelése
#   • Szenzoradatok továbbítása a webes HMI felé


# ================= MODULOK IMPORTÁLÁSA =================
from flask import Flask, render_template, jsonify
from pymodbus.client import ModbusTcpClient
import threading, time


# ================= WEB SZERVER OBJEKTUM =================
# Flask kezeli a HTTP kéréseket (HMI felület)
app = Flask(__name__)


# ================= MODBUS KLIENS (ESP32 FELÉ) =================
# Ez a sor létrehozza a TCP kapcsolatot az ESP Modbus felügyeleti rendszerével.
esp = ModbusTcpClient("192.168.0.200", port=502)


# ================= ÁLLAPOTVÁLTOZÓK LÉTREHOZÁSA =================
# Ezek a változók a rendszer állapotát tükrözik a szerver oldalon

led_state = False        # LED logikai állapot (kimenet)
btn_state = False        # Fizikai gomb állapota
prev_btn_state = False   # Előző ciklus gomb állapota (élérzékeléshez)

temperature = 0          # Hőmérséklet cache
humidity = 0             # Páratartalom cache
```

Structure of the supervisory cycle

```python
# ================= PLC CIKLUS (IPARI LOGIKA MINTÁJÁRA) =================
# Ez a függvény a felügyeleti alkalmazás ciklikusan futó központi része.
# Bár a neve „felügyeleti ciklus”, funkcionálisan nem valós idejű vezérlési ciklust, hanem egy állapotfrissítő, felügyeleti lekérdezési ciklust valósít meg.
# Ez egy végtelen ciklus, amely fix periódusban (200 ms) végzi az I/O beolvasást és a vezérlési logikát.

def plc_cycle():
    global led_state, btn_state, prev_btn_state, temperature, humidity

    while True:

        # ===== DIGITÁLIS BEMENET OLVASÁSA (FIZIKAI GOMB) =====
        # Modbus Input Status regiszter 0
        # Az ESP az aktuális gombállapotot a Discrete Input 0 címre írja.
        rr_btn = esp.read_discrete_inputs(address=0, count=1)

        if not rr_btn.isError():
            btn_state = rr_btn.bits[0]


        # ===== ÉLÉRZÉKELÉS (EDGE TRIGGER) =====
        # Ez a PLC logika:
            # ha a gomb éppen most lett lenyomva → LED állapot vált,
            # a parancs visszakerül az ESP-re
        # Ipari PLC logikában ez "RISING EDGE DETECT".
        # Csak akkor reagálunk, amikor a gomb 0→1 állapotba vált

        if btn_state and not prev_btn_state:
            led_state = not led_state          # LED állapot váltás
            esp.write_coil(0, led_state)       # Fizikai LED frissítés Modbuson

        prev_btn_state = btn_state


        # ===== ANALÓG ADATOK OLVASÁSA =====
        # Holding Register 0 → hőmérséklet
        # Holding Register 1 → páratartalom
        # Az ESP tizedes pontossággal küldi az adatot.

        rr_temp = esp.read_holding_registers(address=0, count=2)

        if not rr_temp.isError():
            temperature = rr_temp.registers[0] / 10.0
            humidity = rr_temp.registers[1] / 10.0


        # ===== PLC CIKLUS IDŐZÍTÉS =====
        # 200 ms ciklusidő (ipari PLC mintára)
        time.sleep(0.2)
```

Web (HMI) display

```
# ================= WEB OLDAL =================
# A HMI felület betöltése

@app.route("/")
def index():
    return render_template("index.html")


# ================= ÁLLAPOT API =================
# A webes felület innen kapja az aktuális rendszerállapotot

@app.route("/status")
def status():
    return jsonify({
        "led": led_state,       # LED állapot
        "btn": btn_state,       # Gomb állapot
        "temp": temperature,    # Hőmérséklet
        "hum": humidity         # Páratartalom
    })


# ================= VIRTUÁLIS GOMB (WEB HMI) =================
# Ez a fizikai gomb funkcióját utánozza a weboldalon keresztül

@app.route("/toggle_led", methods=["POST"])
def toggle_led():
    global led_state

    led_state = not led_state          # LED állapot váltás
    esp.write_coil(0, led_state)       # Parancs küldése az ESP-nek

    return "OK"


# ================= PROGRAM INDÍTÁS =================
if __name__ == "__main__":

    # PLC ciklus külön szálon fut, hogy a web szerver ne blokkolódjon
    threading.Thread(target=plc_cycle, daemon=True).start()

    # Flask webszerver indítása
    app.run(host="0.0.0.0", port=5000)
```

6.10. Logging sensor data to SQL database

As the next step in the development of the monitoring system, we set the goal of not only displaying sensor data in real time on the web HMI interface, but also recording it in a structured, searchable format. To achieve this, an SQL-based database integration was created to supplement the existing Python application.

This development added a new dimension to the project, as the system now performs not only monitoring but also data collection functions, enabling their later analysis.

The role of data logging in the surveillance system

Saving sensor data in a database allows for tracking changes over time, longer-term analysis of operation, and later processing of the data. This clearly demonstrated to the students that in real systems, displaying the current state alone is not enough; historical storage of data is a fundamental requirement.

During development, emphasis was placed on understanding that data collection does not serve continuous control purposes, but rather creates a decision-support and analysis basis.

Database selection and logical structure

A simple, SQL-based database structure was created to fit the educational environment, which is easy to use, yet clearly illustrates the principles of relational data management. The database records the measured data in a single, well-defined table. We used a MariaDB (MySQL-based) database engine for the database.

The stored data is typically (**sensor data**table):

- recording identifier (**dataid**, int, auto-increment, primary key)
- timestamp (**time_of_measurement**, datetime, default current_timestamp)
- temperature values (**pace**, double),
- humidity data (**hum**, double),

This structure allows for chronological retrieval of data and simple statistical processing.

Complementing your Python application with database management

During the expansion of the existing Flask-based application, database management was built in as a separate logical unit. Sensor data is still read cyclically via Modbus TCP communication, but updated values are also entered into the database.

Although in the current implementation, data is continuously sent to the database, we drew the students' attention to the fact that in a complex system, different functions operate with different time criticalities. Based on this, the next step in the development of the application could be to reduce the frequency of data logging to only every few seconds.

To extend the Python program, it was first necessary to import the mysql.connector module.

```
# ================= MODULOK IMPORTÁLÁSA =================
from flask import Flask, render_template, jsonify
from pymodbus.client import ModbusTcpClient
import threading, time
import mysql.connector #Az SQL kapcsolatot kezelő modul importálása.
```

In the next step, we created the objects needed to manage the database connection and submit data.

```
# ================= ADATBÁZIS KAPCSOLAT KONFIGURÁLÁSA =================
# Az adatbázis kapcsolatot kezelő objektum létrehozása.
db = mysql.connector.connect(
    host="localhost",
    port="3306",
    user="root",
    password="",
    database="sensordatabase"
)

#Az adstbázist kezelő cursor objektum létrehozása.
dbcursor = db.cursor()
```

Finally, when querying analog sensor data, we extended the code by sending the data to the database server:

```
        # ===== ANALÓG ADATOK OLVASÁSA =====
        # Holding Register 0 → hőmérséklet
        # Holding Register 1 → páratartalom
        # Az ESP tizedes pontossággal küldi az adatot.

        rr_temp = esp.read_holding_registers(address=0, count=2)

        if not rr_temp.isError():
            temperature = rr_temp.registers[0] / 10.0
            humidity = rr_temp.registers[1] / 10.0
            dbcursor.execute("INSERT INTO sensordata (temp,hum) VALUES (" + temperature + ","+humidity+")")


        # ===== PLC CIKLUS IDŐZÍTÉS =====
        # 200 ms ciklusidő (ipari PLC mintára)
        time.sleep(0.2)
```

Educational significance and student experiences

The introduction of SQL database integration significantly expanded the learning value of the project. Students were introduced not only to real-time data management, but also to structured data storage and later usability.

During development, the following evolved:

- basic knowledge of data management and data modeling,
- systems thinking,
- and sensitivity to data security and data consistency.

Recording sensor data in a database thus organically fit into the project as a whole and further strengthened the view that the basis of modern IT and industrial systems is the conscious and structured management of data.

6.11. Web management layer – its role in the system

The Python application provides access to data via HTTP endpoints, while the HTML + JavaScript interface running in the browser takes care of display and user interaction.

This layer:

- does not directly control physical I/O
- displays a digital shadow of the ESP status
- forwards user commands to the Python application

The data path:

LED status block

This button does not directly control the ESP, but the Python management system.

```
<!-- ====================== LED KÁRTYA ====================== -->
<!-- Ez a blokk a LED állapotát mutatja és innen lehet vezérelni -->
<div class="card">
    <h2>LED állapot</h2>

    <!-- A LED vizuális visszajelzője
         A színét a JavaScript módosítja:
         zöld = világít
         piros = nem világít -->
    <div id="led" class="led"></div>

    <!-- Virtuális kapcsoló gomb
         onclick esemény → toggleLED() JS függvény fut le
         Ez HTTP POST kérést küld a Flask szervernek (/toggle_led)
         Ez a gomb nem közvetlenül az ESP-t vezérli, hanem a Python felügyeleti rendszert.-->
    <button onclick="toggleLED()">LED kapcsolás</button>
</div>
```

Button status block

```
<!-- ====================== FIZIKAI GOMB ÁLLAPOT ====================== -->
<!-- Ez CSAK kijelzés, nem vezérlő szerv -->
<div class="card">
    <h2>Gomb állapot</h2>

    <!-- A fizikai gomb állapotának szöveges kijelzése.
         A Flask szerver PLC ciklusa frissíti.
         "Nyomva" / "Nincs nyomva" szöveg jelenik meg -->
    <div id="btn">Nincs nyomva</div>
</div>
```

Text display of the physical button status.

Sensor block

```
<!-- ====================== KÖRNYEZETI ADATOK ====================== -->
<!-- DHT11 szenzor által mért adatok -->
<div class="card">
    <h2>Környezeti adatok</h2>

    <!-- Hőmérséklet kijelzés
         Ezek a DHT szenzor értékei számára fenntartott mezők.
         A program ide írja be a /status API-ból érkező értéket. -->
    <div>🌡 Hőmérséklet: <span id="temp">--</span> °C</div>

    <!-- Páratartalom kijelzés -->
    <div>💧 Páratartalom: <span id="hum">--</span> %</div>
</div>
```

These are fields reserved for DHT sensor values.

JavaScript - live update

```
// ======================== ÁLLAPOTFRISSÍTŐ FÜGGVÉNY ========================
// Ez a függvény 200 ms-onként lefut.
// Feladata: lekérdezni a Flask szervertől az aktuális rendszerállapotot és frissíteni a weboldalon látható adatokat.

function update(){

  // HTTP GET kérés a szerver /status végpontjára
  // A szerver JSON formátumban küldi vissza az adatokat:
  // { led: bool, btn: bool, temp: float, hum: float }
  fetch("/status")

    // A válasz átalakítása JSON objektummá
    .then(r => r.json())

    // A kapott adatok feldolgozása
    .then(data => {

      // ===== LED VIZUÁLIS ÁLLAPOT =====
      // A LED kör színének beállítása:
      // zöld ha világít (true), sötétpiros ha nem (false)
      document.getElementById("led").style.background =
        data.led ? "limegreen" : "darkred";


      // ===== FIZIKAI GOMB ÁLLAPOT KIÍRÁS =====
      // A Flask PLC ciklus által olvasott bemenet jelenik meg
      document.getElementById("btn").innerText =
        data.btn ? "Nyomva" : "Nincs nyomva";


      // ===== HŐMÉRSÉKLET KIÍRÁS =====
      // DHT11 szenzor adata az ESP → Modbus → Flask útvonalon
      document.getElementById("temp").innerText = data.temp;


      // ===== PÁRATARTALOM KIÍRÁS =====
      document.getElementById("hum").innerText = data.hum;
    });
}
```

JavaScript retrieves and updates these variables.

LED switch button operation

```
// ======================== VIRTUÁLIS LED KAPCSOLÓ ========================
// Ez a függvény akkor fut le, amikor a felhasználó megnyomja a weboldalon a "LED kapcsolás" gombot.

function toggleLED(){

  // HTTP POST kérés küldése a szervernek
  // A Flask oldalon ez a /toggle_led útvonalat hívja meg, ami megfordítja a LED állapotát és kiírja Modbus-on az ESP-re.
  fetch("/toggle_led", { method: "POST" });

  // Fontos: itt nincs válaszfeldolgozás, a következő update() ciklus már lekéri az új állapotot.
}
```

Process:

Update cycle

```
// ======================== IDŐZÍTETT FRISSÍTÉS ========================
// A böngésző 200 ms-onként automatikusan meghívja az update() függvényt.
// Ez kvázi PLC ciklus a weboldalon (HMI polling).
// Ez megfelel a Python programban lévő ciklus ütemének.

setInterval(update, 200);
```

This matches the Python cycle rate, so the system stays synchronous.

The role of style in the system

CSS ensures that the HMI web interface:

- be readable even in low light conditions
- provide clear visual feedback about statuses
- separate information blocks (HMI panel logic)

This structure follows the minimalist approach of SCADA / industrial HMI (Human–Machine Interface).

Full CSS – commented in detail

```
/* ========================= OLDAL ALAPSTÍLUS ========================= */
body {
    font-family: Arial;
    background: linear-gradient(135deg,#0f2027,#203a43,#2c5364);
    color: white;
    text-align: center;
    margin: 0;
}

/* ========================= KÁRTYA (PANEL) STÍLUS =========================
   Minden információs blokk ezt használja:
   LED, Gomb, Szenzor adatok*/
.card {
    background: #222;
    margin: 20px auto;
    padding: 20px;
    width: 320px;
    border-radius: 12px;
    box-shadow: 0 0 10px #000;
}

/* ========================= LED VISSZAJELZŐ KÖR =========================
   Ez a vizuális LED indikátor*/
.led {
    width: 40px;
    height: 40px;
    margin: 15px auto;
    border-radius: 50%;
    background: darkred; /* Alapértelmezett állapot: LED kikapcsolva. A program dinamikusan felülírja:
    limegreen = bekapcsolva */
}


/* ========================= GOMB STÍLUS =========================
   Virtuális LED kapcsoló*/
button {
    padding: 10px 15px;
    border-radius: 8px;
    border: none;
    background: #00aaff;
    color: white;
    cursor: pointer;
}
```

This corresponds to the display principles of real PLC HMI, SCADA terminal and industrial control panel.

Why is this important in education?

Students not only display data but also learn how an HMI (Human–Machine Interface) can:

- functional visual system
- where color, size, and arrangement have meaning

The web layer is now complete:

6.12. Student development of the web HMI interface in teamwork

After implementing the basic functions of the monitoring system, the next element of the project was the further development of the web HMI (Human–Machine Interface). In this phase, the students no longer implemented predefined solutions, but rather developed the existing HTML and CSS-based interface based on their own ideas and in teamwork.

The goal of the task was for students to experience how a working but basic-looking interface can be made more user-friendly, more clear, and more visually informative, while maintaining the functional stability of the system.

Teamwork and task sharing

Students worked in small working groups, in which the division of tasks was deliberate. Some students focused primarily on transforming the structure (arranging HTML elements, logical grouping), while others focused on refining the appearance (colors, layout, visual feedback).

This form of work organization allowed students to experience the benefits and challenges of joint development, as well as practice coordination and development based on each other's work.

Functional and visual enhancement of the interface

The web interface, which served as a starting point, already included the display of the system's basic states and basic control options. The students further developed this interface:

- the display of information has been rearranged,
- they expanded the range of displayed sensor data, preparing for a later development,
- clearer visual feedback was developed,
- and the appearance style was modified according to the needs of their team.

During the modifications, special attention was paid to clarity and usability, especially to ensure that the interface is easy to interpret in a real-world monitoring environment.

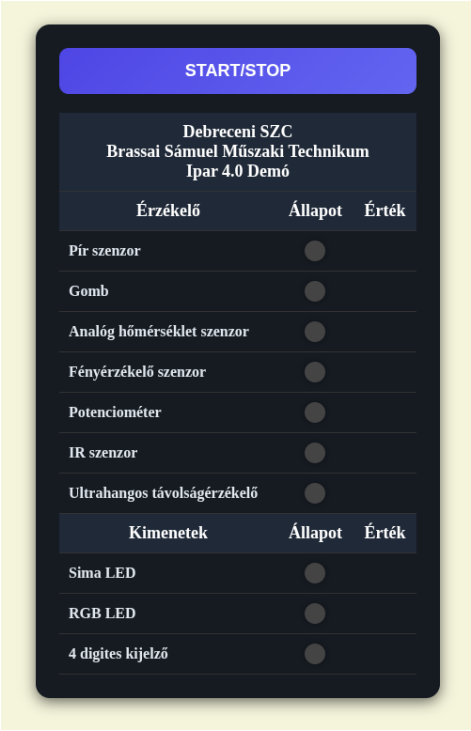


Figure 69: Students' improved work 1. (Source: Own editing)

Emergence of individual needs and creative solutions

During the development process, students were able to shape the appearance and functionality of the interface according to their own ideas. As a result, various solutions were created, for example:

- different color coding for status indicators,
- alternative layouts for displaying sensor data,
- more pronounced feedback during interventions.

This freedom provided an opportunity for creativity to flourish, while students learned to balance individual ideas with the technical and functional limitations of the system.

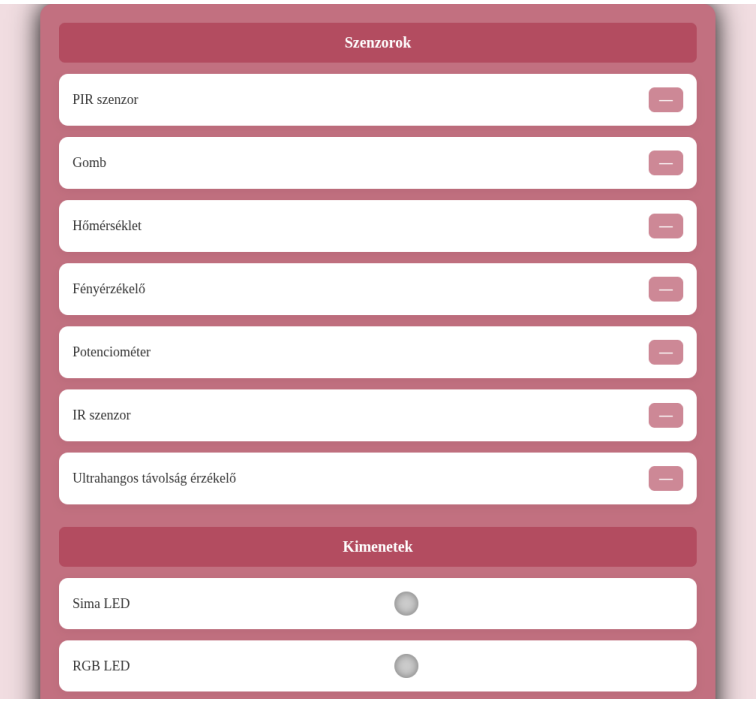


Figure 70: Students' improved work 2. (Source: Own editing)

Educational significance and development experiences

During the further development of the web HMI interface, the students applied their previously acquired knowledge in a complex way. The task simultaneously developed:

- digital and IT competences,
- cooperation and communication skills,
- and user-centered thinking.

The process was well aligned with the project objectives, as the students did not just create technical solutions, but thought about the interface as a working system, considering its role in the overall surveillance architecture.

6.13. Summary – system integration and learning outcomes

The activities presented in this chapter constituted one of the most complex and pedagogically valuable phases of the project. In this phase, the students implemented a complete, working surveillance system in which the hardware elements, network communication, software control logic, and web display functioned as a unified system.

Preparing the Raspberry Pi-based runtime environment and deploying the Flask-based application gave students the opportunity to apply their programming and system operation skills in real IT and industrial environments. Using Modbus TCP communication was a particularly important experience, as students were able to

They were able to work with an industry-wide protocol and understand its role in a surveillance architecture.

During the further development of the web HMI interface, the students not only created technical solutions, but also mastered a user-centered approach. During the developments carried out in teamwork, individual creativity, independent decision-making, and joint professional consultation emerged. The customization of the interfaces contributed to the students feeling that the system was their own and taking responsibility for its operation.

From a pedagogical perspective, it is worth highlighting that students received continuous feedback on the consequences of their decisions throughout the entire development process. Testing the system's operation, identifying and correcting errors, and jointly evaluating solutions strengthened self-reflection and problem-solving thinking. Dealing with errors was not seen as a failure, but as a natural part of the learning process.

The activities implemented in Chapter 6 contributed to the development of several key competences, in particular:

- digital competence,
- collaboration and communication skills,
- algorithmic and systems thinking,
- and in the areas of independent learning and responsibility.

Overall, this chapter is a good example of how a project-based, practice-oriented approach can connect theoretical knowledge with a real-world application environment. Students not only created a working technical system, but also gained experiences that will lay the foundation for their professional development and career orientation in the long term.

7. Summary

The document presents a complex, Industry 4.0 and IoT-based educational project, which focuses on the development of a smart home/smart farm model. According to the project philosophy, learning is not a passive reception, but an active creative process, where students reach solutions through real problems. The emphasis is not on ready-made answers, but on the thought process, decisions and iteration. Students do not simply use technology, but design, analyze and develop it, while experiencing that the operation of a system is always the result of compromises and conscious choices.

The educational goals accordingly go beyond classical technical knowledge. The project simultaneously develops programming and electronics knowledge, systems thinking and collaboration skills. Students learn to interpret the phenomena of the physical world in digital systems, recognize cause-and-effect relationships and are able to handle complex problems in several steps. It is particularly important that the project provides an opportunity to involve students with different backgrounds, which strengthens explanatory thinking and joint problem solving.

From a technical perspective, the project aims to create an IoT-based system in which sensors, controllers and actuators work together as a unified logical system. The starting point is a factory-made Smart Farm Kit, which, however, only serves as a basis: the students' task is not to reproduce it, but to understand the system and then further develop it towards their own, industrial-oriented architecture. This process models real-world engineering work well, where new, optimized systems are created after analyzing existing solutions.

Implementation is carried out through a clearly structured yet flexible process, with clearly identifiable key milestones. The learning journey is not linear, but typically follows the following steps:

- learning and assembling the factory system (Smart Farm Kit),
- analysis of the operation of individual components (sensors, actuators, control),
- revealing the limitations and shortcomings of the system,
- defining own development goals and functions,
- design of individual mechanical elements and housing parts (e.g. 3D modeling),
- production of physical elements (e.g. 3D printing),
- redesign and expansion of the electronic system,
- programming (block-based → text-based),
- integration of subsystems into a single system,
- testing, debugging and iterative development.

This process not only results in technical advancement, but also teaches students that a complex system is never “finished” but is constantly evolving and being refined.

Figure 71: Smart home (Source: Own image, AI generated background)

The difficulties encountered during the project – lack of basic concepts, understanding the differences between analog and digital signals, and troubleshooting – turned into valuable learning experiences by the end of the process. Technical problems, such as noisy measurement data or unstable operation due to inappropriate limit values, helped the students to recognize that the behavior of the system is a direct consequence of the design decisions.

By the end of the project, the students were able to think in systems, interpret input processing–output logic, and understand the relationship between the physical and digital worlds. The solutions created were already consciously built, functioning systems.

The most important lesson is that the project not only provided technical knowledge, but also developed an approach that can be directly applied in the modern industrial and technological environment.

Funded by the European Union. The information and statements contained herein represent the views of the author(s) and do not necessarily reflect the official opinion of the European Union or the Tempus Public Foundation. Neither the European Union nor the funding authority can be held responsible for them.