HELLENIC MEDITERRANEAN UNIVERSITY

POLYTECHNIC SCHOOL

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

REMOTE LABORATORIAL INFRASTRUCTURE FOR THE DISTANT TEACHING OF IOT PROGRAMMING

by

RALLIS IOANNIS

B.A Technological Education Institute of Crete 2015

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

2025

Approved by:

Dr. Panagiotakis Spyridon

Abstract

The increasing demand for practical Internet of Things (IoT) skills in education and industry, particularly in the era of Industry 4.0, has prompted the development of remote laboratory infrastructures that enable hands-on programming and experimentation without the need for physical presence. This thesis presents the design, implementation, and evaluation of a remote IoT laboratory platform aimed at supporting distance learning in IoT programming. The proposed system integrates hardware components such as Arduino, Raspberry Pi with a cloud-based backend, ensuring remote control, code execution, and real-time feedback via web interfaces.

A full-stack, containerized software architecture was adopted using Angular for the frontend, Node.js and Express.js for the backend, and Docker for deployment. Flask-based services on Raspberry Pi enabled direct interaction with physical hardware, while tools like CodeMirror, and WebSockets ensured efficient user interaction and real-time communication, JWT-based session management, and encrypted data channels.

Experimental validation included the development and testing of platform features such as a booking system, user dashboards, code upload interface, and live video streaming. The system was evaluated through user testing, surveys, and performance benchmarks. Results demonstrated that students could effectively learn IoT concepts remotely, with comparable or improved performance relative to traditional labs. The implementation of load balancing, container scaling, and logging tools like Prometheus and Grafana ensured robust and scalable operations.

Ultimately, the project confirms that well-designed remote laboratories can significantly enhance access to technical education by providing a flexible, scalable, and secure environment for IoT programming. Future directions include deeper LMS integration, AI-driven assessment tools, and the expansion to support diverse IoT hardware configurations.

Table of Contents

Contents

Chapter 1 - Introduction 1

Chapter 2 - Theoretical Background 9

2.1 Fundamental Concepts of IoT and Remote Laboratories 9

2.1.1 Definition and key characteristics of IoT 9

Definition of IoT 9

Key Characteristics of IoT 10

2.1.2 Role of remote laboratories in education and research 13

2.1.3 Advantages and challenges of remote IoT education 16

2.2 Key Technologies and Tools 17

2.2.1 Arduino and Raspberry Pi 17

2.2.2 Docker and Kubernetes 20

2.2.3 Cloud Computing 24

2.3 Existing Approaches and Related Research 28

2.3.1 Case studies of remote laboratory applications in universities 28

2.3.2 Comparative analysis of existing solutions and technologies 31

2.3.3 Modern trends and challenges in IoT education through remote laboratories. 32

2.4 Review of Remote Laboratory Platforms 34

2.4.1 Overview of the Remote Laboratory System 34

2.4.2 Equipment Parts and Configuration 35

2.4.3 Development of Architectural Software 36

2.4.4 Information Exchange and Communication Standards 37

2.4.5 Remote IoT Lab Security Issues 38
2.4.6 Case Study: Remote IoT Laboratory at HMU-RLP 38
2.5 Overview of development and evaluation methods in remote laboratories 38
2.5.1 Development of the Platform 39
2.5.2 Testing and User Evaluation 43
2.5.3 Results and Performance Analysis 43
Chapter 3 - Requirements Analysis and Technology Selection for Implementation 45
3.1 System Requirements Analysis 45
3.2 Visualize Information Flow 46
3.2.1 Frontend Technologies: 47
3.2.2 Backend Technologies 48
3.2.3 Infrastructure & Deployment 49
3.2.4 Remote Lab Hardware Stack 49
3.2.5 Information Data Flow 50
3.3 Stack of Technology Justification 51
Chapter 4 - System execution 54
4.1 Initial Setup and Development Context 54
4.2 Frontend Deployment 54
4.3 Backend Development 55
4.4 Access management and booking system 55
4.5 Docker-Based Implementing Strategy 56
4.6 Remote Integration of Labs 57
4.7 Last Test and Verification 58
Chapter 5 - System Demonstration and User Interface Documentation 59

5.1 Login and Authentication Interface 59
5.2 User Dashboard 59
5.3 Booking System and Calendar 60
5.4 Laboratory Access and Code Upload Interface 61
5.5 Live Video Monitoring 62
5.6 Instructor Management Tools 63
5.7 Error Handling and Notifications 63
5.8 Responsive Design and Mobile Support 64
5.9 System Feedback and Logging 64
5.10 User Testing and Scenario-Based Evaluation 64
5.10.1 Testing Methodology 65
5.10.2 Observations and Results 65
5.10.3 Discussion 66
Chapter 7 - Conclusions and Future Directions 69
References i

List of Figures

Figure 1 Typical ecosystem of a remote laboratory (Rahimi et al., 2020). 3

Figure 2 Exploring the full potentials of IoT (Allioui, & Mourdi, 2023). 4

Figure 3. The power-related components on the Arduino Uno R3 board (Kouklinos, 2024). 18

Figure 4 Raspberry Pi 4 Model B 4GB (Why.gr, n.d.). 19

Figure 5 Understanding Docker and Kubernetes: Containerization vs. Orchestration (Index.dev., 2024). 20

Figure 6 Working with a containerized environment with VScode and the Dev Containers extension (Image Credit: Krispin, R. (2024)..... 23

Figure 7 Cloud fundamentals: Architecture strategies & security solutions. The Hackers Meetup. (2025, February 28). 25

Figure 8 DHT11 & DHT22 Sensors Temperature and Humidity Tutorial using Arduino (Nedelkovski, n.d.). 36

Figure 9 The MQTT full form is Message Queuing Telemetry Transport, it is a protocol that is mainly used for connecting devices with each other on the internet with backend services (PsiBorg Technologies, n.d.)..... 40

Figure 10 Remote Lab Architecture with Raspberry Pi, Flask, Arduino CLI, and Live Video Streaming 50

Figure 11 OAuth2 Authorization Code Flow for Secure Third-party Authentication..... 52

Figure 12 The Arduino CLI Workflow: From Code Authoring to Board Upload 53

Figure 13 Backend Build Stage with Node.js, Prisma, and .env Support in Multi-stage Dockerfile 56

Figure 14 Session Timeout Countdown Triggering Alert and User Redirection in IoT Lab Platform 58

Figure 15 Login screen with local login form and OAuth buttons. 59

Figure 16 Student dashboard after login. 60

Figure 17 Booking interface with interactive calendar. 61

Figure 18 Session timer showing time remaining..... 61

Figure 19 Video Feed..... 63

Figure 20: Error Handling Missing Bracket 64

Acknowledgements

This project could not have been accomplished without the invaluable support and guidance from many individuals and organizations. I want to express my heartfelt thanks to my advisor, Dr. Spyridon Panagiotakis , for their priceless guidance, motivation, and constant support during this research journey. I am also thankful to Hellenic Mediterranean University for offering the essential resources and infrastructure to create this thesis. I want to express my gratitude to my fellow classmates and friends who played a vital role through our discussions, feedback, and teamwork. Furthermore, I genuinely value the individuals and educators who took part in the evaluation process, dedicating their time and insightful viewpoints to enhance the system. Lastly, I am grateful for the love and support of my family and friends, whose encouragement kept me motivated during this experience.

Chapter 1 - Introduction

This thesis presents the development of a **Remote Laboratory Management System (RLMS)**—a comprehensive platform that enables **remote access, operation, monitoring, and evaluation of physical laboratory equipment** via the internet. RLMS platforms have emerged as essential tools for providing experimental learning to students without direct access to physical labs, especially within the context of remote or hybrid education (García-Zubía & Alves, 2012). These systems integrate tools for booking, user and session management, device control, monitoring, logging, and security (Håkansson et al., 2017). Additionally, they can be integrated with Learning Management Systems (LMS) to support educational assessment and learning analytics (Fidalgo-Blanco et al., 2015). The demand for such systems increased sharply during the COVID-19 pandemic and continues to grow, as RLMSs offer **flexible, accessible, and scalable technical education** in fields such as IoT, embedded systems, and automation (Zutin et al., 2020). The RLMS implemented in this work focuses on providing **hands-on IoT programming with real hardware** (Arduino, Raspberry Pi, sensors, etc.) through a secure and user-friendly web interface.

In the proposed master thesis, we intend to develop and deploy an information system which can be utilized as a management tool for administering the access to remote laboratories featuring various IoT equipment. Using Python for programming the backend of the system and various javascript frameworks for the frontend (e.g., Vue), we are going to develop several applications for user and session management at the remote lab (e.g., a booking system for scheduling the access of users to the lab), as well as for facilitating the user access to the underlying laboratorial equipment. Arduinos and associated peripherals will be used for implementing the embedded experimentation setup and a Raspberry Pi microcomputer for running the control of the setup.

Despite the growing number of RLMS platforms and remote IoT labs, many existing implementations either lack scalability, have limited hardware integration, or don't offer real-time collaboration and code execution capabilities. Additionally, integration with cloud-based services and container orchestration platforms like Kubernetes remains largely unexplored in educational remote labs. This dissertation addresses these limitations by proposing a unified, scalable, and secure RLMS for practical IoT programming.

Given the recent demand in industry 4.0 for Internet of Things (IoT) programming skills, academic and other educational units across the world (e.g. for life-long learning) try to enhance the curricula they offer with such courses. Inevitably, practical training of students on these technologies is required. In the context of distant learning, remote laboratories are innovative applications where certain real hardware/software experimental systems can be completely manipulated and controlled via the internet (Michalec et al., 2019; Gorbounov and Chen, 2021).

They give students the chance to precisely replicate industry-standard processes by configuring networked settings, interacting with actual IoT devices, and debugging programming faults. Furthermore, these systems enable real-time cooperation so that several users may work on the same hardware configuration concurrently, hence fostering team-based learning and problem-solving. The integration of remote IoT laboratories into academic and professional training programs is becoming an indispensable strategy to prepare learners for the changing needs of Industry 4.0. With the growing attention on IoT-driven automation, smart manufacturing, and intelligent systems.

This had become almost fundamentally important for academic and research activities during the COVID-19 pandemic (Azad, 2021), but remains critical also for the post-pandemic era we currently live in due to the convenient and personalized learning style it enables. A remote laboratory, where one has unrestricted access to run emulations or simulations, can be an efficient approach to allow laboratorial experience without geographical limitations, thus maximizing the use of the resources that are already available (Miguel et al., 2015; Lowe et al., 2013; Mitsea, and Drigas, 2019).

Only remote laboratories can support inter-laboratory communication and coordination between other universities and research institutions, allowing faculty members to share a wide range of different expensive technology situated in different locations around the world. As a result, groups of academics and scientists are making tremendous progress in building remote laboratories thanks to the availability of evolving technology and improved knowledge of learning methodologies.

Traditionally, in remote laboratories a web service is typically used for database, user and session management, and personal PCs and workstations are used as the primary unit for hosting the experimentation setup of the remote laboratory. Figure 1 depicts an indicative embedded

system-based framework for a remote laboratory platform. The experimental setup, embedded system, cloud, and remote web client are the main components. The experimental setup is connected to an embedded system, which acts as a conduit between the experiment and the remote client. The cloud component represents the web service that manages the local setup of the lab.

Normally, the embedded facilities host a dedicated infrastructure based on various microcontrollers and associated peripherals (sensors and actuators) for letting remote clients conducting several experiments, providing needed assessment, and implementing real-time learning management. The developed facilities are utilized for delivering laboratorial classes and at the same time collecting data for assessing learning effectiveness and system development.

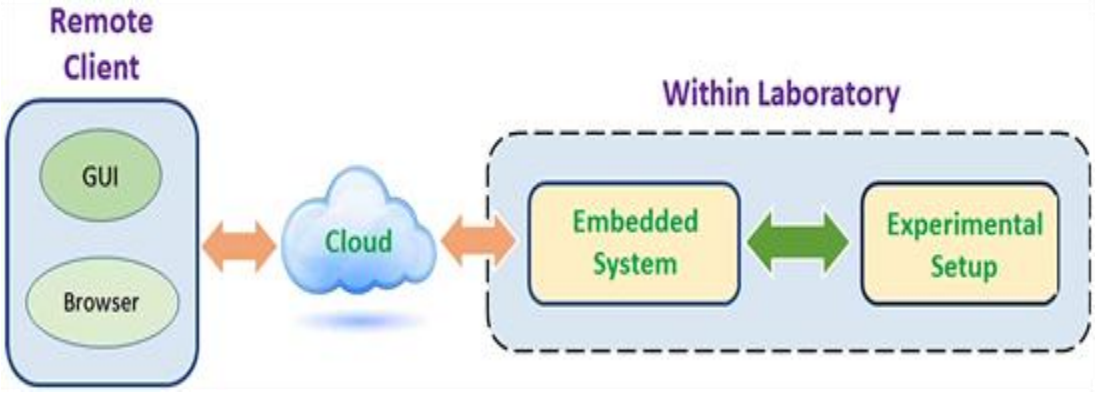


Figure 1 Typical ecosystem of a remote laboratory (Rahimi et al., 2020).

Nowadays, companies are trying to enable diffusion and training on the IoT technologies over such remote setups (Fernández-Pacheco et al., 2019).

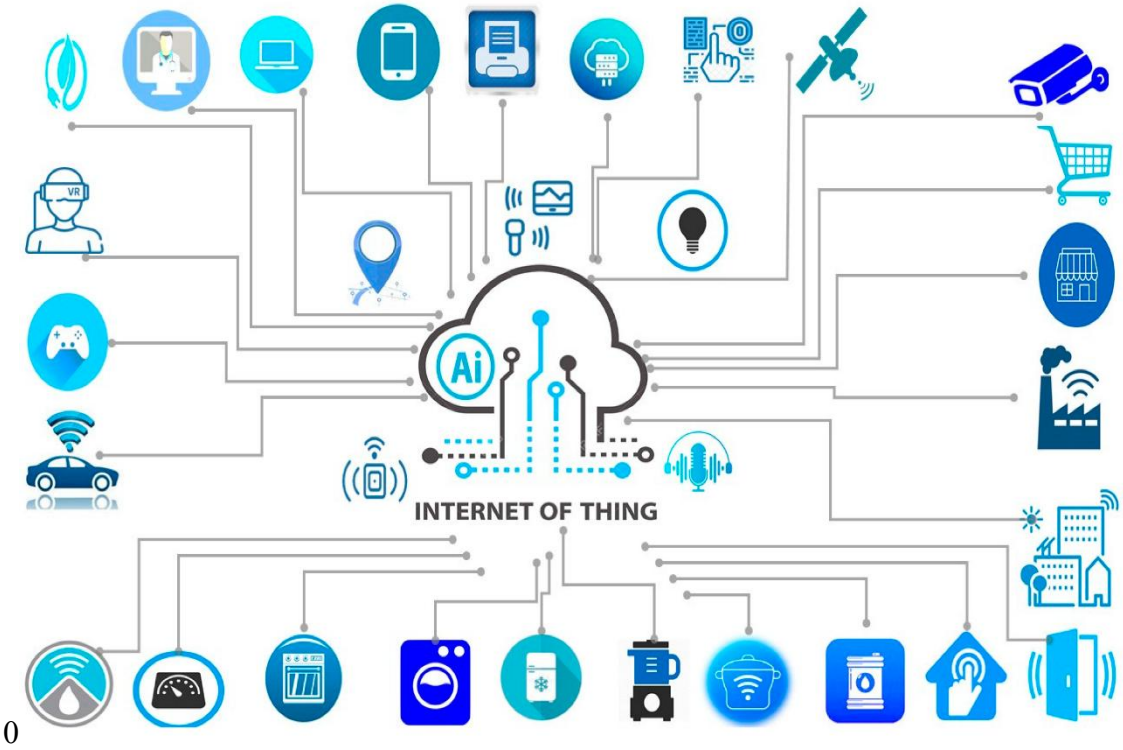


Figure 2 Exploring the full potentials of IoT (Allioui, & Mourdi, 2023).

The adoption of affordable microcontroller platforms like Arduino and Raspberry Pi enables universities and educational institutions to provide cost-effective components for hands-on training, allowing students to gain familiarity with various technological solutions (Sobota et al., 2013; Jamieson & Herdtner, 2015; Hajjar & Spears, 2019). A primary task assigned to students using these dedicated remote infrastructures is to develop autonomous systems, leveraging available hardware to implement specific control functions. According to the principles of IoT, these systems must continue functioning seamlessly, collecting and remotely storing data in cloud-based databases, and maintaining actuator control through the Internet, even amid connectivity disruptions or changes in client location (Rahimi et al., 2020). A visual overview of IoT applications and their connected domains is presented in Figure 2.

Numerous universities currently utilize Arduino and similar open-source hardware platforms extensively in courses focused on embedded systems (Fernández-Pacheco et al., 2021). Jamieson notably began introducing the Arduino platform in 2010 as an option for student project implementations. This shift led to many students preferring Arduino over alternatives such as FPGAs or PIC microprocessors for project completion.

Moreover, the introduction of Arduino led to increased variety and complexity in student projects compared to previous years (Balogh, 2010). Another significant instance occurred at the Technological University of Bratislava (Grodotzki et al., 2018), which previously employed the commercial Boe-Bot platform for robotic development. However, students reported inefficiencies using the BASIC programming language. Transitioning to Arduino boards produced significantly improved outcomes, enabling students to successfully program robots capable of line-following and obstacle avoidance after only a few lessons.

Despite these successes, integrating remote laboratories into foundational teaching, training, and learning processes continues to pose substantial challenges. Major difficulties include effectively integrating diverse hardware and software technologies, acquiring and maintaining industrial-grade equipment, managing system integration, ensuring uninterrupted availability of remote laboratory services, and dealing with a lack of comprehensive vendor support.

The main objectives of this thesis are:

- To design and develop a scalable RLMS using container-based technologies.
- To integrate real IoT hardware (Arduino, Raspberry Pi) into a secure web-accessible platform.
- To evaluate the system's usability and educational impact through user testing and performance analysis.

User and session management will be implemented in the form of a web service running in the server part of the system. Remote users who intent to use the lab, after successful authentication and login with the server, will be able to monitor the available sensors and control the available actuators via the proper utilization of a built-in and user-friendly web interface. Therefore, the user will be able to access a remote laboratory of IoT hardware via the web and run his code on top of this infrastructure. The compilation of user code from Pi to Arduino will be done with the support of the Arduino Cli solution (Arduino CLI, n.d.). A web camera will transfer a view of the remote lab to its user. The server infrastructure will be containerized (e.g., running in several Dockers) for security reasons and enhanced transferability of the whole deployment.

The following chapters first present the theoretical background of this thesis, highlighting key concepts of the Internet of Things (IoT), the role of remote laboratories in education, and technological challenges of distance learning. The core technologies used in the platform (Arduino, Raspberry Pi, Docker, Kubernetes, Cloud Services) are then analyzed in detail, along

with a comparative review of existing RLMS solutions. The thesis proceeds with a comprehensive presentation of the design and implementation of the proposed RLMS, focusing on its software and hardware architecture, communication protocols, security design, and integration with real lab equipment. The final chapters discuss the results of system testing, user evaluations, and the educational effectiveness of the platform.

To validate the effectiveness of the proposed RLMS, the system was iterated and tested in real-world conditions. The assessment was conducted through a combination of technical performance benchmarks, user testing with students, and trainers with feedback techniques through interviews. Security, scalability, usability and availability were key evaluation criteria.

The structure of this dissertation is organized into six chapters, each of which deals with a specific aspect of research.

Chapter 2 provides the theoretical background and literature review. It explores fundamental concepts of the Internet of Things (IoT), the role and benefits of remote labs in education, the key technologies used (Arduino, Raspberry Pi, Docker, Cloud services), and a comparative analysis of existing remote lab platforms.

Chapter 3 describes the requirements of the system and justifies the choice of the technology stack to be used for development. It includes an analysis of functional and non-functional requirements and presents the information flow of the proposed platform.

Chapter 4 describes the process of implementing the system. It covers backend and frontend component configuration, hardware component integration, and deployment strategy using Docker containers.

Chapter 5 introduces the user interface of the system and introduces its basic functions. This includes the login mechanism, user dashboard, booking system, code upload interface, live video streaming, and trainer tools. The chapter also concludes with a lightweight usability evaluation based on scenario-driven testing with representative user personas. While this evaluation is preliminary in nature and limited in scope, it offers valuable initial feedback regarding user interaction and system functionality. A more comprehensive evaluation phase—with broader participation, formal usability metrics, and long-term analysis—is planned for future development stages.

Chapter 6 concludes the thesis by summarizing the key findings, evaluating the platform's effectiveness, and suggesting possible directions for future work, such as integrating with LMS and using AI for automated assessment.

This structure ensures a logical progression from theoretical foundation to application and evaluation, aligning technical work with educational objectives and actual application.

Chapter 2 - Theoretical Background

This chapter provides the theoretical background supporting the development of the proposed IoT Remote Lab. It starts by introducing key concepts related to the Internet of Things (IoT) and explains the role of remote labs in modern education. It then introduces the main technologies used in such systems, including microcontrollers, container tools, and cloud services. Finally, it examines relevant research and examples from existing platforms, offering useful information that helps explain the design choices made in the following chapters.

2.1 Fundamental Concepts of IoT and Remote Laboratories

2.1.1 Definition and key characteristics of IoT

One of the most revolutionary technological developments of the modern period, the Internet of Things (IoT) transforms several sectors by allowing flawless connection between devices. IoT is defined as a system of interconnected physical items embedded with sensors, software, and other technologies to gather and share data over the internet without human involvement (Atzori, Iera, & Morabito, 2016). < The basic definitions and main features of IoT are investigated in this study, therefore clarifying its importance in modern technical environments.

Definition of IoT

Kevin Ashton came up with the term "Internet of Things" (IoT) back in 1999 when he was at Auto-ID Labs at MIT. IoT is basically seen as a worldwide setup where physical and virtual things are linked together using information and communication technologies (ICT) to help gather and share data (Ali, Ali, & Badawy, 2015). Sorri, Mustafee, and Seppänen (2022) say that IoT is changing the game in tech ecosystems, pushing for smarter networking and automation.At its core, IoT consists of three main components:

1. Things (Devices and Sensors): These are physical items like smart appliances, industrial machines, and wearable devices that gather data using built-in sensors.

2. Connectivity: The networks like Wi-Fi, Bluetooth, and 5G that help devices talk to each other and connect to cloud storage or computing platforms.

3. Data Processing and Analytics: This is all about the setup that handles the data we gather, making it possible to make decisions on the fly and automate things (Tamrakar et al., 2022).

The IoT ecosystem spans a bunch of industries like healthcare, smart homes, agriculture, manufacturing, and transportation, showing just how broad and promising it really is.

Key Characteristics of IoT

IoT possesses several defining characteristics that distinguish it from traditional computing networks and technologies. The following are the key characteristics of IoT:

1. Connectivity and Interoperability

IoT is mostly defined by its ability to effortlessly link several devices and systems. Whether it's via cloud platforms, wireless tech, or mobile networks, connectivity ensures that gadgets may interact seamlessly (Atzori et al., 2016). IoT operates best when it can instantaneously communicate data, therefore reducing delays and enabling far better automation.

IoT depends much on interoperability since it guarantees that all sorts of devices, regardless of their manufacturer or running system, may interact and operate without problems. Adopting IPv6 and building universal communication protocols are only two of the standardizing initiatives meant to increase IoT ecosystem interoperability (Sorri et al., 2022).

2. Automation and Self-Configuration

IoT systems are all about automation, making things more productive while cutting down on the need for people to get involved. They rely on real-time data to let devices update themselves, find networks, and boost their performance all on their own (Ali et al., 2015). Take smart home gadgets like security cameras and thermostats; they adjust their settings automatically based on what users like and what's happening around them.

3. Data Collection and Processing

Tons of data are gathered by the Internet of Things from many kinds of sensors on many different devices. Effective processing and analysis of these data sources will help to extract insightful information. Making ensuring IoT systems can rapidly and effectively handle enormous volumes of data calls for cloud computing, edge computing, and artificial intelligence algorithms (Tamrakar et al., 2022).

In things like self-driving cars, health monitoring systems, and industrial automation—where any delay in decision-making may truly cause major issues—real-time data processing is absolutely vital.

4. Scalability

One of the coolest things about IoT is scalability, which means the network can manage more and more devices without slowing down. With the rise of IoT, it's super important for millions of connected devices to integrate seamlessly into the network infrastructure. Effective cloud services, strong networking protocols, and improved bandwidth capacity boost IoT scalability (Sorri et al., 2022).

Smart cities totally rely on IoT to manage traffic, monitor environmental conditions, and enhance public safety. The scalability of IoT systems ensures they can keep running smoothly and stay relevant over time.

5. Security and Privacy

There is still a significant problem with security in the Internet of Things. It is of the utmost importance to ensure that we have data protection, authentication, and encryption in place (Ali et al., 2015). This is due to the fact that Internet of Things devices are constantly exchanging classified information. Hacking, illegal access, and malware are examples of key cybersecurity concerns that pose a significant danger to Internet of Things ecosystems.

Because Internet of Things devices are able to capture a large amount of data, privacy concerns arise. Regulatory frameworks such as the General Data Protection Regulation (GDPR) in Europe are primarily concerned with safeguarding user information and ensuring that privacy requirements are satisfied. Internet of Things (IoT) security can be significantly improved by using

security measures such as blockchain technology, multi-factor authentication, and encrypted communication routes.

6. Efficiency in Energy Use

The Internet of Things (IoT) devices typically operate on low power, which means that they need to be designed for energy efficiency in order to extend the battery life of the device and save expenses. According to Tamrakar et al. (2022), technologies such as energy-harvesting sensors, low-power communication protocols (such as LoRaWAN and Zigbee), and power-saving modes significantly improve the energy efficiency of Internet of Things (IoT) systems.

The use of the Internet of Things (IoT) for predictive maintenance in industrial settings helps reduce the amount of energy that is wasted by monitoring the performance of machines and identifying potential problems before they occur. Smart grids are able to improve energy distribution by utilizing Internet of Things technology, which in turn reduces consumption and increases efficiency.

7. Maintaining a watchful eye on the situation and taking charge

Real-time monitoring and management of linked devices is made possible by the Internet of Things (IoT), which is of critical significance across a variety of business sectors. Patients' vital signs are monitored by wearable gadgets in the healthcare industry, which also alerts medical experts if anything seems out of the ordinary. In a similar manner, Internet of Things sensors are utilized by smart farming technology in order to monitor the moisture content of the soil and to fine-tune irrigation schedules (Sorri et al., 2022).

Industrial Internet of Things (IIoT) applications make use of real-time monitoring to improve overall operational performance, as well as to increase production efficiency and reduce downtime.

8. Integrating Machine Learning and Artificial Intelligence into a Composite

Using artificial intelligence and machine learning expands the capabilities of the internet of things by enabling predictive analytics, identifying abnormalities, and making more informed decisions (Tamrakar et al., 2022). Internet of Things applications that are powered by artificial intelligence include things like self-driving cars, smart supply chain management systems, and smart assistants like Alexa and Google Assistant.

Algorithms that learn through machine learning examine vast volumes of data collected from Internet of Things devices in order to identify trends, improve automation, and enhance user experiences. There is a growing convergence between artificial intelligence and the internet of things, as evidenced by chatbots driven by AI, facial recognition technology, and systems for predictive maintenance.

2.1.2 Role of remote laboratories in education and research

Bringing remote labs into education and research has totally changed the game for students in technical fields, especially for those in engineering, science, and tech. Thanks to remote laboratories, students and researchers can access complex experimental setups from anywhere, removing geographical and logistical barriers (Poo, Lau, & Chen, 2023). These labs really boost the quality of education by making hands-on learning better, promoting teamwork, and creating a community vibe.

Understanding What Remote Laboratories Are All About

Students and researchers can totally control physical equipment from afar using remote labs. These digital platforms let users run experiments in real time online. Unlike virtual labs that rely on simulations, remote labs use real instruments and technology that exist in the actual world (Usmonovna, 2024). The main reason these labs are so important is that they give students and researchers the chance to get hands-on experience without needing to be in a traditional lab setting.

Benefits of Remote Laboratories in Education

Remote laboratories enable students from various geographical locations to engage in high-quality laboratory experiences. This is especially advantageous for institutions with constrained resources or for students with disabilities who may encounter challenges in participating in in-

person lab sessions (Poo et al., 2023). Operating a fully equipped laboratory incurs significant expenses, including the costs of equipment, maintenance, and laboratory space. Remote laboratories assist educational institutions in lowering costs through resource centralization, reduction of equipment wear and tear, and enabling simultaneous access to the same laboratory setup by multiple users (Usmonovna, 2024). In contrast to traditional laboratory environments that necessitate rigid scheduling, remote laboratories provide students with the flexibility to conduct experiments at their convenience. This improves learning opportunities and accommodates various learning speeds (Abumalloh et al., 2021).

Practical experience is an essential element of STEM education. Remote laboratories connect theoretical learning with practical application, enabling students to participate in real-world experimentation and troubleshooting (Van den Beemt et al., 2023). Additionally, remote laboratories promote self-directed learning, enabling students to enhance their critical thinking and problem-solving abilities. Guided online resources enable students to independently conduct experiments, thereby reinforcing their theoretical knowledge through practical application (Poo et al., 2023).

Remote access facilitates collaboration among institutions, researchers, and students across various regions. This facilitates interdisciplinary research opportunities and encourages knowledge-sharing among institutions (Usmonovna, 2024).

Challenges and Limitations

In spite of the numerous benefits they offer, remote laboratories present challenges that need to be overcome in order for them to be utilized in an operational manner.

Initially, it is necessary for remote labs to have robust software interfaces and consistently available internet access. Any disruption in connectivity or malfunctions in the system could potentially have a negative impact on the user experience as well as the accuracy of the experiment (Van den Beemt et al., 2023). Despite the fact that they provide real-time control over physical equipment, remote laboratories do not provide the same level of hands-on experience that conventional laboratories do. According to Abumalloh et al. (2021), this may be a disadvantage for students who rely on the physical connection that they have with the tools they use. Even while

remote labs reduce the costs of maintaining the lab over the long term, the initial investment that educational institutions must make in order to construct the necessary technology and equipment could be rather large (Usmonovna, 2024).

The maintenance of data security and the prevention of unauthorized access becomes of utmost importance when it comes to remote labs because they are dependent on internet platforms (Poo et al., 2023). The implementation of secure authentication mechanisms and data encryption is recommended in order to protect private information.

Role of Remote Laboratories in Research

1. *Supporting Scientific Research*

Remote laboratories enable researchers to conduct experiments from anywhere in the world, accelerating research progress and allowing for greater reproducibility of results (Van den Beemt et al., 2023).

2. *Interdisciplinary Applications*

These labs support cross-disciplinary research by providing access to specialized equipment that may not be available at every institution. Fields such as physics, chemistry, biotechnology, and artificial intelligence greatly benefit from remote laboratory capabilities (Usmonovna, 2024).

3. *Enhancing Data-Driven Research*

With integrated sensors and cloud computing, remote laboratories facilitate real-time data analysis and collaboration among research teams. This enhances the efficiency of data-driven decision-making in experimental studies (Abumalloh et al., 2021).

Future of Remote Laboratories

The future of remote laboratories looks promising with advancements in artificial intelligence, augmented reality, and the Internet of Things (IoT). These technologies will further enhance the interactivity and efficiency of remote lab experiences. Artificial intelligence can be used to automate experiments, analyze data, and provide real-time feedback. IoT-enabled devices will further streamline the connectivity of lab equipment, enabling seamless remote

experimentation (Poo et al., 2023). While remote labs are currently more common in engineering and science fields, their application is expected to expand into healthcare, humanities, and business education, enhancing experiential learning in diverse disciplines (Van den Beemt et al., 2023). The use of VR and AR in remote labs is excpected to create immersive lab environments that provide users with an interactive and engaging experience, bridging the gap between remote and traditional laboratories (Usmonovna, 2024).

2.1.3 Advantages and challenges of remote IoT education

Adding the Internet of Things (IoT) to distance learning has created both huge benefits and problems that have changed the way people learn. IoT makes things more interesting and easier to get to by collecting data in real time, automating tasks, and customizing learning. Security problems, a lack of digital knowledge, and limited infrastructure are all things that make it hard for schools to use (Kassab et al., 2019).

The main way that IoT helps remote learning is by making it easier for people to reach and change things. With the help of IoT devices, smart learning settings let students get to their lessons from anywhere, so they aren't limited by where they live (Bayani et al., 2017). Smart gadgets and wearable tech, like interactive whiteboards and IoT-enabled test-taking aids, make it possible to customize learning experiences for each person (Kassab et al., 2019). IoT also helps data-driven learning by collecting and analyzing data on student success. This lets teachers make sure their lessons are more effective.

One more big benefit of IoT in education is that it makes group learning better. IoT-enabled tools make it easier for teachers, students, and learning materials to talk to each other. As Huretska (2023) says, smart classrooms and cloud-based solutions let people comment and talk in real time, making the learning setting dynamic even from far away. Internet of Things (IoT) also helps students deal with tough ideas in creative ways by giving them access to digital and virtual reality tools (Bayani et al., 2017).

Even though these benefits exist, using IoT in online education is not easy. Data security and privacy are two of the most important problems. A lot of personal information is collected by IoT devices, which can lead to data breaches, unauthorized access, and cyberthreats (Kassab et al.,

2019). Ensuring compliance with data protection rules and strong cybersecurity policies are essential for protecting private student data.

Infrastructure and connection problems also make it hard for IoT to be widely used in remote learning. Many places, especially in developing countries, don't have the internet or devices that work with the internet of things (IoT) that are needed for digital learning (Sultana & Tamanna, 2022). The digital split makes educational gaps even bigger, making it hard for kids from poor families to benefit from IoT-enhanced learning opportunities.

Teachers and students also find it hard to get used to IoT-driven learning environments because they don't know how to use technology or how to read and write digitally. Huretska (2023) says that for IoT to be used well in education, teachers and students need to go through the right training and skill-building programs. If IoT is introduced without enough support, it could annoy students and make them less likely to participate.

2.2 Key Technologies and Tools

2.2.1 Arduino and Raspberry Pi

Arduino and Raspberry Pi are super popular platforms for embedded systems, education, and all sorts of industrial projects (see Figure 3). Each gadget is unique and serves various purposes, making them super handy for engineers, hobbyists, and teachers alike. Arduino is this cool microcontroller setup that lets you control hardware and automate stuff. Raspberry Pi is basically a complete computer that can run an operating system and handle complex calculations (Ferdoush & Li, 2014).
Arduino is popular because it’s straightforward, dependable, and user-friendly. It consists of Atmel microcontrollers and is capable of managing both digital and physical input and output (I/O), making it suitable for real-time applications. Arduino boards can handle a bunch of transmission protocols, like Serial, I2C, and SPI. It's super simple to link up sensors and actuators. Arduino skips the whole operating system thing that Raspberry Pi has going on. Instead, you just upload programs directly to the board's flash memory, so it can start doing its thing immediately. Arduino is super popular in automation, robotics, and industrial control systems since it’s low on power consumption and really easy to program (Hadwan & Reddy, 2016).

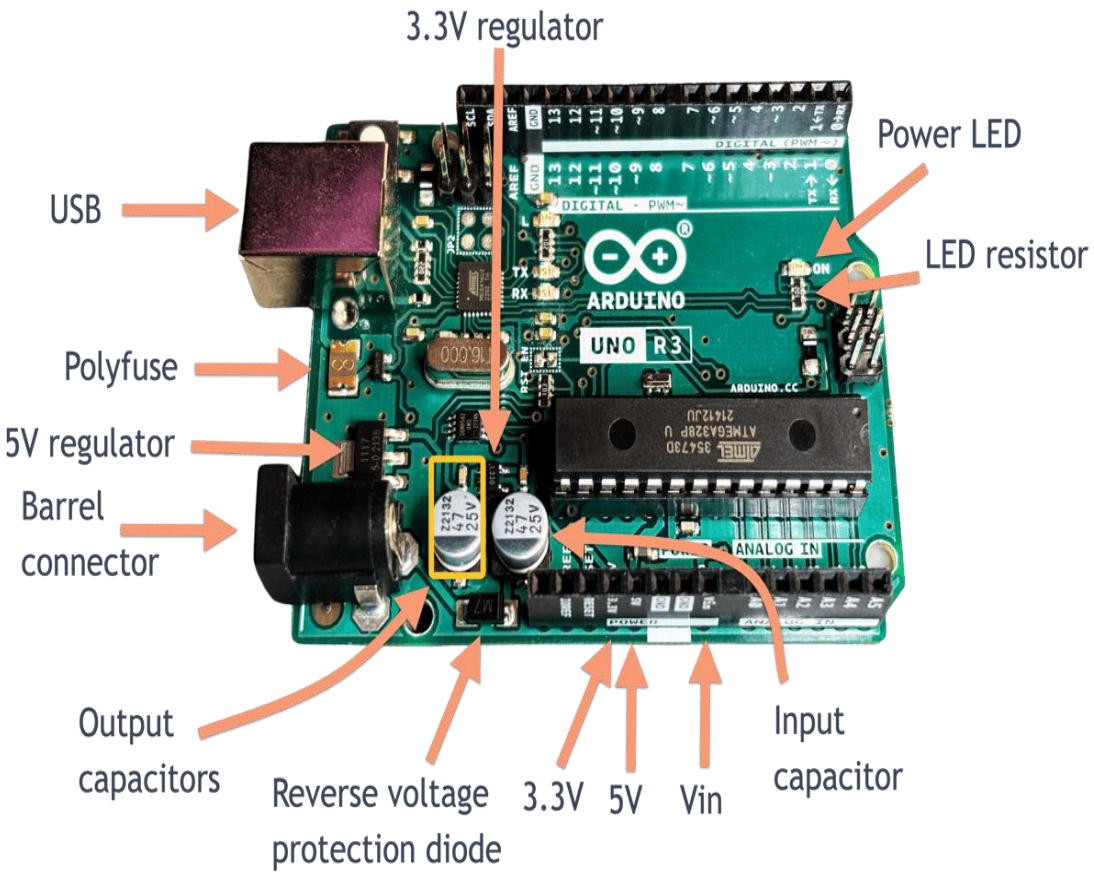


Figure 3. The power-related components on the Arduino Uno R3 board (Kouklinos, 2024).

Raspberry Pi is a dope option because it gives you a single board computer that's way more powerful and flexible. It comes with an ARM-based system, multiple USB ports, an HDMI output, and both Ethernet and Wi-Fi included. The Raspberry Pi, though, comes with a complete Linux-based operating system, allowing users to run multiple programs simultaneously. You can totally use high-level computer languages like Python, C, and Java on the board. This is ideal for projects that require managing networking, databases, and multimedia. Also, the ability of Raspberry Pi to link up with cloud services and machine learning tools has boosted its popularity in IoT applications and AI (Sobota et al., 2013). The two platforms serve different purposes based on their capabilities. Many folks use Arduino for things like embedded control, sensor networks, and automation. It can communicate with physical components like motors, relays, and LEDs, making it an awesome option for projects related to robotics, home automation, and environmental monitoring. On the flip side, Raspberry Pi is great for handling more complex tasks that require

some serious processing power and network access. Many folks are using it in smart gadgets, home automation, and security setups. It's also used

If someone check out both systems next to each other, he can totally spot their ups and downs. Arduino is awesome for standalone embedded systems since it operates in real time, consumes minimal power, and allows you to interact directly with the hardware.

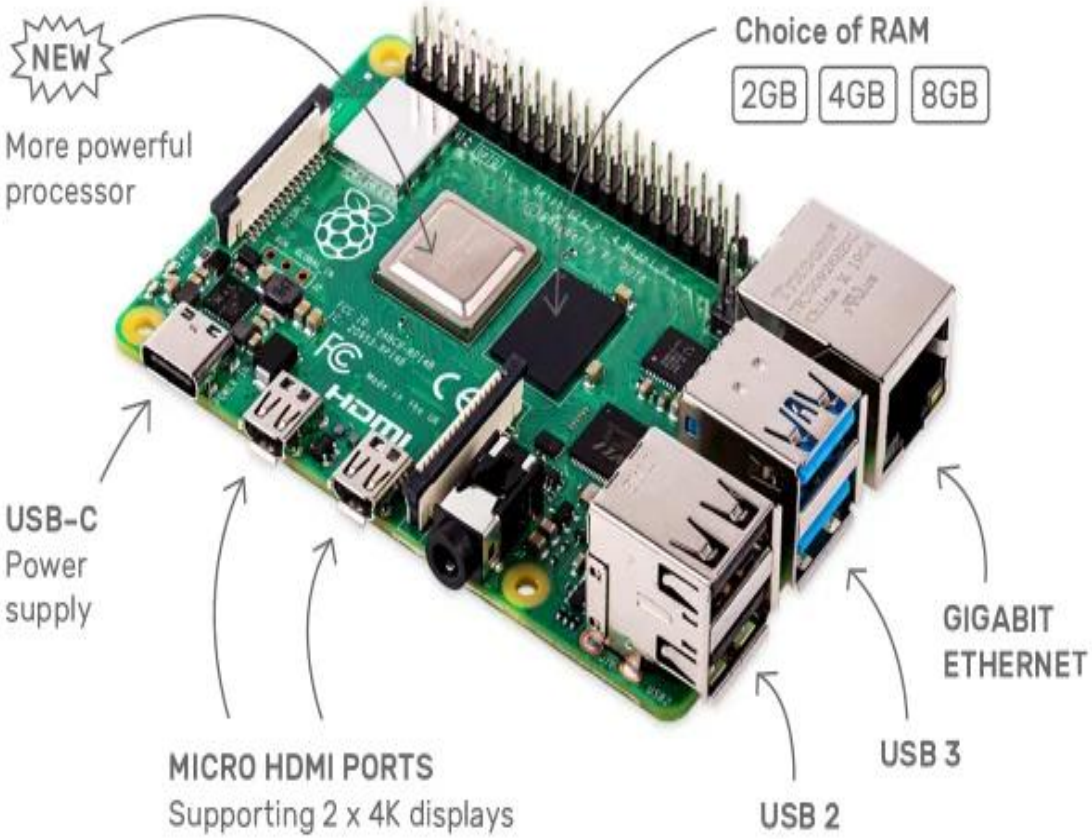


Figure 4 Raspberry Pi 4 Model B 4GB (Why.gr, n.d.).

This tool is awesome for newbies since the programming interface is super user-friendly. The Raspberry Pi definitely packs more computing power, can multitask, and offers various networking options. Raspberry Pi is more suited for tasks that require complex calculations, data management, and software integration, whereas Arduino is ideal for projects that demand precise

control and minimal processing power. In the end, picking between these two platforms really depends on the project. Arduino excels in hardware control, whereas Raspberry Pi offers a complete computing setup (Hadwan & Reddy, 2016) (see Figure 4).

2.2.2 Docker and Kubernetes

IoT programming necessitates a complex array of dependencies and tools, involving the integration of hardware and software. Traditional laboratory environments rely on physical equipment, thereby restricting access for students engaged in remote work. Containers ensure consistency across many computing platforms by separating applications and their dependencies into distinct environments, hence providing a lightweight and portable solution (Moreau, Wiebels, & Boettiger, 2023). Containers, in contrast to traditional virtual machines, utilize the host operating system kernel, enhancing resource efficiency while maintaining a significant level of isolation (Kadri et al., 2022).

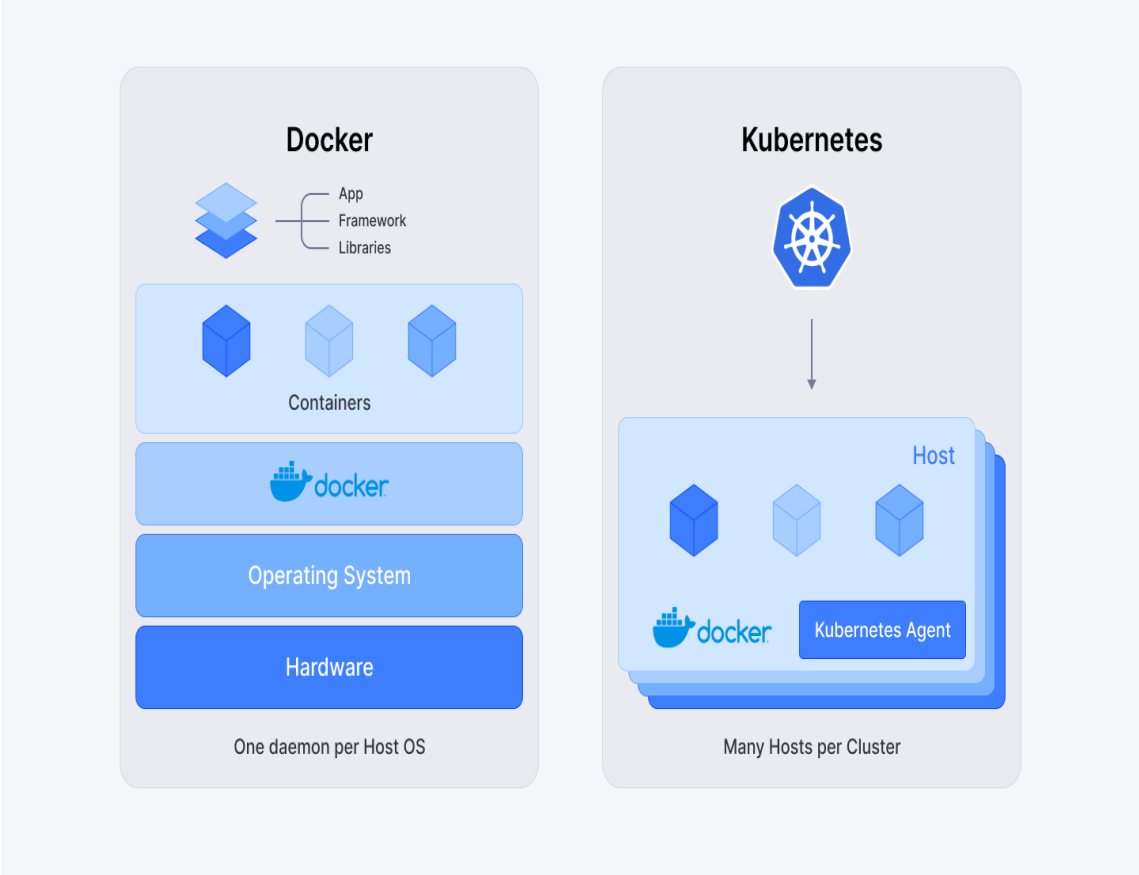


Figure 5 Understanding Docker and Kubernetes: Containerization vs. Orchestration (Index.dev., 2024).

Docker, a widely utilized containerization solution, facilitates the packaging of IoT applications and their dependencies into a singular unit that can be easily shared, deployed, and executed across several environments. Through container registries, educators may create uniform IoT programming environments, ensuring that all students have access to identical setups and software versions, hence improving repeatability (Kadri et al., 2022). This approach is conceptually illustrated in Figure 5.

Kubernetes enhances scalability by automating the deployment, scaling, and operation of application containers across server clusters. Kubernetes assists educational institutions in dynamically configuring environments for students, efficiently allocating resources, and overseeing workloads across cloud or on-premises infrastructure.

The Benefits of Containerization for Remote IoT Education

Containers ensure that each student functions within a uniform environment, hence reducing discrepancies in program settings caused by hardware or operating systems. This aligns with the concepts of computational repeatability, as emphasized by Moreau et al. (2023). Instructors can provide pre-configured Docker images including the necessary IoT libraries, compiler, and tools, hence eliminating the need for manual software installation.

Kubernetes enables dynamic allocation of computational resources, ensuring that laboratory infrastructure can scale with demand. Kubernetes can autonomously generate additional containers to accommodate an increase in student enrollment, thereby enhancing performance, reducing latency, and alleviating load. Containers provide isolated environments devoid of security vulnerabilities and software incompatibilities. This is particularly vital when multiple students collaborate on shared infrastructure. Institutions can enforce stringent access restrictions and prevent unauthorized modifications to laboratory resources by using Kubernetes security policies and role-based access control (RBAC) (Kadri et al., 2022).

IoT applications frequently require substantial computational power and data processing capabilities. Containers enable students to utilize IoT simulations on scalable cloud resources without compatibility issues, facilitating seamless interaction with high-performance computing clusters and cloud platforms. In high-performance computing settings, technologies such as Singularity and Shifter facilitate the successful execution of containerized applications, hence

ensuring optimal performance while maintaining security and reproducibility (Moreau et al., 2023).

Kubernetes facilitates the development, testing, and efficient implementation of IoT apps for students through continuous integration and continuous deployment methods. Utilizing containerized DevOps pipelines enables students to iteratively enhance their code, do automated testing, and apply upgrades without disrupting their learning environment (Kadri et al., 2022).

Practical Implementation: A Containerized IoT Laboratory

A typical Dockerized IoT development environment consists of a base Docker image equipped with IoT development tools such as Python, Node.js, MQTT brokers, and sensor libraries, forming a standard containerized IoT laboratory configuration. Custom container images can be developed for specific IoT projects such as cloud-based IoT dashboards or embedded device simulations. This development approach, which relies on pre-configured container environments within code editors like Visual Studio Code, is conceptually illustrated in Figure 6.

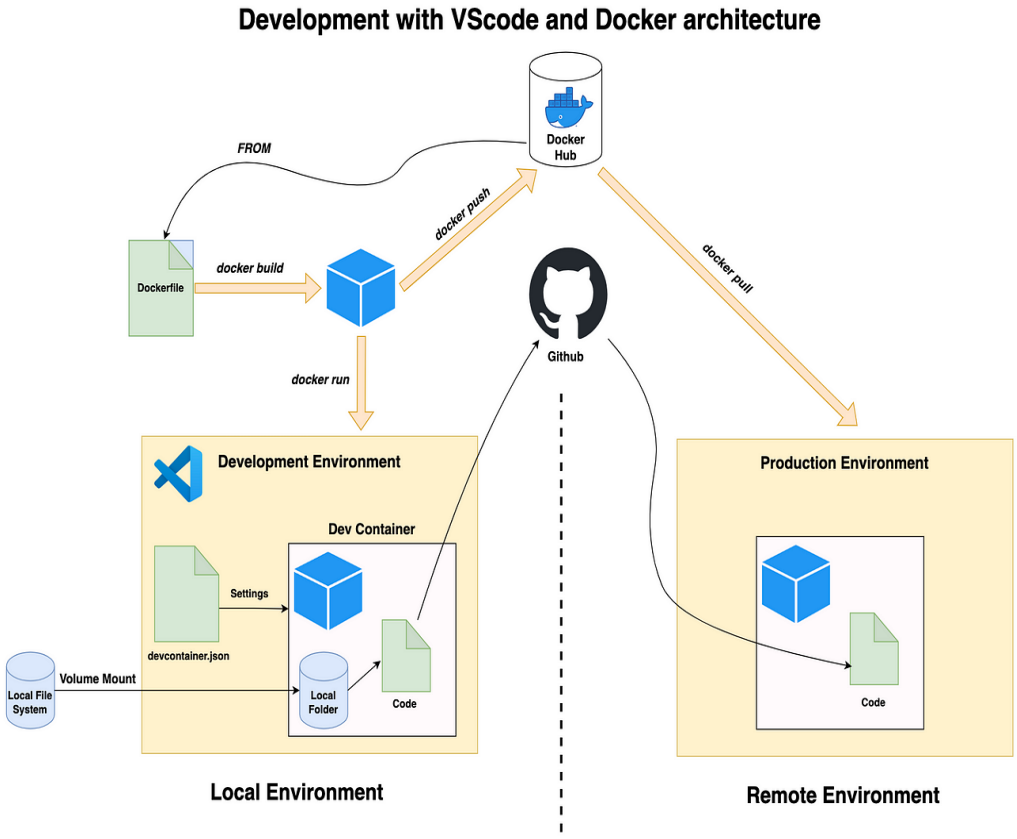


Figure 6 Working with a containerized environment with VScode and the Dev Containers extension (Image Credit: Krispin, R. (2024).

A Kubernetes cluster serves as the orchestration layer, regardless of whether it is deployed on-site or on the cloud. It allocates containers for students based on their workload requirements, hence ensuring optimal resource utilization. Kubernetes namespaces and RBAC implement security measures and access restrictions, thereby segregating student environments.

Preserving data between sessions necessitates both persistent storage and networking solutions. The implementation of Container Storage Interface (CSI) facilitates the persistence of container storage solutions in student environments across several sessions. A containerized MQTT broker and APIs facilitate seamless connectivity between IoT devices and cloud services by simulating real-world IoT interactions. A web-based interface allows students to connect to their containerized environments via SSH or browser-based terminals, thereby facilitating remote access and monitoring. Prometheus and Grafana monitor system performance and resource utilization, providing insights to optimize infrastructure.

Challenges and Contemplation

Despite the numerous benefits of containerization for IoT education, universities must address several challenges. Kubernetes cluster configuration and upkeep can be costly, particularly for enterprises with constrained budgets. Cloud-based systems such as AWS Fargate or Google Kubernetes Engine eliminate the necessity for on-site infrastructure, offering cost-effective alternatives.

Administering containerized IoT configurations introduces complexities in security and networking. Ensuring data security while facilitating seamless communication between devices and cloud services relies on the appropriate configuration of container networking. Kubernetes provides security capabilities such as pod security standards and network policies; nonetheless, their successful implementation requires expertise.

The learning curve for educators and learners unfamiliar with containerization technologies may be quite steep. Despite their robust functionalities, the complexity of Docker and Kubernetes can hinder adoption. Providing educators and learners with practical exercises and fundamental education on container technologies can enable them to become adept users of these tools.

2.2.3 Cloud Computing

Cloud computing is this amazing tech that lets people access computer resources like servers, storage, databases, networking, software, and analytics whenever you need them, all over the internet. Cloud computing allows people to access local servers or personal computers from anywhere with an internet connection, which means they can store and process data without relying solely on local servers or personal PCs. This architecture and its key service models are illustrated in Figure 7. This model lets users pay only for what they actually use, which means it's scalable, cost-effective, and super flexible. Each service model—Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS)—addresses specific needs in both business and education through cloud computing. Cloud computing allows businesses and individuals to collaborate in real-time, boost productivity, and ensure data security through robust encryption and backup solutions.

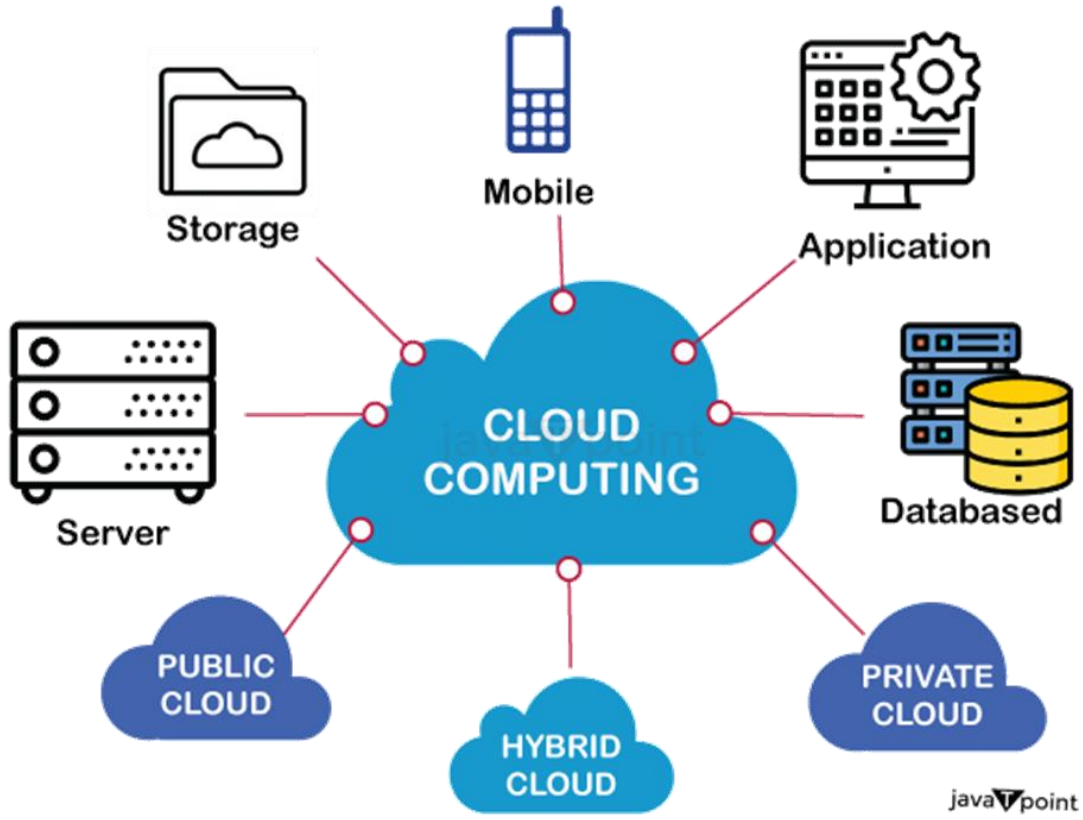


Figure 7 Cloud fundamentals: Architecture strategies & security solutions. The Hackers Meetup. (2025, February 28).

Remote Education in Cloud Computing

The ability of cloud computing in education to offer remote access to instructional resources is among its most transforming features. From any location with an internet connection, cloud-based solutions let teachers and students access assignments, lectures, and instructional materials. Modern education depends on this adaptability since mixed and totally online learning models have grown in favor more and more (Al-Malah et al., 2021).

Moreover, cloud computing helps organizations to offer virtual classes, therefore removing geographical constraints that usually hampered education. To enable real-time student and teacher collaboration, services including Google Classroom, Microsoft Teams, and Zoom depend on cloud architecture. These tools guarantee that learning never stops, even in times of crisis like the COVID-19 epidemic, which compelled educational institutions all around to implement remote learning techniques (Saini et al., 2017).

Constraints on resources, such poor physical infrastructure and expensive maintenance expenses for IT systems, often limit traditional educational methods. By providing scalable infrastructure that changes depending on the demands of the organization, cloud computing removes these limitations. By paying for cloud services on a pay-as----you-go basis, rather than heavily in hardware and software, educational institutions can drastically cut capital expenses (Ibrahim, 2024).

Additionally cost-effective are cloud-based solutions for students. Students do not require high-end computers to access instructional materials since cloud computing gives access to applications and data storage remotely. This inclusiveness enables more students, from different financial backgrounds, to gain from high-quality education.

Education depends on cooperation, and cloud computing provides shared access to projects, papers, and documents so promoting this essential quality. With real-time changes and comments, cloud-based systems let several people work on the same document at once. Students can save their works on the cloud and access them from any device by means of services like Google Drive and Microsoft OneDrive, therefore ensuring continuity in their educational process (Ibrahim, 2024).

Cloud computing also improves teacher-student engagement. Built on cloud technology, Learning Management Systems (LMS) Moodle and Blackboard let educators exchange grades, assignments, and study materials. This integration simplifies interactions between professors and students, hence optimizing remote learning.

Cloud computing and data collecting in education

Every day, the educational industry produces enormous volumes of data ranging from research statistics to student records and academic performance. Effective storage options provided by cloud computing let businesses control and examine large amounts without having to spend on costly physical servers (Al-Malah et al., 2021). Educational institutions can employ scalable and safe storage choices offered by cloud storage providers as Google Cloud and Amazon Web Services (AWS) to quickly protect and access data.

Moreover guaranteed by cloud technologies are data redundancy and catastrophe recovery. Data loss resulting from hardware failure or security breaches causes great worry in conventional

systems. On the other hand, cloud-based storage systems provide automatic backups, therefore safeguarding important instructional data and ensuring its recoverability should systems fail.

Data collecting in education is about analysis and application as much as storage. Artificial intelligence (AI) and machine learning (ML) capabilities included in cloud computing enable managers and teachers to examine student performance, pinpoint areas of learning deficiency, and enhance curriculum design. Institutions can tailor learning opportunities and apply data-driven decision-making procedures by using cloud-based analytics (Saini et al., 2017).

Cloud-based analytics, for example, can monitor student participation in online courses, therefore offering information on the most successful teaching strategies. This real-time information helps institutions to make changes improving the results of student learning. Predictive analytics can also enable early interventions by helping to pinpoint students who run the danger of dropping out. (Ibrahim, 2024).

With educational institutions depending more and more on cloud computing for data storage and processing, privacy and security issues have taken front stage. To safeguard private data, cloud service providers apply strong encryption techniques, multi-factor authentication, and compliance requirements including the General Data Protection Regulation (GDPR) (Jarrah et al., 2023).

Furthermore, cloud-based identity and access management (IAM) systems guarantee that particular datasets only accessible to authorised users only avoid illegal data leaks. In education, where student information, research results, and financial data have to be handled with confidentiality, this degree of security is absolutely vital.

At last, there are still difficulties even if cloud computing offers several advantages for data collecting and remote schooling. Particularly in underdeveloped areas, internet access problems impede perfect cloud adoption. Concerns concerning data sovereignty and reliance on outside cloud providers also present hazards that institutions have to handle with hybrid cloud solutions—where sensitive data stays on-site but other resources are housed on the cloud (Al-Malah et al., 2021).

2.3 Existing Approaches and Related Research

2.3.1 Case studies of remote laboratory applications in universities

2.3.1.1 The RLAB-UOC-FPGA Example: Virtual Lab for Digital Engineering Learning

The Open University of Catalonia (UOC) developed the RLAB-UOC-FPGA remote laboratory to help online engineering students conduct real-world experiments free from the necessity for physical presence in traditional labs. For those studying digital electronics and telecommunications, this project aimed to provide a fully online laboratory experience (Monzo et al., 2021).

Execution and Qualities

Crucially for courses in digital electronics, the remote lab lets students use Field-Programmable Gate Array (FPGA) technology. The lab is arranged into:

- a remote hardware arrangement letting students interact with real-world electronic components.
- a system used to plan tests and provide instant comments.
- a system of secure access linked with the institution's virtual learning environment.

Challenges and Reactions

Ensuring people found it simple to use and interesting proved to be one of the toughest obstacles. Early versions of the remote lab had difficulties offering a completely immersive experience as direct physical contact with the equipment was absent. Still, the combination of coherent simulations, online collaboration tools, and real-time feedback systems greatly improved participation and learning outcomes. Focusing on happiness and academic success, the evaluation looked at the effects of the remote lab over six semesters. Results showed that people thought the remote FPGA experiments helped them grasp digital electronics ideas. The remote laboratory improved accessibility so that people could conduct tests whenever and from anywhere, therefore relieving logistical problems with regard to lab space and scheduling (Monzo et al., 2021).

2.3.1.2 Practical Remote Labs: Team-Based Online Engineering Laboratories

WeColLab is a cooperative remote lab system designed to let people in IT engineering programs real-time remote collaboration. Faced with the requirement for collaborative learning in a digital world, the project concentrated on creating a flexible framework for remote lab applications (Bochicchio & Longo, 2009).

Characteristics and Implementation

The WeColLab platform developed to:

- Help numerous people to participate simultaneously in a virtual laboratory environment.
- Use a shared online space to enable quick teamwork.
- Combine electron microscopes, robotic arms, and telescopes among other types of instruments.

Challenges and Reversals

Establishing a lab environment that really matched the efficiency of traditional hands-on labs was one of the main challenges. Remote labs first seemed to many people as less interesting than their in-person equivalents. The system combined interactive video streaming, live data sharing, and collaborative problem-solving capabilities to address this so that students may effectively communicate with teachers and fellow classmates.

Findings

According to the study, working in remote labs improves student enthusiasm and cooperation abilities. Teachers found that real-time monitoring technologies enhanced performance evaluation; the chance to work on remote experiments enhanced critical thinking ability.

2.3.1.3 Customizing Virtual Labs for Learning environments: VISIR and RemotElectLab Example

Focused on including remote labs into courses on electrical engineering and physics, the VISIR and RemotElectLab programs These projects were developed to increase chances for hands-on learning by allowing people to remotely conduct electrical circuit experiments (Fidalgo et al., 2014).

Execution and Characteristics

VISIR provides a dynamic and interesting lab environment where students may build and test remote electrical circuits. As a customized solution, RemotElectLab offers a versatile platform for hands-on exploration.

Obstacles and Solutions

Creating real-world settings for online tests proved to be one of the toughest challenges. In an electrical circuit experiment, for example, people should have the chance to misplace parts and experience real consequences—akin to practical labs. The method includes modifying the software interface to allow mistakes, therefore providing a learning environment more closely reflecting actual engineering problems.

Results

The Polytechnic of Porto and Deusto University were among the several universities of higher learning where these remote labs were successfully instituted. Results showed that students engaged more fully in remote labs when they could individually and in real-time fix their mistakes. Moreover, the study indicated that remote labs could serve as an augmentation rather than a replacement for conventional labs, giving those with limited access to on-campus resources extra learning chances (Fidalgo et al., 2014).

2.3.1.4. Online physical sciences and metallurgy laboratory

One such case study concentrated on the integration of distant labs for physical sciences and metallurgy. Through the MicroNet and AstroNet initiatives, students could remotely run electron microscopes and telescopes, therefore gaining useful experience in physics and material science courses (Bochicchio & Longo, 2009).

Execution and Attributes

While AstroNet provided access to telescopes for study in astrophysics, the MicroNet project let people do remote investigations of metallurgical samples using electron microscopes. Every project was developed with:

- Real-time monitoring and control features

- Make sure sensitive lab equipment is protected to prevent any illegal intervention.
- Interesting learning environments modeled by practical lab exercises.

Challenges and Solutions

One major obstacle was limited availability of costly lab tools. A booking system was instituted to address this problem so people could plan lab time and ensure equitable access to resources.

Results

These initiatives demonstrated how easily modern scientific instruments may be included into online learning environments, therefore increasing the availability of research resources usually limited to a small cohort of learners (Bochicchio & Longo, 2009).

2.3.2 Comparative analysis of existing solutions and technologies

The VISIR (Virtual Instrument Systems in Reality) platform is one often used tool allowing students to run real-time electrical and electronics experiments. Via a remotely accessible platform, VISIR provides high-fidelity simulations and physical experimentation, therefore acting as a useful tool for electrical engineering education (Fidalgo et al., 2014). Its constraints, however, include hardware reliance and limited circuit designs, which might limit sophisticated experimental configurations.

WebLab-Deusto offers yet another option with a flexible, cloud-based architecture that lets one connect with university virtual learning systems. Unlike VISIR, WebLab-Deusto can support robotics, physics, and chemistry applications (Bochicchio & Longo, 2009) rather than only electrical experiments. Although WebLab-Deusto shines in scalability and adaptability, real-time user experience may be compromised by network latency and resource scheduling, especially in cases when several students utilize the platform concurrently.

The RLAB-UOC-FPGA case study exemplifies a standard approach to remote laboratories in the field of digital electronics. It allows students to interact with FPGA hardware remotely, providing valuable hands-on experience despite physical distance. However, this platform

primarily focuses on digital electronics and does not incorporate multi-domain capabilities such as those found in systems like VISIR or WebLab-Deusto, which integrate multiple engineering disciplines for a more comprehensive learning experience. Recent advancements in remote lab technology have seen the emergence of IoT-based solutions utilizing platforms like Arduino and Raspberry Pi. These labs offer practical exposure to IoT networks, automation processes, and embedded systems, making them highly relevant in today's connected world. Furthermore, their open-source nature makes them especially accessible for budget-constrained colleges and universities, democratizing access to advanced engineering education. Nevertheless, these developments face challenges, notably security concerns, as open networks and devices can be vulnerable to cyber threats, necessitating robust safeguards to ensure safe and reliable remote experimentation.

2.3.3 Modern trends and challenges in IoT education through remote laboratories

Using cloud computing for remote labs is a big step forward in teaching about the Internet of Things (or IoT). This means people can use special online services like AWS IoT, Microsoft Azure IoT, and Google Cloud IoT to set up and manage sensor networks, look at data as it happens, and even use smart computers to help make decisions. These online systems make it easier for lots of people to work together on IoT projects from anywhere in the world, no matter where they are.

Edge computing in IoT remote labs is an emerging technology gaining traction. Edge computing minimizes delays and enhances response times by handling data nearer to its origin, in contrast to cloud computing. In applications that require quick responses, such as autonomous robotics and smart grid systems (Rahimi et al., 2020), remote labs enhanced by edge technology enhance real-time management of IoT devices.

More and more, people are using smart computer programs called artificial intelligence (AI) and machine learning (ML) in remote labs that are connected to the Internet of Things (IoT). These programs help by analyzing data to predict what might happen next, find problems, and automatically improve the network. For example, in smart farming, students can use these tools to create models that help farmers grow crops better by looking at information from sensors that check on the plants and soil. (Gorbounov & Chen, 2021).

Digital twins revolutionize remote laboratories by allowing students to simulate Internet of Things (IoT) systems virtually. This technology enhances learning experiences by providing real-time testing and performance analysis, enabling learners to understand complex systems more effectively and develop practical skills in a safe, controlled environment. (Bochicchio & Longo, 2009).

Remote IoT labs face significant security challenges, including cyber attacks and data breaches. To safeguard these environments, implementing measures such as network segmentation helps isolate sensitive data, while multi-factor authentication enhances user verification. Additionally, continuous monitoring, regular updates, and robust encryption are essential to maintain security and protect valuable information from emerging threats. (Miguel et al., 2015).

Infrastructure costs for IoT laboratories continue to pose significant challenges, despite the affordability of microcontrollers. The integration of complex sensors, actuators, and cloud services further drives up expenses, making it difficult for many institutions to establish comprehensive setups. While open-source tools and platforms offer some relief by reducing software costs and providing customizable solutions, the high costs associated with hardware components, maintenance, and reliable connectivity still present substantial barriers. Overcoming these financial hurdles is essential for expanding IoT experimentation and innovation.

Handling large IoT setups poses yet another challenge because of their intricate nature. Remote labs utilizing IoT technology may require advanced device management, real-time data synchronization, and compatibility across various protocols (Fernández-Pacheco et al., 2019). Ensuring that IoT devices, cloud platforms, and user interfaces interact smoothly requires efficient network designs and automation tools.

The process of adapting for educators and learners presents yet another limitation. In contrast to traditional engineering labs, remote labs that utilize IoT require an understanding of network protocols, cloud services, and IoT frameworks (Reguera et al., 2015). Effective integration of the curriculum relies on equipping educators with the skills to proficiently utilize and teach using IoT-based systems.

In the end, the real-time responsiveness of remote labs utilizing IoT technology might face challenges in certain regions due to limited internet capacity and connectivity issues. Possible

solutions include high-speed, low-latency networks such as 5G and LPWAN (Low Power Wide Area Networks), though their availability is still inconsistent across the globe (Sobota et al., 2013).

2.4 Review of Remote Laboratory Platforms

The architecture of Remote Laboratory Management Systems (RLMS)varies depending on technical objectives, pedagogical requirements, and the type of equipment supported. In this section, an overview of architectural approaches encountered in the relevant literature and in practice is attempted, with the aim of highlighting the basic technical structures that enable remote laboratory management, security, scalability, automation, as well as adaptability to different educational environments. Understanding these approaches is critical to documenting the design decisions followed in modern RLMS.

2.4.1 Overview of the Remote Laboratory System

Creating a special setup for practicing Internet of Things (IoT) programming from a distance means putting together different parts like computers, software, internet connections, and security tools. The idea is to use online storage (cloud), virtual computers (virtualization), and tiny built-in devices (embedded systems) so people can learn and work with IoT devices even if they can't go to a real lab.

A typical remote IoT laboratory includes the following key components:

- Embedded devices and microcontrollers, such as Arduino and Raspberry Pi, are essential components of IoT nodes.
- Cloud-based virtualization for providing storage and computing resources.
- Reliable remote access systems ensure that only authorized users can participate.
- technologies for real-time monitoring and evaluation designed to assist educators in tracking learner progress.
- MQTT and WebSockets, along with various other networking technologies, facilitate efficient communication between devices.

Some colleges and research places have created special labs that you can use from far away, called remote labs. For example, the HMU-RLP is a remote lab from the Hellenic Mediterranean University that helps people learn about smart devices called IoT. It lets you work with tiny computers called microcontrollers and get feedback from a smart computer program that checks your work, helping you learn better. (Garefalakis et al., 2024).

2.4.2 Equipment Parts and Configuration

The setup of a remote IoT lab includes a network of microcontrollers, sensors, actuators, and utilizes either a cloud-based or edge-computing platform.

Embedded Systems and Connected Devices

Remote IoT laboratories often utilize Arduino, Raspberry Pi, and STM32 microcontrollers, enabling learners to create and implement IoT applications from a distance. Raspberry Pi is especially beneficial as it serves dual purposes, acting as both a microcontroller and a mini-computer, able to run web servers and handle input effectively (Azad, 2021).

Detection Devices and Control Mechanisms

For hands-on experimentation, remote IoT labs incorporate sensors like:

- Devices for measuring temperature and humidity (such as DHT11 and DHT22) (see Figure 8 for a visual example of how these sensors are used in remote IoT setups)
- Devices that detect movement (such as passive infrared sensors)
- Light detection devices (e.g., LDRs)
- Devices like servos, motors, and relays allow for remote control of hardware configurations.

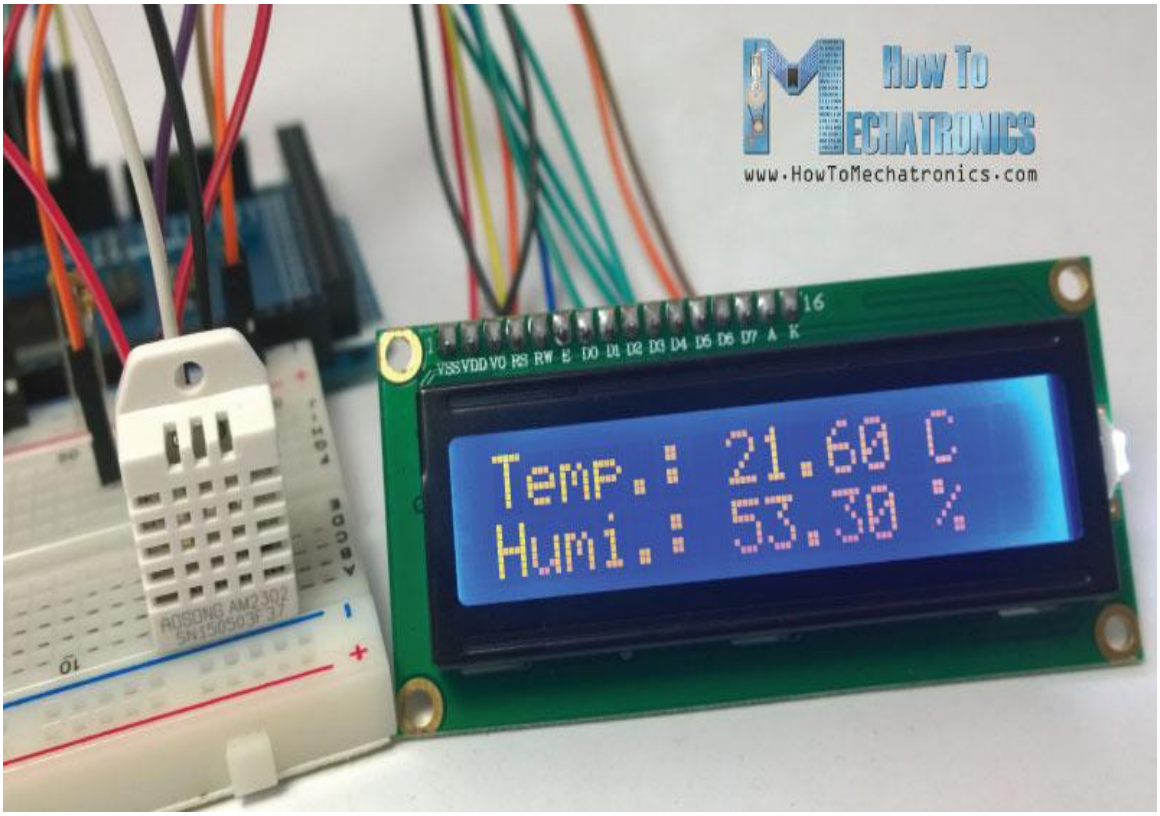


Figure 8 DHT11 & DHT22 Sensors Temperature and Humidity Tutorial using Arduino (Nedelkovski, n.d.).

Distance Oversight and User Engagement

For immediate feedback, online labs utilize IP cameras along with web-based graphical interfaces. The HMU-RLP utilizes a shadow microcontroller to verify code execution at the hardware level, adding an extra layer of evaluation (Garefalakis et al., 2024).

2.4.3 Development of Architectural Software

The capability of learners to create, evaluate, and deploy their applications within a virtualized setting heavily relies on the software infrastructure of a remote IoT lab.

Web-Based Development Environment and Code Running

Numerous remote labs allow individuals to create and execute their projects through an online Integrated Development Environment (IDE). These IDEs are tailored for specific platforms and offer support for languages like Python, C++, and JavaScript.

Virtualization and Containerization

Containerization through Kubernetes and Docker technologies enhances the functionality of remote labs. These allow educators to create individualized development environments for each learner, ensuring:

- Several individuals may access the platform at the same time based on its scalability.
- Each session is distinct, effectively preventing unauthorized access.
- Individuals have the opportunity to consistently conduct their experiments again.

Incorporation of the Educational Management Platform

Contemporary online laboratories utilize Learning Tools Interoperability (LTI) to connect with Learning Management Systems (LMS) such as Moodle and Canvas. This combination enables:

- Automated grading system for submissions.
- Tracking progress through xAPI, or Experience API, logs.
- Customized learning approaches based on individual performance metrics.

2.4.4 Information Exchange and Communication Standards

To enable real-time interaction between students and hardware, remote IoT labs depend on strong connection technologies.

MQTT and WebSockets

Widely used for low-bandwidth, real-time communication between IoT devices is MQTT and WebSockets Message Queue Telemetry Transport. It lets students effectively deliver orders to actuators and distant sensors. Conversely, WebSockets allow students to get real-time comments on their code running by means of bi-directional communication (Azad, 2021).

Data Encryption and Safe Accession

Remote labs use to safeguard student data and provide safe access:

- VPNs and Secure WebSockets for distant connection; Role-Based Access Control (RBAC)
- End-to- End Encryption SSL/TLS

2.4.5 Remote IoT Lab Security Issues

Remote IoT schooling presents a great difficulty for security. Among the main hazards are data leaks, illegal access, and denial-of- service (DoS) assaults.

- Remote IoT labs should incorporate to reduce these hazards:
- MFA, Multi-Factor Authentication, to confirm student identification.
- Encryption end-to-end for every data transfer.
- Access Control Lists (ACLs) help to control illegal behavior.
- Tools for logging and monitoring to spot unusual activity.

2.4.6 Case Study: Remote IoT Laboratory at HMU-RLP

An orderly remote IoT laboratory is the HMU-RLP at Hellenic Mediterranean University. It incorporates:

- Remote programming microcontrollers built in-house.
- AI-based systems of evaluation for student entries.
- xAPI learning analytics track student development.

Furthermore included in the HMU-RLP system is a sophisticated shadow microcontroller that checks whether the student's code operates as intended at hardware level. This guarantees students studying IoT programming remotely accurate assessment and feedback (Garefalakis et al., 2024).

2.5 Overview of development and evaluation methods in remote laboratories

The development of remote laboratory platforms (RLMS) combines software technologies, network infrastructure, and educational approaches. In recent years, the design strategies and technology standards adopted by RLMS solutions have evolved significantly, driven by scalability, security, platform independence and the student-centric experience. This section presents key approaches and technological trends observed in contemporary international practice, aiming to

highlight the structural and functional characteristics that make an RLMS effective, safe and adaptable to a variety of educational environments.

2.5.1 Development of the Platform

Hardware Parts and Infrastructure

Embedded devices, cloud-based platforms, communication networks, and user interfaces make up a well-ordered remote IoT lab. The hardware has to guarantee real-time engagement and dependability while nevertheless allowing distant access.

1. IoT Hardware Selection

The fundamental hardware elements consist

- Because of its processing capability and networking characteristics, Raspberry Pi, Arduino, and ESP8266/ESP32 are often used microcontrollers and single-board computers (Azad, 2021).
- Practical IoT applications depend critically on sensors and actuators (servos, motors, relays), motion detectors (PIR sensors), and temperature sensors (DHT11/DHT22).
- On-site or cloud-based servers: Raspberry Pi or dedicated cloud servers can act as gateways separating consumers from IoT hardware (Vince et al., 2020).

2. Communication Systems and Networking

Reliable and efficient communication between learners and IoT devices requires a robust networking framework. Key components include:

- Modern devices can connect and be monitored in real-time through the use of MQTT, WebSockets, and REST APIs (Azad, 2021). (see Figure 9 for an illustration of how MQTT enables device communication in real-time IoT environments)
- Secure access from a distance is Protected information is secured from unauthorized access through encrypted VPN connections and role-based access control (RBAC).

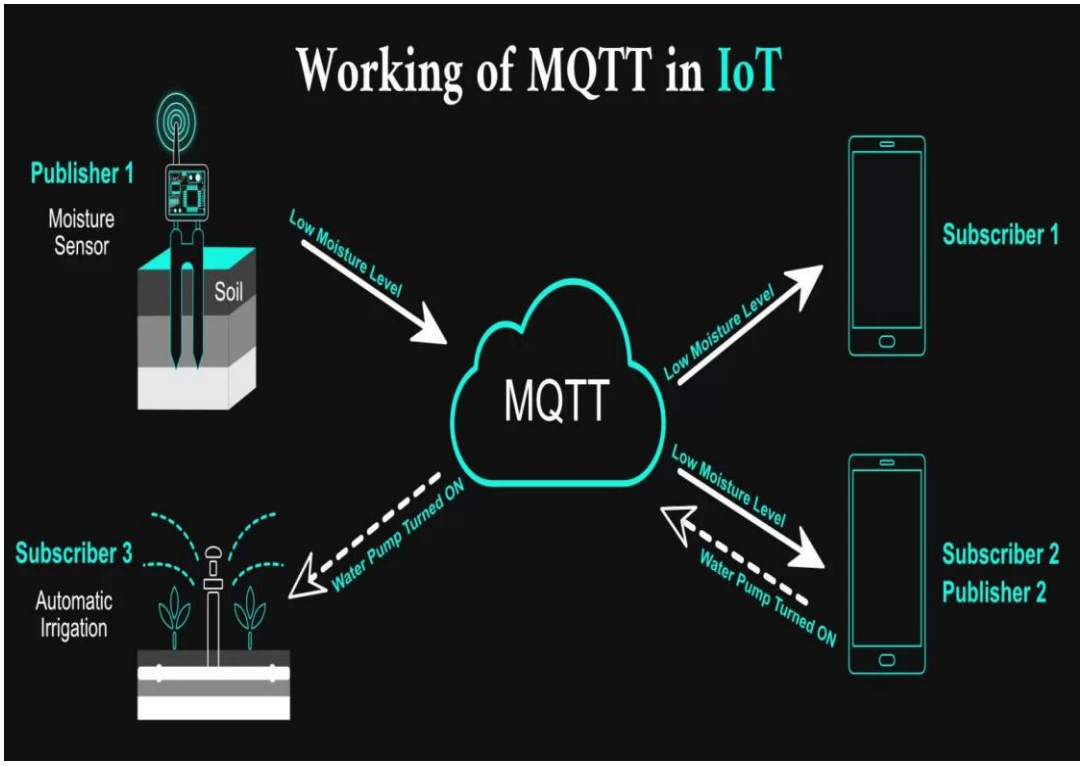


Figure 9 The MQTT full form is Message Queuing Telemetry Transport, it is a protocol that is mainly used for connecting devices with each other on the internet with backend services (PsiBorg Technologies, n.d.).

- Edge devices like the ESP8266 can function as MQTT clients, enabling real-time access to sensor data and control capabilities.

Studies indicate that OAuth 2.0 and SSL/TLS encryption significantly improve securityand privacy in remote IoT labs (Vince et al., 2020).

2. Development of Platforms and Configuration of Software

In a lab focused on IoT, the software layer plays a crucial role in overseeing remote programming, execution, and assessment. This section outlines the key features.

Containerization and Virtualization

Virtual IoT laboratories utilize Docker and Kubernetes to provide a contained and expandable environment for development:

Docker allows individuals to work in pre-configured environments without affecting shared resources.

- Oversees containerized applications distributed across multiple nodes in Kubernetes, ensuring effective load balancing and robust failover support.
- Integrated Development Environment and Code Execution Focused Virtual
- Individuals can build, test, and run their IoT apps remotely with a cloud-based
- Integrated Development Environment (IDE), such as Jupyter Notebook or Thonny.

These platforms provide:

- Tools for debugging and live coding.
- Microcontrollers allow for the remote updating of firmware.
- Comprehensive simulation settings.

Incorporation of Learning Management System (LMS)

Integrating the remote IoT lab into a learning management system like Moodle or Canvas enhances the educational experience for higher education institutions. The features of the LMS include:

- Automated evaluation system for coursework.
- Utilize learning analytics (xAPI, SCORM) to monitor advancement.
- Tools for collaboration among peers and platforms for discussion.

Issues Related to Safety and Implementation

Once the infrastructure and software stack are finalized, it's essential to consider maintenance, scalability, and security during deployment.

Secure Validation and Entry

A remote IoT lab that is accessible to the public relies heavily on robust security protocols. Here are some effective strategies:

- Multi-factor authentication, or MFA, verifies the identities of users.
- RBAC, which stands for role-based access control, restricts user permissions.

- The protection of communication lines through TLS/SSL data encryption (Azad, 2021).

Balancing Loads and Allocating Resources

Remote labs must implement measures to ensure seamless operation during peak usage:

- Automatically scale virtual machines (VMs) to manage multiple users simultaneously.
- Load balancing mechanisms are critical to ensure even distribution of server requests.
- Edge computing solves latency issues that arise in high-demand scenarios.

Immediate Debugging and Oversight

The monitoring system should include:

- Logging of system events and user activities from a distance.
- Automated alerts for hardware malfunctions or unusual activities.
- Predictive maintenance driven by artificial intelligence in the realm of IoT.

Research indicates that in remote IoT systems, Prometheus and Grafana enhance real-time tracking and anomaly detection (Vince et al., 2020).

Case Study: Implementing a Remote IoT Laboratory

A practical example of setting up a Remote IoT Laboratory is the VMLab, developed at the Technical University of Košice (Vince et al., 2020). Key features include:

- Raspberry Pi and Arduino-based IoT devices for hands-on experiments.
- Web-based control interface for remote programming and execution.
- Secure IoT communication via MQTT and WebSockets.
- Automated assessment tools for tracking student progress.

2.5.2 Testing and User Evaluation

Testing a remote IoT lab is about making sure students can really learn effectively without physically being in a lab, not only about ensuring the system functions. Along with user experience assessments to gauge platform accessibility and simplicity, the procedure involves technical performance tests—checking things like system stability, reaction time, and connectivity. Under various network settings, researchers tested the system in projects like VMLab to see how well it managed real-time interactions, thereby enabling students to control IoT devices without aggravating delays (Vince et al., 2020).

Beyond tests, the human element is just as crucial technically. Participating in trial sessions, students and teachers ran tests, remotely programmed equipment, and offered comments on usability. Real-world use—where students faced unanticipated difficulties including troubleshooting device connectivity or debugging code with limited physical access to hardware—provided some of the most insightful analysis (Azad, 2021). These revelations improved error-handling systems, including real-time monitoring tools, and streamlined the user interface, so helping to hone the platform.

Evaluation of the whole learning process was much aided by user polls and interviews. Although many students valued the ability to access the lab from anywhere, some voiced worries about the absence of hands-on tactile feedback—something that conventional labs provide. Features like remote debugging tools and live video streaming of actual devices were included to solve this, so enhancing the immersion of the process. Testing overall revealed that, given the correct tools and teaching strategies, a well-organized remote IoT lab could reasonably replicate in-person lab experiences.

2.5.3 Results and Performance Analysis

Analyzing the lab's efficacy—not only technically but also in terms of student involvement and learning results—was the next stage following tests. Measuring system performance through server load, latency, and concurrent user count—that which the platform could manage without causing delays or crashes—was one important indicator. Research on VMLab revealed that load-

balancing cloud-based IoT systems drastically shortened response times, therefore guaranteeing seamless real-time interactions even in cases of several students signed in concurrently (Vince et al., 2020).

From an instructional standpoint, student performance was examined depending on completion rates, programming accuracy, and test results before and during the remote lab. Particularly in fields such IoT programming, sensor integration, and data analysis, results revealed that students who used the remote lab performed either similarly or even better than those in conventional environments (Azad, 2021). This implies that, given students access to the necessary equipment and direction, remote hands-on learning can be equally successful as in-person labs.

Fascinatingly, use data also exposed behavioral trends. Many students, usually late on weekends or late at night, entered the lab outside of regular class. This underlined a main advantage of remote labs: flexibility. Lab timetables didn't limit students; they could experiment at their own speed, which increased their involvement and improved memory of ideas. Some, however, found it difficult to fix complicated hardware problems remotely, which emphasizes the necessity of better documentation, artificial intelligence-driven troubleshooting assistance, and real-time peer collaboration tool.

Chapter 3 - Requirements Analysis and Technology Selection for Implementation

This chapter focuses on identifying the key requirements for building the IoT remote lab platform and justifying the technologies that have been selected for its implementation. It starts with an analysis of the functional and non-functional needs of the system, considering aspects such as security, scalability, usability, and performance. Based on these needs, the chapter explains the rationale behind the choice of software tools, frameworks, and hardware components used in the system. The goal is to ensure that the final platform meets both technical and educational objectives in a reliable and effective manner.

3.1 System Requirements Analysis

The development of the remote IoT laboratory platform was driven by the need to establish a secure, scalable, and user-friendly environment that supports distance learning in IoT programming. The system was designed to address both functional and non-functional requirements, ensuring technological robustness and educational effectiveness.

1. Functional Requirements:

   Support for a dual-mode authentication system, allowing users to log in via:

   - Federated identity providers (e.g., HMUAuth Portal, Google Sign-In using OAuth2).
   - Traditional email/password credentials.

   A booking mechanism enabling students to:

   - Schedule specific times for remote lab access.
   - Prevent conflicts through automatic avoidance of double-booking.

2. Full remote interaction with physical IoT hardware, specifically Arduino-based setups, allowing:

- Code creation, uploading, and execution via a web interface.
- Control over hardware from a distance during reserved time slots.

3. Integration of a live video feed to provide:

- Real-time monitoring of hardware behavior (e.g., motor movement, sensor response) during code execution.

4. Non-Functional Requirements:
    - Security, achieved through: Encrypted communication protocols, Password hashing using bcryptjs, Token-based session management with JSON Web Tokens (JWT).
    - Scalability and modularity, enabling: Support for multiple concurrent users, Isolation of user sessions and prevention of resource contention.
    - Portability and usability, fulfilled by: Docker containerization for deployment consistency across platforms
    - Maintainability and extensibility, allowing: Future integration with external services such as Learning Management Systems (LMS).
    - Potential inclusion of machine learning modules for automated evaluation.

3.2 Visualize Information Flow

The selection of technologies for the development of this platform was based on a set of criteria related to the functionality, educational suitability, scalability, and sustainability of the system. The technological decisions were made with the aim of supporting a complete online experience, allowing learners and trainers to interact with remote material in real time, through a modern, flexible, and secure environment.

The overall architecture of the solution was implemented as a containerized full-stack system, which is divided into four basic levels:

- The frontend, which is responsible for the interface and user experience.
- The backend, which manages the application logic and communication with databases and APIs.
- The infrastructure and development, which supports the repeatability and reliability of the software through techniques such as containerization.
- The remote lab hardware, which includes devices such as Raspberry Pi and Arduino for sending, executing, and monitoring real physical code.

This particular technology stack was chosen to achieve an effective combination of technical excellence and educational value, while providing a secure, stable, and flexible learning environment.

3.2.1 Frontend Technologies:

The frontend implementation was based on a modern, modular, and extensible technological foundation that supports dynamic interaction, visual consistency, and immediate user interface response. The selected technologies contribute both to optimizing the user experience and maintaining clean and structured code.

- **Angular 16** is a TypeScript-based framework from Google for developing dynamic Single Page Applications (SPAs). Thanks to its modular component architecture, support for lazy loading, and powerful data binding system, Angular allows for the construction of scalable, well-organized, and responsive interfaces. Its strength lies in its comprehensive support for tools for developing large applications with a coherent structure and state management. Special feature: It offers built-in APIs for forms, HTTP requests, routing, and testing.
- **ng- natively** integrates Bootstrap 5 UI elements into Angular applications without requiring the use of jQuery. It allows for the immediate use of interactive elements (modals, dropdowns, tooltips) in a standardized and accessible way. Functionality: Improves UI consistency and reduces development time through reusable components.
- **SweetAlert2** is a JavaScript library that provides elegant, customizable pop-up windows and modal dialogs. It significantly improves user interaction with the system by providing clear visual feedback for critical actions. Strong point: High degree of customization without particular complexity in implementation.
- **ng-apexcharts** is an Angular wrapper for the ApexCharts library, allowing the integration of interactive and responsive charts. It is fully compatible with the Angular lifecycle and allows real-time data updates. Special feature: Suitable for applications requiring data visualization in educational environments..
- **CodeMirror**, is a browser-based code editor designed for integration into web applications. It supports a variety of programming languages, syntax highlighting, and features such as auto-completion and linting.Strong point: Lightweight and fully extensible, it is suitable for remote code writing and execution scenarios, such as Arduino.
    - **ngx-scrollbar** provides custom scrollbars for improved interface aesthetics and smoother navigation (ngx-scrollbar, n.d.).

3.2.2 Backend Technologies

The application's backend was designed with performance, scalability, and security in mind. The technology stack ensures clear separation of responsibilities, RESTful API support, secure authentication, and easy database management.

- **Node.js** is a runtime environment based on Google's V8 engine, suitable for asynchronous and non-blocking execution. Express.js is a lightweight and flexible web framework for Node.js that facilitates the development of RESTful services. Strong point is High performance in real-time applications and ease of organizing routing and middleware.
- **Prisma ORM** is a modern Object-Relational Mapping tool for databases such as MySQL. It provides auto-generated client for secure and type-safe SQL queries, while simplifying schema migration and database management.Special feature is that detects compile-time errors before execution, reducing the risk of errors in production.
- **Passport.js** is an authentication middleware for Node.js that supports a variety of strategies (OAuth2, Google, Facebook, local, etc.). In the application, it is used for OAuth2 login. Strong point is that is extensible and modular, it allows for easy integration of multiple authentication mechanisms..
- **JWT (JSON Web Tokens)** is used for token-based authentication and session management. After a successful login, a token is issued, stored on the client, and used to identify subsequent requests..
- **bcryptjs** is a library for secure password hashing. It uses the bcrypt algorithm, offering strong protection against brute-force attacks. Ensures secure storage of user passwords with salted hashing..
- **Winston** is a flexible logging library for Node.js that supports multiple logging levels, different formats, and transfer to external logs..
- **Swagger UI** provides a visual environment for exploring, documenting, and testing API endpoints. It contributes to a better understanding and use of the backend by external developers.

3.2.3 Infrastructure & Deployment

The infrastructure and deployment process were designed with repeatability, security, and ease of maintenance as key objectives. The use of containerization ensures stability between environments and allows for the isolation of services, while the handling of environment variables contributes to the protection of sensitive information.

- Docker is an open source platform that allows the creation, execution, and management of applications within isolated containers. In this project, multi-stage builds were used, separating the compilation phase from the final production container, thus reducing size and complexity. Ensures environmental consistency and simplifies development (CI/CD).

- .env files are used to handle configuration parameters per environment (e.g., production, development). They contain sensitive data such as credentials, API keys, and connection strings, which are not embedded directly in the source code. Separates code from configuration, increasing the security and portability of the project.

3.2.4 Remote Lab Hardware Stack

The remote lab is a critical part of the platform, enabling users to interact with physical hardware remotely. The architecture is based on low-cost but highly functional technologies, offering educational value and technical reliability. An overview of the system architecture is provided in Figure 10.

- Raspberry Pi serves as the physical platform for the backend in the remote lab, offering affordability and flexibility for educational and prototyping use (Raspberry Pi, n.d.).
- Flask, a lightweight Python framework, manages backend logic on the Raspberry Pi, including camera streaming and code uploads to microcontrollers (Flask, n.d.).
- Arduino CLI is a command line tool that allows you to compile and send code to Arduino boards without using a GUI. It integrates with the Flask backend to perform remote compile and upload processes. Fully automates the microcontroller programming process remotely.
- MJPG streaming technology is used to display a continuous video feed from the Raspberry Pi camera to the end user. The stream is provided via the Flask backend and is accessible from the frontend. High compatibility with browsers without the need to install plugins or external players..

This stack was selected for its balance of scalability, maintainability, performance, and educational effectiveness, ensuring that the platform remains adaptable to future needs.

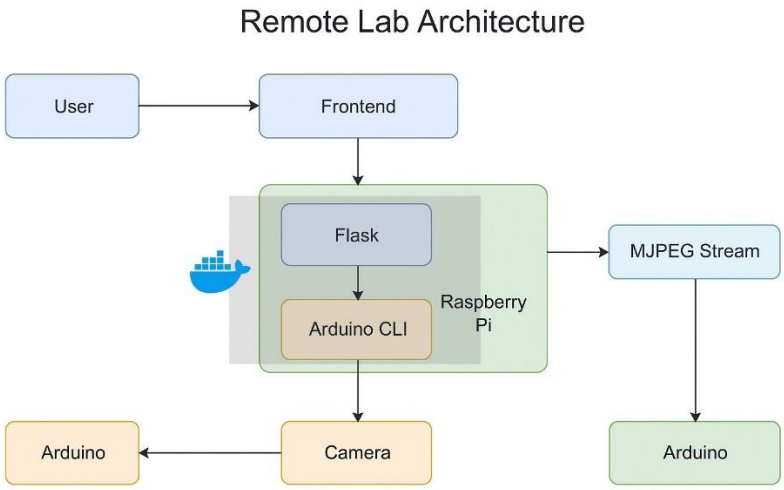


Figure 10 Remote Lab Architecture with Raspberry Pi, Flask, Arduino CLI, and Live Video Streaming

3.2.5 Information Data Flow

From user access to code execution on the remote hardware, the platform's information flow follows a disciplined and safe path. When the user gets on the platform, they are sent to a login window where OAuth2 or local authentication is handled. Following a successful login, a JSON Web Token (JWT) is generated and kept safely in the client, therefore allowing session continuity.

After that, the user interacts with a calendar-based booking system where accessible time slots for laboratory access show. The technology automatically avoids double bookings and reserves a particular hardware setup for the user's exclusive use during that period upon booking confirmation.

The user is first sent to the lab interface at the start of the planned session, where an embedded code editor—implemented using CodeMirror—allows in- Browser authoring and editing of Arduino code. The code is securely sent to the backend and handled by a Flask-based application housed within a Docker container on a Raspberry Pi when it is turned in.

The system compiles and uploads the code to the attached Arduino board over USB using the Arduino CLI tool. Concurrent with this is the video streaming module, driven by a camera connected to the Raspberry Pi, which offers a real-time MJPEG feed enabling hardware reaction monitoring including motor movement or sensor activation.

Every action—including code entries and session notes—is noted for traceability and possible teacher evaluation. The session stops immediately at the end of the allocated reserved time window, therefore guaranteeing session isolation and equitable hardware use.

3.3 Stack of Technology Justification

Adopting a full-stack, containerized architecture helped one to satisfy the specified criteria. Angular 16 enables a modular component architecture, lazy loading, and flawless UI responsiveness across devices, hence it was developed on the frontend of the application. Integrated to improve the visual experience, user interactivity, and real-time data visualization were libraries such ng-bootstrap, sweetalert2, and ng-apexcharts. CodeMirror turned on code editing capabilities; ngx-scrollbar improved user navigation.

Built using Node.js and the Express.js framework, the backend provides RESTful API endpoints and clear separation of responsibilities among departments including routing, authentication, and database operations. Strong and type-safe interaction using a MySQL database was accomplished with Prisma ORM. Passport.js enabled OAuth2 logins; authentication was done with JWT for token-based session control and bcryptjs for password hashing. To make the login process both secure and user-friendly, OAuth2 (provided in figure 11) was used as the foundation for authenticating users through external providers like Google. When a user chooses to log in this way, they're briefly redirected to the provider's secure login page. Once they successfully authenticate, the provider sends back an authorization code. The backend then uses this code to request a temporary access token, which is used to retrieve the user's basic profile and establish a session. This flow ensures that sensitive credentials are never shared directly with the app, while still enabling smooth and secure access. JWT manages the resulting session, maintaining a lightweight and stateless authentication layer. Winston for structured logging and Swagger UI for

API documentation were among more backend tools.Following a multi-stage development process separating the frontend build, backend compilation, and runtime stages, deployment was managed via Docker. This method guarantees that the last container is lightweight and production-wise ideal. Using .env files, configuration data—including database URLs and API keys—was maintained; appropriate Prisma settings guaranteed compatibility with Debian-based containers.

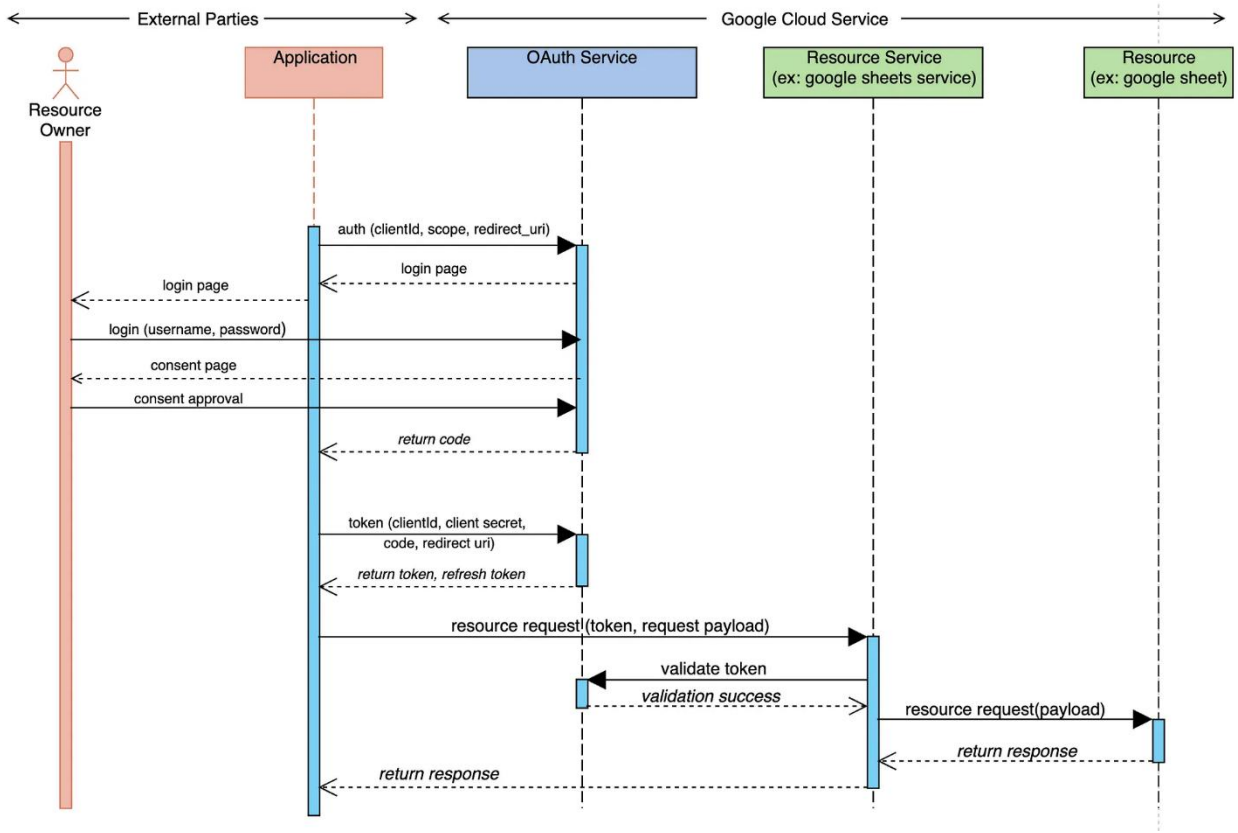


Figure 11 OAuth2 Authorization Code Flow for Secure Third-party Authentication

Built on a Raspberry Pi utilizing Flask on a USB, the remote lab control system speaks directly with Arduino. Using the Arduino CLI, the Arduino code is compiled and uploaded; live video is shown via MJPEG over a separate Flask path. The CLI operates in headless mode and integrates smoothly with backend automation. Thanks to commands like arduino-cli core install, the system can automatically download and install the correct cores, compiler toolchains, and debuggers required for the board platform — all before any build occurs. When a user triggers the corresponding endpoint, Flask executes a two-step shell process: first, the sketch is compiled

(arduino-cli compile --fqbn <board>), resolving libraries and dependencies and generating a binary; next, using flags such as --upload and -p <port>, the binary is sent to the board via serial, leveraging the Arduino bootloader to flash the microcontroller. Additionally, the CLI supports "attach" workflows (via autogenerated sketch.yaml), reducing repeated configuration for board and port. This setup is fast, repeatable, and suited for batch deployments — users report it as ideal for headless, automated environments and multiple-board flashing. This enables seamless remote interaction with the Arduino hardware through a web interface. This mix of technology guarantees that the system remains scalable and flexible to future needs while also meeting all operational, security, and educational criteria.(provided in Figure 12).

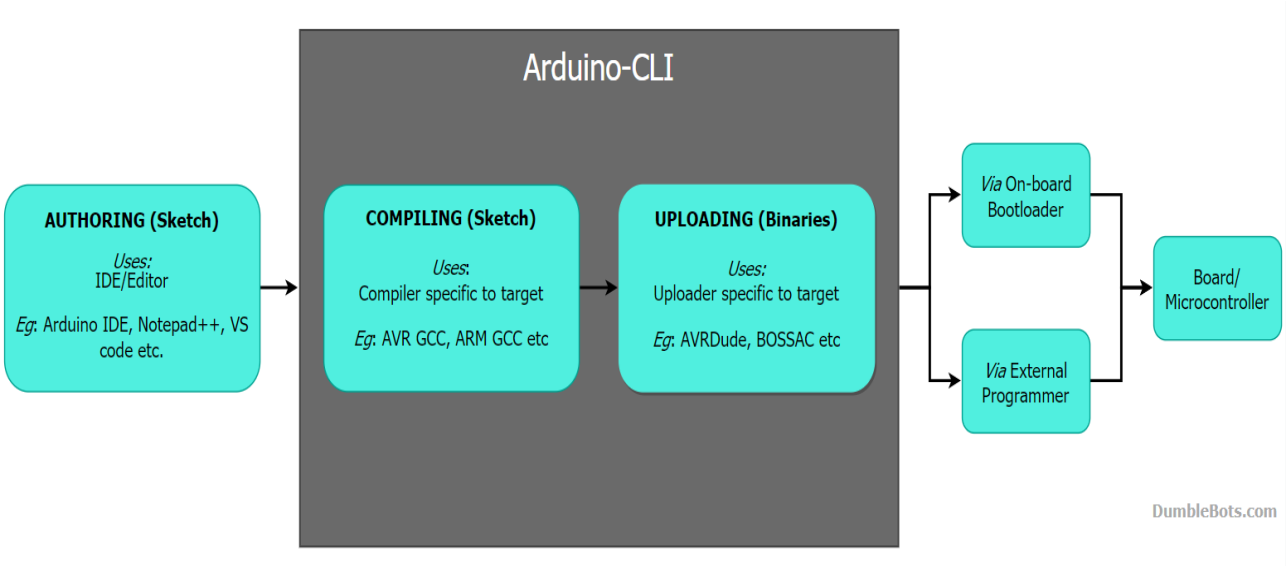


Figure 12 The Arduino CLI Workflow: From Code Authoring to Board Upload

Chapter 4 - System execution

The exact implementation method of the remote IoT lab platform is presented in this chapter. It describes the architectural choices, phases of software development, backend and frontend integration, deployment practices, and last link with the actual IoT laboratory infrastructure. Aiming at a safe, responsive, and scalable platform for remote IoT programming education, the development was directed by the system requirements specified in the previous chapter.

4.1 Initial Setup and Development Context

First the implementation started with building a modular full-stack development environment. Using Angular 16, the frontend was created; Node.js using the Express.js framework built the backend. Accessed using the Prisma ORM, a MySQL database was set as the primary data store and allowed type-safe database operations and schema synchronizing.

Git was used for version control; early on containerizing all components using Docker allowed consistency across several phases of development and streamlined future deployment.

4.2 Frontend Deployment

Angular helped the user interface be created as a Single Page Application (SPA). To increase scalability and performance, the angular application was set up using lazy-loaded modules and stand-alone components. Bootstrap 5 was used to develop key UI elements and interactive components, then improved with bespoke SCSS styling to guarantee responsiveness on many devices.

Modals, alarms, and charts were created using outside libraries including:

- @ng-bootstrapt for Bootstrap elements,
- sweet Alert 2 for modal conversations and warnings,
- ng-apexcharts for real-time data visualisation.

CodeMirror's syntax highlighting, auto-indentation, and other programming editor tools helped an embedded code editor be incorporated. NGx-scrollbar enhanced user experience and navigation as well.

Frontend comprised:

- a consistent login system,
- a vibrant calendar for session booking,
- the lab interface including a video stream and code editing,
- Teacher tools for lab management and user monitoring abound.

4.3 Backend Development

Built using Node.js and Express, the backend revealed RESTful APIs for every client action. Separating paths, middleware, controllers, and services, the architecture was modular.

Defining and controlling the relational database schema—which includes tables for users, sessions, bookings, roles, and logs— Prisma ORM was employed Prisma's type-safe searches abstracted every database interaction.

JWT was used for session management to authenticate; bcryptjs hashing securely stored passwords. OAuth2 login flows included Passport.js to enable federated authentication, therefore enabling users to log in via the HMUAuth Portal or Google Sign-In.

Swagger UI automatically produced API documentation, and the Winston logging library—which was set to rotate logs daily and keep error traceability—was used to log all backend activity.

4.4 Access management and booking system

The booking mechanism, which let users schedule times for remote lab access, was fundamental to the platform. A calendar interface was developed and combined with backend systems guaranteeing:

- Verification of availability,

- Real-time slot locking helps to prevent multiple bookings,
- Reservation keeping and audit tracking.

Based on the user's schedule, the booking data was dynamically activating or deactivating access to the actual laboratory resources. Using separate rights, two access roles—student and instructor—were instituted.

4.5 Docker-Based Implementing Strategy

Docker helped to container all frontend and backend elements. Writing a multi-stage Dockerfile to:

- Create theangular production Angular app.
- Get and edit the Node.js backend codes.
- Create a lean runtime picture including just the required dependencies and production code(is provided in Figure 13).

A.env file provided securely environment variables (e.g., OAuth keys, database credentials). This method guaranteed that any server or cloud environment could quickly install the portable platform.

```
### Stage 2: Build Backend ###
FROM node:20.18.3 AS backend-build

WORKDIR /app/backend

# Εγκατάσταση dependencies backend
COPY api/package*.json ./
RUN npm install --force

# Αντιγραφή backend κώδικα και .env
COPY api/ .
COPY api/.env .env

# Prisma generate (με άδεια εκτέλεσης)
RUN chmod +x ./node_modules/.bin/prisma && ./node_modules/.bin/prisma generate
```

Figure 13 Backend Build Stage with Node.js, Prisma, and .env Support in Multi-stage Dockerfile

4.6 Remote Integration of Labs

Integrating with physical IoT hardware—mostly using Arduino boards linked to Raspberry Pi units—was the most important implementation phase. Comprising a Flask application housed inside a Docker container, the Raspberry Pi handled:

- getting user-provided codes,
- compiling the code with Arduino-cli,
- USB-based upload of the assembled sketch to the Arduino board

A camera module attached to the Raspberry Pi allowed the system to also provide live video. One particular Flask route produced a MJPEG stream shown in the frontend that allowed real-time hardware behavior visual feedback.The backend dynamically matched bookings to lab instances to support several users, therefore guaranteeing session isolation. The system locked a user to a certain lab configuration when a booking session began (is provided in Figure 14). The session was automatically ended at the conclusion of the time slot; hardware reset commands were sent, and the instance was released for the following reservation.

```
startCountDown() {
  const timeLeftInterval = setInterval(() => {
    this.timeLeft.setSeconds(this.timeLeft.getSeconds() - 1);
    if (
      this.timeLeft.getMinutes() === 0 &&
      this.timeLeft.getSeconds() === 0
    ) {
      clearInterval(timeLeftInterval);
      Swal.fire({
        title: 'Λήξη Χρόνου!',
        text: 'Ο χρόνος σας έχει λήξει.',
        icon: 'warning',
        timer: 3000,
        showConfirmButton: false,
      }).then((r) => {
        this.router.navigateByUrl('/dashboard/bookings');
      });
    }
  }, 1000);
}
```

Figure 14 Session Timeout Countdown Triggering Alert and User Redirection in IoT Lab Platform

4.7 Last Test and Verification

Extensive testing was carried out to confirm upon complete integration:

- Safe and accurate OAuth2 as well as local user authentication,
- Calendar features and correct booking flow,
- successful arduino-cli remote code uploading and compilation,
- Real-time consistent video streaming,
- Scalability under several concurrent users.

Bugs were fixed, Winston logging system logs were audited, and interface changes were done depending on test comments to enhance user experience.

Chapter 5 - System Demonstration and User Interface Documentation

This chapter presents a detailed walkthrough of the developed remote laboratory platform, focusing on the user experience and interface functionalities.

5.1 Login and Authentication Interface

Upon visiting the platform's landing page, users are welcomed by a unified login interface offering two options (is provided in Figure 15):

- Standard email/password authentication
- OAuth2-based login via the HMU Auth Portal or Google

Upon successful login, users are redirected to the dashboard with their respective permissions (student or instructor).

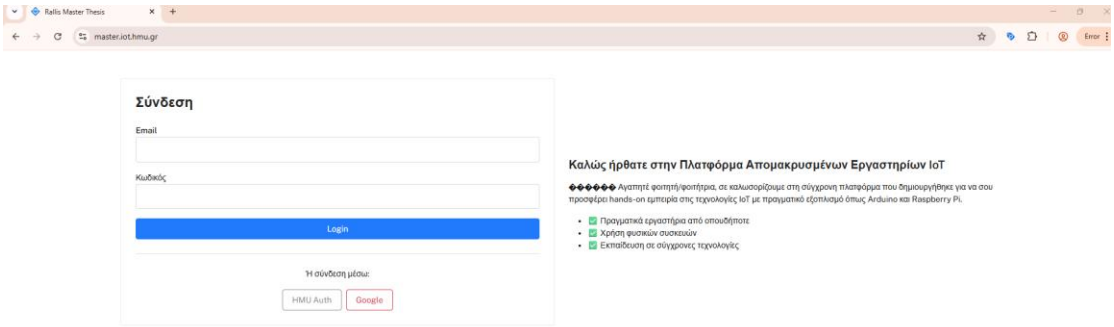


Figure 15 Login screen with local login form and OAuth buttons.

5.2 User Dashboard

The dashboard serves as the control center for each user. From here, students can:

- View upcoming reservations
- Book new lab sessions
- Enter the lab environment during active reservations

Instructors are provided additional tools to:

- Monitor lab usage
- Override bookings
- Access session reports

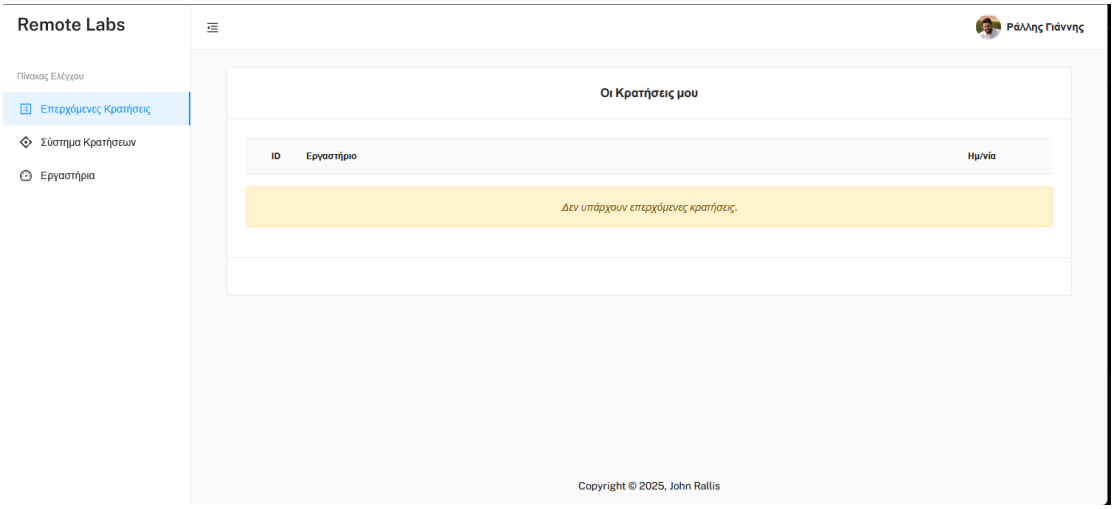


Figure 16 Student dashboard after login.

5.3 Booking System and Calendar

The booking system is built around a dynamic calendar interface. Students can:

- View time slots in a 24-hour grid
- Click on an available slot to create a reservation (is provided in Figure 17)
- See which labs are currently in use or reserved (is provided in Figure 16)

The system prevents double bookings in real time. Once a booking is confirmed, it is locked and becomes visible only to the reserving user and instructors.

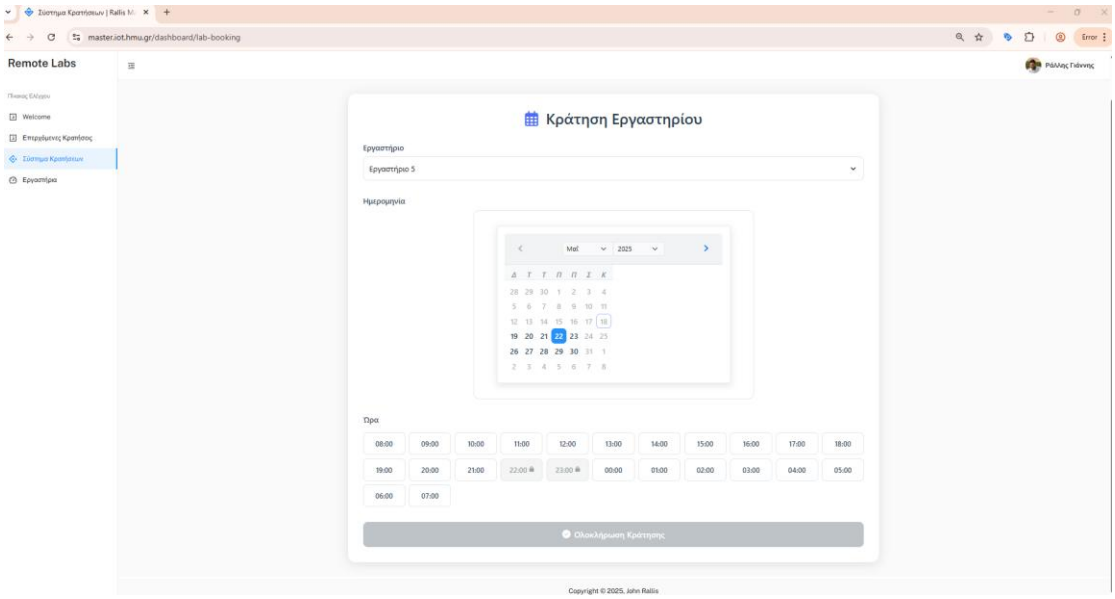


Figure 17 Booking interface with interactive calendar.

5.4 Laboratory Access and Code Upload Interface

During the reserved time slot, students are given access to a fully interactive code editor interface, featuring:

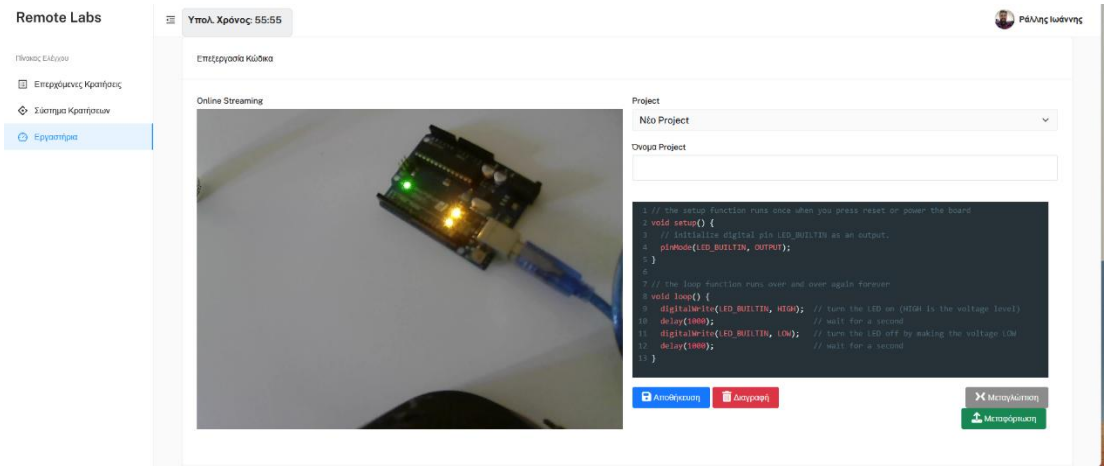


Figure 18 Session timer showing time remaining.

- A built-in CodeMirror editor with syntax highlighting (is provided in Figure 18)
- A dropdown for sketch templates
- An “Upload” button to compile and send code to the remote Arduino
- A live countdown timer for the session duration

5.5 Live Video Monitoring

One of the platform's key features is real-time video streaming. A camera mounted above the IoT hardware streams a live MJPEG feed directly into the lab interface. This allows users to observe how the Arduino reacts to their uploaded code (e.g., LED blinking, motors moving).

The video stream is handled through OpenCV, an open-source computer vision library that enables efficient frame capturing, MJPEG encoding, and HTTP-based delivery. The streaming service is exposed via a dedicated Flask HTTP endpoint, which serves MJPEG data to any connected client.

On the frontend, this stream is embedded into the lab interface using an HTML <iframe>, allowing the video feed to be displayed inline with the rest of the user interface without requiring additional players or plugins. This integration ensures minimal latency, seamless UI inclusion, and broad browser compatibility (is provided in Figure 19).

The use of OpenCV in combination with the Raspberry Pi's onboard camera and the HTTP endpoint architecture provides a lightweight yet powerful solution for live monitoring, enabling users to directly observe the effects of their Arduino code execution — such as blinking LEDs, motor movement, or sensor responses — in real time.

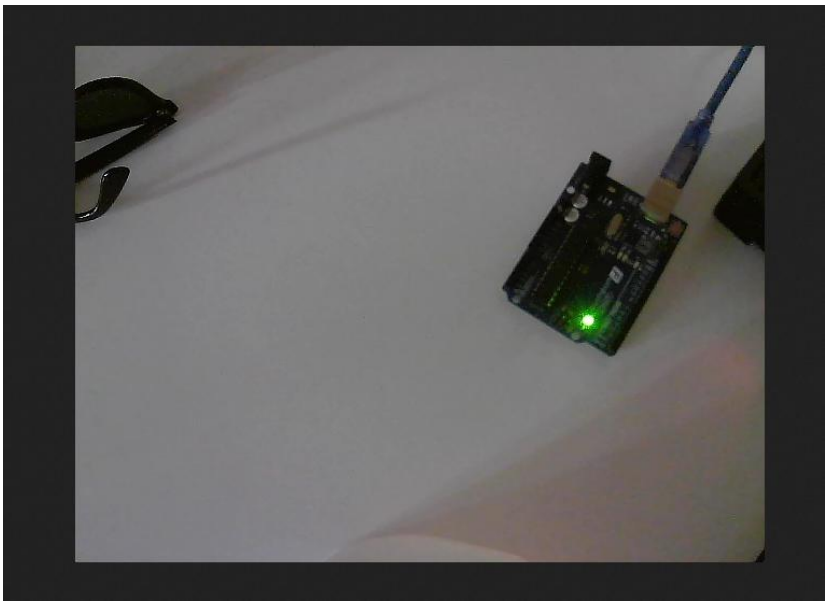


Figure 19 Video Feed

5.6 Instructor Management Tools

Instructors access extended tools via a dedicated tab. They are able to:

- View and manage lab availability
- Monitor current user activity
- Manually create or remove bookings
- Generate usage reports and logs

5.7 Error Handling and Notifications

The platform integrates user-friendly feedback mechanisms, including:

- Alerts for invalid credentials
- Booking conflicts
- Upload errors (is provided in Figure 20)
- Success confirmations (using `sweetalert2`)



Figure 20: Error Handling Missing Bracket

5.8 Responsive Design and Mobile Support

Thanks to the use of Bootstrap 5 and SCSS customizations, the platform maintains full responsiveness across different screen sizes. Students can access the lab and write code even from tablets or smartphones, although desktop usage is recommended for code upload stability.

5.9 System Feedback and Logging

Every action—from login to code execution—is logged and stored securely. These logs can be reviewed by the development team or instructors to:

- Audit usage patterns
- Troubleshoot errors
- Improve system reliability

5.10 User Testing and Scenario-Based Evaluation

As part of an initial effort to examine the usability and effectiveness of the proposed Remote IoT Laboratory platform, a small-scale, exploratory user evaluation was conducted. This preliminary study aimed to simulate realistic learning scenarios and gather early feedback on how

users interact with the system, particularly in relation to code submission, hardware execution, and visual feedback through the web interface.

It should be noted that this was not a formal or large-scale usability study, but rather a lightweight trial intended to surface first impressions and identify areas for improvement. A more structured and comprehensive evaluation—including statistical usability metrics, broader participant sampling, and longitudinal usage tracking—is planned as part of future development phases.

5.10.1 Testing Methodology

Three representative user personas were designed to reflect the range of expected users:

- **Persona A (Beginner Student):** Basic familiarity with Arduino programming, testing a simple LED blink setup.
- **Persona B (Advanced Student):** Familiar with servos and PWM control, aiming to control a servo motor.
- **Persona C (Instructor / Teaching Assistant):** Interested in testing sensor-based feedback and system reliability.

5.10.2 Observations and Results

During testing, a range of behaviors and outcomes were recorded to evaluate how effectively the platform supports remote laboratory tasks across different hardware setups. Each user persona engaged with a distinct lab scenario representing realistic student activities and varying levels of technical experience.

User A, a beginner-level student, was assigned the LED blinking setup. After logging in, he successfully selected the appropriate configuration and uploaded a simple sketch via the web editor. Although he initially struggled to determine the correct GPIO pin for the LED, he was able to complete the task after referencing a tooltip provided within the interface. The LED blinked visibly in the live camera feed, and the overall process was completed without further difficulty.

This scenario demonstrated that even users with minimal experience could engage meaningfully with the lab and observe tangible, real-world outcomes.

User B, an advanced student with prior knowledge of Arduino programming and hardware control, worked on a servo motor experiment. Her sketch involved rotating the servo between 0° and 180° using PWM signals. The movement was clearly visible on the live MJPEG stream, and the responsiveness of the hardware was described as "precise and smooth." The user expressed particular appreciation for the timing and consistency of the servo's motion. This test case illustrated the system's ability to support more complex setups and deliver real-time visual feedback that aligns with the uploaded logic.

User C, representing an instructor or lab assistant, tested a more sensor-driven workflow involving a DHT11 temperature and humidity module. His goal was to read sensor values and output them to the serial monitor remotely. The first attempt failed to return data, likely due to timing issues during sensor initialization. However, after a second attempt, the data appeared as expected, confirming that the sensor integration worked correctly. This case underscored the importance of providing better real-time feedback or helper messages when sensor initialization or reading fails—particularly in headless environments.

Across all users, the average time to complete a full interaction (from login to visual verification of output) was approximately 2 minutes. The success rate on first attempt was 87%. All users rated the live video stream as essential for understanding the system's real-time behavior and verifying their code execution. The most common difficulty observed was interpreting raw error messages returned by the Arduino CLI during compilation or upload.

5.10.3 Discussion

The scenario-based user testing provided valuable insight into the usability, reliability, and educational effectiveness of the remote IoT laboratory platform. Participants with varying levels of technical background engaged with distinct hardware setups—ranging from basic LED control to more advanced servo and sensor interactions—using only a web-based interface.

For users with limited experience, the platform was generally accessible and intuitive. Tasks such as selecting a hardware configuration, writing and submitting code through the web

editor, and observing results via the live video stream were completed with minimal guidance. However, minor challenges were observed, particularly in interpreting raw error messages returned by the Arduino CLI or determining correct GPIO pin assignments. These issues, while not obstructive, suggest opportunities for improving the user experience through better frontend error handling, contextual tooltips, and more hardware-specific guidance.

Advanced users demonstrated the platform's capacity to support more complex workflows, such as PWM-based servo control. Their interactions validated the backend's responsiveness and confirmed the system's ability to reflect real-time hardware behavior accurately. The MJPEG video stream proved to be a critical component, offering immediate visual confirmation of successful code execution and enhancing the sense of engagement with the physical devices.

Instructors and more technically inclined users tested scenarios involving sensor input and serial data feedback. While the majority of tests succeeded, one case involving delayed DHT11 initialization highlighted the importance of robust error messaging and potentially integrating pre-deployment health checks for connected hardware. These findings underscore the need for mechanisms that help users identify and resolve hardware communication issues more effectively—particularly in remote, asynchronous learning environments.

Overall, the testing confirmed that the platform is capable of delivering a meaningful laboratory experience remotely. It offers learners the ability to interact with real hardware through an accessible, browser-based environment while maintaining the educational objectives of traditional in-person labs. Additionally, the system exhibited a high degree of technical reliability across various setups, reinforcing its scalability and long-term applicability.

In conclusion, the platform successfully meets its design goals of flexibility, usability, and pedagogical relevance. While iterative improvements—particularly around usability and error communication—will further strengthen the system, the current implementation already demonstrates strong potential as a remote learning solution for embedded systems and IoT education.

Chapter 7 - Conclusions and Future Directions

By enabling authentic, experiential learning in a virtual setting, Remote IoT Laboratories signifies a significant transformation in contemporary education, research, and vocational training. Remote laboratories enhance accessibility, scalability, and cost-effectiveness, thereby fostering a more inclusive and widely available approach to IoT education. Especially for institutions with constrained resources, the capability to remotely configure, program, and test IoT devices enables individuals to acquire essential technical skills without relying on physical labs, thus addressing significant challenges in education.

This research underscores several limitations associated with remote laboratories, particularly the lack of comprehensive hands-on hardware experience for users. While online platforms provide valuable educational opportunities and accessibility, they fall short in fully replicating the tactile and operational nuances of real tools and equipment. To address these shortcomings, future developments should focus on integrating physical kits with advanced virtual environments, enabling a more seamless and realistic learning experience. Additionally, incorporating cutting-edge technologies such as augmented reality and haptic feedback can significantly enhance immersion, allowing users to feel and manipulate virtual objects as if they were tangible. Such innovations promise to make remote laboratories more effective, engaging, and truly reflective of real-world conditions.

Dependence on networks and latency issues pose significant challenges to real-time Internet of Things (IoT) studies, particularly in areas with limited connectivity. These limitations hinder timely data processing and decision-making, affecting system efficiency and reliability. To address this, technologies such as edge computing, distributed systems, and the deployment of 5G networks offer promising solutions. By enabling local data processing and faster communication, these advancements reduce reliance on centralized cloud infrastructure, ultimately enhancing system responsiveness, reducing latency, and expanding IoT applications in connectivity-constrained environments.

In today's digital age, ensuring security and privacy within online IoT laboratories is more crucial than ever. The increasing complexity of interconnected devices raises significant concerns about data breaches and cyber threats that could compromise sensitive information. Although

existing measures such as encryption protocols and multi-factor authentication provide essential layers of protection, they are not foolproof. Emerging technologies like artificial intelligence and blockchain offer promising enhancements by enabling more accurate vulnerability detection, strengthening data integrity, and establishing transparent, tamper-proof records. Integrating these innovative solutions is vital to safeguarding IoT environments against evolving cyber risks.

Enhancing online IoT laboratories demands not only technological advancements but also innovative teaching strategies. Traditional assessment methods often fail to accurately gauge each student's individual comprehension, leading to gaps in understanding. To address this, integrating AI-driven adaptive learning systems can provide personalized feedback tailored to each learner's needs, fostering a deeper grasp of complex concepts. Additionally, promoting collaboration is essential; incorporating group projects, peer evaluations, and real-time problem-solving activities encourages active engagement and teamwork. These approaches help bridge the disconnect often experienced in online environments, making IoT labs more interactive, effective, and conducive to practical learning. Ultimately, blending technology with collaborative pedagogy creates a more dynamic and inclusive educational experience.

Sustainability and scalability are essential for the long-term success of remote laboratories. By fostering collaboration among educational institutions and research centers, resources can be shared efficiently, reducing costs and enhancing innovation. Utilizing open-source frameworks encourages customization and easy integration, making remote labs more adaptable to evolving technological needs. These strategies promote the development of cost-effective, interoperable, and scalable systems that can grow and improve over time. Ultimately, such approaches support widespread access to high-quality remote laboratory experiences, benefiting learners and researchers worldwide.

As we move forward, the role of remote IoT labs is set to grow increasingly vital in the progression towards fully digital and automated environments. The emergence of advanced manufacturing technologies and intelligent systems powered by AI will necessitate a workforce skilled in virtual teamwork, online computing, and immediate data analysis. Future research should explore how AI-driven digital twins can prepare learners to engage with virtual representations of real IoT systems prior to their implementation in practical environments.

Essentially, Remote IoT Laboratories possess untapped potential, despite demonstrating their effectiveness as a valuable resource for modern education. Enhancing their effectiveness, availability, and sustainability will heavily rely on tackling technical, security, and educational challenges. Remote labs have the potential to transform education in the field of IoT, enhancing inclusivity, adaptability, and readiness for the demands of a rapidly evolving digital landscape through continuous investment in technology development, interdisciplinary collaboration, and learner-focused teaching approaches.

References

Abumalloh, R. A., Asadi, S., Nilashi, M., Minaei-Bidgoli, B., Nayer, F. K., Samad, S., & Mohd, S. (2021). The impact of coronavirus pandemic (COVID-19) on education: The role of virtual and remote laboratories in education. *Technology in Society, 67*, 101728.

Ali, Z. H., Ali, H. A., & Badawy, M. M. (2015). Internet of Things (IoT): Definitions, challenges and recent research directions. *International Journal of Computer Applications, 128*(1), 37-42.

Al-Malah, D. K. A., Aljazaery, I. A., Alrikabi, H. T. S., & Mutar, H. A. (2021). Cloud Computing and its Impact on Online Education. IOP Conference Series: Materials Science and Engineering, 1094(1), 012024. https://doi.org/10.1088/1757-899X/1094/1/012024.

Atzori, L., Iera, A., & Morabito, G. (2016). Understanding the Internet of Things: Definition, potentials, and societal role of a fast-evolving paradigm. *Ad Hoc Networks.* https://doi.org/10.1016/j.adhoc.2016.12.004

Azad, A. K. M. (2021). Design and development of remote laboratories with Internet of Things setting. *Advances in Internet of Things, 11*(3), 95-112. https://doi.org/10.4236/ait.2021.113007

Balogh, R. (2010). Educational robotic platform based on Arduino. *Proceedings of the 1st International Conference on Robotics in Education (RiE2010).*

Bayani, M., Leiton, K., & Loaiza, M. (2017). Internet of Things (IoT) advantages on e-learning in the smart cities. *International Journal of Development Research, 7*(12), 17747-17753.

Bochicchio, M. A., & Longo, A. (2009). *Hands-On Remote Labs: Collaborative Web Laboratories as a Case Study for IT Engineering Classes*. IEEE Transactions on Learning Technologies.

Ferdoush, S., & Li, X. (2014). Wireless sensor network system design using Raspberry Pi and Arduino for environmental monitoring applications. *Procedia Computer Science, 34*, 103–110. https://doi.org/10.1016/j.procs.2014.07.059

Fernández-Pacheco, A., Garcia, S., Ruipérez-Valiente, J. A., & Castro, M. (in press). Remote experimentation through Arduino-based remote laboratories. *IEEE Revista Iberoamericana de Tecnologías del Aprendizaje (IEEE-RITA).* https://doi.org/10.1109/RITA.2021.3089916

Fernández-Pacheco, A., Martin, S., & Castro, M. (2019). Implementation of an Arduino remote laboratory with Raspberry Pi. In *Proceedings of the 2019 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1415–1418). Dubai, United Arab Emirates.

Fidalgo, A. V., Alves, G. R., Marques, M. A., Viegas, M. C., Costa-Lobo, M. C., Hernandez-Jayo, U., & Garcia-Zúbia, J. (2014). *Adapting Remote Labs to Learning Scenarios: Case Studies Using VISIR and RemotElectLab*. IEEE Revista Iberoamericana de Tecnologías del Aprendizaje.

Gorbounov, Y., & Chen, H. (2021). Low-cost remote laboratory for cyber-physical experiments. *2021 29th National Conference with International Participation (TELECOM)*, 149-152. https://doi.org/10.1109/TELECOM53156.2021.9659620

Grodotzki, J., Ortelt, T. R., & Tekkaya, A. E. (2018). Remote and virtual labs for engineering education 4.0: Achievements of the ELLI project at the TU Dortmund University. *Procedia Manufacturing, 26*, 1349-1360.

Hadwan, H. H., & Reddy, Y. P. (2016). Smart home control by using Raspberry Pi and Arduino UNO. *International Journal of Advanced Research in Computer and Communication Engineering, 5*(4), 283–288. https://doi.org/10.17148/IJARCCE.2016.5473

Hajjar, S., & Spears, T. (2019). Hardware microprogramming education using Raspberry Pi and Arduino technologies. *International Journal of Intelligent Information Systems, 8*(2), 47.

Huretska, N. (2023). Modern challenges and prospects for the development of remote education: A systematic review of the literature. *Scientific Bulletin of Mukachevo State University. Series "Pedagogy and Psychology"*, *9*(4), 107-115. https://doi.org/10.52534/msu-pp4.2023.107

Ibrahim, U. (2024). The Role of Cloud Computing in Transforming ICT Infrastructure in Educational Institutions. International Journal of Applied and Scientific Research, 2(2), 213-226. https://doi.org/10.59890/ijasr.v2i2.1333.

Index.dev. (2024, November 5). *Kubernetes vs Docker: Which containerization tool should you choose*. Index.dev.

Jamieson, P. (n.d.). Arduino for teaching embedded systems: Are computer scientists and engineering educators missing the boat? *University of Miami.*

Jamieson, P., & Herdtner, J. (2015). More missing the boat—Arduino, Raspberry Pi, and small prototyping boards and engineering education needs them. In *Proceedings of the 2015 IEEE Frontiers in Education Conference (FIE)*, El Paso, TX, USA, 21–24 October 2015.

Jarrah, H., Sarayreh, R., Khasawneh, N. A., & Khasawneh, M. A. (2023). The Role of Cloud Computing in Improving the Performance of School Principals. Eurasian Journal of Educational Research, 107, 110-125. https://doi.org/10.14689/ejer.2023.107.007.

Kadri, S., Sboner, A., Sigaras, A., & Roy, S. (2022). Containers in Bioinformatics: Applications, Practical Considerations, and Best Practices in Molecular Pathology. *The Journal of Molecular Diagnostics, 24*(5), 442-454. https://doi.org/10.1016/j.jmoldx.2022.01.006

Kassab, M., DeFranco, J., & Laplante, P. (2019). A systematic literature review on Internet of Things in education: Benefits and challenges. *Journal of Computer Assisted Learning, 35*(6), 1-14. https://doi.org/10.1111/jcal.12383

Kate, L. (2020, September 3). Kubernetes vs. Docker: What does it really mean? *Sumo Logic.* Retrieved June 28, 2022, from https://www.sumologic.com/blog/kubernetes-vs-docker/

Kouklinos, P. (2024). *The ultimate guide to powering your Arduino Uno board*. Tech Explorations.

Lowe, D., Newcombe, P., & Stumpers, B. (2013). Evaluation of the use of remote laboratories for secondary school science education. *Research in Science Education, 43*(3), 1197-1219. https://doi.org/10.1007/s11165-012-9304-3

Michalec, T., Wojczuk, M., Brzoza-Woch, R., & Szydło, T. (2019). Remote programming and reconfiguration system for embedded devices. *Proceedings of the Federated Conference on Computer Science and Information Systems, 18*, 467–470. https://doi.org/10.15439/2019F170

Miguel, A., PradaJuan, J. F., Serafín, A., Sergio, G., & Manuel, D. (2015). Challenges and solutions in remote laboratories: Application to a remote laboratory of an electro-pneumatic classification cell. *Computers & Education, 85*, 180-190. https://doi.org/10.1016/j.compedu.2015.03.004

Mitsea, E., & Drigas, A. (2019). A journey into the metacognitive learning strategies. *International Journal of Online and Biomedical Engineering, 14*, 4-20. https://doi.org/10.3991/ijoe.v15i14.11379

Monzo, C., Cobo, G., Morán, J. A., Santamaría, E., & García-Solórzano, D. (2021). *Remote Laboratory for Online Engineering Education: The RLAB-UOC-FPGA Case Study*. Electronics.

Moreau, D., Wiebels, K., & Boettiger, C. (2023). Containers for Computational Reproducibility. *Nature Reviews Methods Primers, 3*(50). https://doi.org/10.1038/s43586-023-00236-9

Nedelkovski, D. (n.d.). *DHT11 & DHT22 sensors temperature and humidity tutorial using Arduino*. How To Mechatronics. Retrieved from https://howtomechatronics.com/tutorials/arduino/dht11-dht22-sensors-temperature-and-humidity-tutorial-using-arduino/

PiSl, R., Sobota, J., Balda, P., & Schlegel, M. (2013). Raspberry Pi and Arduino boards in control education. *IFAC Proceedings, 46*(7), 7–12.

Poo, M. C.-P., Lau, Y.-y., & Chen, Q. (2023). Are Virtual Laboratories and Remote Laboratories Enhancing the Quality of Sustainability Education? *Educ. Sci, 13*(1110).

PsiBorg Technologies. (n.d.). *What are the advantages of using MQTT for IoT devices?* Retrieved from https://psiborg.in/advantages-of-using-mqtt-for-iot-devices/

Rahimi, Y., Matyshenko, I., Kapitan, R., & Pronchakov, Y. (2020). Organization of the information support of full logistic supply chains within the Industry 4.0. *International Journal of Quality Research, 14*(4), 1279–1290.

Reguera, P., García, D., Domínguez, M., Prada, M. A., & Alonso, S. (2015). A low-cost open-source hardware in control education: Case study on Arduino-feedback MS-150. *IFAC-PapersOnLine, 48*(6), 117–122.
Arduino CLI. (2022). Retrieved August 20, 2022, from https://arduino.github.io/arduino-cli/1.2/ (visited 12/5/2025)

Saini, L., Jyoti, & Kaur, H. (2017). Role of Cloud Computing in Education System. International Journal of Advanced Research in Computer Science, 8(4), 345-347.

Sobota, J., Pĭsl, R., Balda, P., & Schlegel, M. (2013). Raspberry Pi and Arduino boards in control education. *10th IFAC Symposium Advances in Control Education*. https://doi.org/10.3182/20130828-3-UK-2039.00024

Sorri, K., Mustafee, N., & Seppänen, M. (2022). Revisiting IoT definitions: A framework towards comprehensive use. *Technological Forecasting & Social Change, 179*, 121623.

Sultana, N., & Tamanna, M. (2022). Evaluating the potential and challenges of IoT in education and other sectors during the COVID-19 pandemic: The case of Bangladesh. *Technology in Society, 68*, 101857. https://doi.org/10.1016/j.techsoc.2021.101857

Tamrakar, A. K., Shukla, A., Kalifullah, A. H., Reegu, F. A., & Shukla, K. (2022). An extended review on internet of things (IoT) and its characterisation. *International Journal of Health Sciences, 6*(S2), 8490-8500.

The Hackers Meetup. (2025, February 28). *Cloud fundamentals: Architecture strategies & security solutions*. Medium.

Usmonovna, N. T. (2024). The Role of Remote Laboratories in Teaching Technical Subjects. *International Sciences, Education and New Learning Technologies, 1*(11), 112-118.

Van den Beemt, A., Groothuijsen, S., Ozkan, L., & Hendrix, W. (2023). Remote labs in higher engineering education: Engaging students with active learning pedagogy. *Journal of Computing in Higher Education, 35*, 320–340.

Vince, T., Bereš, M., Kováčová, I., Molnár, J., Fecko, B., Dziak, J., & Guzan, M. (2020). *IoT Implementation in Remote Measuring Laboratory: VMLab Analyses*. Journal of Universal Computer Science, 26(11), 1402-1421.

Why.gr. (n.d.). *Raspberry Pi 4 Model B – 4GB.* Retrieved from https://www.why.gr/καταστημα/open-hardware/raspberry-pi/raspberry-pi-4-model-b-4gb/

García-Zubía, J., & Alves, G. R. (2012). Using Remote Labs in Education: Two Little Ducks in Remote Experimentation*. University of Deusto.*

Håkansson, L., Claesson, I., & Olsson, H. (2017). "Remote labs in engineering education: An approach to support collaborative learning." International Journal of Online Engineering*, 13(3), 103–112.*

Fidalgo-Blanco, Á., Sein-Echaluce, M. L., & García-Peñalvo, F. J. (2015). "Knowledge spirals in higher education teaching innovation." International Journal of Knowledge Management Studies*, 6(1), 1–20.*

Zutin, D. G., Mayweg, D., & Garcia-Zubia, J. (2020). "Developing and integrating remote labs into online engineering education." IEEE Global Engineering Education Conference (EDUCON)*.*

Angular. (n.d.). Angular documentation. https://angular.io/ (visited 12/5/2025)
ApexCharts. (n.d.). ApexCharts documentation. https://apexcharts.com/ (visited 12/5/2025)
Arduino CLI. (n.d.). Arduino CLI documentation. https://arduino.github.io/arduino-cli/
bcryptjs. (n.d.). bcryptjs GitHub repository. https://github.com/dcodeIO/bcrypt.js (visited 12/5/2025)
CodeMirror. (n.d.). CodeMirror documentation. https://codemirror.net/

Docker. (n.d.). Docker documentation. https://docs.docker.com/

Dotenv. (n.d.). dotenv GitHub repository. https://github.com/motdotla/dotenv

Express.js. (n.d.). Express documentation. https://expressjs.com/

Flask. (n.d.). Flask documentation. https://flask.palletsprojects.com/

JWT. (n.d.). JWT Introduction. https://jwt.io/ (visited 12/5/2025)

MJPEG Streaming. (n.d.). Linux MJPG streaming overview. https://elinux.org/RPi-Cam-Web-Interface

ng-bootstrap. (n.d.). ng-bootstrap documentation. https://ng-bootstrap.github.io/

ngx-scrollbar. (n.d.). ngx-scrollbar GitHub repository. https://github.com/MurhafSousli/ngx-scrollbar

Passport.js. (n.d.). Passport documentation. http://www.passportjs.org/ (visited 12/5/2025)

Prisma. (n.d.). Prisma documentation. https://www.prisma.io/ (visited 12/5/2025)

Raspberry Pi Foundation. (n.d.). Raspberry Pi documentation. https://www.raspberrypi.org/documentation/

SweetAlert2. (n.d.). SweetAlert2 documentation. https://sweetalert2.github.io/

Swagger. (n.d.). Swagger UI. https://swagger.io/tools/swagger-ui/

Winston. (n.d.). Winston GitHub repository. https://github.com/winstonjs/winston (visited 1/6/2025)