Leveraging Zephyr RTOS for modern embedded systems A perspective from education and research

Sven Grunwald[1, †], Kevin Krebs[1, †]

[1] TUD Dresden University of Technology, Chair of Information Technology for Traffic Systems (ITVS), 01069 Dresden Germany

Abstract

Due to its strong community support and portability, the Zephyr Real-Time Operating System (RTOS) is gaining increasing popularity in both academic and industrial environments. This paper discusses the integration of Zephyr RTOS into research projects with a particular focus on its application at the Chair of Information Technology for Transport Systems (ITVS) at TU Dresden. We highlight how Zephyr facilitates cutting-edge research in networked embedded systems and serves as a foundation for practical instruction in IoT and telematics. Additionally, we examine the educational benefits but also the technical challenges.

Keywords
Zephyr RTOS, education, lectures, classroom courses

1. Introduction

For students in engineering and information technology hand-on experience with Internet of Things (IoT) technologies and telematics is becoming unavoidable. As modern systems rely on rapid connected equipment and real -time data exchange it is necessary that students not only understand theoretical concepts but also develop practical skills in designing, implementing and analyzing the network embedded system. Working directly with IoT platforms exposes students' complications of device connectivity, wireless communication and data management. It also helps them to appreciate the importance of strong systems design, safety and interspecification. In this context a solid understanding of the Open Systems Interconnection (OSI) model is important. The OSI-model provides a structured approach to understand how data is broadcast and processed in different layers of a network, providing students with a valuable structure for troubleshooting and adaptation of IoT solutions and this is our starting point when entering the classroom. At the Chair of Information Technology for Traffic Systems (ITVS) at TU Dresden, we aim to give students both the theory and hands-on skills they need for future smart and connected systems in the wide field of traffic science. We mix IoT and telematics into our courses and recent research projects to tackle real-world connectivity issues. A stated a big part of our teaching is focused on the principles of the OSI model, guiding students to analyze, design and refine network-based solutions. These skills are essential for careers in smart mobility and digital infrastructure.

2. Rationale for using Zephyr RTOS in education and research

When talking about embedded devices there are usually two main approaches: classic bare-metal programming approaches and RTOS-based approaches. Easily someone would say that the bare-metal approach provides direct control over hardware resulting in maximum efficiency in

[1†] These authors contributed equally.
sven.grunwald@tu-dresden.de(S. Grunwald); Kevin.Krebs@tu-dresden.de (K.Krebs)

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-03-13

combination with minimal delay and a small memory footprint that is ideal for simple or high resource-world systems. Which may be correct, but this means an increase in the complexity of development, lack of portability and the absence of underlying multitasking is difficult to manage concurrent functions or scale the system. On the other hand, using an RTOS (real-time operating system) provides support for structured work management, hardware abstraction and multitasking that greatly simplifies development for complex systems and increases scalability and maintenance but there's a certain learning curve and the need to understand some basic concepts in programming and RTOS-driven software development. Central to our teaching and research is the Zephyr RTOS. It plays an important role in enabling practical and innovative work with embedded systems since 2020 within the classroom as well as within research projects at the chair at TU-Dresden.

Zephyr's modular architecture allows students and researchers to tailor the operating system to specific project needs, gathering a deep understanding of embedded system design by selecting only the necessary components and protocols [1]. Due to the wide range of processor architecture support and hardware platforms it enables seamless transitions between various devices and projects, preparing students for various engineering challenges they will cope with [2]. Zephyrs real-time capabilities ensure resistant behavior and low delays which are important for time-sensitive applications in telematics and IoT [4]. Also integrated features such as memory protection and cryptographic libraries enable the development of strong and secure applications.

This is becoming more and more important in a globally connected world and is underlining the importance of cybersecurity in modern IoT systems [5]. The open source of Zephyr provides widespread documentation, development equipment and learning resources which accelerate the learning process, promotes experiments and supports collaborative research. With Zephyrs' extensive adoption in the IoT industry our students get very relevant skills that transfer directly to the professional environment effectively removing the distance between educational education and industry requirements. Once students and developers become familiar with the configuration-driven approach they will find that enabling hardware features such configuring GPIOs and switching on LEDs is like activating more sophisticated subsystems such as wireless connectivity or networking. This stability means that whether you are working with basic devices / functionalities or advanced connectivity stacks the process usually involves enabling appropriate configuration options and adjusting the device tree [6]. As a result, the learning state invested in understanding Zephyr's configuration model pays off across the entire platform making it easier to scale from simple projects to more sophisticated and feature-rich applications. This cohesive design also reduces the potential for errors and accelerates the transition from prototype to production, highlighting the robustness and versatility of the Zephyr ecosystem. Thats a welcome relief for anyone who's ever found themselves debugging cryptic faults or stuck in a hard fault handler. With Zephyr the path from idea to implementation is more predictable and far less daunting.

2.1. Curriculum integration

At TU Dresden, we offer a variety of courses for both undergraduate and graduate students that use the Zephyr Real-Time Operating System (RTOS) to connect theory with hands-on practice. Our students come from many different fields including electrical engineering, computer science, aeronautics, rail transport and traffic telematics. This diversity brings some challenges but also great opportunities. Notably, almost none of the students have prior experience working with microcontroller boards and many lack a formal background in communications engineering. Yet, understanding connectivity technologies is becoming increasingly important across all areas of technology. To support this, we have developed a series of "crash courses" that run alongside our main lectures. These courses focus on key concepts in embedded systems, networking and wireless communications while giving students practical experience using Zephyr as a teaching tool. This approach helps ensure that all students, regardless of their background or previous hands-on experience gain the essential knowledge and skills needed to work effectively in today's connected world.

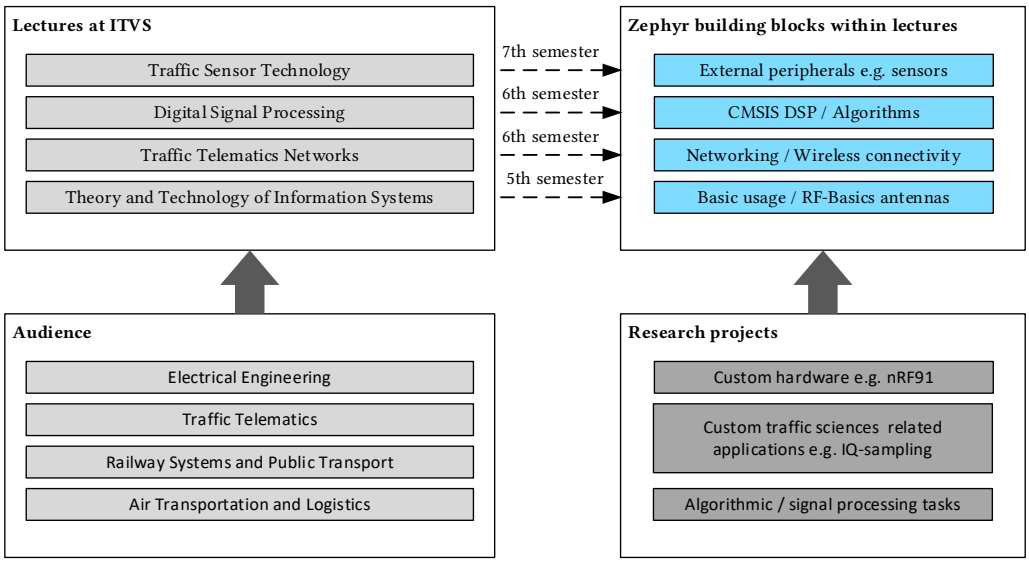

Figure 1: – Zephyr integration in education and research at TU-Dresden / ITVS

To provide a clearer picture of how these concepts are integrated into our curriculum, the following section outlines some of the key lectures and their focus areas. Within lectures such as Theory and Technology of Information Systems we are establishing a strong foundation by teaching the fundamentals of radio frequency technology, antenna principles and propagation effects essential wireless basics for students without a background in communications engineering [9]. This lecture focuses on more low-level hardware near topics. Those skills are important because the underlying hardware in any communication system needs to be understood to make decisions about its suitability in a future application. This theoretical groundwork is seamlessly connected to practical experience: students leverage Zephyr RTOS to implement and analyze real-world scenarios such as state-of-the-art network protocol operations gaining direct insight into how data propagates through complex systems. In complementary courses like Traffic Sensor Technology, Digital Signal Processing and Traffic Telematics Networks our students further explore the practicalities of modern connectivity [9]. They work with widely used protocols (e.g., CoAP, UDP and TCP) as well as emerging IoT standards such as Thread, Matter and UWB for wireless sensor nodes. Another important topic in the curriculum when dealing with networked devices is cybersecurity. By teaching the core concepts of cryptography with the help of Alice and Bob, the students can quickly delve into more specific areas in the field. Learning how methods can be applied to grant access to data only to specific users (authorization and authentication) and means to check whether information was altered or tampered with (integrity). These principles are not only introduced in lectures but are also explored in laboratory exercises where students design and develop sensor nodes, gateways and smart mobility testbeds in combination with RF-measurement equipment. Through these projects, they gain experience with embedded systems concepts like RF-performance, data throughput, real-time scheduling, power management, multi-threading and device driver development, e.g. accelerometer sensitivity range adjustments for a given application. Our curriculum tackles the following key areas:

- Fundamentals in RF-technology, antenna principles and propagation effects
- The ISO/OSI layer model and practical networking, reinforced through hands-on work with protocols like MQTT / MQTT-SN, CoAP, UDP and TCP
- Introduction and application of modern IoT connectivity standards such as Thread, Matter, UWB preparing students to engage with real-world smart device ecosystems

- Fundamentals in Cybersecurity and best-practices for data security in IoT applications and why it is necessary
- Core embedded systems topics e.g. power, constraints and memory management
- Design and implementation of sensor nodes and gateways for smart mobility and traffic telematics, effectively bridging theoretical concepts with industry-relevant topics

By integrating Zephyr into the curriculum, we ensure that students from diverse academic backgrounds acquire the connectivity and embedded systems skills that are now fundamental across all modern engineering and technology fields. Leveraging Zephyr's extensive library of sample applications and board support packages enables students to focus on high-level system design, debugging and application development fast. This approach accelerates the learning process but also mirrors best practices in industry where rapid prototyping and cross-disciplinary collaboration are the norm and kind of expected. All in all the curriculum is designed to make complex technical topics accessible and relevant to students from all engineering backgrounds ensuring that they acquire the essential skills needed to thrive in a world where connectivity is ubiquitous and foundational to innovation in every technical field. Through this holistic and application-oriented integration of Zephyr we empower our students to understand but also design and troubleshooting modern connected systems.

2.2. Case Study: research project on real-time data acquisition

At TU Dresden a recent research project showcased how powerful Zephyr RTOS can be for real-time data acquisition in traffic monitoring. The team worked with a custom hardware platform based on the Nordic Semiconductors® nRF9160 cellular IoT module. Due to the modular software design and the available configuration tools the researchers were able to build a robust multi-layer network stack and integrate secure cloud connectivity via MQTT. The components of the setup are shown in figure 3. The protocol MQTT was chosen to ensure flexibility, scalability and *loose coupling* to have freedom in the mode of operation on both sides. The development of a live web dashboard for data visualization was part of the outcome as shown in figure 4.

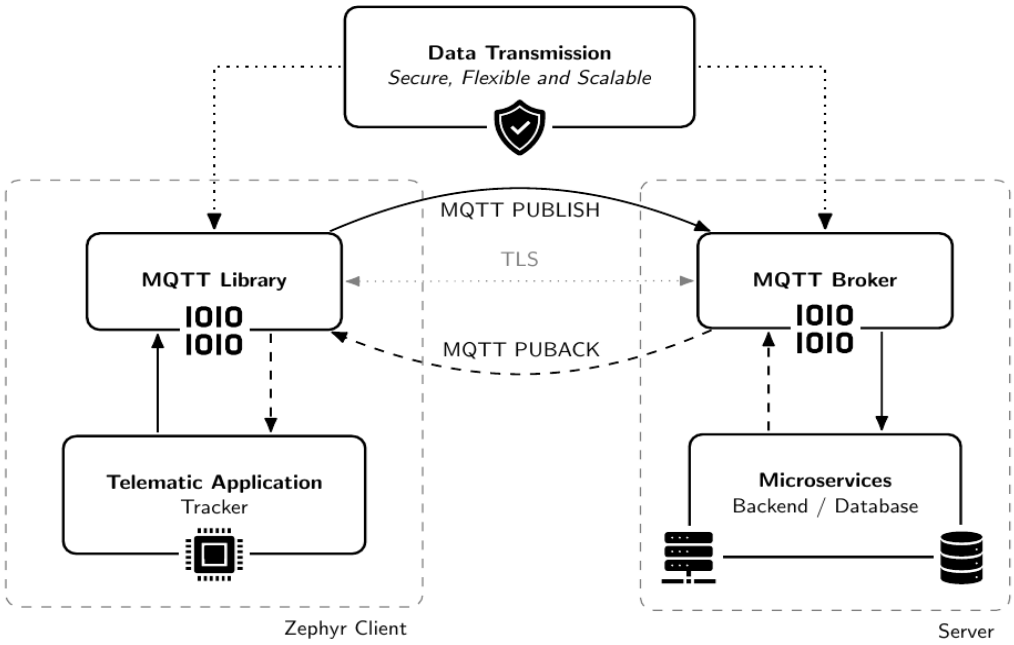

Figure 2: – Setup of telematic application in research project using Zephyr and backend services with MQTT (QoS 1 is shown)

One of the main advantages was as mentioned the Kconfig and device tree system, which allowed the team to configure everything from basic hardware settings like pin-controls to advanced subsystems in a straightforward way. This flexibility proved invaluable particularly as the project

moved from theoretical models such as real-time scheduling and secure communication protocols into real-world implementation. The team could work “close to the metal” when needed, optimizing low-level peripherals (e.g. ADC) and power consumption, but also take advantage of abstractions for rapid development which is usually the case when dealing with research projects and tight timelines especially in a university context.

A clear illustration of Zephyr’s strengths emerged when the team needed to implement a custom low-power sleep mode that dynamically adjusted based on physical movement and incoming sensor data rates and network activity. Using Zephyr’s flexible power management APIs and real-time kernel features they could fine-tune the device’s energy consumption without sacrificing responsiveness. For instance, the system was able to wake instantly to process high-priority traffic events and then return to deep sleep ensuring to minimize power consumption. This was an essential requirement for cellular IoT deployments within the remote traffic monitoring context because the test carrier was quite sensitive to quiescent current. This level of control in combination with Zephyr’s support for seamless integration of custom analytics modules enabled the team to process and filter large volumes of sensor data locally e.g. sending only relevant information to the cloud. As a result they achieved both efficient energy usage and reduced the costs of the cellular data demonstrating how Zephyr empowers developers to create optimized and field-ready solutions.

Figure 3: – Zephyr based hardware installed in test carrier (tram) of local transport operator

The outcomes of the project were not confined to research activities alone. They were systematically integrated into our teaching framework, facilitating the adoption of the Zephyr RTOS and nRF9160 platforms in our curriculum. Challenges encountered and solutions developed during the research phase were translated into new laboratory exercises and project modules that closely reflect real-world engineering scenarios. This approach provides students with concrete examples that bridge theoretical instruction and practical application, thereby demystifying complex embedded systems concepts through experiential learning. For instance, students engage directly with the same hardware and firmware utilized in our research allowing them to observe firsthand the implementation of advanced topics such as power management and secure IoT communication including the practical application of cryptographic techniques. Furthermore, students experiment with various cryptographic algorithms gaining insight into their impact on system resources such as memory utilization. This hands-on exposure equips students with the skills and understanding necessary to address contemporary challenges in embedded and connected systems. A more complex and therefore challenging task for the participants is setting up the Zephyr environment to work

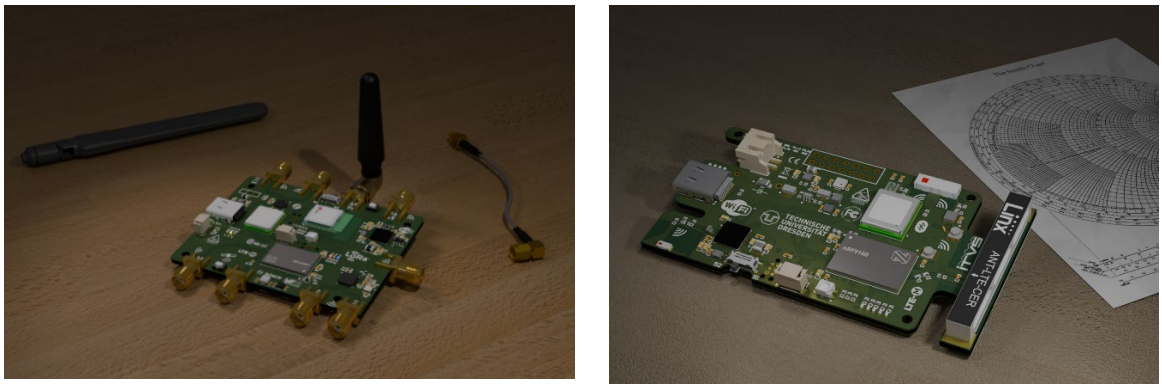

Figure 4: – Custom Hardware for teaching purposes as outcome of research projects

with network protocols like MQTT and MQTT-SN making sure everything runs well on low-power devices. One important assignment is to keep the system reliable even when things go wrong, like losing the network or server issues arise. Through this practical experience, students not only tackle the tough topics that come with modern RTOS development but also gain confidence to play around with advanced features like remote diagnostics and cloud data visualization (Figure 5). They learn to fix real problems, adapt to changing needs and understand why solid design matters. This hands-on approach gives them a deeper grasp of embedded systems and gets them ready to face challenges in the industry with practical skills and creative thinking.

Figure 5: – Data visualization and result presentation of various measurement parameters (GNSS, velocity, temperature and RSSI)

By engaging with industry-standard tools and development workflows students gain valuable skills that are directly transferable to professional settings. The integration of Zephyr and the nRF9160 platform into the curriculum has inspired many students to pursue their own projects and even take a closer look into open-source initiatives. Over time this work has also helped strengthen our partnerships with leading industry players like Nordic Semiconductor, ST Microelectronics, Memfault e.g. guest talks from an industry perspective. These collaborations have opened doors for students to work with cutting-edge hardware and software such as Nordic's nRF Connect SDK, state-

of-the-art hardware platforms like nRF53/54/91, Memfault's advanced device monitoring tools and even STMicroelectronics™ MEMS sensors featuring machine learning core technology. Not only do students gain hands-on experience with the same technologies used by professionals but they also have opportunities to connect with industry mentors and build networks early. This is especially valuable in the Dresden region where the embedded sector is thriving. Through workshops, guest lectures and joint projects our students are exposed to real-world challenges and current industry practices further enhancing their readiness for the workforce. As more companies adopt Zephyr this practical exposure and industry engagement give our students a real advantage in their studies and as they launch their professional journeys. This synergy between academia and industry ensures that our graduates are well-prepared to contribute meaningfully to the rapidly evolving field of embedded systems.

3. Lessons learned and best practices

The inclusion of Zephyr RTOS in our embedded systems class has been a driving force in advancing the adoption of teaching ideas in real-time and embedded software. The scalability of Zephyr's design, with the aid of a microkernel that is supported by many hardware platforms, is an environment relevant to the industry for students. Early engagement with Zephyr allows students to master essential RTOS design patterns, including preemptive multitasking, priority-based scheduling and inter-thread communication.

3.1. Transformative impact and collaborative learning with Zephyr RTOS

The adoption of Zephyr RTOS within embedded systems education has led to significant changes in both instructional content and pedagogical methods. Curricula increasingly combine theoretical lectures with practical laboratory sessions enabling comprehensive coverage of essential RTOS principles including kernel initialization context switching, interrupt management and various memory management techniques such as heap, stack and memory pool utilization. Laboratory-based activities reinforce these foundational concepts by requiring participants to deploy Zephyr on diverse hardware platforms modify device trees for hardware abstraction and interact with subsystems such as file systems, network stacks and peripheral drivers. This emphasis on experiential learning facilitates the transition from theoretical understanding to practical competence. Application-oriented tasks such as the implementation of sensor fusion algorithms or the development of Bluetooth Low Energy (BLE) solutions serve to bridge the divide between academic instruction and industry practices. A central factor in this educational evolution is the open-source nature of the Zephyr ecosystem. The collaborative environment fosters exploration of kernel source code contributions to bug fixes and feature enhancements and active engagement with a global developer community. Such exposure demystifies the software development process and encourages the adoption of professional practices including code review, version control and continuous integration. By shifting the focus toward application logic and overall system design Zephyr's abstraction mechanisms make it possible to engage with sophisticated subjects including Bluetooth LE channel sounding without requiring exhaustive familiarity with low-level hardware details we learned recently in a student thesis we supervised at the chair [10].

3.2. Addressing the learning curve

Although it has its advantages, Zephyr does have a non-negligible learning curve, particularly for users from bare-metal or proprietary embedded development backgrounds. Key hurdles to overcome are understanding Zephyr's build system (west and CMake), the Kconfig configuration language and adapting to the device tree hardware description paradigm. These are elegant concepts but can be mysterious to the uninitiated and require a shift in thinking from imperative register manipulation

to declarative hardware configuration. To assist in mitigating these concerns, we are and continue to make the following best practices:

- Begin with foundational Zephyr topics such as the basic application structure and build workflow before advancing to more complex features like multi-threading and inter-process communication
- Integrate comprehensive, hands-on tutorials that walk students through essential tasks such as configuring hardware with device tree overlays and customizing system behavior with Kconfig using practical examples
- Structure coursework so that each project builds upon the previous one, gradually introducing new Zephyr subsystems and concepts to reinforce learning and promote a sense of achievement
- Encourage teamwork through code reviews, group debugging sessions and peer-to-peer mentoring encouraging students to share insights and troubleshooting together
- Utilize regular check-ins, surveys and reflective exercises to identify learning bottlenecks and adapt instructional methods accordingly
- Empirical results collected in the form of pre- and post-course survey indicate a substantial rise in students' self-rated confidence and skill level in RTOS concepts after completing Zephyr-based modules. Performance indicators such as lab completion rates and project success rates also affirm these results.

3.3. Paradigm shift: from register-level programming to system-level design

One significant result of bringing Zephyr into the classroom is the shift it encourages in how students approach embedded systems. Instead of focusing on programming at the register level students begin to think in terms of system-level design and abstraction. In many traditional embedded systems courses the emphasis is placed on directly manipulating peripheral registers. While this method has educational value it does not always align with the way embedded software is developed in industry today especially when talking connected embedded systems. Zephyr changes this by providing device drivers and standardized application programming interfaces. This results in highly portable code that can run on a variety of architectures e.g. ARM Cortex-M or RISC-V. This abstraction makes it easier for students to direct their efforts toward building systems that are scalable and easier to maintain. For example, instead of writing code that is specific to a particular vendor just to initialize the UART. Students can use Zephyr's common interface for serial communication. This optimizes the development process and introduces important concepts such as platform independence as well as modularity and reuseable code. These are qualities that are highly valued in the current embedded systems job market. It is important to acknowledge that Zephyr's abstraction does not always cover every possible use case with complete consistency. There are situations where applications need access to advanced or highly specialized hardware features and in these cases the generic interfaces may not provide all the necessary functionality. For example, Nordic Semiconductors® nRF devices using features like the programmable peripheral interconnect (PPI) for advanced analog-to-digital conversion often require developers to work directly with the vendor's software development kit stepping outside of Zephyr's abstraction. The same is true for the Raspberry Pi Pico where the basic analog-to-digital converter can be accessed through Zephyr but making use of advanced features such as direct memory access, round-robin sampling or detailed configuration options typically means turning to the Pico SDK or manipulating hardware registers directly. These examples illustrate a challenge in the design of portable system code. There is often a trade-off between generating code that's portable across different hardware platforms and to be able to access the full range of features that a specific device offers. While Zephyr is designed to promote portability and modularity developers sometimes need to fall back on hardware-specific code to achieve the best performance or to use unique features. This reality highlights why it remains important for students to understand both the higher-level concepts of operating systems and the

lower-level details of hardware programming especially when working on advanced or performance-sensitive embedded applications. By explicitly addressing these limitations in the curriculum students gain a realistic understanding of both the strengths and boundaries of RTOS-based development. They learn not only to appreciate the benefits of system-level abstraction but also to recognize scenarios where direct hardware access remains essential for leveraging the full potential of modern microcontrollers.

3.4. Quantitative evaluation of learning outcomes

The integration of Zephyr RTOS into our embedded systems courses has produced measurable improvements in student performance, confidence and project success. Oral surveys before and after implementation indicate that students' self-reported confidence in developing embedded applications increased significantly. From around 40% confidence initially to 75% upon completing Zephyr-based courses illustrating how hands-on exposure to a modular, industry-focused RTOS bridges the knowledge-practice gap. Project and laboratory success rates also consolidate this trend. Successful project submission went up from 65% in previous cohorts to 85% after Zephyr was integrated, with significant improvement in applications requiring lots of resources, including real-time processing of data and power management. Additionally, lab exercise debugging time for minor firmware issues decreased by approximately 50%, from 30 minutes average time to 15 minutes. This indicates that students rapidly gain debugging tool proficiency and system knowledge for less complex programming flaws. These gains are complemented by more instances of using optional advanced projects, such as building secure IoT communication modules or low-power sensor networks, which leverage Zephyr's on-board security libraries and power management APIs. In combination, these quantitative results demonstrate that Zephyr RTOS increases technical expertise and builds a deeper knowledge of system-level design principles critical to modern embedded development. In the future, longitudinal studies will attempt to measure how these initial successes impact students' transition into industry roles, with the hope that the experience and exposure accrued through Zephyr will mean greater employability and readiness for future needs within the field of embedded systems.

3.5. Debugging Challenges and the need for improved tooling

The most longstanding problem encountered is most likely debugging within the Zephyr platform. Tightly integrated but proprietary IDEs such as KEIL® uVision® and IAR Embedded Workbench®, for instance, offer a more straightforward setup process and experience. Since Zephyr relies on open-source and free tools like GDB, OpenOCD and SEGGER Ozone. This setup procedure can be intimidating for both students and instructors alike, especially for those new to embedded systems development. Moreover, the use of tools occasionally results in unpredictable behavior during debugging or programming sessions. Issues such as devices failing to program without a visible feedback or unexpected deadlocks at startup are not uncommon and can detract from the overall learning experience. To address these debugging challenges there is a need for better tooling and integrated development environments that simplify setup and troubleshooting. Developing or adopting Zephyr-specific IDEs with enhanced features such as real-time thread visualization and integrated peripheral monitoring could significantly improve the user experience. Obviously debugging multi-threaded, real-time programs introduces additional and challenging complications. Deadlocks, priority inversion and race conditions are intrinsically difficult to identify, as context switching and task scheduling events are not always obvious in the source code or easily traced using traditional debug tools. Current solutions are mainly for single-threaded static platforms and do not adequately deal with the dynamic behavior of RTOS-based applications. To address these challenges, we recommend the following from a teaching perspective:

- Creation or adoption of a graphical interface supporting real-time thread visualization, event tracing and integrated peripheral monitoring
- Comprehensive instructional materials and video guides for setup and use of debugging tools emphasizing common challenges and advanced RTOS debugging methods
- Incorporation of advanced analysis solutions e.g. Percepio Tracealyzer® or SEGGER SystemView to provide accessible, real-time insight into thread execution, system events and resource usage in a more streamlined and easy to follow way
- Collection and analysis of data on debugging outcomes, resolution times and user feedback to continuously improve curriculum effectiveness

3.6. Recommendations for future improvement and value additions

To further enhance the educational value of Zephyr in the classroom and contribute to the broader community the following initiatives could be useful:

- Systematically gather and analyze quantitative and qualitative data on student performance and learning outcomes
- Contribute modular, open-source teaching materials including lab manuals, annotated code samples and debugging guides to the Zephyr documentation repository
- Establish stronger ties between academia and the Zephyr developer community by encouraging student / instructor participation in forums and open-source contributions as part of coursework
- Establish feedback loops with students and industry partners to ensure the curriculum remains aligned with evolving best practices and technological advancements

4. Conclusion and outlook

Integrating Zephyr RTOS into the curriculum and research activities at TU Dresden has significantly modernized embedded systems education. Its modular design and community support facilitate quick adaptation for diverse projects and foster practical skill development. Notably, even students with limited prior experience in embedded systems have successfully engaged with Zephyr, overcoming initial challenges to design and implement projects at bachelor's and master's levels. This demonstrates Zephyr's effectiveness in bridging the gap between theoretical knowledge and real-world application. This demonstrates with the right guidance and resources the challenges of adopting modern RTOS are surmountable. Nevertheless, our experience has highlighted several opportunities for further improvement within our courses and labs. The initial learning curve remains a significant challenge particularly for newcomers to embedded systems. To address this, we plan to develop more structured onboarding materials e.g. beginner-friendly tutorials in combination with step-by-step lab guides. Interactive workshops that clarify key concepts like device trees, KConfig and Zephyr's build system could also improve the learning experience. Expanding the peer-to-peer learning and mentoring opportunities can also help students navigate early hurdles more effectively. Moreover, keeping instructional materials and documentation up to date is essential in a fast-moving ecosystem. A good example for managing different Zephyr and in this case SDK versions is the Nordic Semiconductors® developer academy [9]. Establishing a robust feedback loop with students and regularly updating course content ensures that our teaching remains relevant and impactful. The inclusion of real-world case studies or projects from industry can further bridge the gap between academic learning and professional practice. By constantly expanding available hardware platforms and project topics in our laboratories, upcoming students will be able to fully explore the potential of Zephyr and apply their knowledge to a wider range of applications. The aim is to establish interdisciplinary collaboration and enhance the learning experience to enable innovative solutions to complex challenges. Although the introduction of Zephyr RTOS at TU Dresden has already brought significant benefits for students and research there is still considerable potential for growth but as always that's a process.

Declaration on Generative AI

During the preparation of this manuscript, the authors employed DeepL to assist with sentence refinement and rephrasing. Following the use of this tool, the authors thoroughly reviewed and edited all content as required and assume full responsibility for the accuracy and integrity of the final publication.

References

[1] Rivera-Matos, Zephyr RTOS explored: From system design to academic applications, Zephyr Project, 2024. URL: https://www.zephyrproject.org/zephyr-rtos-explored-from-system-design-to-academic-applications/ (accessed 29.05.2025).

[2] K. Albes, Design und Implementierung eines Zephyr-RTOS-Modells für ARA zur statischen Whole-System-Analyse, Bachelorarbeit, Leibniz Universität Hannover, 2021.

[3] R. Rivera-Matos, Zephyr RTOS explored: From system design to academic applications, Zephyr Project, 2024. URL: https://www.zephyrproject.org/zephyr-rtos-explored-from-system-design-to-academic-applications/ (accessed 29.05.2025).

[4] M. Vedral, Zephyr RTOS, Bachelorarbeit, České vysoké učení technické v Praze, 2023.

[5] Integration of Zephyr RTOS in motor control systems: Challenges and solutions, in: International Journal of Computer Science and Engineering (IJCSE), 2024.

[6] R. Vágó, Development of a safe architecture for embedded systems using Linux and Zephyr RTOS, Master's thesis, KTH Royal Institute of Technology, Stockholm, 2022.

[7] S. Berg, A. Nyffenegger, Connecting constrained devices to the cloud using Zephyr, Bachelorarbeit, Universität Freiburg, 2020.

[8] TU Dresden, Lehrstuhl für Verkehrstelematik (ITVS), Lehrveranstaltungen und Modulbeschreibungen, 2025. URL: https://tu-dresden.de/bu/verkehr/itvs (accessed 29.05.2025).

[9] Nordic Semiconductor, Nordic Developer Academy – Online learning platform, 2024. URL: https://academy.nordicsemi.com (accessed 29.05.2025).

[10] M. Böhme, Empirische Evaluation von Bluetooth LE basierten Abstandsmessungen, Studienarbeit, Technische Universität Dresden, Fakultät für Verkehrswissenschaften "Friedrich List", Professur Informationstechnik für Verkehrssysteme (ITVS), 2025.