Taming the Challenges of Teaching the IoT

Bernd-Christian Renner
christian.renner@tuhh.de
Hamburg University of Technology
Hamburg, Germany

Peter Oppermann
peter.oppermann@tuhh.de
Hamburg University of Technology
Hamburg, Germany

Fabian Steinmetz
fabian.steinmetz@tuhh.de
Hamburg University of Technology
Hamburg, Germany

Abstract

University-level courses on the Internet of Things, Cyber-Physical and Embedded Systems are in high demand. To improve the learning progress of students and prepare them for the job market—be it in industry or academia—both theoretic and practical aspects should be covered. This particularly implies the design and implementation of software for real hardware. However, the development process differs significantly and is new to many students. Likewise, resource constraints and the imperfect behavior of the hardware poses additional, typically unknown challenges. At the same time, the field is developing rapidly, with new protocols and algorithms evolving and vanishing. Thus, picking the right contents and dozes is a challenge by itself, which is aggravated by the diverse backgrounds of students from various degree programs. We report on the major challenges, our approaches, findings and feedback from a decade of teaching.

CCS Concepts

• **Social and professional topics** → **Computer science education**; **Computer engineering education**; **Software engineering education**.

Keywords

Embedded systems, Cyber-Physical Systems, Internet of Things, Teaching, Software, Hardware

1 Introduction

The Internet of Things (IoT) comprises networks of physical devices embedded with sensors, actuators, software, and communication capabilities. These interconnected systems have enabled transformative advances across numerous domains, including smart cities (e.g., infrastructure monitoring, traffic management, public safety) [10], agriculture (e.g., crop health monitoring, precision farming, water management) [9], and industrial automation (e.g., predictive maintenance and realtime monitoring) [1]. Closely linked with Cyber-Physical Systems (CPS) and Embedded Systems (ES), the IoT gains traction and drives innovation in both industry and academia. At the same time, large-scale deployment poses environmental and technical challenges [13], necessitating the use of low-cost, energy-efficient, and resource-constrained devices.

From a pedagogical perspective, the interdisciplinary and fast-evolving nature of the IoT introduces substantial challenges for university-level teaching. These include mismatches between student backgrounds and the cross-domain knowledge IoT requires, as well as the sheer diversity of existing hardware and software platforms [5]. Most degree programs focus narrowly—e.g., on either computer science or electrical engineering—leaving students well-versed in one area but underprepared in the other. For instance, computer science students often lack experience interfacing with hardware, while electrical engineers may struggle with software abstractions such as device drivers. Students from hybrid programs like mechatronics exhibit strong motivation but often lack foundational skills in computer science. For example, they may intuitively understand the kinematics of a drone like the CrazyFlie [2], yet struggle to implement control algorithms at the hardware level. Effective course design must, therefore, foster a shared understanding across domains and support integrative learning experiences.

These challenges are compounded by the specific demands of the IoT domain. First, most students are unfamiliar with programming embedded hardware, leading to steep learning curves in development and debugging. Second, the strict constraints of IoT systems—e.g., minimal memory, no full-fledged operating system (OS), and the need for real-time responsiveness—require students to consider their program image size and memory management, which they are seldom used to. Third, few students have practical experience with low-power communication, including issues like packet loss, interference, and variable connectivity. Standards such as IEEE 802.15.4 or LoRa are rarely covered in networking courses, despite their relevance. A solid theoretical foundation in wireless communication and its practical constraints is thus essential. Fourth, hands-on engagement with real hardware is critical. Students learn best when they can deploy and test their software on physical devices. Motivation increases when tasks are clearly defined and yield rewarding outcomes—ideally solving real-world problems. However, acquiring and maintaining hardware infrastructure, especially amid rapid obsolescence, remains a persistent challenge.

From a course design perspective, the rapid evolution of hardware platforms, communication protocols, and OSs poses ongoing challenges for developing sustainable, high-quality teaching content. Limited instructional time further constrains learning, making it difficult to offer personalized support or address all student questions. Moreover, the scope of IoT is so broad and complex that no single course can comprehensively cover all relevant subtopics.

Drawing on more than a decade of experience in IoT-related teaching, we have encountered recurring challenges and developed strategies in response. While a comprehensive treatment would exceed the scope of this paper, we highlight the most salient observations and design choices.

This paper presents the evolution of our IoT teaching approach at the university graduate level in Sect. 2. We focus on the structure and lab modalities of two complementary courses. In Sect. 3, we reflect on student feedback and outline ongoing challenges and areas for improvement.

2 IoT Teaching Aspects

To address the inherent complexity of IoT and the diverse prior knowledge of students, we implemented two complementary but independent courses, each targeting a specific aspect of IoT education at a different level of abstraction:

- *Software for Embedded Systems* (SES) builds foundational skills in embedded systems and focuses on programming resource-constrained microcontrollers at the hardware-level in the C language.
- *Autonomous Cyber-Physical Systems* (ACPS) addresses low-power wireless networking and energy-harvesting strategies, where an open real-time operating system is being leveraged.

In light of the environmental and cost constraints associated with IoT, we deliberately avoid more convenient, high-level platforms such as the Raspberry Pi, opting instead for energy-efficient microcontrollers that reflect real-world constraints.

2.1 Software for Embedded Systems

2.1.1 Overview. SES has been originally devised by Volker Turau, where the initial set of labs was designed by the first author of this paper. Since 2020, the course has been fully adopted and partially revised by the authors. In the current semester, around 60 students from diverse degree programs (e.g., computer-science, electrical engineering, microsystem engineering, and mechatronics) participate.

SES focuses on the foundational aspects of ES programming, particularly relevant to IoT and CPS contexts. Students are challenged to implement at hardware-level in the C programming language; i.e., bare-metal on the microcontroller without support of an embedded OS or libraries. Learning goals are to gain a deep understanding of typical hardware (microcontroller, internal and external peripheral interfacing) and the requirements and challenges of embedded software.

The course commences by exposing students to core hardware concepts; i.e., microcontroller operation (instruction execution) and low-level programming in the C language (e.g., bit manipulation). They learn how to write software for a remote target and how to compile, flash, and debug it. Subsequently, we delve into the configuration and use of internal (e.g., timers) and external peripherals (e.g., LEDs, buttons, a fan), address interrupt handling, the link between hardware and software, and race conditions including atomic blocks. Next, we study the implementation of a light-weight scheduler, culminating in the concept of preemptive scheduling. Complementary, we study software modeling through UML and conclude with an outlook on real-time operation, OSs, and task scheduling.

2.1.2 Course Organization & Hardware. SES consists of a partly interactive lecture and a project-based learning lab. The lecture explains the topics thoroughly, leaving sufficient room for questions and interactive elements (quizzes, discussions, and ad-hoc tasks). It is tightly synchronized with a weekly lab, in which the students work in teams of two. Throughout the semester, the teams work on six assignments with real, custom-designed hardware, in which they implement a library of functions that serves as a hardware-abstraction layer (HAL) and is step-wise expanded to a light-weight OS with a scheduler. Students steadily reuse their software components—treating them as a library—and finally combine them into a fan control and a complex alarm clock.

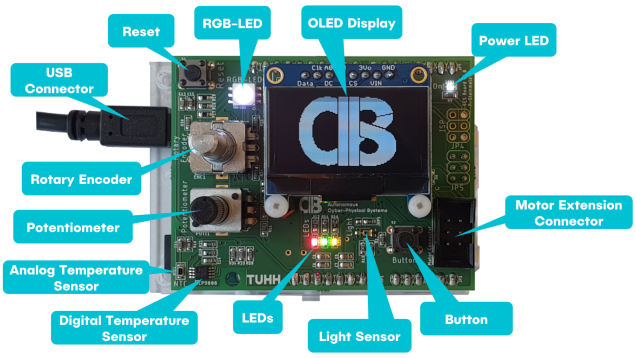

Figure 1: The SES board extends off-the-shelf Arduino hardware with frequently used real-world peripherals.

While assignments are self-contained, they build upon each other, exemplarily showing the benefit of good software design and code reuse through well-defined libraries. To clarify expectations we provide binary images to showcase a proper solution without revealing its code. All exercise sheets contain detailed explanations; however, students must read data sheets and hardware documentation to solve all tasks appropriately. While we expect students to prepare their solutions at home, we run weekly 3-hour tutor-guided meetings, where they have access to the lab hardware for testing and debugging.

Given the rapid obsolescence of commercial embedded platforms, we designed a custom hardware built from off-the-shelf components (Fig. 1). It extends a standard Arduino-compatible board with additional peripheral interfaces, enabling full control over hardware features while maintaining a compact and cohesive form factor suitable for lab use. The board is based on a MicroChip ATmega32U4, a deliberate choice aligned with the educational objectives and typical applications: As an 8-bit device with 2.5 kB RAM and 32 kB ROM, it represents many low-power ESs, offering an ideal balance between complexity and pedagogical clarity. Compared to modern 32-bit microcontrollers, the ATmega32U4 is simpler to program, making it more accessible to students—many of whom have limited experience with low-level C programming or embedded hardware. Our extension board contains different button types, analog sensors, LEDs, and a display. Moreover, a fan controller can be attached, which is a larger peripheral and only used for one of the lab tasks.

We refactored and improved the hardware for summer term 2023, where an Arduino Leonardo (ca. € 20) is extended via a full-custom peripheral board (ca. € 40 material cost) and a full-custom motor controller expansion board (ca. € 30 material cost). The two custom boards where designed by the authors based on a previous design from the group of Prof. Turau. We ordered 15 peripheral boards from a German PCB manufacturer for ca. € 50 per board[1] excluding the display (ca. € 20). This allowed us to supply 15 teams

[1] prices applicable to the corresponding volume only

of 2 students each in 2–3 groups per semester, thus servicing 60–90 students in total.

2.1.3 Remote Access and the Pandemic. The inaugural run of the revised SES course coincided with the onset of the COVID-19 pandemic in Spring 2020. Although the large number of participants made it infeasible to hand the few devices we have to the students for take-home, we were determined to keep the hands-on component. Therefore, we implemented a remote testbed that allowed students to program and interact with physical devices over the Internet.

The setup comprised six hardware units connected to a central host computer and monitored via webcams. Buttons were emulated using relays, and students could book time slots via a web interface to upload their code, view serial output, and control inputs. Visual feedback was provided through real-time video streams focused on the booked device (see Fig. 2).

While the remote lab was a useful replacement during the pandemic, we returned to on-campus labs afterward. The main reasons were the relatively high maintenance costs (staff) and the inability to replace all physical input (e.g., a rotary encoder) and output (e.g., fan rotation) stimuli sufficiently well. Moreover, the hardware revision of the boards (see above) would have required modification of the remote lab, and there were legal constraints when providing the equipment outside office times.

2.2 Autonomous Cyber-Physical Systems

2.2.1 Overview. Introduced in 2022, the *Autonomous Cyber-Physical Systems (ACPS)* course complements SES by addressing higher-level, research-oriented topics. ACPS emphasizes algorithms and protocols for energy-autarkic operation and low-power wireless networking. Higher-level programming is achieved through the use of a real-time OS—currently Mbed OS. In the current semester, about 60 students were interested in the course, although we could only accommodate 30 students in the lab due to available resources.

The course begins with an in-depth exploration of Mbed OS's real-time features. This is followed by coverage of energy-harvesting technologies, such as solar and vibrational sources, alongside a discussion of their operational strengths and limitations. Key paradigms, including energy-neutral operation (ENO) [7] and intermittent computing (IC) [6], are examined in conjunction with storage technologies ranging from capacitors to rechargeable batteries. Students study algorithms for energy harvesting prediction and long-term energy modeling [3, 4].

Subsequently, we discuss the limitations and implications of low-power communication, where we focus on the LoRa physical and network layer, but also touch upon IEEE 802.15.4 and standard protocols for low-power medium access and routing to widen the scope. For future iterations of the course, we plan to include other emerging techniques such as Bluetooth Low Energy (BLE). To achieve a system-level perspective, we conclude the course on the networking layer covering MQTT and general networking aspects.

2.2.2 Course Organization & Hardware. ACPS follows a similar structure to SES, combining lectures with weekly lab sessions. The lectures establish foundational concepts through a mix of traditional instruction and interactive elements such as quizzes and problem-solving activities. Lab assignments are coordinated with lecture topics. Unlike SES, ACPS labs are only loosely coupled and each highlights a different aspect from the lecture, reflecting the diversity of topics covered. Each lab emphasizes implementation on real hardware rather than simulation. Hence, not all lecture materials—e.g., energy prediction algorithms—can be replicated in the lab.

Our hardware platform (Fig. 3) is based on the STM B-L072Z-LRWAN1 [12] development board (ca. € 50), which features a low-power 32-bit STM32L072CZ microcontroller and an SX1276 LoRa transceiver. We designed a custom extension board incorporating our energy-harvesting circuit from [4], combining a small solar panel and a 50 F-supercapacitor for energy storage. The system includes ambient and housekeeping sensors (e.g., temperature, pressure, capacitor voltage, solar current) and an SD card interface for persistent storage. The component cost of this full-custom board sums up to ca. € 60 and is currently assembled in-house.

In the final lab, the students implement a full-fledged IoT application for distributed plant-watering. Therefore, the hardware is extended with a humidity sensor, and a Raspberry Pi controls water flow through a valve (Fig. 4). The students implement the embedded water-level-sensing and valve-control logic in C++ and develop a coordinating Python server application. This challenging exercise requires them to consider potential faults and safety mechanisms and to interface between heterogeneous devices. An issue with this complex setup is that it cannot be made available on a permanent basis and outside the lab hours.

2.2.3 Experiences from Past Iterations. The current structure of ACPS is the result of several iterations. Before 2022, the course adopted a conventional Wireless Sensor Networks (WSN) curriculum [8], paired with basic lab exercises using TinyOS. These proved uninspiring and failed to engage students meaningfully.

In response, we experimented with a project-based model similar to the one reported in [5], where student teams tackled diverse, open-ended tasks—such as transforming a toy boat into an autonomous surface vehicle. Although these projects fostered creativity, they demanded extensive instructor support that is not manageable with the large number of participants. Moreover, the open-ended projects were highly susceptible to dropouts, mismatched expectations, and issues due to varying skill levels.

To improve learning outcomes and manageability, we transitioned to the current format: a hybrid of structured lectures and focused, hands-on labs within a controlled environment. This design strikes a balance between student engagement and feasibility while maintaining high educational value.

3 Student Feedback and Insights

To continuously refine our courses, we actively participate in the structured teaching evaluations at TUHH. These are typically conducted during a dedicated 10 min to 15 min slot in the final lecture to encourage participation.

Feedback is regularly discussed in group meetings and, where feasible, informs course improvements. Some suggestions, however, are constrained by resources or diverge from our teaching philosophy. Below, we highlight recurring themes, summarize student input, and reflect on how we have responded.

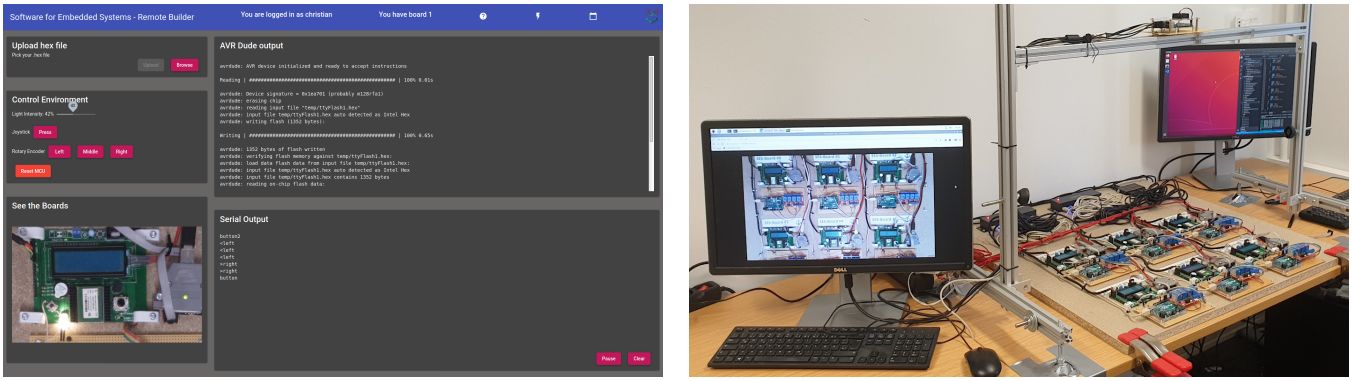

Figure 2: The remote lab integrates six devices with live video footage (right). A web-interface (left) enables remote software flashing, actuation (e.g., button presses) and receiving feedback (LED state, display, serial interface).

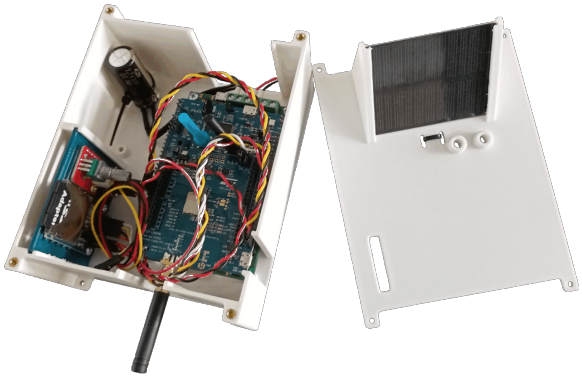

Figure 3: The ACPS hardware extends an off-the-shelf development board with custom solar energy-harvesting and a diverse set of sensors.

3.1 The Obvious Case for Real Hardware

Unsurprisingly, *the* most consistent finding is that students appreciate and demand using real hardware. During the undergraduate studies, most courses are theoretic and labs are done on paper—which in many cases is an obvious and reasonable choice—or consist of writing high-level, often math-centered code—e.g., in MATLAB or Python. Particularly in engineering-focused degree programs, students often seek practical engagement beyond theory and simulation. Working with tangible IoT and CPS platforms, including sensors and actuators, provides immediate, gratifying feedback—often more impactful than abstract computations:

> *The Lab serves as good practice to apply the concepts from the lecture. (SES 2023)*

Students also recognize the long-term benefits, citing increased job readiness and opportunities to reuse course knowledge in personal projects:

> *Also, the course is really advanced enough to make us more job market fit. (SES 2020)*

> *Very practice-oriented course. Former attendees report that the code is being reused for private projects. (SES 2023, translation from German)*

3.2 Synchronizing the Lecture and Lab

Students consistently value the tight alignment between lecture content and lab exercises. This coherence facilitates learning and demonstrates thoughtful course design:

> *The labs allow you to directly apply what you've learned in the lecture and work with the hardware. This deepens your understanding. (SES 2024, translation from German)*

Likewise, students appreciated additional content and explanations in the labs and appreciate their coherent design; e.g.:

> *The course was extremely well structured. Complementary to the lab materials. Lab materials covered everything from basics to slightly advanced level. (ACPS 2024)*

Even when labs are time-intensive, students acknowledge their educational value:

> *I really enjoyed the labs, even if they took a long time. The explanations given in class were also really clear, interesting and easy to follow. Thanks for a great semester :) (SES 2023)*

3.3 A Lack of Time and Hardware

A frequently mentioned nuisance is restricted hardware access. Several students expressed frustration over being unable to test code outside scheduled sessions:

> *I disliked not being able to test my code outside the available time slots. (SES 2023)*

Moreover, they mention the labs' high time-demand:

> *I had too [sic!] work a lot for the lab although I loved it. (SES 2020)*

> *Lab tasks require a high amount of time to complete (SES 2020)*

This stems partly from a mismatch between the designed workload (based on ECTS guidelines) and on-site supervised lab hours. While students are expected to invest 90 h per component (lecture

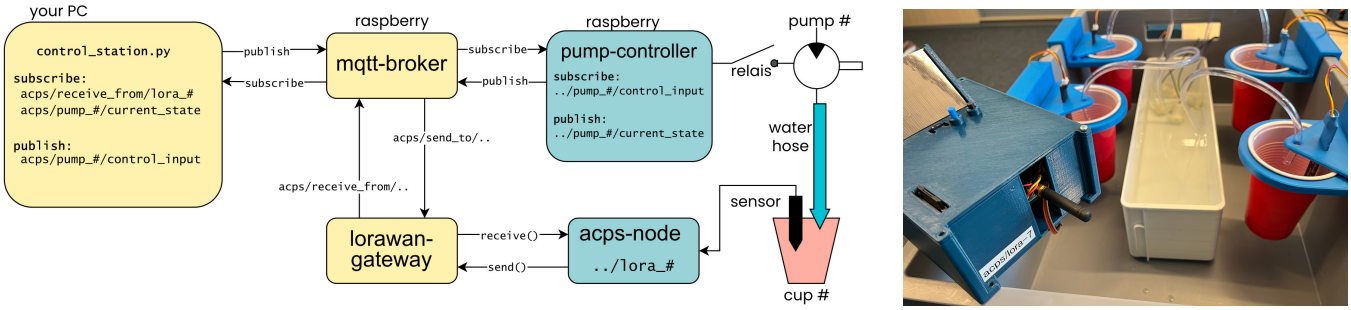

Figure 4: The ACPS course culminates in a networked plant-watering system, where students implement both the microcontroller firmware and a python server to collect data.

and lab), on-campus access is typically limited to 20 h to 30 h over the semester (13 weeks with 1.5 h to 2.25 h each). Thus, students request additional time slots (supervised or unsupervised), and are even willing to buy the hardware:

> *Being able to buy a copy of the board (or at least a kit with a breadboard and all necessary components) would have been really helpful. (SES 2023)*

> *I think you should change the boards to a commercial option so students are not [re]stricted by a certain schedule in order to test their code. (SES 2022)*

Likewise, they call for reusing the remote lab from the pandemic:

> *The idea with the remote lab was very good and should also be kept when the lectures are again held in person on the campus. (SES 2020)*

While appealing, these suggestions face practical challenges. Maintaining the remote lab proved resource-intensive and raised safety concerns (e.g., fire risk of custom-built, uncertified hardware when unattended). Additionally, the convenience of remote access may discourage students from attending in-person sessions, thereby limiting opportunities for hands-on learning, peer interaction, and real-time support. It also restricts our ability to monitor progress and offer immediate, tailored feedback—factors we consider essential to effective learning in hardware-centric courses.

On the other hand, commercial hardware limits the choice of sensors and actuators and creates dependency on external availability. We have explored open-sourcing our designs but have postponed this due to time constraints in maintaining documentation and support. However, we consider making the boards available for buying or renting, e.g, from the library. As a compromise, we provide daily checkout times, where students can work with the hardware unsupervised in our building. Given high dropout rates, long-term rentals are currently infeasible. However, we believe and see indication in the working habits of the students, that certain steps can and should be done without the effort. Conveying this understanding is a learning goal of our courses.

Nevertheless, we prototyped a virtual board with on SimAVR [11], a simulator for AVR microcontrollers, enabling students to test their solutions at home. Moreover, such a simulation enables unit-testing the embedded software, providing students with clear feedback if their solutions fits the requirement. While we consider sharing this prototype in next year's course, it comes with the risk of reducing student attendance, similar to the remote lab.

3.4 Mixed Prior Knowledge

One issue that we frequently face is that of very mixed prior knowledge and expectations. As a consequence, some students request more support:

> *one supported lab per week was not sufficient. (ACPS 2022)*

or are even frustrated with the overall experience:

> *The Lab is just such a MASSIVE time sink [...]. (SES 2022)*

At the same time, a few students considered some course contents as not new or a summary of their previous courses, stating in the evaluation that

> *Some of the examples in the lecture were discussed in too much detail. (SES 2024, translation from German)*

Likewise, some students complained about too short and fuzzy lab sheets, while others reported too long sheets with too detailed explanations.

Such discrepancies are difficult to fully resolve, given the diverse academic backgrounds of our cross-disciplinary audience. To help bridge knowledge gaps, we offer a self-paced C crash course and explicitly outline course requirements and expectations in the first lecture. Nonetheless, some students may still lack the foundational skills necessary to succeed in the lab. To minimize frustration and reduce late dropouts, we aim to identify these cases early by monitoring lab attendance and introducing mandatory milestones (e.g., toolchain setup prior to the first lab). These measures have increased awareness and preparedness. Ultimately, however, we maintain that students bear responsibility for evaluating course suitability and, if needed, making an informed decision to withdraw—ideally early in the semester.

3.5 Technological Progress and Discontinued Development

Another issue that we faced—and that is sometimes raised by students—is the ongoing technological progress and discontinued development and support of (open-source) projects. One example

is the use of Mbed OS, which will reach end of life in July 2026. We chose this platform because of STM' previous vivid development and support and hoped that it would outlive other embedded research OSs such as TinyOS. For the future, we will explore (hopefully more long-lived) alternatives. However, changing the OS entails a massive redesign and revision of a vast chunk of the course material (lecture slides, assignments, templates, sample solutions). This burden may force us to keep up Mbed OS as long as possible.

Another example is the use of low-power communication technologies, where IEEE 802.15.4 (used by ZigBee) and LoRa are two popular choices for home automation and industrial applications. Recently, other technologies such as BLE and Narrowband IoT (NB-IoT) are gaining traction, so that we intend to widen the scope in the lecture. However, we feel that we need to restrict at least the labs to one technology in order not overload the students with too many inputs. Moreover, not all technologies are equally suitable for small lab setups.

4 Conclusion

In this paper, we shared our course design for teaching programming on resource-constrained embedded and CPS devices, and the challenges of IoT systems. The curriculum is grounded in hands-on, hardware-based learning, which students value highly, though it also presents logistical and pedagogical challenges, particularly in accommodating diverse student backgrounds and ensuring hardware access. Feedback-informed refinements, such as synchronized lecture-lab design and milestone tracking, help balance structure with flexibility. Overall, the approach prioritizes practical skills, environmental relevance, and scalable teaching practices to prepare students for real-world IoT development.

Acknowledgments

The authors would like to thank all current and former team members involved with the courses presented and discussed in this paper. In specific, we want to thank Prof. Turau and his team for sharing all material of SES and handing over this well-thought-out course to us. A warm thank you goes to our former team member Christian Busse for revising the ACPS lab contents and material. Finally, we would like to thank our students for the provided feedback over the last decade, which has helped us tremendously to improve and fine-tune the courses.

References

[1] Bilal Babayigit and Mohammed Abubaker. 2024. Industrial Internet of Things: A Review of Improvements Over Traditional SCADA Systems for Industrial Automation. *IEEE Systems Journal* 18, 1 (2024), 120–133. doi:10.1109/JSYST.2023.3270620

[2] Bitcraze. [n. d.]. bitcraze Crazyflie 2.1+ Product Website. https://store.bitcraze.io/collections/platforms/products/crazyflie-2-1-plus Last visited: 2025/05/23.

[3] Alessandro Cammarano, Chiara Petrioli, and Dora Spenza. 2016. Online Energy Harvesting Prediction in Environmentally-Powered Wireless Sensor Networks. *IEEE Sensors* 16, 17 (2016). doi:10.1109/JSEN.2016.2587220

[4] Volker Turau Christian Renner, Stefan Unterschütz and Kay Römer. 2014. Perpetual Data Collection with Energy-Harvesting Sensor Networks. *Transactions on Sensor Networks (TOSN)* 11, 1 (November 2014), 12:1–12:45. http://www.bcrenner.de/publications/2014_TOSN_PerpetualDataCollection.pdf

[5] Anna Förster, Jens Dede, Andreas Könsgen, Asanga Udugama, and Idrees Zaman. 2017. TEACHING THE INTERNET OF THINGS. *GetMobile: Mobile Comp. and Comm.* 20, 3 (Jan. 2017). doi:10.1145/3036699.3036707

[6] Josiah Hester and Jacob Sorber. 2017. The Future of Sensing is Batteryless, Intermittent, and Awesome. In *Proc. of the 15th ACM Conference on Embedded Network Sensor Systems (SenSys)*. ACM, Delft, Netherlands, Article 21, 6 pages. doi:10.1145/3131672.3131699

[7] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani Srivastava. 2007. Power Management in Energy Harvesting Sensor Networks. *Transactions on Embedded Computing Systems (TECS)* 6, 4 (Sept. 2007).

[8] Holger Karl and Andreas Willig. 2007. *Protocols and architectures for wireless sensor networks.* John Wiley & Sons.

[9] Sameer Qazi, Bilal A. Khawaja, and Qazi Umar Farooq. 2022. IoT-Equipped and AI-Enabled Next Generation Smart Agriculture: A Critical Review, Current Challenges and Future Trends. *IEEE Access* 10 (2022), 21219–21235. doi:10.1109/ACCESS.2022.3152544

[10] Abderahman Rejeb, Karim Rejeb, Steve Simske, Horst Treiblmaier, and Suhaiza Zailani. 2022. The big picture on the internet of things and the smart city: a review of what we know and what we need to know. *Internet of Things* 19 (2022), 100565. doi:10.1016/j.iot.2022.100565

[11] SimAVR [n. d.]. simavr – a lean and mean Atmel AVR simulator for linux. https://github.com/buserror/simavr Last visited: 2025/05/27.

[12] STMicroelectronics. [n. d.]. B-L072Z-LRWAN1: STM32L0 Discovery kit LoRa, Sigfox, low-power wireless. https://www.st.com/en/evaluation-tools/b-l072z-lrwan1.html

[13] Wenpeng Wang, Victor A. Leal Sobral, Md Fazlay Rabbi Masum Billah, Nurani Saoda, Nabeel Nasir, and Bradford Campbell. 2023. Low Power but High Energy: The Looming Costs of Billions of Smart Devices. *SIGENERGY Energy Inform. Rev.* 3, 3 (Oct. 2023), 10–14. doi:10.1145/3630614.3630617