"Do cool stuff!" – The Incredible Charm of Open-Ended Tasks

Felix Büsching
f.buesching@ostfalia.de
Ostfalia – University of Applied Sciences
Wolfenbüttel, Germany

Samira Nazemi
s.nazemi@ostfalia.de
Ostfalia – University of Applied Sciences
Wolfenbüttel, Germany

Mia Sörensen
m.soerensen@ostfalia.de
Ostfalia – University of Applied Sciences
Wolfenbüttel, Germany

Maurice-Christian Ziesmann
mauricezie@googlemail.com
Ostfalia – University of Applied Sciences
Wolfenbüttel, Germany

ABSTRACT

This paper deals with modern didactic approaches for teaching the fundamentals, possibilities, and applications of IoT in a university context. Teaching IoT is particularly challenging because it combines a variety of technical disciplines – from computer architecture and hardware-related programming to communication technologies, middleware, and backend development. Building on solid theoretical and practical foundations, such as microcontrollers and their programming, the authors present a multi-stage teaching concept that combines structured tasks with exploratory, open-ended project work. Particular emphasis is placed on advanced, open-ended tasks in which students work independently on projects developed individually or within the framework of overarching themes. This approach promotes creativity, intrinsic motivation, and practical problem-solving skills, but presents teachers with the didactic challenge of fair and comparable assessment. The evaluation shows a high level of satisfaction among students, but this is offset by an increased amount of supervision required. In the discussion, both approaches – completely open versus thematically structured – are compared and their respective advantages and disadvantages are highlighted. The conclusion emphasizes that although open, exploratory tasks are more time-consuming in terms of supervision, they can significantly increase student learning success and the sustainability of project results.

CCS CONCEPTS

• **Computer systems organization** → *Embedded hardware*; *Special purpose systems*; • **Applied computing** → **Collaborative learning**; • **Hardware** → *Wireless integrated network sensors.*

KEYWORDS

WSN, IoT, Teaching, Lab, Experiments, Learning

1 INTRODUCTION

The teaching of Internet of Things (IoT) fundamentals, possibilities, and applications is diverse. In general, the question of what constitutes good IoT teaching can be answered very differently depending on one's point of view. It is usually undisputed that certain fundamentals are needed in order to build on them. Thus, we cannot start from scratch in IoT teaching either. However, this immediately raises the question of which perspective to take on IoT: From the perspective of computer engineering and electrical engineering, one could start with the hardware used and its possibilities and limitations:

- What are suitable computer architectures?
- How can battery-powered devices achieve the longest possible service life?
- What are suitable and appropriate sampling rates for connected sensors?
- How can clock gating be used to switch off certain areas in order to save as much energy as possible?

These are just a few of the questions that can be addressed in terms of hardware and hardware-related programming. Coming from the field of communications technology, one is likely to be more interested in (energy)-efficient data transmission:

- What are suitable protocols for data exchange between nodes and, if applicable, a gateway and/or backend?
- Which frequency bands and data rates are necessary and useful?
- How much energy does which radio range cost me?

With more focus on software and its development, you are likely to think more in terms of frameworks, middleware, and applications:

- What are suitable operating systems for IoT devices?
- What should a backend look like and what functions should it provide?
- Which interfaces should be offered and used?

All these questions are only examples and by no means exhaustive – they are merely intended to illustrate the broad range of issues relevant to IoT.

Since it seems impossible to cover all these questions in sufficient depth in a single course, we would like to present a concept in this paper that consists of courses that build on each other and are loosely interlinked, ultimately enabling exploratory IoT teaching. To this end, we will first present the necessary theoretical fundamentals (Section 2) and consider the practical basics in Section 3 in order to ultimately arrive at the actual topic, namely open-ended tasks in advanced courses in Section 4. The evaluation of the feedback from students is given in Section 5. We conclude this paper with a discussion and outlook in Section 6.

2 THEORETICAL FOUNDATIONS: COMPUTER ARCHITECTURES AND COMPUTER NETWORKS

In order to program microcontrollers efficiently – which are, after all, the core component of IoT devices – we believe that a certain understanding of computer architectures is necessary. At the very least, a basic understanding of how computers work in principle and what memory and registers are will be extremely helpful when implementing them in real systems later on. Ideally, students already have a basic knowledge of digital electronics (gates, flip-flops, full adders, etc.) and then learn the basics of data paths, registers, and control units in *Computer Architectures*, for example. We do this using Tanenbaums abstract MIC-1 architecture [14] but other (simple!) architectures such as MIPS [10] or AVR [2] are certainly just as suitable.

Since we mainly recruit our students from the field of electrical engineering, they learn from simple and familiar elements of electrical engineering (transistors) to digital electronics (gates) and computer architectures (data path, etc.) that a computer does not work "magically" and that they could (at least theoretically) build it completely themselves from elements they are familiar with.

Another field that should at least be covered to some extent is *communication networks*: basics of packet switching, the ISO/OSI layer model, and examples of protocols at each layer –possibly using wired TCP/IP networks with underlying Ethernet – is certainly a useful foundation. This can and should then also serve as the basis for understanding wireless communication and its specific challenges.

Equipped with these basics, we can venture into practical fundamentals:

3 PRACTICAL FUNDAMENTALS: MICROCONTROLLERS

In the *Information and Electrical Engineering* department at Ostfalia, the course is called *Data Technology Laboratory* and is rewarded by 2 Credit Points (CPs) in the European Credit Transfer and Accumulation System (ECTS); at TU Braunschweig, it was called *Practical Course in Embedded Systems Programming*; elsewhere, it is probably simply called *Microcontrollers.* This is a practical course in which students program simple Microcontroller (μC) and thus acquire knowledge about programming μC by completing specified tasks. We use simple μC such as the STM32 architecture [11] or, in other courses, AVR architecture, in order to finally apply the theoretical basics. Fundamental topics are covered, such as reading from and writing to registers, debugging by reading register values via a debug interface, and the use of logic analyzers. The individual tasks are structured in such a way that visible success is achieved quickly: the LED lights up, the display outputs text, and the Analog to Digital Converter (ADC), to which a temperature sensor is connected, is read, converted, and displayed. Only in a final task is there freedom through optional task components (such as setting the backlight or implementing small gimmicks).

Students who have no previous experience in the field of μC programming will learn the basics of programming in C and debugging small projects. Students with prior knowledge who have already gained experience with μC will learn a more structured approach and can prove themselves in the optional parts of the final assignment.

Our 2-CP course *Data Technology Laboratory* consists of 6 subtasks designed to introduce students to μC programming:

(1) Simple polling: input via button, output via LED
(2) Use of logic analyzers: measuring given UART signals
(3) Interrupts and timers: implementation compared to polling
(4) Assembly & disassembly: viewing and analyzing assembler code with different optimizations
(5) Connection of an LCD and use of USART
(6) Final project combining what has been learned: Read out temperature sensor connected to ADC and write formatted to display; free extensions

Since the coronavirus pandemic, we have also introduced the toolchain to run not only on the lab computers, but also on the students' computers. Code and documentation are submitted via GitLab. This means that students can complete the tasks anywhere in the world. To further motivate them, they are given the μC boards as gifts in the hope that they will later use them to implement further projects of their own.

4 ADVANCED IOT: OPEN-ENDED TASKS

The fundamental courses aimed to equip students with the necessary preliminary knowledge. There were clearly structured subtasks that had to be completed and solved. This led to clearly defined requirements and evaluation criteria – but also to many standard solutions, some of which were copied from previous semesters. This means that either the tasks are changed continuously each semester so that simply copying the solutions from the previous semester is not possible without at least some thought, or intensive code reviews are carried out with the students to check their actual understanding. The latter, in turn, leads to assessment criteria that are no longer entirely clear.

To address this problem, at least in the advanced courses, we have decided to allow students to work on self-defined tasks. In the event that there are no (or only a few) ideas, we also provide examples and suggestions. However, the declared goal is for students to get involved in the task identification process in order to stimulate their motivation to solve the tasks.

We tried out different approaches in various courses, which we would like to briefly report on here:

4.1 Completely open-ended tasks

In the master's course *Wireless Networking Lab* at TU Braunschweig, we provided participants with INGA [5] sensor nodes. Contiki [7] or RIOT [1] were used as operating systems. After a brief motivation and introduction to programming, the simple task was: *Do cool stuff!*

The work could be done alone or in teams of 2 to 5 people, and there were basically no restrictions on what kind of project the students had to come up with themselves. The students were asked to first write a short exposé in which they had to explain their goal. This is also the first opportunity for supervisors to exert a degree of control: Although the declared goal is to promote "to

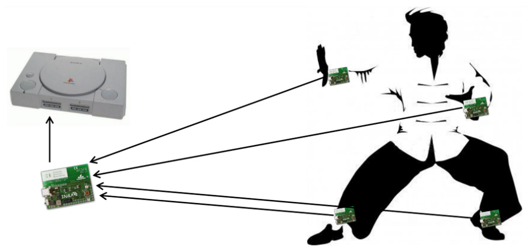


Figure 1: Tekken 3 - Controller – BAN of 5 WSN nodes.

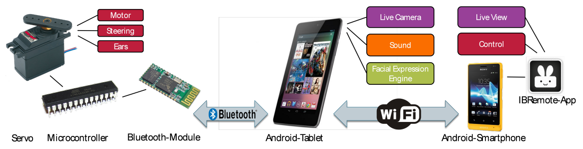


Figure 2: Control concept of remote-controlled rabbit.

each according to their abilities, to each according to their needs,"[1] certain minimum criteria should be met so that a team of five does not set itself the task of implementing an LED flashing light in five minutes. However, experience has shown that students tend to choose tasks that are too complex rather than too simple.

These completely open-ended tasks resulted in very diverse and completely different projects that covered a wide range of aspects of IoT, depending on the interests of the students involved: node hardware, node and backend software, connected sensors and actuators, communication protocols, middleware, and visualization. The examples listed here are just three of over 50 – they are purely illustrative and are only intended to show the possible range. All of the projects presented here were subsequently used for publicity purposes at events such as open days:

4.1.1 *Body Area Network (BAN) controllers for PlayStation.*

A team of two students built a BAN consisting of four Wireless Sensor Network (WSN) nodes each and connected them wirelessly to a PlayStation controller. This allowed the fighters in a Tekken 3 game to be controlled and virtual battles to be fought using body movements, as shown in Figure 1.

4.1.2 *Remote-controlled rabbit.*

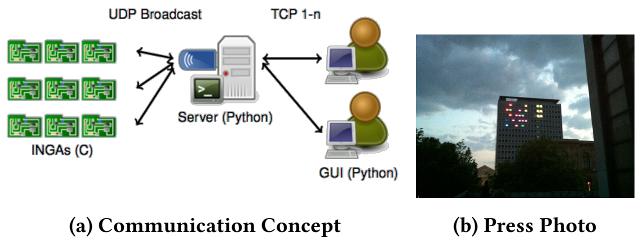
A team of three students implemented a mobile rabbit (Figure 2) that could be controlled via a smartphone. Various sensors and actuators were connected, including a mechanism for dropping rabbit droppings (candy) and a live video transmission.

4.1.3 *Blinkenlights.*

The original *Blinkenlights*[2] project started in 2001 in Berlin and claimed to be world's biggest interactive computer display. A total of 18 times 8 pixel where generated by large lamps, facing to the windows of an empty building – with a huge effort in cabling.

The distributed approach – to be seen in Figure 3 – of a group of 4 students based on a WSN, whereas a variable number of lamps can be connected to any node. The lamps used were part of a decommissioned perimeter advertising display belonging to a local soccer club, which the students organized themselves.

[1]Karl Marx: *Critique of the Gotha Program* (1875)
[2]http://blinkenlights.net/blinkenlights/



(a) Communication Concept (b) Press Photo

Figure 3: Implemented concept of a distributed Blinkenlights lighting control

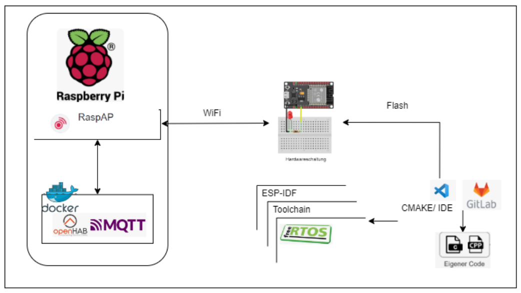


Figure 4: Workflow and Components in Embedded Systems – Smart Home Alarm System.

Other (initial) student projects led to separate publications such as *RAIM: Redundant Array of Independent Motes* [13], *RATFAT: Real-Time FAT for Cooperative Multitasking Environments in WSNs* [12], and *A Smart Spa: Having Fun with Physical Activities* [4].

4.2 Given general topic

We took a slightly different approach in the *Embedded Systems* bachelor's course at Ostfalia. Here, we also wanted to allow students to choose their tasks as freely as possible, but these tasks had to function under a common general topic. Here, too, the tasks of the individual groups varied, but the framework was more narrowly defined.

For both examples from the last two years, ESP32 nodes (RISC-V [8]) with FreeRTOS [3] as the operating system were used as IoT devices. Three years ago we used RIOT-OS on ESP32 nodes. The minimum requirement communicated was to connect at least one sensor and/or actuator to the ESP32 and integrate it into an overall system.

In order to familiarize students with the necessary basics, the frameworks and protocols used, presentations were given in the first fifth of the course: Groups (2-3 people) prepared a short presentation on the respective topic and were then considered "experts" and contact persons for the other groups.

In a final presentation, all groups demonstrate the functionality and interaction of the components they have developed in a joint demonstration.

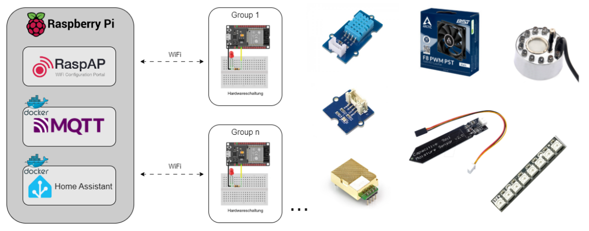


Figure 5: Components ueded in Growbox Scenario.

4.2.1 Smart home alarm system. An introductory presentation covered topics such as FreeRTOS, REST APIs, MQTT, OpenHAB, RISC-V, and multithreading, which were then divided among the six groups.

The tasks to be selected and implemented should all fall under the given umbrella topic of a smart home alarm system. To this end, the students were provided with various sensors (CO_2, IR, radar, ultrasound) and actuators (speakers, display, LED light), which they were free to use, but they were also encouraged to contribute their own suggestions and ideas. One group, for example, used a KNX connection to control KNX devices actually available in the laboratory via the OpenHAB [9] appliance.

For development within the team, each group was provided with a Raspberry Pi with an SD card. The software components necessary for MQTT communication were already pre-installed on the card: RaspAP, MQTT Broker, Open HAB – see Figure 4. This allowed the overall communication to be tested on a small scale before the final meetings focused on concrete coordination and cooperation between the groups.

4.2.2 Growboxes. For this implementation, we switched from OpenHAB to Home Assistant and chose a new overarching theme: a smart greenhouse – see Figure 5.

Our motivation for changing the main topic was based on several assumptions:

- If we had kept the same main topic as last semester, some students would have been tempted to borrow tasks and solutions from fellow students in higher semesters.
- A new main topic brings new approaches from the students, which in turn lead to interesting questions and tasks.
- Student projects of this scope are also suitable for public presentations and can thus be reused in a variety of ways.

The sensors and actuators we initially provided consisted of air and soil moisture sensors, UV LED strips, brightness sensors, and a fan. The students themselves came up with the idea of a water pump and designed a complex ventilation duct with several air pressure sensors and control options. Otherwise, the setup and procedure were similar to those described in the previous section.

4.3 Similarities and differences between the two approaches

In both approaches, the first few hours are spent introducing students to basic concepts and ideas from the world of IoT. This also fills remaining theoretical gaps and actively shapes the transition from previous courses to the open-ended tasks. In addition, this joint familiarization and preparation serves as inspiration and a source of ideas for later tasks. This usually takes place in a lecture-style format over 2 to 4 sessions, but in the laboratory facilities so that students can immediately implement and try out what they have learned.

Both approaches – i.e., completely open-ended assignments and a narrower framework provided by a given overarching theme – have their charms.

In the first case, students can incorporate their own problems from everyday life and thus find assignments that one would never have come up with oneself. This expands the scope and creates a high level of satisfaction among students: they work on what they want to and learn a lot in the process. Experience also shows that with so many different results and approaches, the likelihood increases that some of the projects will be pursued further or even published. The presentations of the results are just as diverse as the different tasks. This does not always make a comparative assessment easy. There may also be a certain imbalance in terms of the scope of the work and the results presented.

The presentation of results under a given overarching theme, on the other hand, is more structured and easier to plan. Even if the results are not as diverse as with completely open tasks, other important aspects come into play here: cross-group collaboration. In addition, the amount of supervision required is lower, as the groups can support each other better when they are working on similar topics or under an overarching theme. Due to the somewhat narrower framework, the assesment is also slightly less complex.

4.4 Assessment of performance

The respective study or examination regulations often require individual grading of student performance. Sometimes team assessments are permitted. For clearly defined tasks in basic courses, uniform assessment is relatively straightforward: requirements for the type, depth, and correctness of the various implementations are defined, and the individual submissions can then be assessed according to a scheme. However, difficulties arise here if it is suspected or recognized that model solutions may have been used. At this point, at the latest, an additional code review or colloquium is necessary, which at least softens the previous uniformity.

The self-selected tasks of the open assignments proclaimed here are certainly more difficult to compare with each other. When choosing tasks, care must be taken to ensure that the respective level of creative achievement is appropriate. In order to get an impression of individual commitment, it is probably unavoidable in this type of teaching that a teacher is at least partially present while the work is being carried out.

In the courses currently being held with a given general topic, we are currently evaluating the following performance criteria, which are then calculated into an individual overall grade:

- Presentation on the given topic (in terms of depth, comprehensibility, presentation, etc.)
- Implementation of the self-selected task (functionality, scope, code quality, etc.)
- Documentation and presentation of the self-selected task
- Integration into the given main topic (technical, collaboration with other groups, etc.)

In this way, we try to bring at least some structure to the assessment using a given scheme.

In the event that it was not immediately obvious in group work which person had which part in the joint solution, we also allowed the students within a group to partially evaluate themselves: In the form of a questionnaire to be completed anonymously, each team member had to evaluate both their own contribution and the contribution of the other team members in terms of scope and quality for each subtask. The results of these "votes" were surprisingly consistent and served as further input for the assessors' evaluation schemes. This approach also helped in the case of disputed groups, where each team member was asked to anonymously assess themselves and the members. Here too, a good level of consistency was observed, which in the vast majority of cases coincided with the assessors' impressions.

5 EVALUATION AND FEEDBACK

All courses have been and continue to be evaluated regularly by students according to a standard scheme. This shows a consistently high level of student satisfaction with the respective course content and structure. In the free text fields of the questionnaires, some respondents stated that they personally liked the type of tasks and that these contributed to their learning success. On the other hand, there were no opposing opinions. In other words, no one actively called for a return to standardized lab experiments.

However, since these evaluations were completed by different groups at different times, they are not really comparable with each other.

The impression among the teaching staff is similar. However, in discussions, the above-average supervision effort required for exploratory work, as opposed to standard laboratory experiments, is noted: there is usually no model solution and all participants have to rethink their approach. We were able to compensate for this by allowing all those involved in teaching (research assistants and student assistants) to influence the overall topic or define one together in order to strengthen their own motivation.

Feedback on the respective stage of progress is provided by the teaching staff during the course. After the presentation, all participants currently only ask questions to the individual groups – occasionally, this also leads to a fruitful discussion.

6 DISCUSSION AND CONCLUSION

While completely open-ended assignments offer the most freedom, assignments under a given overarching theme are more structured. The former may be more suitable for master's students, while the latter is probably challenging enough for bachelor's students.

In both cases, assessment cannot be 100 % transparent. It is always influenced by the teacher's perception, which means that absolute neutrality is not possible. On the other hand, except for simple math problems, where is that the case?

We will continue the course in a similar form in the coming semesters. However, there is still room for improvement, especially for a final group-individual feedback round.

The undeniable extra effort involved in open-ended assignments cannot be ignored: clearly structured laboratory experiments are easier to supervise and evaluate. However, the additional effort also leads to increased benefits: Not only are the students' general problem-solving skills strengthened [6], but the possibility of reusing individual projects should also be emphasized here. In the past, this has led to the creation of several high-profile demonstrators (e.g. Figures 1 to 3) and publications that ultimately emerged from open-ended tasks in IoT teaching.

ACKNOWLEDGMENTS

This work was partially funded by Land Niedersachsen (NBank) under grant number *ZW 787014063*.

REFERENCES

[1] Emmanuel Baccelli, Cenk Gündoğan, Oliver Hahm, Peter Kietzmann, Martine S. Lenders, Hauke Petersen, Kaspar Schleiser, Thomas C. Schmidt, and Matthias Wählisch. 2018. RIOT: An Open Source Operating System for Low-End Embedded Devices in the IoT. *IEEE Internet of Things Journal* 5, 6 (2018), 4428–4440. https://doi.org/10.1109/JIOT.2018.2815038

[2] Richard H Barnett, Sarah Cox, and Larry O'Cull. 2006. *Embedded C programming and the Atmel AVR.* Thomson Delmar Learning. https://dl.acm.org/doi/book/10.5555/1197726

[3] Richard Barry. 2009. *FreeRTOS reference manual: API functions and configuration options.* Real Time Engineers Limited.

[4] Felix Büsching, Nicole Holzhauser, Peter Knapp, and Lars Wolf. 2016. A smart spa: having fun with physical activities. In *Proceedings of the 2nd Workshop on Experiences in the Design and Implementation of Smart Objects* (New York City, New York) *(SmartObjects '16).* Association for Computing Machinery, New York, NY, USA, 1–5. https://doi.org/10.1145/2980147.2980149

[5] Felix Büsching, Ulf Kulau, and Lars Wolf. 2012. Architecture and evaluation of INGA an inexpensive node for general applications. In *SENSORS, 2012 IEEE.* 1–4. https://doi.org/10.1109/ICSENS.2012.6411295

[6] Man Ching Esther Chan and David Clarke. 2017. Structured affordances in the use of open-ended tasks to facilitate collaborative problem solving. *ZDM Mathematics Education* 49, 6 (2017), 951–963. https://doi.org/10.1007/s11858-017-0876-2

[7] A. Dunkels, B. Gronvall, and T. Voigt. 2004. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *29th Annual IEEE International Conference on Local Computer Networks.* 455–462. https://doi.org/10.1109/LCN.2004.38

[8] Riya Gautam, Dhyanik Pujara, Maurya Shah, and Dhaval Shah. 2024. Adapting a Real-Time Operating System to the RISC-V Based ESP32. In *Smart Trends in Computing and Communications*, Tomonobu Senjyu, Chakchai So-In, and Amit Joshi (Eds.). Springer Nature Singapore, Singapore, 459–468.

[9] Florian Heimgaertner, Stefan Hettich, Oliver Kohlbacher, and Michael Menth. 2017. Scaling home automation to public buildings: A distributed multiuser setup for OpenHAB 2. In *2017 Global Internet of Things Summit (GIoTS).* 1–6. https://doi.org/10.1109/GIOTS.2017.8016235

[10] John Hennessy, Norman Jouppi, Steven Przybylski, Christopher Rowen, Thomas Gross, Forest Baskett, and John Gill. 1982. MIPS: A microprocessor architecture. *SIGMICRO Newsl.* 13, 4 (Oct. 1982), 17–22. https://doi.org/10.1145/1014194.800930

[11] Patrik Jacko, Matej Bereš, Irena Kováčová, Ján Molnár, Tibor Vince, Jozef Dziak, Branislav Fecko, Šimon Gans, and Dobroslav Kováč. 2022. Remote IoT Education Laboratory for Microcontrollers Based on the STM32 Chips. *Sensors* 22, 4 (2022). https://doi.org/10.3390/s22041440

[12] Sebastian Schildt, Wolf-Bastian Pöttner, Felix Büsching, and Lars Wolf. 2013. RATFAT: Real-Time FAT for Cooperative Multitasking Environments in WSNs. In *2013 IEEE International Conference on Distributed Computing in Sensor Systems.* 388–393. https://doi.org/10.1109/DCOSS.2013.70

[13] Dominik Schürmann, Felix Büsching, Sebastian Willenborg, and Lars Wolf. 2017. RAIM: Redundant array of independent motes. In *2017 International Conference on Networked Systems (NetSys).* 1–8. https://doi.org/10.1109/NetSys.2017.7903957

[14] Andrew S. Tanenbaum and Todd Austin. 2012. *Structured Computer Organization* (6th ed.). Prentice Hall Press, USA. https://dl.acm.org/doi/10.5555/2424048