TEACH: Transferable RTOS Education with Accessible MicroController Hardware

Christian Hakert*, Kay Heider*, Daniel Biebert*, Kuan-Hsun Chen†, Jian-Jia Chen*
*Department of Computer Science, TU Dortmund University, Germany
†Department of Computer Science, University of Twente, the Netherlands

***Abstract*—Comprehensive real-world environments for fundamental teaching of real-time operating-systems concepts are challenging to find, meeting the requirement of sufficient simplicity for allowing a focus on the relevant curricular concepts. Especially when it comes to implementations on real systems, a full-featured, yet simple to use platform is usually lacking. Despite the pure execution of a real-time operating-system, such a platform has to include proper I/O, such as a display, GPIO pins and buttons, and communication standards in order to deploy realistic workloads. Additionally, making the target platform accessible, including setting up the tool chain, installing device drivers and connecting the system can impose a further challenge.**

In this demo, we present an easily deployable RTOS teaching platform, namely TEACH, which is proven to work in three international teaching collaboration formats within Germany, the United States and the Netherlands. This platform consists of ESP32-based systems, namely the SQFMI Watchy and the ESP32-S3-DevKitC, which comprises the required I/O functionality and an accessible port of FreeRTOS. We equip these systems with software libraries to make the I/O functionality usable without steep learning curves and support dev containers and readily deployed remote access setups in order to minimize setup overheads. In combination with a collection of teaching assignments dedicated to this platform, we enable low overhead teaching for real-time operating-systems, which allows a focus on the fundamental concepts, escaping infrastructural overheads.

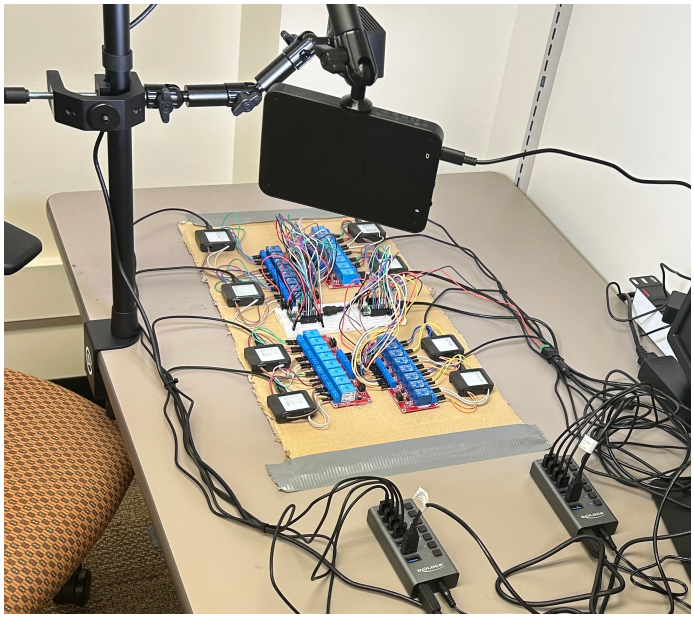

Fig. 1. Remote Setup for Online Access to the SQFMI Watchy

I. Overview

Teaching of real-time operating systems in the context of a university requires a suitable platform, which carefully needs to be tailored to the curricular needs. Overcomplicated tool chains, lacking functionality or bug prone environments can form a significant factor for making teaching more difficult. When students have to spend high amounts of time for the aforementioned points, the actual curricular goals may not be achieved. In consequence, we designed a dedicated platform for easily accessible, hassle-free teaching of low-level programming and real-time operating system, yet featuring sufficient functionality to create interesting and motivating applications and use cases.

The initial creation of this platform was conducted under the scope of the IVAC program for international virtual academic collaboration from the German Academic Exchange Service (DAAD)[1]. Despite acquirement of the needed hardware and implementation of proper templates, dev containers and programming environments, the IVAC program supported the integration of the developed platform in two collaborative teaching formats between the TU Dortmund University on the German side and the University of Cincinnati on the US side in 2023. After the execution of the IVAC project, the deployed platform was further transferred into a Dutch doctoral course, where it served as a basis for hands-on sessions and practical programming. This article gives some technical insights into the development from both activities.

[1]https://www.daad.de/de/infos-services-fuer-hochschulen/weiterfuehrende-infos-zu-daad-foerderprogrammen/ivac/

II. Case Study: Collaborative Teaching @ TU Dortmund and University of Cincinnati

TEACH was initially developed as part of a collaborative teaching effort between TU Dortmund University and University of Cincinnati for the fall semester of 2023. The setup was integrated into the graduate course "Real-Time Operating Systems Design and Implementation" at TU Dortmund University as the evaluation platform for all practical assignments. In this course, students learn fundamental concepts of real-time operating systems that are subsequently implemented in FreeRTOS and run on the setup provided by TEACH. Together with students from the University of Cincinnati, a joint exercise session was held where students of both universities worked on a shared assignment. Students of both

sides were able to evaluate their solutions directly on an SQFMI Watchy or via an established remote setup present at both universities.

The SQFMI Watchy[2] is an open-source wearable smartwatch. Its SOC is a ESP32-PICO-D4. It is equipped with an e-ink display, a 3-axis accelerometer, and a real-time clock. Furthermore, it has Wi-Fi, Bluetooth and USB connectivity. Lastly, it can be interacted with four tactile buttons. Given its range of features, the SQFMI Watchy can be used for diverse hands-on exercises. Since the watch is based on an ESP32, the ESP-IDF toolchain can be utilized to implement the concepts taught in the course. The framework provides a version of FreeRTOS for ESP32-based systems that supports unicore and multi-core setups. Additionally, the framework provides APIs to configure low-level hardware features, such as hardware timers and hardware-based interrupts. This allows for students to effortlessly configure the hardware and focus on achieving curricular goals.

To be able to accommodate students that could not receive a physical watch to test the experiments on, remote setups were deployed. The setups consisted of multiple watches, which could be programmed remotely. To view and interact with the watches, a live-stream of the watches was provided, which showed the e-ink display. To be able to interact with the watch through the buttons, button presses were simulated using an Arduino Nano[3] connected to the remote setup server. The button connections were bypassed through a set of relays controlled by the Arduino. By opening and closing the relays, the connection could be closed, just as a physical button press would. To further enable support for long presses, the length of the closed connection could be passed as a parameter to the Arduino. While these button presses could also be simulated in software on the watch itself, this might impact the concrete exercise. The button presses were therefore simulated using the Arduino as an independent source. A button press could be initiated directly next to the watch live-stream for ease of use. The setup is shown in Figure 1.

III. Case Study: Dutch Doctoral Course

In June 2024, TEACH was deployed as the core material for hands-on sessions in the doctoral course "Design and Implementation of Real-Time Systems," offered through the Advanced School for Computing and Imaging (ASCI), a Dutch research school focusing on high-quality research and education in computer and imaging systems. The course combines theoretical lectures with practical labs, where participants developed real-time applications on FreeRTOS[4]. Spanning four days of effective lecture time, the hands-on sessions took place every afternoon, directly aligning practical assignments with the content covered in the morning lectures.

As part of the course evaluation, ASCI conducted an anonymous survey involving 14 doctoral students from various Dutch universities, including TU Delft, University of Amsterdam, and Leiden University. The course content received an overall rating of 8.7/10. Notably, 11 out of 14 participants expressed high satisfaction with the hands-on sessions, highlighting the clarity and accessibility of the provided guidelines, which clearly showcase the applicability of the proposed teaching platform.

IV. Opportunities

Due to the positive reception of the developed teaching platform, we are further integrating and refining the platform to other curricular activities. In order to be able to provide an application where multiple systems have to interact and have to communicate, we are designing a simple robotic car, which will be part of the platform. As the basis, we take a development kit for a robotic car, namely the JoyCar[5]. This kit features simplicity and easy understandability of the components. It can be set up in a short time duration without specific knowledge. We replace the main board of the car by a custom main board, that hosts an ESP32-S3-DevKitC and allows programming the car with the same tool chain and software platform as the watch. Due to the employment of two separate instances of microcontrollers, namely the car and the watch, this platform allows serving for applications of wireless communication. Furthermore, the robotic car serves as a premier application of a cyber-physical system due to the employment of sensors and actuators. The current progress of the integration of the robotic car will be presented in RTSS@Work 2024.

Acknowledgment

This result is part of a project "WEAR: WEarable resource Aware programming and RTOS" (Project 57681578) that has received funding from the German Academic Exchange Service (DAAD) under the program "IVAC - International Virtual Academic Collaboration 2023", and an educational grant from ASCI Research School for a doctoral course named "Design and Implementation of Real-Time Systems".

[2]https://watchy.sqfmi.com/

[3]https://www.arduino.cc/

[4]More details about the course are available here: https://asci.school/project/asci-a31-design-and-implementation-of-real-time-systems-course/.

[5]https://joy-it.net/de/products/mb-joy-car