Combining Physical and Remote Hardware for Teaching Hardware-Oriented Programming

Daniel Versick; Marcus Soll; Louis Kobras

2025 7th Experiment@ International Conference (exp.at'25), 2025/2026

IEEE Xplore: https://ieeexplore.ieee.org/document/11348452

DOI: 10.1109/exp.at2565440.2025.11348452

Combining Physical and Remote Hardware for Teaching Hardware-Oriented Programming

Daniel Versick
NORDAKADEMIE gAG Hochschule der Wirtschaft
25337 Elmshorn, GERMANY
daniel.versick@nordakademie.de

Marcus Soll
NORDAKADEMIE gAG Hochschule der Wirtschaft
25337 Elmshorn, GERMANY
marcus.soll@nordakademie.de

Louis Kobras
NORDAKADEMIE gAG Hochschule der Wirtschaft
25337 Elmshorn, GERMANY
louis.kobras@nordakademie.de

***Abstract*—This demo presents an experiment for teaching hardware-oriented programming. The experiment contains both physical (in the form of a Joy-Pi system) and remote (3-axis portal) hardware. It focuses on the technical realisation (including the use of the CrossLab architecture for remote connection) and the course in which the experiment is embedded. First results show that the experiment was perceived as a valuable addition to the course and that the many usability aspects are well implemented.**

***Index Terms*—Hardware-orientated programming, GPIO, Remote Laboratory, Computer Science Education, Learning Outcomes**

I. Motivation and Goal of the Experiment

Laboratories of *hardware-oriented programming* in computer science are not yet well discussed, although singular papers exist (e.g., see [1]). A literature review in [2] showed that between 2017 and 2021 laboratories for hardware-oriented programming are not reported on. We therefore want to close the gap and show how we designed a hardware-oriented programming experiment combining physical and remote hardware.

We present an experiment focusing on teaching *General Purpose Input/Output* (GPIO) programming. GPIO pins are pins on a circuit that can be used for multiple purposes, hence the *general* in the name. They can be used to control a multitude of hardware models, from small self-built apparatuses to large industrial machines. We therefore believe that teaching GPIO programming to students is important, and thus aim to motivate other instructors to employ more experiments for GPIO programming while utilising of up-to-date technology.

II. Technical Description

We use the CrossLab architecture [3] for our experiment to provide hardware models students can use. The architecture was chosen since it allows for flexible remote experiments. At the same time, we want to provide students with hardware they can *play* with and hold in their hands. To realise this, we provide the students with a system that exposes two layers of GPIOs as two different chips:

1) One chip with its corresponding GPIOs is located physically on the Raspberry Pi. The GPIOs are connected to a Joy-Pi[1] (see Fig. 1, left side), thus offering the students a multitude of input/output options to work with.
2) The other chip is a virtual chip provided by a Linux kernel module. The GPIOs are virtually available[2] through the CrossLab architecture [3] and thus can be, in general, connected with a plethora of devices (for an example see Fig. 1, right side). The only prerequisite is that any target remote device implements the *Electrical Connection*

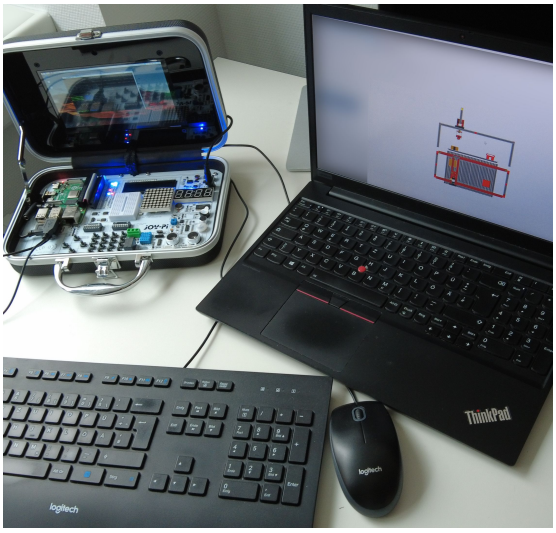

Fig. 1. Experiment setup: On the left side, the Joy-Pi can be seen and on the right side, a virtual three-axis-portal (see [4, Fig. 4]) can be seen.

This research was part of the project *Flexibel kombinierbare Cross-Reality Labore in der Hochschullehre: zukunftsfähige Kompetenzentwicklung für ein Lernen und Arbeiten 4.0 (CrossLab)*, which is funded by the *Stiftung Innovation in der Hochschullehre*, Germany.

[1] https://joy-pi.net/en/joypi-family, last accessed 2025-02-19.

[2] Utilising the Linux kernel driver `gpio-sim`: https://www.kernel.org/doc/html/latest/admin-guide/gpio/gpio-sim.html, last accessed 2025-02-24.

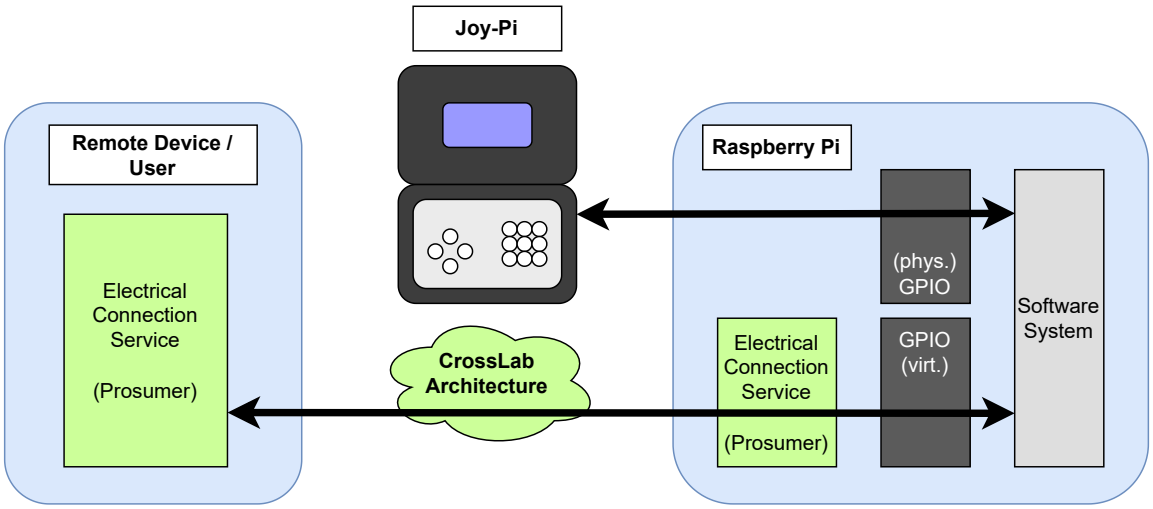

Fig. 2. The basic architecture of the experiment. Users can access both the Joy-Pi and the model through GPIO pins. The routing is different: One set is connected directly to the Joy-Pi, the other set is sent remote using the CrossLab architecture [3].

Service[3]. For our experiment, we use a 3-axis portal, both physical [5] and virtual [4, Fig. 4]. The virtual and physical system can be exchanged without any change for users.

The whole architecture of the experiment can be seen in Fig. 2. It is important to note that it does not make any difference to the user whether the GPIOs are physical or virtual, both behave exactly the same and can be accessed using any standard tools. The only difference is how it is handled by the operating system.

The setup, as described, allows for *Take-Home Laboratories*, enabling students to conduct part of the experiment at home. This is important in case of emergencies, as shown by the COVID19 pandemic.

III. Course Design

The laboratory system presented in the last section was used and evaluated in the course *Hardware-oriented programming*, which is taught during the third semester of the study programme *IT Engineering* at NORDAKADEMIE.

Main learning objectives of that course are to understand the concepts of the C programming language and its practical application for controlling hardware systems.

Students should deepen their theoretical knowledge in a practical experiment targeting the control of GPIO pins of a Raspberry Pi. They should learn to use those pins as both inputs and outputs. The 3-axis portal (physically or virtually) described earlier is controlled remotely, as a corresponding hardware system was not available locally. The Raspberry Pi operates under Linux. Since there are several ways to control the GPIO pins under Linux, students should select an interface that they consider suitable based on pre-defined requirements.

[3]The service can at the same time consume data and produce data, thus enabling bi-directional communication. Therefore, it is called a *prosumer*.

The goals of the course are defined as learning outcomes (cf. [6, Chapter 7]). After the experiment, students should therefore be able to:

- independently make simple architectural decisions based on defined requirements;
- understand and correctly use an existing C interface in this context and, in particular, correctly use the associated data structures;
- configure GPIO pins in the C programming language, read their value and set it;
- use a command line interface to communicate with GPIO pins for debugging;
- implement a C program that is able to control a 3-axis portal crane and fulfils essential requirements for readability and maintainability;
- write an experiment protocol.

The experiment is part of the course assessment and was divided into three phases:

1) In a preparatory phase, the students received a preparation sheet with a description of the objectives, requirements, and preparatory tasks, which had to be carried out independently in advance. This work had to be brought to the attendance phase of the experiment.
2) The second phase was conducted in the form of a 3-hour practical session. Students were given a Raspberry Pi and a Joy-Pi to implement their software system based on their preparations. The students could use this time to implement, debug, and improve the software.
3) Afterwards, the students had one week to summarise the preparation tasks, problems and results in an experimental protocol. They had to submit the protocol as part of the course assessment.

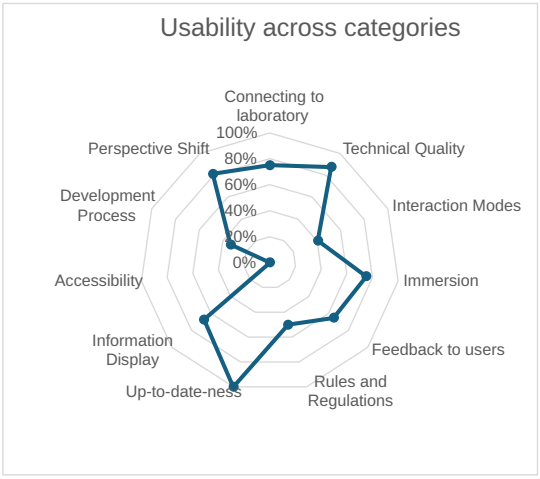

Fig. 3. Spider Plot of the usability evaluation conducted using the CrossLab usability checklist. Strengths lie in *Technical Quality*, *Immersion*, *Up-to-date-ness* and *Perspective Shift*. The greatest weakness lies with *Accessibility*.

Since the students have to complete all learning outcomes as a prerequisite for writing the experiment protocol, we consider the course to be applying *Constructive Alignment*.

IV. First Results

To evaluate the experiment, two dimensions were measured: student feedback and usability.

A. Student Feedback

Feedback from students collected during normal teaching evaluation indicated that they were generally happy with the experiment. Adding more practical elements through an experiment was perceived as a well-fitting expansion of the course. However, there were some problems with the hardware/operating system, where some pins would not respond or connecting to the internet was not possible. This indicated to us that we need to put more time into preparation and testing of the Joy-Pi and the operating system before the next run of the course.

B. Usability

To evaluate the usability of our experiment, we used the usability checklist (https://doi.org/10.5281/ZENODO.14329173) developed by the CrossLab project. The checklist is not designed to give a total score, but to show strengths and weaknesses of a system. Our results, shown in Fig. 3, indicated that many aspects of the experiment are already well implemented in terms of usability, especially *Technical Quality*, *Immersion*, *Up-to-date-ness* and *Perspective Shift*. However, *Interaction mode* should be improved since at the moment, students only get an editor or integrated development environment and no specific tools / help was available for the experiment. While this mirrors the situation often found in industry, it nevertheless is not satisfactory. Furthermore, the aspects *Rules and Regulations* and *Accessibility* should be improved in further iterations, as well as including usability more in the *Development Process*.

V. Contributions

With presenting the experiment, we make three main contributions:

1) We demonstrated that the CrossLab architecture allows for flexible experiments with virtual and remote hardware, as well as for *Take-Home Laboratories*.
2) We designed a didactically sound course including learning outcomes to ensure that students learn how to use GPIO.
3) We conducted a primary usability analysis, which is important since usability is often not regarded for remote experiments.

Acknowledgements

The source code of the CrossLab client on the Joy-Pi using the virtual GPIO driver can be found at https://github.com/Cross-Lab-Project/linux-gpio-device.

References

[1] E. Wen, S. Ma, P. Denny, E. Tempero, G. Weber, and Z. Yue, *KernelVM: Teaching Linux Kernel Programming through a Browser-Based Virtual Machine*. New York, NY, USA: Association for Computing Machinery, 2025, p. 1204–1210. [Online]. Available: https://doi.org/10.1145/3641554.3701831

[2] M. Soll, "What exactly is a laboratory in computer science?" in *2023 IEEE Global Engineering Education Conference (EDUCON)*, 2023, pp. 1–9.

[3] J. Nau and M. Soll, "An extendable microservice architecture for remotely coupled online laboratories," in *Open Science in Engineering*, M. E. Auer, R. Langmann, and T. Tsiatsos, Eds. Cham: Springer Nature Switzerland, 2023, pp. 97–109.

[4] J. Nau and K. Henke, "Goldi labs as fully integrated learning environment," in *2024 IEEE World Engineering Education Conference (EDUNINE)*, 2024, pp. 1–6.

[5] K. Henke, J. Nau, H.-D. Wuttke, M. Poliakov, G. Tabunshchyk, A. Parkhomenko, and O. Poliakov, "Expanding the remote experiment set with the 3axis portal physical model," *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 18, no. 04, p. pp. 21–30, 2022. [Online]. Available: https://online-journals.org/index.php/i-joe/article/view/28857

[6] J. Biggs and C. Tang, *Teaching for quality learning at university: what the student does*, 4th ed. Maidenhead, England New York, NY: Society for Research into Higher Education & Open University Press, 2011.