EMPIRICAL
Journal Section

Manipulating a CI/CD Pipeline in an IoT Embedded Project: A Quasi-Experiment

Igor Pereira[1] | Tiago Carneiro[2] | Eduardo Figueiredo[3]

[1]Computer and Systems Departament, Federal University of Ouro Preto, Minas Gerais, Brazil

[2]Computer Departament, Federal University of Ouro Preto, Minas Gerais, Brazil

[3]Computer Science Departament, Federal University of Minas Gerais, Minas Gerais, Brazil

Correspondence
Igor Pereira, Computer and Systems Departament, Federal University of Ouro Preto, João Monlevade, 35931-008, Minas Gerais, Brazil
Email: igormuzetti@ufop.edu.br

Funding information

Given the multidisciplinary complexity of embedded Internet of Things (IoT) projects and the demand for qualified professionals, this study investigates the influence of continuous integration and delivery (CI/CD) skills and developers' perceptions regarding applying these practices in this domain. We conducted a quasi-experiment with 98 students from 3 undergraduate courses at 2 Brazilian federal universities, analyzing the impact of developer skills on CI/CD. The results showed that developers with no previous CI/CD skills faced more significant difficulties in practical activities. It was interesting to note that most participants in our sample already had some experience with real software development projects. However, most have never had real experience with an embedded IoT project or CI/CD tools. The approach we followed resulted in 92% success. Attendees expressed interest in more hands-on training on CI/CD pipeline, DevOps, and embedded IoT projects. We also noticed a great need for them to have more practical experience with Git, GitHub, GitHub Actions, and GNU/Linux.

KEYWORDS
Continuous Integration (CI), Continuous Delivery (CD), Pipeline CI/CD, IoT Embedded Systems

1 | INTRODUCTION

The high investments of the global technology industry in Internet of Things (IoT) systems corroborate the diversity of applications in areas such as transportation, retail, industry, health, finance, oil and gas, energy, agriculture, and livestock [1]. IoT is a technological paradigm transforming how we interact with the surrounding environment, leading to significant multidisciplinary technological changes [2]. IoT systems interconnect machines and things to process information and offer intelligent services to users [3].

Software is fundamental to IoT systems, from firmware to edge and cloud computing infrastructures. For this reason, Software Engineering (SE) is crucial for designing, developing, delivering, deploying, and maintaining reliable IoT systems [4]. Since IoT systems sometimes deal directly with human lives and business processes, they require a high level of quality, and their development needs to be subject to reliable processes. These processes typically occur through various process models, such as traditional, agile, and distributed [5, 6, 7]. This work will focus on the distributed software development model, such as in FLOSS (Free/Libre Open Source Software) projects using solid software engineering practices. The software development model distributed through GitHub (GH) is driven by Pull Requests (PR) [8].

Integrating new contributors into distributed software development communities is vital to their sustainability and continued innovation [9]. However, the onboarding process can be challenging due to several barriers newcomers face. These barriers include broken expectations, reception issues, incorrect or outdated documentation, and a different learning curve than imagined [10]. In addition to these traditional challenges, a growing trend in these communities is adopting continuous integration and continuous delivery (CI/CD) pipelines on GH [11]. While these pipelines are essential to ensuring development quality and efficiency, correctly following pipeline procedures can present a new barrier for newcomers.

The pipeline CI/CD is typical in DevOps and automates actions from source code on the developer's computer to the target device in production. Additionally, there are DevOps tools that automate source code generation and deliver results to contributors and integrators [12]. DevOps is a collaborative and multidisciplinary organizational effort to automate the continuous delivery of new software updates, ensuring their correctness and reliability [13]. In turn, agile practices have a lightweight, iterative, and collaborative approach to systems development and have been used successfully in the software industry and the construction of embedded systems [14, 15, 16, 17].

IoT systems project teams are commonly multidisciplinary [18]. The backgrounds of these people are varied, and collaboration between the teams needs to be continuous [19]. Capturing qualified professionals in the different areas of these projects is challenging. Therefore, teaching and training approaches can be attractive to reduce the learning curve of a newly arrived professional who still needs to acquire all the skills necessary to develop the project [20]. This research investigates whether developers' prior skills in CI/CD challenge their performance, influence ease of use, and their perceptions during the configuration and execution of a pipeline in an IoT embedded systems project following a process model PR-driven development.

Several challenges persist in IoT embedded system development, highlighting the need for more extensive SE practices. While CI/CD adoption is increasing, its implementation in IoT projects remains complex due to domain-specific constraints, including sensor limitations, environmental conditions, and connectivity issues [21]. Additional difficulties arise from the necessity for multidisciplinary expertise, simulation environments, and test emulation [22]. Although there is growing recognition of SE principles in IoT, further empirical studies are needed to refine these methodologies [23].

To address these challenges, we conducted a quasi-experiment with undergraduate software engineering

students from 5 classes across 3 IT programs at two Brazilian Federal Universities [24]. The study involved 98 students, with approximately 80% having some professional software development experience. The quasi-experiment consisted of two phases: a theoretical lecture covering IoT embedded systems, CI/CD pipelines, and GitHub Actions (GHA), followed by a practical session in computer labs where at least two researchers assisted students. A pilot study with three students helped refine the instructional content and data collection methods. Quantitative analysis was performed using hypothesis testing and the Mann-Whitney test for independent non-parametric samples [25, 26]. Additionally, we collected open-ended responses, observational notes, and conducted qualitative analysis using Thematic Analysis [27].

Many studies in software engineering employ students as subjects as we do in this work[24, 28, 29]. Salman et al. (2015) compared students and professionals to understand how well students represent professionals as experimental subjects in software engineering research. They noted that except for minor differences, neither group is better than the other; both performed similarly when participating in test-driven development tasks as a first-time approach. They further emphasized that a carefully defined quasi-experiment in a new approach to students and professionals results in similar performances [30].

Finally, we found that only 8% of developers could not complete all the tasks in the practical class script. Those without prior knowledge of CI/CD took longer and encountered more challenges. The theoretical, practical, and mentoring approach resulted in 92% success. Doubts and barriers were expected, but some participants faced more difficulties, and many helped each other. Despite some familiarity with GH, practical use of the source code seemed limited, highlighting the newness of GHA for most. Dealing with GNU/Linux operating systems and version control through the terminal was also challenging for many participants. According to them, the quasi-experiment approach motivated them to seek further studies on CI/CD pipeline, embedded systems, and IoT technologies. It also revealed the need for participants to acquire more practical experience in technologies such as Git, GH, GHA, and GNU/Linux.

The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 contains the research methods we followed that led to the results we present in Section 4. Section 5 discusses the threats to validity and the mitigating actions we took. Final considerations and future work are part of Section 6.

2 | RELATED WORK

Given the rise in the use of GitHub Actions (GHA) in FLOSS projects [31, 11] and growing interest in IoT project studies [32, 33, 34], we chose GHA to implement a CI/CD pipeline in our quasi-experiment. López-Viana et al. (2020) highlight edge computing, which brings cloud services closer to devices and end users [35]. Zampetti et al. (2022) analyzed CI/CD in CPS FLOSS projects, finding 17 projects using GHA and identifying challenges in integrating simulators or Hardware-in-the-Loop (HiL) [36]. Our study employs a reusable Espressif Action to simulate hardware and test an IoT embedded system in a CI/CD pipeline.

López-Viana et al. (2022) explored GitOps for IoT edge solutions, revealing challenges like infrastructure provisioning and device limitations [37]. While their study focuses on high-end edge devices, our IoT system uses an ESP32 microcontroller with an RTOS, which lacks virtualization and containerization.

Moedt et al. (2023) found that only 40% of professionals in IoT use integration and continuous delivery, with most lacking hardware development knowledge [32]. Our study similarly explores continuous integration and delivery in embedded IoT systems, where participants also have limited hardware knowledge.

Zampetti et al. (2023) examined CI/CD in 10 organizations, highlighting challenges like balancing HiL

and simulators, and the need for interdisciplinary knowledge [33]. While our pipeline doesnt yet automate deployment, we demonstrated a prototype using ESP32 and conducted end-to-end tests with HiL.

Cristea and Paduraru (2023) found that including IoT in the curriculum increased student interest in software-hardware interactions [34]. Similarly, our research motivates participants to develop IoT systems using CI/CD practices.

Talekar and Harpale (2023) proposed a CI/CD pipeline for embedded systems but did not evaluate it with a case study [38]. Our study focuses on software engineering and evaluates a prototype, with future collaborations with hardware experts planned to expand the research.

3 | STUDY DESIGN

This section outlines the design of a quasi-experiment to evaluate developers' time, understanding, and challenges in configuring and using a CI/CD pipeline with GHA in an IoT embedded project, comparing those with and without prior CI/CD skills. The experiment had two stages: a theoretical class and a practical class where participants followed a script. Data was collected using printed forms designed for the study. The following sections describe the goal, research questions, hypotheses, and research method.

3.1 | Study Goal, Research Questions, and Hypotheses Formulation

We set the goal of our study using the Goal/Question/Metric (GQM) template of [39], as outlined below. Analyze prior skills in continuous integration and delivery for the purpose to evaluate with respect to time, difficulty, and perceptions from the point of view of developers in the context of adopting a CI/CD pipeline with GHA in an IoT embedded project.

To achieve our goal, we based our evaluation method on the following research questions.

RQ_1 - How does prior CI/CD knowledge impact the effort to configure, use, and understand a CI/CD pipeline in an IoT embedded project?

RQ_2 - What were developers' perceptions about the setting, use, and understanding of the CI/CD tasks?

We define hypotheses for RQ_1: prior knowledge in continuous integration and continuous delivery impacts on the configuration, use, and understanding of the files of a CI/CD pipeline in an embedded IoT project. To answer RQ_1, we evaluated the time spent on each task and the ease each developer reported in configuring, using, and understanding the CI/CD pipeline files in terms of the scale: 0 (very easy), 1 (easy), 2 (moderate), 3 (hard) and 4 (very hard). Then, RQ_1 was transformed into null and alternative hypotheses as follows. H_0: Prior knowledge about continuous integration and delivery does not impact the configuration and use of the CI/CD pipeline with GHA in an embedded IoT project. $H\alpha$: Prior knowledge about continuous integration and delivery impacts the configuration and use of the CI/CD pipeline with GHA in an embedded IoT project. H_0: Prior knowledge about continuous integration and delivery does not impact on understanding the files in a CI/CD pipeline with GHA in an embedded IoT project. $H\beta$: Prior knowledge about continuous integration and delivery does impact on understanding the files in a CI/CD pipeline with GHA in an embedded IoT project.

3.2 | Research Method

We planned and performed an empirical study to answer the research questions, as shown in Figure 1.

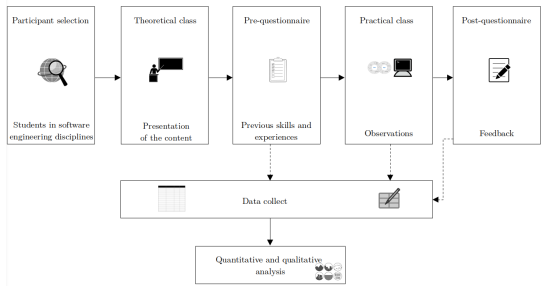


FIGURE 1 Study design

Participant selection. We selected undergraduate students from Computer Science, Computer Engineering, and Information Systems courses at 2 Brazilian Federal Universities (UFOP and UFMG). These courses are distributed across three campuses of these universities in three different cities in the state of Minas Gerais. The chosen subject in the 3 classes was Software Engineering. We ran the quasi-experiment 3 times to consolidate the data and extract our findings. 98 students performed the quasi-experiment. Section 4.1 presents an overview of the ninety participants completing the quasi-experiment.

Quasi-experiment design. First, we held a theoretical class on the topics involved in the practical class, such as DevOps, CI/CD Pipeline, IoT Embedded Systems, Git, GH, and GHA. The duration of the theoretical class was 100 minutes, consisting of explaining the subject through slides (available in our repository[1] for replication) and addressing students' doubts on demand. At the end of the theoretical class, a researcher implemented it in a prototype present in the laboratory to demonstrate the destination of the source code that they would manipulate in the practical class. The schematic of this prototype is available on the experiment's GH page and is shown in Figure 2.

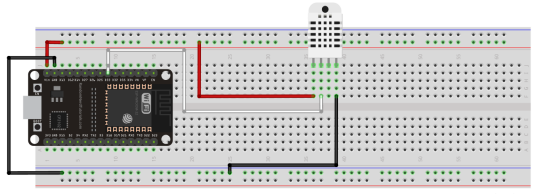


FIGURE 2 Prototype of an IoT Embedded System

Our replication package includes two directories, rq1 and rq2, containing raw questionnaire responses, descriptions, the tutorial on the GH page, and slides used in the theoretical class. The rq1 directory includes files for statistical calculations, boxplot generation, and the Mann-Whitney method implementation. The rq2 directory holds the spreadsheet with the steps for thematic analysis, including category occurrences that formed the map.

On the second day of the Software Engineering class, participants completed a demographic and historical

[1]https://zenodo.org/records/15053535

form, followed by tasks described in a script. After each task, they filled out a second form on their experiences and answered open-ended questions about the experiment. These forms are available in our repository for replication. This procedure was repeated in three Software Engineering classes at two universities.

We conducted a pilot study with three Computer Engineering students from a research laboratory, who are not part of the 98 developers in the quasi-experiment. The study considered factors influencing software developers' self-efficacy, as described by Ribeiro et al. (2023), to boost participants' confidence [40]. Based on pilot feedback, we adjusted the theoretical class and practical class script. The script is available online for future replications/extensions in the GH repository.

The IoT architecture considered here has four layers, each with distinct roles in data processing and management [41, 42, 43]. Our prototype is located in the edge layer, which processes data close to the source to reduce latency and network load. Embedded IoT systems, part of this layer, collect and process data locally before transmitting it to other network layers [? 44].

Experiment tasks. The practical class for this study was structured as a tutorial on GitHub (GH) with assistance from researchers to help participants configure and run a CI/CD pipeline on an IoT embedded system. The pipeline involved software for ambient temperature monitoring using the ESP-IDF library, C language, and the Criterion testing library. Participants completed three tasks:

(i) Configure a CI/CD Pipeline in GHA: Participants were instructed to create a classic token on GH, fork the project, clone it, and set up a CI/CD pipeline by creating a YAML file with the provided code. They used commands like mv, cd, ls, mkdir, and touch in a terminal, following the tutorial's guidance. Once configured, they ran git add, commit, and push to complete the setup.

(ii) Create a Pull Request with a Bug: Participants introduced a simple bug in the test by changing a value in the test.c file to an unaccepted temperature (e.g., 45řC). They then created a new branch, committed the change, and pushed it to GH. The CI/CD pipeline failed during the testing phase, showing that the pipeline halted without passing all tests.

(iii) Create a Pull Request with the Bug Fix: In the final task, participants corrected the bug by setting the sensor value to 26řC, committed and pushed the changes, and created a new pull request. The pipeline successfully ran through the build, test, and release stages, and the resulting artifacts were made available on the project's main page.

These tasks involved using the C programming language, the Espressif IoT Development Framework (ESP-IDF v5.0), the Criterion library for testing, and basic GitHub, Git, and file system commands. The goal was to provide participants with hands-on experience in program development, unit testing, and CI/CD pipeline configuration. The project involved the ESP32 microcontroller, the DHT22 sensor from Adafruit, and the FreeRTOS API. Participants reported their time and perceived difficulty for each task on a form.

Data collection. We collected data from the demographic and background form, the experimental task form, and observations noted by the researchers during the practical class. All data was analyzed, interpreted, and reported in the results. All forms and the practical guide are available in our online package for future replications and expansions of the work.

The pre-questionnaire consisted of 5 questions as shown below. The developers answered this form before running the practical class.

1) What is your undergraduate degree? Open

2) What is your experience including internship, academic projects, job market in software development? I have not experience ( ); I have experience of up to 6 months ( ); I have experience between 7 and 36 months ( ); I have more than 36 months of experience ( ).

3) What is your experience combining internship, academic projects, job market in embedded and IoT development? I have not experience ( ); I have experience of up to 6 months ( ); I have experience between 7 and 36 months ( ); I have more than 36 months of experience ( ).

4) Mark according to how you evaluate your theoretical and practical knowledge in relation to each of the following topics: C Programming Language; Version Control System; Software Testing; Continuous Integration; Continuous Delivery and Deployment; Electric circuits; Microcontrollers and Microprocessors; Embedded systems; Internet of Things. None ( ); Little ( ); Moderate ( ); Advanced ( ); Very advanced ( ).

5) Describe any other observations that you consider important about your training and practical experience in software engineering and/or in the development of IoT embedded systems. Open

The post-questionnaire had 7 questions, as represented below. The developers answered this form during and after the practical class. For questions 1 to 6, they had to choose one of the options: Very easy ( ); Easy ( ); Moderate ( ); Hard ( ); Very hard ( ). Questions 7 and 8 were discursive.

1) According to task #1, enter when you started and finished. According to the configuration, mark the degree of difficulty you found.

2) According to task #2, enter when you started and finished. According to the creation of a branch and PR with a bug, mark the degree of difficulty you found.

3) According to task #3, enter the start and finish. When creating a new branch and PR with the bug fix, mark the degree of difficulty.

4) Analyze the contents of the experiment-ci-cd.yml file. Which is the level of your understanding of the contents of this file.

5) Analyze the contents of the main.c file. What is the level of your understanding of it?

6) Analyze the contents of the test.c file. What is the level of your understanding of it?

7) What are your thoughts on CI/CD servers using GHA? Consider the automated way it builds, tests, and delivers a version of the application to the simulated target.

8) Would you like to comment on your experience? Do you have any questions or suggestions that you would like to share with us about the quasi-experiment? You can consider the script, the source code files (main.c and test.c), the CI/CD pipeline configuration file (experiment-ci-cd.yml), GH, or any other technical and dynamic aspect implemented.

Quantitative and qualitative analysis. We collected quantitative and qualitative data from forms and notes taken during practical classes. Section 4 presents the descriptive analysis of this data, the Mann-Whitney test [25, 26] and the thematic synthesis method [27]. We applied the Mann-Whitney test to check statistical significance and answer RQ_1. This test is non-parametric and can be used for two independent samples. To answer RQ_2, we created a thematic map.

Ethical considerations. This research involves experiments with human subjects. All participants consented to their responses being used in the study. They signed a term indicated by the Ethics and Research Committee of the institution's first author of this work. Regarding participant data, all sensitive information (i.e., names and registration numbers) was previously anonymized to guarantee participant privacy.

4 | STUDY RESULTS

This section presents the results for each research question of this study. These results provide insights into the participant's perspective.

4.1 | Participant Overview

We conducted a study with 98 participants to evaluate users' speed, difficulties, and perceptions in configuring and using a CI/CD pipeline in an IoT embedded system. However, 8 participants withdrew before completing the practical script. The remaining 90 developers were analyzed into 2 groups of participants, with forty-five people in each. One group with prior skills in integration and continuous delivery declared in the first part of the questionnaire, and the other group indicated they did not have prior skills on the topics. Table 1 shows participant profile information related to gender, degree course, and whether they have practical experience with software development and IoT embedded projects. Finally, it also presents whether they have prior continuous integration and delivery skills.

Only 19% of participants declared themselves to be female. This small portion of female students in our sample is related to the smallest number of women in the field of Computing in the world [45]. Even though gender equity in Computing has been discussed for years, the data available in Brazil is still insufficient. However, the Brazilian Computing Society (SBC) compiles some Higher Education[2] data. Its annual report supports the Girls in Computing[3] program to present a general overview of the topic in the country and encourage more women to enter the field of Computing.

The undergraduate course with the most female representatives was Information Systems. UFOP (Federal University of Ouro Preto) had students who participated in the experiment and were from Information Systems, Computer Science, and Computer Engineering courses. UFMG (Federal University of Minas Gerais) had students from the Information Systems course.

Participants reported experience in software engineering in internships, on the job, and in research and extension projects. We asked that these experiences be counted cumulatively and in months, as [29] suggests. Regarding software engineering experiences, most participants responded that they had practical experience between 7 and 36 months, mainly in jobs and internships. 77% of our sample has experience with professional software development, according to Table 1. The vast majority (82%) reported needing to gaining developing IoT embedded systems. In this case, the most reported experiences came from participating in university research projects and technical courses before entering the university.

Previous skills in CI/CD were also reported in the first form, and participants needed to inform whether they had any or no skills ranging on a Likert scale that went from no skills (0) to very advanced skills (4) [46]. Half the participants responded that they had no CI/CD skills.

4.2 | How does prior CI/CD knowledge impact the effort to configure, use, and understand a CI/CD pipeline in an IoT embedded project? - RQ_1

In this section, we present results related to how the impact of prior practical skills in continuous integration, delivery, and deployment make configuring, using, and understanding a CI/CD pipeline in an embedded IoT project less or more challenging for developers. In our quasi-experiment, the practical script contained the same tasks for groups of developers with and without previous practical skills in CI/CD. All developers had the opportunity to understand the concepts and fundamentals of CI/CD in the theoretical class. The practical class tasks are described in the script.

They are related to configuring a CI/CD pipeline by creating a .yml file with GH Actions components

[2]https://www.sbc.org.br/documentos-da-sbc/category/133-estatisticas

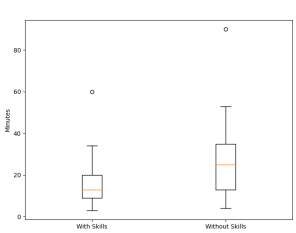
[3]https://meninas.sbc.org.br/

TABLE 1 Profiling information of the participants.

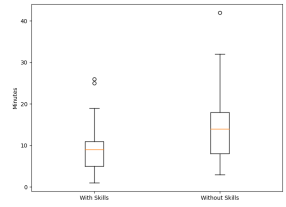
Category	Subcategory	#	%
Gender	Female	17	19
	Male	73	81
Degree	Computer Science	15	17
	Computer Engineering	18	20
	Information Systems	57	63
Experience in Software Engineering	None	20	22
	Up to 6 months	16	18
	Between 7 and 36 months	45	50
	Greater than 36 months	9	10
Experience in Embedded IoT	None	74	82
	Up to 6 months	10	11
	Between 7 and 36 months	5	6
	Greater than 36 months	1	1
Skills in Continuous Integration			
	None	45	50
	Little	23	25
	Moderate	12	14
	Advanced	8	9
	Very advanced	2	2
Skills in Continuous Delivery and Deployment			
	None	45	50
	Little	22	24
	Moderate	12	14
	Advanced	10	11
	Very advanced	1	1

and creating application and test source code files in the C language. The participants also had to handle the terminal emulator and use basic Git and GH functions, such as add, commit, push, checkout, clone, fork, and pull requests.

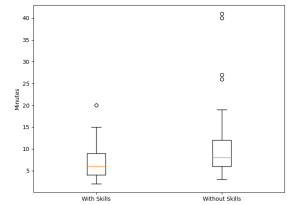
Figure 3 presents the boxplots of time spent by developers when performing the three tasks in the practical script. We used the Mann-Whitney statistical method to analyze the data in the three script tasks.



(a) Task 1: Set up GitHub Actions



(b) Task 2: Create a PR with a bug



(c) Task 3: Fix a PR

FIGURE 3 Time boxplots using Mann-Whitney in groups with and without prior CI/CD skills in the three tasks

Table 2 presents the median values and minimum and maximum time in minutes of the two groups in the three tasks. Since we used the Mann-Whitney method for the non-parametric data in our experiment, we calculated the p-value for each task. All p-values were less than 0.01. In other words, with a minimum of 99% confidence, there is a significant difference in the execution time of tasks in groups with and without previous CI/CD skills.

TABLE 2 Statistic Table - Mann-Whitney - Time in Minutes.

	With skills			Without skills			
	med	min	max	med	min	max	p-value
Task 1	13	3	60	25	4	90	0.00072
Task 2	9	1	26	12	3	42	0.00220
Task 3	6	2	20	8	3	41	0.00203

When analyzing the graphs in Figure 3 and the values in Table 2, we noticed that the median time spent performing the three tasks is more in the group without previous practical skills in CI/CD. The minimum times are shorter in the three tasks in the group that had some skill. The maximum times were the highest in the group of developers who reported having no prior practical skills in CI/CD.

We use categorized data to analyze developers' understanding of the code files used in pipeline configuration applications and testing. As stated in the literature, the Mann-Whitney method accepts these data types with the categories very easy (0); easy (1); moderate (2); hard (3); very hard (4) [25, 26]. Each developer then marked in the form the difficulty encountered when performing the tasks and their understanding of the code corresponding to the configuration, application, and test files. Table 3 shows the mode, minimum, and maximum values of the two groups in the three tasks of the practical script.

All p-values were less than 0.01. In other words, with a minimum of 99% confidence, there is a significant difference in the difficulty encountered by developers in executing tasks in groups with and without previous CI/CD skills. The mode values were the lowest found in the three tasks for the developers in the group who reported having some prior CI/CD skills. Both groups had developers who found it very easy to perform the tasks. In the group of developers with no skills, in the three tasks, the maximum that developers reported

TABLE 3 Statistic Table - Mann-Whitney - Degree.

	With skills			Without skills			
	mod	min	max	mod	min	max	p-value
Task 1	0	0	4	1	0	4	0.00218
Task 2	1	0	3	2	0	4	0.00022
Task 3	0	0	2	1	0	4	0.00251

about the difficulty encountered in executing each task was very hard. Table 4 shows the mode values minimum and maximum degree of the two groups in understanding the three task files of the practical script.

TABLE 4 Statistic Table - Mann-Whitney - Degree.

	With skills			Without skills			
	mod	min	max	mod	min	max	p-value
File 1	2	0	3	2	1	4	0.00000
File 2	1	0	3	2	0	4	0.02778
File 3	0	0	2	1	0	4	0.00338

All p-values were less than 0.05. In other words, with a minimum of 95% confidence, there was a significant difference in understanding the .yml and .c code files of the script tasks in the groups with and without previous CI/CD skills. Only the answers about file 2 that correspond to the .c code of the application, the p-value was less than 3%. The answer for the other two files, the p-value, was less than 1%. Both groups' modes were identical in files 1 and 2, but higher in the group without declared prior ability. The minimum value was the lowest for the group with some skill displayed in files 1 and 2 and the same in both groups for file 3. Regarding the maximum value, the highest values were from the group with no experience over the three files.

We noticed that even though sometimes the values were not discrepant, there was a significant difference to refute the null hypotheses (H_{0a}) Prior knowledge about continuous integration and delivery does not impact the configuration and use of the CI/CD pipeline with GHA in an embedded IoT project, and (H_{0b}) Prior knowledge about continuous integration and delivery does not impact understanding the files in a CI/CD pipeline with GHA in an embedded IoT project. Therefore, we accept the alternative hypotheses, Hα Prior knowledge about continuous integration and delivery impacts the configuration and use of the CI/CD pipeline with GHA in an embedded IoT project; and Hβ Prior knowledge about continuous integration and delivery does impact understanding the files in a CI/CD pipeline with GHA in an embedded IoT project.

We designed and built this quasi-experiment to provide all participating developers an experience configuring and using CI/CD in an embedded IoT project. Our pilot study helped us to make the theoretical class well explained and the practical guide intuitive for students in both groups. The tutoring approach during the execution of the script and the resolution of doubts in the two classes contributed to ninety participants completing the quasi-experiment regardless of the time spent and the difficulties reported when performing the tasks and understanding the files. This domain needs multifunctional and capable teams to meet the industry's and society's demands. Given this, such an approach may be interesting to present such technologies to new

professionals in the embedded IoT area. In addition to training students in IT courses to meet the software industry's and society's needs. And arouse the interest and understanding of these future professionals about using software engineering in IoT embedded systems.

Prior skills in continuous integration and delivery challenge setting up, using, and understanding a CI/CD pipeline with GHA in an IoT embedded system project. There was a significant statistical difference between the time developers with skills spent to complete the three tasks in the script and those without skills spent to complete these three tasks. The degree of difficulty reported by the groups regarding the tasks and their understanding of the files that were part of the script also showed a statistical difference between developers with and without continuous integration and delivery skills. Developers with previous skills in CI/CD spent less time completing the three tasks and found them generally very easy to perform and the files moderate to understand. Developers without previous skills in CI/CD, in addition to spending more time performing the tasks, found them easy and moderate to complete and found the files moderate to comprehend.

Therefore, prior knowledge of CI/CD can significantly impact the effort required to configure, use, and understand a CI/CD pipeline in an integrated Internet of Things (IoT) project. Assume that developers already have CI/CD experience. In this case, they will likely be familiar with configuration patterns that can speed up the initial process of configuring your IoT project's CI/CD pipeline, as they will already have a basic understanding of what is needed. They may also be more familiar with automated build, testing, integration, and deployment concepts, which can reduce the learning curve and errors when configuring and using the pipeline. They will also be familiar with monitoring and debugging tools and have a deeper understanding of potential pipeline failure points. However, although prior knowledge of CI/CD can be advantageous, it is still necessary to adapt and customize the CI/CD pipeline to meet the specific needs of the IoT project, and valuing collaboration between members in resolving queries can be a good strategy.

In our study, we observed that developers with prior CI/CD knowledge completed tasks significantly faster than those without such training. This aligns with findings from existing literature, which underscore the importance of hands-on training in DevOps education as a means of fostering practical skills that are directly applicable in the industry [47, 48]. Similar studies emphasize project-based and collaborative learning as effective approaches to DevOps education, suggesting that practical engagement accelerates learning and enhances task performance [49, 50]. Our results reinforce these conclusions, highlighting that practical experience with CI/CD tools helps bridge the gap between theoretical knowledge and real-world application, as noted in research on DevOps skill acquisition in educational settings [51].

To ensure control of Type I errors when conducting multiple statistical tests, we applied the Bonferroni correction, which is widely used in experiments involving multiple comparisons [52]. This approach adjusts the original significance level (α) by dividing it by the number of tests performed. In the context of this study, we conducted 9 statistical tests as we analyzed the results of 3 practical tasks applied to 3 distinct files from the CI/CD pipeline. Despite this, the alternative and null hypotheses are focused on two primary aspects: (H1) "prior CI/CD skills reduce task execution time," and (H2) "prior CI/CD skills reduce perceived difficulty and improve understanding of configuration files."

With the original significance level set at $\alpha = 0.01$, we adjusted this value using the Bonferroni correction to $\alpha' = 0.01/9 = 0.0011$. Recalculating the tests with this new threshold, we found that 3 out of the 9 tests had p-values lower than 0.0011, while the others did not meet the adjusted threshold. Consequently, we rejected the null hypothesis for the 3 significant tests and accepted it for the remaining 6, considering the differences significant only in the adjusted cases.

Applying the Bonferroni correction reduces the risk of false discoveries to below 0.01, thereby enhancing the

reliability of the results. However, we acknowledge that this approach also reduces statistical power, which may render the tests more conservative. Despite this limitation, we opted for this method due to its simplicity and widespread acceptance in empirical studies. Alternative methods, such as the Holm or Hochberg corrections, could be explored in future research to better balance statistical rigor and test power.

4.3 | What were developers' perceptions about the setting, use, and understanding of the CI/CD tasks? - RQ$_2$

In this section, we summarize the developers' perceptions shared during classes and their responses to open questions in the forms. The data extraction followed the five steps of thematic synthesis recommended for software engineering [27].

First, we gathered observations made by researchers during the classes, noting doubts and comments from the developers, along with answers to open questions. These notes were transferred to a qualitative data analysis tool[4] for coding. Data coding used an integrated (water) approach: an inductive approach for bottom-up code generation followed by a deductive approach using preliminary codes from the tool. This generated a general code scheme with broad domains.

The third step was translating codes into themes. Codes were combined to create comprehensive themes, with redundancies removed. The fourth step involved creating a high-level model, aligned with the research question (RQ$_2$). This resulted in the thematic map in Figure 4. The final step focused on evaluating the reliability of the synthesis, minimizing biases as discussed in Section 5.

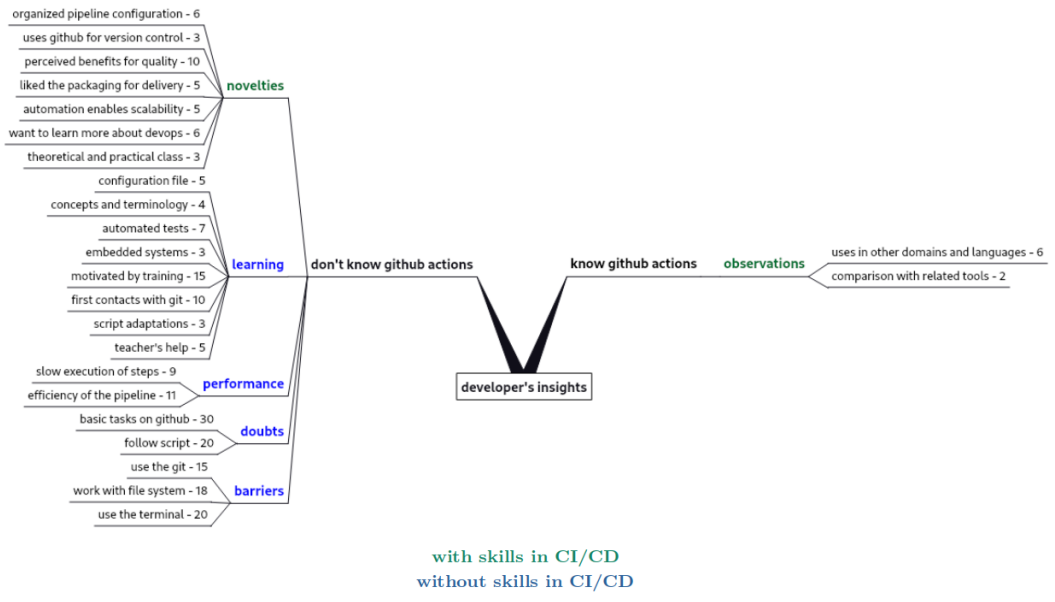


FIGURE 4 Thematic map

[4]https://atlasti.com/

The thematic map (Figure 4) includes two higher-order themes: developers familiar with GHA and those who are not, the latter being the majority. From these, six themes emerged: observations, novelties, learning, performance, doubts, and barriers. These themes were based on 24 codes, reflecting the descriptive and theoretical ideas observed. The themes and their relationships are shown in the thematic map.

Only codes identified more than once were included in the thematic map. The developers familiar with GHA generated the observation theme with two codes. Some of these developers used GHA in other types of projects and compared it to tools like Jenkins and GitLab CI/CD. For the majority who were unfamiliar with GHA, five themes emerged. The novelty theme had seven codes, and these developers were motivated to learn more about DevOps in IoT embedded systems. The learning theme highlighted areas of interest, such as understanding CI/CD pipeline configuration, automating tests, and working with Git. The performance theme presented mixed views on pipeline efficiency, with some appreciating it and others criticizing the execution speed. The three stages (construction, testing, and delivery) were seen as slow by some due to network connectivity and server latency. The doubts theme focused on participants' concerns about tasks and script interpretation, showing some mistrust and need for clarification. Barriers included using Git, working with GNU/Linux file systems, and using a terminal emulator, which required assistance from researchers.

Overall, developers had varied perceptions of CI/CD. Some were familiar with GHA, while most were beginners. Even those with prior CI/CD knowledge lacked experience with GHA or similar tools, indicating that theoretical knowledge did not always translate into practical skills. The hands-on experience led to new discoveries and increased motivation, especially for those who had not used version control tools before. Despite recognizing pipeline efficiency, many noted the slowness of execution. Challenges with basic tasks, script interpretation, and technical issues like Git and GNU/Linux highlighted the need for adapting training to better align with practices in FLOSS communities. Researchers acted as mentors, and their guidance helped participants appreciate the importance of these technologies, motivating them to seek further knowledge for future opportunities.

For developers with prior CI/CD experience, the setup and task completion were easier. However, even inexperienced developers could overcome initial challenges and benefit from exposure and gradual learning. This study contributes to DevOps education by offering insights into the measurable impact of hands-on CI/CD training. While prior research discusses challenges in integrating DevOps into curricula due to industry tool dynamics [53, 54], our findings show that practical training enhances task efficiency and better prepares students for industry demands. These results underscore the effectiveness of skills-based approaches in DevOps education and support the integration of CI/CD competencies into engineering programs, improving graduates' workforce readiness.

We used Spearman's correlation to analyze the relationships between the non-parametric variables in this study, including self-reported experience scales, task completion times, and perceived difficulty measures. The heatmap illustrating the strong positive correlations between the dependent and independent variables is presented in Figure 5. The algorithm used to compute Spearman's correlation, along with the results and raw student response data, can be found in our replication repository.

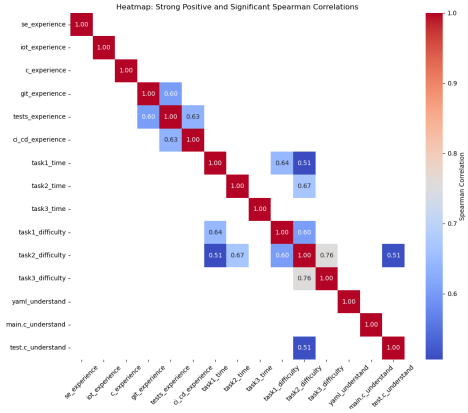


FIGURE 5 Spearman heatmap

The results revealed statistically significant associations between participants' prior experience, perceived difficulty, and task execution time. Strong correlations were found between software testing experience and Git experience ($\rho = 0.60$, $p < 0.001$), as well as between CI/CD experience and testing experience ($\rho = 0.63$, $p < 0.001$). These results suggest that participants with greater exposure to software testing tend to have more familiarity with Git and CI/CD, though this relationship may be influenced by confounding variables such as general experience in software projects or the structure of university curricula, where these topics are often taught together.

The strongest correlations involved perceived difficulty and task execution time, particularly in Task 2, whose difficulty was associated with the time spent on Task 1 ($\rho = 0.51$, $p < 0.001$) and its own execution time ($\rho = 0.67$, $p < 0.001$). Additionally, Tasks 2 and 3 showed a high correlation in perceived difficulty ($\rho = 0.76$, $p < 0.001$), indicating progressive and interconnected challenges. Understanding the `test.c` file was also correlated with the difficulty of Task 2 ($\rho = 0.51$, $p < 0.001$), suggesting that familiarity with the C language may have played a role. These findings highlight potential confounding factors, such as task complexity, clarity of instructions, and prior familiarity with tools like YAML and GitHub Actions, which may have impacted both execution time and perceived difficulty.

5 | THREATS TO VALIDITY

Although we have carefully planned, this research may be affected by several factors that could threaten our findings. We discuss these factors and decisions to mitigate their impact in our study, which is divided into categories of threats to validity [25].

Construct Validity. This validity concerns the extent to which the study measurements reflect real-world situations. In this research, we selected students studying Software Engineering, a subject taken after the basic cycle of their courses, as participants. This choice may raise concerns about the realism of the results and their adaptability to the software industry. However, some studies suggest that, with a carefully designed experiment and a new approach to the subjects, similar results can be obtained from both students and professionals

[30, 28]. To minimize potential threats, we thoroughly discussed all experimental procedures and conducted a pilot study to improve the forms used. Qualitative studies are also considered essential in software engineering experimentation [55, 56]

Internal Validity. Threats to internal validity may be compromised by the presence of uncontrolled confounding variables, such as prior familiarity with the GNU/Linux terminalwhich could have influenced efficiency in using Git and CI/CD commandsand experience with the GitHub interface, which might have reduced the perceived difficulty regardless of technical skills. To mitigate these threats, future studies could incorporate more objective measurements of participants' experience and control for additional variables that may influence the outcomes.

External Validity. External validity refers to the ability to generalize findings to other settings. One potential threat to external validity is the use of statistical tools. When reporting our results, we ensured the correct application of statistical tests, such as the Mann-Whitney test, to minimize the likelihood that our results were influenced by random events. Another threat could be the reliability of our thematic synthesis. To address this, we aimed for consistency in defining themes and their correlation with identified codes, providing a detailed explanation of the steps followed in Section 4.3 and making the raw data available in our repository for replication.

Conclusion Validity. Conclusion validity concerns the ability to draw correct conclusions. To mitigate this threat, we present descriptive statistics, statistical tests, and a thematic synthesis with raw data for replication. All researchers participated in the analysis to reduce bias. A potential threat is the Bonferroni correction, which reduces false discoveries but may increase Type II errors. We recommend considering alternative methods, like Holm or Hochberg, for future work.

6 | CONCLUSION AND FUTURE WORK

This study explored the challenges developers face when adopting Continuous Integration and Delivery (CI/CD) pipelines in embedded IoT projects. Only 8% of developers failed to complete all tasks in the practical script. Developers with no prior CI/CD knowledge spent more time on tasks and found them more difficult than those with experience. Interestingly, regardless of their reported CI/CD skills, all developers needed to familiarize themselves with the necessary tools. In other words, hands-on practice was essential. Many participants reported having never used version control tools or automated tests before. The combination of theoretical, practical, and mentoring classes provided the developers with opportunities to understand key technologies and resolve doubts, resulting in a 92% success rate in the quasi-experiment.

It is important to note that some developers faced greater challenges than others, and peer collaboration played a key role. The themes of doubts and barriers were most frequently coded, reflecting the researchers' role as tutors and problem solvers during the practical tasks. The theoretical class helped clarify the domain, concepts, and terminology associated with the technologies. However, some participants struggled with interpreting the practical script and showed significant resistance to using a terminal emulator to interact with GNU/Linux and Git.

We observed that most developers use proprietary operating systems and software with graphical interfaces, which abstract away the need for file, directory, and program manipulation via the command line. This led us to question how such tools align with the growing preference for Free/Libre and Open Source Software (FLOSS) in the IoT and edge computing development market. Numerous sources indicate a strong trend toward

using open-source software in these areas, including academic research, developer surveys, and market reports. Despite the widespread awareness of GitHub (GH), most developers in our sample had limited knowledge of it, mainly using it for code research or version control, and rarely interacting with FLOSS project communities. The quasi-experiment exposed many developers to GitHub Actions (GHA) for the first time.

Overall, developers expressed appreciation for the opportunity to expand their knowledge of CI/CD practices and IoT embedded systems, recognizing their relevance to their careers and the expanding IoT market. Many reported increased motivation to further study CI/CD pipelines and IoT embedded systems. These challenges and insights provide valuable opportunities for future learning, including the development of targeted training sessions for novice professionals or students, as well as focus groups and interviews to gain qualitative insights into their experiences. Most participants demonstrated a need to acquire practical experience in IoT embedded systems, as they had little or no prior expertise in the domain. There is also a significant opportunity to form study groups involving undergraduate and graduate students, researchers, and others interested in contributing to IoT embedded FLOSS projects. We can even conduct training sessions with novice professionals or students and develop focus groups and interviews to qualitatively understand their perceptions and acquired experiences.

ACKNOWLEDGEMENTS

This research was partially supported by the Brazilian funding agencies CAPES, CNPq, FAPEMIG, and the universities where the authors are assigned.

REFERENCES

[1] Lee I, Lee K. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Business Horizons 2015;58(4):431–440.

[2] Schubert L, Tsitsipas A, Jeffery K. Establishing a basis for new software engineering principles. Internet of Things 2018;3-4:187–195.

[3] Rayes A, Samer S. Internet of Things From Hype to Reality: The Road to Digitization. 1st ed. Springer Publishing Company, Incorporated; 2016.

[4] Reggio G, Leotta M, Cerioli M, Spalazzese R, Alkhabbas F. What are IoT systems for real? An experts' survey on software engineering aspects. Internet of Things 2020;12:100313.

[5] Ronkainen J, Abrahamsson P. Software Development under Stringent Hardware Constraints: Do Agile Methods Have a Chance? In: Marchesi M, Succi G, editors. Extreme Programming and Agile Processes in Software Engineering (XP); 2003. p. 73–79.

[6] Manhart P, Schneider K. Breaking the Ice for Agile Development of Embedded Software: An Industry Experience Report. In: 26th International Conference on Software Engineering (ICSE); 2004. p. 378–386.

[7] Gousios G, Pinzger M, van Deursen A. An exploratory study of the pull-based software development model. In: Proceedings of the 36th International Conference on Software Engineering; 2014. p. 345 – 355.

[8] Peterson K. The github open source development process; 2013. `https://api.semanticscholar.org/CorpusID:19529670`.

[9] Trinkenreich B, Guizani M, Wiese I, Conte T, Gerosa M, Sarma A, et al. Pots of Gold at the End of the Rainbow: What is Success for Open Source Contributors? IEEE Transactions on Software Engineering 2022;48(10):3940–3953.

[10] Steinmacher I, Conte T, Gerosa MA, Redmiles D. Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects. In: Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW); 2015. p. 13791392.

[11] Wessel M, Vargovich J, Treude MAGC. GitHub Actions: The Impact on the Pull Request Process. Empirical Software Engineering 2023;.

[12] Zhang X, Yu Y, Gousios G, Rastogi A. Pull Request Decisions Explained: An Empirical Overview. IEEE Transactions on Software Engineering 2023;49(2):849–871.

[13] Leite L, Rocha C, Kon F, Milojicic D, Meirelles P. A Survey of DevOps Concepts and Challenges. ACM Computing Surveys 2019;52(6).

[14] Silva CC, Goldman A. Agile Methods Adoption on Software Development: A Pilot Review. In: Lacey M, editor. Agile Conference (AGILE) IEEE Computer Society; 2014. p. 64–65.

[15] Diebold P, Theobald S. How is Agile Development Currently Being Used in Regulated Embedded Domains? Journal of Software: Evolution and Process 2018;30(8).

[16] Berg V, Birkeland J, Nguyen-Duc A, Pappas IO, Jaccheri L. Achieving agility and quality in product development - an empirical study of hardware startups. Journal of Systems and Software 2020;167:110599.

[17] Pradhan S, Nanniyur V. Large scale quality transformation in hybrid development organizations A case study. Journal of Systems and Software 2021;171:110836.

[18] Taivalsaari A, Mikkonen T. A Roadmap to the Programmable World: Software Challenges in the IoT Era. IEEE Software 2017;34(1):72–80.

[19] Dobrilovic D, Zeljko S. Design of open-source platform for introducing Internet of Things in university curricula. In: 2016 IEEE 11th International Symposium on Applied Computational Intelligence and Informatics (SACI); 2016. p. 273–276.

[20] Corno F, Russis LD, Sáenz JP. On the challenges novice programmers experience in developing IoT systems: A Survey. Journal of Systems and Software 2019;157:110389.

[21] Dias JP, Restivo A, Ferreira HS. Designing and constructing internet-of-Things systems: An overview of the ecosystem. Internet of Things 2022;19.

[22] Aly M, Khomh F, Gueheneuc YG, Washizaki H, Yacout S. Is Fragmentation a Threat to the Success of the Internet of Things? IEEE Internet of Things Journal 2019;6(1):472–487.

[23] Fahmideh M, Ahmad A, Behnaz A, Grundy J, Susilo W. Software Engineering for Internet of Things: The Practitioners' Perspective. IEEE Transactions on Software Engineering 2022;48(8):2857–2878.

[24] Kampenes VB, Dyba T, Hannay JE, Sjoberg DIK. A systematic review of quasi-experiments in software engineering. Information and Software Technology 2009;51:71–82.

[25] Wohlin C, Runeson P, Hst M, Ohlsson MC, Regnell B, Wessln A. Experimentation in Software Engineering. Springer Publishing Company, Incorporated; 2012.

[26] Siegel S, Jr NJC. Nonparametric Statistics for the Behavioral Sciences. Mcgraw-Hill Book Company; 1988.

[27] Cruzes DS, Dyba T. Recommended Steps for Thematic Synthesis in Software Engineering. In: International Symposium on Empirical Software Engineering and Measurement(ESEM) IEEE Computer Society; 2011. p. 275–284.

[28] Falessi D, Juristo N, Wohlin C, Turhan B, Münch J, Jedlitschka A, et al. Empirical software engineering experts on the use of students and professionals in experiments. Empirical Software Engineering 2018;23:452–489.

[29] Feldt R, Zimmermann T, Falessi GRBD, Jedlitschka A, Juristo N, Munch J, et al. Four Commentaries on the Use of Students and Professionals in Empirical Software Engineering Experiments. Empirical Software Engginering 2018;23:38013820.

[30] Salman I, Misirli AT, Juristo N. Are Students Representatives of Professionals in Software Engineering Experiments? In: IEEE/ACM International Conference on Software Engineering (ICSE); 2015. p. 666–676.

[31] Golzadeh M, Decan A, Mens T. On the rise and fall of CI services in GitHub. In: IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER); 2022. p. 662–672.

[32] Moedt W, Bernsteiner R, Hall M, Fruhling A. Enhancing IoT Project Success through Agile Best Practices. ACM Transactions on Internet of Things 2023;4(1).

[33] Zampetti F, Tamburri D, Panichella S, Panichella A, Canfora G, Penta MD. Continuous Integration and Delivery Practices for Cyber-Physical Systems: An Interview-Based Study. ACM Transactions on Software Engineering and Methodology 2023;32(3):1–44.

[34] Cristea R, Paduraru C. An experiment to build an open source application for the Internet of Things as part of a software engineering course. In: IEEE/ACM International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT); 2023. p. 13–16.

[35] López-Viana R, Díaz J, Díaz VH, Martínez JF. Continuous Delivery of Customized SaaS Edge Applications in Highly Distributed IoT Systems. IEEE Internet of Things Journal 2020;7(10):10189–10199.

[36] Zampetti F, Nardone V, Penta MD. Problems and Solutions in Applying Continuous Integration and Delivery to 20 Open-Source Cyber-Physical Systems. In: IEEE/ACM International Conference on Mining Software Repositories (MSR) IEEE; 2022. p. 646–657.

[37] López-Viana R, Díaz J, Pérez JE. Continuous Deployment in IoT Edge Computing: A GitOps implementation. In: 17th Iberian Conference on Information Systems and Technologies (CISTI) IEEE; 2022. p. 1–6.

[38] Talekar M, Harpale VK. CI-CD Workflow For Embedded System Design. In: 7th International Conference On Computing, Communication, Control And Automation; 2023. p. 1–3.

[39] Basili VR, Weiss DM. A Methodology for Collecting Valid Software Engineering Data. IEEE Transactions on Software Engineering 1984;SE-10(6):728–738.

[40] Ribeiro D, Lima R, Franca C, Souza A, Silva I, Pinto G. Understanding Self-Efficacy in Software Engineering Industry: An Interview Study. In: International Conference on Evaluation and Assessment in Software Engineering (EASE); 2023. p. 101 – 110.

[41] Zhang D, Chan CC, Zhou GY. Enabling Industrial Internet of Things (IIoT) towards an emerging smart energy system. Global Energy Interconnection 2018;1(1):39–47.

[42] Geihs K, Baraki H, de la Oliva A. Performance Analysis of Edge-Fog-Cloud Architectures in the Internet of Things. In: IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC); 2020. p. 374–379.

[43] Kumar R, Agrawal N. Analysis of multi-dimensional Industrial IoT (IIoT) data in EdgeFogCloud based architectural frameworks: A survey on current state and research challenges. Journal of Industrial Information Integration 2023;35:100504.

[44] Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge Computing: Vision and Challenges. IEEE Internet of Things Journal 2016;3(5):637–646.

[45] Guzman I, Berardi R, Maciel C, Tapia PC, Marin-Raventos G, Rodriguez N, et al. Gender Gap in IT in Latin America. Americas Conference on Information Systems (AMCIS); 2020. .

[46] Likert R. A Technique for the Measurement of Attitudes. Archives of Psychology; 1932.

[47] Paez N, Fontela C. Software Engineering Education in the DevOps Era: Experiences and Recommendations. In: Anais do XXVI Congresso Ibero-Americano em Engenharia de Software Porto Alegre, RS, Brasil: SBC; 2023. p. 130–137.

[48] Ferino S, Fernandes M, Cirilo E, Agnez L, Batista B, Kulesza U, et al. Overcoming Challenges in DevOps Education through Teaching Method. In: IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET); 2023. p. 166–178.

[49] Pang C, Hindle A, Barbosa D. Understanding DevOps Education with Grounded Theory. In: IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET); 2020. p. 107–118.

[50] Alves I, Rocha C. Qualifying Software Engineers Undergraduates in DevOps - Challenges of Introducing Technical and Non-technical Concepts in a Project-oriented Course. In: IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET); 2021. p. 144–153.

[51] Perez JE, Gonzalez-Prieto A, Diaz J, Lopez-Fernandez D, Garcia-Martin J, Yague A. DevOps Research-Based Teaching Using Qualitative Research and Inter-Coder Agreement. IEEE Transactions on Software Engineering 2022;48(9):3378–3393.

[52] Hernan MA, Robins JM. Casual Inference: What if. 1st ed. Boca Raton:Chapman and Hall/CRC; 2020.

[53] Hobeck R, Weber I, Bass L, Yasar H. Teaching DevOps: a tale of two universities. In: ACM SIGPLAN International Symposium on SPLASH-E SPLASH-E 2021, New York, NY, USA: Association for Computing Machinery; 2021. p. 2631.

[54] Kuusinen K, Albertsen S. Industry-Academy Collaboration in Teaching DevOps and Continuous Delivery to Software Engineering Students: Towards Improved Industrial Relevance in Higher Education. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET); 2019. p. 23–27.

[55] Basili VR, Shull F, Lanubile F. Building knowledge through families of experiments. IEEE Transactions on Software Engineering 1999;25(4):456–473.

[56] Kitchenham BA, Pfleeger SL, Pickard LM, Jones PW, Hoaglin DC, Emam KE, et al. Preliminary guidelines for empirical research in software engineering. IEEE Transactions on Software Engineering 2002;28(8):721–734.

[57] Chettri DDSK. Internet of Things: Current Research, Challenges, Trends and Applications. In: Kumar XZGR, Srivastava S, Soni BP, editors. Applications of Artificial Intelligence in Engineering Springer Singapore; 2021. p. 679–694.

[58] Martikkala A, David J, Lobov A, Lanz M, Ituarte IF. Trends for Low-Cost and Open-Source IoT Solutions Development for Industry 4.0. Procedia Manufacturing 2021;55:298–305.