A Comprehensive Exploration of the Use of Software Testing Tools for IoT Systems

Caio Guimarães Herrera[1], **Vinicius Gomes Ferreira**[1], **Simone R S Souza**[1], **Ricardo Santos**[2], **Paulo Sergio Lopes de Souza**[1]

[1] Universidade de São Paulo (USP)
LASDPC/SSC/ICMC – São Carlos – SP – Brazil

[2]Universidade Federal do Mato Grosso do Sul (UFMS)
Campo Grande – MS – Brazil

Abstract. *Numerous tools are proposed to support the execution of software testing in IoT systems; however, their availability remains limited. There is a notable absence of suitable testing tools to facilitate a robust and comprehensive approach to structuring the validation of these systems. This paper assesses the tools available in the scientific literature for supporting software testing in IoT systems, examining their potential applicability in scenarios different from those originally proposed. We have analyzed tools identified in a previous systematic mapping and sought artifacts and resources to guide developers and testers in their usage. The primary outcome of this paper is to present evidence regarding the overall quality and limitations of tools designed to support software testing in IoT systems. This aims to provide programmers and testers working with IoT applications with a more nuanced understanding of the field and available tools.*

Resumo. *São diversas as ferramentas propostas para a execução de testes de software em sistemas de IoT. No entanto, essas ferramentas são frequentemente pouco acessíveis para uso prático. Existe uma carência de ferramentas que ofereçam suporte robusto para uma estruturação abrangente de testes de software nesses sistemas. Este estudo avalia as ferramentas disponíveis para suporte a testes de software em sistemas de IoT na literatura científica, investigando a viabilidade de utilizá-las em contextos diferentes dos originalmente propostos. Isso é realizado por meio da exploração de ferramentas identificadas em um mapeamento sistemático prévio, além da busca por artefatos e recursos que possam orientar desenvolvedores e testadores em sua utilização. O principal resultado deste artigo consiste na apresentação de evidências sobre as qualidades e limitações das ferramentas de suporte à atividade de teste. Isso proporciona aos programadores e testadores que lidam com aplicações IoT um entendimento mais aprofundado da área e das ferramentas disponíveis.*

1. Introduction

The Internet of Things (IoT) concept describes the set of integrated components, devices, and systems connected through a network capable of generating and processing data in a distributed manner. The adoption of IoT systems is increasing, and the number of IoT devices is expected to reach 25.4 billion by 2030, leading to a global economic value of US$ 3.9 trillion [Gartner 2020].

IoT systems gather unique characteristics, such as intrinsic heterogeneity, due to the variety of sensors, actuators, and other components, as well as different network connections and communication protocols widely used for multiple purposes. Those characteristics often generate additional difficulties in testing IoT applications [Bures et al. 2018].

Software testing is still essential in IoT systems to ensure quality and reliability. Such difficulties motivate developers to adopt different approaches and tools for running the testing activity compared to more traditional systems, such as simulators and emulators [Bures et al. 2018].

Although many scientific proposals are revisiting and discussing testing tools for IoT systems [Ahmed et al. 2019, Bures et al. 2020], it is necessary still to investigate which tools are available for usage and replication in contexts that differ from those initially presented in such proposals. To do so, a developer or tester working with IoT applications must access artifacts that allow them to download, install, and set the desired tools in their environment.

This paper presents a practical overview of software testing tools applied to IoT systems, aiming to understand their availability for developers and testers. Using an example of an IoT system, we assess the contributions of the proposed tools. In previous work, we identified a set of testing tools proposed in the scientific literature for IoT systems [Ferreira et al. 2023]. However, as far as we can see, these works do not present evidence of the practical use of these testing tools, which makes their use difficult.

This paper is helpful for programmers and testers working with IoT applications once they find guidelines to start using the available tools for software testing according to their IoT application context.

The remainder of this paper is organized as follows: Section 2 presents some background concepts and challenges related to Software Testing in IoT systems. Section 3 lists other works related to this paper found in the literature. Section 4 describes the methods used for the selected tools' literature review and evaluation. Section 5 presents the results for the tool selection and the search for artifacts and usage of the available resources while discussing the main findings of those results. Section 6 provides the identified threats to the validity of this study, and Section 7 concludes this paper, suggesting directions for future work.

2. Background

2.1. Software Testing

Software testing, a process that integrates a broader set of software quality activities, is the most common approach to quality assessment in the software engineering industry [Ammann and Offutt 2016]. It is a dynamic approach, dependent on software execution, to search for defects that may have been inserted during coding activities or are collateral results of poor software modeling or requirements engineering.

Conventionally, software testing depends on inputs feeding into a running system, leading the control flow to reach a point in the system where a defect is contained and causing a change in the system state that must be propagated to an observable point

[Ammann and Offutt 2016], [Delamaro et al. 2013]. Regardless of the software under test (SUT), these assumptions must be met for the test to be said to be successful.

2.2. IoT systems

The Internet of Things (IoT) paradigm defines the interconnection of physical devices - sensors, actuators, smartphones, wearable devices, and vehicles - with the Internet and other networks, making it possible to obtain, process, and transmit data between them. This technology has become a promising and innovative application for many domains, such as smart cities, health, agriculture, industry, and education. According to a Gartner report from 2020, the expected number of IoT devices should reach 25.4 billion by 2030, with an estimated economic value of US$ 3.9 trillion [Gartner 2020].

IoT systems present unique characteristics and challenges compared to other digital systems (e.g., web applications). One of the main differences concerns the system architecture. Usually, a 3-layer model is adopted to represent an IoT system, composed of a **perception layer**, a **network layer**, and an **application layer** [Wu et al. 2010].

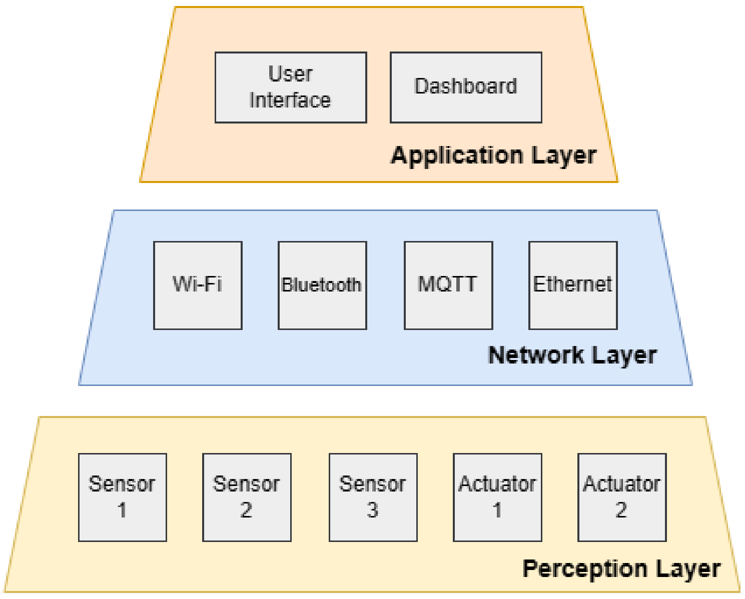


Figure 1. The three-layer model for IoT systems.

The **perception layer** comprises sensors responsible for collecting data from the real world. In this layer, there can also be a set of actuators that must interact with the environment to execute an action or trigger an event. In the **network layer**, the collected data in the previous layer transits and is sent to the **application layer**, which contains the interfaces through which the users will be exposed to the data and operate the system [Wu et al. 2010].

2.3. Software Testing in the IoT context

IoT systems present a set of characteristics that need to be observed and dealt with to adequately address the software testing activities [Saraiva et al. 2019]. Due to the composition of multiple device types, there is usually a degree of heterogeneity between its components regarding hardware, software, and communication protocols. This highlights the need for extensive integration testing [Kuroiwa et al. 2019].

IoT devices have fewer resources available than a conventional system in processing, memory, power, and storage [Bures et al. 2018] [Murad et al. 2018], which might limit the amount of resources used by and dedicated to testing strategies. This resource limitation, associated with the heterogeneity of devices, usually implicates a stronger coupling between hardware and software, making testing more complex and costly [Bures et al. 2018] [Bures et al. 2020].

Since a big part of an IoT system relies on its interaction with the environment in which it is located, a quality assurance process has to account for the influence of real-world inputs coming from sensors, which might not be predictable. This may cause a certain degree of uncertainty in the output as well [Cristea et al. 2022]. Despite that, environmental characteristics may affect the system's behavior as a whole, as it can be located (or moving) into a region with low connectivity issues, for example, [Bures et al. 2018].

These characteristics bring special conditions for the quality assurance of these systems compared to other types of applications. To address them, many approaches and custom tools are usually explored to make the testing activity less complex and, simultaneously, less costly.

2.4. Supporting testing tools for IoT systems

Testing tools have been developed to address the mentioned challenges related to test of IoT systems [Bures et al. 2020, Saraiva et al. 2019], including specific techniques and approaches, testbeds, test case generation and treatment tools, and simulation or emulation tools. In particular, the latter aims to address the trade-off between creating a realistic operating environment for IoT systems and the cost of providing an expensive and complex infrastructure for testing purposes [Patel et al. 2019]. In the literature, one can find simulators whose goal is to allow the execution of compliance tests [Demirel et al. 2019], interoperability tests [Estebsari et al. 2018, Haris et al. 2019], connectivity tests [Klima et al. 2023], among others.

Since IoT systems can often be composed of many devices, their simulators must be able to handle an equally large number of simulated entities. For example, a simulator designed to evaluate the interoperability of IoT devices must be able to handle behavioral anomalies caused by the connection between heterogeneous devices. However, it cannot affect the tests because of packet transfer degradation caused by the simulation of many devices.

3. Related Work

There are multiple cases in the literature of systematic mappings and reviews that consolidate the available tools that support Software Testing on IoT Systems. [Murad et al. 2018] present a list of tools not constrained to simulators, as well as a meaningful discussion about the challenges of the subject. However, the authors do not use nor test the mentioned tools. They also explore the heterogeneity of IoT systems as part of the challenges without investigating the mentioned tools.

[Chernyshev et al. 2018] also provide a list of simulation tools available for software testing IoT systems with objective details regarding their application, use cases, features, practicality, and other criteria. Despite having a suitable exploration, the author does not mention the ability to install and use each tool, whether artifacts exist, and whether documentation is available.

On the other hand, [Minani et al. 2024] provide a multimethod study of the way IoT systems are tested in the industry, evaluating the most common levels and goals of the software testing activity while also investigating the most used tools and approaches. While this provides a complementary view regarding the scientific literature, it does not explore the feasibility and usage of each tool.

[Ferreira et al. 2023] present a systematic mapping of the software testing for IoT systems and organize the found tools and approaches in a development life cycle while also investigating the techniques and testing levels covered by each solution. However, it also does not explore the usage of such tools.

Unlike the related work described in this section, this paper presents a practical look at the existing tools for software testing in the scientific literature related to IoT systems. We aim to fill this gap by providing developers and testers with information on using the tools in their test environments. This paper directly evolves from the work described in [Ferreira et al. 2023] by further evaluating if the identified tools are still applicable and can be replicated and used in a new proposed IoT application.

4. Methods

To guide this study, a comprehensive literature review was previously conducted to identify the most promising tools and approaches for supporting software testing activities in IoT systems [Ferreira et al. 2023]. The mapping was designed to gain insights into the distribution of testing solutions throughout a typical development life cycle process. To accomplish this, a set of research questions was defined as follows:

- **RQ1** - At which moment of the IoT development life cycle is the testing applied?
- **RQ2** - What IoT testing literature covers test techniques and levels?
- **RQ3** - What problems motivated the development of the tool or approach?
- **RQ4** - What contributions are reported in studies that propose such a tool or approach?

This literature review was conducted on IEEE[1], SpringerLink[2], ACM[3], and Scopus[4] databases, in which articles were selected and evaluated by two independent researchers. The chosen approaches were classified according to the categories presented in Table 1.

A summary of the entire mapping process can be seen in Figure 2. The detailed method used for the mapping and its results can be seen in [Ferreira et al. 2023]. In summary, 42 articles were initially retrieved by searching the mentioned databases, composed of primary and secondary studies. From the latter, two papers were chosen to be the target of a backward and forward Snowballing process [Wohlin 2014], in which 30 new relevant papers were retrieved, forming a study corpus of 72 papers.

[1]http://ieeexplore.ieee.org/
[2]https://www.springer.com/
[3]https://dl.acm.org/
[4]https://www.scopus.com/

Table 1. Categories considered in this study.

Category	Description
Test Level / Technique	This category recovers the test level or technique of the traditional software testing addressed in the paper.
Approach/Tool	Classification of the article's contribution as an approach or tool.
Testing stage	Classification of the paper's contribution according to the IoT testing life cycle discussed in this study.

As a secondary contribution of the study and a direct result from the category **Approach/Tool**, mentioned in Table 1, we built a list of tools that could be investigated for their usage in an actual application.

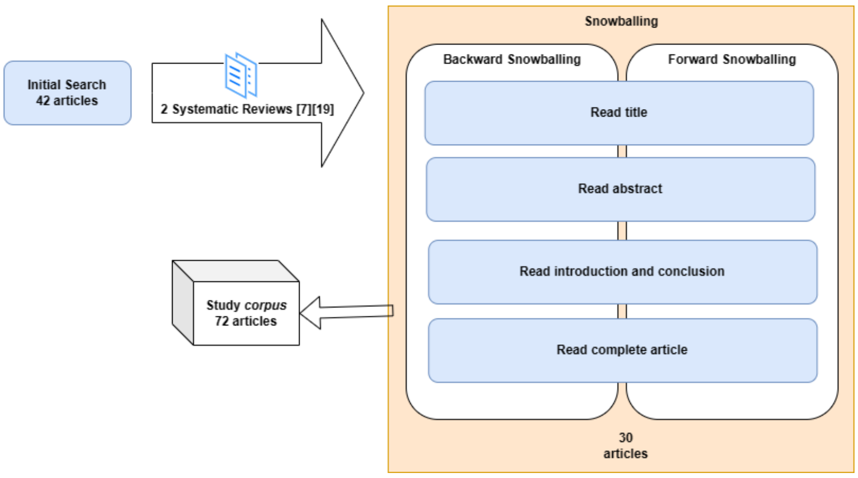


Figure 2. The literature review process.

From the tools list, a set of criteria was defined, translated in the following questions, and used to filter out the available tools that could be explored in this study:

- C1: Is the tool identifiable through the article's content?
- C2: Is the tool anything other than a Testbed?
- C3: Is the tool available for stand-alone free use (from an installer or source code)?
- C4: Do the tool's artifacts guide on installing and using it?

In this context, C1 evaluates which of the selected articles provide any identification for the proposed tool. This can vary from a simple name to a website or code repository but indicates whether searching for the artifacts in common search mechanisms is feasible. Also, despite the mapping results providing a category distinction between approaches and tools, all the selected articles were considered when searching for identifiable tools.

For this study, we will focus solely on software tools and products that do not rely on physical hardware but rather simulated ones. This is an adopted strategy in IoT systems testing to reduce the cost and impact of hardware defects in the activity [Patel et al. 2019] [Bosmans et al. 2019]. Due to that constraint, C2 excludes Testbed proposals.

C3 and C4 evaluate how practical the found artifacts are for the tool to be used. C3 looks for ways to download and set up the tool in a personal environment. This could be done through a Downloads page on a website, a direct link to an executable file, a code repository that can be cloned, etc. C4 grants a straightforward process for setting up the downloaded tool, with well-defined prerequisites and installation instructions.

To answer C3, we conducted an ad hoc search for code repositories, websites, downloadable applications, or any other artifact, allowing the use of the selected tools. The steps included in guidance, tutorials, and README files were followed for the found artifacts. The response to this criterion may vary from one search to another, considering that they might be conducted in different search engines, with other queries, and at various times.

The final list of tools was also characterized through the following set of Yes/No questions:

- Q1 - Did the tool receive any updates in the last 12 months?
- Q2 - Is the tool available through a package or executable file?
- Q3 - Was the tool successfully installed and used?

Questions Q1 and Q2 were not used to filter the tools but rather to understand their maturity and maintainability. Q3 is a summary of the results of this study in terms of which tools were properly instantiated.

For cases where the instructions were unclear or an error was encountered, the article's authors were contacted to request guidance or pointers on how to solve the issue. We have used a Dell Vostro 3520 laptop with Windows 11 x64 installed, an Intel i7-1255U 1.70GHz processor, and 16GB of RAM to work with the tools. Since most of the resources targeted a Linux environment, WSL2 was used to install an Ubuntu 22.04.3 LTS subsystem within Windows 11.

For the tools we could instantiate and use, we explored the capabilities of representing and testing an example of SUT, inspired by an IoT Architecture for Cattle Supplementation, portrayed by [Defalque et al. 2023]. This SUT intends to convey a Programmable Automatic Feeder (PAF), described by a **feeder module** - which supervises the food dispenser and the amount of portion left -, connected to a cattle monitoring system (**cattle module**), which is responsible for tracking the movements of each bovine. There is also an **analytics module**, used by the system users to track what is happening. An MQTT server is responsible for ensuring communication between modules. The SUT can be represented using the three-layer model [Wu et al. 2010], according to Figure 3, where both cattle and feeder modules are associated with the **perception layer**, the analytics module describes the **application layer**, and the MQTT protocol fills the role of the **network layer**.

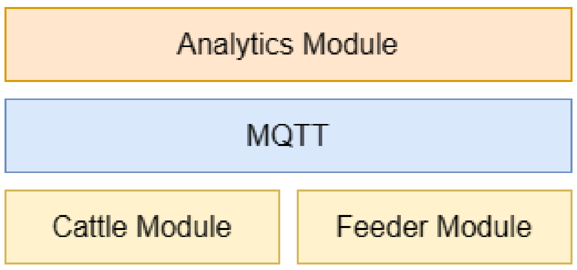


Figure 3. Three layer model of the SUT example.

5. Results

The defined criteria were applied after the selected articles in the literature review, and the selection steps can be seen in Figure 4. From the original 72 articles, 44 (approx. 61%) present an identifiable tool. The reduction of almost 40% of the articles in this criterion is a direct result of most of the articles (54%) presenting approaches for testing [Ferreira et al. 2023], which were not necessarily associated with a specific tool. This reduction can also reflect on tools used but not explicitly mentioned, identified, or implemented.

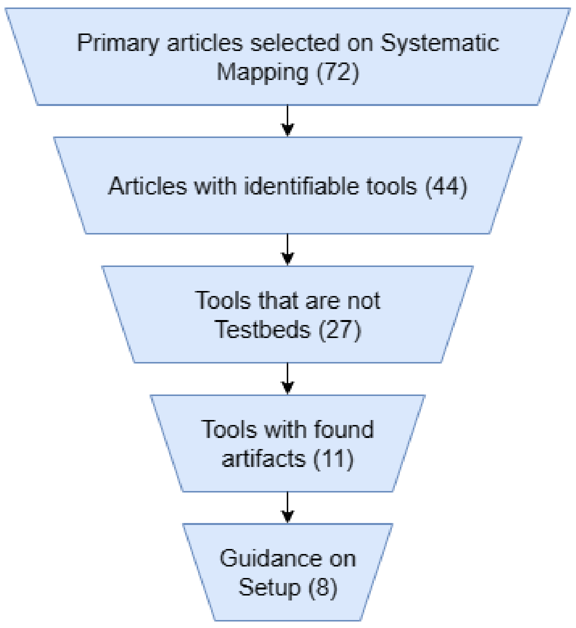

Primary articles selected on Systematic Mapping (72)
Articles with identifiable tools (44)
Tools that are not Testbeds (27)
Tools with found artifacts (11)
Guidance on Setup (8)

Figure 4. Criteria-based selection of the articles.

From the 44 tools, 17 were related to Testbed proposals. This also aligns with the results of [Ferreira et al. 2023], since 24% of the articles relied on Testbeds to execute software testing. However, since this study intends to focus on software solutions, these tools were removed from our analysis.

From the remaining subset of 27 tools, artifacts were found for only 11 of them, of which 8 (18% of the original 44 identifiable tools) provided clear instructions for complete installation and use. The final list of the selected tools is described below and can be seen in Table 2.

Table 2. Selected tools and artifacts.

Tool	Artifact URL	Q1	Q2	Q3
Node-Red [Clerissi et al. 2018]	https://nodered.org/	X	X	X
Selenium [Varghese and Sinha 2020]	https://www.selenium.dev/	X	X	X
PatrIoT [Bures et al. 2021]	https://patriot-framework.io/	X	X	
MobIoTSim [Pflanzner et al. 2016]	https://github.com/sed-szeged/MobIoTSim	X		
DSL IoTECS [Li et al. 2022]	https://github.com/sednalab/IoTECS	X		
IoT-Testware [Schieferdecker et al. 2017]	https://projects.eclipse.org/projects/technology .iottestware	X		
IoTSim-Edge [Jha et al. 2020]	https://github.com/DNJha/IoTSim-Edge			
DPWSim [Han et al. 2014]	https://github.com/sonhan/dpwsim			

- **Node-Red** [Clerissi et al. 2018] [Varghese and Sinha 2020]: A low-code programming tool for event-driven applications.
- **Selenium** [Varghese and Sinha 2020]: A well-known web-based testing automation tool commonly used for system and interface testing, as well as robot process automation.
- **PatrIoT** [Bures et al. 2021]: An automated testing framework that allows for the integration of simulated and physical devices.
- **MobIoTSim** [Pflanzner et al. 2016]: An IoT simulator that relies on mobile devices to simulate IoT sensors and devices.
- **DSL IoTECS** [Li et al. 2022]: A domain-specific language that allows for building simulators to represent large numbers of edge devices.
- **IoT-Testware** [Schieferdecker et al. 2017]: A framework that allows the setup of test suites and test cases for IoT technologies.
- **IoTSim-Edge** [Jha et al. 2020]: a simulator for IoT devices that extends existing simulators and accounts for specific challenges, such as device heterogeneity.
- **DPWSim** [Han et al. 2014]: A simulation toolkit that mimics the software and protocol features of the OASIS standard Devices Profile for Web Services (DPWS).

From this list, Node-Red and Selenium had the most straightforward setup and usage [Clerissi et al. 2018, Varghese and Sinha 2020]. Being well-known tools in their respective niche, both software provided a website with good documentation and instructions on installing the tool.

Node-Red[5] is a low-code programming tool for event-driven applications. It allows developers to create applications based on events and input streams while accessing extensive user-driven libraries and services. It provides libraries related to communication protocols (e.g., MQTT) and hardware integration. Despite not being a tool built to test IoT systems, these features allow this use case to be handled. Node-Red also provides a drag-and-drop user interface that allows for an easy construction of the necessary workflows to implement the SUT.

Selenium[6], on the other hand, is an automation tool for web applications designed for testing purposes. It was also not created to be used for IoT systems. However, since it can interact with browsers and websites, it can be integrated with Node-Red (that provides

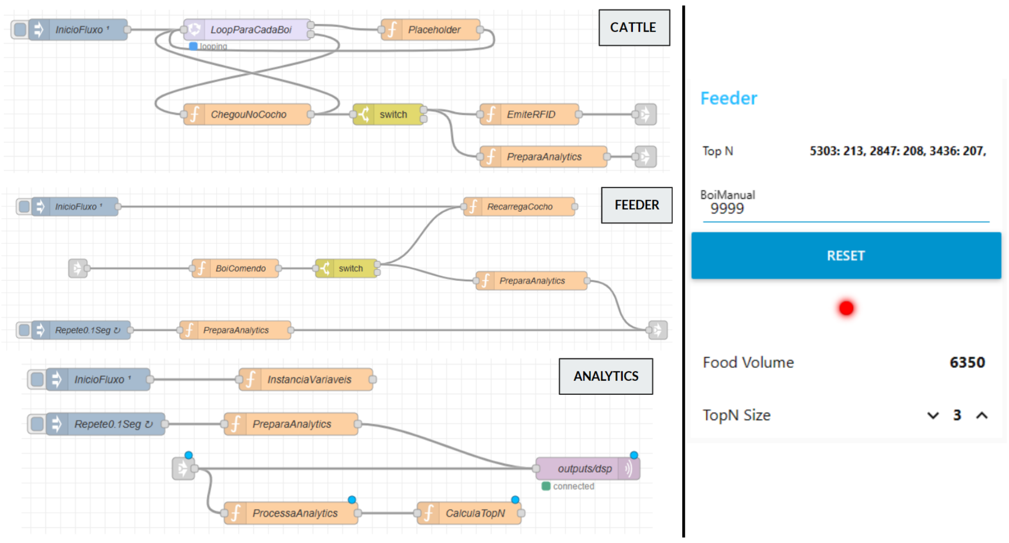
[5]https://nodered.org
[6]https://www.selenium.dev/

a web interface with interactive UI elements through a dashboard) as an attempt to explore functional testing over a simulated system.

The Node-Red implementation of the proposed SUT can be seen in Figure 5, together with the Dashboard used for Selenium to interact with the system (through UI automation). Despite the tool allowing for integration with hardware, only software resources were used for this implementation, with the sensors being simulated within the workflow.

Other tools, such as PatrIoT [Bures et al. 2021], DSL IoTECS [Li et al. 2022], MobIoTSim [Pflanzner et al. 2016], and IoT-Testware [Schieferdecker et al. 2017] provided a good set of instructions for installing and using the tools, and also had their respective repositories updated in 2023. While PatrIoT and IoT-Testware have extensive documentation with multiple examples, MobIoTSim authors responded promptly to our questions and issues with the setup. They ensured that there was still work being done around the application.

However, to the best of our capabilities, we were not able to successfully install and use these tools during the execution of this study, due to library incompatibilities, unexpected errors and/or lack of a detailed guidance in the documentation. Despite that, the found artifacts provided some help, and the fact that their code is still receiving updates is promising that in the near future, these issues could be fixed.



Top N 5303: 213, 2847: 208, 3436: 207,

BoiManual 9999

Food Volume 6350

TopN Size 3

Figure 5. Node-Red implementation and Dashboard of the SUT.

For the remaining tools (IoTSim-Edge and DPWSim), we were unsuccessful in the setup [Jha et al. 2020, Han et al. 2014]. We have contacted IoTSim-Edge authors requesting guidance but have not received a response yet.

The main challenges associated with the tools that were not successfully installed were related to incomplete instructions on how to set them up, aside from old and incompatible dependencies. Developers could address this with up-to-date documentation and setup instructions and the possibility to install the tool from a concise package or executable file. This would enable a wider adoption of the proposed software.

Regarding software testing in IoT systems, most of the proposed tools in the literature are focused on validating the application through the use of simulators and emulators [Ferreira et al. 2023]. This approach provides a series of benefits to the testing activity, such as a degree of control over the real world and the sensors and reducing the cost of having dedicated hardware for testing [Patel et al. 2019].

However, there are still open challenges that this approach does not fully address. For instance, there is a lack of simulators and emulators that can provide a platform for end-to-end testing [Chernyshev et al. 2018]. There is also the impact of duplicating effort since a code written to work on real devices might not be suitable for a simulator of the same devices.

As an alternative to these challenges, developers and testers can search for tools that allow integration with real hardware components. Node-Red and Selenium are good examples since the former provides for communication with real devices, and the latter can interact with interfaces generated by those devices. This factor can contribute to the high adoption of these tools in the industry, as reported by [Minani et al. 2024].

6. Threats to Validity

One of the threats to validity is the final choice of the tools based on the availability of source codes or artifacts in the standard search engines. This was mitigated by an attempt to contact the authors of each tool, which generated some feedback and guidance. This strategy should be expanded to gather more testing tools for further replications and expansions of this study.

A second threat to validity derived from the first one is that, to the best of our efforts, we were only able to set up and test 2 of the final set of tools, which were not proposed directly to solve the issue of testing IoT systems. This situation can limit the comprehension of the software testing scenario for this type of application. For some of the tools we were unable to use, however, there is still ongoing work and updates from the authors, which seems promising that the issues faced could be fixed soon.

Another threat to validity is the reduction in the number of tools that passed all the established criteria compared to the original set of articles. To ensure that the reductions were coherent with the actual state of the tools, the results were constantly compared with what was identified in [Ferreira et al. 2023], as described in Section 5. The tools used for the SUT implementation are also common tools from the industry for testing IoT systems, as discussed in [Minani et al. 2024], reiterating this study's findings. However, future work could address different types of tools that were excluded in this study, such as testbeds.

7. Conclusion and Future Work

Despite many scientific studies and papers exploring the activity of Software Testing in IoT systems, few provide concrete guidance and resources for using and replicating the tools. From a previous mapping, our analyses reveal that only 18% of the software testing tools provided resources and artifacts with enough guidance and documentation to allow developers and testers to use them. This indicates that we still have opportunities to support testing activities in IoT systems.

Among the evaluated tools for software testing in IoT systems, Node-Red and Selenium were the most straightforward tools to install and use. This might be because both are consolidated tools adopted mainly by the software community. Those tools were not designed for this purpose despite being used for testing IoT systems. This adoption reiterates the need for tools dedicated to testing IoT systems with a holistic approach, allowing broad use and replication, as this area of study already lacks tools that will enable end-to-end testing [Chernyshev et al. 2018].

Other tools, such as PatrIoT, DSL IoTECS, MobIoTSim, and IoT-Testware, have very detailed documentation, installation guides, and a code repository that has recently been updated and is responsive to authors. These tools should be the first target of future work in practically exploring testing tools for IoT systems.

Future work could explore the techniques and levels of software testing for IoT systems currently covered by the existing tools and if this coverage can be improved by combining multiple tools simultaneously. Also, as mentioned in Section 6, other studies could look into the tools that were excluded from our criteria, such as Testbeds.

Finally, another evolution to this work could also explore the selected tools to evaluate different aspects associated with their use, such as their accuracy, level of automation, available functionalities, challenges addressed, and so on.

References

Ahmed, B. S., Bures, M., Frajtak, K., and Cerny, T. (2019). Aspects of quality in internet of things (iot) solutions: A systematic mapping study. *IEEE Access*, 7:13758–13780.

Ammann, P. and Offutt, J. (2016). *Introduction to software testing*. Cambridge University Press, Cambridge, UK.

Bosmans, S., Mercelis, S., Denil, J., and Hellinckx, P. (2019). Testing iot systems using a hybrid simulation based testing approach. *Computing*, 101:857–872.

Bures, M., Ahmed, B. S., Rechtberger, V., Klima, M., Trnka, M., Jaros, M., Bellekens, X., Almog, D., and Herout, P. (2021). Patriot: Iot automated interoperability and integration testing framework. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 454–459, Porto de Galinhas, Brazil. IEEE, IEEE.

Bures, M., Cerny, T., and Ahmed, B. S. (2018). Internet of things: Current challenges in the quality assurance and testing methods. In *9th iCatse Conference on Information Science and Applications*, pages 625–634, Hong Kong, China. Springer.

Bures, M., Klima, M., Rechtberger, V., Bellekens, X., Tachtatzis, C., Atkinson, R., and Ahmed, B. S. (2020). Interoperability and integration testing methods for iot systems: A systematic mapping study. In de Boer, F. and Cerone, A., editors, *Software Engineering and Formal Methods*, pages 93–112, Cham. Springer International Publishing.

Chernyshev, M., Baig, Z., Bello, O., and Zeadally, S. (2018). Internet of things (iot): Research, simulators, and testbeds. *IEEE Internet of Things Journal*, 5(3):1637–1647.

Clerissi, D., Leotta, M., Reggio, G., and Ricca, F. (2018). Towards an approach for developing and testing node-red iot systems. In *Proceedings of the 1st ACM SIGSOFT*

International Workshop on Ensemble-Based Software Engineering, pages 1–8, Lake Buena Vista, FL, USA. ACM.

Cristea, R., Feraru, M., and Paduraru, C. (2022). Building blocks for iot testing - a benchmark of iot apps and a functional testing framework. In *2022 IEEE/ACM 4th International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT)*, pages 25–32, New York, NY, USA. IEEE.

Defalque, G., Santos, R., Pache, M., and Defalque, C. (2023). A review on beef cattle supplementation technologies. *Information Processing in Agriculture*.

Delamaro, M., Jino, M., and Maldonado, J. (2013). *Introdução ao teste de software*. Elsevier Brasil, Rio de Janeiro, RJ.

Demirel, S. T., Demirel, M., Dogru, I., and Das, R. (2019). Interopt: A new testing platform based on onem2m standards for iot systems. In *2019 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6, Istanbul, Turkey. IEEE, IEEE.

Estebsari, A., Patti, E., and Barbierato, L. (2018). Fault detection, isolation and restoration test platform based on smart grid architecture model using intenet-of-things approaches. In *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*, pages 1–5, Palermo, Italy. IEEE.

Ferreira, V. G., Herrera, C. G. a., Souza, S., Santos, R. R. d., and Souza, P. S. L. d. (2023). Software testing applied to the development of iot systems: Preliminary results. In *Proceedings of the 8th Brazilian Symposium on Systematic and Automated Software Testing*, SAST '23, page 113–122, New York, NY, USA. Association for Computing Machinery.

Gartner (2020). Gartner forecasts worldwide iot endpoint electronics revenue to reach $389 billion in 2030.

Han, S. N., Lee, G. M., Crespi, N., Heo, K., Van Luong, N., Brut, M., and Gatellier, P. (2014). Dpwsim: A simulation toolkit for iot applications using devices profile for web services. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 544–547, Seoul, Korea. IEEE, IEEE.

Haris, I., Bisanovic, V., Wally, B., Rausch, T., Ratasich, D., Dustdar, S., Kappel, G., and Grosu, R. (2019). Sensyml: Simulation environment for large-scale iot applications. In *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, page 3024–3030, Lisbon, Portugal. IEEE Press.

Jha, D. N., Alwasel, K., Alshoshan, A., Huang, X., Naha, R. K., Battula, S. K., Garg, S., Puthal, D., James, P., Zomaya, A., et al. (2020). Iotsim-edge: a simulation framework for modeling the behavior of internet of things and edge computing environments. *Software: Practice and Experience*, 50(6):844–867.

Klima, M., Bures, M., Ahmed, B. S., Bellekens, X., Atkinson, R., Tachtatzis, C., and Herout, P. (2023). Specialized path-based technique to test internet of things system functionality under limited network connectivity. *Internet of Things*, 22:100706.

Kuroiwa, T., Aoyama, Y., and Kushiro, N. (2019). A hybrid testing environment between execution test and model checking for iot system. In *2019 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–2, Las Vegas, USA. IEEE, IEEE.

Li, J., Nejati, S., Sabetzadeh, M., and McCallen, M. (2022). A domain-specific language for simulation-based testing of iot edge-to-cloud solutions. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*, pages 367–378, New York, NY, USA. Association for Computing Machinery.

Minani, J. B., Sabir, F., Moha, N., and Guéhéneuc, Y.-G. (2024). A multimethod study of internet of things systems testing in industry. *IEEE Internet of Things Journal*, 11(1):1662–1684.

Murad, G., Badarneh, A., Qusef, A., and Almasalha, F. (2018). Software testing techniques in iot. In *2018 8th International Conference on Computer Science and Information Technology (CSIT)*, pages 17–21.

Patel, D., Mehtre, M., and Wankar, R. (2019). Simulators, emulators, and test-beds for internet of things: A comparison. In *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 139–145, Palladam, India. IEEE, IEEE.

Pflanzner, T., Kertesz, A., Spinnewyn, B., and Latré, S. (2016). Mobiotsim: Towards a mobile iot device simulator. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 21–27, Vienna, Austria. IEEE.

Saraiva, R., Mello, P., Soares, P., Souza, M., and Cortés, M. (2019). Adoption of software testing in internet of things: A systematic literature mapping. In *Proceedings of the IV Brazilian Symposium on Systematic and Automated Software Testing*, pages 3–11, Salvador, Brazil. ACM, ACM.

Schieferdecker, I., Kretzschmann, S., Rennoch, A., and Wagner, M. (2017). Iot-testware - an eclipse project. In *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pages 1–8, Prague, Czech Republic. IEEE.

Varghese, N. and Sinha, R. (2020). Can commercial testing automation tools work for iot? a case study of selenium and node-red. In *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pages 4519–4524, Singapore. IEEE, IEEE.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, pages 1–10, London, England, United Kingdom. ACM, ACM.

Wu, M., Lu, T.-J., Ling, F.-Y., Sun, J., and Du, H.-Y. (2010). Research on the architecture of internet of things. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, volume 5, pages V5–484–V5–487.