IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 22, 2021, accepted December 12, 2021, date of publication December 15, 2021,
date of current version December 28, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3136025

A Comparison of Open-Source Home Automation Systems

BRIAN SETZ[1], (Graduate Student Member, IEEE), SEBASTIAN GRAEF[2], DESISLAVA IVANOVA[2], ALEXANDER TIESSEN[2], AND MARCO AIELLO[1], (Senior Member, IEEE)

[1]Service Computing Department, University of Stuttgart, 70569 Stuttgart, Germany
[2]Master in Computer Science, University of Stuttgart, 70569 Stuttgart, Germany

Corresponding author: Brian Setz (brian.setz@iaas.uni-stuttgart.de)

This work was supported by the Netherlands Organisation for Scientific Research (NWO) in the Framework of the Indo-Dutch Science Industry Collaboration Program with Project NextGenSmart Data Center (DC) under Grant 629.002.102.

ABSTRACT Homes are becoming an ecosystem of digital devices and appliances, which can be interconnected and controlled. This interconnection can be facilitated by a central smart hub on which home automation software is deployed. Commercially available hubs, while easy to install and use, often support a limited set of devices and protocols, and have a high total cost of ownership. Open-source home automation systems provide an affordable and open alternative, bringing support for devices and services that are unsupported by commercial alternatives. In recent years, the number of available open-source home automation systems has increased drastically. Each system comes with its own set of functionalities and limitations, making choosing a specific solution challenging, as a wrong decision may be costly. In this work, we overview 20 of the prominent open-source home automation systems, from which we select the four most promising ones. To evaluate and compare these systems, we identify key features from a set of use cases and extract specific features for home automation. This results in a two phase study. In the first phase, we perform a use case based analysis based on the extracted features. In the second phase, we perform a criteria-based analysis with 34 criteria that covers aspects such as setup time, quality of documentation, pricing, and hardware requirements. We also identify the commonalities in the architecture that emerge from the systems. The results help to identify the strengths and weaknesses of the various systems and can help the developer and the practitioner make an informed choice when selecting an open-source home automation solution.

INDEX TERMS Home automation systems, Internet of Things, open-source projects, home automation architecture.

I. INTRODUCTION

The Internet of Things paradigm embraces the idea that devices, related to physical things, are always connected to the Internet and able to provide data and actuate in the physical world [1]. Residential buildings are becoming a prominent example of this trend as everyday objects are increasingly equipped with digital controllers that are connected to home local networks. These buildings are usually referred to as *smart homes*. Typically, in smart homes, devices are connected to a central hub on the local network or directly to the Internet via a home router. IoT provides monitoring and control of physical spaces and instruments, [2], e.g., monitoring the room temperature and keeping it within a given range in a certain time of the day. Monitoring and control are the foundation for home automation and, in turn, the enabler of truly smart homes. The goals of a smart home are manifold. They range from support to people with specific inabilities or disabilities, to simply increase the comfort and pleasure of home living. A goal that is becoming gaining relevance is that of improving the energy efficiency and overall sustainability of homes [3].

Today, we are able to talk about smart homes because the pervasiveness of IoT devices and the ease of their installation and interoperation has brought them to the masses. This has been a long journey. About 40 years ago, home automation components were expensive and with non standardised interfaces, making them isolated components that

The associate editor coordinating the review of this manuscript and approving it for publication was Tyson Brooks.

would not coordinate and cooperate with other home components [4]. Vendors' lock-in was the standard practice. Slowly standards for home automation started to appear and interoperability efforts occurring in other areas of ICT contaminated also the home environment. More than 15 years ago, we proposed the use of XML based web services standards for resolving the home automation problem and we illustrated an application for supporting the elderly in their own homes [4], [5]. Today, the situation is very different. Not only are internetworking and interoperation standards widely adopted, but there are also commercially available smart home hubs and even (open source) software for device integration. Smart hubs facilitate the integration of products from different vendors, such as the Fibaro Home Center 3, Athom Homey, or the Samsung SmartThings Hub v3. While these hubs are easy to install, and easy to use, they do have their limitations. Typically, these hubs support only a few protocols and device types, and have a high total cost of ownership [6]. Vendor lock-in is also still a challenge that consumers face when using commercial smart home hubs, as these systems often promote the use of devices that are manufactured by the same company. On the contrary, open-source home automation systems, as open-source software enables the use of well-developed system free of charge, though quality control may suffer [7]. In recent years there has been a significant increase in the number of free and open-source home automation systems. Their open-source nature allows them to provide support for hundreds, if not thousands, of diverse devices, overcoming the vendor lock-in issues that some of the commercial solutions have. At the same time, there is no cost associated with the software itself, which can often run on cheap single board computers such as the Raspberry Pi. Therefore, the total cost of ownership is also reduced. These types of systems have their own features and limitations, as will become clear later in this work.

The challenge today is thus that of selecting the appropriate open-source home automation system. First of all, there is a vast number of available systems, each with a varying number of functionalities, different levels of support for devices and protocols, and also variations in overall maturity and quality of the software. Second, while vendor lock-in is not a critical issue with these open platforms, migrating from one system to another system is time consuming due to the lack of migration tools. Finally, open-source projects come and go, which means the longevity of the system also needs to be considered. Projects with fewer contributors and low commit activity in the community are at higher risk of becoming stale or inactive. Therefore, it is important to be able to choose the system that fits the requirements of the users. To the best of our knowledge, there are no works that: (1) present an overview of available systems, (2) identify which of these systems are actively developed, or (3) perform any type of comparisons between these systems to identify differences in functional and non functional requirements, as well as identify gaps in the state of the art.

The present work is multi-purpose: it is a framework which can be applied to related domains by researchers, it is a tool for the practitioner to help make an informed decision when designing a home automation system, and it is an overview of open-source software for smart homes for the hobbyist. This work achieves these goals by evaluating 20 home automation systems based on five core criteria, and making a selection of the top four systems based on the combined score of these criteria. These four systems are then methodologically compared in greater detail. The detailed comparison consists of two views: a use case based analysis to determine what is supported, and a criteria-based analysis that considers other useful aspects such as setup complexity and pricing. This evaluation is practical in nature; to evaluate each system, we installed and configured it on specific hardware. When applicable, the criteria is evaluated using the deployed system. Furthermore, a reference architecture for home automation is identified based on the commonalities that emerge from the analysis of the four systems.

The rest of this paper is organised as follows. In Section 2, we review related work and discuss the gaps in the literature. The methodology is described in Section 3, from initial selection of the systems to the detailed criteria. In accordance with the methodology, the system selection is discussed in Section 4, as well as the emerging architecture. The use case based analysis is the content of Section 5, while the following Section 6 contains the analysis of the selected systems in greater details using a predetermined set of criteria. The discussion of the results from the comparisons is presented in Section 7. Conclusions and open perspectives are the content of the final section, Section 8.

II. RELATED WORK

Home Automation Systems and Smart Homes have been the focus of many publications, though, to the best of our knowledge, none of them review and analyze existing open-source project for them in detail. Next we review recent surveys on home automation systems.

Taiwo, *et al.* propose a taxonomy of home automation systems, with the main focus on technology, trends, and challenges [8]. The systems included come from existing literature and are primarily closed-source studies with limited to no adoption. The taxonomy highlights five components of a home automation system: (1) application area, (2) automation layers, (3) protocols, (4) platforms, and (5) sensors. These components are also important to study open-source systems discussed in the present work. Taiwo, *et al.* also identify trends and challenges in home automation. The current research trends focus on: energy efficiency and energy reduction, privacy and security, and innovative technologies. Furthermore, the following challenges are identified: authentication and authorization, privacy, high cost and incompatibility, and energy management. The authors conclude by stating that the next steps are the incorporation of home automation systems within smart cities infrastructures.

Jerabandi, *et al.* provide an overview of existing home automation and IoT frameworks [9]. While their work recognizes the prevalence of generic commercial and open-source systems, the focus is on academic works that address very specific problems. Based on the review of 14 systems, the authors identify numerous design issues related to home automation systems. The issues are: serviceability, scalability, programmability, auto-configuration, centralised vs. decentralised architectures, heterogeneity, transparency, security, open standards, robustness, and energy efficient communication. A generic framework for home automation systems is briefly described. In our work, we recognise that the issues identified by Jerabandi, *et al.* also exist in open-source systems, and that some systems are more effective at addressing specific issues.

Smirek, *et al.* investigate the criteria required for universally usable interfaces in the domain of home automation systems [10]. From the user perspective three criteria are identified: abstraction, pluggable user interfaces, and adaptive user interfaces. From the designer's perspective the following three criteria are mentioned: modularity and clearly defined interfaces, expandability, and openness. The authors highlight that openness is an important prerequisite for experts to contribute to the system. This confirms the findings in our work that highlight the benefits of open-source home automation systems. Our work differs in the fact that the focus is not exclusively on user interfaces, and the criteria that are defined in this work are less abstract and more granular.

Faroom, *et al.* perform a comparison of home automation approaches for people with a mobility or physical disability [11]. Their work focuses on six home automation approaches which are specifically tailored at usability and accessibility. The focus is on four aspects: ease of use, installation costs, scalability, and security. The authors conclude that home automation systems are solidifying their position in the market. While their focus is on highly customized and tailored solutions, the focus of our work is on generic, open-source home automation systems. The aspects covered by Faroom, *et al.* are also relevant for the present work.

Derhamy, *et al.* focus on commercial IoT frameworks [12]. In their work they analyse 14 different frameworks and platforms. There is particular emphasis on commercially available IoT frameworks for home automation, including: IPSO Alliance, IoTivity, AllJoyn, and Thread. The authors recognize that for a platform or framework to succeed they must: (1) securely expose API's for third parties, (2) provide protocol interoperability with third party API, as well as protocol extensibility, (3) enable constrained devices to participate, and (4) enable management and governance of heterogeneous networks of device and applications. While our work focuses on freely available and open-source systems, we remark that for home automation systems to succeed they need to adhere to the same values as defined for IoT platforms and frameworks, this can also be deduced from the results of our work.

Risteska Stojkoska, *et al.* feature a review of the state-of-the-art in Internet of Things applications in smart homes [13]. They have identified five challenges related to this topic. First of all, the use of edge computing shows great promise despite a lack of adoption in existing platforms and frameworks. The second challenge relates to big data, specifically regarding the overall performance when handling big data. Selecting the appropriate network protocols is another challenge, it is commonly a trade-off between cost and performance. The fourth challenge pertains to interoperability, which is caused by a heterogeneous smart home landscape. Finally, security and privacy is another challenge which is highlighted, which is primarily due to the way data is transmitted wirelessly in smart homes.

To summarize, to the best of our knowledge, there are no works that present an overview of available home automation systems, that identify which of these systems are actively developed, or that perform comparisons between these systems to identify differences in functional and non functional requirements, as well as identify gaps in the state of the art.

III. METHODOLOGY

The approach that is taken in this work consists of three steps. First, a list of home automation systems is compiled and ranked. The list of systems is compiled by searching for open-source systems on numerous search engines (Wikipedia, Bing, Google) and online source code management platforms (GitHub, BitBucket, GitLab, Launchpad). Search terms that were used include: Home Automation, Building Automation, IoT Platform. The top four home automation systems are selected, based on their individual scores, for further analysis. Next, a catalogue of 13 system features is created based on 17 use cases, and each of the four systems is subjected to this catalogue of features in order to determine which features are supported. The final step is an extensive analysis of 34 different criteria to which the four systems are subjected, each of these criteria are scored from 0 to 5. For the evaluation, the systems are deployed on a ThinkPad E490 with an Intel Core i5-8265U CPU. What follows is a description for each of the three steps.

A. INITIAL SELECTION

There is a wide variety of open-source home automation systems available. To reduce the number of systems that are subjected to the detailed analysis, we perform an initial selection by ranking each system based on five criteria. Each criteria has a score i where $i \in \mathbb{Z} : i \in [0, 5]$. The criteria scores for each system are summed to obtain the final score. The criteria for the initial selection are as follows:

S1 **Commits**: the number of commits to the source code repository can be an important indicator of the level of activity within a project. While not all commits are of equal importance, their frequency is useful indicator. The commit score is calculated according to Equation 1.

$$s(x) = log_{10}(x) \tag{1}$$

TABLE 1. Use cases and their corresponding features.

User Cases \ Features	F3.1: Mobile Notifications	F4.3: Automation Rules	F2.1: GPS-Tracking	F4.1: Sensor – Read	F5.1: Mobile Remote Control	F4.2: Device – Actuation	F4.4: Media Streaming	F5.2: External API Calls	F1.1: Display (Paper-White)	F1.2: Dashboard	F2.2: Presence Detection	F5.3: Scheduler	F5.4: Biometric User Auth.
In case of a fire, I would like to be informed and receive notifications on my mobile phone.	✓	✓		✓									
When it is winter and it is cold, I want my heating to turn on before I get home from work.		✓	✓	✓		✓							
Even when my lamps and light fixtures are from different manufacturers, I want to be able to control all of them from one device.					✓	✓							
While doing the groceries I want to be able to access a video stream inside my fridge to check its contents.							✓						
I would like to know when I should empty my rain barrels in order to benefit the most from an upcoming rain storm.	✓	✓		✓				✓					
I want to see, in real time, much power my solar panels currently generate.		✓		✓					✓	✓			
When I leave the house, and I am the last person to do so, all lights should be turned off in case I forget.		✓	✓			✓					✓		
If the CO detector is low on battery, I would like to receive an email notification that the battery needs to be replaced. This will avoid triggering the alert that goes off when the battery is low, and usually causes panic as people may think CO is leaking into their house [14]	✓	✓		✓									
Every morning at 5:00 AM start brewing my coffee automatically so it will be ready for me when I wake up [14].						✓						✓	
During the day, whenever I walked into the bathroom the light would come on. However, after a certain time, when it is night, when I walk in the bathroom I would want a much softer light to come on [14].		✓				✓					✓		
When someone rings the doorbell it triggers both the television and the lights. If the TV is on then when the doorbell rings the TV should mute itself and/ or pause depending on what the input the TV is. Additionally, the lights should change color to indicate someone is at the door, in case the doorbell was not heard [14].		✓		✓		✓							
I want the sprinkler system to water the lawn only when: it is not raining already, and it is between 4 am and 6 am, and the temperature is above 50 degrees Fahrenheit. [14]		✓		✓		✓		✓					
When I am home and shut my windows, I want all of my doors to lock automatically [14].		✓		✓		✓							
I would like to receive a notification at home when the stock market passes a certain threshold.	✓	✓						✓					
I only want people I know, or people I have given permission, to be able to access my smart home.					✓			✓					✓
My home dashboard should indicate how many unread mails I have.									✓	✓			
My appointments for the day should be visualised on the display of my dynamic calendar, my garbage collection schedule should also be integrated.								✓				✓	

where x is the number of commits. When $s(x) > 5$ then $s(x) = 5$.

S2 **Stars**: the number of stars on a source code repository is comparable to the number of likes on social media platforms. These stars are commonly used as a proxy to determine the overall popularity of a repository [35], [36]. The score for this criteria is calculated in the same manner as the number of commits, using Equation 1, where x becomes the number of stars.

S3 **Latest Commit**: the date of the latest commit is checked in order to penalise inactive projects. This is done by looking at the number of years since the latest commit, and applying Equation 2.

$$s_{commits}(x) = \begin{cases} 5 - x, & \text{if } x \leq 5 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where x is the number of years since the last commit.

S4 **Documentation**: the quality of the documentation is crucial for home automation systems. For the initial selection, if the system has any documentation, 5 points are assigned. If the system has no documentation, 0 points are assigned. This is done in order to penalise projects without any documentation. A more detailed review of the documentation will be performed for the top 4 systems, as part of the criteria-based analysis.

S5 **Contributors**: the number of people who contribute to the home automation system are also taken into account. The number of contributors is important for multiple reasons. First of all, the longevity of the project can be jeopardized when there are only a handful of contributors. Furthermore, there is a positive correlation between the number of contributors and the ease with which the source code of a system can be extended or modified. And finally, it is another indication of the

TABLE 2. Initial selection results.

Names	Rank	Score	Commits	Stars	Latest Commit	Docs	Contributors
Home Assistant [15]	1	24	4.6	4.6	5	5	5
Domoticz [16]	2	21	4.1	3.5	5	5	3
openHAB [17]	3	17	3.2	2.7	5	5	1
ioBroker [18]	4	17	3.2	2.7	5	5	1
HomeGenie [19]	5	16	3	2.4	5	5	1
Calaos [20]	6	16	3.1	2.2	5	5	1
Wirehome [21]	7	16	2.6	2.3	5	5	1
OpenMotics [22]	8	16	3.5	1.4	5	5	1
Freedomotic [23]	9	16	3.2	2.6	4	5	1
FHEM [24]	10	15	4.3	1.1	5	5	0
MisterHouse [25]	11	15	3.6	2.3	3	5	1
OpenNetHome [26]	12	14	2.7	1.6	5	5	0
Ago Control [27]	13	14	3.6	0.5	4	5	1
TheThingSystem [28]	14	12	3.1	2.5	0	5	1
üAutomate [29]	15	11	2.2	1.1	3	5	0
Neon HomeControl [30]	16	11	1.9	1.4	3	5	0
Pytomation [31]	17	11	3	1.9	0	5	1
Smarthomatic [32]	18	10	2.8	1.5	0	5	1
Smart Haus [33]	19	8	2.1	0.8	5	0	0
AutoBuddy [34]	20	8	2.4	1.2	4	0	0

popularity of the system. The score for this criteria is calculated using Equation 3.

$$s_{contributors}(x) = \begin{cases} 5, & \text{if } x > 1000 \\ 3, & \text{if } 100 < x \leq 1000 \\ 1, & \text{if } 2 < x \leq 100 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where x is the number of contributors to the project.

In case home automation systems have identical total score after evaluating all criteria, a tie breaker is decided as follows: whichever system has the higher sum of scores for the stars, latest commits, and documentation criteria. In case the scores remain equal, the precise values of Equation 1 are used. After determining the overall ranking, the top four systems are selected, and these systems are analysed in great detail for the remainder of this work.

B. USE CASE BASED ANALYSIS

The goal of the use case based analysis is to obtain a high-level overview of the functionality which the four selected home automation systems offer. The list of features to which each of the selected systems is subjected has been extracted from 17 different use cases. These use cases have been partially selected from a survey by Abbas [14]. The remaining use cases are defined based on the collective academic and industrial experience of the authors. Each use case requires the home automation system to provide a certain set of features in order to fulfil the requirements. The collection of use cases and the corresponding set of features that have been extracted from the use cases is shown in Table 1.

The thirteen high-level features, that have been identified based on the uses cases, are divided into five categories: Visualisation (F1), Localization (F2), Notification (F3), Data-Handling (F4), Interaction (F5). The high-level features of the four selected systems will be evaluated. When a system supports a feature natively, or provides an official plugin (near-native) then no points are deducted. If the feature is supported only through third-party plugins or applications, or requires a workaround, then 0.5 points are deducted. In case the feature is entirely unsupported, 1 point is deducted. The final score is calculated according to Equation 4.

$$s_{features}(x) = \begin{cases} 5 - x, & \text{if } x \leq 5 \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

where x is the number of points that are deducted for features that are not (fully) supported. A complete description of each of the features that have been identified based on the use cases can be found in the Appendix A "Description of Features".

C. CRITERIA-BASED ANALYSIS

In the final step of the analysis, 34 distinct criteria are evaluated for each of the four remaining home automation systems. Similar to the initial selection, each criteria in this part of the analysis also has a score i where $i \in \mathbb{Z} : i \in [0, 5]$. Where 5 indicates that a criterion has been fully satisfied, and 0 indicates that the criterion is entirely unfulfilled. A complete description of each criteria can be found in Appendix B "Description of Criteria".

IV. SYSTEM SELECTION

Home automation systems available range from recent projects to quite mature ones. Based on the proposed methodology (see Section III-A), 20 systems are discovered and considered for further analysis. As the number of systems is great, a selection is made to reduce this number to 4 systems. As part of the selection process, each of these systems are evaluated based on the five metrics: commit count, number of stars, date of latest commit, documentation, and number of contributors. A total score for each system is obtained by summing the scores of the individual metrics.

The results of the initial selection process are presented in Table 2. The Table is populated with data collected on the 16th of July, 2021. The scores of the individual metrics for each of the 20 systems are shown, as well as the total score of each system. Based on the scoring, the top 4 systems are: Home Assistant, Domoticz, openHAB, and ioBroker. These

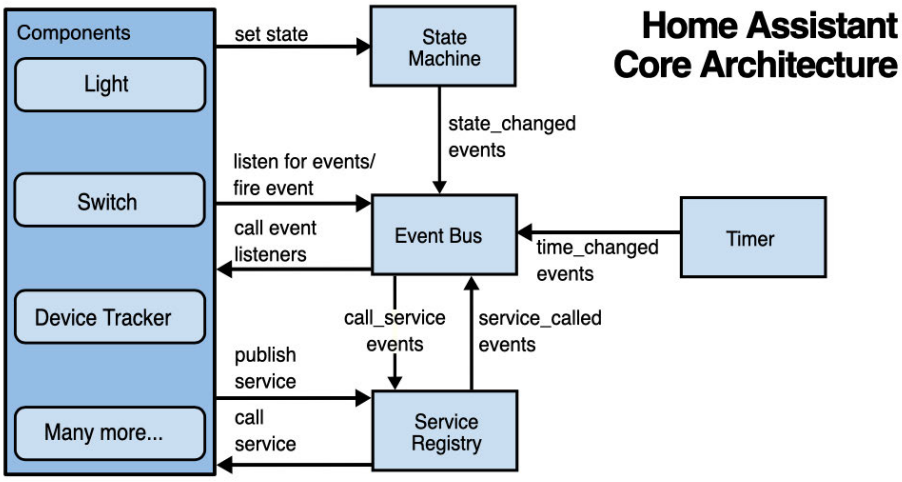

FIGURE 1. Home Assistant core system architecture [37].

are the systems that are selected and analysed in greater detail. What follows next is a brief description of each of the selected system, covering its history, how to contribute as a developer, the software license that applies to the system, as well as the architecture and conceptual model. And finally, based on the commonalities between the systems, a generic home automation architecture is defined.

A. HOME ASSISTANT

Home Assistant was founded by *Paulus Schoutsen*, and is an open-source system maintained by a worldwide community. It currently provides over 1.500 different integrations. These integrations add support for new devices, adapter protocols, user interface modifications or extensions, and the integration of external services. The configuration of Home Assistant is mainly done through the use of YAML-configuration files. Though more configuration options are being added to the user interface instead [38].

The Home Assistant core and its integrations are written in the Python programming language. The architecture of the system is shown in Figure 1. As can be seen in this figure, the event bus is the core of the system, listening to and firing events to other components. One of these components is the State Machine, used to keep track of the state of entities. Each change in a state fires an event that is handled by the event bus. The Timer component generates regular 'time changed' events. The Service Registry allows other components to register services, and allows these services to be discovered.

In Home Assistant, every sensor, controller, and actor is a represented as a device which can be organized into groups. Every device and integration is represented as one or more entities, each having attributes representing the state of the entity. For automation, there is the ability to create automation rules which can be triggered by various things, including certain states of entity attributes, when a user enters a defined area, when the sun is set, or when the server restarts. In addition, the automation rules can have conditions to prevent the execution of the rule if the conditions are not satisfied. Lastly, there are actions that specify what should be executed when the automation rule fires.

B. DOMOTICZ

Domoticz is a home automation system that is able to monitor and configure a variety of devices. The first version of Domoticz was released in December 2012. Thanks to a responsive user interface, the system is usable on both desktop and mobile devices. It is maintained by a large and active community of developers.

Domoticz is implemented in the C++ programming language. The project also implements its own web server, written in C++ as well. Unfortunately, little documentation is available on the architecture of Domoticz and on the underlying concepts of the project. However, Figure 2 shows an example of a typical Domoticz setup. As is shown, sensors and actuators are connected to Domoticz through MQTT. The actuators can be triggered by automation rules defined in an external system such as Node-RED, or Domoticz's own Blockly rule builder. The backend handles the incoming data, which can then be displayed on the GUI.

C. openHAB

Open Home Automation Bus, commonly known as openHAB, is an "open-source home automation controller" [40]. The first lines of code were added to the project in 2010. The project does not have a singular creator, instead, it has been implemented by a community of volunteers. The openHAB system is vendor-independent and it works with many protocols and devices. This is one of its main strengths and goals: providing a uniform user experience regardless of the vendors and subsystems it interfaces with.

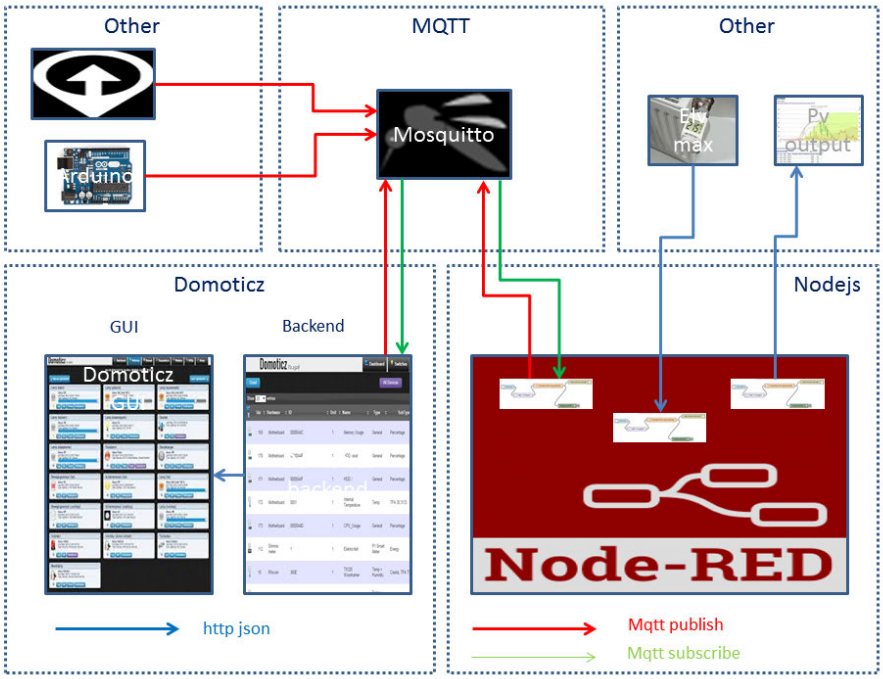

FIGURE 2. Domoticz system architecture [39].

openHAB is primarily written in the Java programming language. A representation of the openHAB architecture can be seen in Figure 3. The figure clearly demonstrates that the Event Bus is the central component of the system, enabling communication between the other openHAB components. Bindings enable uniform communication between the system and the devices or services. Thanks to openHAB's extensibility, there are many different user interfaces available to interact with the system, as well as a REST API.

Conceptually, openHAB consists of five important elements: Things, Channels, Bindings, Items, and Links. Things are objects that are physically added to the system, and that can provide one or more functions. A temperature and humidity sensor is one physical Thing that provides two functionalities: temperature sensing, and humidity sensing. Each functionality of a Thing is exposed through a Channel. Bindings are adapters, they enable access to Things through the system and hide hardware specific details. Items are stateful, and provide functionality that can be used by application or in automation logic. A Link connects one or more Channels to one or more Items. The act of linking Channels and Items enables the functionality provided by an Item to be access through that specific Channel. Figure 4 illustrates this connection between Things, Items, Channels, and Links.

D. ioBroker

The first version of ioBroker was published in 2014 by the company *ioBroker GmbH*. It is the successor of the *CCU.IO* project, which was terminated in April 2015 [38]. The goal of ioBroker is to integrate heterogenous smart home devices and systems. At the time of writing, ioBroker offers 350+ adapters to integrate with different devices and systems. The system is non-commercial software, and is developed and maintained by volunteers. One of the main advantages of ioBroker is that all configuration can be done through a web interface. This makes the system accessible by a wide range of users [38]. Additionally, ioBroker uses the local API of a device, when this is supported, in order to bypass the online cloud services of the vendors. The benefit of this approach is that sensitive data remains local and any security vulnerabilities that might exist in the cloud service are avoided [41].

The ioBroker core is primarily written in JavaScript. The adapters are also written in JavaScript, though Typescript can be used as well. The architecture of the system is shown in Figure 5. It is clear from the figure that two databases play an important role: the objects database and the states database. The object database is responsible for storing meta data and configurations, whereas the states database is used to keep track of the state of devices and services. By default in-memory databases are used, but there are adapters available to support other types of databases. Adapters are used to integrate with different IoT devices and systems, resulting in a loosely coupled architecture. The controller is responsible for managing the adapter processes.

From a conceptual viewpoint, ioBroker is an extremely modular system. Each module or adapter is responsible for a specific function. Even the administration user interface is developed as a separate adapter. A central coordinator, also known as the js-controller, is responsible for managing the adapters and realising the communication between them.

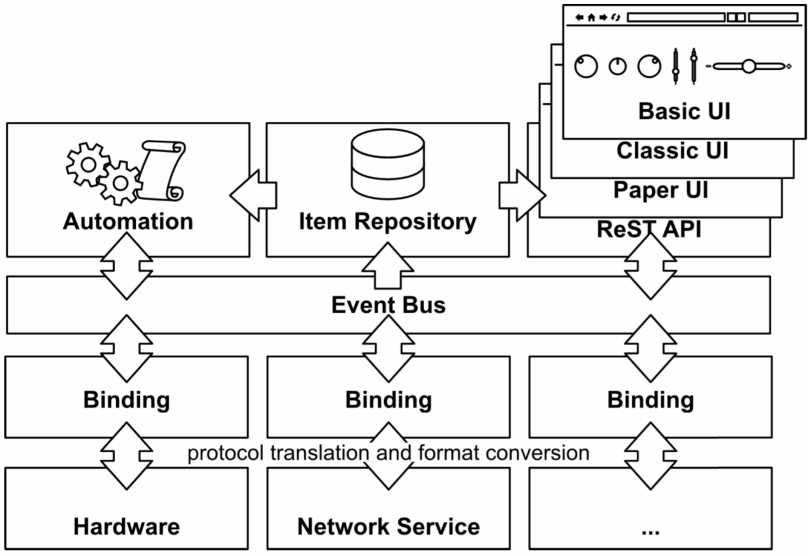

FIGURE 3. openHAB system architecture [40].

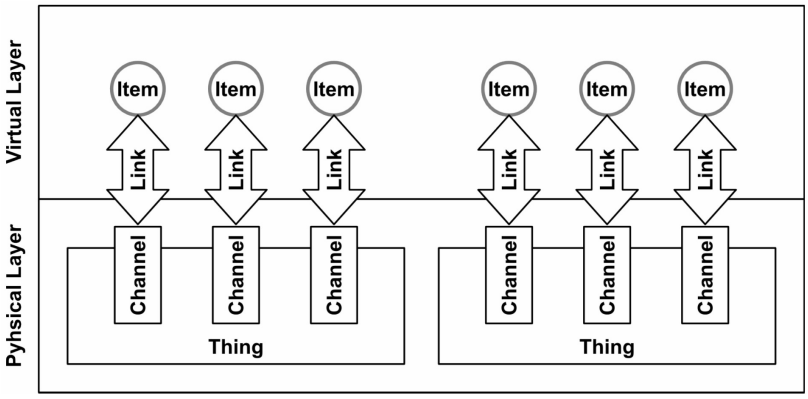

FIGURE 4. openHAB conceptual architecture [40].

E. THE HOME AUTOMATION SYSTEM ARCHITECTURE

The architecture of a home automation system influences the capabilities and the characteristics of the system itself. It is clear that the architectures of the top four systems show a large number of commonalities. On the basis of these, we discuss concepts, components, and designs that are present in multiple systems and can be considered jointly core architectural principles for a home automation system. Due to a lack of documentation, especially with respect to system architecture, the Domoticz system is not included. This emerging architecture is shown in Figure 6. What follows next is a description for each of the components.

- *Database* is a critical component to store historical device data. The database is generally also used to store the current state of devices and other entities within the system. Furthermore, it is also used to track all devices and extensions. All three systems, Home Assistant, openHAB, and ioBroker support a multitude of database systems.
- *Web-based User Interface* is the most popular option to interact with the home automation system, though not the only one. While Home Assistant leans heavily on a single user interface which is highly personalizable, openHAB and ioBroker support multiple, entirely distinct, user interfaces each with their own strengths and weaknesses.
- *(REST) API* to interact with the system by means other than the web-based user interface. All of the investigated systems have an API of some form and shape to enable interactions with the system, such as triggering actuators or automation rules.
- *Event Bus* is present in both Home Assistant and openHAB, the event bus plays a central role in facilitating the asynchronous communication between the components. The event bus is also used to listen to events generated by devices or by the system itself. On the contrary, ioBroker opts for a direct TCP/IP connection for communication between components.

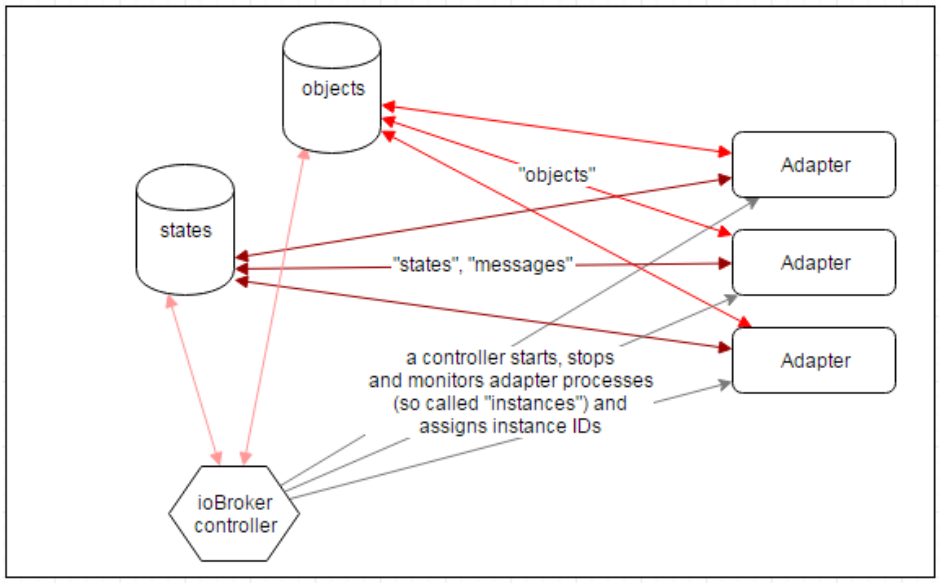

FIGURE 5. ioBroker system architecture [42].

- *Physical Device,* such as a temperature sensor or relay, needs to be represented in the home automation system. Home Assistant simply refers to a physical device as 'device', openHAB uses the label 'thing'. ioBroker does not make a clear distinction, though in general physical devices are abstracted and represented as the 'object' data type.
- *Virtual Device* can represent a physical device that has multiple sensors, and sometimes multiple actuators, to uniformly abstract individual features provided by physical devices. In Home Assistant, a virtual device is called an 'entity', while openHAB uses the name 'item'. Again, ioBroker does not support such abstraction.
- *Extensions* are critical for home automation system given the high dynamicity of the related ecosystem. Extensions allow the systems to offer support for many devices and services. Though, there is no agreement on what they should be called; openHAB talks about 'bindings', Home Assistant considers them 'integrations', and ioBroker uses the concept of 'adapters'.
- *Rule Engine* enables the automation of the home to define and execute rules. Home Assistant and openHAB provide their own components to wire together devices and events. ioBroker relies on extensions, such as Blockly and Node-RED. Though both Home Assistant and openHAB also support Node-RED.

V. USE CASE BASED ANALYSIS

The features supported by the four selected systems largely overlap, though they are not exactly the same. Let us consider the four system under the lenses of those identified in Table 1. The support for each of these features is determined and translated into a numeric score as described in Section III, Equation 4. In order to determine which features are supported, each system is deployed in practice in order to verify the support of each feature. The results are summarised in Table 3 where the symbol "✓" indicates native or near-native support of the feature; "**O**" indicates that the feature is supported by means of a third-party solution, or that it requires significant effort (e.g. writing custom scripts) from the user; and finally, "**-**" indicates that the feature is not supported.

F1.1 (ePaper / eInk Display): Home Assistant is the only system that supports ePaper Displays natively. There are third-party alternatives available for Home Assistant, such as Basic-Hass-Dash or HASS eInk Display, which can display the Home Assistant dashboard in an ePaper-friendly manner. The remaining systems only support this feature through third-party solutions. Domoticz has the "Dashticz" third-party dashboard that could be user on ePaper devices with a browser. For openHAB there is a third-party project called PaPiRus-MQTT, which uses the lightweight message queue MQTT to transmit the data from openHAB to the ePaper device. For ioBroker a solution suggested in the community forums is to use RPC calls and a Homematic display.

F1.2 (Dashboard): All systems have native support for dashboards to display data and to control devices. Both Home Assistant and ioBroker make use of the Lovelace UI. All systems also support third-party dashboards. For example, by connecting to an InfluxDB time-series database and using the Grafana visualization platform.

F2.1 (GPS-Tracking): Home Assistant includes a native companion app that can be installed on mobile devices to enable detailed tracking information. No native support is provided by Domoticz, though there are workarounds to include GPS data or to use the GeoFence mobile app. openHab includes native add-ons that offer integration with applications such as OpenPaths and OwnTracks to provide detailed

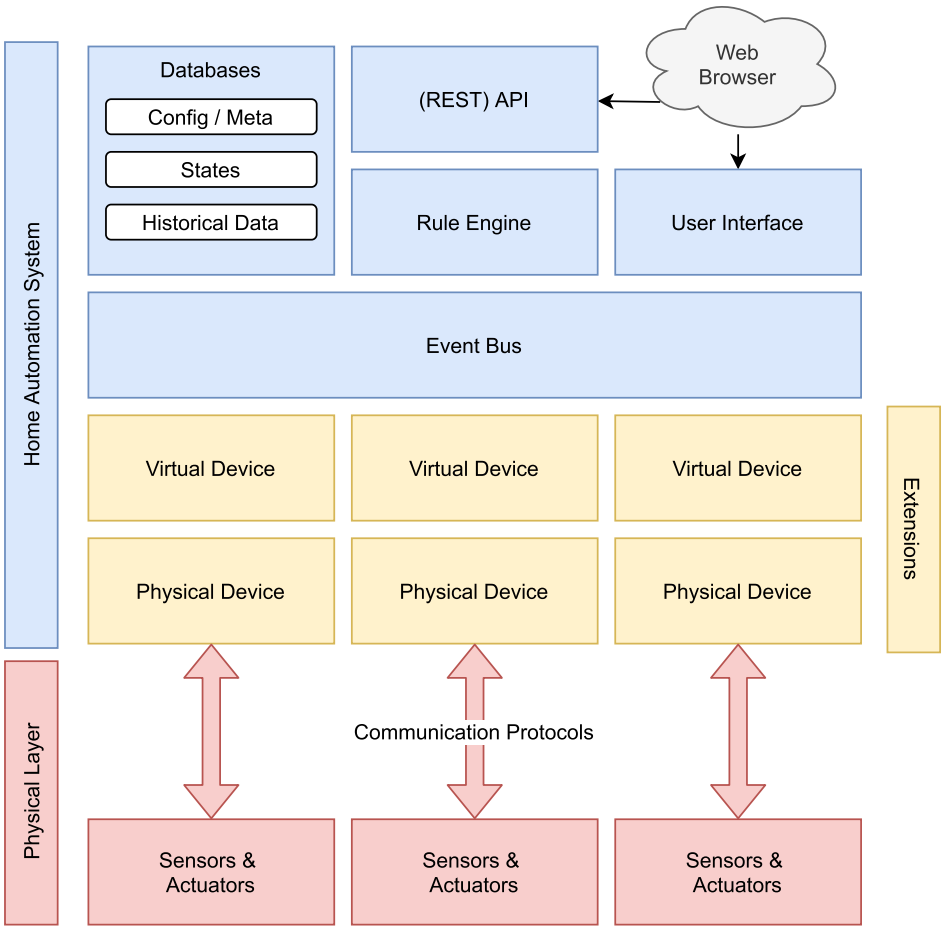

FIGURE 6. The generic home automation system.

tracking. For ioBroker, the community provides third-party adapters, such as ioBroker.places, which adds support for mobile apps such as OwnTracks.

F2.2 (Presence Detection): All systems provide support for detecting the users presence using IP-based approaches, tracking the presence of mobile devices in the home network. Additionally, Home Assistant, Domoticz, and openHab provide native support for Bluetooth Low Energy (BLE) beacons. Third party adapters exist for ioBroker to enable presence detection through BLE.

F3.1 (Mobile Notification): All systems support mobile notifications. Email notifications are supported on all systems, and addons are also provided for Telegram notifications. Push notifications are also supported on all systems.

F4.1 (Sensor – Read): All systems support the reading of sensor data from devices.

F4.2 (Device – Actuation): All systems support the actuation of devices.

F4.3 (Automation Rule): The creation of automation rules is an integral part of the dashboard provided by Home Assistant and openHAB. Domoticz and ioBroker rely on scripts written by the user, or the use of Blockly, a third-party visual programming editor.

F4.4 (Media Streaming): This criteria of Media Streaming is evaluated based on the support for the Real Time Streaming Protocol. Home Assistant, openHAB, and ioBroker have out-of-the-box support for it. Domoticz does not offer native support for the protocol, though third-party workarounds do exist.

F5.1 (Remote control): Home Assistant and openHAB provide official mobile applications, for both Android and iOS, that enable the remote control of devices connected to the system. ioBroker only offers an official iOS application, and a third-party Android application. Domoticz provides support for multiple third-party applications, such as Domoticz for Android, and ImperiHome.

F5.2 (External API Calls): All systems have support for external service REST API calls.

F5.3 (Scheduler): Home Assistant, openHAB, and ioBroker offer native solutions for time-based triggers and scheduling. Whereas, Domoticz relies on user-defined scripts or Blockly functionality.

F5.4 (Biometric User Authentication): The only system providing native support from biometric features is Home Assistant. There are official integrations available for Dlib Face Detect, Facebox, and Microsoft Face Detect, among others. Third-party projects also explain how to integrate fingerprint recognition. For both openHAB and ioBroker, third party projects are available that demonstrate the addition of face recognition. Domoticz offers no support for this feature.

TABLE 3. Features overview: ✓ (near-)native support, *O* 3rd-party support, – no support.

Features \ System	Home Assistant	Domoticz	openHAB	ioBroker
F1.1: ePaper / eInk Display	O	O	O	O
F1.2: Dashboard	✓	✓	✓	✓
F2.1: GPS-Tracking	✓	O	✓	O
F2.2: Presence Detection	✓	✓	✓	✓
F3.1: Mobile Notification	✓	✓	✓	✓
F4.1: Sensor – Read	✓	✓	✓	✓
F4.2: Device – Actuation	✓	✓	✓	✓
F4.3: Automation Rules	✓	✓	✓	✓
F4.4: Media Streaming	✓	O	✓	✓
F5.1: Mobile Remote Control	✓	O	✓	O
F5.2: External API Calls	✓	✓	✓	✓
F5.3: Scheduler	✓	✓	✓	✓
F5.4: Biometric User Auth.	✓	-	O	O
Score	4.5	2	4	3

VI. CRITERIA-BASED ANALYSIS

To complete the analysis of the system, after having studied the functionalities derived from uses cases, we proceed with checking the features identified in Section III-B. Each criteria is evaluated, and a score between 0 (not fulfilled) and 5 (completely fulfilled) is assigned to each system. The practical evaluation is performed on the previously deployed systems by young junior domain experts.

C1.1 (Activity): All four systems are actively maintained. The latest commit to the source code repository for each system was made on the 16th of July, 2021. As a result, the four systems receive the maximum score of 5 for this criterion.

C1.2 (Developer Popularity): Popularity is based on the Stargazers metric as defined by Jarczyk *et al.* [43]. The metric is defined as follows:

$$f(x) = log_{10}(x + 10) \tag{5}$$

where x is the number of stars a source code repository has received. Applying Equation 5 to each of the systems and rounding to the nearest integer value yields the results shown in Table 4.

C1.3 (Overall Popularity): The *Google Popularity Index* is used to measure overall popularity. Figure 7 shows the trend of the relative popularity index over time. The value represents the search interest relative to the highest point on the chart. A value of 100 represents the peak popularity. Scores are assigned based on the results of the latest relative popularity as is shown in Table 5, where the most popular system receives the highest score.

TABLE 4. Developer popularity.

System	Stars	Stargazers Metric	Score
Home Assistant	33511	4.53	5
Domoticz	2700	3.43	3
openHAB	297	2.49	2
ioBroker	684	2.84	3

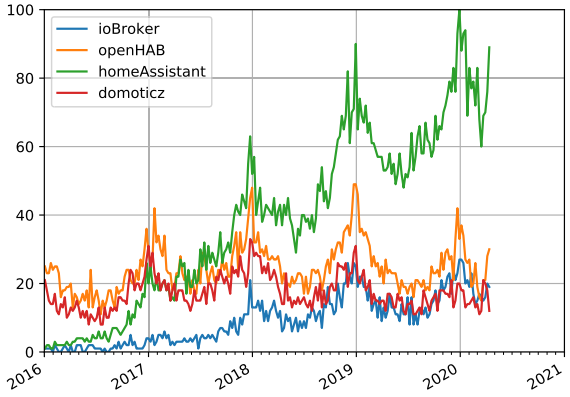

FIGURE 7. Google popularity index (2016 - April 2020).

TABLE 5. Relative popularity (2016 - April 2020).

System	Average	Gradient	Latest	Score
Home Assistant	38.92	37.09%	89	5
Domoticz	17.39	0.47%	12	2
openHAB	24.50	2.82%	30	4
ioBroker	9.35	9.39%	19	3

C2.1 (Support Plans): None of the four home automation system provide support plans. The support is provided by the communities in the shape of chat rooms and discussion boards. Therefore, for this category, all of the systems receive zero points.

C2.2 (Minimum Hardware Requirements): Single board computers such as the Raspberry Pi are popular hardware platforms for home automation systems. Not only do they have a small form factor, they also are one of the most affordable hardware options available for these systems. Table 7 shows the mapping between price and score. The minimum hardware requirements for each system, their price, and the score, are shown in Table 6. These prices are in accordance with what is expected in the market for 2021.

TABLE 6. Minimum hardware requirements.

System	Required Hardware	Price	Score
Home Assistant	Raspberry Pi 3 Model B	34,90 - 38,17 Euro	4
Domoticz	Raspberry Pi 1 Model B+	26,99 - 31,92 Euro	5
openHAB	Raspberry Pi 2 Model B	33,90 - 39,79 Euro	4
ioBroker	Raspberry Pi 2 Model B	33,90 - 39,79 Euro	4

TABLE 7. Price to score.

≤ 30 €	≤ 35 €	≤ 40 €	≤ 60 €	≤ 100 €	> 150 €
5	4	3	2	1	0

C2.3 (Recommended Hardware Requirements): Table 7 shows the mapping between price and score. The recommended hardware requirements for each system, their price, and the score, are derived and shown in Table 8. These prices are in accordance with what is expected in the market for 2021.

TABLE 8. Recommended hardware requirements.

System	Recommended Hardware	Criterion C2.2	Score
Home Assistant	Raspberry Pi 4 Model B	38,73 - 39,56 Euro	3
Domoticz	Raspberry Pi 3 Model B	34,90 - 38,17 Euro	4
openHAB	Raspberry Pi 2 Model B	33,90 - 39,79 Euro	4
ioBroker	Raspberry Pi 4 Model B	38,73 - 39,56 Euro	3

C3.1 (System Start-Up Time): Each home automation system provides Docker container images for deployment. Since Docker containers automate many of the installation steps, using these containers is generally the fastest way to setup a system. Therefore, the time to download and fully initialise these containers is measured. The results are shows in Table 9. Scores are assigned as follows, where time is measured in seconds: 5 points for ≤ 40, 4 points for ≤ 80, 3 points for ≤ 120, 2 points for ≤ 180, and 1 point for ≤ 300.

TABLE 9. System start-up time.

System	Mean Setup Time	Score
Home Assistant	85s	3
Domoticz	84s	3
openHAB	69s	4
ioBroker	94s	3

C3.2 (Basic Sensor Setup): To evaluate this criteria, a MQTT broker (Eclipse Mosquito) and a virtual MQTT sensor are set up using Docker containers. The virtual MQTT sensor generates random sensor data. The number of required steps for each system to integrate this MQTT-based sensor is measured. The results, shown in Table 10 show that setting up the sensors requires a comparable number of steps, the minimum is three for Domoticz and the maximum is 11 for ioBroker.

TABLE 10. Interaction steps to setup an MQTT sensor.

System	# Interaction steps	Score
Home Assistant	5	4
Domoticz	3	5
openHAB	8	3
ioBroker	11	2

C4.1 (Effort): To measure the effort, we consider the implementation of the following task from the Smart Home Scenario data set is measured: *"During the day, whenever I walked into the bathroom the light would come on. However, after a certain time, when it is night, when I walk in the bathroom I would want a much softer light to come on"* [14]. All systems are tested with virtual sensors and lighting fixtures, which are set-up ahead of time. The automation rule creation of Domoticz is tested through its integrated rule engine Blockly.[1] The creation of the automation rule and the corresponding scene takes 43 clicks using Domoticz. In turn, openHAB offers an experimental UI-based rule engine [44], which is used in this test. Overall, it takes 70 clicks to install the rule engine and create the necessary rules in openHAB. Home Assistant comes with a built-in automation rule editor, which is used in this test, taking 38 clicks in total to setup the rules to fulfil the scenario. Due to an unexpected error within a required module of ioBroker, the score for this system cannot be evaluated. Table 11 provides an overview of the results and the scoring, which is assigned by rank.

TABLE 11. Effort of creating the automation rules for the user story.

System	# clicks	Score
Home Assistant	38	5
Domoticz	43	4
openHAB	70	3
ioBroker	-	0

C4.2 (Task Time): The scenario and tasks used to evaluate the Task Time criteria are identical to *41 (Effort)*. The same setup is also used. In our evaluation, the time to complete the task takes 1:39 minutes with Domoticz, 2:16 minutes with openHAB, and 1:00 minute with Home Assistant. Because of an error with a required module, ioBroker cannot be evaluated. An overview of the test result is shown in Table 12.

C4.3 (Extensibility): In terms of extensibility, Home Assistant has 25 official widgets available for its Lovelace UI. Panels can also be customized using ReactJS. Domoticz does not have built-in widgets. There are ways the dashboard can be customized, but they are quite limited, such as applying skins and modifying icons. Domoticz can be used with a

[1] https://developers.google.com/blockly (June 21, 2020)

TABLE 12. Completion time of the user story.

System	Duration	Score
Home Assistant	60s	5
Domoticz	99s	4
openHAB	136s	3
ioBroker	-	0

number of third-party dashboards, including Reacticz, New Frontpage, and Dashticz. openHAB has 13 built-in widgets. It also allows for inclusion of custom widgets, which can then be accessed through the widget gallery. The fourth and final system, ioBroker, has a considerable number of third party widgets, that can be imported into the system. The widgets can also be customized further by changing the CSS attributes.

Home Assistant and openHAB both provide official UI widgets and other means to customise the UI, therefore they receive the maximum score of 5. The other two systems, Domoticz and ioBroker, appear to only support third party widgets, thus receiving a score of 3.

C4.4 (Responsiveness): All home automation systems provide a UI that is responsive, ensuring that it functions on smartphones, tablets, and other mobile devices. However, the UI of Domoticz is lacking with regards to its responsiveness, for example, the automation rule editor is borderline unusable on a mobile device as it does not scale well. Therefore, two points are deducted from its score. To conclude, Home Assistant, Domoticz and openHAB receive full marks, and Domoticz has a score of 3.

C5.1 (User Authentication): Domoticz, openHAB, and ioBroker only offer basic-auth as a login method. Home Assistant is the only system which provides webauthn as an additional login method. In addition, Home Assistant supports Multi-Factor Authentication and 2-Factor Authentication for self-hosted solutions. The other systems only support 2FA for their online cloud-hosted solutions. Table 13 shows the results. Basic-auth yields a score of 3, additional points are given for webauthn support, as well as MFA/2FA support.

TABLE 13. Protected settings by authentication.

System	basic-auth	MFA/2FA	webauthn	Score
Home Assistant	✓	✓	✓	5
Domoticz	✓	(✓)		3
openHAB	✓	(✓)		3
ioBroker	✓	(✓)		3

C5.2 (Multiple User Accounts): Multiple user accounts are either supported (5 points), or not (0 points). All four systems support multiple user accounts, therefore receiving the full score of 5.

C5.3 (Authorization management): Both Domoticz and ioBroker provide some form of authorization management. Domoticz supports a single admin user and multiple regular users, who can be invited by the admin. ioBroker has a similar authentication concept to Domoticz, there is one admin user and there can be multiple regular users. However, in ioBroker, users be separated into different user groups with different permissions. Currently, Home Assistant does not provide authorization management. Thus, all users have the same level of access. openHAB also does not provide any kind of authorization management. ioBroker receives a score of 5, as it provides the most comprehensive authorization management. Domoticz receives a score of 3, as the features are more limited. Home Assistant and openHAB do not currently have authorization management and therefore receive a score of 0.

C6.1 (Custom Extensions): All four home automation systems provide the means for their systems to be extended with additional functionality using custom extensions or plugins. One thing of note is that ioBroker, unlike the other systems, does not provide a well documented framework for implementing extensions. As all systems support custom extensions, they all receive the full score of 5 points.

C6.2 (Extension Count): The number of available extensions for each home automation system is obtained by analysing official wiki's and available documentation, as well as plugin repositories or marketplaces, when available. An overview of the number of available extensions for each system can be found in Table 14. Score is assigned in accordance with the system ranking.

TABLE 14. Extension count.

System	C6.2	Score
Home Assistant	1611	5
Domoticz	79	2
openHAB	323	3
ioBroker	352	4

C6.3 (Quality of Documentation): The quality of documentation is defined by the following sub-criteria: 6(3)1 Actuality, 6(3)2 Completeness / comprehensiveness, 6(3)3 Examples), 6(3)4 Findability, 6(3)5 Readability, and 6(3)6 Skimmability. The overall quality of the documentation is determined as the average over all sub-criteria. The final scores are presented in Table 15. What follows next is a more detailed analysis for each of the sub-criteria.

TABLE 15. Quality of documentation.

System	C6.3.1	C6.3.2	C6.3.3	C6.3.4	C6.3.5	C6.3.6	Score
Home Assistant	5	5	5	5	5	4	5
Domoticz	5	0	3	5	5	4	4
openHAB	5	3	0	5	5	3	4
ioBroker	5	0	0	5	5	4	3

C6.3.1 (Actuality): The actuality of the documentation is determined by the date of the last change. This data was

gathered either from the documentation of the system, or from its source code repository. At the time of writing, each of the four system had their documentation updated within the last two days, and therefore they all receive 5 points.

C6.3.2 (Completeness / Comprehensiveness): The completeness and comprehensiveness of the documentation is determined by how often other sources besides the official documentation had to be consulted for the analysis in this work. These other sources may include forums and chat rooms, or external websites. The scores are assigned as follows: 5 for a complete and comprehensive documentation and rarely needing other sources, 3 for a mostly complete documentation where other sources had to be consulted multiple times, and 0 in case there is no documentation or external sources had to be primarily used to perform the analysis in this work. Home Assistant scores full marks for its very complete documentation, openHAB scores 3 points for sufficient but incomplete documentation. Both Domoticz and ioBroker are lacking critical information in their documentation, requiring external sources to be accessed regularly and therefore receive a score of 0.

C6.3.3 (Examples): Home Assistant provides a separate section dedicated to examples of many different use cases. Therefore, it receives the full score for the availability of examples criterion. Domoticz offers a manual that shows how many of its features can be used, individually. Though, it does not provide complete example use cases. Therefore, Domoticz receives a score of 3. Both openHAB and ioBroker do not provide any examples use cases in their documentation, and therefore receive a score of 0.

C6.3.4 (Findability): Since all of the home automation systems have made it simple to locate the documentation, either through links on their websites or via the source code repositories, each of them receives the full score.

C6.3.5 (Readability): Each of the home automation systems use clear terms for the description of their system and features. It is interesting to note that the same concepts have different terms in different systems, but the terms are well defined. Despite this, it is not difficult to understand the similarities between the systems. Therefore, each of the four systems receives a score of 5 for this criterion.

C6.3.6 (Skimmability): To determine skimmability, each system's documentation is analyzed in terms of: (1) informative headlines, (2) short paragraphs, (3) table of content, (4) global index, and (5) glossary . If an item is partially or not available, then 1 point is deducted from the score. The results are shown in Table 16.

TABLE 16. Quality of Documentation - Skimmable.

System	(1)	(2)	(3)	(4)	(5)	Subscore
Home Assistant	✓	✓	-	✓	✓	4
Domoticz	✓	✓	✓	✓	-	4
openHAB	✓	✓	-	✓	-	3
ioBroker	✓	✓	✓	✓	-	4

C6.4 (Variety of Support): Each home automation system has some presence on social media for support. Home Assistant is on Twitter, Facebook, and Reddit, as well as Discord. Domoticz is on Facebook, Reddit, and Twitter, though the Twitter account appears to be inactive. openHAB is on Twitter, Facebook, Reddit, and YouTube. Though, both Facebook and YouTube accounts appear to be inactive. ioBroker is on Twitter, Facebook, and Reddit. In this case, the Twitter account appears to be inactive. When it comes to email support, openHAB and ioBroker provide an email address, Home Assistant does too but not for support purposes, and Domoticz has no email contact details. Each of the four systems has a forum available for support. The final score can be found in Table 17.

TABLE 17. Variety of support.

System	Social Media	Forum	Email	Score
Home Assistant	✓	✓	-	3
Domoticz	✓	✓	-	3
openHAB	✓	✓	✓	5
ioBroker	✓	✓	✓	5

C7.1 (Concurrency): To evaluate the concurrency criterion, a virtual MQTT sensor is created and connected to each of the home automation systems. The sensor updates at a frequency of one message per second. The number of virtual sensors is gradually increased until noticeable performance issues arise. These performance issues can include, but are not limited to, out-of-order message processing, data inconsistency, high system response time, UI freezes, or system crashes. The systems are deployed on a Raspberry Pi 3 Model B+ using the preconfigured Docker containers that each system provides. The virtual sensors and MQTT broker are deployed on separate machines.

When testing, ioBroker was able to handle at least 100 concurrent sensors. However, at this point problems do arise when displaying the real-time logs of the incoming sensor data, which becomes notably slow until the page is refreshed. Domoticz was able to handle up to 10 concurrent sensors, at which point the processing of messages was notably delayed, and the order in which the messages were processed became inconsistent. Home Assistant was able to process the data from 70 concurrent sensors. When increasing beyond this number of sensors, Home Assistant is unable to process the messages on time, and the visual representation of the data will be delayed. Unfortunately, openHAB does not support this automated test bench, and each virtual sensor has to be manually configured. Therefore, only 10 sensors were set up. These manual tests indicated that openHAB was able to handle at least 10 concurrent sensors. Table 18 shows the results, the score is assigned in accordance with the ranking.

C7.2 (Update Rates): To evaluate update rates, the same setup as in 71 is used, with the difference that there is only one virtual MQTT sensor. However, this single virtual

TABLE 18. Maximum concurrent sensors.

System	C7.1	Score
Home Assistant	70	4
Domoticz	10	2
openHAB	10+	3
ioBroker	100+	5

MQTT sensor is able to transmit data at varying rates: 1, 10, and 100 messages per second. Tests show that all of the systems are able to handle a single sensor with update rates of up to 100 messages per second. The results are shown in Table 19.

TABLE 19. Maximum update rates.

System	1/s	10/s	100/s	Score
Home Assistant	✓	✓	✓	5
Domoticz	✓	✓	✓	5
openHAB	✓	✓	✓	5
ioBroker	✓	✓	✓	5

C7.3 (Scalability): Two of the four home automation systems that were examined offer horizontal scaling. ioBroker supports a multi-host-mode setup, which makes it possible for multiple instances to operate in parallel. And openHAB supports a special version called openHAB-cloud, that enables off-loading to the cloud or a local cluster. Both ioBroker and openHAB receive a score of 5, whereas Home Assistant and Domoticz receive 0 points due to the lack of scalability.

C8.1 (Code reviews): Each of the home automation systems requires code reviews before a pull or merge requests is accepted. Home Assistant and openHAB provide a contribution document, in which the process to contribute to the source code is detailed. Domoticz and ioBroker are less restrictive, but do enforce code reviews. Therefore, all systems receive the full score.

C8.2 (End-to-End Test Metric): None of the four home automation systems provides end-to-end tests, neither in their main source code repositories, or their UI-specific source code repositories. Therefore, a score of 0 is assigned to each of the systems.

C8.3 (Formal Code Metric): The maintainability of the source code is critical for the longevity of the project. Each of the home automation systems provides a core source code repository which contains the code for the backend of the system. To determine the size of the backend, the lines of code are determined for each of the core repositories. Scores are assigned in accordance with the lines of code, where fewer lines of code equates a higher score. The results are shown in Table 20

C8.4 (Pipeline Support): Each of the home automation systems uses some form of CI/CD pipeline to automate the software engineering process such as testing, and the creation of Docker images. All of the systems use either Travis or GitHub workflows for this purpose. To conclude, each system receives 5 points for this criterion.

TABLE 20. Lines of code of the core repository.

System	LOC	Score
Home Assistant	542013	2
Domoticz	208295	3
openHAB	127745	4
ioBroker	37017	5

C8.5 (Quality checks for 3rd party plugins): Home Assistant provides an elaborate quality scale to determine the quality of 3rd party plugins [45]. The focus of Domoticz is more on the technical specification of the plugins, rather than on the process that developers have to follow. openHAb on the other hand, performs static code analysis, and provides a detailed list of requirements for contributions [46]. ioBroker ensures that all 3rd party plugins are checked before they are made available in their adapter repository. Additionally, there is a forum on which developers search for members to evaluate the new plugins, and verify that they are functioning. The final scores are as follows: Home Assistant, openHAB, and ioBroker receive full marks for their elaborate checks. Whereas Domoticz receives 3 points as their quality checks are significantly less elaborate.

C8.6 (Unit test metric): Home Assistant makes use of unit tests, though the available information on how they should be written is limited. openHAB provides a detailed explanation about how the unit tests and integration tests should be coded, highlighting the importance of these tests in the project. ioBroker also provides comprehensive documentation on the practices that should be used when writing tests. Domoticz on the contrary does not have unit tests. The three systems that do have unit tests receive a score of 5, whereas Domoticz receives a score of 0.

VII. DISCUSSION

The comparison of the systems on the basis of the use cases and features has highlighted many similarities among the currently available open source projects, but also some key differences. What follows is a discussion of the findings of the previous sections in terms of use cases realizability and feature possession.

A. USE CASE BASED ANALYSIS (F1-F5)

The use case analysis has highlighted Home Assistant is the highest scoring system, followed closely by openHAB; ioBroker is in third place, and Domoticz has the lowest score, as summarized in Table 3. Each system supports the basic operations that one may expect from a home automation system, such as reading sensor data, sending commands to actuate devices, and creating automation rules. Though, none of the systems have native support for ePaper and eInk displays. That said, almost all functionality that is not supported natively can be provided by third party extensions or

plugins. This highlights the strong points of open-source and extensible home automation systems, which is extensibility and the ability for anyone to contribute to the project. It is of note to mention that the use cases were defined before any practical evaluation, and that the use cases are considered to be comprehensive for smart buildings, for both residential and non-residential buildings. The fact that all systems are able to support every use case, either natively or through extension, demonstrates the maturity of these systems.

B. POPULARITY AND COMMUNITY (C1)

The popularity among developers, as well as users, are an important factor to consider when choosing a system, as these factors influence the available support and longevity of the project. Table 21 summarises the results of the Popularity and Community category, which consists of three individual criteria. While all systems are actively developed by their community of developers, there is a clear difference in overall popularity. Home Assistant takes the lead in both Developer Popularity, as well as Overall Popularity. Interestingly, while openHAB appears to be the least popular among developers, their overall popularity is second highest.

TABLE 21. Popularity and community score.

Criterion	Home Assistant	Domoticz	openHAB	ioBroker
C1.1	5	5	5	5
C1.2	5	3	2	3
C1.3	5	2	4	3
Average	5	3.3	3.7	3.3
Rank	**1st** (5.0)	**3th** (3.3)	**2nd** (3.75)	**3rd** (3.3)

C. PRICING (C2)

The upfront costs can be a significant hurdle when first purchasing a home automation system. Deciding on an open source solution instead of a commercial system can help drastically reduce them, while sacrificing paid support. Table 22 provides an overview of the scores related to the Pricing category, considering both support plans and hardware costs. First of all, none of the systems provides paid support plans. For any kind of assistance, one has to rely on the community of developers and users. While commercially available closed-source home automation systems are expensive (€250-€600), the hardware required to deploy the open source systems considered in this work are significantly cheaper. All four systems can be deployed on the Raspberry Pi single board computer. The minimum and recommended hardware requirements are nearly identical for all systems. Note that we excluded the cost of the power supply and the non-volatile memory card, as these costs are identical for all four systems.

D. SETUP COMPLEXITY (C3)

The complexity of a system is an important aspect to consider, while uncomplicated systems make it easy to connect hardware to the system, a more complex system may offer more

TABLE 22. Pricing score.

Criterion	Home Assistant	Domoticz	openHAB	ioBroker
C2.1	0	0	0	0
C2.2	4	5	4	4
C2.3	3	4	4	3
Average	2.3	3	2.6	2.3
Rank	**2nd** (2.3)	**1st** (3.0)	**4nd** (2.6)	**2nd** (2.3)

options and functionality that can be configured. Table 23 presents the results of the Setup Complexity category. While all systems are easy and quick to start from a fresh setup, thanks to Docker support, there are still some differences in the time it takes to start the Docker container. openHAB starts the fastest, in 69 seconds. Whereas ioBroker takes the longest, with 94 seconds. A bigger discrepancy can be found in the number of steps it takes to setup a basic MQTT sensor. In Domoticz this only takes 3 steps, whereas ioBroker requires 11 steps.

TABLE 23. Setup complexity score.

Criterion	Home Assistant	Domoticz	openHAB	ioBroker
C3.1	3	3	4	3
C3.2	4	5	3	2
Average	3.5	4	3.5	2.5
Rank	**2th** (3.5)	**1st** (4.0)	**2nd** (3.5)	**4rd** (2.5)

E. USER INTERFACE AND USER EXPERIENCE (C4)

Many of the interactions with the home automation system take place through the web interface provided. Therefore, the experience and functionality that the interface offers are important when deciding which system to adopt. Table 24 shows the results for the User Interface and User Experience categories. While Home Assistant and Domoticz require nearly the same number of clicks to setup an automation rule (around 40), openHAB requires nearly double that amount (70 clicks). The time it takes to setup an automation rule also widely differs, from 60 seconds for Home Assistant, to 136 seconds for openHAB. When it comes to extensibility of the UI, Home Assistant and openHAB support this functionality natively, whereas the other two systems have third party support for extending the UI. Domoticz is the only system which has a number of issues with its responsive design, making it less prone to be used on mobile and handheld devices. Unfortunately, 41 and 42 could not be evaluated for ioBroker, due to an error in one of the modules.

F. SECURITY AND AUTHENTICATION (C5)

Home automation systems deal highly sensitive data in terms of privacy, which should be managed with care. Table 25 presents the results of the Security and Authentication category, which consists of three criteria. All four systems provide support for basic-auth. Only Home Assistant provides MFA/2FA and webauthn support for their self-hosted

TABLE 24. UI and UX score.

Criterion	Home Assistant	Domoticz	openHAB	ioBroker
C4.1	5	4	3	N/A
C4.2	5	4	3	N/A
C4.3	5	3	5	3
C4.4	5	3	5	5
Average	5	3.75	4	2
Rank	**1st** (5.0)	**3rd** (3.75)	**2nd** (4.0)	**4th** (2.0)

solution. The other systems provide MFA/2FA only in their cloud-hosted solutions. Perhaps unsurprisingly, all four systems have support for multiple user accounts. Authorization management is in general lacking; in most systems there is either only two predefined roles (user and admin), or just one single role that provides everyone with the same level of access. The main exception here is ioBroker, which provides the most detailed level of control over permissions.

TABLE 25. Security and authentication score.

Criterion	Home Assistant	Domoticz	openHAB	ioBroker
C5.1	5	3	3	3
C5.2	5	5	5	5
C5.3	0	3	0	5
Average	3.3	3.6	2.6	4.3
Rank	**3rd** (3.3)	**2nd** (3.6)	**4th** (2.6)	**1st** (4.3)

G. EXTENSIBILITY AND SUPPORT (C6)

A home automation system should provide support for as many devices and protocols as possible. Though in practice not every system supports every device or protocol. While popular protocols, such as Z-Wave and ZigBee, are widely supported, more obscure or niche devices and protocols may not be supported. Table 26 summarises the result for the Extensibility and Support Category. While all systems provide support for extensions, the number of available extensions varies widely. Home Assistant has over 1.600 extensions available, whereas Domoticz has about 80. The quality of the documentation also varies widely, with Home Assistant having excellent documentation, while ioBroker's documentation is clearly lacking, especially when it comes to completeness and the inclusion of examples. All of the systems rely heavily on the community to provide support to users. The support is often provided via forums or social media.

H. SYSTEM PERFORMANCE (C7)

Current trends show that there is an increase in the number of devices and appliances that can be connected to home automation systems [47]. Therefore, it is important that the system is able to scale to support this increasing number of home smart devices. Table 27 provides an overview of the results for the System Performance category. The results

TABLE 26. Extensibility and support score.

Criterion	Home Assistant	Domoticz	openHAB	ioBroker
C6.1	5	5	5	5
C6.2	5	2	3	4
C6.3	5	4	4	3
C6.4	3	3	5	5
Average	4.5	3.5	4.3	4.3
Rank	**1st** (4.5)	**4rd** (3.5)	**2st** (4.3)	**2nd** (4.3)

show that ioBroker is best suited to handle many sensors working concurrently. Domoticz appears to struggle with more than 10. All systems support at least up to 100 messages per second from a single sensor. Horizontal scalability is lacking in Home Assistant and Domoticz, whereas both ioBroker and openHAB support it.

TABLE 27. System performance score.

Criterion	Home Assistant	Domoticz	openHAB	ioBroker
C7.1	4	2	3	5
C7.2	5	5	5	5
C7.3	0	0	5	5
Average	3	2.3	4.3	5
Rank	**3rd** (3.0)	**4th** (2.3)	**2nd** (4.3)	**1st** (5.0)

I. SOFTWARE QUALITY (C8)

The quality of software is not trivial to quantify, however, there are indicators to the quality such as the presence of tests (unit, e2e, etc.), lines of code, and the general processes and checks in place when contributing to a project. Table 28 summarises the results for the Software Quality category. Source code reviews are present for all four systems. Interestingly, none of the systems performs end-to-end testing, which could benefit the overall user experience. Though unit tests are performed by all systems with the exception of Domoticz. ioBroker's lines of code are significantly less than for the other systems, with Home Assistant having by far the most lines of code. Quality checks are also in place for third party plugins, though Domoticz's checks are not as elaborate as the checks that are in place for the other systems.

TABLE 28. System software quality score.

Criterion	Home Assistant	Domoticz	openHAB	ioBroker
C8.1	5	5	5	5
C8.2	0	0	0	0
C8.3	2	3	4	5
C8.4	5	5	5	5
C8.5	5	3	5	5
C8.6	5	0	5	5
Average	3.7	2.7	4	4.2
Rank	**3rd** (3.7)	**4th** (2.7)	**2nd** (4.0)	**1st** (4.2)

J. DISCUSSION AND LIMITATIONS

While at first glance all four systems appear to offer very similar functionalities, as deduced from the use case based

TABLE 29. Overview of all categories.

Criterion	Home Assistant	Domoticz	openHAB	ioBroker
F1-5	4.5	2	4	3
C1	5	3.3	3.75	3.3
C2	2.3	3	2.6	2.3
C3	3.5	4.0	3.5	2.5
C4	5	3.75	4	2
C5	3.3	3.6	2.6	4.3
C6	4.5	3.5	4.3	4.3
C7	3.0	2.3	4.3	5
C8	3.7	2.7	4	4.2
Average	3.9	3.1	3.7	3.4
Rank	**1st** (3.9)	**4th** (3.1)	**2nd** (3.7)	**3rd** (3.4)

analysis, on closer inspection, several differences emerge. These differences are visualised in Table 29, which presents an overview of the satisfaction of the criteria for each one of the top four systems. The results show quite some variance for each criteria going from two to five stars for some of them, e.g., for criteria C7. This indicates that the systems are not equivalent and making an informed choice when selecting one will have consequences for the effort and success of a home automation project. Furthermore, the importance of each criteria may differ based on the desired application; some deployments may desire high performance, while others value the support of a wide range of devices. Therefore, the criteria are not weighted: this should be done on a per-deployment basis.

A limitation of the present work is that the results obtained in the analysis are a snapshot of a specific moment in time. The code base of these systems, especially for open source projects, is highly volatile and ever changing as many contributors are continuously making changes to the source code on a daily basis. That said, the underlying architecture of the systems is fundamental and therefore not as susceptible to continuous changes. We therefore do not expect the common architecture that emerges from the four systems to be subjected to significant changes in the near future.

As the systems continuously change, it is challenging to obtain a snapshot of critical privacy and security vulnerabilities. Furthermore, these vulnerabilities are also strongly influenced by the deployment environment: from the connected IoT hardware, to the configuration of the operating system on which the software is installed. Therefore, the analysis of vulnerabilities is not included in this work.

VIII. CONCLUSION AND OPEN PERSPECTIVES

The selection of available home automation systems is vast, making an informed decision when designing a smart home is difficult. Not only because the available systems offer different functionalities, but also because retroactively migrating from one system to another one is not a trivial task. This means that the expected longevity of the open source project is of importance too. In this work, these concerns are addressed by providing an overview of the 20 most-known open source home automation systems, and investigating the four most relevant ones in greater detail.

Of these four systems, the architecture is analyzed and seven distinct components that all systems have in common are identified and discussed. We deem these components to be essential parts of a modern home automation system architecture. Furthermore, the use case based analysis shows that almost all systems support the list of thirteen essential features. The support is either native or through third party extensions, highlighting the importance of an extensible home automation platform. A further analysis is done based on thirtyfour criteria. This criteria-based analysis reveals the strengths and weaknesses of the selected systems. With Home Assistant presenting a very solid user experience, Domoticz providing an uncomplicated setup, openHAB having strong scores in all categories, and ioBroker offering great system performance. It is for the practitioner or requirements engineer to assign the appropriate weights to each criteria, as each deployment will have distinct requirements. Significant shortcomings among the systems were also identified, and include: the lack of role-based access control, no horizontal scalability, and no options for enterprise or paid support. Finally, we have also described the common architecture that emerges from the design of the four systems, identifying the key components of which a home automation system is comprised.

The present work serves several purposes. The basic service is that of inventorising existing open source home automation systems. The second one is to provide meaningful features to actually evaluate these kind of systems. The features and methodology adopted may also be useful for research in other related fields beyond home automation systems. The final, and perhaps, more relevant service is that of acting as a decision tool for the practitioner and for the developer who intend to deploy and contribute to open source home automation systems.

APPENDIX A
DESCRIPTION OF FEATURES

What follows is the list of features and their descriptions. These features were extracted from 17 different uses cases. Each feature is categorized in one of 5 categories.

- F1 **Visualization**
  - F1.1 *ePaper/eInk*: allows the user to employ a paper-white display in a static way. Most commonly used for low resolution wall displays.
  - F2.2 *Dashboard*: allows the user to create customised dashboards, which can display a lot of live information (e.g. news feeds, states of other sensors). Also provides the user with control of various devices from a single location.
- F2 **Localization**
  - F2.1 *GPS-Tracking*: enables the system to track the location of the user using GPS coordinates.

F2.2 *Presence Detection*: detects the presence of a human in an area of interest. Typically realised with Bluetooth Low Energy beacons, sometimes also realised with motion sensors. This includes geo-fencing.

F3 **Notification**

F3.1 *Mobile Notification*: includes all types of notifications that can be received on a mobile device, for example: e-mail, SMS, Telegram, WhatsApp, push-notifications, and voice calls.

F4 **Data-Handling**

F4.1 *Sensor – Read*: forms the foundation of a smart home, allows the system to collect data from Internet of Things devices.

F4.2 *Device – Actuation*: forms the foundation of a smart home, allows the system to actuate Internet of Things devices.

F4.3 *Automation rule*: transforms the "Internet of Things home" into a smart home. Rules can be defined which react to certain events.

F4.4 *Media Streaming*: allows the user to video and audio streams. It can be used for both surveillance cameras and baby monitors.

F5 **Interaction**

F5.1 *Mobile Remote Control*: extends feature 42, to enable of any Internet of Things device connected to the system using a mobile application.

F5.2 *External API Calls*: enables the system to use services on the Internet. For example to show weather information to the user or interact with an external calendar.

F5.3 *Scheduler*: allows a user to schedule an action at a specific time. Also enables the creation of recurring events.

F5.4 *Biometric User Authentication*: enables the user to use biometric authentication. It can be used for private information on dashboards via "face id" or fingerprint for door lock access control.

APPENDIX B
DESCRIPTION OF CRITERIA

What follows is the list of criteria and their descriptions. Each criteria is categorized in one of 8 categories.

C1 **Popularity and Community**

C1.1 *Activity:* is the system actively maintained? This is determined by the time of the latest commit to the core repository.

C2.2 *Developer Popularity:* how popular is the system amongst developers? This is measured using the Stargazers Metric as defined by Jarczyk *et al.* [43]. The metric makes use of the number of stars that a source code repository has received.

C3.3 *Overall Popularity:* what is the popularity index of the system? The popularity index is determined using Google Trends.[2] The name of the system is used as a search term. The mean popularity over the last 24 months is used to quantify this criteria.

C2 **Pricing**

C2.1 *Support Plans:* are there support plans available, and what is their cost? The cheapest support plan is considered.

C2.2 *Minimum Hardware Requirements:* what are the minimum hardware requirements? And is the cost associated with these requirement? Prices are gathered from German retailers.

C2.3 *Recommended Hardware Requirements:* what are the recommended hardware requirements? And is the cost associated with these requirement? Prices are gathered from German retailers.

C3 **Setup Complexity**

C3.1 *System Start-Up Time:* how much time does it take to download and install the system? Only fresh installations are considered. The time is measured from download until the moment the system user interface becomes responsive.

C3.2 *Basic Sensor Setup:* how many steps does it take to setup the system and add a MQTT based sensor? After the completion of all the steps, the sensor data should be in the desired format, ready to be displayed and used. A step in this context is not individual clicks or keystrokes, but a significant step in the overall process. For example, opening the plug-in marketplace.

C4 **User Interface and User Experience**

These criteria will be measured by means of a self-review in the context of a workbench use-case experiment. The metrics defined by Sauro *et al.* [48] are used.

C4.1 *Effort:* how much effort is needed to create the required automation rules to fulfil a scenario? The scenario is taken from the Smart Home Scenarios data set [14]. The results are measured by the number of clicks necessary to finish the tasks.

C4.2 *Task Time:* how much time does the user need to complete the task from 41? The time required to complete all tasks is measured.

C4.3 *Extensibility:* does the system allow for extension of the user interface? The number of available user interface widgets in the source code repository or on the marketplace are measured.

C4.4 *Responsiveness:* does the system provide a user interface that adheres to the responsive web design principles? Responsiveness is the ability to correctly render the user interface on a variety of devices with different display sizes, from smartphones and tablets to 4K TV's.

[2]https://trends.google.com/trends/

C5 **Security and Authentication**
- C5.1 *User Authentication:* is there any form of user authentication available? If so, what type of user authentication is supported?
- C5.2 *Multiple User Accounts:* is it possible to create multiple user accounts within the system?
- C5.3 *Authorization Management:* does the system provide *role or attribute based access control*, to limit and control a user's permission?

C6 **Extensibility and Support**
- C6.1 *Custom Extensions:* does the system provide the possibility to implement custom plug-ins?
- C6.2 *Extension Count:* how many extensions are available that extend the system to enable support for different IoT devices and protocols? Other types of extensions, such as user interface extensions, are excluded.
- C6.3 *Quality of Documentation:* the quality of the documentation is defined by six sub-criteria, adapted from [49]–[51].
  - C6.3.1 *Actuality:* does the system provide up-to-date documentation? The date of the most recent change to the documentation is taken.
  - C6.3.2 *Completeness and Comprehensiveness*: does the documentation cover all aspects of the system?
  - C6.3.3 *Examples:* does the documentation include examples on standard techniques to use the system.
  - C6.3.4 *Findability:* how easy is it to find the documentation? Is the documentation referenced on the homepage or the source code repository of the system?
  - C6.3.5 *Readability:* does the documentation use clear terms for describing the system and its features?
  - C6.3.6 *Skimmable:* is the documentation made to skim through quickly? The following sub-criteria are investigated: (1) informative headlines, (2) short paragraphs, (3) table of content, (4) global index, and (5) glossary
- C6.4 *Variety of Support:* does the system provide a variety of different support methods? This is determined by the availability of e-mail support, forum support, and social-media support.

C7 **System Performance**
- C7.1 *Concurrency:* how many concurrent MQTT-sensors are supported?
- C7.2 *Update Rates:* how many updates per second can the system handle without performance drops?
- C7.3 *Scalability:* does the system support horizontal scaling?

C8 **Software Quality**
- C8.1 *Code reviews:* does the system require code reviews before new features are added to the codebase?
- C8.2 *End-to-end Test Metric:* does the system have end-to-end tests?
- C8.3 *Formal Code Metric:* how many lines of code does the source code of the system have?
- C8.4 *Pipeline Support:* does the system use Continuous Integration / Continuous Delivery pipelines?
- C8.5 *Quality Checks for Third-Party Plugins:* does the system guarantee a certain level of quality of third-party plug-ins added to the repositories?
- C8.6 *Unit Test Metric:* what is the unit test coverage level?

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.
[2] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Bus. Horizons*, vol. 58, no. 4, pp. 431–440, 2015.
[3] A. ElShafee and K. A. Hamed, "Design and implementation of a WIFI based home automation system," *World Acad. Sci., Eng. Technol.*, vol. 68, pp. 2177–2180, Aug. 2012.
[4] M. Aiello and S. Dustdar, "Are our Homes ready for services? A domotic infrastructure based on the web service stack," *Pervas. Mobile Comput.*, vol. 4, no. 4, pp. 506–525, Aug. 2008.
[5] M. Aiello, M. Zanoni, and A. Zolet, "Exploring web-service notification: Building a scalable domotic infrastructure," *Dr. Dobb's J., Softw. Tools Prof. Developer*, vol. 371, pp. 48–51, Oct. 2005.
[6] A. J. B. Brush, B. Lee, R. Mahajan, S. Agarwal, S. Saroiu, and C. Dixon, "Home automation in the wild," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, May 2011, p. 2115.
[7] V. Vukovic and L. Rakovic, "Open source approach in software development—Advantages and disadvantages," *Int. Sci. J. Manage. Inf. Syst.*, vol. 3, pp. 29–33, Dec. 2008.
[8] O. Taiwo, L. A. Gabralla, and A. E. Ezugwu, "Smart home automation: Taxonomy, composition, challenges and future direction," in *Computational Science and Its Applications—ICCSA 2020*. O. Gervasi, B. Murgante, S. Misra, C. Garau, I. Blečić, D. Taniar, B. O. Apduhan, A. M. A. C. Rocha, E. Tarantino, C. M. Torre, and Y. Karaca, Eds. Cham, Switzerland: Springer, 2020, pp. 878–894. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-030-58817-5_62, doi: 10.1007/978-3-030-58817-5_62.
[9] M. Jerabandi and M. M. Kodabagi, "A review on home automation system," in *Proc. Int. Conf. Smart Technol. Smart Nation*, Aug. 2017, pp. 1411–1415.
[10] L. Smirek, G. Zimmermann, and D. Ziegler, "Towards universally usable smart homes-how can MyUI, URC and openHAB contribute to an adaptive user interface platform," in *Proc. IARIA*, Nice, France, 2014, pp. 29–38.
[11] S. Faroom, M. N. Ali, S. Yousaf, and S. U. Deen, "Literature review on home automation system for physically disabled peoples," in *Proc. Int. Conf. Comput., Math. Eng. Technol. (iCoMET)*, Mar. 2018, pp. 1–5.
[12] H. Derhamy, J. Eliasson, J. Delsing, and P. Priller, "A survey of commercial frameworks for the Internet of Things," in *Proc. IEEE 20th Conf. Emerg. Technol. Factory Autom. (ETFA)*, Dec. 2015, pp. 1–8.
[13] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," *J. Cleaner Prod.*, vol. 140, no. 3, pp. 1454–1464, 2017.
[14] T. Abbas. (Sep. 2018). *Smart Home Scenarios*. Accessed: Dec. 14, 2021. [Online]. Available: https://figshare.com/articles/dataset/SMART_HOME_SCENARIOS_27_09-2018_xlsx/7140428/1
[15] P. Schoutsen. *Home Assistant*. Accessed: Dec. 14, 2021. [Online]. Available: https://www.home-assistant.io/
[16] Domoticz Team. *Domoticz*. Accessed: Dec. 14, 2021. [Online]. Available: https://www.domoticz.com/
[17] K. Kreuzer. *Empowering the Smart Home*. Accessed: Dec. 14, 2021. [Online]. Available: http://www.openhab.org/

[18] V. Khaeva. *Iobroker–Automate Your Life–Open Source Automation Platform*. Accessed: Dec. 14, 2021. [Online]. Available: https://www.iobroker.net/
[19] G-Labs Open Source Factory. *Homegenie*. Accessed: Dec. 14, 2021. [Online]. Available: http://www.homegenie.it/
[20] *Calaos*. Accessed: Dec. 14, 2021. [Online]. Available: https://calaos.fr/
[21] C. Kratky. *Wirehome*. Accessed: Dec. 14, 2021. [Online]. Available: https://github.com/chkr1011/Wirehome.Core/
[22] O. Bv. *Openmotics*. Accessed: Dec. 14, 2021. [Online]. Available: https://wiki.openmotics.com/
[23] E. Nicoletti, M. Cicolella, G. Pulido de Torres, A. Mengoli, and M. Mazzoni. *Freedomotic*. Accessed: Dec. 14, 2021. [Online]. Available: https://freedomotic-user-manual.readthedocs.io/
[24] R. König. *Fhem Wiki—Informationsportal Zum Fhem Smarthome-Server*. Accessed: Dec. 14, 2021. [Online]. Available: https://wiki.fhem.de/wiki/Hauptseite
[25] K. R. Keegan. *Misterhouse—It Knows Kung-Fu*. Accessed: Dec. 14, 2021. [Online]. Available: http://misterhouse.sourceforge.net/
[26] S. Strömberg. *Opennethome*. Accessed: Dec. 14, 2021. [Online]. Available: http://opennethome.org/
[27] H. Klein. *Ago Control*. Accessed: Apr. 15, 2021. [Online]. Available: https://wiki.agocontrol.com/
[28] B. Morgan and B. T. Hill. *The Thing System–Take Control of Things*. Accessed: Dec. 14, 2021. [Online]. Available: http://thethingsystem.com/
[29] V. Goel and S. Sharma. *Äautomate*. Accessed: Dec. 14, 2021. [Online]. Available: https://uautomate.herokuapp.com
[30] T. Giachi. *Neon Homecontrol*. Accessed: Dec. 14, 2021. [Online]. Available: https://neon-home-control.readthedocs.io/
[31] K.-D. GitHub. *Pytomation*. Accessed: Dec. 14, 2021. [Online]. Available: http://www.pytomation.com/
[32] U. Freese. *Smarthomatic*. Accessed: Dec. 14, 2021. [Online]. Available: https://www.smarthomatic.org/
[33] *Smart Haus—How to Build the Most Robust and Secure Home Automation System*. Accessed: Dec. 14, 2021. [Online]. Available: https://medium.com/free-code-camp/the-most-robust-and-secure-home-automation-system-6d0ddbb39f29
[34] F. Wautier. *Autobuddy*. Accessed: Dec. 14, 2021. [Online]. Available: http://frawau.github.io/AutoBuddy/
[35] A. Sanatinia and G. Noubir, "On GitHub's programming languages," *CoRR*, vol. abs/1603.00431, pp. 1–10, Mar. 2016.
[36] M. Papamichail, T. Diamantopoulos, and A. Symeonidis, "User-perceived source code quality estimation based on static analysis metrics," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. (QRS)*, Dec. 2016, pp. 100–107.
[37] Home Assistant. *Architecture*. Accessed: Dec. 14, 2021. [Online]. Available: https://developers.home-assistant.io/docs/architecture/core
[38] P. Hüwe and S. Hüwe, *IoT at Home*. München, Germany: Carl Hanser Verlag GmbH, May 2019.
[39] Domoticz Team. *Domoticz MQTT Architecture*. Accessed: Dec. 14, 2021. [Online]. Available: https://www.domoticz.com/wiki/MQTT
[40] F. Heimgaertner, S. Hettich, O. Kohlbacher, and M. Menth, "Scaling home automation to public buildings: A distributed multiuser setup for OpenHAB 2," in *Proc. Global Internet Things Summit (GIoTS)*, 2017, pp. 1–6.
[41] I. Fischer, "Weitergepuzzelt-die wichtigsten neuen erweiterungen der smart-home-steuersoftware iobroker sein," in *Proc. Heise Medien*, 2019, pp. 178–179.
[42] IoBroker GmbH. *IoBroker Architecture*. Accessed: Dec. 14, 2021. [Online]. Available: https://github.com/ioBroker/ioBroker/blob/master/img/architecture.png
[43] O. Jarczyk, B. Gruszka, S. Jaroszewicz, L. Bukowski, and A. Wierzbicki, "GitHub projects quality analysis of open-source software," in *Proc. Int. Conf. Soc. Informatics*, 2014, pp. 80–94.
[44] openHAB Community and the openHAB Foundation. *Next-Generation Rule Engine*. Accessed: Dec. 14, 2021. [Online]. Available: https://www.openhab.org/docs/configuration/rules-ng.html
[45] Home Assistant. *Home Assistant Quality Scale*. Accessed: Dec. 14, 2021. [Online]. Available: https://www.home-assistant.io/docs/quality_scale/
[46] OpenHAB Community. *Contributing to the Development of OpenHAB*. Accessed: Dec. 14, 2021. [Online]. Available: https://www.openhab.org/docs/developer/contributing.html
[47] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, "Internet of Things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios," *IEEE Access*, vol. 8, pp. 23022–23040, 2020.
[48] J. R. Sauro, *Quantifying the User Experience: Practical Statistics for User Research*, 2nd ed., J. Sauro and J. R. Lewis, Eds. Burlington, MA, USA: Morgan Kaufmann, 2016.
[49] A. D. Scott, *Collaborative Web Development*. Newton, MA, USA: O Reilly Media, 2017.
[50] N. Carvalho, A. Simáes, and J. Almeida, "DMOSS: Open source software documentation assessment," *Comput. Sci. Inf. Syst.*, vol. 11, no. 4, pp. 1197–1207, 2014.
[51] A. Wingkvist, M. Ericsson, R. Lincke, and W. Lowe, "A metrics-based approach to technical documentation quality," in *Proc. 7th Int. Conf. Qual. Inf. Commun. Technol.*, Sep. 2010, pp. 476–481.

BRIAN SETZ (Graduate Student Member, IEEE) received the master's degree in computing science from the University of Groningen. He is currently pursuing the Ph.D. degree with the Institute of Architecture of Application Systems (IAAS), University of Stuttgart, where he is also a member of the Service Computing Department. He is also a Research Assistant with the Institute of Architecture of Application Systems (IAAS), University of Stuttgart. His research interests include the Internet of Things, green computing, cloud computing, and energy efficient data centers.

SEBASTIAN GRAEF received the bachelor's and master's degrees in software engineering from the University of Stuttgart, in 2018 and 2021, respectively. He is currently a Software Developer and a Consultant at Novatec Consulting GmbH, where he mainly focuses on web-based enterprise application architecture. His research interests include cloud computing, AI-planning, the Internet of Things, and system automation.

DESISLAVA IVANOVA received the bachelor's degree in business informatics and the master's degree in software engineering from the University of Stuttgart, Germany, in 2018 and 2021, respectively. From 2016 to 2021, she worked as a Student Assistant at the Department of Information Systems, University of Hohenheim, Stuttgart, Germany. She is currently pursuing a career as a Software Engineer. Her research interests include home automation, the IoT, and NLP.

ALEXANDER TIESSEN received the bachelor's degree in software engineering from the University of Stuttgart, Germany, in 2019. He is currently writing his master's thesis in cooperation with ITK Engineering with the University of Stuttgart. Together with the Fraunhofer Society, he worked on a project regarding explainable artificial intelligence, in 2020. His research interests include deep learning, natural language processing, and explainable AI.

MARCO AIELLO (Senior Member, IEEE) received the Ph.D. degree in logic from the University of Amsterdam, the Habilitation degree in applied informatics from TU Wien, and the master's degree in engineering from the La Sapienza University of Rome. He is currently a Professor of computer science and the Head of the Service Computing Department, University of Stuttgart, Germany. He is also an Elected Member of the European Academy of Sciences and Arts and an Honorary Professor of distributed systems at the University of Groningen, The Netherlands, where he was a Faculty Member, from 2006 to 2018. His research interests include service computing and smart energy systems.