Journal of *Sensor and Actuator Networks*

Article

SensoMan: Social Management of Context Sensors and Actuators for IoT

Georgia M. Kapitsaki [1,*], Achilleas P. Achilleos [2,*], Philippos Aziz [1] and Athina C. Paphitou [1]

[1] Department of Computer Science, University of Cyprus, 1 University Avenue, Nicosia 2109, Cyprus; philaziz92@gmail.com (P.A.); athensp22@gmail.com (A.C.P.)
[2] Department of Electrical and Computer Engineering and Informatics, Frederick University, Nicosia 1036, Cyprus
* Correspondence: gkapi@ucy.ac.cy (G.M.K.); com.aa@frederick.ac.cy (A.P.A.)

Abstract: Sensor networks that collect data from the environment can be utilized in the development of context-aware applications, bringing into sight the need for data collection, management, and distribution. Boards with microcontrollers, such as Arduino and Raspberry Pi, have gained wide acceptance and are used mainly for educational and research purposes. Utilizing the information available via sensors connected to these platforms requires extended technical knowledge. In this work, we present a sensor management framework, SensoMan, that manages a collection of sensors spread in the environment connected to microcontroller boards. We present the framework's architecture, a method for sensor data management, and a prototype system. Sensor data can also trigger the execution of actions on actuators. Thus, we further propose a rule engine as well as social connectivity following a scheme where sensors and their data can be shared among users. Our work shows that the creation of such a system is feasible and can use simple equipment (e.g., sensors, controller plugs) that can be replicated in other environments. The use of SensoMan is demonstrated via two scenarios that show its potential in combining simple tools that do not require an extended learning curve. A small-scale user study was also performed.

Keywords: Internet of Things; microcontroller boards; rule engine; social network

Citation: Kapitsaki, G.M.; Achilleos, A.P.; Aziz, P.; Paphitou, A.C. SensoMan: Social management of context sensors and actuators for IoT. *J. Sens. Actuator Netw.* **2021**, *10*, 68. https://doi.org/10.3390/jsan10040068

Academic Editors: Dionisis Kandris, Nikolaos Tselikas, George Tselikis, Ilias Politis and Lei Shu

Received: 9 October 2021
Accepted: 24 November 2021
Published: 27 November 2021

1. Introduction

The growing popularity of the Internet of Things (IoT) and the smart cyber physical systems (cPHS) are providing a world of interconnected devices that communicate with different protocols in order to provide a variety of applications [1,2]. Components in these environments communicate with minimal human intervention, whereas various application areas have emerged ranging from smart buildings to smart cities and smart factories. Different commercial vendors provide their platforms with tools and technologies tailored to their systems. However, open-source microcontroller boards, such as Arduino and Raspberry Pi, have also gained wide popularity and have been adopted in many IoT solutions. Various sensors of different size and accuracy can be connected to such boards, offering a view of the board surroundings. This information is valuable for a plethora of applications that can perform composite actions on these measurements. In order to be able to use these sensors, programming at the level of the board is required.

However, the programing power of such boards is constrained, making the processing of large volumes of data on the board hard. For this reason, the utilization of information from these boards can function more efficiently when integrated in systems that can be replicated for use in various application domains offering a data management core with main modules. Cloud computing and fog computing give a solution by providing Infrastructure as Service (IaaS) that can be combined with IoT architectures to store the data and perform complex processing actions [3]. Fog Computing also provides computation, storage, and networking services between end devices and traditional Cloud Computing Data Centers [4]. Context-awareness plays additionally a major role in these environments, as information collected from sensors serving as context can be used in various applications providing personalized experience to users. Any information relevant to a user, a service, and their interaction can be considered context [5]. Context information of diverse complexity can be generated by processing raw sensor data [6].

Managing sensors spread in the environment can be a valuable process for a variety of applications in domains, such as agriculture, smart homes, medical technology, and healthcare, where context-aware features and a smart interpretation of the surrounding environment is necessary. As mentioned in a previous work, one of the challenges of IoT concerns the development of techniques that convert these raw data into usable knowledge [7]. Approaches that provide interoperability have emerged, such as openHAB [8] and Eclipse IoT [9] that consists of many projects. However, they both have a certain level of complexity and are not targeting users with limited technical knowledge.

Having as motivation the above, in this work, we are describing SensoMan including its design and our work toward a sensors management framework that allows a uniform management of data collected from diverse sensors connected to microcontroller boards providing also mechanisms for the automatization of activities related to smart-enabled devices: triggering events on smart devices based on specific measurements in other devices or external conditions, and sharing of sensors and measurements between users of the platform via the SensoMan social network (see Figure 1). Via SensoMan, we provide a new method for sensor data management while offering additional functionality to the users (e.g., social network), outlining thus a framework that can be replicated in other environments and locations.

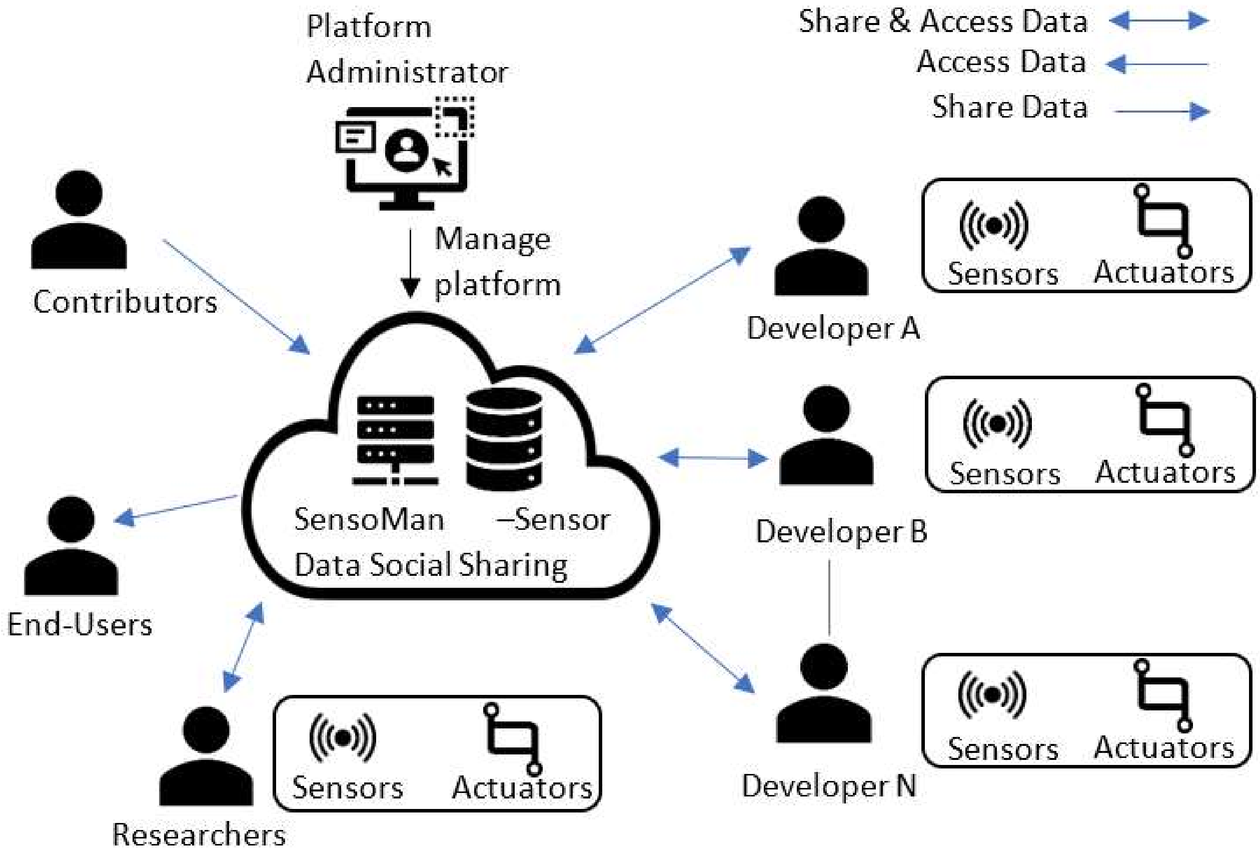


Figure 1. The SensoMan platform—concept.

The initial concept of SensoMan has been presented in a previous work [10]. In this work, the initial architecture is extended with additional functionality that provides an integrated solution for sensor management. SensoMan is a management system for the collection of sensor data connected to Arduino and Raspberry Pi microcontroller boards, whereas additional boards, such as Intel Galileo and BeagleBone, have also been tested. The monitoring of measurements and the management of installed sensors is performed via the web management platform. SensoMan introduces additionally a rule engine module that gives users the possibility of defining specific sensor-driven conditions that can fire other events. The events that can be triggered when the condition is satisfied or not satisfied are also defined as part of the rules. In the current state of implementation, these events are related with the activation or deactivation of specific devices via an Arduino platform that communicates with the devices with infrared signals acting as actuator-board. We are also outlining the use of a social network for composite applications supported by SensoMan, where users can share sensors with other users of the system.

The main contributions of our work are as follows: (1) the introduction of an extensible management framework, i.e., a system architecture, a method for sensor data management, and a prototype system, using open source software that can assist the vision of integrating IoT, enabling interoperability among different devices, (2) the creation of a rule engine that allows users to specify rules for the management of actuators in a simplified environment without the need to perform the rule implementation, and (3) the introduction of social aspects allowing the reuse of data in applications required by different users. Although various vendor specific and more complete open-source solutions exist, we argue that our approach, which is also based on open-source solutions, contains features that, to the best of our knowledge, are not available in other systems (i.e., social network), and it focuses on functionalities that can assist users with limited technical expertise. Nevertheless, existing solutions, such as openHAB [8] that has gained the interest of the developers' community, provide more holistic approaches for automation purposes. The continuation and focus of this work is mainly found on social sharing of sensor data that can be used, e.g., by researchers and developers, for experimentation and research purposes.

The rest of the paper is structured as follows. Section 2 covers related work in the area. Section 3 provides an overview of SensoMan describing the architecture of the system, whereas it also outlines the use of the SensoMan social network. The rule engine and its features are presented in Section 4. Section 5 provides implementation details for the platform, whereas Section 6 is dedicated to the demonstration of use of SensoMan via a field study captured in two scenarios and a small-scale user study. Finally, Section 7 concludes the paper outlining directions of future work.

2. Related Work

Since the appearance of single boards with microcontrollers, different applications have been designed utilizing the capabilities of such platforms in order to perform different actions, including the provision of frameworks for facilitating the programming of microcontroller boards even for novice users [11]. The problem of integrating heterogeneous, geographically and administratively, dispersed sensors and IoT services in a semantically interoperable fashion is discussed by Soldatos et al. [12]. OpenIoT introduces an ontology, the Sensor Networks (SSN) ontology, for representing physical and virtual sensors constructed in the framework of the EU OpenIoT project, whereas it uses a cloud database for storing sensor information. However, it seems that it is not maintained any more.

The system closest to SensoMan is the openHAB open-source system [8]. The openHAB platform can integrate various home automation systems and technologies, whereas it allows the definition of rules and the offering of dedicated user interfaces. It can run on any device capable of executing JVM (Java Virtual Machine). It offers a number of APIs to allow the connection of other systems to openHAB. Various works have used openHAB in order to create dedicated applications, such as the creation of a home automation framework in a Philippine setting that employed RESTful services and the Message Queuing Telemetry Transport (MQTT) [13]. The work describes the technologies and the creation process.

Previous works have addressed various aspects of social web of things and connected smart objects. Chen et al. view the Social Internet of Things (SIoT) as a mix of traditional peer-to-peer networks and social networks and focus on trust management in this setting [14]. They introduce a distributed trust protocol, where each node can update trust in relevance to other nodes when there is an encounter or interaction. The dynamic discovery of smart services has also been addressed, with the aim of finding situation-aware services that match the needs of the user [15]. A filtered list of services is generated after semantic matching between user needs and the available services is performed. A proof-of-concept prototype has been implemented for the case of a smart airport.

An architecture that provides a foundation for the development of lightweight microservices based on socially connected web objects has also been proposed [16]. The architecture considers the social relationships among objects, introducing a relevant social relationship model for the discovery of web objects, along with an ontology model. A prototype implementation using a public museum as a use case is provided. The fundamentals of SioT are presented by Roopa et al., where prerequisites and challenges are discussed, along with relevant techniques and the review of relevant research publications and future directions in the area [17]. The Social Internet of Things (SIoT) is also discussed in [18].

The social network attempt closest to the concept of SensoMan can be found in the Paraimpu platform that allows people to connect, use, share, and compose physical and virtual things, services, and devices in order to create personalized and pervasive applications [19]. This work introduces two scenarios where Paraimpu can be useful: an ambient assisted living scenario concerning not forgetting medications, and social controlled urban lights, allowing users to change the tone of the colors of a glass tower in the city.

Rule engines are available in a number of existing frameworks. The authors in [20] claim that although several rule engines are already available, limited effort has been devoted to rule-based solutions that are tailored to the IoT and consider rule configurability and extensibility according to application requirements. This work proposes a RESTful rule management framework for IoT applications that satisfies these requirements. The framework is centered around a resource-based graph, which enables the uniform representation of things (e.g., sensors and domain entities) and rules as URI-addressable resources.

Another work that provides a rules engine for IoT is openHAB, which adopts a more graphical oriented rules definition method in comparison to the RESTful rule management framework above [8]. In fact, the rule engine of openHAB has a similar functionality with the engine of SensoMan. OpenHAB recently allows editing rules in a graphical fashion but uses a different approach to SensoMan that targets specific rule cases targeting users with limited technical knowledge. Flow diagrams or pipes are an alternative technology that has been used to build rules for IoT applications. Node-RED is a flow-based tool for wiring together hardware devices, APIs, and online services in new and interesting ways [21]. The editor of Node-RED works in the browser, which simplifies wiring together flows, with the extensive set of nodes that is provided in the palette of the editor. Moreover, rules can be defined using Particle, which is an open-source rules engine with a drag-and-drop application builder based on Node-RED. Flow based programming has become even more popular with the introduction of "serverless" computing, where cloud applications can be built by chaining functions.

IFTTT (If This Then That) follows a similar approach that is based on the concept of everything as a service [22]. IFTTT allows interconnecting heterogeneous services that allow a user to program a response to events in the world. In fact, IFTTT provides a platform that connects applications, hardware devices, and software services from different developers in order to trigger one or more automations involving those apps, devices, and services. IFTTT programs are called applets or recipes and can be created graphically with a web interface or iOS or Android application. Another highly similar tool to IFTTT is Zapier, which allows creating workflows that are defined as actions and triggers for interconnecting web applications and automating workflows [23]. The tool focuses on applications and does not provide access to hardware devices, such as in the case of IFTTT.

Overall, in relation to previous works, SensoMan provides an approach with the use of various open-source tools that targets a smaller setting than more holistic but complex approaches such as openHAB and Node-RED. It integrates with other frameworks, i.e., HTML5 Context Middleware (H5CM) [24], to cover additional functionality. The rule engine and the other features provide users with limited technical expertise the possibility of using the system. It introduces social network support that is not found, to the best of our knowledge, in other attempts, but is vital, as it extends the set of scenarios where the infrastructure can be used. As can be seen in Table 1, which presents a comparison with relevant platforms, only Paraimpu provides a social network attempt closest to the concept of SensoMan.

Table 1. Example rule and corresponding format for storing the rule.

	Senso Man	**OpenIoT**	**OpenHab**	**Paraimpu**	**Node-RED**	**IFTTT**	**Zapier**
Open-Source	√	√	√	×	√	×	×
Extensibility (Plugin-based)	√	√	√	×	√	×	×
Connectivity/Device Management	High	High	High	Medium	High	High	High
Sensor Data Social Sharing	√	×	×	√	×	×	×
IoT Rules Configurability	High	High	High	×	Medium	Medium	Medium

3. Overview and Design of SensoMan

3.1. SensoMan Architecture

SensoMan was designed having as its main aims the creation of a system that is based on open source components, integrates well with selected web frameworks, and is relatively simple to understand and use for users with limited technical knowledge. These were the main challenges we faced during its design and are reflected in the SensoMan architecture that was created using an iterative process in order to keep the design modular. Additional challenges were the creation of a simple rule engine (detailed in the next section) and the introduction of a social network that would provide possibilities not available in similar solutions.

The SensoMan system consists of specific modules and can be expanded with the addition of new modules if additional functionality is desired. A set of core modules is required for supporting activities that carry out the main functions of SensoMan. All modules are independent following a scalable architecture, giving space for additional functionality. Since SensoMan is based on the use of microcontroller boards, both hardware and software modules are used. Hardware modules consist of different boards, sensors, sensor-enabled devices, and actuators that can be connected to any device, upon which specific actions can be triggered, such as lamps, air conditioning systems, televisions, etc. Regarding sensors and actuators, we adopt the definition provided in an earlier work [19]:

- ***Sensors***: anything capable of producing data of a specific type;
- ***Actuators***: anything able to perform actions by consuming data items produced by a sensor.

The software management applications that support the core functionality for end users consist of an Android management application and a web application, which are both visible in Figure 2. The mobile application has been provided for management purposes in addition to the web platform in order to support users when they are on the move, providing a subset of the capabilities of the web platform, where the main user dashboard is found. Via these management applications, users have the possibility of monitoring their boards and performing different actions on the sensors they have access to. Specifically, the SensoMan management application enables users to have a look at the sensors' latest, i.e., current, as well as historic values. Different presentation options are supported with users having access to tables and graphs that display the changes in sensor values over time. More information on the SensoMan management Android application can be found in prior relevant work [10].

The interconnection and the required communication for all software and hardware modules is performed via a number of RESTful services, whereas a database is used for storing necessary data, e.g., sensor measurements (SensoMan Database in Figure 2). A separate module allows the connection of sensor data with actions on actuators via rules defined in the respective rule engine of SensoMan detailed in the next section.

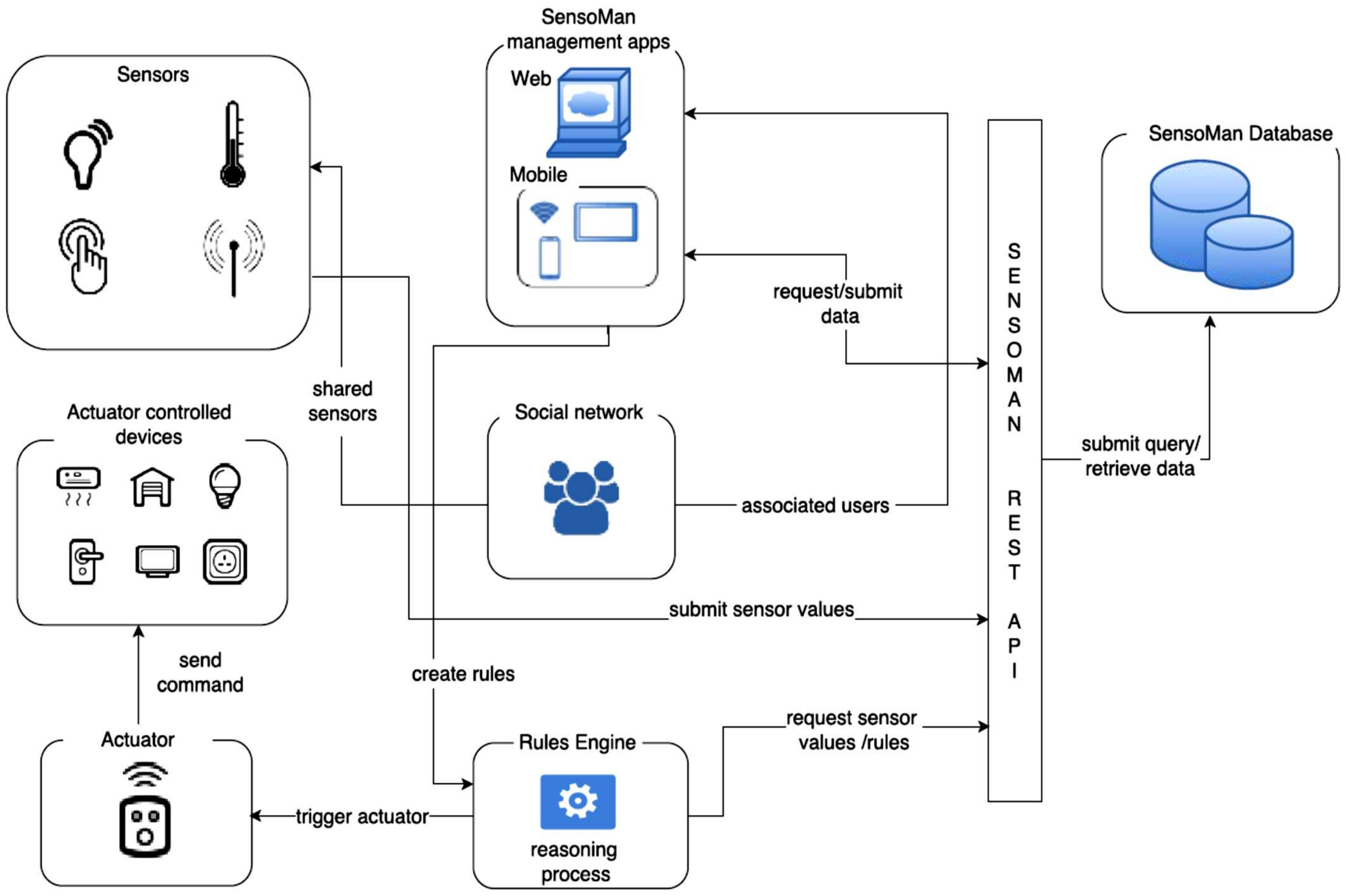


Figure 2. SensoMan architecture.

The remaining SensoMan modules and their interconnections are also depicted in Figure 2. The interplay of all modules is required for the functionality of SensoMan. Sensors of different types (e.g., motion detectors) connected to microcontroller boards obtain different values for the environment and store them in the database using the relevant RESTful service (via the submit sensor values action in Figure 2). Users of the management applications (web or mobile) can invoke the RESTful service to retrieve any information required for the sensors connected to the microcontrollers considering the access policies (e.g., a user has to be in close proximity of a microcontroller in order to be able to view the respective sensor values, or be the board owner in order to add new sensors to the board) via the request/submit data action. Users can utilize the SensoMan rule engine via its web interface accessible from the main dashboard of the web management application in order to automate the management of specific smart devices by creating rules that depend on sensor values (request sensor values/rules actions). This automatization is performed via the use of the microcontroller boards that act as device actuators (trigger actuator action).

The scalable design of SensoMan allows the addition of new microcontrollers and new sensors, as long as internet connectivity is available. Microcontroller boards have a dual role in the system acting either as data collectors via the sensors attached to them (i.e., sensor controllers) or as device actuators (i.e., actuator controllers). SensoMan uses an ownership system where each user owns a set of sensors and microcontrollers. SensoMan has the following user types:

- ***Simple***: Simple users can only view measurements from sensors connected to boards in close proximity to their physical location or additionally from sensors shared with them by other platform users;
- ***Owner***: In addition to simple users, owners can add new boards at a specific location (existing or new location), add new sensors to boards owned by them, create rules in the SensoMan rule engine and view existing rules on their measurements, or edit/delete existing rules. Owners can also change the access credentials of other users to specific boards or specific sensors attached to the board in the framework of the SensoMan social network;
- ***Admin***: These users have a general monitoring of the system receiving notifications for new boards and sensors added to the system, whereas they can also change the frequency of collecting measurements for sensors values. The default interval for collecting measurements is currently set to 2 s. Some users may be both admin and owner (admin–owner), when the owner of microcontrollers has also admin privileges.

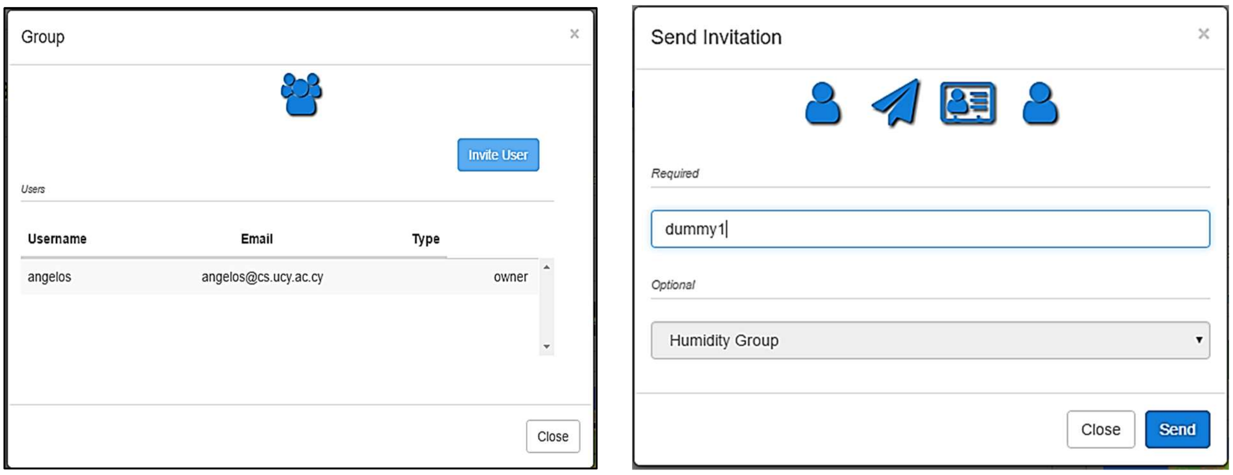
3.2. SensoMan Social Network

Social network sites (SNSs) are defined as "web-based services that allow individuals to (1) construct a public or semi-public profile within a bounded system, (2) articulate a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system" [25]. The social network properties of SensoMan allow users to connect with each other and exchange information, promoting collaboration between users. Adding social network properties to SensoMan ensures that users benefit the most from information collected from different sensors. The social network allows users to have access to sensors owned by other users, having thus, for instance, the opportunity to gain access to expensive sensors they cannot afford (e.g., equipment used in earthquake early warning systems shared by scientists that perform research at nearby locations, or any type of equipment shared between research institutes in the same city) or gain from collective intelligence, where data collected from different sensors (e.g., belonging to the neighbors or a building nearby) can be aggregated to provide knowledge that would not be feasible to obtain with separate use of the sensor equipment (e.g., fire detection by aggregating temperature data coming from different sensors in a specific floor in a building).

SensoMan can use information from different sources combining them to draw useful conclusions on the raw data received from the different sensor measurements of the system and other locations, such as web services (e.g., for weather conditions), posts from social networks such as Facebook, or tweets. Currently, the SensoMan framework is providing the methods and software capabilities that enable social sharing of data from sensors, defining and imposing rules for reasoning on these data and triggering actions on actuators, e.g., turning on/off a smart light. This knowledge extraction can be further facilitated via the use of machine learning algorithms on the sensor data [26], which are currently implemented by the IoT use case developers. An example application (i.e., IoT use case) of a Smart Laboratory Intrusion System was presented with the initial concept of SensoMan that uses the SensoMan infrastructure to learn the patterns of human presence in a laboratory and subsequently combine this information with information from SNSs check-ins to detect suspicious presence in the laboratory adopting machine learning techniques [10]. This concept is not analyzed further in the current text, and the reader can refer to the previous publication for more information. In order to support the inverse direction of information sharing as well, information from SensoMan has been integrated to the aforementioned HTML5 Context Middleware to provide an additional source of information for developers of context-aware HTML5 web applications; i.e., developers have the opportunity to retrieve SensoMan sensor values and utilize them in their web application in order to provide context-aware capabilities [24,27]. These concepts can be strengthened via the sharing of information between users, since sharing can also provide access to more data coming from different users and provide better accuracy when applying machine learning techniques.

Actions typically found in a social network are available in SensoMan in order to allow connectivity between users (i.e., connection requests, approving/rejecting connection requests). All user types can use the social network to view the measurements of other users connected to them or of users that belong to the same group. Connected users can request access to specific boards and sensors of the other user. Simple users can only request read access to the sensors of other users in order to read sensor measurements, whereas all other user types can request read or write access to the sensors. Each user can belong to different groups and can either have the default sharing mechanism with the other group members or define a custom sharing type for specific user(s) of the group. Users of one group can request access to the sensors of another user of the same group. Only the group owner can add or invite new members to the group. All users can become members to groups, whereas all but simple users can create a new group and become group owners. The group concept has been introduced in order to allow users that belong to the same organization, block of buildings, or discipline to share their data. Figure 3 shows how an invitation to join a group is sent to another user.

Users that no longer wish to have access to the sensors of another user can disconnect from the user or leave the group based on the type of connection between the users. Notifications about social network events (e.g., connection requests) appear to the user as typically performed in SNSs. Connected users (either via peer connection or a group) that share sensors have access to the sensors and can manage sensors from their dashboard.



Users

Username	Email	Type
angelos	angelos@cs.ucy.ac.cy	owner

Required: dummy1

Optional: Humidity Group

Figure 3. Sending user an invitation to join a group in SensoMan's social network.

The Authentication, Authorization, and Accounting (AAA) framework is used to enforce the social network's security. At first, the identity of the user is confirmed with a username and an encrypted password-based authentication; then, token-based authorization (with time expiration) is used to access any resources in the social network, and finally, the access to resources from end users is logged to enable accountability.

4. The SensoMan Rule Engine

4.1. Sensors and Actuators

As aforementioned, every microcontroller board can act as a sensor controller or as an actuator controller or as both. Data from the environment are collected via sensors, whereas actuators are used to control devices in the environment (e.g., light up a lamp). A previous work defines actuators as data sinks that consume data [19]. This is similar with our case, where an actuator acts based on the sensor data it receives. A more general definition is also compliant, where actuators are indicated as "*a type of tool which is used to put something into automatic action. It is used on a wide variety of sources, from humans putting something into action to computers starting up a program.*" [28]. Thus, the data are used by the actuator to perform an action or trigger an event. Actuators can change the parameters of the environment they are installed in, helping to trigger actions to smart-enabled devices that can receive appropriate signals from the actuators.

Examples of actuators that are used in the framework of the SensoMan system are the infrared transmitter and the controller plug. They are used in order to activate or deactivate a device based on the user rules and are mounted to microcontroller boards, making specific boards act as actuator controllers.

4.2. Engine and Reasoning Process

The aim of the rule engine of SensoMan is to automate procedures that require continuous sensor data monitoring in order to work properly. The human intervention is kept to a minimum, making it feasible to detect with a high accuracy the sensor conditions that can trigger an event. Moreover, it provides the possibility for the creation of event automatization in different environments minimizing the coupling with the hardware infrastructure, since all activities are managed by the SensoMan platform. SensoMan users can define their rules that can trigger actions on the environment based on indicated conditions.

The rule engine defines a situation that can trigger an action based on input sensors measurements. Each rule has two parts:

- *Condition* definition: This can be a simple (e.g., temperature lower than 30 degrees Celsius) or a composite condition (e.g., the three last temperature measurements below 30 degrees Celsius and the motion sensor detecting movement in its last measurement). The structure of conditions is shown in Figure 4. Comparison operators (>, >=, ==) and logical operators (AND, OR) are used for combining conditions. Although the creation of more complex rules would be feasible, currently, the SensoMan rule engine supports only the conditions shown in the figure in order to keep a balance between complexity and simplicity for the users that define rules.
- Triggering action: This part contains the device and the action on the device described by the rule (e.g., lamp light on, air conditioning system off). Triggering actions are activated if a condition definition is met.

Each rule has a specific owner, i.e., its creator. A rule that affects or is based on specific sensor values is triggered only if the user is the owner of these sensors and their respective controllers. A user with write access to the sensors of another user or of the users of a group she belongs to via the connections of the SensoMan social network is also allowed to create rules combining values from boards and sensors that are owned by her or by the other user(s).

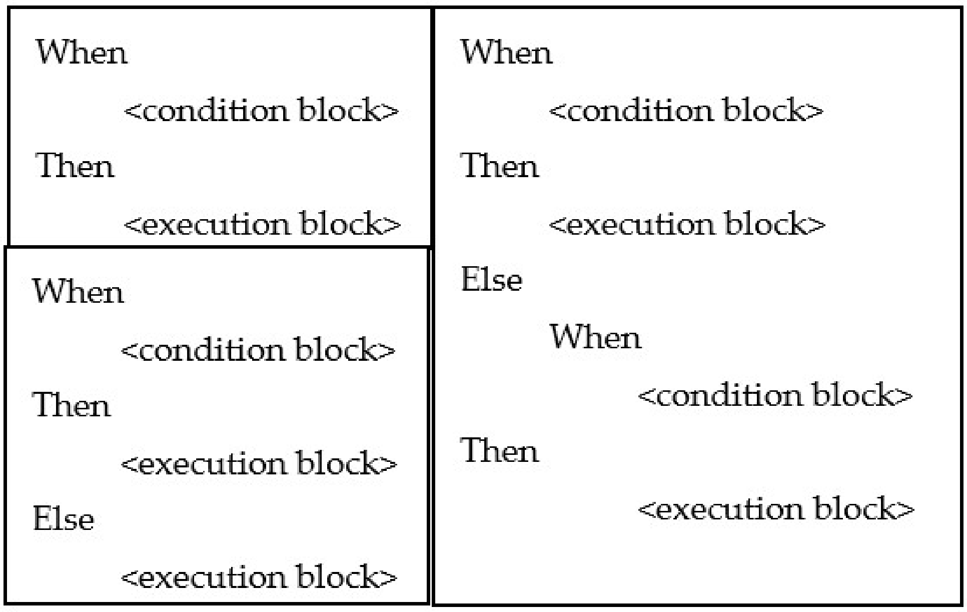


Figure 4. Condition definition format in SensoMan’s rule engine.

Rules are stored in XML (Extensible Markup Language) format. This format was chosen, since XML is also used in the rule engine of OpenHAB that has been adopted in previous works allowing fast retrieval and processing of the rules [29]. An example rule is depicted in Table 2 along with its mapping to XML elements showing a portion of the XML file used to store the rule. Users can indicate the type of condition (in the example case, this is a “when–else” condition), the names of the sensors involved as selected from the dropdown menus provided in the rule engine’s user interface, e.g., temperature sensor (DHTEMPSEN11 in the example), the range or values expected for each sensor, e.g., >0 (greater—GT than 0), whereas they can also mention how many sensor values should be examined to verify whether the condition is satisfied, e.g., only the last sensor measurement or the last X measurements values can be considered (e.g., sensitivity: check last 0 values). In the latter case, the average value of the measurements is calculated when parsing the rule in order to assess whether the condition is satisfied. The XML structure contains a second element with name NUM and value 0 to denote that only the current measurement is considered in the rule (<field name = “NUM” > 0 < /field >).

Table 2. Example rule and corresponding format for storing the rule.

Rule Structure	XML Format
	*<**block** type=“ control_when” id=“…” x=“−137” y=“38”>*
	*<**value** name=“when”>*
	*<**block** type=“logic_operation” id…”>*
	*<**field** name=“OP”>AND</**field**>*
when	*<**value** name=“A”>*
DHTEMPSEN11 > 0 |	*<**block** type=“my_logic_compare” id=“…”>*
sensitivity:	*<**field** name=“OP”>GT</**field**>*
check last 0 values	*<**field** name=“NUM”>0</**field**>*
AND	*<**field** name=“NUM”>0</**field**>*
	*<**value** name=“sensor_name”>*
	*<**block** type=“sensor” id=“…”>*
	*<**field** name=“select_sensor”>DHTEMPSEN11</**field**>*
	*</**block**>*
	*</**value**>*

LDRArduinoBase-1 == 0 | sensitivity: check last 0 values	</**block**> </**value**> <**value** name="B"> <**block** type="my_logic_compare" id="…"> <**field** name="OP">EQ</field> <**field** name="NUM">0</field> <**field** name="NUM">0</field> <**value** name="sensor_name"> <**block** type="sensor" id="… "> <**field** name="select_sensor">LDRArduinoBase-1</**field**> </**block**> </**value**> </**block**> </**value**> </**block**> </**value**>
then *LampArduino-2 state to ON*	<**statement** name="then"> <**block** type="execution" id="…"> <**field** name="set_value">LampArduino-2</field> <**value** name="actuator_value"> <**block** type="device_state" id="…"> <**field** name="state">1</**field**> </**block**> </**value**> </**block**> </**statement**> </**block**>

The SensoMan rule engine parses the XML file and performs necessary actions. This way, the information on sensor names along with their recently measured values and the values of the condition can be parsed. Regarding the actions, the device names and the type of action are also read. Thus, the structure of the XML rules is transferred to code in order to apply the rule when the conditions are met. In order to perform an action on a device, the device IP address and port is retrieved, and the respective code for the device activation or deactivation is sent to the device. The IP address and the port are required, so as to be able to send a message to the device (via infrared).

Regarding the interaction with the user, in order to ensure that minimal coding effort will be required to use the rule engine, an environment with blocks or bricks from a puzzle was designed as a user interface for the SensoMan rule engine. No technical knowledge is required in order to use the rule engine, as the same environment is used, for instance, in MIT App Inventor [30]. Three types of blocks are used: (1) blocks that are used to define the condition logic, (2) blocks for sensor selection participating in the condition, and (3) blocks for device activation/deactivation. Users can select the microcontroller boards they wish to use for the condition from a respective drop-down menu, and the sensors attached to these boards are accordingly displayed in a similar drop-down menu. The same applies to the actuator boards, where the user can select among the devices connected to the selected board. The appearance of the main blocks as displayed to the users is shown in Figure 5. As aforementioned, users can also edit and delete existing rules via the same graphical environment. Existing rules are displayed to the user in a textual format for better visualization, whereas the user is transferred to the block environment when she selects the editing of the rule. Figure 6 depicts the textual format of the rule for the example rule described previously.

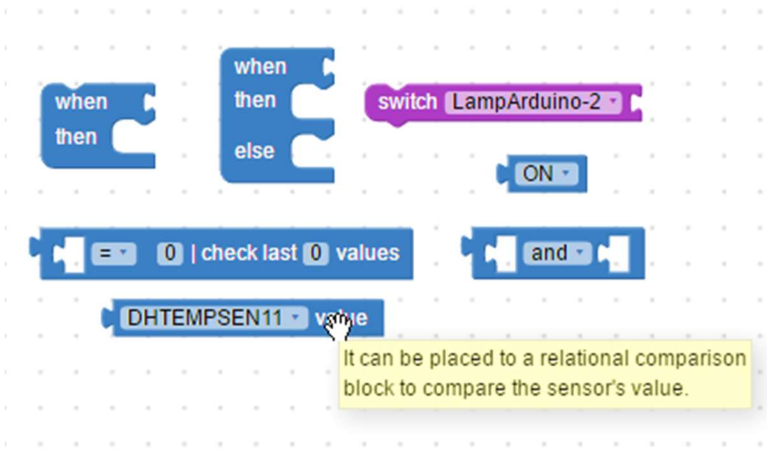
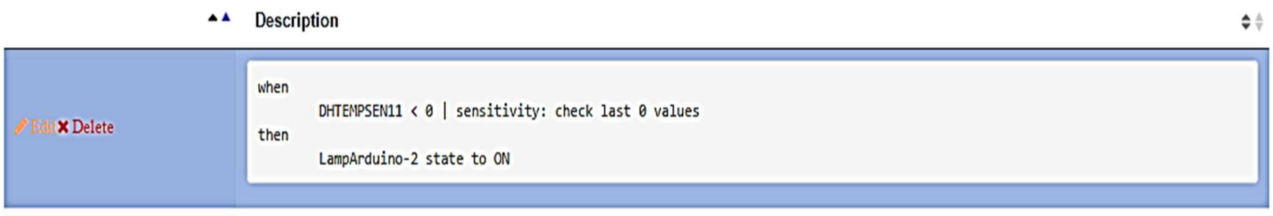

when
then
when
then
else
switch LampArduino-2
ON
= 0 | check last 0 values
and
DHTEMPSEN11 value
It can be placed to a relational comparison block to compare the sensor's value.

Figure 5. Available blocks for rule selection (and respective informative tooltip).



```
when
      DHTEMPSEN11 < 0 | sensitivity: check last 0 values
then
      LampArduino-2 state to ON
```

Figure 6. Rule description in textual format.

5. Implementation Details

We are using Arduino as the main microcontroller platform, although Rasbperry Pi, Intel Galileo, and BeagleBone have also been tested for collecting sensor data. All data sensor information is stored in the SensoMan database. Scaling to larger volumes of data may require a more scalable solution, especially if support for real-time data management is required, which is currently outside the scope of the current work [31]. Sensor value information, along with the rules specified by users, are stored in the database. The communication with the database and the boards is performed via RESTful services. SensoMan has also been connected with the H5CM context-aware middleware that provides an API for programming access to the SensoMan system for use in context-aware cross-platform applications that are supported by H5CM [24]. This was one of the most challenging techniques that was applied in SensoMan's implementation, as it concerned integration with an existing framework. RESTful services have also been used for the communication with the rule engine concerning the rule management and the communication with the actuators. User login and log out, sensor data management, actuator management, and microcontroller management, as well as rule management, are all covered by the use of RESTful services.

Registered users can connect to their accounts and view diverse information along with a map displaying the locations of her boards and sensors in close proximity to her current location. New board and sensor locations can be added via the user's dashboard, whereas update and delete actions are also available. Rules can be defined via the respective interface implemented with the use of Google's Blockly editor [32]. Finally, the monitoring of sensor measurements is provided. The web management dashboard has been implemented using web technologies (HTML, CSS, JavaScript, PHP) and more dedicated libraries (e.g., SweetAlert, DataTables). The SensoMan web platform is available in an online source code repository [33].

In terms of hardware equipment, different sensors can be supported. Experiments and testing of different scenarios have been performed using the indicative equipment listed in Table 3. Note that a web server was also used for implementation purposes, whereas additional sensors used in experiments are described in the initial version of SensoMan [10]. Examples of connections that include the equipment as used in the smart lightning scenario described in the next section are depicted in Figure 7.

Table 3. Indicative hardware equipment of SensoMan.

Equipment	Type	Description
Arduino Uno R3, Raspberry Pi	board	Microcontroller board.
Temperature and humidity	sensor	Adafruit Industries DHT and DHT22 temperature-humidity sensors.
Motion	sensor	Pir Sensor Module (can detect motion from up to 7 m).
Light	sensor	Simple photoresistor.
Gas	sensor	MQ gas sensor, and gas and smoke sensor MQ-2.
Infrared receiver	module	Can receive radio frequency of 433 MHz. It receives and demodulates a modulated radio-frequency signal using a super-regenerative receiver. Radio-frequency modules were used instead of infrared receivers, since no optical connection is required for connectivity.
Infrared transmitter	module	Can transmit radio-frequency signals with the data required in order to activate or deactivate a device and specifically, the containers attached to these devices. The transmitters used in the framework of SensoMan can transmit data without optical connection. As in the case of the receiver, the radio frequency of 433 MHz was used. The data are in the form of binary numbers with a specific number encoded as a device activation action and another number as a device deactivation action.
Controller plug	plug	Mercury Remote Controlled Plug is used in SensoMan. In order to be able to control a device using infrared, an infrared container is required. This needs to be attached to the device to allow its communication via infrared. Since most devices do not allow remote management or communicate only via infrared that requires optical connectivity, the use of the container ensures that the currency flow to the device connected to it is either enabled or disabled based on the triggered action.
Wi-Fi	shield	Required module for Internet connectivity. The module used (Adafruit CC3000 WiFi) supports 802.11b-g, open/WEP/WPA/WPA2 security, TKIP, and AES. The WiFi shields ensure that the collected data can be sent to the database, facilitating the connection to a WiFi access point. The board (i.e., Arduino), in our case,

		acts as a web server that allows communication in specific IP addresses and ports.
GPS	shield	Can provide location and time information (SparkFun GPS Shield—GPS-10710). The shield was used in conjunction with the TinyGPS++ library that allows the transformation of the values to geographical coordinates.

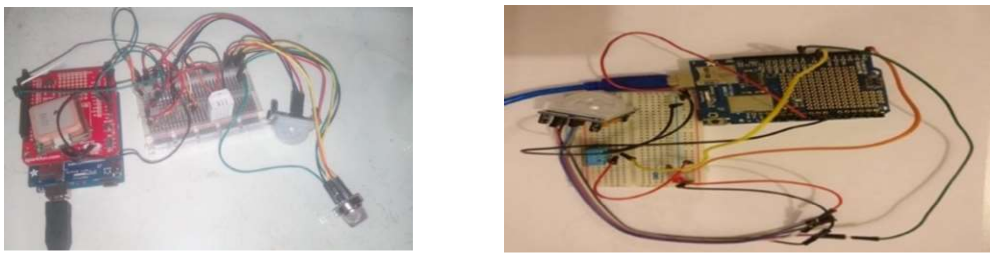


Figure 7. SensoMan sensor connections examples.

Security is a vital aspect in the SensoMan system. For this purpose, each user can perform calls to the RESTful services only for 3 min after login time. This is performed via the use of a token that comprises a specific hash key along with the time the token expires. This token information is checked every time the RESTful service is invoked via an HTTP call from the SensoMan web platform to ensure that unauthorized users cannot perform the invocation. The token acts as the user password for using the SensoMan services. The same token can be used in subsequent uses of the system by the same user (i.e., if the user logs out and then logs in again), as long as the token has not expired.

6. SensoMan Field Study and User Study

6.1. Smart Lightning Scenario

We are showcasing the use of SensoMan via two typical scenarios in the context of a field study conducted to show how SensoMan can be used in real scenarios. Energy management is an important issue nowadays. The management of energy consumption and attempts to decrease it in different environments using IoT solutions have been addressed in previous works for industrial use and use in smart building [34,35]. Smart lightning can provide a solution by controlling the lights on the streets (e.g., by optimizing the street lamp intensity according to the time of the day, the weather conditions, and the presence of people) and in buildings by means of different types of sensors and actuators that control lights [36]. In a similar fashion, being able to turn lights on and off accordingly based on the existing conditions of a room or a series of rooms in a building is a valuable measure for reducing energy consumption and relevant costs. Moreover, the automatization of such a procedure requires the use of specialized equipment. Even in cases where open-source microcontroller boards are used, the automatization requires technical knowledge in order to implement specific rules that allow turning lights on and off based on conditions in the environment.

In a home environment, SensoMan can assist in controlling lights by placing light sensors at specific locations inside the house with natural light. Existing lights can be smart-enabled by placing controller plugs on the normal plugs of all lights. These plugs will give the opportunity to control these lights with infrared signals. Using these signals, the current stream flow from the lights will be controlled. Light sensors can be connected to controllers with internet connectivity, so that the values from the sensors can be communicated to the SensoMan system. The same applies to the infrared transmitters that will send the appropriate signals to the controller plugs attached to the lights. Using the above connections, users are able to define rules for when to turn the lights on and off. If the conditions defined in the rules are not met, no action will be performed. Using this procedure, light management will be feasible at all times, whereas more parameters can be taken into consideration, such as movement detected in motion sensors, triggering, for instance, turning off the lights when a person exits a room.

The above scenario provides a typical use of SensoMan in a home or building or set of buildings (for instance, a university campus) and has been fully implemented in the framework of the current work (with connectivity as depicted in Figure 7). Additional functionality is supported via the system that allows the integration of new components, sensors, and devices.

The smart lightning scenario can be supported by rules defined using the SensoMan rule engine. The condition of activation/deactivation of the lights will be based on the measurements of the surrounding light sensors. Once the rule has been defined, it will be parsed each time new measurements for the light sensors become available. Based on the last measurements of a number of light sensors, the rule engine will decide on the action to be performed on one or more controller plugs in order to activate them or deactivate them or perform no action at all. The series of steps in this example using the SensoMan system is shown in Figure 8. A screenshot with a rule example is displayed in Figure 9. It indicates a simple rule about when to turn on the smart light using the last four values of another sensor. The user can select the microcontroller board to be used for the rule.

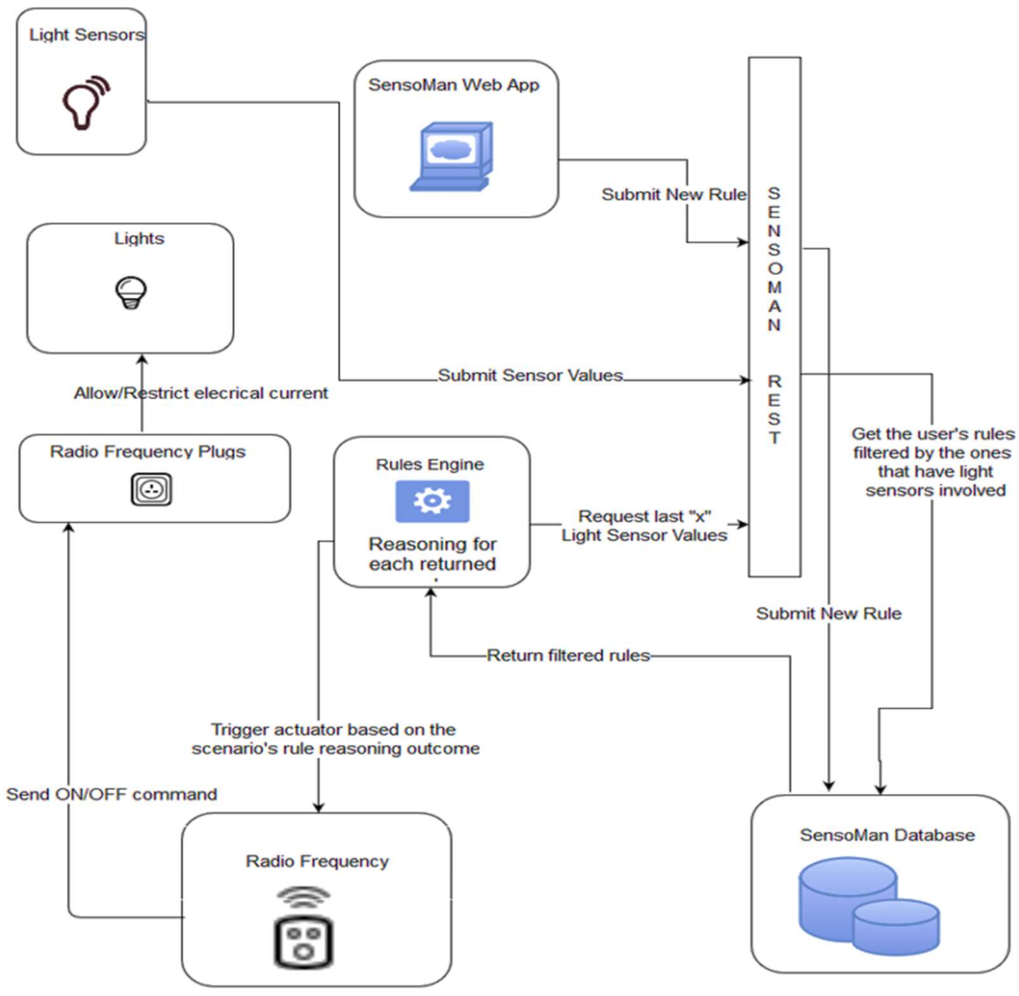
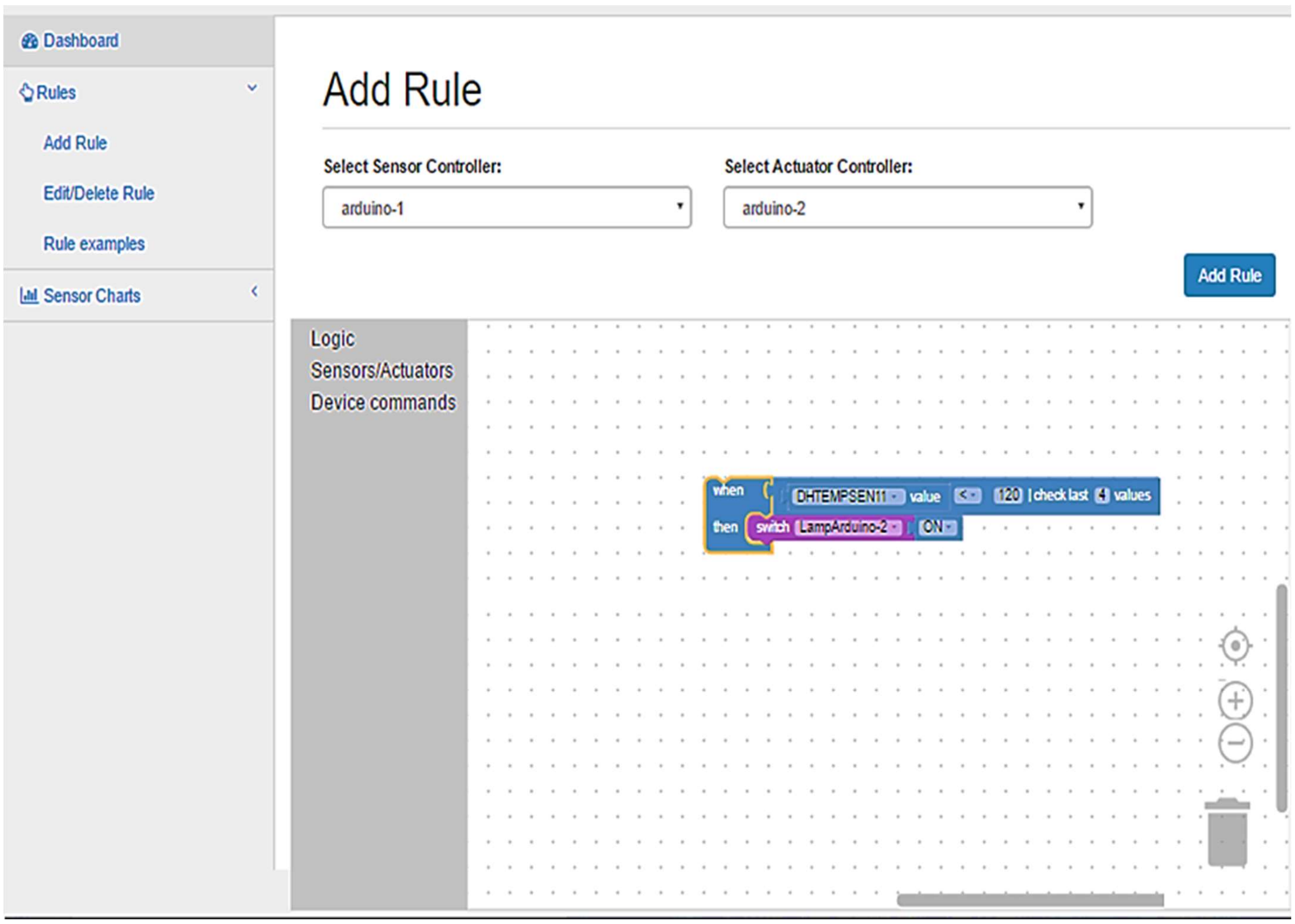

Light Sensors
SensoMan Web App
Submit New Rule
SENSOMAN REST
Lights
Submit Sensor Values
Allow/Restrict elecrical current
Radio Frequency Plugs
Rules Engine
Reasoning for each returned
Request last "x" Light Sensor Values
Get the user's rules filtered by the ones that have light sensors involved
Submit New Rule
Return filtered rules
Trigger actuator based on the scenario's rule reasoning outcome
Send ON/OFF command
Radio Frequency
SensoMan Database

Figure 8. Series of steps in the lightning control scenario.



Select Sensor Controller: arduino-1

Select Actuator Controller: arduino-2

when DHTEMPSEN11 value < 120 | check last 4 values
then switch LampArduino-2 ON

Figure 9. Rule creation in the smart lightning scenario.

6.2. Smart Greenhouse Scenario

SensoMan can find also use in temperature management via smart air conditions and heaters or integrated solutions for smart greenhouse management systems [37]. For the latter case, the temperature and the humidity of the greenhouse will be controlled via its environment, saving energy and deactivating (if needed) the watering of the plants to save water. Different temperature and humidity sensors are required for this scenario so that they can send measurements to the SensoMan RESTful service. An infrared transmitter and a radio frequency transmitter, along with a controller plug, need to be installed in all controlled systems (e.g., irrigation system, air conditioning system), acting in a similar fashion as in the smart lightning scenario.

The user needs to define the necessary rules using the above infrastructure. The control flow for this scenario is depicted in Figure 10, whereas the relevant rules and definitions are found in Figure 11.

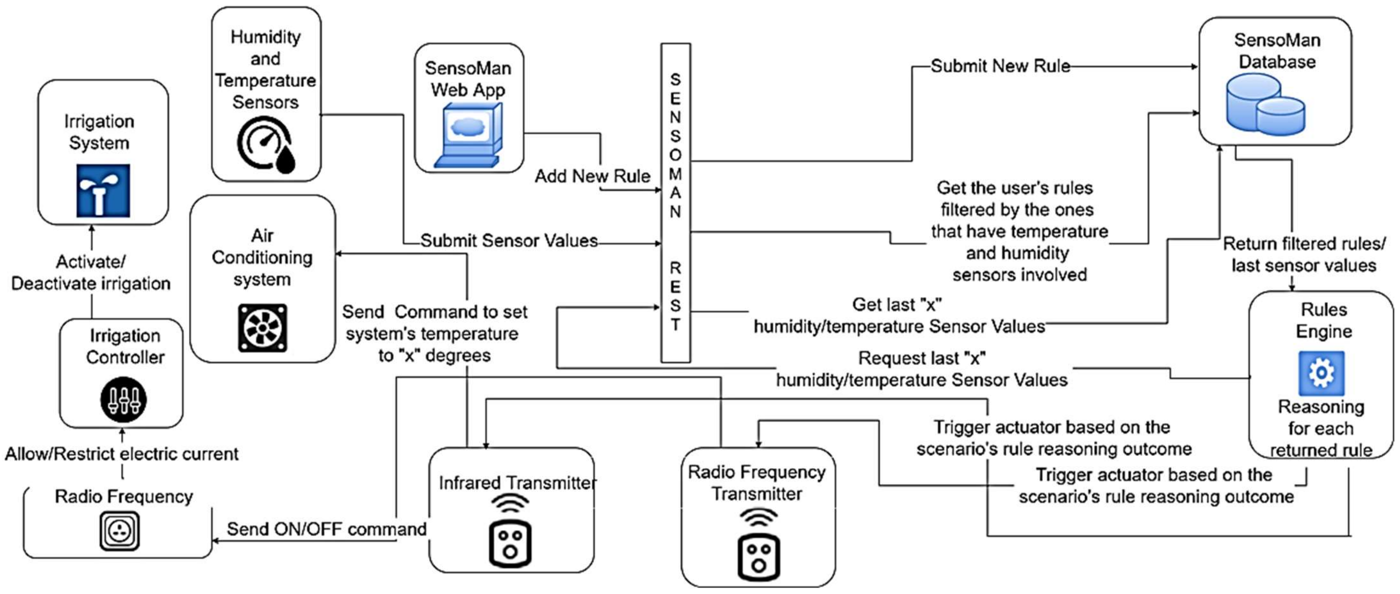
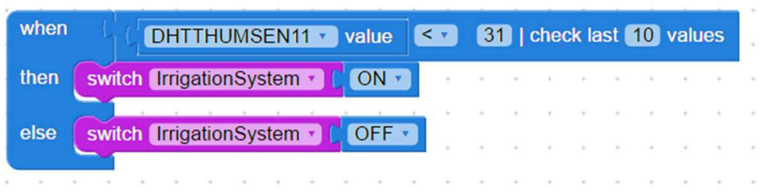

Humidity and Temperature Sensors
Irrigation System
SensoMan Web App
SENSOMAN REST
Submit New Rule
SensoMan Database
Add New Rule
Get the user's rules filtered by the ones that have temperature and humidity sensors involved
Submit Sensor Values
Air Conditioning system
Activate/ Deactivate irrigation
Return filtered rules/ last sensor values
Get last "x" humidity/temperature Sensor Values
Rules Engine
Irrigation Controller
Send Command to set system's temperature to "x" degrees
Request last "x" humidity/temperature Sensor Values
Reasoning for each returned rule
Trigger actuator based on the scenario's rule reasoning outcome
Allow/Restrict electric current
Infrared Transmitter
Radio Frequency Transmitter
Trigger actuator based on the scenario's rule reasoning outcome
Radio Frequency
Send ON/OFF command

Figure 10. Series of steps in the smart greenhouse scenario.


when DHTTHUMSEN11 value < 31 | check last 10 values
then switch IrrigationSystem ON
else switch IrrigationSystem OFF

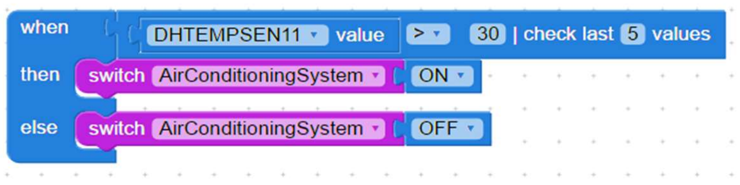
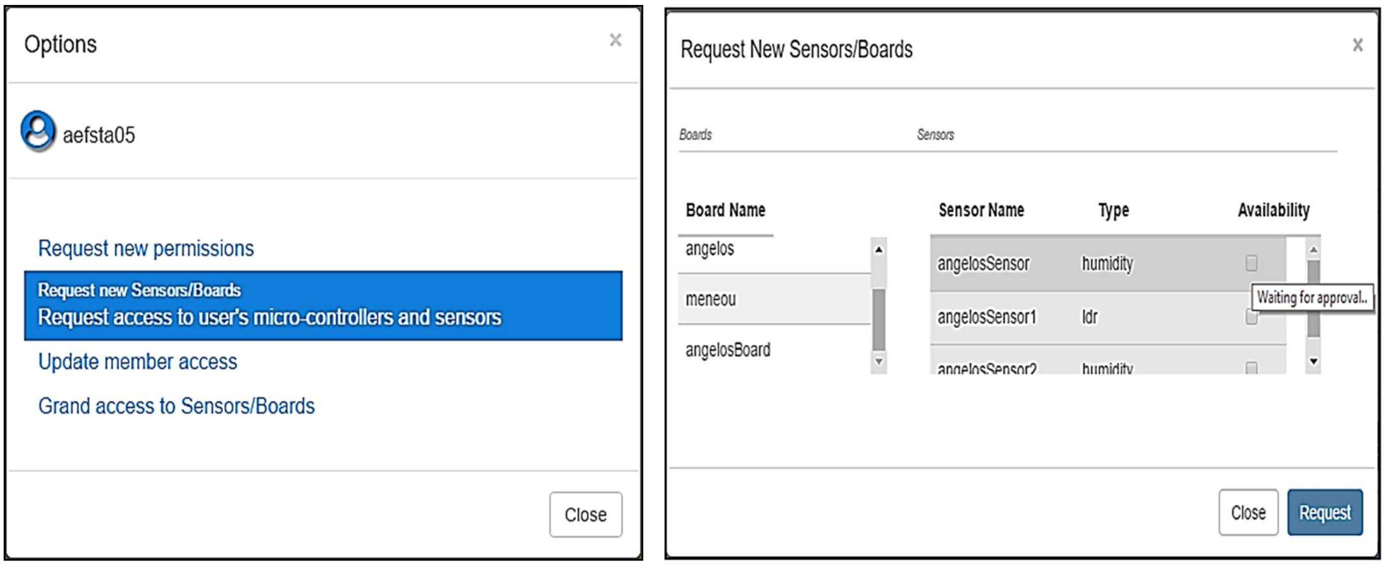
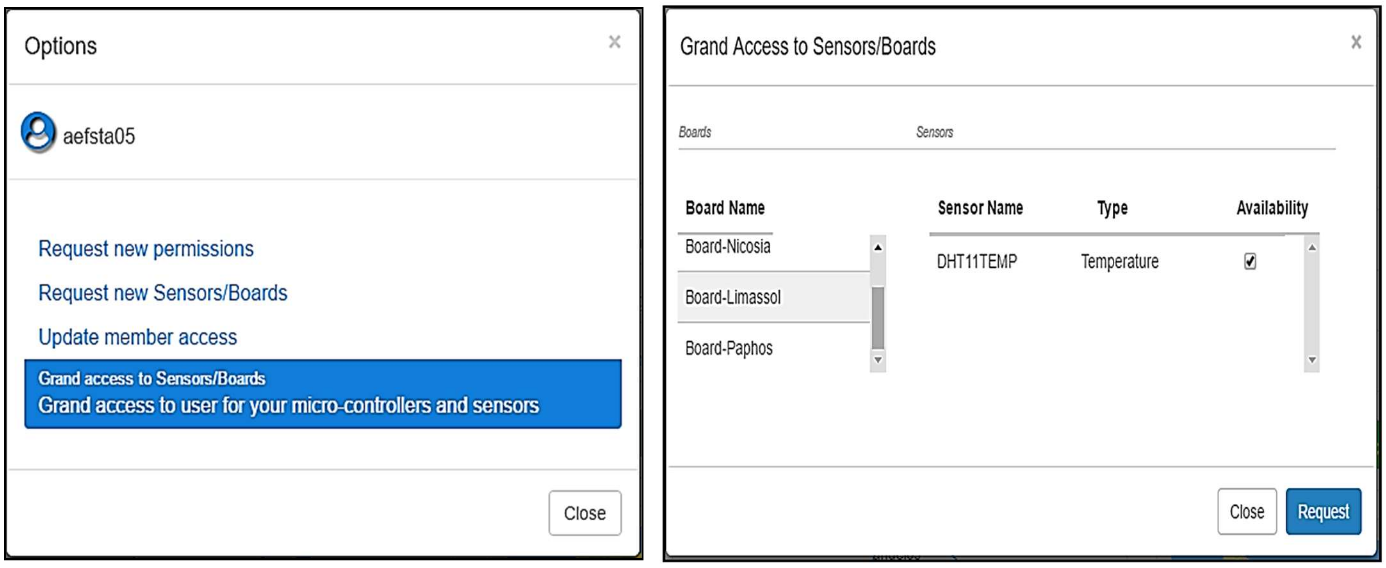

when DHTEMPSEN11 value > 30 | check last 5 values
then switch AirConditioningSystem ON
else switch AirConditioningSystem OFF

Figure 11. Rules definition in the smart greenhouse scenario.

In this scenario, the SensoMan social network can also be utilized, where users can share temperature and humidity sensors, with one user having this way the chance to observe the measurements of another greenhouse in the area with ideal conditions so that she can replicate the conditions in her greenhouse. Figure 12 depicts screenshots of the system dashboard, showing how a user can request access to a humidity sensor of another user (Figure 12a) and how the user can give access to a temperature sensor to another user (Figure 12b) selecting the appropriate microcontroller board and sensor.



(**a**)

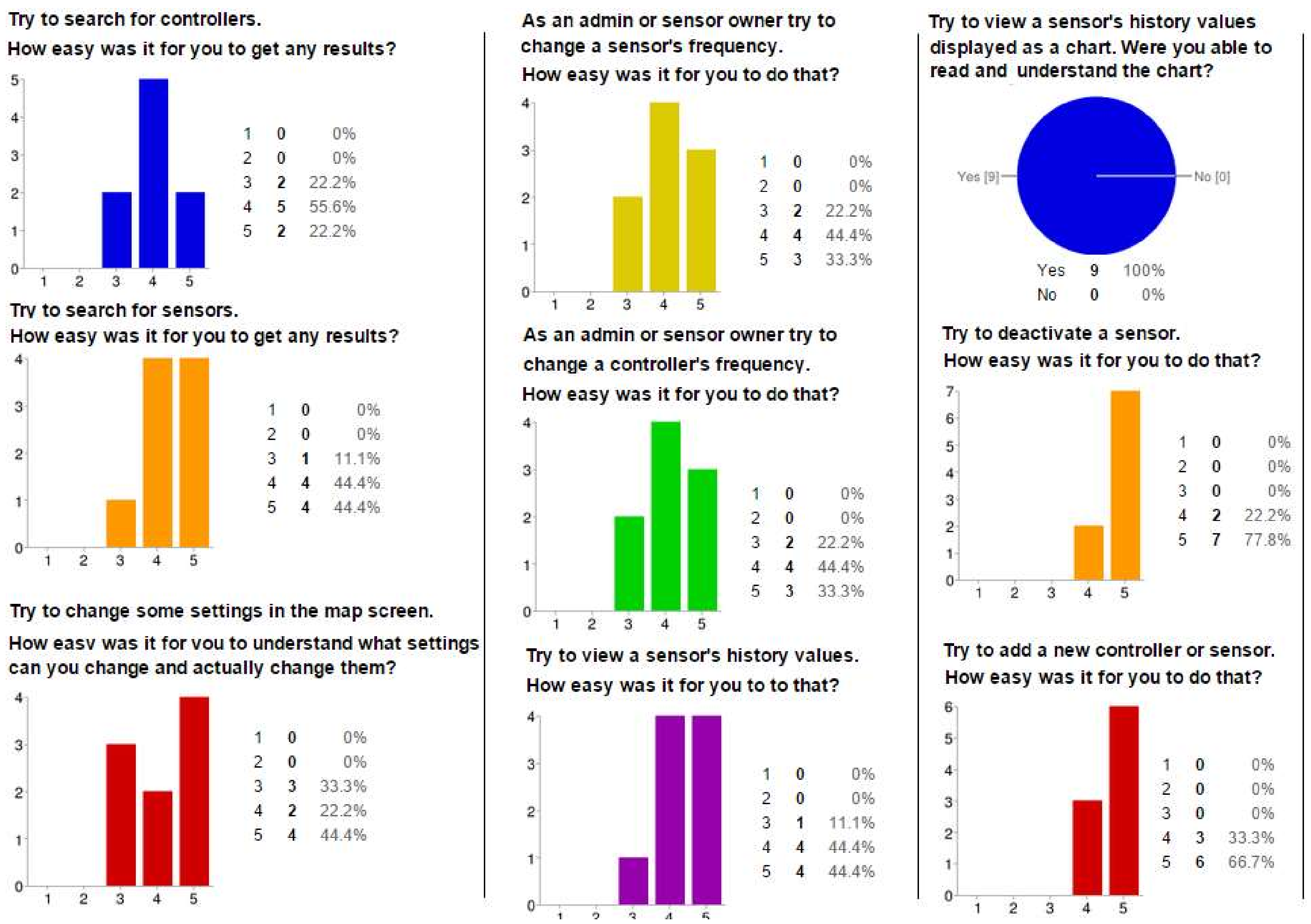


(**b**)

Figure 12. (**a**) Requesting and (**b**) granting access to sensors in the SensoMan social network (greenhouse scenario).

6.3. Small-Scale User Study

A small-scale user study was performed in order to assess the usefulness and usability characteristics of SensoMan's main features for sensor management. Students and young computer scientists with technical knowledge but no knowledge on the programming of microcontroller boards were recruited for this purpose. The users were asked to use SensoMan to perform simple tasks (e.g., add new sensors, change the frequency of data collection) and were then provided with a short questionnaire that assessed mainly the usability and ease of use of the SensoMan platform, specifically for performing the requested tasks. Nine (9) individuals participated in the user study, and most did not find any significant difficulty in using SensoMan. Figure 13 shows the user study results for nine example questions that concern different tasks to be performed by the users to assess usability and ease of use of SensoMan.



1	0	0%
2	0	0%
3	2	22.2%
4	5	55.6%
5	2	22.2%

1	0	0%
2	0	0%
3	1	11.1%
4	4	44.4%
5	4	44.4%

1	0	0%
2	0	0%
3	3	33.3%
4	2	22.2%
5	4	44.4%

1	0	0%
2	0	0%
3	2	22.2%
4	4	44.4%
5	3	33.3%

1	0	0%
2	0	0%
3	2	22.2%
4	4	44.4%
5	3	33.3%

1	0	0%
2	0	0%
3	1	11.1%
4	4	44.4%
5	4	44.4%

Yes	9	100%
No	0	0%

1	0	0%
2	0	0%
3	0	0%
4	2	22.2%
5	7	77.8%

1	0	0%
2	0	0%
3	0	0%
4	3	33.3%
5	6	66.7%

Figure 13. User study example results.

The average results clearly indicate that the usability and ease of use of the SensoMan platform is high, with changing the frequency of data collection for viewing the sensor values to be slightly more difficult. *Even more importantly though, regarding the question "How useful can this application be for you", 66.6% answered useful as very useful while 33.3% answered a little as moderately useful.* Nevertheless, the sample is too small to make statistically significant conclusions, and that is why the aim of determining the prospective use and adoption of SensoMan will lead to focusing on performing a more extended evaluation that will assess additional parameters and draw more generic conclusions.

7. Conclusions

In this paper, we have presented our work on SensoMan, which is a modular IoT management platform that can be used for the automatization of activities in collecting and triggering actions in the environment. SensoMan uses sensors and actuators to collect data and trigger actions accordingly based on rules defined by users in its rule engine. Its modules are based on open-source tools, whereas it supports the use of different sensors connected on microcontroller boards, rendering the infrastructure extensible with more sensors to cover various scenarios. SensoMan users may access their boards but may also have access to measurements from other users' sensors when connected to these users via the SensoMan social network. We have demonstrated the use of SensoMan showing its applicability via a field study using two scenarios that showcase the use of the rule engine and of the social network, whereas a small-scale user study was also conducted. We argue that especially users with limited technical expertise would profit from the use of the platform, as it is based on a simple architecture, and users have the opportunity to utilize the parts of the system that best suit their needs (e.g., SensoMan social network).

As future work, we intend to focus on the real-time management of sensor data, adding relevant support to SensoMan. We are also working toward a more elaborated use of actuators that is not limited to device activation/deactivation but includes also other actions, such as setting a specific temperature to an air conditioning system via the use of infrared codes. We are further working toward the provision of an elaborated reasoning mechanism that would allow users to utilize sensor values in their applications combining them with machine learning techniques and other sources of information in order to draw useful conclusions on the data that cannot be detected when viewing or using the data independently. The integration of SensoMan with other frameworks including openHAB is also a useful direction of work. We also intend to address scalability issues that may emerge if the number of users, sensors, actors, and rules increase significantly, and we perform an extended user evaluation covering all functionalities of SensoMan. Finally, we will integrate security and data protection mechanisms following the General Data Protection Regulation (GDPR), as this is an important emerging area in IoT [38–40].

Author Contributions: Conceptualization, G.M.K. and A.P.A.; methodology, G.M.K.; software, P.A., A.C.P.; validation, G.M.K., A.P.A., P.A. and A.C.P.; investigation, G.M.K., A.P.A., P.A. and A.C.P.; writing—original draft preparation, G.M.K.; writing—review and editing, A.P.A.; visualization, P.A.; supervision, G.M.K.; project administration, G.M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to thank Angelos Efstathiou for his insights on the SensoMan social network.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Borgia, E. The Internet of Things vision: Key features, applications and open issues. *Comput. Commun.* **2014**, *54*, 1–31.
2. Kim, K.D.; Kumar, P.R. Cyber–physical systems: A perspective at the centennial. *Proc. IEEE* **2012**, *100*, 1287–1308.
3. Elazhary, H. Internet of Things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions. *J. Netw. Comput. Appl.* **2019**, *128*, 105–140.
4. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, ACM, Helsinki, Finland, 17 August 2012; pp. 13–16.
5. Abowd, G.; Dey, A.; Brown, P.; Davies, N.; Smith, M.; Steggles, P. Towards a better understanding of context and context-awareness. In *Handheld and Ubiquitous Computing*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 304–307.
6. Perera, C.; Zaslavsky, A.; Christen, P.; Georgakopoulos, D. Context aware computing for the internet of things: A survey. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 414–454.
7. Stankovic, J.A. Research directions for the internet of things. *IEEE Internet Things J.* **2014**, *1*, 3–9.
8. openHAB. Available online: https://www.openhab.org/ (accessed on 7 October 2021).
9. Eclipse IoT. Available online: https://iot.eclipse.org/ (accessed on 7 October 2021).
10. Paphitou, A.C.; Constantinou, S.; Kapitsaki, G.M. SensoMan: Remote management of context sensors. In Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, Larnaca, Cyprus, 13–15 July 2015; 6p.
11. Kaufmann, B.; Buechley, L. Amarino: A toolkit for the rapid prototyping of mobile ubiquitous computing. In Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services, Lisbon Portugal, 7–10 September 2010; pp. 291–298.
12. Soldatos, J.; Kefalakis, N.; Hauswirth, M.; Serrano, M.; Calbimonte, J.P.; Riahi, M.; Aberer, K.; Jayaraman, P.P.; Zaslavsky, A.; Žarko, I.P.; et al. Openiot: Open source internet-of-things in the cloud. In *Interoperability and Open-Source Solutions for the Internet of Things*; Springer: Cham, Switzerland, 2015; pp. 13–25.
13. Parocha, R.C.; Macabebe, E.Q. Implementation of home automation system using OpenHAB framework for heterogeneous IoT devices. In Proceedings of the 2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), IEEE, Bali, Indonesia, 5–7 November 2019; pp. 67–73.

14. Chen, R.; Bao, F.; Guo, J. Trust-based service management for social internet of things systems. *IEEE Trans. Dependable Secur. Comput.* **2015**, *13*, 684–696.
15. Hussein, D.; Han, S.N.; Lee, G.M.; Crespi, N.; Bertin, E. Towards a dynamic discovery of smart services in the social internet of things. *Comput. Electr. Eng.* **2017**, *58*, 429–443.
16. Ali, S.; Kibria, M.G.; Jarwar, M.A.; Lee, H.K.; Chong, I. A model of socially connected web objects for IoT applications. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 20.
17. Roopa, M.S.; Pattar, S.; Buyya, R.; Venugopal, K.R.; Iyengar, S.S.; Patnaik, L.M. Social Internet of Things (SIoT): Foundations, thrust areas, systematic review and future directions. *Comput. Commun.* **2019**, *139*, 32–57.
18. Ortiz, A.M.; Hussein, D.; Park, S.; Han, S.N.; Crespi, N. The cluster between internet of things and social networks: Review and research challenges. *IEEE Internet Things J.* **2014**, *1*, 206–215.
19. Pintus, A.; Carboni, D.; Piras, A. Paraimpu: A platform for a social web of things. In Proceedings of the 21st International Conference on World Wide Web, Lyon, France, 16–20 April 2012; pp. 401–404.
20. Paganelli, F.; Mylonas, G.; Cuffaro, G. A RESTful Rule Management Framework for Internet of Things Applications. *IEEE Access* **2020**, *8*, 217987–218001.
21. Node-RED. Available online: https://nodered.org/ (accessed on 7 October 2021).
22. IFTTT. Available online: https://ifttt.com/ (accessed on 7 October 2021).
23. Zapier. Available online: https://zapier.com/ (accessed on 7 October 2021).
24. Achilleos, A.P.; Kapitsaki, G.M. Enabling cross-platform mobile application development: A context-aware middleware. In *Proceedings of the International Conference on Web Information Systems Engineering*; Springer: Cham, Switzerland, 2014; pp. 304–318.
25. Ellison, N.B. Social network sites: Definition, history, and scholarship. *J. Comput.-Mediat. Commun.* **2007**, *13*, 210–230.
26. Ruta, M.; Scioscia, F.; Loseto, G.; Pinto, A.; Di Sciascio, E. Machine learning in the Internet of Things: A semantic-enhanced approach. *Semant. Web* **2019**, *10*, 183–204.
27. Kapitsaki, G.M.; Prezerakos, G.N.; Tselikas, N.D.; Venieris, I.S. Context-aware service engineering: A survey. *J. Syst. Softw.* **2009**, *82*, 1285–1297.
28. Complete Guide to Actuators. Available online: http://www.thomasnet.com/about/actuators-301168.html (accessed on 7 October 2021).
29. Hosek, J.; Masek, P.; Kovac, D.; Ries, M.; Kropfl, F. Universal smart energy communication platform. In Proceedings of the 2014 International Conference on Intelligent Green Building and Smart Grid (IGBSG), Taipei, Taiwan, 23–25 April 2014; pp. 1–4.
30. MIT App Inventor. Available online: http://appinventor.mit.edu/explore/ (accessed on 7 October 2021).
31. Kandt, J.; Batty, M. Smart cities, big data and urban policy: Towards urban analytics for the long run. *Cities* **2021**, *109*, 102992.
32. Blockly. Available online: https://developers.google.com/blockly/ (accessed on 7 October 2021).
33. SensoMan Repository. Available online: https://bitbucket.org/paziz001/sensoman (accessed on 7 October 2021).
34. Shrouf, F.; Ordieres, J.; Miragliotta, G. Smart factories in Industry 4.0: A review of the concept and of energy management approached in production based on the Internet of Things paradigm. In Proceedings of the 2014 IEEE International Conference on Industrial Engineering and Engineering Management, Selangor Darul Ehsan, Malaysia, 9–12 December 2014; pp. 697–701.
35. Constantinou, S.; Vasileiou, A.; Konstantinidis, A.; Chrysanthis, P.K.; Zeinalipour-Yazti, D. IMCF: The IoT Meta-Control Firewall for Smart Buildings. In Proceedings of the 24th International Conference on Extending Database Technology (EDBT), Nicosia, Cyprus, 23–26 March 2021; pp. 658–661.
36. Mehmood, Y.; Ahmad, F.; Yaqoob, I.; Adnane, A.; Imran, M.; Guizani, S. Internet-of-things-based smart cities: Recent advances and challenges. *IEEE Commun. Mag.* **2017**, *55*, 16–24.
37. Tripathy, P.K.; Tripathy, A.K.; Agarwal, A.; Mohanty, S.P. MyGreen: An IoT-Enabled Smart Greenhouse for Sustainable Agriculture. *IEEE Consum. Electron. Mag.* **2021**, *10*, 57–62.
38. Kounoudes, A.D.; Kapitsaki, G.M. A mapping of IoT user-centric privacy preserving approaches to the GDPR. *Internet Things* **2020**, *11*, 100179.
39. Dammak, M.; Aroua, S.; Senouci, S.M.; Ghamri-Doudane, Y.; Suciu, G.; Sachian, M.A.; Roscaneanu, R.; Gungor, M.O. A Secure and Interoperable Platform for Privacy Protection in the Smart Hotel Context. In Proceedings of the 2020 Global Information Infrastructure and Networking Symposium (GIIS), Tunis, Tunisia, 28–30 October 2020; pp. 1–6.
40. Kafle, K.; Moran, K.; Manandhar, S.; Nadkarni, A.; Poshyvanyk, D. Security in Centralized Data Store-based Home Automation Platforms: A Systematic Analysis of Nest and Hue. *ACM Trans. Cyber-Phys. Syst.* **2020**, *5*, 1–27.