





Article

Prototyping IoT-Based Virtual Environments: An Approach toward the Sustainable Remote Management of Distributed Mulsemmedia Setups

Telmo Adão ^{1,2,*} , Tatiana Pinho ³, Luís Pádua ^{3,4} , Luís G. Magalhães ², Joaquim J. Sousa ^{3,4} 
and Emanuel Peres ^{3,4} 

¹ Graphic Computation Center (CCG), University of Minho Campus de Azurém, Edifício 14, 4800-058 Guimarães, Portugal

² Centro Algoritmi, Universidade do Minho Campus de Azurém, Av. da Universidade, 4800-058 Guimarães, Portugal; lmagalhaes@dsi.uminho.pt

³ Centre for Robotics in Industry and Intelligent Systems (CRIIS), INESC Technology and Science (INESC-TEC), 4200-465 Porto, Portugal; tatiana.m.pinho@inesctec.pt (T.P.); luispadua@utad.pt (L.P.); jjsousa@utad.pt (J.J.S.); eperes@utad.pt (E.P.)

⁴ Engineering Department, School of Science and Technology, University of Trás-os-Montes e Alto Douro, 5000-801 Vila Real, Portugal

* Correspondence: telmoadao@utad.pt; Tel.: +351-253-510-580

Abstract: Business models built upon multimedia/multisensory setups delivering user experiences within disparate contexts—entertainment, tourism, cultural heritage, etc.—usually comprise the installation and in-situ management of both equipment and digital contents. Considering each setup as unique in its purpose, location, layout, equipment and digital contents, monitoring and control operations may add up to a hefty cost over time. Software and hardware agnosticity may be of value to lessen complexity and provide more sustainable management processes and tools. Distributed computing under the Internet of Things (IoT) paradigm may enable management processes capable of providing both remote control and monitoring of multimedia/multisensory experiences made available in different venues. A prototyping software to perform IoT multimedia/multisensory simulations is presented in this paper. It is fully based on virtual environments that enable the remote design, layout, and configuration of each experience in a transparent way, without regard of software and hardware. Furthermore, pipelines to deliver contents may be defined, managed, and updated in a context-aware environment. This software was tested in the laboratory and was proven as a sustainable approach to manage multimedia/multisensory projects. It is currently being field-tested by an international multimedia company for further validation.

Keywords: IoT; multimedia; multisensory; mulsemmedia; IoT prototyping; virtual environments; simulation; system planning



Citation: Adão, T.; Pinho, T.; Pádua, L.; Magalhães, L.G.; J. Sousa, J.; Peres, E. Prototyping IoT-Based Virtual Environments: An Approach toward the Sustainable Remote Management of Distributed Mulsemmedia Setups. *Appl. Sci.* **2021**, *11*, 8854. <https://doi.org/10.3390/app11198854>

Academic Editor: KeeHyun Park

Received: 2 September 2021

Accepted: 21 September 2021

Published: 23 September 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The Internet of Things (IoT) rests on the premise that each and every common object may be equipped with identification, sensing and communication (networking) mechanisms. It would enable information exchange—without human intervention—between objects and also with Internet-based services, to achieve a given goal. The evolution of both the Internet and the World Wide Web, as well as all the development accomplished around consumer technologies (e.g., tablets, smartphones) ended up by defining IoT as what it is known for today: a massive network of different types of elements that interchange data. Among them and besides the usual tablets and smartphones, media players, watches, bracelets, cars, wearables, sensors, actuators, appliances, surveillance cameras and others can also be found to be a part of IoT networks [1]. Indeed, the number of applications that fall under the IoT technological umbrella have been growing so fast—in areas that include

precision agriculture (e.g., [2]), security and surveillance (e.g., [3]), smart cities (e.g., [4]), and multimedia/multisensory environments (e.g., [5,6])—that by 2020, connected elements are expected to surpass 50 billion [7–9].

A recent survey focusing IoT for multimedia [8] presents a comprehensive review on the subject, in which some architectures are referred: agent/multi-agent [10,11], artificial intelligence [12], fog-cloud [13], as well as big data capable approaches [14] and security-based schemes [15]. In [7], an architecture for IoT multimedia (IoTMM) that brings together modern cloud services and mobile edge computing (MEC) is proposed, highlighting the importance of characteristics such as scalability, flexibility, efficiency and interoperability. IoTMM device level, IoTMM-to-cloud, IoTMM-to-MEC-to-cloud, or additional intermediate layers constitute different available options. Application and research areas are numerous. For example, in [16] traffic flow optimization was addressed resorting to cameras, ontologies and Bayesian networks. Furthermore, ocular recognition-based authentication has been reported as a suitable technology for IoT domain [17]. Within smart industry, wearable devices and Bluetooth were combined to determine visitors' position inside a museum and present cloud-based artwork information [18]. Regarding E-health, an IoT-based system to train surgeons with existing data and real-time feedback [19] is just an example among many other available solutions. Other works focusing on IoT-based user-centered multisensory approaches can be found in the literature. For example, in [6], a home entertainment system was the reference to an architecture proposal. It consists of a four-layer architecture—application, aggregation, virtualization, and physical—that intends to define templates integrating virtual objects (device properties abstraction) with Micro-Engines (i.e., tiny sensors that enable connectivity and control in all the “things”). Concerned with the multiplicity of sensors, actuators, and computation devices that can be found in multisensory IoT systems, in [20], a design proposal was based on service-oriented architecture for alleviating heterogeneity issues (interoperability of components, portability, scalability, etc.). Proof of concept was made with a RGB camera and smart insoles, focusing identity recognition. In the health context, a monitoring system based in blockchain, multiple sensors and multimedia big data was proposed to monitor physiological and mental states of cancer patients, with medical real-time decision support capabilities [21]. Also, the “Magic Room” proposed in [22] reports an IoT-based multisensory system to develop therapies for children with neurodevelopmental disorder. It consists in completely customizable environments that allow stimuli combination through “smart” objects control. Electrocardiogram devices, wearable sensors, location systems and machine learning have been associated to IoT to monitor patients' heart-rate while performing distinct physical activities. Targeting therapeutic interventions, but also novel didactic methods, Magika was reported as an inclusive system that allows educators to interact with children by stimulating their tactile, auditory, visual, and olfactory senses, through IoT-based digital projections and physical smart objects [23]. Other studies that use multisensory interaction in citizen science stream monitoring training have also been reported [24]. Hotel hosting is another example of a potential business model that can benefit from multisensory IoT. Both managers and guests associate IoT-based sensory experiences to an increase in satisfaction and loyalty levels [25].

The combination of multimedia and multisensory can be defined as mulsemedia [26]. Alone, these environments have great potential to deliver outstanding experiences to users based on human senses exploration (usually, more than one), but when combined with adequate distributed architectures, can unlock very important features, such as sustainable scalability, full availability in terms of data access, as well as large content flow and bandwidth management. Notwithstanding this, planning and design are very important steps to achieve success in any project, and those scoped within IoT mulsemedia are not an exception. Such steps have the goal to not only guide further development stages, but also ensure specification adequacy to stakeholders' requirements, as well as to mitigate the waste of resources. Whilst purpose and location are certainly decisive in laying out a mulsemedia setup, they are also key factors when planning and creating each experience.

The results are setups that can be composed of different software, hardware and contents combinations. Moreover, when delivering context-aware interactive experiences—time-, spatial- or sensor-based—contents may change and be experienced through one or more of the human senses, which implies being delivered by different types of hardware. All of this adds up to the complexity of planning, designing, implementing, managing, maintaining and updating mulsemmedia setups.

Planning capacity, what-if hypothesis formulation and analysis/proactive management can be explored through IoT simulation environments, as highlighted in [27], which proposed a design-oriented approach, using as a case-study smart services for smart cities and decentralized areas. It was pointed out that multi-level simulation techniques provide means to mimic wide geographical areas, with a multitude of replica-based agents. A significant variety of reported works followed the same research line. For example, Sensym1 [28] consists of an environment capable of simulating large-scale data during application development, in which multiple types of sensors with physical and geographical characteristics similar to real devices can be hosted, allowing performance to be tested under edge conditions (power consumption, data growth) and to evaluate scale-up potential. Focusing on IoT big data processing, in [29], a platform IOTSim was designed and implemented upon known cloud simulation services—such as CloudSim [30]—with the goal of allowing the analysis of the impact and performance of IoT-based applications before a real system setup. In [31], a cloud-centric IoT application store was designed and implemented to host virtual objects of different IoT domains, which can be integrated by technologists to build IoT applications requiring little or no modification. Relying on the specific Devices Profile for Web Services (DPWS) standard, DPWSim [32] is a Java-based prototyping tool designed to speed up the simulation of IoT applications. Admittedly, more work related to the generation of virtual devices is required for a better acceptance of this simulator by its target audience, which is Industry. More focused on interaction, in [33], the new generation of VR technology (head-mounted displays and game engines) was explored in the form of a prototyping tool to control and monitor a virtual smart home environment.

This paper considers the apparent scarcity of works addressing the specific topic of IoT-based mulsemmedia setups simulation and proposes a prototyping software towards the suppression of such gap, which is comprised of: (1) a web-based application to manage virtual IoT setups; (2) “things” side virtual environment; and (3) MQTT connections to wire everything. This work was developed within the Cooperative Holistic view on Internet and Content (CHIC) project scope. It is a funded initiative that, among other goals, aims at the digital innovation of multimedia field at both industrial and entrepreneurial levels, with the purpose to develop tools capable of providing enhanced multisensory virtual/augmented/mixed reality experiences based on setups that may be sustainably remotely managed. The following main aspects were prioritized in this work:

- to settle agnostic specification for IoT simulation—as far as possible independent from hardware and software—aiming to lessen complexity in providing projects of sustainable management;
- to provide flexible programming-oriented processes capable of ensuring scalability for devices and respective functionalities;
- to align a general IoT simulation solution that, although, can meet mulsemmedia setups specific requirements;
- to outline interfaces and engagement structures envisaging the integration of digital twin-based approaches.

Rather than covering IoT security issues, this paper limits its focus to IoT simulations under an architectural/structural/setup perspective, more centered in stimulus delivery pipeline, either based on event scheduling or through sensor condition triggering. However, the transposition to real-world deployments must encompass suitable levels of protection across IoT layers, considering different security aspects, as addressed, for example, in [34].

This paper continues by presenting the proposed IoT mulsemmedia environments prototyping software architecture, followed by implementation details. Demonstrations, conclusions, and future work are drawn at the end of the document.

2. Prototyping Software Architecture

As previously stated, this work was developed within the CHIC project scope. As such, it is important to briefly present it to provide readers with a more thorough context. CHIC aimed to develop, test, and demonstrate a set of innovative processes, products, and services for both the audiovisual and multimedia sectors. Due to the underlying transversality, results are expected to have an impact in other important business sectors within culture's scope (e.g., cultural heritage, exhibiting arts). An extended set of activities was carried out toward three essential domains: (i) open platforms for the management and distribution of digital contents, supported by the cloud; (ii) management of cultural heritage related contents, based in open-systems for preservation and interaction; and (iii) content creation, production and consumption focusing quality of service and experience, resorting to very high-definition immersive environments.

One of the main tasks established within CHIC's scope was the development of an independent IoT system as-a-service, qualified to enable the swift configuration of scalable mulsemmedia setups and their remote management. These mulsemmedia setups must support the configuration of groups of wireless mulsemmedia equipment—"things", such as displays, fans, lights, smell delivery devices, varied types of sensor, etc.—within a given interior space, i.e., an interactive room. To provide meaningful experiences to users and induce engagement, involved device sets—mostly, actuators—are expected to work on-demand, in an orchestrated and time-effective manner, to adequately deliver a specified stimulus. Considering the requirements laid out, Figure 1 presents the proposed architecture for the IoT mulsemmedia environments prototyping software. It is composed of a virtual environment of "things" and a remote-control panel, which are connected by a publishing/subscriber (P/S) messaging protocol.

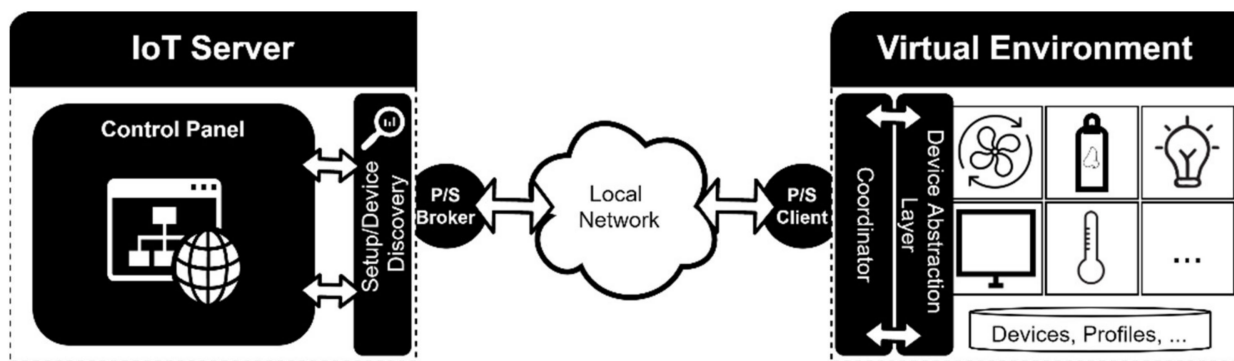


Figure 1. Architecture overview of the IoT mulsemmedia environments prototyping software. It is composed of a control server and, at least, one virtual environment of "things", managed by a virtual coordinator.

The virtual environment can be regarded as an IoT mulsemmedia environment virtual representation, providing both a software and hardware agnosticity layer—device abstraction layer—as well as a virtual representation of each device that is a part of the different mulsemmedia experiences. Moreover, each virtual device representation has a profile containing its properties and available functions. As for the IoT server, it enables remote and on-demand control and management capabilities through a context-aware integrated discovery service (setup/device discovery). Communication between both elements is assured by a P/S protocol, specifically the ISO/IEC 20922 standard, also known as message queue telemetry transport (MQTT). The P/S broker is placed in the server side, as usual.

2.1. IoT Server

The IoT server provides an interface to remotely monitor, control and manage devices that are a part of a virtual multimedia environment. It is composed of two essential components: (i) a setup discovery service responsible for obtaining information about connected virtual configurations grouping virtual devices; and (ii) a control panel—through a graphical interface—with the software’s available operations, namely to obtain virtual devices’ features, manage control packets, schedule actuators, and define sensor-based events. A web-browser establishes the applicational layer for this service, allowing direct access to the control panel graphical interface.

2.2. Virtual Environment

The virtual environment has a coordinator that interfaces with incoming messages and provides appropriate responses. It is supported by a device abstraction layer that handles queries and matches them with the targeted virtual equipment to retrieve data (e.g., current feature status) or to prompt state alteration events (e.g., activate artificial smell device). As for data model support, the virtual environment keeps track of virtual devices and their relation with profiles, features, states, and data (Figure 2). A virtual device represents equipment with both an internet protocol (IP) address and a media access control (MAC) address. Each virtual device is also associated with a profile, i.e., settings describing the device’s interface. These settings are mapped into a structure called feature (power, rotation, video content, sensing channel, etc.) that, in turn, can have a set of states (e.g., on/off, fan rotation level). The data structure keeps a record of timestamped and device-aware state alterations.

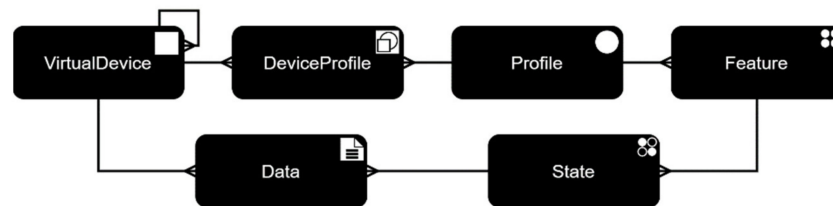


Figure 2. Data model proposed for the IoT multimedia prototyping software. It is formed by many associations involving virtual devices (representing real equipment), profiles (or capabilities descriptors) grouping features and states, and data for handling timestamped action requests (e.g., turn on a lamp, set display to reproduce a video) to virtual devices.

VirtualDevice has a context-sensitive graphic user interface (GUI) that changes according to its state. Moreover, it can be further characterized in one of two types: *Sensor* or *Actuator*. While the former consists in a device that simulates environmental variables measurement (e.g., temperature, motion), the latter represents equipment capable of delivering stimuli (e.g., wind, light, heat, cold). Figure 3 presents the relations between *VirtualDevice*’s elements.

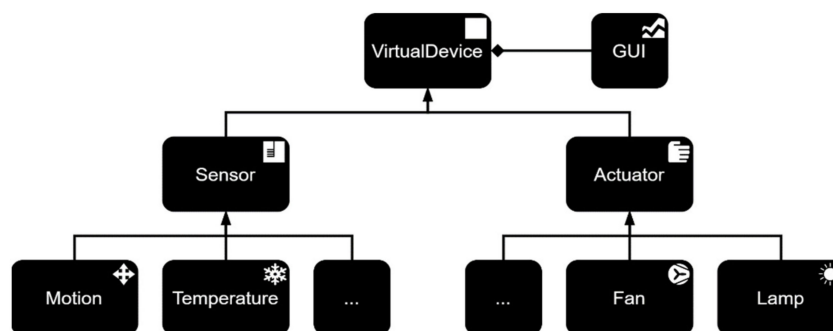


Figure 3. *VirtualDevice* structure and generalization: it is composed of a representational state and it can be specialized in *Sensor* or *Actuator*.

Details about communication between the IoT Server and the virtual environment will be provided in the next subsection.

2.3. IoT Server/Virtual Environment Communication

Resorting to a P/S protocol and following a simple messaging strategy, essential capabilities for the communication of both IoT server and virtual environment are ensured. Whenever a new virtual environment is put together, a kick-off acknowledgment message composed of the coordinator’s MAC address and a verification code is sent to the IoT server. There, the message is handled by an authenticated user—with management access—that carries out the virtual environment registration in the system. Once that happens, four main inter-operations can be launched by the IoT server (Figure 4):

- Devices request: retrieves a list of endpoints and respective capabilities associated to a given virtual environment coordinator;
- Control packet request: orders a condition change in a virtual environment’ actuator;
- Actuators scheduling: configures a relative time-based event list to chronologically engage actuators’ capabilities in a virtual environment;
- Sensor-based event triggering: configures an event listener that is looking out to the virtual environment’ sensors values. When a sensor measures the defined value, an automatic action is triggered in the virtual environment’s actuators.

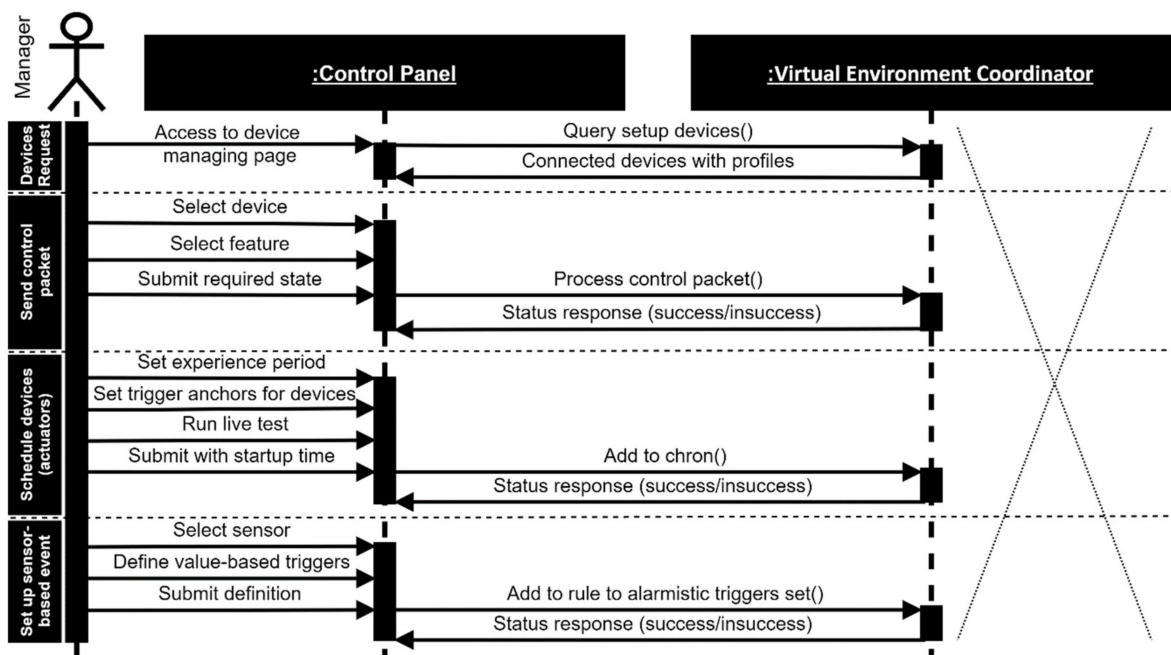


Figure 4. Sequence diagram depicting the four possible interactions between the IoT server (via control panel interface) and the virtual environment (using the virtual environment coordinator), regarding control and managing activities: (1) *devices request* returns a list of usable equipment; (2) *send control packet* results in a device state switching; (3) in *schedule devices*, a time-based rules set (*trigger anchors*) is defined to establish a limited temporal sequence for device state switching; finally, (4) *set up sensor-based event* shows the process to specify triggers for switching the state of a given device as a consequence of another device behavior—considering that, typically, a sensor can influence an actuator—as long as both belong to the same setup.

The next section will present an implementation proposal for the IoT mulsemmedia environments prototyping software.

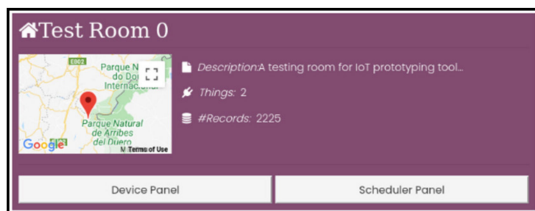
3. IoT Mulsemmedia Environments Prototyping Software

A proof-of-concept implementing the architectural elements described in Section 2 is used, so that this section may be more perceptible to the reader. It comprised a web-

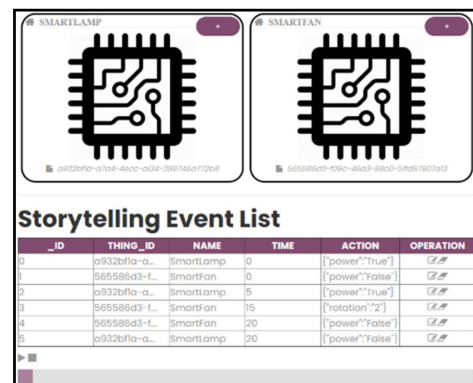
based application manager (IoT Server) and a couple of virtual environments—2D and 3D—wrapping up virtual devices that simulate sensors’/actuators’ capabilities. Whilst the application manager provides a control panel to remotely manage events in the “things” side, each virtual environment implements a coordinator that acts as a request dispatcher engaged with a set of associated virtual devices. Moreover, it also binds a virtual device’ state with a corresponding graphical representation, for visual intuition purposes. A local MQTT broker needs to be kept alive to enable connectivity between both ends. For this implementation, Mosquitto 1.5.31 release—compatible with Windows 10—was used.

3.1. MQTT Enhanced Web-Based Application

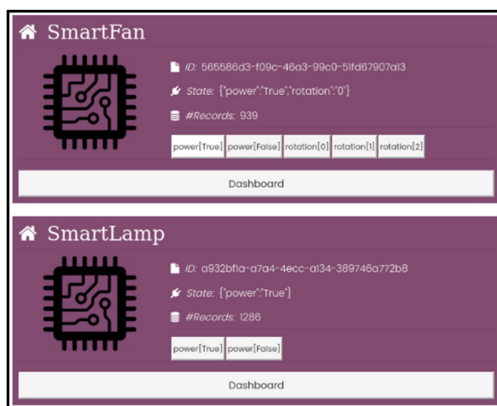
Figure 5 presents the developed web-based application manager. Its modules—control panel and dashboard—were developed using modern programming languages and scripting tools PHP, JavaScript, resulting in an Internet-based client-server approach to enable access through common web browsers. Furthermore, it resorts to a Mongo database to store useful information, mainly for synchronization purposes with the virtual environment side (e.g., setup registration, virtual devices’ status, event-driven rules). MQTT over WebSocket ensures connectivity between both sides. Paho-MQTT JavaScript was the library adopted on the server side.



(a)



(b)



(c)

power	rotation	ts
True	2	06/11/2020, 00:06:41
True	2	06/11/2020, 00:06:36
True	2	06/11/2020, 00:06:31
True	2	06/11/2020, 00:06:26
True	2	06/11/2020, 00:06:21
True	2	06/11/2020, 00:06:16
True	2	06/11/2020, 00:06:11
True	2	06/11/2020, 00:06:06
True	2	06/11/2020, 00:06:01
True	2	06/11/2020, 00:05:56
True	2	06/11/2020, 00:05:51
True	2	06/11/2020, 00:05:46
True	2	06/11/2020, 00:05:41
True	2	06/11/2020, 00:05:36
True	2	06/11/2020, 00:05:31

(d)

Figure 5. Web-based application manager: (a) general access panel to a room of test, corresponding to a virtual setup; (b) event scheduler to configure actuators features activation in a chronological manner; (c) control panel interfacing endpoints’ features at the virtual setup side; (d) data table regarding the registries associated to the activity of an exemplifying device (smart lamp).

The web-based application manager enables the registration of new virtual environments. These are represented by a coordinator (handshake) through both a pseudo-MAC

address and a verification code, which may be validated by the user responsible for the configuration process. After a successful pairing procedure, the application manager gains access to resources associated to the virtual environment coordinator, specifically virtual devices, and respective capabilities. Through MQTT requests—that also declare a topic to which the coordinator answers back—the IoT server side can perform queries concerning virtual devices for data retrieval purposes, as well as send actuating commands. Table 1 presents the defined message format.

Table 1. IoT server-side publishing topics for communication with subscriptions assigned in the virtual environments side. Different topics/messages are available: “info” topic can be used to retrieve a whole virtual environment; “out” topic requests a state alteration in an endpoint at the virtual environment side; “rule” topic specifies an actuator trigger according to a given sensor’ behavior, at the virtual environment side; “query” topic variants can be issued to retrieve sensors’ readings and actuators’ actions, integrally or confined to a time interval. Note that in each case, an expiring subscription topic with a timestamp (/response/#timestamp) is specified in the messages, allowing the coordinator of the targeted virtual environment side to reply.

Topic	Message	Description
/info/{c}	/response/#timestamp	Requests coordinator features, as well as associated endpoints along with respective capabilities.
/out/{e}/{f}/{s}	#value /response/#timestamp [optional]	Requests a state change regarding a given virtual coordinator’s endpoint.
/rule/{e}/{f}/{s}	{“expression”: #exp, “trigger”: #trigger} /response/#timestamp	Specifies a rule for a given virtual coordinator that consists in opportunely triggering an endpoint action in response to another’s behavior.
/query/{e}, /query/{e}/{f}, query/{e}/{f}/{s}	/response/#timestamp #optional_time_range	Retrieves data (readings/actions) from a given coordinator/endpoint/feature, optionally framed by a time range.

{c}—Virtual coordinator ID (MAC); {e}—Virtual endpoint ID (MAC); {f}—Feature ID; {s}—State ID; #timestamp—Unix-based timestamp taken at runtime; #value—mutable attribution inducing a change in a given endpoint state; #exp—simple mathematical expression defined to establish a rule for a given feature associated to a virtual coordinator/endpoint device; #trigger—state change repercussion in a feature of a given virtual coordinator/endpoint device as a response to a verification of an #expression_rule; #optional_time_range—definition of an hyphen separated time interval (year1/month1/day1 hour1:minute1:second1–year2/month2/day2 hour2:minute2:second2).

3.2. Virtual Environment

The prototyping software’s part simulating the “things” side is built upon 2D and 3D virtual environments composed of entities role-playing coordination, actuation and sensing devices, thus inducing real object-like behaviors. Virtual devices’ profiles are stored in a JSON file, along with their respective features and states (summarized in Table 2). This aims not only to define a general template for profiles management, but also to provide a data source to smoothly associate virtual devices during code-based programmatic specification. These associations allow capabilities for virtual devices to be set up, which on the IoT simulation/application boot are signed up in the midway coordinator as MQTT services, thus enabling end-to-end bidirectional communication.

Behavioral simulation relies on a hierarchical folder structure concordant with set up JSON-based profiles, wherein the assets are placed. Representations for 2D environments are images, while wavefront files (.obj format) storing 3D models’ configurations are preferred for 3D environments. Folders and files are organized according to the path format “./{profile_id}/{feature_id}/{state_id}”, where state_id must have a set of possible representational assets, named after the value they stand for, considering the value set specified in Table 2, for each feature and profile. Whenever a virtual device state changes, an adjustment in the graphical representation—associated asset—is immediately reflected, accordingly the type of entity involved. While sensors change their state with a pseudo-random approach that selects a set of supported values—established by the JSON-based profiles archive—and considering an interval in seconds, actuators are affected on-demand. Figure 6 depicts the process associated to virtual devices’ state changing.

Table 2. Profiles, features, and states considered for the proposed IoT mulsemmedia environments prototyping software proof-of-concept, mapped in a JSON structure: Lamp, Fan, Display, Temperature and Motion.

Profile		Feature			State		
ID	Name	ID	Name	Type	ID	Name	Value Set
1	Lamp	1	Power	Actuator	1	Switch	1 On 0 Off
2	Fan	1	Rotation	Actuator	1	Level	0 Off 1 Low 2 Medium 3 High
3	Display Comics	1	Power	Actuator	1	Switch	1 On 0 Off
		2	Video	Actuator	1	Content	DC Marvel
4	Temperature	1	Value	Sensor	1	Degrees	13 18 25 32
5	Motion	1	Value	Sensor	1	Condition	0 inactive 1 active

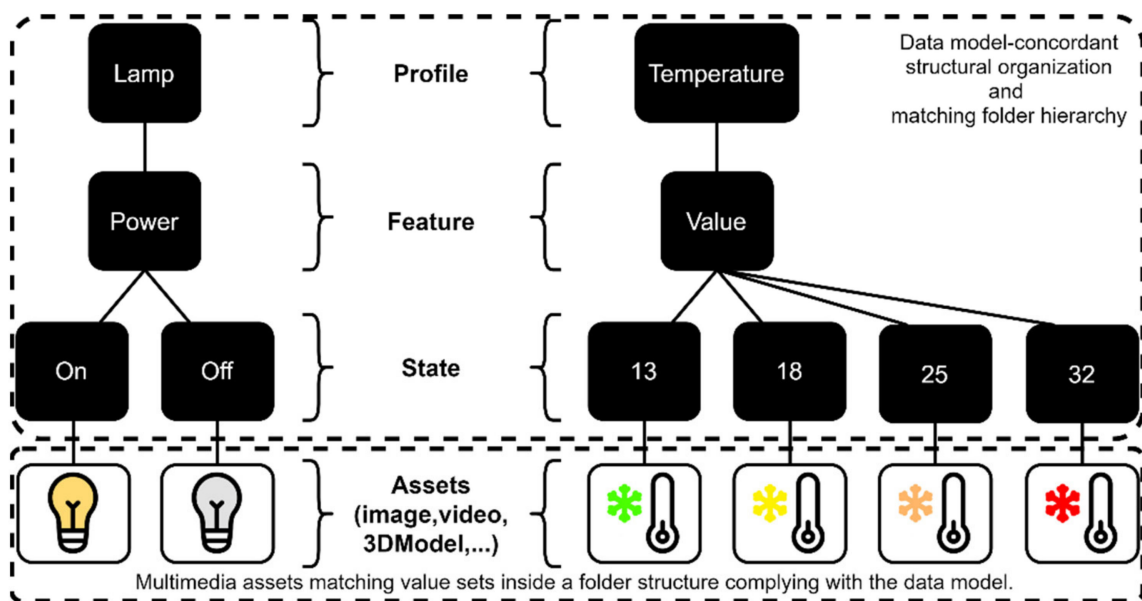


Figure 6. A pair of virtual devices associated to different profiles—an actuator and a sensor—exemplifying things’ state changing process: lamps show a representation of a turned on or off bulb according to a given power state transition, while temperature sensors toggle between different levels of heat, considering a pseudo-random value selected from the respective profile’s predefined set, for instance 13, 18, 25, and 32 (in °C).

Two-dimensional environment implementation was carried out in Python (v. 3.7), installed in a Linux Ubuntu virtual machine. A coordinator object was set up as an MQTT-based (Paho-mqtt 1.5.1. Available in <https://pypi.org/project/paho-mqtt/>, accessed on 1 December 2020) dispatcher that replies to queries and redirects control requests to associated virtual devices (please revisit Table 1). These operate as follows: actuators subscribe MQTT topics for each profile-based assigned feature; sensors are pseudorandom-based agents that select values out of a profile-defined range, publishing them to a proper co-

ordinator's MQTT topic. The part regarding the graphical representation reflecting the state of each virtual device is implemented over Python's Tkinter library (Tkinter. Available in <https://docs.python.org/3/library/tkinter.html>, accessed on 1 December 2020). To run a simulation within the software, one must simply execute, asynchronously: (1) the server file, holding an instance of the coordinator; (2) a script with basic instances of the required virtual devices. The instantiation of these prebuilt objects is done by simply code writing *Server()*, *Fan()*, *Motion()*, etc. in the aforementioned Python scripting files. A Mongo database keeps track of the data exchange within the environment of virtual devices. Figure 7 depicts virtual device examples working on the prototyping software.

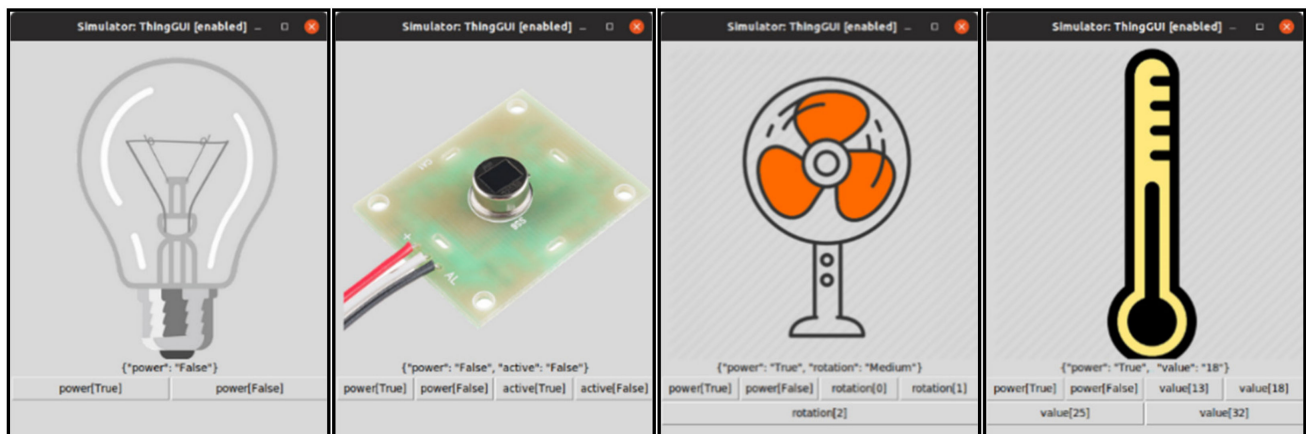


Figure 7. Two-dimensional environment virtual devices with associated profiles. From left to right: a lamp that can be turned on and off, a passive infrared (PIR) sensor that randomly changes its state at 30 s intervals from idle to activated and vice versa, a fan with three levels of rotation and a temperature sensor with four heat levels that randomly toggles each 60 s.

Three-dimensional environment implementation was addressed resorting to Unity 3D (Unity 3D. Available in <https://unity.com/>, accessed on 1 December 2020)—2018.x version—, following a similar approach as the proposed in the 2D environment. A coordinator manages querying and controls messages. It is wired by a MQTT library (M2Mqtt. Available in <https://github.com/eclipse/paho.mqtt.m2mqtt>, accessed on 1 December 2020), which is the protocol that also suits the associated virtual devices that subscribe topics for state manipulation—in case of actuation profile—or publish state messages to the main broker as sensors. For each prebuilt device set for instantiation—by simply code writing *new Lamp()*, *new Display()*, and so on in the Unity's *MonoBehavior* class responsible for graphically managing environment actions and events—representations are assumed that rely on 3D models (wavefront files) loaded from locally stored folders structures that respect the relation between profile, feature, and state, as described above. In fact, the possible 3D models regarding a given virtual device profile are imported into the environment as an object group organized according to the structure “*{device_id}/{profile_id}/{feature_id}/{state_id}*” when the application boots up, facilitating the efficient representational switching. Import operations are ensured by ObjReader plugin (Unity Asset Store—The Best Assets for Game Making | ObjReader. Available in <https://assetstore.unity.com/packages/tools/input-management/objreader-152>, accessed on 1 December 2020), which allows 3D models to be integrated from local storage, in execution time. Due to the powerful abstractions for handling complex computer graphics transformations provided by Unity, classes providing devices' visual representations can be quick and easily scripted for simple animations (e.g., fan rotation). For the particular case of displays, a folder acting as a video repository also needs to be specified. Figure 8 depicts a fully functional IoT virtual environment.

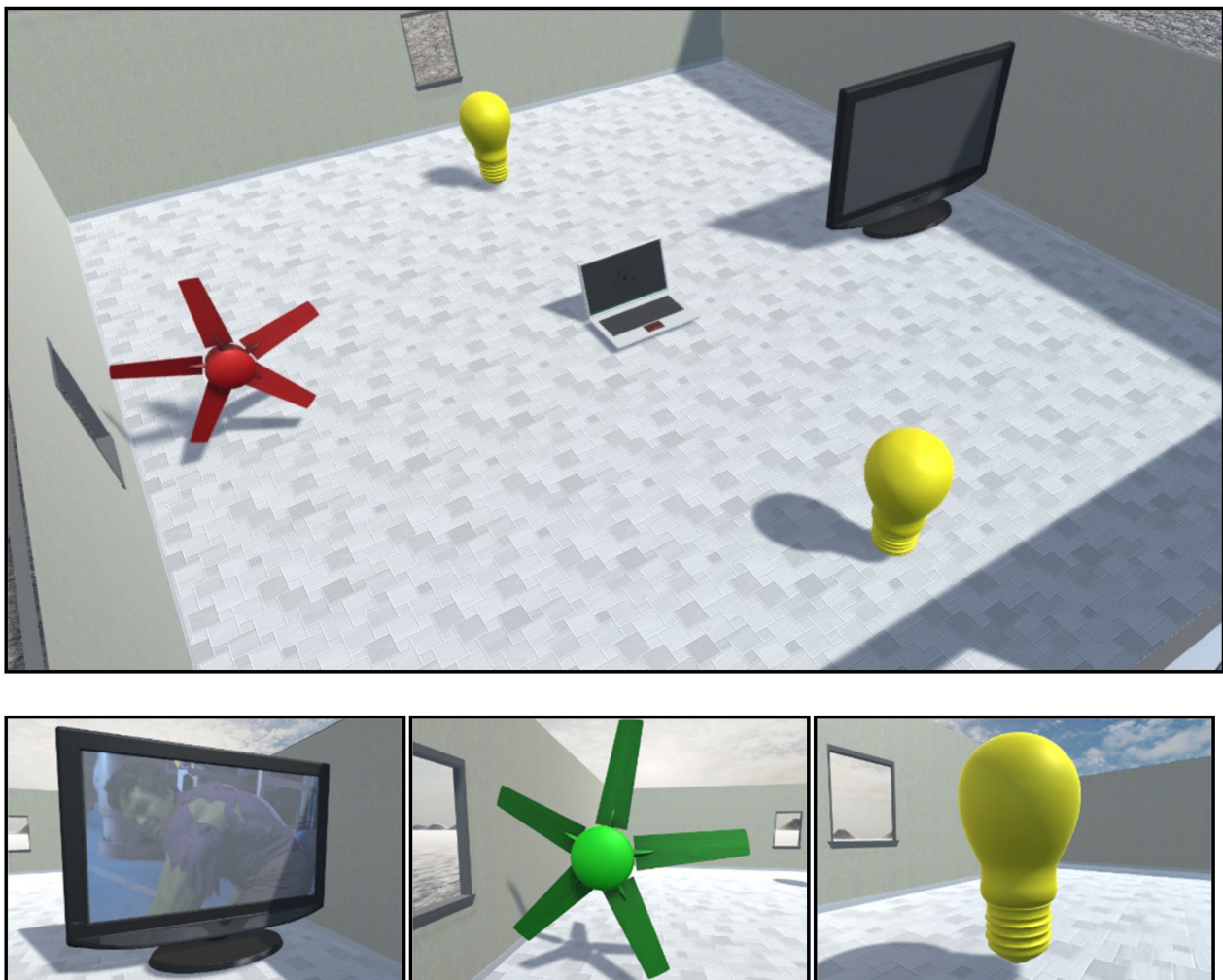


Figure 8. A 3D virtual environment composed of a coordinator and profiled virtual devices. A display, a pair of lamps, and a fan can be identified.

For both virtual environments—2D and 3D—the inclusion of new devices is relatively easy to achieve. It takes two steps: (i) specialization from base classes actuator or sensor; and (ii) ID association to a profile lying in the already mentioned JSON-based list. New profiles can also be added to the existing set, along with the desired representational assets properly placed, following the previously suggested folder organization.





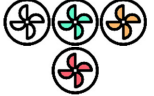






The next section will present the results regarding both the 2D and 3D IoT virtual environments, built with the proposed prototyping software.

4. Results

A set of 2D and 3D testing scenarios was established to demonstrate both event and schedule-oriented behavior modelling. On the one hand, event-oriented operations specify trigger-based dependencies between virtual devices, i.e., an influence chain among simulation virtual devices. On the other hand, schedule-oriented execution relies in defining associations between actuators' capabilities to time-based key-points. To provide a better interpretation of the possible behaviors that can be assumed by the involved virtual entities, contextual symbology was defined and adopted along this section, according to Table 3. Such entities are of three types: coordinator (central manager/dispatcher), sensor (temperature and motion), and actuator (lamp, fan, and display). This section ends with performance tests providing an overview of time consumption associated with local

MQTT connections, with a few variations in the number of publishing requests and target subscription topics.








Table 3. Demonstration symbology regarding the 2D and 3D virtual environments established by using the proposed prototyping software.

Representation	Environment	Designation	Type	Possible States
	2D/3D	Coordinator	Manager/Dispatcher	N/A
	2D/3D	Lamp	Actuator	
	2D/3D	Fan	Actuator	
	3D	Display	Actuator	
	2D	Temperature	Sensor	
	2D	Motion	Sensor	

4.1. Testing Scenario 1: 2D Virtual Environment Schedule-Oriented Behavior Modelling

The 2D virtual environment is composed of a coordinator, two lamps, two fans, a temperature sensor, and a motion detector. Each is identified with a unique random MAC address. Table 4 presents the details of the virtual devices associated with this scenario.











Table 4. Two-dimensional virtual environment devices. Lamp, fan, temperature, and motion detection profiles were used to set up this virtual environment.

Representation	Name	Identifier
	Test Room 2D	56-DB-F6-6E-84-37
	Lamp A	A8-DD-A4-75-FF-EE
	Lamp B	AE-85-48-8D-A8-F3
	Fan A	E3-8D-9E-C2-6E-2D
	Fan B	9B-F7-F4-42-16-1E
	Temperature	6E-A4-07-13-FE-CD
	Motion	7B-03-4D-95-AB-8E

To test the event-oriented control through the definition of activation rules that trigger actuators' state changing, a scenario was specified accordingly to Table 5. While a set of rules regarding the temperature sensor was set up to induce different levels of rotation in *Fan A*, variations in the motion sensor were considered as triggers for *Lamp A*. Indeed, a temperature control rules set was intentionally established: the occurrence of the two highest levels of temperature triggers the fan to a medium rotation level, while the lowest temperature events lead the fan to be turned off. Following the exact same process,

a motion monitoring rules set was defined to induce the lamp to be turned off or turned on, according to the motion sensor state.

Table 5. Two-dimensional virtual environment devices: event-oriented control testing scenario.

Description	Request	JSON	Events	Trigger
Temperature Control	/rule/6E-A4-07-13-FE-CD/1/1	{"exp": ">18", "trigger": "/out/E3-8D-9E-C2-6E-2D/1/1 Medium"}	 or 	
	/rule/6E-A4-07-13-FE-CD/1/1	{"exp": "<=18", "trigger": "/out/E3-8D-9E-C2-6E-2D/1/1 Off"}	 or 	
Motion Monitoring	/rule/7B-03-4D-95-AB-8E/1/1	{"exp": "=1", "trigger": "/out/A8-DD-A4-75-FF-EE/1/1 On"}		
	/rule/7B-03-4D-95-AB-8E/1/1	{"exp": "=0", "trigger": "/out/A8-DD-A4-75-FF-EE/1/1 Off"}		

The configuration of a virtual scenario exemplifying actuators scheduling can be observed in Table 6. A lamp (Lamp B) and a fan (Fan B) were the virtual devices in this mulsemmedia experience simulation, which was configured to last 30 s. Initially, both virtual devices were set up to be turned off. Some time-based key-points were defined to show their possible variation of states, which end up being turned off when the simulation finishes.

Table 6. Two-dimensional virtual environment devices: schedule-oriented control testing scenario.









Description	Time (s)	Request	Output
Initialization	0	/out/AE-85-48-8D-A8-F3/1/1 Off	
Initialization	0	/out/9B-F7-F4-42-16-1E/1/1 Off	
Turn on lamp B	5	/out/AE-85-48-8D-A8-F3/1/1 On	
Set fan B rotation to 1	10	/out/9B-F7-F4-42-16-1E/1/1 Low	
Set fan B rotation to 2	20	/out/9B-F7-F4-42-16-1E/1/1 Medium	
Set fan B rotation to 3	25	/out/9B-F7-F4-42-16-1E/1/1 High	
Turn off lamp B	30	/out/AE-85-48-8D-A8-F3/1/1 Off	
Turn off fan B	30	/out/9B-F7-F4-42-16-1E/1/1 Off	

Figure 9 depicts the set of devices used in the 2D environment testing scenario, wherein the variation of states can be observed for both lamps and fans, as well as for temperature and motion sensors.

The next subsection will also address actuators scheduling but focusing on the 3D virtual environment.

4.2. Testing Scenario 2: 3D Things' Virtual Environment

The 3D environment being presented is composed of a coordinator, two lamps, a fan, and a display. Each one of these virtual devices has an attributed random MAC address, as presented in Table 7.

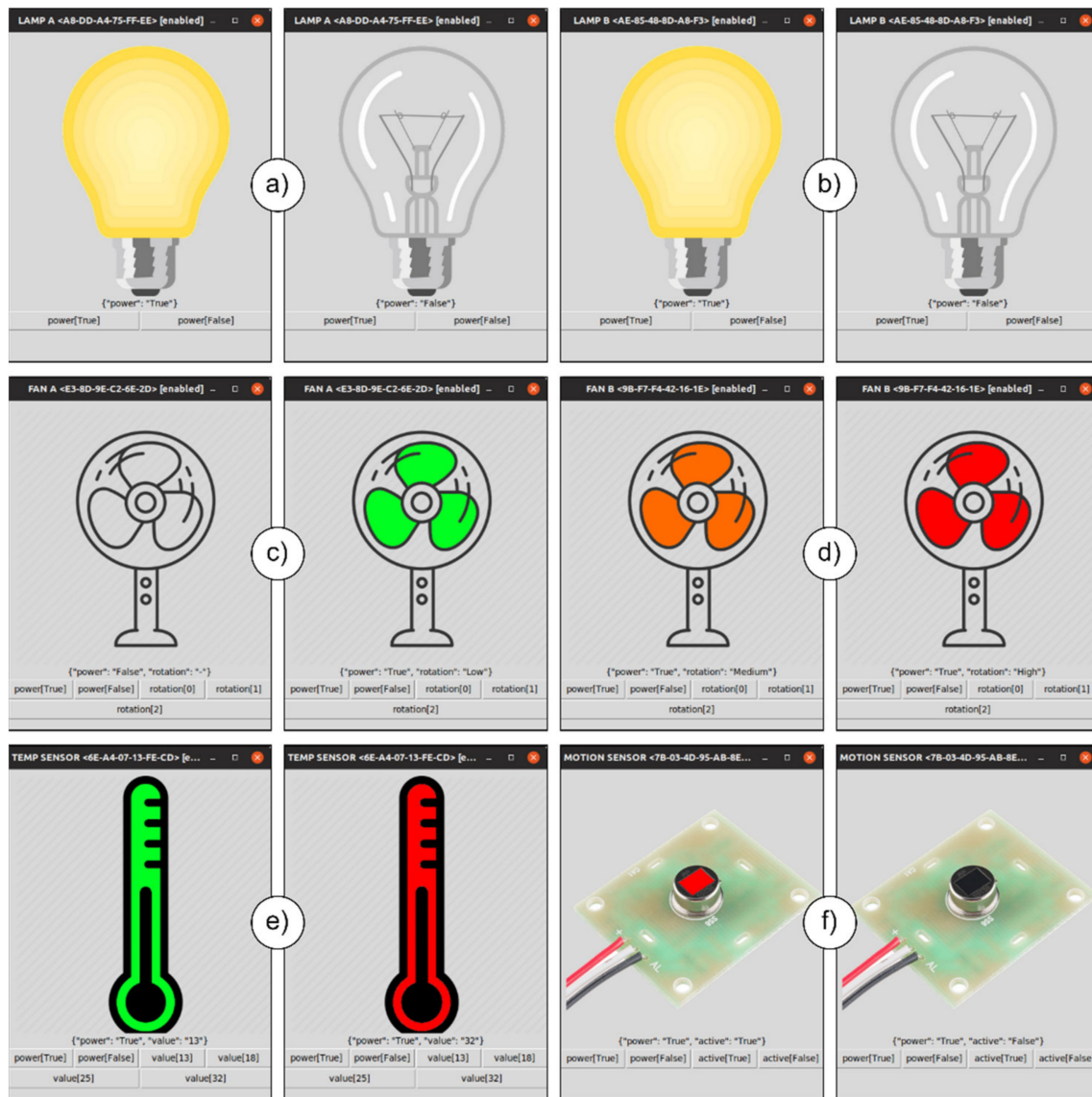






















Figure 9. Two-dimensional actuators and sensors in action: (a,b) with regard to Lamp A and B, respectively, turned on and turned off; (c,d) represent the pair of fans (Fan A and Fan B) simulating different rotation intensities, from stopped to highest level; (e) depicts the temperature sensor with the lower and the higher reading; and, finally, (f) shows the motion sensor, performing in idle and active state.

Table 7. Three-dimensional environment virtual devices, composed of two lamps, a fan, and a display.

Representation	Name	Identifier
	Test Room 3D	6C-D5-1C-76-E3-85
 A	Lamp A	D6-BB-99-92-99-3A
 B	Lamp B	E1-8C-83-02-28-DD
	Fan	1C-B2-89-0F-20-01
	Display	A9-94-43-C9-EB-8C

The schedule-driven scenario entailed (manual) synchronization of video content with both the virtual fan and the lamp, aiming the simulation of air-flow increase in (observationally perceived) windy scenes, as well as to impact visual perception with induced illumination wobbling. As shown in Table 8, this scenario lasted for 60 s and starts by turning off the fan, display and both lamps. After displaying a DC Comics video, some effects with the *Lamp A* and virtual fan were coherently set up to match the scenes characterized by an apparently reduced air-flow, as well as to cope with higher or poorer illumination frame sets. After swapping the display content to a Marvel video, the fan was set up for maximum rotation and *Lamp A* switched state with *Lamp B*. In the end (second 60), all the virtual devices were shutdown.

Table 8. Three-dimensional environment virtual devices: schedule-oriented control testing scenario.

Action	Time (s)	Request	Output
Initialization	0	/out/D6-BB-99-92-99-3A/1/1 Off	 A
Initialization	0	/out/E1-8C-83-02-28-DD/1/1 Off	 B
Initialization	0	/out/1C-B2-89-0F-20-01/1/1 Off	
Initialization	0	/out/A9-94-43-C9-EB-8C/1/1 Off	
Turn on display	1	/out/A9-94-43-C9-EB-8C/1/1 On	
Play DC video	1	/out/A9-94-43-C9-EB-8C/2/1 DC	
Turn on lamp A	1	/out/D6-BB-99-92-99-3A/1/1 On	 A
Set fan rotation to 1	1	/out/1C-B2-89-0F-20-01/1/1 Low	
Play Marvel Video	30	/out/A9-94-43-C9-EB-8C/2/1 Marvel	
Set fan rotation to 3	38	/out/1C-B2-89-0F-20-01/1/1 Medium	
Turn off lamp A	45	/out/D6-BB-99-92-99-3A/1/1 Off	 A
Turn on lamp B	45	/out/E1-8C-83-02-28-DD/1/1 On	 B
Turn off lamp B	60	/out/E1-8C-83-02-28-DD/1/1 Off	 B
Turn off fan	60	/out/1C-B2-89-0F-20-01/1/1 Off	
Turn off display	60	/out/A9-94-43-C9-EB-8C/1/1 Off	

The different devices and respective variation states used in the 3D environment testing scenario are depicted in Figure 10.

In both 2D and 3D approaches, important real-time visual insights were provided after each event-based or schedule-driven state change associated to a given device, which allowed us to confirm the proper functioning and usefulness of the proposed prototyping software.

Additional tests were made to assess the system's performance, in a computer characterized by the following relevant specifications:

- Processor: 11th Gen Intel(R) Core (TM) i7-11800H @ 2.30 GHz 2.30 GHz;
- Random Access Memory (RAM): 32 GB @ 2933 MHz SODIMM;
- Graphic Card: NVIDIA GeForce RTX 3080, 16.0 GB GDDR6 RAM (Laptop edition);
- Storage: 1 TB, 3500 MB/R, 3300 MB/W;
- Operative System: Windows 10 Home 64 Bit.

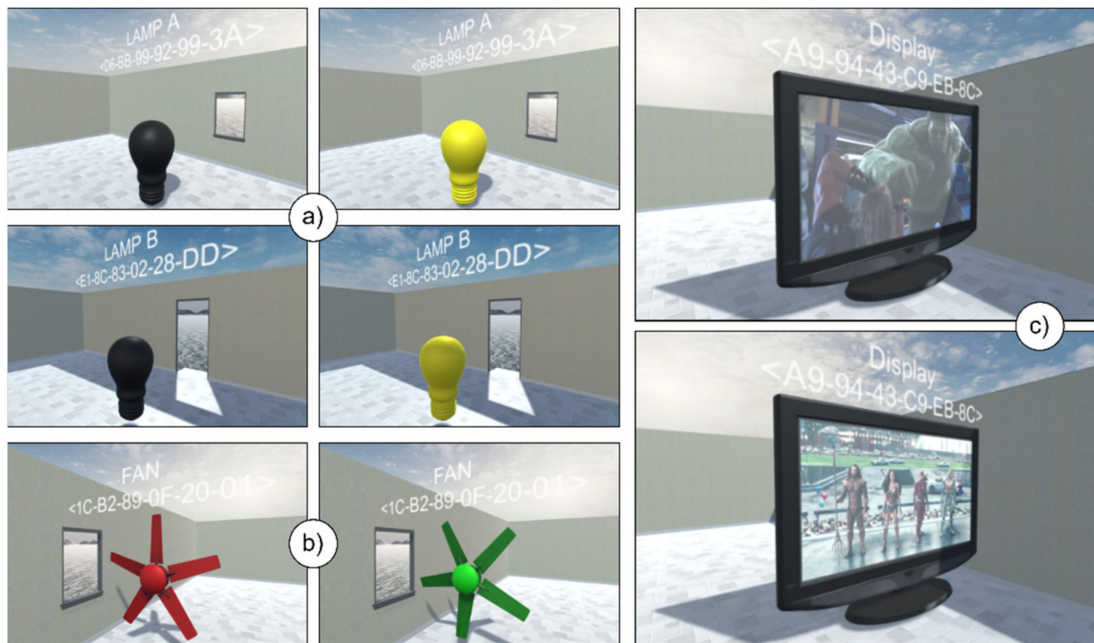


Figure 10. Three-dimensional actuators and sensors in action: the pair of lamps varying from turned off to on are represented by (a,b) regards to the fan, in idle state and during rotation; finally, in (c), lies the display reproducing different content.

Regarding delays impacting in communications and considering that simulations run within a local host environment, MQTT publishing to subscribed topics is instantaneous (equal to or less than 1 ms). Moreover, substantial information on decentralized IoT-related protocols performance, several other studies—such as that found in [35]—are available. Therefore, performance tests focusing both the system’s database and graphical component were carried out. Firstly, timestamp-based monitorization for registrations and updates of virtual devices states in a local Mongo database allowed to measure delays between 17–35 ms. Afterwards, graphical component tests were conducted, starting with the 2D environment, in which a set of timers were implemented to measure delays in between virtual devices state transitions. Figure 11 depicts the results, from which can be denoted a correlation between the number of states that each device may assume, and the average time required by it to consolidate a state switching.

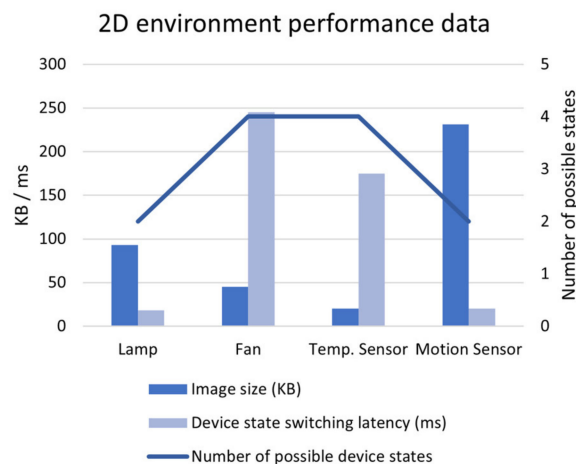


Figure 11. Two-dimensional environment performance results for each implemented virtual endpoint (lamp, fan, as well as temperature and motion sensors). Bars represent both image sizes (KB) and state switching latency (ms) per device, and the implicated axis is the left one. The line regards to the number of possible states that each device may assume, with scale in the right-hand axis.

The 3D environment was also assessed in terms of performance, as shown by Figure 12, considering a couple of perspectives: (1) latency of a device while changing its state; and (2) impact of diverse displays playing content at the same time, in terms of frames per second (FPS). The left plot of Figure 12 shows the time required by the 3D endpoints to switch their state, more specifically, for lamp and fan turning representations according to their condition (power and rotation level) and for displays loading video content. Table 9 data seem to point out that video transmission features influence the computational burden during content loading, i.e., the higher the transmission rates are, the more time is needed to load a video. At the right plot of Figure 12, which intends to show 1 vs. 10 displays' performance impact, a slight performance increase can be noticed when the computational burden is intensified (with the displays number extension). Such an observation may be corroborated by the exceptional specifications of the graphic card in use that was clearly above the demands, leaving space for a frame rate step up.

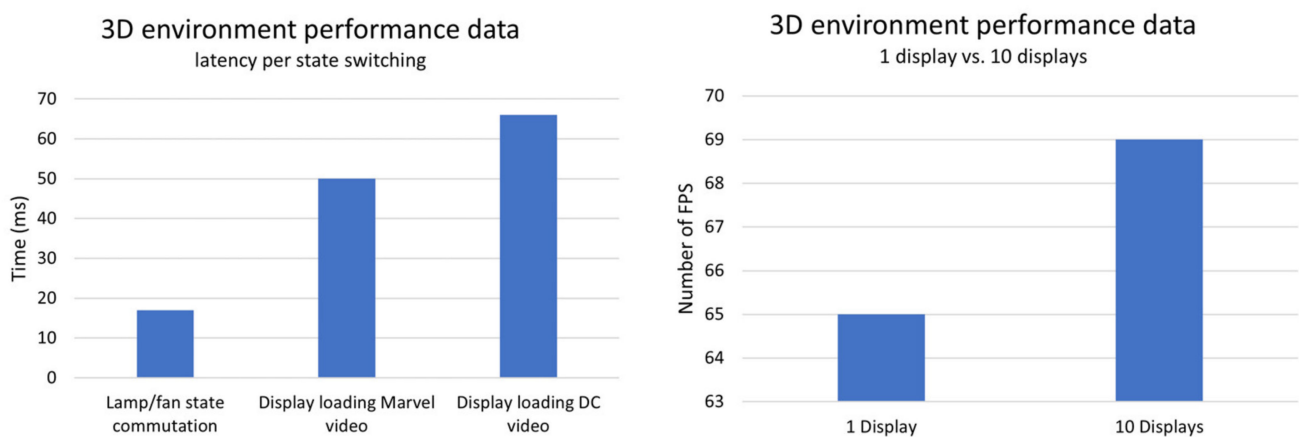


Figure 12. Three-dimensional environment performance assessment: the left plot shows the time required for 3D endpoints (lamp, fan and display) state switching, while in the one placed at the right, the performance impact (in FPS) of having a single display running a video vs. 10 more of them in the same conditions is presented.

Table 9. Specifications of the videos used in the proposed IoT simulation tool.

Video	Size (px)	Data Transmission Rate	Total Transmission Speed (bits)	FPS
Marvel.mp4	1280 × 720	1208 kbps	1334 kbps	30
DC.mp4	1280 × 720	1942 kbps	2069 kbps	24

Overall, content switching seems to be faster in the 3D environment, in spite of the higher complexity associated with this context. Such an observation can be explained by the performances that characterize each adopted development technology. On the one hand, as an interpreted programming language, Python (2D environment)—along with Tkinter graphical user interface library—is a widely used cross-platform tool that, however, suffers from known performance issues. On the other hand, Unity (3D environment) works over .NET platform, which is natively interfaced by Windows (10) operative system, and deploys target-oriented compiled solutions with, usually, better execution times than similar applications resulting from interpreted programming languages.

The next section draws the main conclusions and makes a few recommendations for continuing this work in the near future.

5. Conclusions and Future Work

Multimedia setups can be considered as a unique combination of hardware, software and contents made available in given geographic and social contexts with the purpose to

deliver sensory-aware experiences. Each can also be regarded as a work in progress as they should evolve and interact differently through time, to retain visitors' engagement.

However, to design, maintain and modify mulsemmedia setups requires a team effort and is usually done by specialized companies. Often, scheduled and emergency maintenance operations must be made in loco. These costs can be significant and represent a deterrent for mulsemmedia setups owners to keep their experiences in perfect working order, but mainly to change them and update contents.

This paper proposes a prototyping software to enable IoT mulsemmedia environments simulation, based on virtual environments that enable the design, configuration, maintenance and update of experiences regardless of software and hardware used. Two virtual environments were created and tested, demonstrating success. The prototyping software is currently being used by a multimedia company that manages several mulsemmedia setups both nationally and internationally. It is noteworthy that rather than competing with existing solutions developed with similar purposes (e.g., [28,32]), the proposed system intends to provide an alternative agnostic approach to simulate IoT environments before their deployment, privileging quick planning strategies for arranging mulsemmedia setups, being more flexible to programmers, and capable of being extended with digital twin features for the management of actual infrastructures, supported by more intuitive interfaces.

Future work will, thus, address the extension of this solution to real environments, exploring concepts of digital twinning for remote control and supervision of IoT setups with enhanced levels of interaction engagement, as well as cost-effective maintenance, pursuing a multi-context platform for a wide variety of applications, ranging from business-to-business (to-customer) commercial transaction models to Industry 5.0-related challenges.

Author Contributions: Conceptualization, T.A. and E.P.; data curation, T.A. and T.P.; formal analysis, L.G.M., E.P. and J.J.S.; funding acquisition, E.P.; investigation, T.A., T.P. and L.P.; methodology, T.A., T.P. and E.P.; project administration, E.P.; resources, T.A. and E.P.; software, T.A. and T.P.; supervision, E.P. and L.G.M.; validation, T.P., E.P. and L.G.M.; visualization, T.A., T.P. and L.P.; writing—original draft, T.A., T.P. and E.P.; writing—review and editing, E.P., L.G.M., L.P. and J.J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was financed by project “CHIC—Cooperative Holistic View on Internet and Content” (N° 24498), financed the European Regional Development Fund (ERDF) through COMPETE2020—the Operational Programme for Competitiveness and Internationalisation (OPCI).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

References

1. Coelho, P. *Internet Das Coisas-Introdução Prática*, 1st ed.; FCA-Editora de Informática: Lisboa, Portugal, 2017; ISBN 978-972-722-849-2.
2. Morais, R.; Silva, N.; Mendes, J.; Adão, T.; Pádua, L.; López-Riquelme, J.A.; Pavón-Pulido, N.; Sousa, J.J.; Peres, E. mySense: A comprehensive data management environment to improve precision agriculture practices. *Comput. Electron. Agric.* **2019**, *162*, 882–894. [[CrossRef](#)]
3. Muhammad, K.; Hamza, R.; Ahmad, J.; Lloret, J.; Wang, H.; Baik, S.W. Secure Surveillance Framework for IoT Systems Using Probabilistic Image Encryption. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3679–3689. [[CrossRef](#)]
4. 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. [[CrossRef](#)]
5. Nauman, A.; Qadri, Y.A.; Amjad, M.; Bin Zikria, Y.; Afzal, M.K.; Kim, S.W. Multimedia Internet of Things: A Comprehensive Survey. *IEEE Access* **2020**, *8*, 8202–8250. [[CrossRef](#)]
6. Jalal, L.; Popescu, V.; Murrioni, M. IoT Architecture for Multisensorial Media. In Proceedings of the 2017 IEEE URUCON, Montevideo, Uruguay, 23–25 October 2017; pp. 1–4.
7. Almajali, S.; Abou-Tair, D.E.D.I.; Salameh, H.B.; Ayyash, M.; Elgala, H. A distributed multi-layer MEC-cloud architecture for processing large scale IoT-based multimedia applications. *Multimed. Tools Appl.* **2019**, *78*, 24617–24638. [[CrossRef](#)]
8. Salameh, H.A.B.; Almajali, S.; Ayyash, M.; Elgala, H. Spectrum Assignment in Cognitive Radio Networks for Internet-of-Things Delay-Sensitive Applications Under Jamming Attacks. *IEEE Internet Things J.* **2018**, *5*, 1904–1913. [[CrossRef](#)]
9. Nordrum, A. The internet of fewer things [News]. *IEEE Spectr.* **2016**, *53*, 12–13. [[CrossRef](#)]

10. Alvi, S.A.; Afzal, B.; Shah, G.A.; Atzori, L.; Mahmood, W. Internet of multimedia things: Vision and challenges. *Ad Hoc Netw.* **2015**, *33*, 87–111. [[CrossRef](#)]
11. Kaeri, Y.; Moulin, C.; Sugawara, K.; Manabe, Y. Agent-Based System Architecture Supporting Remote Collaboration via an Internet of Multimedia Things Approach. *IEEE Access* **2018**, *6*, 17067–17079. [[CrossRef](#)]
12. Rego, A.; Canovas, A.; Jimenez, J.M.; Lloret, J. An Intelligent System for Video Surveillance in IoT Environments. *IEEE Access* **2018**, *6*, 31580–31598. [[CrossRef](#)]
13. Rahman, M.A.; Hossain, M.S.; Hassanain, E.; Muhammad, G. Semantic Multimedia Fog Computing and IoT Environment: Sustainability Perspective. *IEEE Commun. Mag.* **2018**, *56*, 80–87. [[CrossRef](#)]
14. Seng, K.P.; Ang, L. A Big Data Layered Architecture and Functional Units for the Multimedia Internet of Things. *IEEE Trans. Multi-Scale Comput. Syst.* **2018**, *4*, 500–512. [[CrossRef](#)]
15. Zhou, L.; Chao, H. Multimedia traffic security architecture for the internet of things. *IEEE Netw.* **2011**, *25*, 35–40. [[CrossRef](#)]
16. Goel, D.; Pahal, N.; Jain, P.; Chaudhury, S. An Ontology-Driven Context Aware Framework for Smart Traffic Monitoring. In Proceedings of the 2017 IEEE Region 10 Symposium (TENSYMP), Cochin, India, 14–16 July 2017; pp. 1–5.
17. Parizi, R.M.; Dehghantanha, A.; Choo, K.R. Towards Better Ocular Recognition for Secure Real-World Applications. In Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 277–282.
18. Sornalatha, K.; Kavitha, V.R. IoT Based Smart Museum Using Bluetooth Low Energy. In Proceedings of the 2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), Chennai, India, 27–28 February 2017; pp. 520–523.
19. De La Borbolla, I.R.; Chicoskie, M.; Tinnell, T. Applying the Internet of Things (IoT) to biomedical development for surgical research and healthcare professional training. In Proceedings of the 2017 IEEE Technology & Engineering Management Conference (TEMSCON), Santa Clara, CA, USA, 8–10 June 2017; pp. 335–341.
20. Poluan, S.E.R.; Chen, Y.-A. A Framework for a Multisensory IoT System Based on Service-Oriented Architecture. In Proceedings of the 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), Daegu, Korea, 22–25 September 2020; pp. 369–372.
21. Rahman, M.A.; Rashid, M.; Barnes, S.; Hossain, M.S.; Hassanain, E.; Guizani, M. An IoT and Blockchain-Based Multi-Sensory In-Home Quality of Life Framework for Cancer Patients. In Proceedings of the 2019 15th International Wireless Communications Mobile Computing Conference (IWCMC), Tangier, Morocco, 24–28 June 2019; pp. 2116–2121.
22. Garzotto, F.; Gelsomini, M.; Gianotti, M.; Riccardi, F. Engaging Children with Neurodevelopmental Disorder through Multisensory Interactive Experiences in a Smart Space. In *Social Internet of Things*; Soro, A., Brereton, M., Roe, P., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 167–184. ISBN 978-3-319-94659-7.
23. Gelsomini, M.; Cosentino, G.; Spitale, M.; Gianotti, M.; Fiscaro, D.; Leonardi, G.; Riccardi, F.; Piselli, A.; Beccaluva, E.; Bonadies, B.; et al. Magika, a Multisensory Environment for Play, Education and Inclusion. In Proceedings of the Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems, Glasgow, Scotland, UK, 4–9 May 2019; ACM: New York, NY, USA; pp. 1–6.
24. Striner, A. Can Multisensory Cues in VR Help Train Pattern Recognition to Citizen Scientists? *arXiv* **2018**, arXiv:1804.00229.
25. Pelet, J.-E.; Lick, E.; Taieb, B. Internet of Things and Artificial Intelligence in the Hotel Industry: Which Opportunities and Threats for Sensory Marketing? In Proceedings of the International Conference on Research on National Brand and Private Label Marketing, Granada, Spain, 1 May 2019.
26. Covaci, A.; Zou, L.; Tal, I.; Muntean, G.-M.; Ghinea, G. Is Multimedia Multisensorial? A Review of Mulsemmedia Systems. *ACM Comput. Surv.* **2019**, *51*, 1–35. [[CrossRef](#)]
27. D’Angelo, G.; Ferretti, S.; Ghini, V. Simulation of the Internet of Things. In Proceedings of the 2016 International Conference on High Performance Computing Simulation (HPCS), Innsbruck, Austria, 18–22 July 2016; pp. 1–8.
28. Haris, I.; Bisanovic, V.; Wally, B.; Rausch, T.; Ratasich, D.; Dustdar, S.; Kappel, G.; Grosu, R. Sensym: Simulation Environment for Large-Scale IoT Applications. In Proceedings of the IECON 2019–45th Annual Conference of the IEEE Industrial Electronics Society, Lisbon, Portugal, 14–17 October 2019; Volume 1, pp. 3024–3030.
29. Zeng, X.; Garg, S.K.; Strazdins, P.; Jayaraman, P.P.; Georgakopoulos, D.; Ranjan, R. IOTSim: A simulator for analysing IoT applications. *J. Syst. Arch.* **2017**, *72*, 93–107. [[CrossRef](#)]
30. Calheiros, R.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.F.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **2011**, *41*, 23–50. [[CrossRef](#)]
31. Ahmad, S.; Mehmood, F.; Mehmood, A.; Kim, D. Design and Implementation of Decoupled IoT Application Store: A Novel Prototype for Virtual Objects Sharing and Discovery. *Electronics* **2019**, *8*, 285. [[CrossRef](#)]
32. Han, S.N.; Lee, G.M.; Crespi, N.; Van Luong, N.; Heo, K.; Brut, M.; Gatellier, P. DPWSim: A Devices Profile for Web Services (DPWS) Simulator. *IEEE Internet Things J.* **2015**, *2*, 221–229. [[CrossRef](#)]
33. Alce, G.; Ternblad, E.-M.; Wallergård, M. Design and Evaluation of Three Interaction Models for Manipulating Internet of Things (IoT) Devices in Virtual Reality. In *Human-Computer Interaction—INTERACT 2019, Proceedings of the Human-Computer Interaction—INTERACT 2019, Paphos, Cyprus, 2–6 September 2019*; Lamas, D., Loizides, F., Nacke, L., Petrie, H., Winckler, M., Zaphiris, P., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 267–286.

-
34. Hassija, V.; Chamola, V.; Saxena, V.; Jain, D.; Goyal, P.; Sikdar, B. A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures. *IEEE Access* **2019**, *7*, 82721–82743. [[CrossRef](#)]
 35. Mishra, B. Performance Evaluation of MQTT Broker Servers. In *Computational Science and Its Applications—ICCSA 2018, Proceedings of the Computational Science and Its Applications—ICCSA 2018, Melbourne, VIC, Australia, 2–5 July 2018*; Gervasi, O., Murgante, B., Misra, S., Stankova, E., Torre, C.M., Rocha, A.M.A.C., Taniar, D., Apduhan, B.O., Tarantino, E., Ryu, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 599–609.