AVIoT: Web-Based Interactive Authoring and Visualization of Indoor Internet of Things

Yuna Jeong, Hyuntae Joo, Gyeonghwan Hong, Dongkun Shin, *Member*, IEEE, and Sungkil Lee

Abstract — *Internet of things recently emerges as a common platform and service for consumer electronics. This paper presents an interactive framework of visualizing and authoring IoT in indoor environments such as home or small office. Building blocks of the framework are virtual sensors and actuators that abstract physical things and their virtual behaviors on top of their physical networks. Their behaviors are abstracted and programmed through visual authoring tools on the web, which allows a casual consumer to easily monitor and define their behaviors even without knowing the underlying physical connections. The user study performed to assess the usability of the visual authoring showed that the visual authoring is easy to use, understandable, and also preferred to typical text-based script programming*[1].

Index Terms — Internet of things, visualization, authoring, sensor, actuator

I. Introduction

As modern devices and sensors continue to grow in power and functionality and to reduce in their cost, internet of things (IoT) emerges as a common platform and service for consumer electronics [1]. IoT enables to be connected to virtually unlimited devices over the internet. It thus has a great potential of communicating and interacting with them in synergistic and creative ways beyond the traditional services of isolated consumer electronics platforms.

Recent advents made in IoT suggest creative ways to control and interact with things (sensors and actuators), which facilitates automated interactions in IoT environments [2], [3]. Sensors perform distributed sensing and are fused together to provide higher-level information from the raw data gathered from the physical world [4]. A user makes a decision on required actions, based on the salient significant information. The decision can then trigger appropriate actions on physical actuators or their abstractions.

Functional mapping predefined between the things is crucial for effective interaction in smart IoT environments [5], [6]. Timely interactive control is often difficult even for the indoor environments where there are dozens of consumer-level sensors and actuators. A key aspect is how to collect effective information from physical sensors and associate it with physical actuators to fulfill desired high-level functions. In general, controlling individual things is not preferred. Intuitive actuation based on the salient information is preferred. For instance, "A human comes in, so turn on the lights in the living room." is preferred to "When motion sensors whose IDs are 3 and 4 detect a velocity of > 10 cm/s in the room, turn the lights whose IDs are 1 and 2." Hierarchical functional grouping is further helpful in creating heterogeneous complex behaviors across sensors and actuators.

Visual authoring particularly well aids the functional mapping such that even casual consumers can easily create and monitor abstractions of their own devices or subsystems, and can appropriately respond significant events [7]. A programming like scripting is required for developers and manufacturers, but is often infeasible for the majority of IoT consumers. Real-time visual feedback with intuitive definition is likely to be better, which actually leverages the definition of mapping between things as the scripting does. This observation inspires exploration of visual authoring for IoT environments, which is not extensively studied in the past.

This paper presents a web-based interactive framework that helps visualization and authoring of indoor IoT environments such as electronic sensors/devices and their systems in homes or small offices. Given a user's hints on the indoor architecture, the framework loads one of the best matches of 3D model templates. The indoor 3D scene is configured to selectively visualize attributes of available physical sensors and actuators. The system acts as a WYSIWYG (What you see is what you get) IoT editor, which allows a user to relate sensors and actuators interactively. The system further supports visual programming of virtual composite sensor/actuators, which defines their complex heterogeneous behaviors. The system was implemented on the WebGL (OpenGL ES on the web) for maximum portability and instant readiness across many different consumer electronics platforms. Finally, a user study was carried out to compare the task performance and usability of creating IoT definitions with and without visual feedback.

II. Related Work

This section reviews the previous work on IoT frameworks, abstraction/virtualization of things, and their visualization and authoring.

A. IoT Frameworks

IoT devices are inherently heterogeneous and non-standardized in their forms and interfaces, and thereby, recent attempts are focusing mostly on frameworks to bring about

[1] This work was supported by Software R&D Center, Samsung Electronics and the National Research Foundation of Korea under Grant Nos. 2012R1A2A2A01045719 and 2015R1A2A2A01003783.

Yuna Jeong, Hyuntae Joo, Gyeonghwan Hong, Dongkun Shin, and Sungkil Lee are with College of Information and Communication Engineering, Sungkyunkwan University, Suwon, Korea (e-mail: jeongyuna, htjoo, redc7328, dongkun, sungkil@skku.edu).

Contributed Paper
Manuscript received 07/01/15
Current version published 09/29/15
Electronic version published 09/29/15.

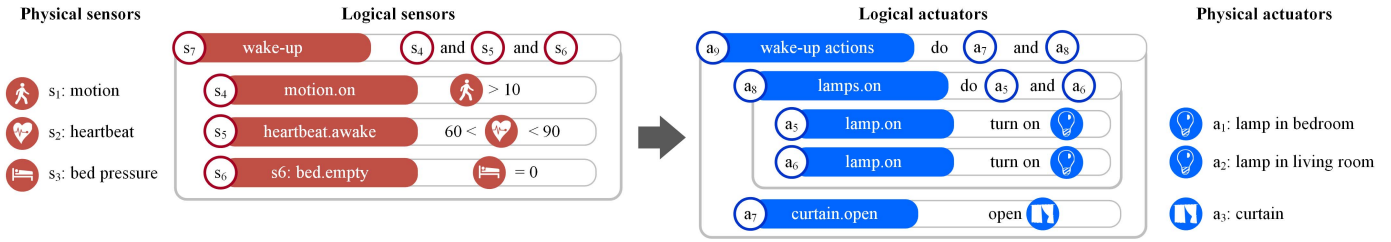


Fig. 1. An example overview of hierarchical abstraction of virtual sensors and actuators. Given the physical sensors and actuators, their hierarchical groupings make logical sensors/actuators. As a result, when the top-level virtual sensor (s_7) detects a user's wake-up event, the top-level virtual actuator (a_9) triggers the actuations of the physical devices.

standardization of IoT. Early sensor frameworks focused mainly on the connectivity protocols [8], sensor data interfaces [9], and remote sensing capability [10]. Recent open-source middle-wares commonly involve service and core (or base) layers, relying on web-based interfaces. Whereas the service layer regards high-level services such as data streaming and control, the core/base layer manages fundamental communication of IoT devices through high-level extensible abstraction of sensors/actuators and their cost-effective fusion [10] are of great interests, which still requires further efforts for the convergence.

B. Data Fusion for Sensors and Actuators

Physical IoT devices and their data, distributed in IoT environments, are usually abstracted or virtualized with data processing and filtering. The sensor data are transformed, processed for precise measurement [4], and fused to form soft sensor data. Likewise, actions of actuator devices are virtualized as an action group. Such abstraction, often dubbed as context-aware smart objects [11], not only provides better connectivity programming via middleware [10], [12], but also is used for deploying synergistic high-level services such as web or cloud server platforms [13], [14]. However, the majority of such approaches are focused for service designers and programmers rather than the consumer-level users. In contrast, the proposed framework builds on top of the middleware for the consumer electronics abstraction through more intuitive visual programming.

C. Sensor Visualization and IoT Authoring

There should be difficulties in visualizing IoT connected to plenty of devices, which spawns effective visualization [15]. The majority of previous work has focused solely on the visualization of spatial configuration and topology of sensors [16]. Early studies visualized sensor locations by overlaying them on the outdoor geographic or satellite maps [17], [18] or indoor 2D plans [16]. Their improvements have been made based on the type visualization [19] and level of details for distraction-free visualization [20].

IoT better manifests itself when it deals with context-driven events [5] and handles them with appropriate actions. Pre-defined 2D graphical user interfaces (GUIs) [21] facilitated the authoring of a user who does not have knowledge of computer programming, but have been limited mostly in static scenarios. Dynamic behaviors were supported by scripting languages (e.g. Python) [22] or modular assembly [23]. They gained flexibility, but lost ease of programming to some extent. A high-level visual metaphor such as block puzzle [24] is better suited for consumer-level users, but limited only in small-scale environments.

The previous studies were generally helpful in understanding the structural organization of sensor network, but have been limited in planar placements and authoring of static simple configurations. On the other hand, the proposed framework supports real-time dynamic 3D visualization with intuitive user interfaces within similar spatial frame of references, which enables to intuitively navigate, examine, and author things of interest. Further, the proposed approach is designed for flexible yet easy-to-make authoring of IoT environments, which bridges underlying scripting models for the IoT middleware and their subsystem owned by consumers through the visual programming.

III. Abstraction of Sensors and Actuators

This section describes the design and definition of virtual things (including sensors and actuators) that can be seamlessly linked together for diverse functions in IoT environments. Fig. 1 illustrates an overview of the design of abstracted things for an example of defining a wake-up event and its actuation.

Conceptually, everything can be connected or grouped with other things. Physical things are first virtualized to interact with other virtual/logical things ($\{s_1, s_2, s_3\}$ and $\{a_1, a_2, a_3\}$ in the figure). Whereas the physical virtual sensors output numeric measurements, physical virtual actuators can trigger actions of corresponding physical devices. As for the virtual sensor, it is necessary to define a boolean outcome that indicates the occurrence of a particular event; for example, motion > 10 cm/s detects a moving human. Further hierarchical grouping enables to define complex events and their handlers. Eventually, pairing across a top-level sensor and actuator (s_7 and a_9) defines an event handler; here, the top-level sensor needs to produce a boolean outcome.

```
struct node             // virtual thing: sensor or actuator
{
    int     ID;         // numeric ID
    string  name;       // literal ID
    int     type;       // type of thing for sensor/actuator
    int     vis_type;   // e.g., bar, donut, pie, line, stack
    vec4    position;   // in either of object or screen space
    node    children[]; // empty for physical things
    string  behavior;   // functional string to evaluate
};
```

Fig. 2. Definition of virtual things.

Every *thing* is defined as a node whose pseudocode-like definition is shown in Fig. 2. The definition includes common attributes of identity, name, type of the thing, type of visualization, position, children, and functional string of behavior. The type of thing distinguishes different types of sensors and actuators as well; its visual icon is defined by the type. The functional behavior of things is defined as a string that actually encodes script codes and evaluated at run time.

Basic logical operations can be visually authored, and complex functions can be also script-programmed. In particular for physical devices, their communications need to be coded here through native programming or scripts. What follows presents more details on virtual sensors and actuators.

A. Virtual Sensors

The primary role of virtual sensors is to detect non-trivial events in which a user might be interested. The majority of physical sensors expose only numeric values, which needs a user or system to interpret. The user interprets the implications of numeric values and write their outcome as a boolean value that indicates the occurrence of an event. Such interpretation is encoded in a Javascript object notation (JSON) format and evaluated on demand with 'eval()' in Javascript.

In many cases, a high-level event usable for practical scenarios needs to aggregate outcomes of multiple sensors. It is possible to program complex behaviors in a single virtual sensor, but an easier way is to simply pick up pre-defined sensors and to hierarchically group them. Then, the boolean outcomes of the corresponding sensors are combined with logical operators to produce a single boolean outcome.

B. Virtual Actuators

Virtual actuators abstract the user-defined behaviors of physical actuators. They can be similarly abstracted as for virtual sensors so that they can trigger complex actions with occurrence of a single event. The hierarchical grouping can be based on a spatial or semantic proximity of actuators; e.g. grouping of all the light bulbs in the living room. The behavior can be also defined in a higher level; e.g. an "electricity saver" triggers to turn off all the redundant light bulbs in the living room and electronic devices and also levels up the temperatures of the refrigerator a bit higher.

Unlike the sensors, the virtual actuators do not necessarily represent their values, since they are usually responsible only for triggering actions in physical devices. Nonetheless, an information visualizer can be also defined to support their monitoring; actually, this is another types of virtual sensor. For example, when the electricity is consumed more than normal amounts, electricity alert sensor can be displayed on the screen or broadcast messages.

IV. Interactive IoT Visualization

This section presents how virtual things are visualized in the indoor IoT environments to facilitate the interactive visual authoring detailed in Section V.

A. Server and Clients for IoT Visualization

The visualization and authoring tools are running on web-enabled client applications, including web browsers or mobile web applications on consumer electronics. In order for the clients to query pre-defined or real-time measurements, a server, often called the IoT gateway [25], is required. Since resource-limited platforms are often provided, a light-weight server is preferred. Whereas direct communications between devices are often possible, the current framework here rely solely on the server to store additional information for visualization and authoring and to communicate with a representational state transfer (REST) interface. It is assumed that the IoT gateway already discovered the devices and holds their connectivity information. In addition, the server holds templates of indoor plans or their 3D models, the specification of physical things, and visual icons/representations. Such static information is queried in advance of running the client application, and measurements are queried asynchronously in run time.

B. Layout and Spatial Configuration

A 3D indoor environment is initialized with the given 3D models fetched from the server. When the very models of the user's real environment are available, the environments and devices can be precisely designed. However, this is not often the case, because casual consumers should be able to easily 3D-capture their housing (still challenging even with modern techniques). Hence, the proposed approach is to let a user select the best match among the available models from the server, based on the layout and size of the housing. In many cases, the layout of modern houses are fairly similar, making it a viable approach.

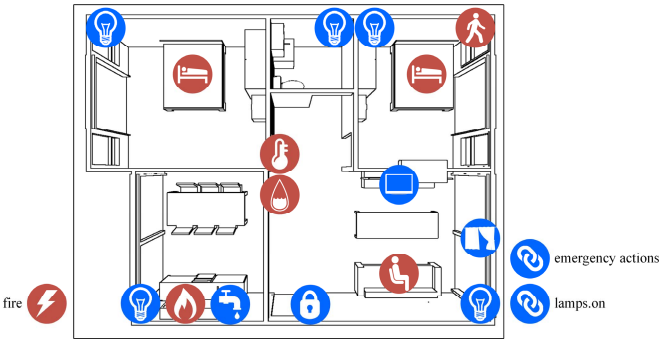


Fig. 3. Top-view 3D visualization of an initial state of an IoT environment. Sensors and actuators are placed on top of the user-chosen 3D model.

Unlike the housing itself, many of physical devices share the same form factors, and thus, pre-built models (can be provided by the device vendors) are enough for the current purpose. Given the static specification of physical IoT devices, they are first located arbitrarily in the scene. Since the models do not present actual location precisely, a user need to relocate them to fit with the real positions (see Fig. 3 for its top-view visualization); note that the accuracy of placements, nonetheless, does not affect those of high-level interaction of things. Then, the user's placements are stored in the server so

that they can be restored later. Unlike the previous 2D layout-based approaches [21], the proposed framework supports interactive 3D navigation, enabling relatively easy placement of the models of consumer devices.

Pure virtual (non-physical) sensors/actuators do not have their natural spatial locations, but locating them aside their child devices can still help a user control them easily. The other way is to locate them completely outside the viewport. If so, only logical things are arranged around the viewport, and the user can easily access the high-level functions. The choice of the pure virtual things depends upon the user's preference.

C. Visualization Scheme

The indoor 3D plan is rendered like wire frames, which visualizes only significant edges in 3D structures. A two-pass rendering is used; geometric attributes of normal vectors and depth values of 3D surfaces are rendered in the first pass, and the second pass determines whether the pixels belong to edges. This is intended to emphasize the device configuration without visual clutter, where the exact color information matching the user's housing is also unavailable.

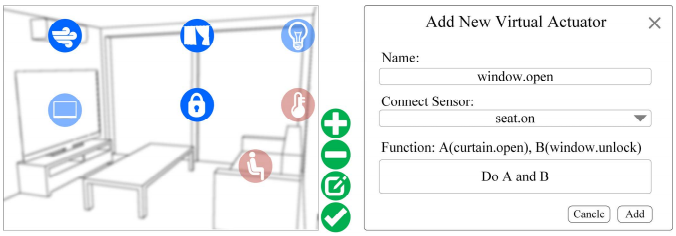
The visual icon of each node is displayed on the screen based on their 3D positions; here, an initial manual placement is necessary. The types of device nodes are distinguished by their colors and icons. The colors identify whether they are sensor or actuators; in the current implementation, red and blue for the sensors and actuators, respectively. The icon images differentiate the specific types of sensors and actuators. Virtual nodes not having their physical positions are placed in the periphery of the viewport. The layout is adaptively organized by the form factor of the display devices (i.e., whether it is displayed on the mobile devices or regular PCs).

When selecting a node (via touch or mouse click), they enter the selection mode. Non-selected nodes are displayed translucently and the background is blurred out for better visibility of the selected node. The current states/values of the selected node are displayed beside the icon. When clicking the state text box, the page is redirected to another page that details the properties such as average values during the fixed period. The details (e.g. bar charts or scatter plots) are selected by the user according to the type of nodes.

V. Interactive IoT Authoring

A. Creating Virtual Nodes

The proposed framework allows a user to create, edit, and delete a virtual node from the given physical nodes (sensors and actuators). The physical nodes are located by the server specification, and user can relocate to the desired positions. A single or multiple nodes can be selected by a long click and subsequent short clicks (see Fig. 4a). The selected sensors activate the buttons for editing, including creation, deletion, behavior editing, and done. By giving the name and type of the node, the user can simply create a virtual composite node (see Fig. 4b).



(a) multiple selection (b) virtual node creation

Fig. 4. Multiple nodes (here, strong-blue actuators) selected by a user (a) can be grouped to form a new virtual node with its behavior definition.

B. Behavior Definition

Whereas the physical sensors output only numeric or boolean values, behaviors of virtual sensors define boolean outcome, which means the occurrence of a particular event. High-level events can be detected by processing multiple sensors; e.g. the fire event can be detected with the combination of a thermometer and smoke sensors. Complex behaviors can be also programmed by simple scripting languages; the proposed framework also supports such scripting. However, only for visual programming, a simple combination of boolean events without explicit scripting is much easier; when there is a user's definition, this simple behavior is overridden. This is similar to modern web search engines that combine all the keywords to find a particular condition. The hierarchical graph for visual debugging helps to this end (see Fig. 5).

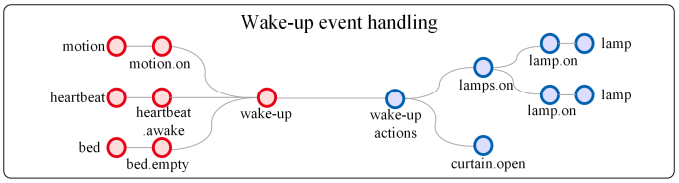


Fig. 5. The hierarchical view visually summarizes sensors, actuators, and their mapping.

Whereas the virtual sensors are focusing on the event detection, the virtual actuators are more related to high-level actions. For instance, when the fire event occurs, the appropriate actions include multiple steps such as calling to the fire station, shutting down the firewall, and sending alarms to the owners and neighbors. Each atomic action is grouped to higher-level actuators, and final hierarchical grouping defines a single high-level action.

C. Event Handling

Since the sensors and actuators are responsible for the event detection and its handling, they need to be linked to work in practice. Each virtual sensor has a link pointer to a particular actuator. Usually, a single top-level sensor is connected to a single top-level actuator to easily connect and edit the event handling. More complex mapping is obviously possible with the scripting, but the casual consumers can do similar mapping only with atomic visual events and handling.

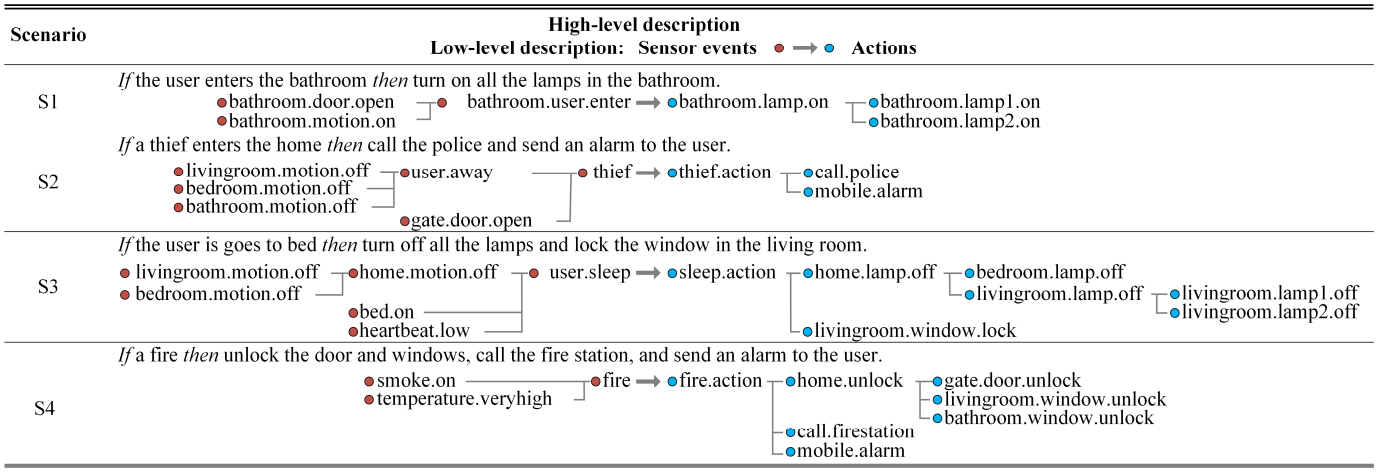


Fig. 6. Four scenarios used in the experiment. S1 is for the training session, and S2, S3, and S4 for the main experiment.

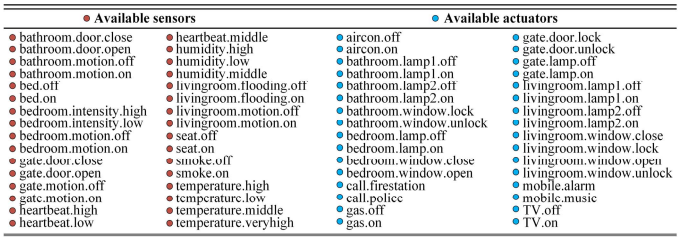


Fig. 7. Available nodes listed in the experiment.

D. Visual Debugging

The authoring results are likely to be complex when handling complex high-level events and actions, and thus, the support for its visual debugging will be highly helpful for the correct authoring. In the proposed framework, a user can obtain a hierarchical view of the whole authoring by clicking a debugging icon located on the top right. The hierarchical view summarizes and presents the authoring, context of nodes, and its mapping using a tree structure. The node is shown with their name, and children of the nodes and their linkage group are connected by lines (see Fig. 5).

VI. Experimental Evaluation

This section reports a user study to experimentally assess the usability of the proposed framework for visual IoT authoring.

A. Implementation Details

The proposed system was implemented on a consumer-level server with 3.4 GHz CPU and 16 GB RAM, which serves all the information of the visualization and authoring including sensor/actuator specification, and definitions of virtual nodes in a database. The authoring and visualization were performed on web browsers. The rendering of the 3D environments used WebGL that is a web-driven abstraction of OpenGL ES. JavaScript was used as an application language, and the user interface used HTML5.

The proposed framework did not install real physical sensors and actuators. The physical sensors were emulated by the server response for the client's data request. The physical actuators were emulated by the logging of the action information on a web-based console.

B. Methods

1) Participants and Apparatus

Twenty four undergraduate and graduate students (23 males and 1 female; 20-27 years old with average 24.2) participated in the experiment. Fourteen participants among them had prior experiences on scripting languages and their programming, and the other had no knowledge of script programming. The participants used an 11-inch consumer-level touch-sensitive tablet (1.7 GHz CPU and 4 GB RAM), a keyboard, and a mouse for authoring.

2) Stimuli

Four example scenarios were used in the experiment (see Fig. 6). The four scenarios (S1-S4), with different indoor plans, evolved along with the complexity in terms of grouping and definitions. S1 among them was used solely for the training session, being excluded from result analysis.

For all the scenarios, the definitions of physical sensors and actuators are given in advance to each session, also with high-level verbal presentation. The participants were instructed to first author sensor definitions and actuator definitions separately, and to eventually define a mapping between the highest levels of sensor and actuator.

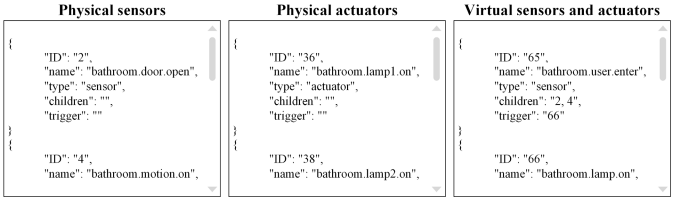


Fig. 8. Examples of JSON definition.

The output of the task used JSON definitions. The set of JSON definitions of all the nodes were loaded in the framework, and tested with the authored configuration. Example excerpts are shown in Fig. 8.

The debugging was supported in two types: the hierarchical graph of sensors and actuators; the console output printed on the screen as a web element. The test button was prepared on the screen and enforced particular events (with the pre-defined fixed behaviors) to confirm that the authored configuration

works correctly. When the authoring is still wrong, the error messages are displayed on the console.

3) Design and Procedure

The experiment used one-factor within-subject design. The factor is the type of authoring, having two levels: text-based scripting and visual authoring with visual feedback of the proposed system. The text-based scripting used a web-based simple text editor, which enables to save and run the script. The output was displayed on the console. On the other hand, the visual authoring used a web browser as a visual editor.

Each participant was first orally instructed with a descriptive document for the experimental procedure. The concepts of the IoT authoring were first explained, followed by distinctions between the text-based scripting and visual authoring. Then, the participant performs the training session on S1 to see how to perform the authoring and debugging. After one-minute break following the training session, the participant went through the six (2 authoring methods × 3 scenarios) successive sessions with one-minute break after each session. The order of sessions was balanced using Latin squares to reduce leaning effects. Each session was continued until the sensors and actuators work correctly as desired, and the timing for completion was measured in seconds. After completing all the sessions, all the participants were asked to fill a subjective questionnaire on learnability, user friendliness, understandability, efficiency, preference, and free-form suggestions (see Table I).

TABLE I
SUBJECTIVE QUESTIONNAIRE RATED IN A 100-POINT SCALE, USED IN THE USER EXPERIMENT.

No.	Questions
Q1	How easily did you understand the method?
Q2	How easily did you perform the authoring task?
Q3	How easily did you understand the given information?
Q4	How efficiently did you perform the task as you want?
Q5	How much did you like the method?

C. Results and Discussion

The average task completion timings measured for each scenario are shown with standard errors in Fig. 9a. For all the scenarios, the visual authoring resulted in much less completion time than the text-based authoring (up to savings of 47 %). Analysis of variance (ANOVA), applied to see the statistical significances of the two authoring methods, also found significant differences in terms of p-values (<0.0001, 0.0128, and <0.0001 for S1, S2, and S3, respectively).

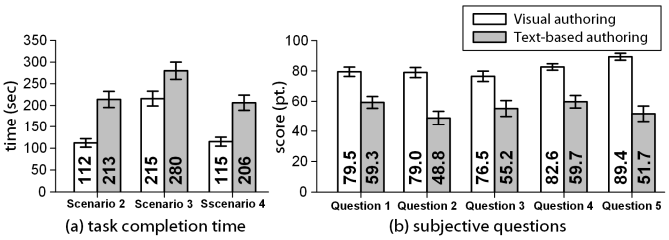


Fig. 9. Average measurements of the user experiment.

The subjective ratings of the participants are plotted with standard errors in Fig. 9b. The proposed visual authoring elicited much positive responses than the text-based authoring in all the questions. This indicates that the visual authoring is apparently easy to learn (Q1), enhances usability in terms of user friendliness (Q2) and efficiency (Q4), the 3D visualization is more intuitive to understand information of IoT nodes (Q3), and the visual authoring is much preferred (Q5). The ANOVA again found the statistical significances of all the items; all the p-values were < 0.0001 except for Q3 (= 0.0013).

The experimental results showed that the visual authoring already has significant benefit over the conventional text-based scripting in many aspects of IoT authoring. The major benefit stems from the easy visual identification of nodes and grouping, whereas the text-based editing shows a clear description but becomes harder with a complex hierarchy. The additional survey for improvements of the system revealed further potentials for better authoring interfaces. The major suggestions include: a better labeling of nodes; emphasis of newly created nodes; a better way to search the nodes; customized icons for each node.

VII. CONCLUSION AND GENERAL DISCUSSIONS

The visual feedback is one of the promising elements in creating and mediating effective user experiences in consumer-level IoT environments. This paper presented the visualization and authoring frameworks based on web implementation, particularly focusing on indoor environments. The intuitive visualization and authoring metaphors allowed consumer users to easily define complex hierarchy of things and their automation. Virtual sensors and actuators served as a basis for constructing high-level IoT environments, decoupling from the low-level sensor/actuator networks.

The user experiment was carried out to assess the usability of the framework. It revealed significant potentials of the visual authoring in many aspects. Whereas the text-based authoring facilitates the precise authoring in relatively simple environments, the visual authoring is better in terms of intuitive identification of complex IoT nodes. This nature led to significantly improved usability in many aspects.

The framework is still a prototype and leaves rooms for further improvements. First, the structures of real environments are not exactly fitted to indoor 3D models that are selected from pre-defined templates. When the framework is combined with accurate modeling of the users' real environments, the usability will be likely to be enhanced greatly. One of feasible future work includes inference of 3D models from 2D indoor plans. Second, specific implementations of the user interface affects the user experiences, and detailed comparison and refinements will be useful. Third, 3D visualization is not likely to be always effective for authoring. The effective combination with 2D visualization will lead to better authoring/visualization. Fourth, the visual icons of IoT nodes can be improved to

reflect the context of the environments. For instance, emergency cases can be emphasized with a salient color of the node different from their surrounding neighbors.

REFERENCES

[1] H. Kopetz,"Internet of things," *Real-time systems*, Springer: USA, 2011, pp. 307-323.

[2] P. Belimpasakis and R. Walsh, "A combined mixed reality and networked home approach to improving user interaction with consumer electronics," *IEEE Trans. Consumer Electron.*, vol. 57, no. 1, pp. 139-144, Feb. 2011.

[3] D. Macagnano, G. Destino, and G. Abreu, "Indoor positioning: A key enabling technology for IoT applications," *in Proc. IEEE World Forum on Internet of Things*, Seoul, Korea pp. 117-118, Mar. 2014.

[4] W. T. Sung and M. H. Tsai, "Data fusion of multi-sensor for IOT precise measurement based on improved PSO algorithms," *Elsevier Computers & Mathematics with Applications*, vol. 64, no. 5, pp. 1450-1461, Sep. 2012.

[5] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 414-454. Feb. 2014.

[6] Q. Wu, G. Ding, Y. Xu, S. Feng, Z. Du, J. Wang, and K. Long, "Cognitive Internet of things: A new paradigm beyond connection," *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 129-143, Apr. 2014.

[7] A. R. Al-Ali, I. A. Zualkernan, A. Chreide, and H. Abu Ouda, "GRPS-Based Distributed Home-Monitoring Using Internet-Based Geographical Information System," *IEEE Trans. Consumer Electron.*, vol. 57, no. 4, pp. 1688-1694, Nov. 2011.

[8] C. Suh and Y. B. Ko, "Design and implementation of intelligent home control systems based on active sensor networks," *IEEE Trans. Consumer Electron.*, vol. 54, no. 3, pp. 1177-1184. Aug. 2008.

[9] W. Brunette, M. Sundt, N. Dell, R. Chaudhri, N. Breit, and G. Borriello, "Open data kit 2.0: expanding and refining information services for developing regions." *in Proc. ACM Workshop on Mobile Computing Systems and Applications*, Georgia, USA, pp. 10, Feb. 2013.

[10] X. Zheng, D. E. Perry, and C. Julien, "BraceForce: a middleware to enable sensing integration in mobile applications for novice programmers," *in Proc. ACM International Conference on Mobile Software Engineering and Systems*, Hyderabad, India, pp. 8-17, Jun. 2014.

[11] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, "Smart objects as building blocks for the internet of things," *IEEE Internet Computing*, vol. 14, no. 1, pp. 44-51, Feb. 2010.

[12] L. Hu, F. Wang, J. Zhou, and K. Zhao, "A Data Processing Middleware Based on SOA for the Internet of Things," *Journal of Sensors*, Article ID 827045, Jan. 2015.

[13] D. Díaz-Sánchez, F. Almenarez, A. Marín, D. Proserpio, and P. A. Cabarcos, "Media cloud: an open cloud computing middleware for content management," *IEEE Trans. Consumer Electron.*, vol. 57, no. 2, pp. 970-978, May. 2011.

[14] J. Yang, H. Park, Y. Kim, and J. K. Choi, "Programmable objectification and Instance Hosting for IoT nodes," *in Proc. IEEE Asia-Pacific Conference on Communications*, Denpasar, Indonesia, pp. 603-608, Aug. 2013.

[15] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Elsevier Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645-1660, Sep. 2013.

[16] M. Simek, L. Mraz, and K. Oguchi, "SensMap: Web framework for complex visualization of indoor & outdoor sensing systems" *in Proc. International Conference on Indoor Positioning and Indoor Navigation*, Montbeliard, France, pp. 1-5, Oct. 2013.

[17] K. A. Delin, "The Sensor Web: A macro-instrument for coordinated sensing," *Sensors, 2nd ed*., MDPI: Switzerland, 2002, pp.270-285.

[18] S. H. Liang, A. Croitoru, and C. V. Tao, "A distributed geospatial infrastructure for Sensor Web," *Elsevier Computers & Geosciences*, Vol. 31, no. 2, pp. 221-231, Mar. 2005.

[19] A. P. Castellani, M. Dissegna, N. Bui, and M. Zorzi, "WebIoT: A web application framework for the internet of things," *in Proc. IEEE Wireless Communications and Networking Conference Workshops*, Paris, France, pp. 202-207, Apr. 2012.

[20] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "Sensorscope: Out-of-the-box environmental monitoring," *in Proc. International Conference on Information Processing in Sensor Networks*, Missouri, USA, pp. 332-343, Apr. 2008.

[21] S. K. Datta, C. Bonnet, and N. Nikaein, "An iot gateway centric architecture to provide novel m2m services," *in Proc. IEEE World Forum on Internet of Things*, Seoul, Korea, pp. 514-519, Mar. 2014.

[22] A. Azzara, D. Alessandrelli, S. Bocchino, M. Petracca, and P. Pagano, "PyoT, a macroprogramming framework for the Internet of Things," *in Proc. IEEE International Symposium on Industrial Embedded Systems*, Pisa, Italy, pp. 96-103, Jun. 2014.

[23] S. Gaur, N. Pereira, V. Gupta, and E. Tovar, "A Modular Programming Approach for IoT-Based Wireless Sensor Networks," *in Proc. International Conference on Embedded Wireless Systems and Networks*, Poster, pp. 3-4, Mar. 2015.

[24] M. Serna, C. J. Sreenan, and S. Fedor, "A visual programming framework for wireless sensor networks in smart home applications," *in Proc. IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, Singapore, pp. 1-6, Apr. 2015

[25] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "Iot gateway: Bridgingwireless sensor networks into internet of things," *in Proc. IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, Hong Kong, China, pp. 347-352, Dec. 2010.

BIOGRAPHIES

Yuna Jeong received the B.S. degree in computer engineering at Korea Polytechnic University (2012). She is a Ph.D. student in computer engineering at Sungkyunkwan University. Her main research interest is real-time rendering.

Hyuntae Joo received the B.S. degree in computer software engineering at Kwangwoon University (2015). He is a M.S. student in computer engineering at Sungkyunkwan University. His main research interest is physically-based real-time rendering.

Gyeonghwan Hong received the B.S. degree in computer engineering from Sungkyunkwan University, Korea in 2013. He is currently pursuing the Ph. D. degree in Embedded Software Laboratory from Sungkyunkwan University, Suwon, Korea. His research interests include embedded software, mobile system software, Internet of Things, and web platform.

Dongkun Shin (M'08) received the B.S. degree in computer science and statistics, the M.S. degree in computer science, and the Ph.D. degree in computer science and engineering from Seoul National University, Korea, in 1994, 2000, and 2004, respectively. He is currently an associate professor in the Department of Software engineering, Sungkyunkwan University (SKKU). Before joining SKKU in 2007, he was a senior engineer of Samsung Electronics Co., Korea. His research interests include embedded software, low-power systems, computer architecture, and real-time systems.

Sungkil Lee received the B.S. and Ph.D. degrees in materials science and engineering and computer science and engineering at POSTECH, Korea, in 2002 and 2009, respectively. He is currently an associate professor in the Software Department at Sungkyunkwan University, Korea. He was a postdoctoral researcher at the Max-Planck-Institut Informatik (2009-2011). His research interests include real-time GPU rendering, perception-based rendering, information visualization, GPU algorithms, and human-computer interaction.