An Approach to User Interactions with IoT-enabled Spaces

Denis Gračanin*, Mohamed Handosa*, Hicham G. Elmongui† and Krešimir Matković‡
*Department of Computer Science,
Virginia Tech, Blacksburg, VA, USA
Email: gracanin@vt.edu, handosa@vt.edu
†Department of Computer and Systems Engineering,
Alexandria University, Alexandria, Egypt
Email: elmongui@alexu.edu.eg
‡VRVis Research Centre,
Vienna, Austria
Email: matkovic@vrvis.at

***Abstract*—IoT-enabled built environments have potential to improve the lives of individuals, groups, and the broader community. Internet of Things (IoT), a collection of networked and interacting embedded devices, provides the necessary infrastructure and enabling technologies to design, develop and deploy smart built environments. We describe an approach to supporting user interaction with IoT-enabled smart built environments. This approach takes advantages of affordances and embodied cognition in a physical space to model user interaction with built spaces. The corresponding implementation leverages standard protocols (MQTT) and IoT-Lite ontology to represent IoT resources, entities and services. A proof-of-concept light-control application demonstrates the approach.**

***Keywords*—Internet of Things, human computer interaction, smart homes.**

I. Introduction

A smart environment is a physical space enriched with smart objects that work continuously to make residents lives more comfortable [1]. In contrast to the rapid development in equipment and capabilities of smart environments, less attention has been paid to the development of user interfaces and user interaction techniques that can present those capabilities to the users in an intuitive and comprehensive manner. Providing smart objects without the appropriate user interfaces can complicate user experience rather than simplifying it. Therefore, supporting convenient user interfaces is a key for the success and wide adoption of smart environments.

Smart built environments incorporate sensors and actuators into the built space to provide new functionalities or enhance the current ones. Sensors collect data about the occupants and the current context, which allows for recommending actions to occupants or performing them autonomously. Actuators can change the physical state of the built space on behalf of the occupants. Applying the Internet of Things (IoT) technologies in built environments to create smart built environments involves the transformation of traditional physical objects into smart objects (or things) by integrating processing and communication capabilities as well as sensors and/or actuators.

Supporting a seamless interaction between (smart) things and occupants is crucial for a successful application of the IoT technologies in built environments. Unlike traditional computing devices, things are usually resource-limited and they lack input and output peripherals. Therefore, the interaction between users and things usually take place through an intermediary computing devices (e.g., smartphones). The role of the intermediary computing device is to provide a user with an interface through which the user can send commands to different things and receive feedback from them.

Relying on the intermediary computing device to provide a Graphical User Interface (GUI) for user interactions has several drawbacks. First, no interaction can take place without the intermediary device. Second, mapping things located in a 3D space into a 2D GUI can be tricky for both developers and users. Third, the intermediary device can act as an explicit barrier between users and things. Consequently, supporting other interaction techniques such as voice based, gesture based, and augmented reality based can address some of the limitations associated with the traditional GUI. In fact, there is no ultimate interaction technique that is best for every scenario. Therefore, smart environments should support multimodal interaction, where the users have the flexibility to select the best interaction technique for a given scenario.

Providing user interfaces for different things usually requires developing different applications, where each application is tailored for a specific set of things. The developer needs to be aware of the peculiarities and functionalities provided by each of those things before being able to build the interaction application. On the user side, interacting with different things may require users to install and use different applications. Installing numerous applications and switching between them to interact with different things can be frustrating to the users. Moreover, users are required to have prior knowledge about the available things and the corresponding applications that they need to install before being able to interact with these things. Although this might be acceptable for a resident, it can be overwhelming for visitors.

Ideally, users should be able to interact with a smart built space as they enter that space with minimum requirements at the user side. Users should not be assumed to have prior knowledge about the space or required to install and use different applications to interact with different spaces or things. In order to achieve that, we need to address a number of questions. First, how will users discover the available things? Second, how will users know about the different supported interaction techniques? Third, how to present a user interface to the users?

In this paper, we introduce a human-centered user interaction framework that addresses the questions mentioned above and achieves two main goals. The first goal is to provide users with intuitive and multi-modal interactions with a built space. The second goal is to avoid the need for different applications to interact with different smart spaces. Consequently, GUIs hosted on intermediary computing devices can be avoided whenever possible in favor of direct interaction.

II. Related Work

The "smart" house, or "home of the future", augments a traditional home by adapting new technology is adapted into the existing patterns of use [2]. Harrison and Dourish emphasize the difference between space and place, defining place as space with added socio-cultural understandings. They argue that "place, not space, frames appropriate behaviour" [3].

Aipperspach et al. discuss the dangers of introducing information technology into the home without considering potential detrimental effects on its inhabitants [4]. Nissenbaum introduces the concept of contextual integrity to address concerns about the effects of information technology on privacy from the perspective of the law [5].

Crabtree and Rodden study domestic routines as related to communication and collaboration [2]. They introduce three major concepts: ecological habitats as places where communication media live, activity centers as places where media are produced and consumed, and coordinate displays as places where media are made available to coordinate activities.

A. Internet of Things

The IoT involves heterogeneous devices provided by different vendors. Those devices have variations in their capabilities and may use different standards. Therefore, there is a need for an architecture that can integrate such a variety of devices and allow cooperation between them. Object virtualization can provide a standard interface that exposes the features and functionalities of objects regardless of their technical implementation details.

The IoT architecture and the corresponding implementation [6], [7] differ form the traditional network architecture. A large number of devices have to be connected, most of them with limited computing and networking capabilities. The IoT devices will be deployed in various contexts [8], including wearable devices, house appliances/sensors, embedded devices/smartphones, manufacturing plants and environmental sensors. They can span large urban areas to support "smart cities" [9].

B. Middleware

Previous systems involving sensors and actuators were mostly closed systems operated by a single organization to serve a predefined specific goal. Thus, applications for such systems were relatively simple and direct interaction between applications and devices was common. On the other hand, the large scale of IoT systems raises the need for a middleware that can provide the interoperability between different devices and applications. A middleware acts as an intermediary that abstracts and standardizes the interaction between things themselves and provides a unified interface for applications and users that hides the specific peculiarities of those things. Several Architectures for the IoT have been proposed and the majority of them follow a Service Oriented Architecture (SOA) approach.

Atzori et al. [10] defined a middleware of three layers: object abstraction, service management, and service composition, where the management of trust, privacy, and security is the responsibility of all layers. The object abstraction layer exposes device functionalities through a standard web service interface and translates messages into device-specific communication commands. The service management layer contains a service repository of services associated with each object. The service composition layer allows for creating complex services by composing services provided by the service management layer and allows application to interact with objects through the provided services.

Domingo [11] proposed an IoT architecture composed of three layers: perception, network, and applications layer. The main function of the perception layer is to identify objects and gather information. The network layer is responsible for transmitting information obtained from the perception layer. The application layer consists of a set of intelligent solution that meets user needs.

C. Architecture

Jia et al. [12] divided the IoT system architecture into three layers: the perception layer, the network (or transport) layer, and the service (or application) layer. The perception layer is concerned with perceiving and collecting information about the physical world. The network layer provides transparent data transmission capability as well as an efficient, reliable, trusted network infrastructure platform to the upper layer. The service layer includes two sublayers: data management and application service. The data management sublayer is concerned with processing complex data as well as providing directory service and Quality of Service (QoS). The application service sublayer transforms information to content presented to applications and end users.

Xu et al. [13] proposed an IoT architecture composed of four layers: the sensing layer, the networking layer, the service layer, and the interface layer. The sensing layer integrates with hardware to sense and control the physical environment. The networking layer provides basic network support and data transfer over wireless and wired networks. The service layer

ConTEL 2017, ISBN: 978-953-184-224-2

creates and manages services. The interface layer provides interaction methods to the users and the applications.

Designing an architecture for the IoT is a big challenge that requires supporting many features including interoperability, openness, scalability, and robustness. The IoT will incorporate billions of heterogeneous devices connected through various types of networks with a variety of communication technologies and protocols. Therefore, the IoT architecture should support interoperability between the underlying technologies and provide users and applications with transparent services.

In order to support openness, the architecture needs to provide unified interfaces and a common language to exchange information across different IoT systems [14]. As the number of interconnected things increases scalability issues may arise, which may cause performance degradation. Supporting such a large scale of things in terms of naming, communication, device and service discovery, access authentication, protection, and control is essential for a successful IoT architecture. The openness and scale of the IoT implies high dynamicity, where things get connected on a continual basis while others may get disconnected probably due to failure. The architecture can support fault tolerance by replicating critical services and providing failure detection and recovery mechanisms.

Over time, the IoT systems may suffer from deterioration. Things may become out of synchronization due to clock drifts, which may result in application failure. Location information may become inaccurate due to unexpected movement of things, which may result in incorrect inferences. Performance of actuators may degrade due to physical wear and tear, which may result in severe safety problems in some applications. In order to maintain a robust system, the architecture must provide self-healing mechanisms such as resynchronization of clocks, relocation of devices, and recalibration of sensors and actuators [14].

Integrating different IoT systems with each other and with existing legacy systems is another challenge. Although the sharing of IoT resources can significantly reduce the deployment cost, different systems may have conflicting needs and/or goals [14]. Solving such conflicts is crucial for a seamless integration between systems. The IoT architecture should carefully define the policies for system integration and resource sharing in order to ensure the correct operation of interacting IoT systems.

The evaluation of IoT and distributed sensor systems includes real-world testing, miniature prototypes and software simulations [15]. Sensor device emulators could be used to fully replicate the behavior of the deployed sensors. While software simulations are more convenient compared to the real-world testing, most of them focus on the low-level networking aspects, with well-defined topologies and arrangements of objects. However, recent efforts are focusing on efficient simulation methodologies for large-scale IoT systems in urban environments from an application-layer perspective [16].

There are increasing efforts to develop the relevant theoretical frameworks, practical approaches, and methodologies [17]. Distributed test system frameworks for open-source IoT software have been proposed [18] to support continuous integration techniques and a permanent distributed plugtest for network interoperability testing. Since one of the key function of IoT systems is data collection and processing, it is important to understand how data analytics solution will work for IoT systems. A benchmark toolkits, such as IoTAbench [19], support testing of IoT use cases.

Although GUIs are widely adopted, other user interfaces such as gesture-based and voice-based interfaces has become more prevalent in the recent years due to the combination of improvements in computing power and access to accurate sensors [20]. Those interfaces are not supposed to replace GUI completely. However, in certain situations it may make more sense to use them rather than a GUI because they allow for direct interaction with the system with no need for a hands-on middle device like a mouse or a keyboard.

Petersen et al. [21] described a user study showing that 80% of the participants preferred to use gestures over more traditional methods such as GUIs. The participants had no problem completing their tasks after the first or second try.

III. User Interactions with IoT-enabled Spaces

Providing a customized and adaptive user interface and interactions can improve the user's performance [22]. In most cases, the customization was based on visually altering the user interface or reconfiguring the input devices based on the user preference. However, we need to move from typical human-computer interaction to human-environment interaction in a smart built space populated by smart things. The users must be able to seamlessly discover the available things, the corresponding supported interaction techniques and user interfaces.

The proposed framework incorporates the front-end clients and the back-end server, as shown in Figure 1. This separation allows for using the same back-end server with different front-ends implementing different interaction techniques. Several front-end clients implementing similar and/or different interaction techniques can communicate simultaneously with the back-end server. A front-end client takes the form of an application running on a computing device leveraging its resources to provide the user interface. For instance, a front-end client running on a smartphone can provide a graphical or a voice-based user interface.

The front-end client can rely on its server discoverer module to discover and connect to the back-end server without user intervention. Thus, the user is not required to have any prior knowledge about the back-end server. Afterwards, the front-end client communicates with the back-end server to obtain a uniform description of the user interface.

This description is represented using a platform-agnostic language that is independent of the intended interaction technique and the capabilities of the devices involved in the interaction. The user interface description interpreter module is responsible for generating a user interface that leverages the interaction device capabilities based on the description obtained from the back-end server. The user observer module

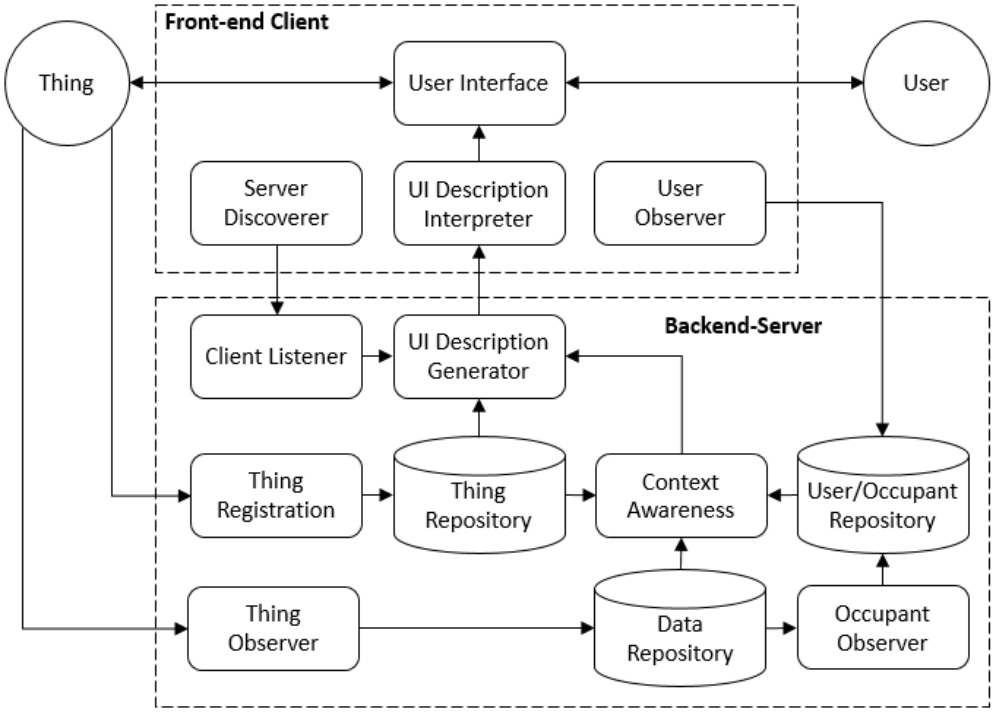

Fig. 1. A human-centered interaction framework for smart built environments.

is responsible for collecting data about the user either through the user interface (e.g. user identity if the interface requires authentication and commands issued by the user) or via the device capabilities (e.g. user location if the device has tracking capability). The information collected by the user observer is sent to the back-end server, where it is stored in the user/occupant repository.

The back-end server keeps track of all things in the built space. Information about newly installed things is added to the thing repository through the thing registration service. The thing repository has a record per thing storing information about its characteristics and functionalities. The data collected by things (e.g. sensory data streams) are monitored by the thing observer, which stores them in the data repository. The occupant observer collects data about all occupants, both users and non-users, using detection, identification, and tracking techniques (e.g. occupancy sensors, face recognition, motion tracking). The occupant observer may collect different types of data ranging from the number of occupants in a given space to the current activity of specific occupants.

The UI description generator is responsible for generating a platform-independent description of the user interface. This can take the form of listing things with potential interest to the user along with information about their characteristics and functionalities as well as a logical structure that relate those things to the current context. The user interface generator depends on the thing repository to obtain information about things and on the context awareness module to obtain rules for filtering and ranking those things.

The context awareness component leverages the data available from the thing repository, data repository, and user profiles to facilitate the generation of a user interface that is tailored to the current needs of the user. For example, the thing repository may provide information about the locations of different light sources, the data repository can provide data about the current state of those lights, and the user profiles can provide information about the current location of the user or even the current user activity. The context awareness module should leverage these information sources to provide the user interface generator with guidance. For example, if a user enters a space and the air conditioning at that space was turned off to save energy, the user interface should include and perhaps promote the air conditioning controls to meet the user's expected needs.

Ideally, a smart environment should take over and meet the needs of occupants with autonomous actions. However, this is only possible if the environment is aware of all the factors affecting all the activities and decisions of all occupants, which is usually not the case. Therefore, a user interface is required through which a user can control the space to meet the needs of the task at hand or override any autonomous decisions that conflict with those needs.

A. *Framework Implementation*

A front-end client can discover the back-end server and connect to it through the server discoverer module. One simple implementation of this module is to broadcast a query message over the entire network to a predefined server discovery port. When the back-end server receives that query, it replies with the connection information that the client needs to use in order

ConTEL 2017, ISBN: 978-953-184-224-2

Fig. 2. FutureHAUS kitchen and living room modules.

Fig. 3. FutureHAUS bathroom module.

to connect to the server. Thus, no prior information about the server is needed except for the server discovery port.

Things communicate with both the user interface and the thing observer using a lightweight communication protocol such as the Message Queue Telemetry Transport (MQTT) protocol [23] or the Constrained Application (CoAP) protocol. The thing registration service can provide an Application Programming Interface (API) using, Representational State Transfer (REST) or Simple Object Access Protocol (SOAP) that things can use to register their characteristics and functionalities [24].

The information stored in the back-end server is based on the IoT-Lite ontology [25]. It is a lightweight ontology that is used to represent different aspects of IoT resources, including smart things and context information. It supports interoperability by allowing heterogeneous platforms to use a common vocabulary. This ontology can be extended to represent the specific details associated with things in built environments. An extended version of the IoT-Lite ontology can be used as a standard data exchange format between things and the thing registration module as well as between the user interface description generator at the server-side and the user interface description interpreter at the client-side. The IoT-Lite ontology classifies IoT concepts into objects, resources, and services and classifies IoT devices into sensing, actuating, and tag devices. It is focused on sensing and describes coverage as a 2D-spatial.

The thing repository includes, among others, information about location and input/output capabilities that be used in support of multi-modal interactions. For example, lights can be used to provide a visual feedback (see Section IV-A).

IV. Case Study

Virginia Tech's FutureHAUS (http://www.futurehaus.tech) is a smart home developed as a part of the ongoing efforts to explore challenges related to creating smart built environments [26], [27] using a pre-fab delivery method. Additionally, the advanced integrated electronics that we expect to have in our homes today can be immediately integrated into the construction and assembly process.

We have constructed FutureHAUS prototype (Figures 2 and 3) using modular "cartridge" prototypes (kitchen, living room, bathroom and bedroom) for this building process which explores the possibilities of prefabricated architectural components for a home. The components can be accessed and controlled through a whole-house interface which manages and monitors appliance performance and energy use. The modules are "wired" with IoT-based devices, electronic actuators and sensors that make using the living space easy, more accessible and energy efficient. The modules are integrated in a single smart space using the proposed approach (Section III) with MQTT as the communication protocol. The goal is to integrate smart technologies into a prefabricated system while elevating the user's experience of all household activities (Figure 4).

The initial idea and architectural conceptual (Figure 4 top left) was used to create the corresponding virtual environment model (Figure 4 center). After exploring and refining the virtual environment model (Figure 4 bottom left) the modules were constructed (Figure 4 bottom right and Figures 2 and 3).

Once the modules were constructed and integrated with the

Fig. 4. FutureHAUS design and development process.

deployed framework, the FutureHAUS became a living testbed for studying user interactions with IoT-enabled spaces. In addition to more common devices, such as multi-touch screens (kitchen backsplash wall, kitchen table - Microsoft Perspective Pixel), the testbed incudes a variety of other input devices (e.g., Garmin LIDAR Lite v3 [28], LeapMotion, Kinect, Amazon Alexa/Echo, fingerprint scanner). In addition, appliances and pieces of furniture were instrumented to create smart things.

As an illustration, the kitchen module has a set of devices (appliances), where each device has an IoT-based controller (Raspberry PI or Arduino) controlling a set of sensors and/or actuators. The living room has controllable color lights and a sofa with pressure sensors. The controllers periodically send the sensor readings to the server and check the server for new commands for the actuators. Applications and users can interact with the system by querying the readings from sensors and input devices and posting commands.

The rich ecology of smart things supports the development of new multimodal interactions with smart built environments and facilitates conducting user studies to evaluate different interaction modalities. We describe one such study, a gesture based lighting control, in more details.

A. Gesture-based Lighting Control

Lighting can play an essential role in supporting user tasks as well as creating an ambiance. Although users may feel excited about the supported functionality when a complex indoor lighting system is first deployed, the lack of a convenient interface may prevent them from taking the full advantage of the system. For decades, traditional lighting systems allowed users to control light bulbs using switches; flipping the switch negates a light's current state. Figure 5 shows an example of a fairly complex set of light switches. However, with recent advances in lighting technologies, a light control system can be very complex.

The complexity of light control arises mainly from two factors. First, the adjustable light parameters (e.g., color and intensity) cannot be configured using traditional switches, which implies the need for a new light control interface. Second, there is an increased number of light sources (LED-based lighting systems can have tens or even hundreds of individually controllable light sources) [29]. Therefore, installing a switch per light source is not a scalable solution. Some lighting systems allow users to control lights using a smartphone. However, mapping lights in a 3D space to a 2D GUI can confuse the user, especially as the number of light sources increases and their distribution becomes more complex.

Mrazovac et al. [30] used a sensing glove for 3D light control. Their system allows for dimming lights as well as switching them on/off. The glove sends accelerometer data using a radio frequency transmitter to a remote module that translates it into lighting commands. The system uses the sensing glove as a remote control for a specific light source. There is no support for selecting different light sources. Only the intensity of the light can be controlled but not its color.

We developed an interaction technique for controlling indoor lights [31]. The light control system was implemented on top of the deployed framework. It is extendable and supports multiple users to control lights using either gestures or a GUI. Using a single Kinect sensor, the user is able to control lights from different positions in the room while standing or sitting down within the tracking range of the sensor. The selection and manipulation accuracy of the proposed technique together with the ease of use compared to other alternatives makes it a promising lighting control technique.

Lighting control tasks can be categorized as primitive tasks, such as turning lights on and off, or as advanced tasks, such as programming specific times for predefined lighting configurations. A 3D user interface for lighting control should support primitive lighting control tasks including the selection of one or more light sources before turning them on/off or adjusting their color if applicable. The 3D user interface should provide users with the feedback for their actions. The system should anticipate for a situation where multiple users might try to interact with the system simultaneously, in which case the system should perform conflict resolution.

B. Implementation

The control system is designed to respond to a single user at a time, even when several users are present. Once a user claims control of the light control, the lights respond only to that user until the control is released. We define a set of gestures to represent different user commands and lights themselves provide a visual feedback. User actions include:

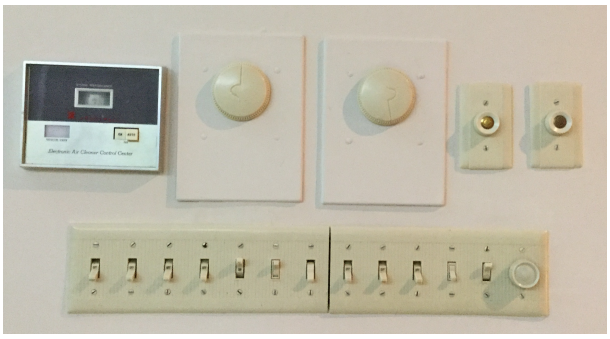

Fig. 5. A fairly complex set of light switches.

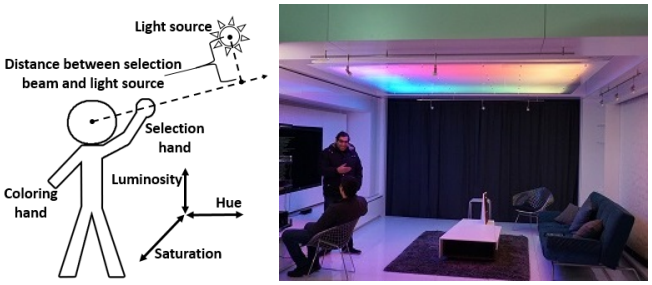

Fig. 6. Left: selecting a light source. Right: The lighting control system deployed in a living room.

ConTEL 2017, ISBN: 978-953-184-224-2

Claiming control: the user raises either hand above the head and makes a closed hand gesture. The raised hand is the selection (painting) hand and the other hand is the coloring hand. No other user can interact with the system until the user releases the control.
Releasing control: the user makes an open hand gesture using the selection hand. In addition, the control is released automatically when the user body is no longer tracked (e.g. the user left the room).
Selection: the user points the selection hand towards light sources. To simplify user's task, the system will select a light source as long as it is within a predefined distance of the selection beam as shown in Figure 6 left. Selected light sources do not have to be continuous. The user can deselect light sources by releasing control.
Switching hands: the user raises the coloring hand above the head and makes a closed hand gesture with it before opening the selection hand. At that point, the old coloring hand becomes the new selection hand. Consequently, the user can make use of both hands to select multiple light sources in various directions.
Coloring: the system assumes three virtual sliders corresponding to three color components, hue, saturation, and luminosity. The user can change the value of a slider by closing the coloring hand and moving it along the slider before opening the coloring hand at the desired value. The user closes, moves, and opens the coloring hand to catch, drag, and release the virtual slider and sees instant feedback as the selected lights change their color in response to the change in slider value. *The hue slider*: moving the coloring hand left and right. *The saturation slider*: moving the coloring hand forward and backward. *The luminosity slider*: moving the coloring hand up and down. The granularity of the hue, saturation, and luminosity sliders can be customized.

The developed lighting control system provides two methods for controlling lights, a 2D GUI and a 3D gesture-based UIs (Figure 1, front-end clients). The MQTT broker facilitates communication between the UI clients and the DMX server (Figure 1, back-end server). The 2D UI client runs on a computing device (e.g., a smartphone) that sends control commands to the DMX server. The 3D UI client uses a Kinect device to capture the user's gestures and map them to lighting control commands sent to the DMX server.

The DMX server is responsible for translating received commands into DMX commands before sending them to the Digital Multiplexing (DMX) controller. The DMX controller is a device that can control a set of light sources individually. The MQTT broker facilitates communication between the 2D UI client, the 3D UI client, and the DMX server. Figure 7 shows the implementation of the lighting control system on to of the implemented framework.

C. Testing

The control system was and tested in a living room (Figure 6 right) with 36 individually controllable LED segments mounted in the ceiling forming a rectangle with ten segments

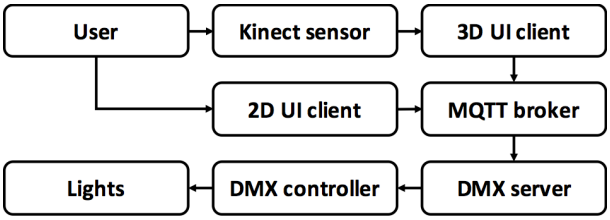

Fig. 7. The lighting control system architecture using the proposed framework.

on each of two opposite sides and eight segments of each or the other two opposite sides. The locations of those light segments in 3D space are fed to the system upon deployment. The selection tolerance can be customized to adjust selection sensitivity. The nearest light source to the selection beam is selected if its distance from the selection beam is shorter than the specified tolerance value.

The control system uses the lights to provide instant feedback to the user. A noticeable delay in response to user commands can degrade the user experience. Moreover, receiving a feedback for a previous command (e.g., change in hue) while performing a new action (e.g., modifying luminosity) can confuse the user. Therefore, the responsiveness of the system is critical for system's usability.

The system's responsiveness is determined by the delay (latency) between the time at which the user makes a given gesture and the time at which the user receives a visual feedback through the lights. Figure 8 shows six consecutive frames from one of the captured videos. The user points to a light source with an open hand (Figure 8a), which should not trigger any action. The user closes hand to claim control of the system and select the light source (Figure 8b). The user is waiting for feedback (Figures 8c, d, e). The user receives a visual feedback, a change in the color of the selected light source (Figure 8f). The total system delay ranges between 100 and 167 milliseconds which makes it very responsive and suitable for gesture based interactions.

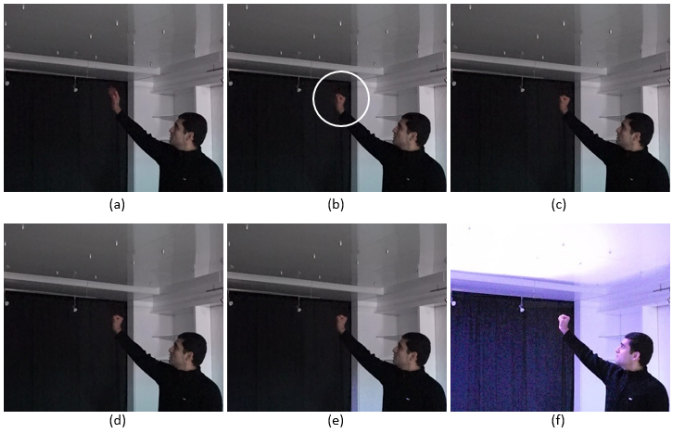

Fig. 8. Video recording frames for selecting a light source.

V. Conclusion

We described an approach and the corresponding framework to support user interactions with IoT-enabled spaces. The proposed framework consists of two main parts, the front-end clients and the back-end server, A case study, FutureHAUS smart house was used to deploy the framework. The developed gesture-based light control user interaction technique demonstrates how to provide users with intuitive and multi-modal interactions in a smart built space.

Future work will focus on refining the developed framework and expanding the supported user interactions with a smart built space. We are in process of building the next generation FutureHAUS that will take advantage of the lessons learned to support sustainability and comfort. The goal is to provide seamless energy-aware and environment-aware interaction between occupants and virtual/physical components in a smart built environment and detection of usage patterns and behaviors to inform context awareness and to identify desired comfort level and related environmental conditions.

Acknowledgment

This work has been partially supported by a grant from the Virginia Tech Institute for Creativity, Art, and Technology and by VRVis Forschungs-GmbH. The VRVis Forschungs-GmbH is funded by COMET — Competence Centers for Excellent Technologies (854174) by BMVIT, BMWFW, Styria, Styrian Business Promotion Agency — SFG and Vienna Business Agency. The COMET Programme is managed by FFG.

References

[1] N. S. Dalton, H. Schnädelbach, M. Wiberg, and T. Varoudis, Eds., *Architecture and Interaction: Human Computer Interaction in Space and Place*. Springer, 2016.

[2] A. Crabtree and T. Rodden, "Domestic routines and design for the home," *Computer Supported Cooperative Work*, vol. 13, no. 2, pp. 191–220, Apr. 2004.

[3] S. Harrison and P. Dourish, "Re-place-ing space: The roles of place and space in collaborative systems," in *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work (CSCW '96)*. New York: ACM, 1996, pp. 67–76.

[4] R. Aipperspach, B. Hooker, and A. Woodruff, "FEATURE: The heterogeneous home," *interactions*, vol. 16, no. 1, pp. 35–38, Jan. + Feb. 2009.

[5] H. F. Nissenbaum, "Privacy as contextual integrity," *Washington Law Review*, vol. 79, no. 1, pp. 119–158, Feb. 2004.

[6] F. daCosta, *Rethinking the Internet of Things: A Scalable Approach to Connecting Everything*. Apress L. P., 2013.

[7] S. C. Mukhopadhyay, Ed., *Internet of Things: Challenges and Opportunities*, ser. Smart Sensors, Measurement and Instrumentation. Springer, 2014, vol. 9.

[8] J. Anderson, L. Rainie, and M. Duggan, "The internet of things will thrive by 2025," Pew Research Center, Washington, D.C. 20036, Tech. Rep., May 2014.

[9] R. Lea and M. Blackstock, "Smart cities: An IoT-centric approach," in *Proceedings of the 2014 International Workshop on Web Intelligence and Smart Sensing*. New York: ACM, 2014, pp. 12:1–12:2.

[10] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[11] M. C. Domingo, "An overview of the Internet of Things for people with disabilities," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 584–596, 2012.

[12] X. Jia, Q. Feng, T. Fan, and Q. Lei, "RFID technology and its applications in Internet of Things (IoT)," in *Proceedinsg of the 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, Apr. 2012, pp. 1282–1285.

[13] L. D. Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.

[14] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, Feb. 2014.

[15] I. Armac and D. Retkowitz, "Simulation of smart environments," in *The Proceedings of the IEEE International Conference on Pervasive Services*, 15–20 Jul. 2007, pp. 322–331.

[16] G. Brambilla, M. Picone, S. Cirani, M. Amoretti, and F. Zanichelli, "A simulation platform for large-scale Internet of Things scenarios in urban environments," in *Proceedings of the First International Conference on IoT in Urban Space*. Brussels: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014, pp. 50–55.

[17] Z. Mahmood, Ed., *Connectivity Frameworks for Smart Devices*. Springer, 2016.

[18] P. Rosenkranz, M. Wählisch, E. Baccelli, and L. Ortmann, "A distributed test system architecture for open-source iot software," in *Proceedings of the 2015 Workshop on IoT Challenges in Mobile and Industrial Systems*. New York: ACM, 2015, pp. 43–48.

[19] M. Arlitt, M. Marwah, G. Bellala, A. Shah, J. Healey, and B. Vandiver, "ioTAbench: An Internet of Things analytics benchmark," in *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*. New York: ACM, 2015, pp. 133–144.

[20] D. Wigdor and D. Wixon, *Brave NUI world : designing natural user interfaces for touch and gesture*. Burlington, MA 01803: Morgan Kaufmann, 2011.

[21] N. Petersen and D. Stricker, "Continuous natural user interface: Reducing the gap between real and digital world," in *Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 19–22 Oct. 2009, pp. 23–26.

[22] S. Benford, D. Snowdon, A. Colebourne, J. O'Brien, and T. Rodden, "Informing the design of collaborative virtual environments," in *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work (GROUP '97)*. New York, NY, USA: ACM, 1997, pp. 71–80.

[23] M. Arlitt, M. Marwah, G. Bellala, A. Shah, J. Healey, and B. Vandiver, "MQTT version 3.1.1 plus errata 01," OASIS, Standard, 10 Dec. 2015.

[24] G. Mulligan and D. Gračanin, "A comparison of SOAP and REST implementations of a service based interaction independence middleware framework," in *Proceedings of the 2009 Winter Simulation Conference*, M. Rossettii, R. Hill, B. Johansson, A. Dunkin, and R. Ingalls, Eds., 13–16 Dec. 2009, pp. 1423–1432.

[25] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, and K. Taylor, "IoT-lite ontology," https://www.w3.org/Submission/iot-lite/, W3C, 26 Nov. 2015, [Last accessed 30 Mar. 2017].

[26] D. Gračanin, D. S. McCrickard, A. Billingsley, R. Cooper, T. Gatling, E. Irvin-Williams, F. Osborne, and F. Doswell, "Mobile interfaces for better living: Supporting awareness in a smart home environment," in *Proceedings of the HCI International 2011: Universal Access in Human-Computer Interaction: Context Diversity*, ser. Lecture Notes in Computer Science, C. Stephanidis, Ed., vol. 6767. Berlin / Heidelberg: Springer, 9–14 Jul. 2011, pp. 163–172.

[27] D. Gračanin, K. Matković, and J. Wheeler, "An approach to modeling Internet of Things based smart built environments," in *Proceedings of the 2015 Winter Simulation Conference (WSC)*, 6–9 Dec. 2015, pp. 3208–3209.

[28] Garmin Ltd., "Lidar Lite v3 operation manual and technical specifications," http://static.garmin.com/pumac/LIDAR_Lite_v3_Operation_Manual_and_Technical_Specifications.pdf, Garmin Ltd., 2017, [Last accessed 10 Feb. 2017].

[29] D. Aliakseyeu, B. Meerbeek, J. Mason, H. van Essen, S. Offermans, A. Wiethoff, N. Streitz, and A. Lucero, "Designing interactive lighting," in *Proceedings of the Designing Interactive Systems Conference*. New York: ACM, 2012, pp. 801–802.

[30] B. Mrazovac, M. Z. Bjelica, D. Simić, S. Tikvić, and I. Papp, "Gesture based hardware interface for RF lighting control," in *Proceedings of the 9th IEEE International Symposium on Intelligent Systems and Informatics*. IEEE, Sep. 2011, pp. 309–314.

[31] M. Handosa, D. Gračanin, H. G. Elmongui, and A. Ciambrone, "Painting with light: Gesture based light control in architectural settings," in *Proceedings of the 2017 IEEE Symposium on 3D User Interfaces (3DUI 2017)*, 18–19 Mar. 2017, pp. 249–250.