Novel Display and Control for IoT-Based Home Automation

Tyler Nicholas Edward Steane
RMIT University

P J Radcliffe
RMIT University

Abstract: Home automation systems have long been dependent on a permanent central controller, which has many problems, but a significant barrier to eliminating this controller is its ability to supply user interfaces to display the status of devices and control them. This paper proposes a novel protocol which allows any device or several devices, such as a smartphone, to control many devices from any manufacturer in one application in a plug-and-play manner without a central controller. Current approaches to home automation do not offer this functionality, requiring many applications from many manufacturers. The proposed novel protocol uses a standardised dictionary of UI elements and a minimalist XML device description that describes not only the UI layout for a device but also the device's capabilities and the control procedures for the device. This light-weight all-in-one XML description is a novel combination of display, capabilities, and control and is ideal for the highly contested domestic 2.4 GHz Wi-Fi space. This is achieved without the need for a permanent central controller or an Internet connection and together with other protocols allows the elimination of the permanent central controller.

Keywords: Home Automation, Internet of things, Smart homes, device description, User interface.

Introduction

Innovation in automated functionality within the home has been an area of interest for many decades now, but there has been little penetration into the mass consumer market. Recent trends have seen individual automation devices make their way into the market, but little change has been made in wider Home Automation Systems (HAS).

Such systems continue to employ a topology dependent on a Permanent Central Controller (PCC), which is responsible for coordinating the system, from configuration to display and control. However, these PCCs make systems expensive, more complex to use, and lock users

http://doi.org/10.18080/jtde.v8n1.234

into a particular manufacturer, thus reducing competition, alienating consumers and inhibiting the ubiquitous uptake of the technology.

A new paradigm for home automation is needed that allows the average home user to purchase low-cost simple Internet of Things (IoT) devices that they can take home and install and configure themselves. These devices should allow users to build up a system that is easy to use and maintain. This would replace current approaches where a different smartphone app is needed for every different device. This new paradigm will eliminate the Permanent Central Controller as the source of much complexity, cost and inflexibility in contemporary designs.

Recent work has proposed the Decentralised Home Automation Protocol (DHAP), which eliminates the PCC and redistributes its responsibilities between individual smart IoT devices and Intermittent Control Devices (ICDs) with a series of novel protocols (Steane & Radcliffe, 2018, 2019a). These ICDs, which can be any smartphone, tablet or PC, are not like PCCs as they only offer an interface to the existing system, but the system is not dependent on the ICDs to operate.

The work with DHAP has thus far demonstrated that a HAS can achieve secure device joining (Steane & Radcliffe, 2017a, 2016, 2017b), efficient device discovery (Steane & Radcliffe, 2019b) and communications (Steane & Radcliffe, 2018, 2019a) without a PCC. The final component to consider is the display and control of devices independent of a PCC. This will require devices to be able to describe their capabilities, how they may be controlled, and a user interface layout. This will allow ICDs to present a user interface that can display the status of a device and allow a user to control or interact with the device.

This paper presents a novel protocol that allows users to display and control home IoT devices and completes the DHAP functionality. This is done using a single XML device description file that brings together the description of a device's capabilities, command and control procedures, and user interface (UI) layout. Using this single and compact file stored out-of-the-box in IoT devices, ICDs can generate a fully functioning UI to display the state of the IoT devices, as well as provide the commands to control the IoT device. This will allow ICDs to achieve plug-and-play integration for IoT devices, with many devices from various manufacturers all controlled from one ICD application. Furthermore, the novel description file summarising multiple aspects of a device will allow for future development to accommodate machine-to-machine (M2M) communication between devices within the home.

Interoperability and the Perils of Lock-in

Commercial home automation products are now available that offer individual smart devices which can be managed as a single device from a smartphone or computer. This approach has

http://doi.org/10.18080/jtde.v8n1.234

some success but reduces the scope of what a smart home can achieve. Since each device may require its own app, this may reduce the number of devices that users are willing to implement or curtail the ease with which they use devices as they navigate the many applications on their phone.

This is a far cry from the vision for home automation proposed by this paper, in which many devices from many manufacturers could be connected together and controlled from one smartphone application. Some manufacturers are expanding their product range, which may have started with a single smart light bulb but now includes smart mains switches and even kitchen appliances. This does little to improve the situation as it reintroduces the lock-in strategy where users who have bought one device from a manufacturer are locked in to that manufacturer's product range and its feature offerings, or are faced with the drastic complexity increase and inconvenience of operating and maintaining multiple systems and their apps.

This is not a new concern: early research has considered more formal Home Automation Systems and the need to integrate these for optimised user experience. The idea pursued then was interoperability (Aragues *et al.*, 2012; Miori, Russo & Aliberti, 2010). The main issue with most of this work was that it inevitably increased reliance on the central controller; in some cases it compounded the issue by creating a kind of 'super' central controller, translating communications between central controllers of other home automation systems (Miori & Russo, 2014).

Recent commercial developments have seen a new approach to interoperability offered by the Google, Amazon and Apple voice assistants. These are offering different manufacturers the ability to make their smart devices compatible with their voice assistants. However, this again compounds the issue of the central controller, this time moving it to the cloud, which raises more concerns, but also offers less interoperability than previous work. In this approach, the Permanent Central Controller (PCC) is now in the cloud but, like all approaches seeking this topology, the central controller cost is multiplied. At this stage the costs of the local CC, the voice assistant hub or speaker, are minimal and cloud resource costs are borne by the voice assistant manufacturer but, as ever, ' if you are not paying for the product, you are the product' and many concerns have already been raised as to the privacy of the data collected by these voice assistants (Brodkin, 2019; Lee, 2019; Washenko, 2019). For all this trade-off, little interoperability is achieved as the voice assistant merely offers a central interface for all devices, a one-app solution as it were. This goes some way in bringing together smart devices from different manufacturers, but the voice assistant market is still new and manufacturers do not support all voice assistants and will even vary their supported assistance between products.

http://doi.org/10.18080/jtde.v8n1.234

Literature Review

The display and control of IoT devices in Home Automation Systems can be categorised into three major types: device hosted, externally hosted and custom apps.

In the device hosted approach the smart device itself hosts its own user interface, often running a cutdown web server to host a webpage interface. Recently, the more common implementations of this approach utilise a Raspberry Pi as the smart device and run a webserver to host the UI (Patchava, Kandala & Babu, 2015; Rukmini & Devi, 2016).

This approach greatly reduces the dependence on wider infrastructure and can be considered independent of a PCC. However, they do not by themselves constitute a HAS as they offer only one specialised functionality and do not easily interact with other devices. So, while devices do not need to describe themselves, this is due to their isolation from other devices. Additionally, the extra computational capability and power requirements add to the initial and maintenance costs of the device.

Externally hosted methods house the interface on a central server and the interface is then accessed via some connection to the central server. This allows greater integration with other devices and reduces the computational load on individual devices when compared with device hosted approaches.

External hosting can be further broken down into locally hosted interfaces like Miori and Russo's DomoNet (Miori & Russo, 2014) or Baresi, Sadeghi and Valla, who proposed TDeX (Baresi, Sadeghi & Valla, 2018), while others have hosted the interface in the cloud (Dickey, Banks & Sukittanon, 2012; Gurek *et al.*, 2013). DomoNet and TDeX have some valuable contributions for device description and UI generation but this approach greatly increases costs with continued reliance on a PCC, which is a single point of failure for the whole system.

This vulnerability is far worse in clouded approaches where manufacturers must maintain the central servers for the products to continue to function; this has already resulted in smart devices losing functionality due to manufacturers closing their servers (Dellinger, 2019; Statt, 2019). While this is a favourable model for manufacturers who can charge ongoing service fees, for example Nest security cameras (Nest, 2019), it raises concerns for users who still have to pay the initial outlay for devices that are only useful while manufacturers support access to the control server in the cloud.

All other hosting options allow for devices to be run offline but, if the central control and interface hosting is in the cloud, then all functionality is lost without an Internet connection. There are additional concerns over security as to who can see your devices and their data, and who can gain control.

http://doi.org/10.18080/jtde.v8n1.234

The use of custom apps splits up the responsibilities of display and control, allowing the interface to be hosted on a control device like a smartphone and the control to be held elsewhere. Implementations such as Cheuque *et al.* (2015), and Thiyagarajan & Raveendra (2015) host a user interface in a custom-designed Android app, which can send control commands to a local central controller or directly to the device. Others have shown that the same approach can be adopted with a clouded central controller (Fahim *et al.*, 2012; Sutiono, Nugroho & Karyono, 2016).

These solutions maintain the reduced computational demands seen in external hosting but may still have increased hardware requirements depending on where the control functionality is hosted, so cost benefits vary greatly. If a home has devices from multiple manufacturers, then many custom apps must be used to control the home, which makes for a poor user experience.

Very little work considers the combination of device display with work already considering device descriptions for capabilities and control. UPnP is perhaps the first exception with its "step 5" presentation allowing for a URL pointing to a user interface (UPnP Forum, 2015). However, this is as far as the standard goes and does not give scope for how the interface might be defined and would ultimately rely on one of the previously discussed methods for presenting a display.

DomoNet and TDeX, however, have perhaps come the closest to bringing device descriptions together with displays. Primarily motivated by interoperability, DomoNet's standard makes XML descriptions of each device from different HAS to allow interconnectivity. The super central controller integrates central controllers from at least 6 different HAS, then generates a webpage to control the devices based on their capabilities. However, this approach does not produce particularly user-friendly interfaces and layout is really an afterthought.

Similarly, TDeX allows for interoperability of devices from different manufacturers; unlike DomoNet it does not host the UI's for smart devices but instead serves out description files to interface devices like an Android smartphone. This approach reduces the system's dependence on a central controller but, none the less, it is still dependent on the "M4HSD" service, which perpetuates the need for a Central Controller.

Others have considered describing graphical user interfaces (GUIs) from XML (*Layouts*, 2019; *XUL*, 2019; Thommes *et al.*, 2012) but no work has considered how a single schema could be used to include device capabilities to control devices.

http://doi.org/10.18080/jtde.v8n1.234

Proposed Solution

As has been described, device description in one form or another is not a new idea, but no work to date has offered a solution that allows one description file to enable an ICD to display a device's status, present a layout for a control UI, and co-ordinate control commands. Our proposed solution achieves these key goals and also requires small and simple packets, and no dependence on a central controller. It is envisioned that this solution will become a standard for describing device capabilities and how to display and control devices but that multiple different applications might be developed to implement the standard, thus allowing customers to choose the implementation that works best for them. Manufacturers could also contribute applications that might be best suited to their devices but that would also connect with devices from other manufacturers.

Devices will be described by a minimalist XML device description stored on the IoT device that will allow open implementation of device interfaces while maintaining some control over the UI's layout and structure for manufacturers. This description will define all control and display elements that should be present in a UI for a device. When an ICD discovers a new IoT device on the network for the first time, it will receive the IoT device's description allowing the ICD to generate a UI based on the description.

ICDs will generate GUIs from the XML device descriptions using a local dictionary parser and a theme/skin generator, installed as part of the ICD app. The dictionary will be an open and public standard that defines what each element is and how it should be described, while the theme/skin generator will be an implementation-specific definition of how an element should appear. This is similar to the concept used in HTML where a web page uses the standard definitions to create a section of text that is "bold" but individual browsers will determine what bold actually looks like. This will allow different ICD developers to produce their own themes and skins that may choose alternative colour schemes or representations of elements — for example a toggle element may be represented as a switch or as a button changing colour — but the standard set of elements will allow manufacturers to control the essential layout and functionality of the GUI. (Manufacturers could produce their own themes/skin generators and apps).

Each element in a description will have an ID, which will allow a device's status data to be directed to the correct display elements. IDs will also be used when control elements are activated; the ICD will send a generic control message to the IoT device identifying the element's ID and any associated payload, thus allowing the ICD to remain agnostic as to the nature of the IoT device or the commands the user is sending. Thus, a description based largely around the UI layout will also cover device control and capabilities.

http://doi.org/10.18080/jtde.v8n1.234

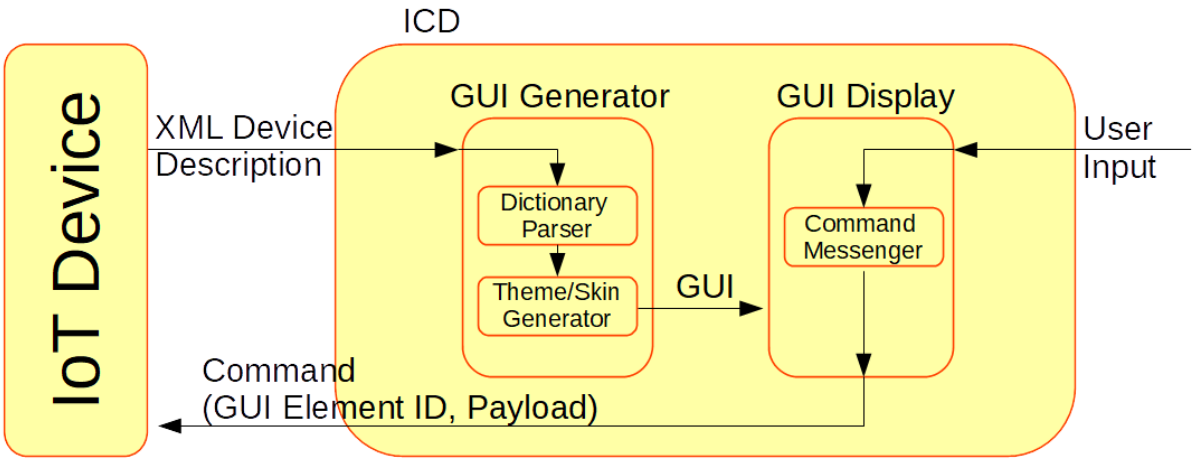

Figure 1. Display and control protocol diagram

The proposed protocol, summarised in Figure 1, uses an XML device description and requires all IoT devices to store this file and distribute it to ICDs as requested. ICDs can then use this description and run it though a dictionary to generate a GUI. Users can then interact with the GUI and requested actions will be sent back to the IoT devices as generic notifications, which can then be interpreted into the appropriate actions.

The abstract description

The abstract XML, see Figure 2, defines individual UI elements and groups them based on how they should be displayed, and IDs are used to help add order to these groups. The ICD takes this definition and generates the appropriate UI, which will return any user interaction as a reference to the UI element ID and any relevant payload that the IoT device can interpret and act accordingly. Thus, the ICD can remain largely agnostic as to the nature and behaviour of any device, while still providing a user-friendly UI and the ability to control the device.

```
<device>
  <name> string </name>
  <location> string </location>

  <group permission = INT visibility = Bool
         frame = Bool orientation = Bool>
    <label> string </label>
    <gui_element> </gui_element>
    .
    .
    .
  </group>

  <gui_element id = UNIQUE INT>
    <type>string from dictionary</type>
    <disp_settings>string</disp_settings>
    <status_location> INT </status _location>
    <comment> String </comment>
    <topic> String </topic>
  </gui_element>
</device>
```

Figure 2. Abstract XML device description

The XML device description does not include any status variable or values, as these are requested later in a separate status update packet. This allows a distinction to be made

http://doi.org/10.18080/jtde.v8n1.234

between regularly changing values and immutable or less variable values, which reduces packet sizes and therefore airtime, which is crucial in the highly contested 2.4 GHz Wi-Fi space (den Hartog *et al.*, 2017). (Though 5 GHz or Ethernet may alleviate this issue, they each come with their own limitations. For example, 5 GHz is still not as common and has less range than 2.4 GHz and Ethernet is less flexible, needing the cable to be installed.) These update packets are ordered (comma separated) and can be decoded in order to update the relevant UI elements. Again, all the specificity of the IoT device is handled by the device itself, while the ICD merely acts based on its definitions of each element; thus no data types are defined by the XML as the definition of each element is standard and thus the data type is implied.

The <device> tag bounds the full description of a device. Devices will also have a name and location specified, using the corresponding tags. These tags will be populated based on how users assign names and rooms/areas to devices. This will allow the device to be easily identified by users and be grouped by rooms or areas. The user experience will therefore be improved as all devices used in the Kitchen, for example, could appear under a heading "Kitchen" as these interfaces will likely be used together, while the user is in that room.

The <group> tag is used to collect <gui_elements>, which correspond to individual elements of the display, and ensure that they are displayed together for increased usability. Groups have permission attributes to allow for different XML descriptions to be generated from a master file to control read/write permissions. The remaining group attributes control layout settings, for example if a group relates to advanced settings and should be hidden by default or if the collection of elements should be bound by a frame and stacked horizontally or vertically.

Each <gui_element> has a unique id attribute for identification and then the <type> tag selects the element type from a predefined dictionary standard, e.g. Button, Toggle, Textbox or Radio. The <disp_settings> tag primarily supplies a label to the elements for human readability, but also includes any other details (using comma separation) for display settings, like the limits of a slider's range or the maximum number of characters for a text field.

The <status_location> tag specifies the position, in a status update packet, of any data to be displayed by this element, while the <comment> tag specifies a help message or prompts for users and the <topic> tag identifies that corresponding status value as a variable of potential interest to other devices, allowing for M2M functionality.

The abstract XML in Figure 2 could be used to describe any device and could specify any gui_elements but, in working towards a public standard, a dictionary of known gui_elements would be available. Manufacturers could be certified as compliant with the standard by using only gui_elements as described in the public dictionary. Likewise, certified applications for

http://doi.org/10.18080/jtde.v8n1.234

ICDs would need to be capable of appropriately handling all specified dictionary gui_elements.

Method

In order to assess how comprehensive this abstract XML is and how easy it is to work with, two developers with experience in Android, iOS and web development were asked to use the abstract XML to develop UIs for 10 IoT smart home devices:

1. Wireless Speaker;
2. Security Camera;
3. Thermostat;
4. Light Globe;
5. Mains switch;
6. Oven;
7. Television;
8. Garage door;
9. Kettle;
10. Curtains/blinds.

From these UIs a list of unique UI elements was compiled, and the developers assessed the viability of properly implementing each element from the XML abstract and provided feedback. After changes were made to the abstract XML from the developers' feedback, two instances of the XML were implemented to generate full UIs for two of the example devices considered.

Refinement

The developers identified 12 unique elements that were required to comprehensively implement convenient user interfaces for the 10 devices considered. In considering the implementation of these interfaces, the developers' feedback highlighted the need to add the orientation attribute to the group tag to give better control of the layout in groups. It was also suggested that the scope of the <disp_settings> tag be expanded from a simple label to include more details that vary between GUI elements, such as ranges for sliders and details for drop-down menus. These changes were integrated as discussed above.

The 12 unique elements identified are summarised in Table 1. Several of these elements have been chosen to offer very specific functions, such as a password field shown in Figure 3 and Figure 4, while others have been kept as general as possible to maximise usage. A good example of maximum used usage is the button Group, shown in Figure 5 and Figure 6, which

http://doi.org/10.18080/jtde.v8n1.234

is a 2-D array of buttons which can also be used to generate a single button (1x1) as well as a row or column of buttons.

Table 1. Essential GUI elements with description

Element	Description
Switch Toggle	A two-state element
Button Toggle	A two-state button
Plus-minus Button	A pair of buttons for increment/decrement functions
Button Group	A 2-D array of buttons, including 1-by-1.
Directional Buttons	Four buttons arranged for directional inputs e.g. N, S, E, W; or up, down, left, right
Drop-down selection	Element to select one option from a list.
Range Input	Element for numeric value selection using a sliding bar.
Status/Label	Text String
Progress	Element to visualize progress
Scheduler	Collection of elements used to schedule events, includes a list of events to select and a date/time picker.
Text Input	Text input field
Password	Text input field which hides input.

```
<group id="1" permission="WR">
  <gui_element id="1">
    <type>password</type>
    <disp_settings>Password,
                Submit</disp_settings>
    <status_location>1</status_location>
    <comment>Enter your password</comment>
  </gui_element>
</group>
```

Figure 3. Example XML instance for a password entry field

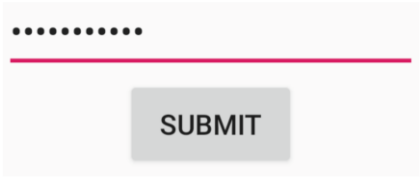

Figure 4. UI-generated password field using XML in Figure 3

http://doi.org/10.18080/jtde.v8n1.234

```
<group id="1" permission="WR">
  <gui_element id="1">
    <type>buttongroup</type>
    <disp_settings>Numpad,1,2,3,4</disp_settings>
    <status_location>1</status_location>
    <comment>Numpad</comment>
  </gui_element>
</group>
```

Figure 5. Example XML instance for a row of 4 buttons

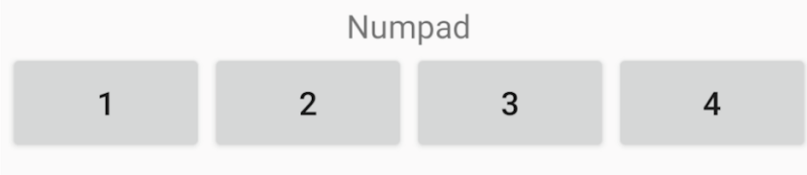

Figure 6. UI-generated row of 4 buttons using XML in Figure 5

Implementation

To support this novel protocol, applications have been developed to run on smartphones under Android and iOS, as well as a cross-platform electron application for desktops, laptops and tablets running Windows, Mac and Linux (Steane *et al.*, 2019). All these applications are supported by libraries that allow an abstract XML file from an IoT device to be read and run through the elements dictionary and generate a fully functional UI that can display the current state of a device and send commands to control the device. All 12 elements identified by the developers have been successfully integrated into these libraries.

The full functionality of this simple abstract definition is demonstrated in the fully functional UIs it can generate. Two fully implemented XML abstract files from the developers were used by the Android application to implement a Thermostat UI, Figure 7, and a Security Camera UI, Figure 8.

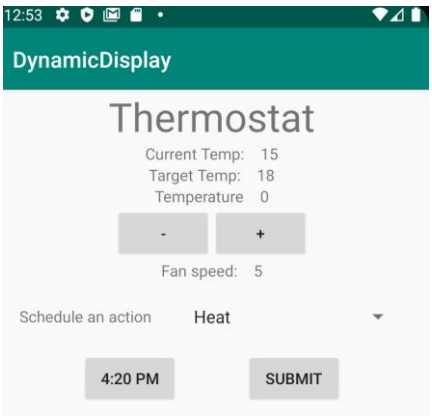

Figure 7. Example of a Thermostat UI generated from an XML device description

Figures 7 and 8 demonstrate the capabilities of the proposed simple XML abstract to develop intuitive and user-friendly interfaces without the need for a central controller.

http://doi.org/10.18080/jtde.v8n1.234

As has been mentioned, the 2.4 GHz Wi-Fi space is highly contested in the home environment and so minimised airtime is crucial to both minimise usage of this space and to maximise successful usage. The protocol proposed already minimises the impact of transmissions by limiting them to one-off communications when an ICD first discovers a new device. The XML device descriptions further minimise this impact by using the simple and concise XML Abstract, as presented. The UIs in Figure 7 and Figure 8 both have file sizes under 2 KB; and this is with a human-readable form that could be surrendered for a more lightweight machine-readable form to further reduce airtime.

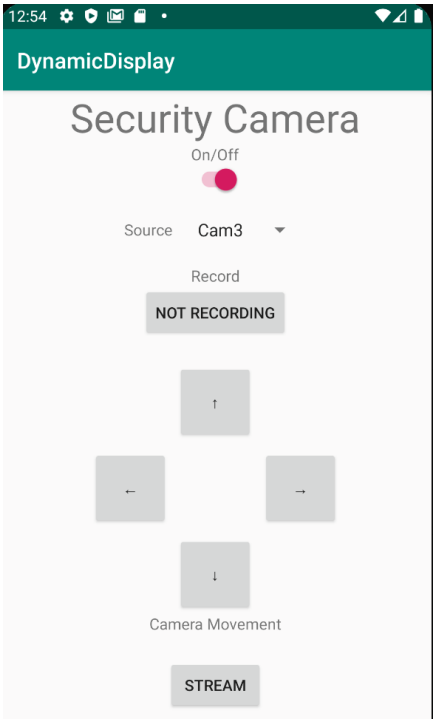

Figure 8. Example security camera UI generated from an XML device description

Analysis and Comparison

Our proposed solution offers an approach that would ensure interoperability from all complying manufacturers and control all devices from a single application. Table 2 shows a comparison of the protocol proposed in this paper and the single product or small product range systems available today from Philips (*Meethue*, 2019), Belkin (*WEMO*, 2019) and Nest (*Nest & Google—The best of Google. The best of Nest.*, 2019). It compares the application size as shown on the Android Play Store for a Pixel 3 (app sizes can vary from device to device), the opensource nature of the application and protocols, manufacturer independence for interoperability, OS independence of the application for interfacing with products and, finally, any dependence on a central controller.

http://doi.org/10.18080/jtde.v8n1.234

Table 2. Comparison of Home Automation products with proposed system

System	App Size (MB)	Open Source	Manufacturer Independent	OS Independent	PCC Dependent
DHAP	2.13	Yes	Yes	Yes	No
Philips Hue	20.36	No	Limited	Custom App per Platform (No Linux)	Yes
Belkin WeMo	39.37	No	Limited	Mobile OS only	No
Nest Thermostat	56.42	No	Limited	Mobile OS + Web App	No

Application size is of most significance if multiple applications are being installed and, while all the applications considered are minimal in comparison to the typical availability of memory on smartphones, it is evident that DHAP has a much smaller size owing to the fact that much of the UI is generated after install and with the addition of XML files from the smart device itself.

Manufacturer independence is the key to interoperability and, while DHAP is designed to completely satisfy this criterion, the other systems are not truly independent but do allow for some interoperability with other devices via a voice assistant or some other negotiated arrangement, all of which require custom integration. Notably, the Nest thermostat is widely compatible with a variety of heating and cooling systems, but this is not the same as interoperability where a smart device is cooperating with another. For example, the Belkin WeMo smart switches controlling power to a lamp, a TV or a radio are not exhibiting interoperability.

Compatibility with different Operating Systems is important for accessibility and market uptake (Steane & Radcliffe, 2017a). This paper has demonstrated the accessibility of the DHAP application on Android, iOS and major desktop OSs including Mac, Linux and Windows thanks to the Electron cross-platform application. The other systems considered have all identified the importance of the mobile Operating Systems but have largely neglected the desktop systems. Philips has covered Windows and Mac but with custom applications which would introduce a heavy development burden, while the Nest thermostat offers a web app for cross-platform access at a reduced development burden, much like the electron app.

Finally, dependence on a Permanent Central Controller (PCC) is a key question, and most of the manufacturers offer some way around using a PCC. However, as has been discussed, this has come at the cost of interoperability. So, while Philips Hue still requires a central controller to bridge Wi-Fi and ZigBee protocols, WeMo and Nest, like DHAP, can operate without a central controller in some configurations.

DHAP shows real promise to provide a universal protocol to connect smart devices in the home. The union of display, status and control provides an application that is small but allows

http://doi.org/10.18080/jtde.v8n1.234

devices from any compliant manufacturer to connect and be controlled from the one application. DHAP allows cross-platform access and is not dependent on a central controller in any configuration.

Future Work

The current state of the DHAP protocol allows many devices from many and any compliant manufacturer to be joined to and discovered on a Wi-Fi network, and now to be viewed and controlled from a single application. This gives the protocol a reasonable level of completeness. Current work is finalising the full integration of all these features into one library for easy development of HAS devices independent of a PCC.

Further work should consider the ability of the proposed abstract XML file to allow M2M communications in which an IoT device may be able to communicate to another IoT device as configured by an ICD using information from the abstract XML files.

Finally, this paper has not considered in great detail the many aspects of security that need to be addressed, not just in home automation but in IoT systems generally. This is an important area worthy of careful consideration and should be the subject of a future paper.

Conclusion

This work has demonstrated that a single application on an Intermittent Control Device (ICD), such as a smart phone, can provide a functional User Interface (UI) for devices from any manufacturer. Recent work has made this protocol open and freely available (Steane *et al.*, 2019).

This novel protocol allows manufacturers to compactly and easily describe the capabilities, control procedures, and UI layouts of their devices in a single XML file, which they embed in their devices, using the novel abstract XML proposed. This allows manufacturers to retain influence over the UI, in its structure, layout and elements — and therefore user experience — without having to maintain custom applications or cloud services.

The novel protocol allows users to easily display and control many home IoT devices from many manufacturers using a single application, without dependence on a permanent central controller or an Internet connection. This allows users to more easily construct their own home automation system, at more competitive prices, and have the ability to select the best device with the best features for their use case.

The abstract XML file proposed in this protocol is a single compact file that describes a device's capabilities, how it can be controlled, and how it should be displayed to users. The combination of all these tasks in a single XML abstract is a novel and powerful approach.

http://doi.org/10.18080/jtde.v8n1.234

A simple abstract XML description and dictionary has been presented. It has been proven to allow IoT developers to create XML device descriptions of complex IoT devices which can automatically generate fully functional UIs on ICDs.

The abstract XML and supporting protocol presented will allow plug-and-play integration of many IoT devices from various manufacturers. Together, these will all be accessible from one application on smartphones, tablets or PCs and will eliminate completely any dependence on a Permanent Central Controller (PCC) for home automation systems.

Acknowledgements

Special thanks to Aiden Garipoli and Daniel Milner for their assistance in testing and implementing this research. This research was supported by an Australian Government Research Training Program Scholarship.

References

Aragues, A., Martinez, I., Valle, P. D., Munoz, P., Escayola, J., & Trigo, J. D. (2012). Trends in entertainment, home automation and e-health: Toward cross-domain integration. *IEEE Communications Magazine*, *50*(6), 160–167. https://doi.org/10.1109/MCOM.2012.6211501

Baresi, L., Sadeghi, M., & Valla, M. (2018). TDeX: A Description Model for Heterogeneous Smart Devices and GUI Generation. *2018 IEEE International Conference on Internet of Things (IThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 97–104. https://doi.org/10.1109/Cybermatics_2018.2018.00049

Brodkin, J. (2019). Google workers listen to your "OK Google" queries—One of them leaked recordings. *Ars Technica*, July 11. https://arstechnica.com/information-technology/2019/07/google-defends-listening-to-ok-google-queries-after-voice-recordings-leak/

Cheuque, C., Baeza, F., Marquez, G., & Calderon, J. (2015). Towards to responsive web services for smart home LED control with Raspberry Pi. A first approach. *2015 34th International Conference of the Chilean Computer Science Society (SCCC)*, 1–4. https://doi.org/10.1109/SCCC.2015.7416594

Dellinger, A. (2019). Social robot Jibo does one last dance before its servers shut down. *Engadget*, March 4. https://www.engadget.com/2019/03/04/social-robot-jibo-shutting-down-message/

den Hartog, F., Raschella, A., Bouhafs, F., Kempker, P., Boltjes, B., & Seyedebrahimi, M. (2017). A pathway to solving the Wi-Fi tragedy of the commons in apartment blocks. *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, 1–6. https://doi.org/10.1109/ATNAC.2017.8215382

Dickey, N., Banks, D., & Sukittanon, S. (2012). Home automation using Cloud Network and mobile devices. *2012 Proceedings of IEEE Southeastcon*, 1–4. https://doi.org/10.1109/SECon.2012.6197003

Fahim, M., Fatima, I., Lee, S., & Lee, Y. K. (2012). Daily life activity tracking application for smart homes using android smartphone. *2012 14th International Conference on Advanced Communication Technology (ICACT)*, 241–245.

Gurek, A., Gur, C., Gurakin, C., Akdeniz, M., Metin, S. K., & Korkmaz, I. (2013). An Android based home automation system. *2013 High Capacity Optical Networks and Emerging/Enabling Technologies*, 121–125. https://doi.org/10.1109/HONET.2013.6729769

Layouts. (2019). Android Developers. https://developer.android.com/guide/topics/ui/declaring-layout

Lee, T. B. (2019). Amazon admits that employees review "small sample" of Alexa audio. *Ars Technica*, April 11. https://arstechnica.com/tech-policy/2019/04/amazon-admits-that-employees-review-small-sample-of-alexa-audio/

Meethue. (2019). Hue. https://www2.meethue.com/en-au

Miori, V., & Russo, D. (2014). Domotic Evolution towards the IoT. *2014 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 809–814. https://doi.org/10.1109/WAINA.2014.128

Miori, V., Russo, D., & Aliberti, M. (2010). Domotic Technologies Incompatibility Becomes User Transparent. *Communications of the ACM*, *53*(1), 153–157. https://doi.org/10.1145/1629175.1629211

Nest. (2019). *Nest Aware | Continuous video recording for Nest Cams*. Nest. https://www.nest.com/au/cameras/nest-aware/

Nest & Google—The best of Google. The best of Nest. (2019). https://store.google.com/ie/category/google_nest?hl=en-IE&GoogleNest&utm_source=nest_redirect&utm_medium=google_oo&utm_campaign=GS102516

Patchava, V., Kandala, H. B., & Babu, P. R. (2015). A Smart Home Automation technique with Raspberry Pi using IoT. *2015 International Conference on Smart Sensors and Systems (IC-SSS)*, 1–4. https://doi.org/10.1109/SMARTSENS.2015.7873584

Rukmini, M. S. S., & Devi, D. B. G. (2016). Remote control of appliances based on Raspberry pi. *2016 Second International Conference on Cognitive Computing and Information Processing (CCIP)*, 1–4. https://doi.org/10.1109/CCIP.2016.7802863

Statt, N. (2019). Best Buy is leaving smart home users in the cold, but its Wi-Fi freezer will still mostly work. *The Verge*, September 6. https://www.theverge.com/2019/9/6/20853671/best-buy-connect-insignia-smart-plug-wifi-freezer-mobile-app-shutdown-november-6

Steane, T. N. E., Milner, D., Garipoli, A., & Radcliffe, P. J. (2019). *Decentralised-home-automation-protocol*. GitHub. https://github.com/decentralised-home-automation-protocol

Steane, T. N. E., & Radcliffe, P. (2018). Multiple Intermittent Controllers for IoT Home Automation. *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, 1–6. https://doi.org/10.1109/ATNAC.2018.8615433

Steane, T. N. E., & Radcliffe, P. J. (2016). An enhanced implementation of a novel IoT joining protocol. *2016 26th International Telecommunication Networks and Applications Conference (ITNAC)*, 22–25. https://doi.org/10.1109/ATNAC.2016.7878776

Steane, T. N. E., & Radcliffe, P. J. (2017a). An Evaluation and Enhancement of a Novel IoT Joining Protocol. *Journal of Telecommunications and the Digital Economy*, *5*(2). https://doi.org/10.18080/jtde.v5n2.92

Steane, T. N. E., & Radcliffe, P. J. (2017b). A universal iot joining protocol for DIY applications. *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, 1–3. https://doi.org/10.1109/ATNAC.2017.8215360

Steane, T. N. E., & Radcliffe, P. J. (2019a). IoT status communication for home automation. *International Journal of Computing*, *18*(3), 240–248. Scopus.

Steane, T. N. E., & Radcliffe, P. J. (2019b). A Novel Discovery Protocol for IoT Based Home Automation. *International Journal of Automation and Smart Technology*, *9*(3), 147-158–158. https://doi.org/10.5875/ausmt.v9i3.2076

Sutiono, M., Nugroho, H., & Karyono, K. (2016). ApplianceHub: A wireless communication system for smart devices (case study: Smart Rice Cooker). *2016 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*, 125–130. https://doi.org/10.1109/ICRAMET.2016.7849597

Thiyagarajan, M., & Raveendra, C. (2015). Integration in the physical world in IoT using android mobile phones. *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, 820–826. https://doi.org/10.1109/ICGCIoT.2015.7380576

Thommes, D., Gerlicher, A., Wang, Q., & Grecos, C. (2012). RemoteUI: A high-performance remote user interface system for mobile consumer electronic devices. *IEEE Transactions on Consumer Electronics*, *58*(3), 1094–1102. https://doi.org/10.1109/TCE.2012.6311361

UPnP Forum. (2015). *UPnP Device Architecture 2.0*. http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v2.0.pdf

Washenko, A. (2019). Siri records fights, doctor's appointments, and sex (and contractors hear it), *Ars Technica*. https://arstechnica.com/gadgets/2019/07/siri-records-fights-doctors-appointments-and-sex-and-contractors-hear-it/

WEMO. (2019). https://www.wemo.com/

XUL. (2019). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XUL