Interoperability of Heterogeneous Appliances in Home Automation Using the AllJoyn Framework

Robert A. Sowah, Kwaku O. Apeadu, Abdul Ofoli*, Koudjo Koumadi, [1]Amevi Acakpovi and Stephen K. Armoo
Computer Engineering Department, University of Ghana
PMB 25, Legon, Accra-Ghana
[*]Electrical and Computer Engineering Department, University of Tennessee Chattanooga
[1]Electrical and Electronics Engineering, Accra Technical University, Accra-Ghana
{rasowah@ug.edu.gh, ekoapeadu@outlook.com, abdul-ofoli@utc.edu, koumadi@gmail.com, acakpovia@gmail.com, and skarmoo@ug.edu.gh}

Abstract— The world is changing rapidly with the advent of advances in microprocessor technology and embedded systems. Human living experiences are being transformed into one where seamless integration with connected devices to our homes, offices and healthcare providers are enhanced. The evolution of the internet has made regular computers become ubiquitous and amorphous networks of everyday machines like refrigerators, washing machines, vending machines, and computers, which are actively exchanging information – The Internet of Things (IoT). One major issue confronting the large-scale development and deployment of the IoT is the lack of standards and common business models. Therefore, this paper sought to develop Magnum: a model hub-based home automation system that uses the AllJoyn framework in the Universal Windows Platform together with web and mobile extensions to facilitate remote access. Testing and implementation of a prototype proved very successful, efficient and innovative.

Keywords— Internet of Things, Interoperability, Seamless integration, AllJoyn Framework, home automation.

I. INTRODUCTION

In recent years, the new normal in technological innovation is to have a connected world where at the click of a button, at touch of a screen, or even at one's word, lights are dimmed, music is played, air conditioning is turned on, doors are locked, personal vitals can be checked, health of a remote industry equipment can be checked and live alerts can be received directly from the machines.

To make things even smarter, the world is moving to a point where a coffee maker brews a warm cup by the time one wakes up, the house has returned cooled or warmed up to a comfortable temperature and the food one left in the oven is already warm when one returns from home, and all lights one may have forgotten go off automatically and one's security system is armed when one goes to bed. This level of home automation is referred to as the Smart Home.

To achieve connectivity of all these devices, the internet is required. Thus, devices that are traditionally not meant for the internet, are now using the internet to actively exchange information. Hence it is no longer an internetwork of everyday computers: it is the Internet of Things (IoT).

The IoT is still in early stages of development. Yet it topped Garner's Hype Cycle for Emerging Technologies in 2014, overtaking Big Data. In ascending to the peak, the IoT has joined a new wave of technologies with the potential to create new living experiences, including 3D printing, gamification, wearable user interfaces and cloud computing [1].

In fact, the term IoT itself is yet to be clearly defined.

Jennifer Pellet in the March/April 2015 issue of the Chief Executive Magazine described the Internet of Things as the employment of advances in embedded sensors, processing, data analytics and wireless connectivity to enable the type of machine-to-machine (M2M) communication that can revolutionize [1].

According to Bandyopadhyay and Sen in [2], the phrase Internet of Things (IoT) heralds a vision of the future Internet where connecting physical things, from banknotes to bicycles, through a network will let them take an active part in the Internet, exchanging information about themselves and their surroundings.

In a report by McKinsey on the IoT, the Internet of Things is defined as sensors and actuators connected by networks to computing systems whereby these systems can monitor or manage the health and actions of connected objects and machines the natural world, people, and animals [3].

Looking at it from the standpoint of the use cases of the IoT, the McKinsey report summarizes the IoT in 9 settings as illustrated in the figure below [3].

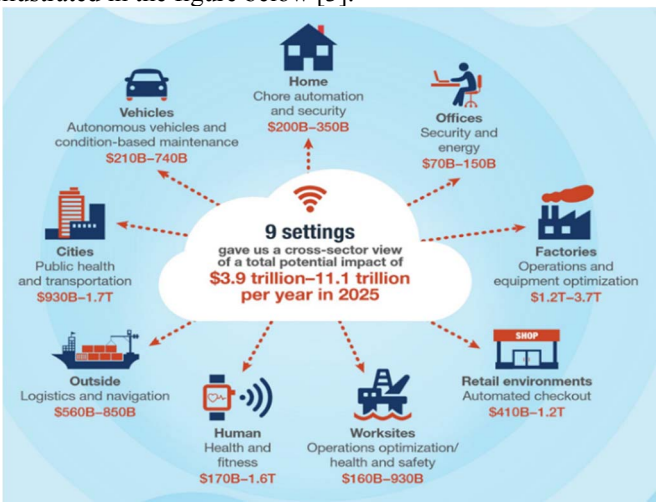

Fig. 1. Use cases of the IoT

The ultimate vision of the Internet of Things as proposed in the CASAGARAS consortium is a global infrastructure, which connects both virtual and physical generic objects [4]. However, many challenges ought to be addressed before this seedling

may mature to fruition. The main hurdle to the large-scale commercial use of the Internet of Things is the lack of standards and a mature business model [5]. As a result, there are several highly capable devices and systems that do not work together because (1) they are vendor-specific and (2) communicate over incompatible media and protocols.

Thus, a new term has even been coined: The Internet of Everything, which is actually the ultimate vision of the Internet of Things [6]. It has however been found out that situations in which two or more IoT systems must work together can account for about 40 percent of the total value that can be unlocked by the Internet of Things [3].

The AllJoyn framework was developed to enable devices to discover other devices on the same network and enables devices to communicate irrespective of the physical media: It is transport layer agnostic. In fact, with the AllJoyn framework, an appliance that uses Power Line Communication or Wi-Fi Direct would be able to communicate with another that uses Ethernet LAN once they are connected to an interconnecting node that has all these network interfaces.

Currently, AllJoyn itself only has support for communication media that use the IP stack i.e. Wi-Fi, Wi-Fi Direct, Ethernet LAN, and Power line communication.

With the introduction of the AllJoyn Device System Bridge by Microsoft, the protocol is now extensible for Bluetooth, Z-wave, ZigBee and other media.

Windows 10 introduced the Universal Windows Platform (UWP), which brings it into the Windows 10 unified core whereby the UWP provides a common app platform available on every device that runs Windows 10. Hence, apps that target the UWP can run on a variety of devices, support adaptive user interface, natural user input, one store, one development center, and cloud services [7]. The AllJoyn Device System Bridge is supported in the Universal Windows Platform.

This paper, therefore, seeks to address the interoperability problem in IoT as applied to home automation by using the primarily AllJoyn framework, and the Universal Windows Platform, coupled with RESTful web services, real-time communication and Apache Cordova based cross-platform mobile apps to develop a model IoT home automation system that can interoperate with several other systems.

The contributions of this paper are namely: (1) development of a system that facilitates communication and interoperability between home appliances made by various manufacturers and across several physical communication media. This is achieved with a hub-based network that uses the AllJoyn framework and the Universal Windows Platform, (2) extension of this hub-based network to enable remote access over the internet using a web service and a mobile application.

This issue of interoperability is very important because though the innovation is contributing well to the development of the Internet of Things, it is creating several “Internets of Things” and blocking the realization of the “Internet of Everything”.

Thus, if the issue is not addressed, consumers would have to purchase appliances from a specific vendor if compatibility is wanted. Manufacturers would have to design solutions from the ground up.

The potential benefits of implementing the system include allowing devices in a home to be controlled from a unified interface. This system would give consumers a wider range of products and product manufacturers to choose from when building a smart home. The system shall also enable autonomous communication between the devices to increase productivity. It will also serve as a platform on which energy saving and smart grids could be built since it will allow devices to be switched on or off from the hub. Thus, it could be integrated with a system, which turns devices on and off based upon occupancy and usage of facilities in homes and offices.

This system will afford consumers the flexibility of choosing from a wide range of appliances from a variety of manufacturers. The usage of wireless communication and device drivers in the plug and play makes the setup and configuration of smart homes very easy.

Section 1 presents the introduction on interoperability of heterogeneous appliances in home automation. Section 2 presents the literature review on related technologies on smart homes with their interoperability issues and challenges. It identifies the strengths and weaknesses of such systems and proposes the use of MAGNUM, a novel developed platform using AllJoyn Framework that addresses these issues and challenges. Section 3 provides the novelty of the system design architecture and methodology including the concepts, algorithms and flowcharts. It highlights the various modules with their design and procedures for implementation. Section 4 focuses on the actual implementations with experimental verification and corresponding integration of the developed modules. This section provides sufficient details on the testing and performance evaluation of MAGNUM, the proposed system for addressing interoperation of home appliances. Finally, section 5 provides conclusions on the systems design and methodology, and relevant recommendations for future enhancements.

II. Related research efforts

In [8], the researchers developed a Bluetooth-based wireless home automation and networking system consisting of a remote, mobile host controller and home appliances as client modules.

The researchers postulated the usefulness of using wireless communication, Bluetooth, for networking devices in home automation. They maintained that, it is a very good choice given that, it consumes less power, it is inexpensive, it has quite a long coverage area, and it is especially useful for the home environment where there exists hardly any infrastructure to interconnect intelligent appliances.

A protocol named Home Automation Protocol (HAP) was developed above the Bluetooth software stack to facilitate communication among the host and client modules. On device initialization, the HC broadcasts a Device Checking Packet over the Bluetooth connection, any DCs that are connected reply with a Device Detection Packet for each AD that is connected to it. The software on the HC had a GUI that enabled device registration/configuration, diagnostics and device status and control. These services were facilitated by the HAP.

The researchers considered and addressed several issues pertaining to home automation systems. For example, rather than, individual Bluetooth device controllers for each device, a shared Bluetooth device controller was used to reduce cost. Again, to make the system scalable, a custom protocol was designed, which supports dynamic addition and removal of devices. Diagnostic services were also included for troubleshooting communication issues with the various device controllers.

The use of wireless technology was prudent. However, Bluetooth can connect up to 8 devices only. Thus, this system will soon become impractical when smart appliances become

ubiquitous. Additionally, though class 1 Bluetooth devices can communicate over a 100m range most mobile devices use class 2 radios, which can only reach 10m. Thus, if the system is extended to a mobile device there could be issues of availability in big houses especially. Further, this system does not support nor make any provisions for other communication media.

In [9], the researchers present the design and implementation of a low cost and yet flexible and secure internet based home automation system, which features a PC component, a master node, and slave nodes.

The PC component, which forms the software aspect, is made up of a user interface, a web server serving an internet page, and a database connected to the user interface and the internet page. The hardware aspect comprises the master node and the slave nodes. The master node receives the control signals from the PC via an RS-232 link and relays the information to the slave nodes, via an RF link. Upon receiving control signals a slave node checks to see if the message was intended for it. If it was, then it executes the command on the appropriate device as specified by some bits in the message. Then it sends a reply in the form of the command that was sent to it plus the results, typically, the status of the device to the master node, which also relays it to the PC.

This work is, however, utterly simplistic. While it focuses on making the system low cost, the researchers simply designed the user interface and packet length based on the device category of the slave node. Yet, there was no mention of a standardization process of device categories and hence it is not clear as to whether the system is easily extensible for other device categories and how new categories may be added by third-party appliance manufacturers if it is extensible. Additionally, the control interface for the devices is achieved via the web server on the PC. This implies that the PC must have a public IP to facilitate control from remote locations and that will increase the cost of the system.

The authors in [10] made a distinction between two aspects of interoperability: the interoperability problem that happens between a user and a device, which was termed the user interoperability problem, and the interoperability problem that happens between two devices, which was termed the device interoperability problem. The authors proposed that the user interoperability problem can be solved through device discovery and device interaction within the home appliances ecosystem.

Insteon is a system that focuses on creating interoperability and reduction of noise and interference in 2.4GHz space by using the 915MHz frequency. It was however only interoperable with Home Kit and use a combination of 915MHz radio frequency and Powerline Communication as the physical layer. Control4 is a system that is similar in many ways to the proposed system. However, it is proprietary and does not use generic drivers. Thus, there is still some level of vendor-specificity.

OpenHAB is a system that focuses on creating interoperability by providing an open source framework that enables innovation and reuse of the bindings developed. It requires the developer to program the connectivity between the devices. To use the concept of plug-and-play and device drivers and the AllJoyn framework to develop a hub-based system for smart homes that will allow devices with various networking capabilities and devices from various vendors to be connected effectively.

This paper focuses on addressing the interoperability problem of incompatibility between the different communication media used and vendor-specificity in the Internet of Things as applied to home automation. It encompasses several subject areas in computer engineering used in developing Magnum. Magnum is a suite of hardware and software components that provides users a way of controlling appliances in their homes while making for interoperability of the appliances and the control application.

III. SYSTEM DESIGN AND DEVELOPMENT

A. System Architecture

Magnum was designed as a hub-based network that uses the AllJoyn framework for communication between a control hub and the appliances in the home. As the AllJoyn framework was limited to a local area network, the system was extended with a web service, real-time communication, and a mobile application to enable control of appliances from remote locations (external networks) over the internet.

Thus, in the Magnum system, there are appliances that connect over the various physical networks supported by the AllJoyn framework through a networking infrastructure like a router to a node that runs a control panel application.

The control panel application is essentially an AllJoyn application that uses the *About service* of AllJoyn to discover and interact with the connected appliances. This forms the LAN part of the system and is meant to be entirely in the user's home. For the purposes of remote access, a web service on the internet acts as a gateway agent for the control panel application while a mobile application gives users a graphical user interface for that matter.

A plug and play device is one with a specification that enables the discovery and automatic configuration of a hardware component in a system without the need for user intervention [11]. This is achieved using device drivers. The control panel application was modeled based on the plug and play architecture in that, among other things, it uses data advertised by the devices discovered on the AllJoyn network to dynamically generate intuitive controls for the user to interact with the system.

The system developed in the implementation used a Wi-Fi network for the LAN. The system developed is illustrated in Fig. 2

For the sake modularization and scalability, the Magnum system was partitioned into 4 subsystems that follow logically from the system architecture

1. Test appliance – **Magnum Hardware:** This consists of a fan controlled by Pulse Width Modulation (PWM) developed for testing and evaluation of developed protocols.
2. Control panel application – **Magnum Control Panel:** This is standard interface for the developed software controls of the associated hardware components.
3. Web service – **Magnum Web:** This is the interactive web application developed that enables the various services to operate efficiently across different protocols.
4. Mobile application – **Magnum Mobile:** This is the mobile application developed for the control of appliances with different communication protocols across the internet.

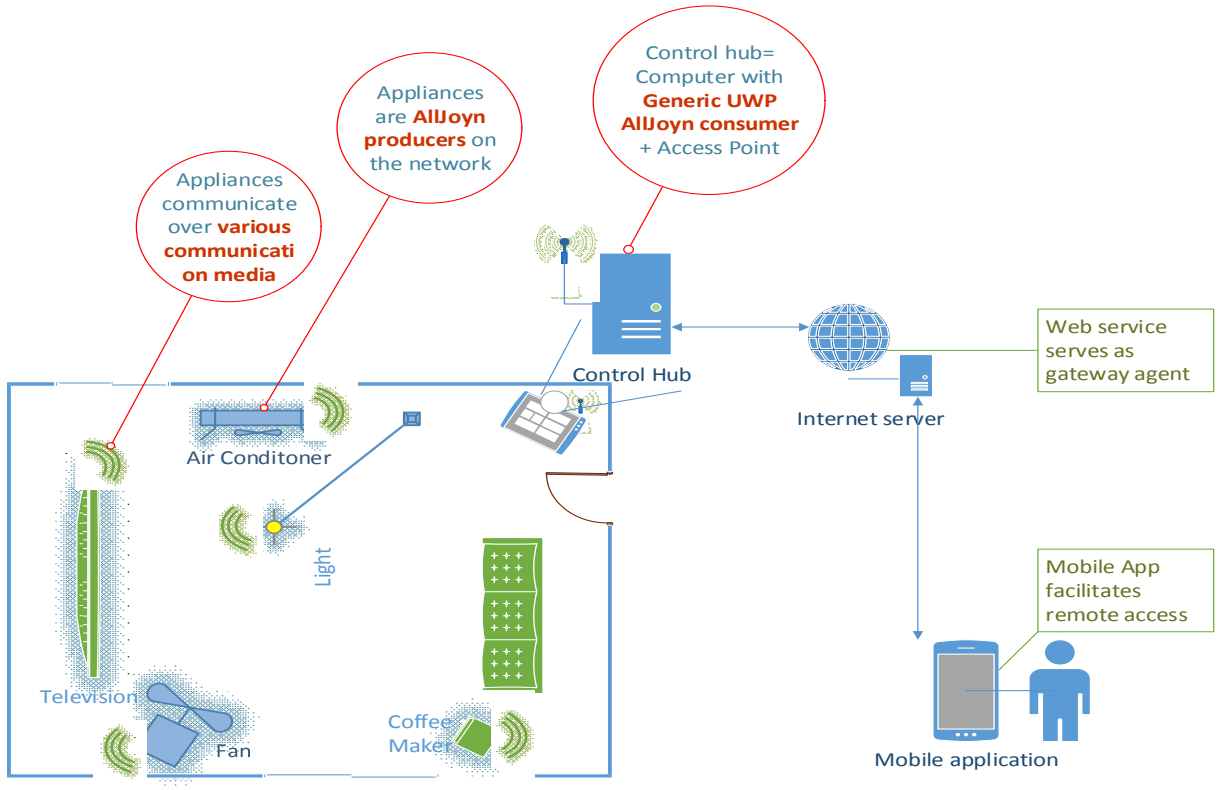

Fig. 2. System Architecture for Magnum

Program flow

The software flow of the mobile application is embodied in the sequence diagram in Fig. 3.

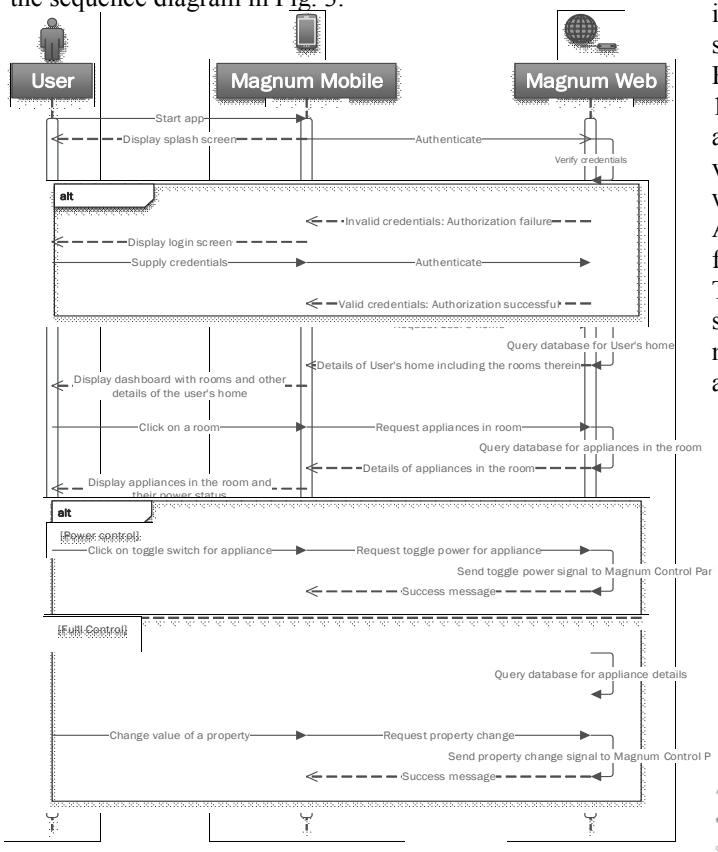

Fig. 3. Sequence diagram for mobile application

B. System Simulation

Before finalizing the design of the system and proceeding to implementation the feasibility of the designs was assessed by simulation.

For the hardware, a Proteus simulation was done with a PIC 16F877A microcontroller. The system was programmed to be able to be able to turn a motor on, turn it off, regulate the speed via Pulse Width Modulation (PWM) and set a timeout after which the motor was supposed to go off. To simulate a real-life AC fan, the motor's power supply was separate and isolated from the controller by a relay.

The diagram in Fig. 4 illustrates the simulation that was run. It shows the system with the power turned on, the speed of the motor set to low and a countdown time (at 1 minute 49 seconds) after which the motor is powered off.

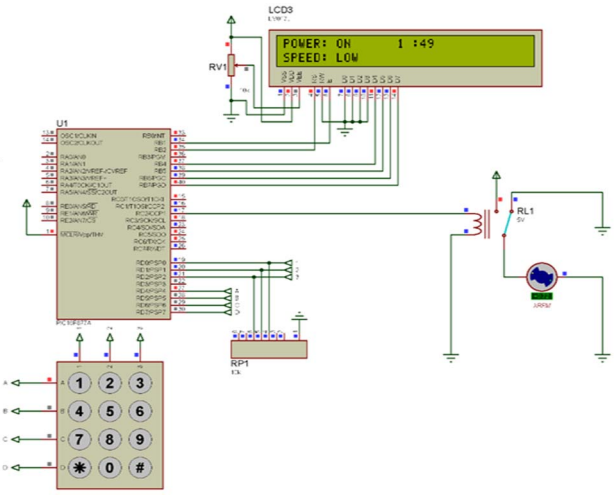

Fig. 4. Fan speed regulator simulation with PIC microcontroller and relay

C. Development Tools

Table 1 gives a summary of the development tools used in the Magnum system.

TABLE 1: SUMMARY OF DEVELOPMENT TOOLS (HARDWARE AND SOFTWARE)

Magnum Module	Component	Tool used
Hardware	Operating System	Windows 10 IoT Core 10.
	Board	Raspberry Pi 2 model B
Control panel	SDK	Tools for Universal App Development 1.3
Web	SDK	ASP.NET WebAPI 2.0
	Web Server	IIS Express for debugging and IIS 10.0 for release
	Database Server	Microsoft SQL Server LocalDB for debugging and Microsoft SQL Server 2014 Express for release.
	Real-time communication	ASP.NET SignalR
Mobile	SDK	Apache Cordova 6
	UI framework	Ionic

Following from the architecture of the Magnum system and the specifications, the implementation was developed broadly into the following:

1. Setting up the development computer and the Raspberry Pi
2. Developing hardware prototypes
3. Implementing user interfaces for the Magnum Control Panel
4. Implementing the local database for the Magnum Control Panel
5. Implementing the AllJoyn generic consumer feature of the Magnum Control Panel.
6. Implementing the user interfaces for the Mobile Application
7. Implementing the RESTful Web Service
8. System integration

IV. SYSTEM IMPLEMENTATION AND TESTING

A. System Implementation Process

As the prototype system was working and stable based on actual hardware implementation and simulation results, a more realistic hardware system that works with mains AC was designed and built. The exploded view of the developed magnum hardware that was built is illustrated in Fig. 5 below.

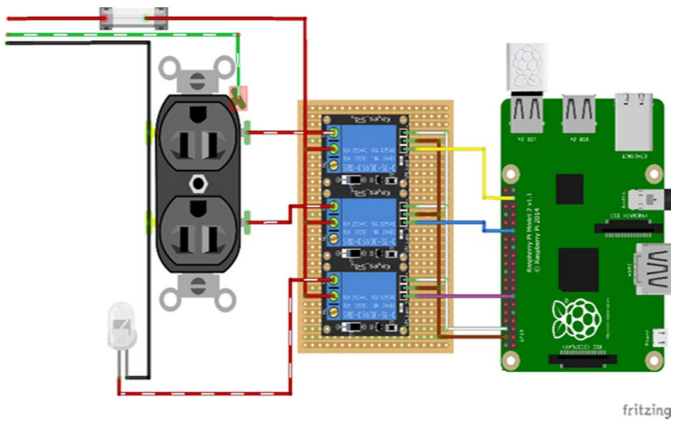

Fig. 5. Exploded view of Magnum Hardware.

On the extreme left are cables that go through a 3-pin plug to the mains power. To the right of them are the wall socket and a lamp. In the middle are relays activated by 5V power and controlled by 5V or 3.3V signal inputs and connected in the normally open mode. On the right side is the Raspberry Pi with a USB Wi-Fi adapter connected to it.

Leveraging the multitasking ability of the Raspberry Pi, three copies of the same program for the prototype LED brightness regulator, each handling a different GPIO pin, were deployed onto the Raspberry Pi.

1) Control panel user interface

Fig 6 shows the entity relationship diagram of the Control panel local database.

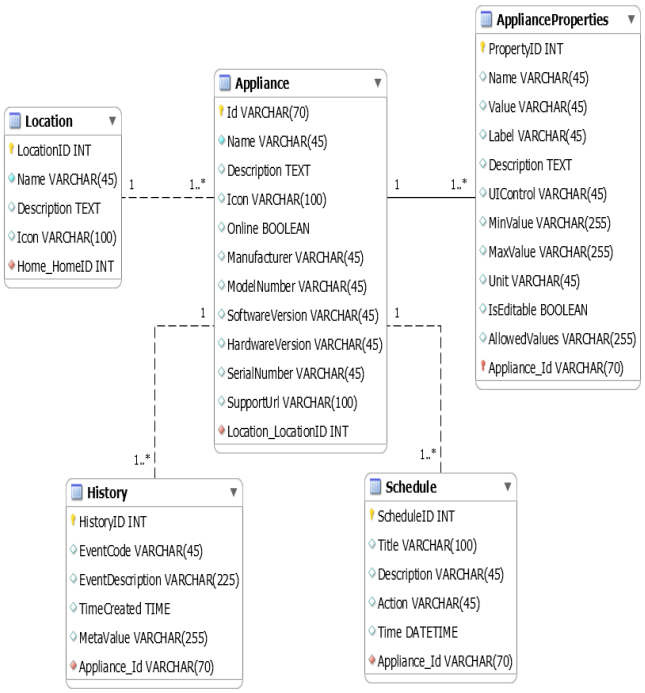

Fig. 6. Entity Relationship diagram of Control Panel local database

2) Control panel AllJoyn generic consumer

Throughout all the examples and tutorials provided by Microsoft on AllJoyn in UWP apps, an XML Introspection definition for a device is used to generate a consumer/producer library such that the consumer is tailored to work specifically with that consumer. This paper, however, required the capability to work with any producer on the network. Thus, it needed to have a generic AllJoyn consumer. Such capabilities were, however, unsupported in the C# AllJoyn library and could only be achieved in the C/C++ version.

Since the IoT Explorer for AllJoyn was similar in functionality to the AllJoyn capabilities required in this system, a question was posted on Stackoverflow [12] and the issue tracking page on the Microsoft IoT GitHub [13] repository for some insights. The developers responded by saying that Microsoft had no intentions of releasing the source code of the then AllJoyn Explorer to public. They, however, released the underlying C/C++ AllJoyn project that does all the heavy lifting as the DeviceProviders project.

Being a Windows Runtime Component, though it was written in C++, the project could be referenced in a Universal Windows C#, C++ or JavaScript project. We included the DeviceProviders project as outlined by a tutorial provided by Morten Nielsen [14].

A class named *DeviceManager* was written, essentially as a wrapper, to provide some abstraction of the DeviceProviders project to our application.
In this class, an instance of the *AllJoynProvider* was enabled as the main handle for the Device providers project.
In the constructor of the DeviceManager class, event handlers were bound to the *ServiceJoined* and *ServiceLost* events raised by the AllJoynProvider when an AllJoyn application is discovered and joins the session and when an AllJoyn application leaves the session respectively. The flow is illustrated in Fig. 7 below.

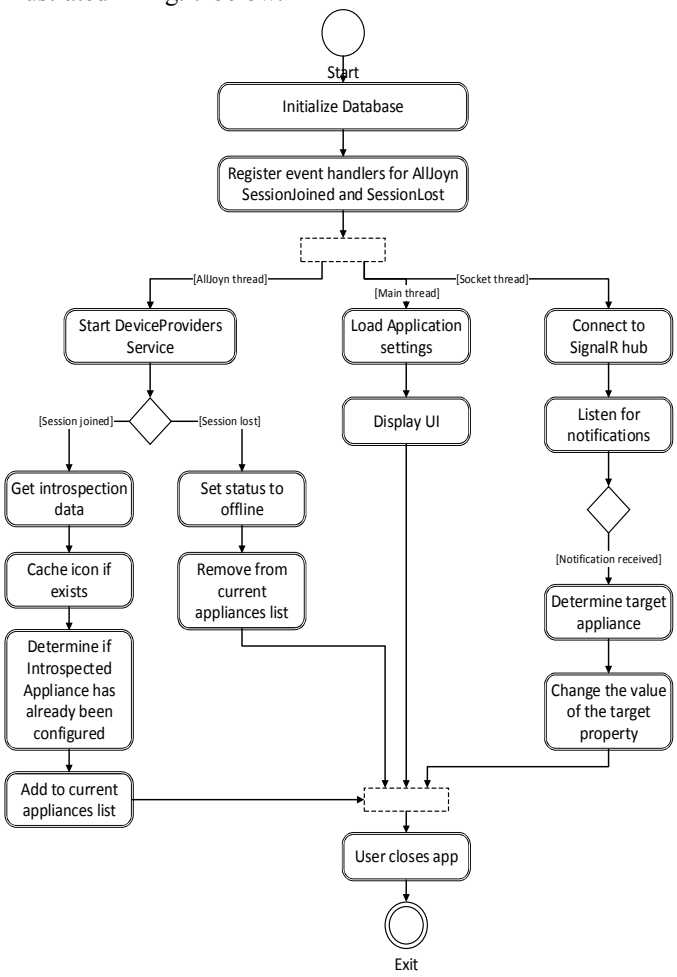

Fig. 7. Magnum Control Panel Application flow diagram

3) *RESTful Web Service*

As the control panel application was then able to discover appliances on the network and interact with them and the mobile application's user interface was ready, the next step was to fetch data from the control panel application and feed it to the mobile application.
This was achieved by writing a data-driven RESTful web service implemented in ASP.NET WebAPI. ASP.NET WebAPI follows the Model-Controller paradigm. In that regard, the models are representations of the data from the database (or another source), while the controllers contain the logic for handling requests from client applications and querying the database if need be. The logic in the controllers is written in methods called actions. Each action has a URI and an HTTP verb (e.g. GET, POST, PUT, DELETE) associated with it and may or may not return some data.

Fig 8 shows the entity relationship diagram for the RESTful Web Service database

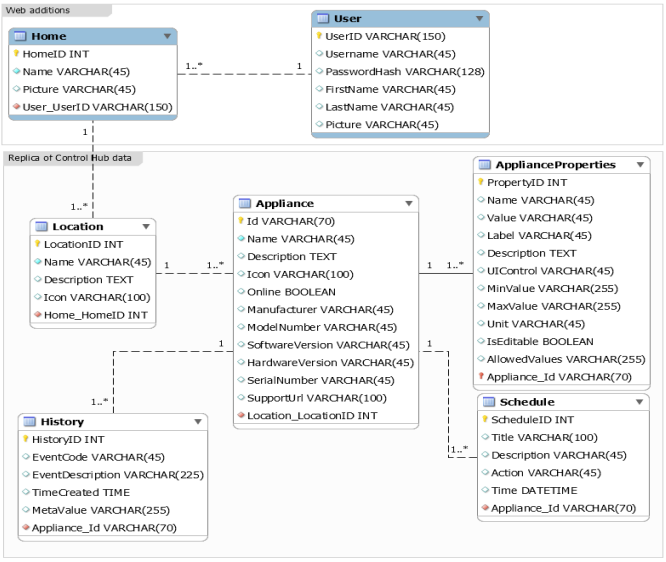

Fig. 8. Entity Relational model for Web service database

4) *Magnum Control Panel + Magnum Web*

To exchange data between Magnum Control Panel and Magnum Web the RESTful web service was the link. The ASP.NET WebAPI client library was included in the control panel project for that matter. The *MagnumAPIService* class was then written with methods containing codes for making requests the web service.
When a new location is created, a POST request is made to the web service with the location's details and the *HomeID* (which is a unique key that identifies the control panel application). When a new appliance is configured by the user and saved to the local database, the details together with the associated *AppliancesProperties* and *HomeID* and the *LocationID* (which identifies the location in which the appliance is) are sent via a POST request to the web service.

5) *Magnum Web + Magnum Mobile*

Data exchange between the Magnum Web and Magnum Mobile was also via the RESTful web service. The ngResource library was included in the mobile application project for that matter. A service, *MagnumAPIService* was written containing methods for accessing the web service.
When the application is launched an authorized GET request (VerifyLogin) is made to the web service to test whether the application is indeed authorized. If the request is successful then the home page just loads up, else the user is asked to supply his credentials or create an account. When the user supplies the required data a POST request is made with the data URL-encoded rather than JSON-encoded for compatibility with the default ASP.NET Identity provider. The Identity provider then creates the account for the user.

6) *System Security*

One of the biggest issues in the IoT is the security of user data. The use of the Hub-based network, the fact that AllJoyn works over a LAN and that AllJoyn has integrated security mechanisms, ensured the security of the home setting (i.e. between Magnum Control Panel and Magnum Hardware) as it is in the first place isolated from the internet.
The requirement of being able to remotely control appliances, however, brought the internet into the picture and this

introduces a potential point of intrusion and man-in-the-middle attacks. To keep the system secure, all Magnum Control Panels (homes) were preregistered on Magnum Web with a Globally Unique Identifier (GUID), which was embedded in the control panel as well. This GUID served as a secure token for the control panel application to access Magnum Web.

ASP.Net Identity was used to create and manage user accounts and verification on Magnum Web.

Another verification process was implemented here. The user is asked to go to the settings page on his/her Magnum Control Panel and click on the add user button. This initiates a GET request for another token that is generated randomly and Base-64 encoded. The token obtained is used to generate and display a QR code. The QR code is represented as a bit matrix of zeros and ones with zero representing white and one representing black. The mobile application is then used to scan and decode the QR code. The decoded character string is sent back to Magnum Web via a PUT request. Magnum Web validates this token, determines which home bears that token, and connects the user to that home. The application is then authorized to fetch the data pertaining to that home.

7) Real-time end-to-end communication

One characteristic of embedded systems is the fact that they have real-time constraints. Thus, in this system for instance, when a user clicks a button on the control panel application or the mobile application, the user would expect the change to take effect on the appliance in less than one second.

Sockets are persistent connections that a client maintains with a server and through which data can be sent reliably and with minimum latency. For ASP.NET the de facto real-time messaging system is SignalR [15].

Thus, the SignalR library was included in Magnum Control Panel, Magnum Web and Magnum Mobile with Magnum Web acting as the SignalR server and the others as SignalR clients.

B. Testing and Results

Testing was done in tandem with the implementation of the system to ensure the system produced the results expected.

During the development of the individual modules unit testing was often carried out after every major addition. For Magnum Control Panel and Magnum Mobile this often involved checking to see if appropriate events were raised on interaction with the application, ensuring the navigation was working properly and that the required data was passed between the pages and states respectively and binding the views with test data to determine if they displayed correctly and nicely. For the Magnum Hardware developed, the IoT Explorer for AllJoyn provided by Microsoft was used to determine if the details were advertised correctly and the change of values were effective.

In the integration of the modules too, system testing was carried. Besides the strategies used in the unit testing, inspection of the debugging windows in Visual Studio to ascertain if the data being sent and received was appropriate was employed. The following table summarizes the system tests that were carried out.

TABLE 2: SYSTEM TESTS AND RESULTS

Test	Expected Result	Remarks
Launch Magnum control panel application.	Display splash screen, initialize the database, start AllJoynProvider and display the main page.	Successful.
Start Magnum hardware application while it is connected to a LAN with the hub and Magnum control panel is running on the hub.	Raise ServiceJoined event in Magnum control panel and use introspection data to add the discovered Magnum Hardware to the New Appliances or Current Appliances list.	Successful.
Stop the Magnum hardware application or power off Raspberry pi while Magnum Control Panel is still running.	Raise ServiceLeft event in Magnum Control Panel and remove the leaver from the New Appliances or Current Appliances list.	Failed some of the time. The cause is a bug in DeviceProviders project. The workaround was to restart the applications.
Create a new location in Magnum Control Panel.	The location should be added to the local database and also posted to and saved on Magnum Web/	Successful.
Add a new appliance in Magnum Control Panel.	The appliance with its properties should be added to the local database and also posted to and saved on Magnum Web.	Successful.
Start the Magnum Mobile application.	Verify OAuth token with Magnum Web, while the splash screen is shown, display login state if the token is invalid.	Successful.
Create a new Magnum Account.	Save the user account details on Magnum Web and issue OAuth token.	Successful.
Click add new user icon on Magnum Control Panel.	Request access token from Magnum Web, transform token into QR code and display it	Successful.
Scan QR code generated on Magnum Control Panel with Magnum Mobile.	Decode the QR code, forward it to Magnum Web, determine the home that bears the token and link user account to home the home is found.	Successful.
Make a request from Magnum Mobile to RESTful web service on Magnum Web.	Verify the OAuth token. If it passes, fetch the required data from the database, serialize as JSON and return to Magnum Mobile.	Successful.
Interact with a control for an	Effect change on Magnum Hardware	Successful to a large extent.

ApplianceProperty on Magnum Control Panel.	and send an update via SignalR to Magnum Web for relaying to all other clients.	Updates received by Magnum Mobile were however not displayed on the Appliance view.
Interact with a control for the power switch of an Appliance in Magnum Mobile.	Send change via SignalR to Magnum Web for relaying to Magnum Control Panel. On receipt of update in Magnum Control Panel, the change was to be effected on Magnum Hardware and on success, the update sent back to connected clients via SignalR.	Successful.

V. CONCLUSIONS AND RECOMMENDATIONS

The landscape for new technological developments warrants the need for individual manufacturers to develop proprietary solutions unique to their ecosystem. This leads to several disparate systems which would otherwise provide more utility if they could interoperate.

For this reason, standards are developed to ensure compatibility of systems and provision of certain bare minimum functionality. However, IoT has reached that stage where it would benefit a great deal from standardization of protocols. As such common models and frameworks that provide interoperability while giving developers freedom of customization and innovativeness are developed and standardized.

The AllJoyn framework seeks to solve this problem by defining a peer-to-peer network, based on the distributed bus architecture, through which IoT applications can communicate via remote procedure calls. Furthermore, it includes the Device System Bridge, which allows on-AllJoyn appliances to work on an AllJoyn network.

The Magnum system developed in this paper deals with the interoperability problem in IoT as applied to home automation. It features AllJoyn enabled Universal Windows Applications in Magnum Hardware i.e. appliances and in Magnum Control Panel i.e. the hub of the AllJoyn network. Magnum also extends the LAN based AllJoyn framework by providing a web service and a mobile application to support remote access.

Given the development of the system in a modular pattern, the use of extensible open source frameworks and the AllJoyn framework the system can easily be extended. The use of the Universal Windows Platform also enables the Control panel to be run on a variety of form factors given all the way from the Raspberry Pi, through Microsoft Phones, through the Xbox One to regular computers. While the web service and real-time communication enable ubiquitous access, the mobile application was also platform independent and could hence be run on Android, Windows Phone, iOS, and Blackberry.

The Magnum system thus serves as a model with the potential of becoming an industry standard in IoT as applied to home automation.

Hence devices that will be made compliant with the system would be able to cross the barriers of physical media, protocols, and vendor-specificity in communicating among each other.

ACKNOWLEDGMENT

The authors would like to acknowledge the technical and nurturing environment provided by the Department of Computer Engineering, University of Ghana. The grant award from the Carnegie Corporation of New York for supporting the University of Ghana Next Generation of African Academics Project is hereby acknowledged. Finally, colleagues in the Department of Computer Engineering provided several helpful suggestions on an earlier draft of this paper.

REFERENCES

[1] J. Pellet, "What CEOs need to know to compete in the next great wave of value creation in manufacturing.," *Chief Executive Magazine*, no. March/April 2015, Greenwich, pp. 59–61, 2015.

[2] D. Bandyopadhyay and J. Sen, "Internet of things: Applications and challenges in technology and standardization," *Wirel. Pers. Commun.*, 2011.

[3] J. Manyika *et al.*, "The Internet of Things: Mapping the value beyond the hype," Toronto, 2015.

[4] K. Govinda and R. A. K. Saravanaguru, "Review on IOT Technologies," *Int. J. Appl. Eng. Res.*, vol. 11, no. 4, pp. 2848–2853, 2016.

[5] X. Zhihao and Z. Yongfeng, "Internet of Things and its future," *Communicate - Huawei Publications*, pp. 23–26, Feb-2010.

[6] M. Villari, A. Celesti, M. Fazio, and A. Puliafito, "AllJoyn Lambda: An architecture for the management of smart environments in IoT," in *2014 International Conference on Smart Computing Workshops*, 2014, pp. 9–14.

[7] T. Whitney, "Guide to Universal Windows Platform (UWP) apps," *MSDN*, 2016. [Online]. Available: https://msdn.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide. [Accessed: 10-Jun-2016].

[8] N. Sriskanthan, F. Tan, and A. Karande, "Bluetooth based home automation system," *Microprocess. Microsyst.*, 2002.

[9] A. Z. Alkar and U. Buhur, "An Internet based wireless home automation system for multifunctional devices," *IEEE Trans. Consum. Electron.*, vol. 51, no. 4, pp. 1169–1174, 2005.

[10] G. Xiao, J. Guo, L. Da Xu, and Z. Gong, "User interoperability with heterogeneous IoT devices through transformation," *IEEE Trans. Ind. Informatics*, vol. 10, no. 2, 2014.

[11] R. Patel, "Plug-and-play," US5999989 A, 1999.

[12] K. Apeadu, "How to program AllJoyn Explorer," *Stack Overflow*, 2016. [Online]. Available: http://stackoverflow.com/questions/33718723/how-to-program-alljoyn-explorer. [Accessed: 27-Mar-2016].

[13] K. Apeadu, "AllJoyn Explorer source code," *Windows 10 IoT Core Samples Issues*, 2015. [Online]. Available: https://github.com/ms-iot/samples/issues/158. [Accessed: 18-Nov-2015].

[14] M. Nielsen, "Discovering and interacting with any AllJoyn device," *hackster.io*, 2016. [Online]. Available: https://www.hackster.io/dotMorten/discovering-and-interacting-with-any-alljoyn-device-0dbd86. [Accessed: 05-Apr-2016].

[15] J. M. Aguilar, *SignalR Programming in Microsoft ASP.*

NET. Redmond: Microsoft Press, 2014.