A Lightweight Framework of Home Automation Systems Based on the IFTTT Model

Supachai Vorapojpisut*
Thammasat University, 99 Moo 18, Paholyothin Road, Pathumthani, Thailand.

* Corresponding author. Tel.: +66-25643001; email: vsupacha@engr.tu.ac.th
Manuscript submitted May 5, 2015; accepted August 29, 2015.
doi: 10.17706/jsw.10.12.1343-1350

Abstract: This paper presents the design and prototype of a home automation framework based on concepts underlying a popular web service called IFTTT. Our study has revealed several interesting concepts which can be expressed as role/application/security models. Adapting these models for the domain of home automation systems, a multi-tier software framework is proposed for automating home activities according to the key concept "IF-This-Then-That". Our framework differs from existing frameworks according to the integration of a simplified core script and a collection of trigger/action scripts. The prototype phase has been carried on the Google App Engine platform, a Raspberry Pi board, and Arduino boards as I/O units.

Key words: Home automation, software framework, open source, raspberry Pi.

1. Introduction

Home automation systems integrate hardware/software systems in order to automate activities within residential buildings. Recent technological advances in networking hardware and software [1] have expanded the scope of home automation applications to smart grid [2] and healthcare [3]. Compared to other types of buildings, software for home automation systems poses several challenges especially the trade-off between functionality and usability and also the trade-off between connectivity and privacy. Consequently, there is still no dominant technology in the field of home automation systems.

There are several academic works aim to implement advance functions for home automation systems, e.g. voice interface [4], remote operations [5], and cloud computing [6]. Nevertheless these works are constrained in the sense of scalability and portability with respect to the scope of applicable hardware platforms. On the other hand, there are several technical books [7], [8] that explain how to build DIY home automation systems using off-the-shelf components and open-source software. As opposed to academic works, their functionality and technical levels are limited by features provide by their underlying software platforms. While almost of existing commercial frameworks provide high-level of abstraction to achieve functionality and usability. These frameworks suffer from the heterogeneity of devices in the market.

This work attempts to propose a software framework that promotes flexibility and maintainability over other quality metrics. The main concept is to formulate a minimal automaton that integrates multiple code segments that a user chooses from a repository. That is, a sequence of automated activities is programmed by its home owner by selecting and configuring code from available pool. As a success story from the domain of web services, this work investigates a web service called IFTTT which offers automated integration between major web services and social networks. The conceptual ideas are used for the design

and prototype of our framework of home automation systems.

This papar is organized into four sections. Section 2 discusses and identifies key characteristics from the IFTTT service. Then our approach is explained as multi-tier software architecture in Section 3. Finally Section 4 concludes benefits and concerns of our framework.

2. IFTTT Model

This work considers the approach of a web-based service called IFTTT which offers a good trade-off between functionality and usability. IFTTT let end users to create, customize and enable chains of conditional statements, called *recipes* (see Fig. 1), which are triggered based on changes to existing web services, e.g., Facebook, Twitter, and Youtube. IFTTT also provides its mobile applications as channels to smartphone features and installed applications. Three key characteristics, namely the role model, the application model, the security model, can be identified as major differences from other web services or mobile applications.

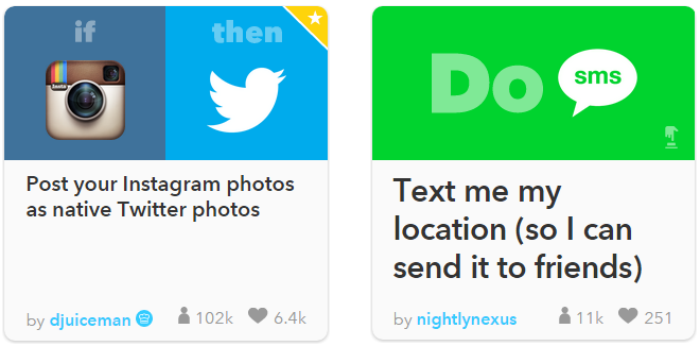

Fig. 1. Example of IFTTT recipes based on web services and smartphone.

2.1. Role Model

Within the IFTTT environment, there are three groups of stakeholders, namely IFTTT itself, channel owners, and end users. Each stakeholder will have their own objective, scope of access, functions, and benefits. The use case diagram (see Fig.2) represents the relationship among stakeholders.

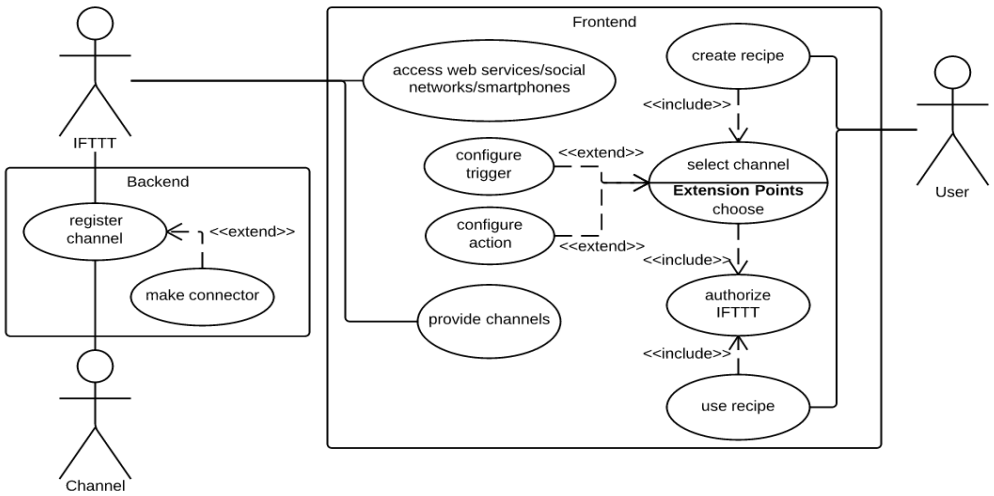

Fig. 2. Role model within the IFTTT environment.

The role of IFTTT itself is to provide channels to end users and to access web services/social networks/smartphone on the behalf of each user. Each channel represents a collection of available

data/operations from its corresponding service. The role of channel owners is to make connectors to bridge with IFTTT backend services. Companies that would like to promote their services/products as IFTTT channels must submit their APIs for the approval. There are two main functions for end users: recipe creation and recipe usage. Each IFTTT recipe is created by joining between a trigger and an action resulting in the information flow from one service to another.

By separating roles of IFTTT/channels/users, good usability is achieved for almost end users due to its simple workflow. Even UI is simple, functionality is compensated by the availability of channels and corresponding triggers/actions. These affect a reasonable trade-off between functionality and usability. This role model should also serve reasonably well for home automation systems in which homeowners prefer customizable automated activities and easy-to-use UI.

2.2. Application Model

There are two different perspectives for the application model of IFTTT; namely developer perspective (channel) and user perspective (recipe). For the provision of channels, IFTTT manages basic channels for popular services, while third-party channel owners are responsible for their web APIs (authenticate/ authorize/access). The scope of development is to abstract and wrap API-related operations as a collection of triggers/actions in each channel. For end users, automated activities can be customized with respect to the creation or selection of recipes. The procedure to create a recipe is outlined as follows:

1) Register and login to IFTTT.
2) Create a trigger and an action:
   a) Select a channel from the provided list
   b) Activate by authorize IFTTT with the service provider underlying such channel.
   c) Select a potential trigger/action from the available list.
   d) Fill needed information for the selected trigger/action.
3) Fill description and activate recipe.
   e) Share recipe if needed.

User may choose to share the recipe to public. Then other users may choose from these shared recipes.

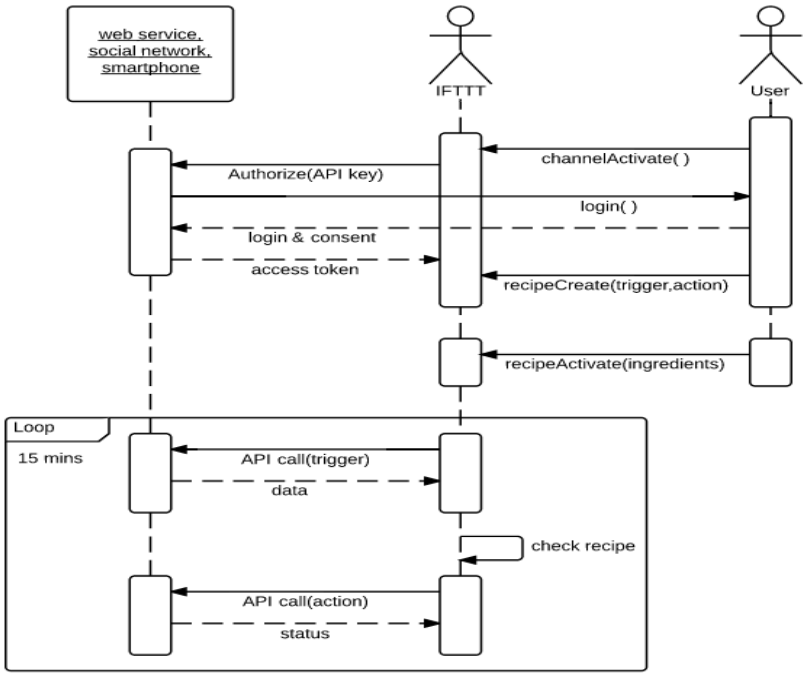

Fig. 3. Application model within the IFTTT environment.

A scenario (see Fig. 3) that the end user programs an automated activity is to route from one abstracted web API (trigger) to another abstracted web API (action). Then IFTTT will handle this recipe by accessing two corresponding services without any user interaction. In other words, each recipe is a program which is executed based on data from those external entities (web service/social network/smartphone).

2.3. Security Model

The security concerns of the IFTTT environment (see Fig. 4) can be classified into two folds: the trustworthiness of channels and the privacy of users. There are two separate review processes to handle these security concerns, namely the channel review process and the channel activation process.

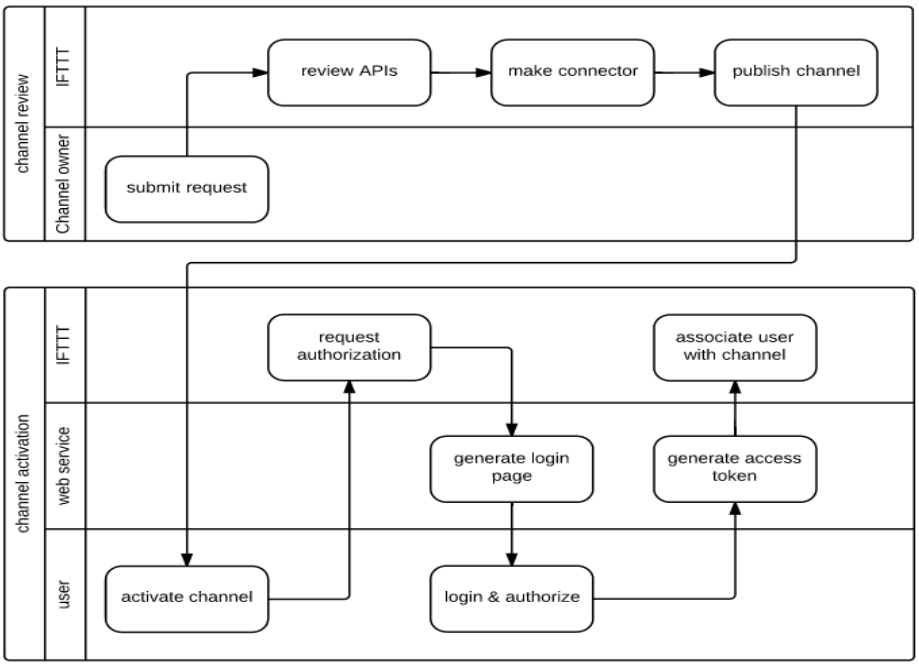

Fig. 4. Security model within the IFTTT environment.

The list of channels and their corresponding triggers/actions are regulated by the IFTTT team. Since IFTTT does not provide any public API to developers, each company must submit the request to establish a channel with its corresponding APIs for the review. This is similar to the review process used by online software stores. The channel review process has posed some constraints for a class of connected devices since these devices cannot act as a connector for IFTTT channel. Therefore only connected devices associated to an official channel (e.g., Nest, Philips) can be used inside the IFTTT environment.

The channel activation process is performed by each user for each channel during the creation or selection of recipes. For web service channels, the channel activation is the authentication/authorization for its corresponding web service. This step will be done by either the 3-legged authorization (e.g., OAuth) or user credential. For smartphone channels, at least one of the official DO applications must be installed into user smartphone. That is, the user privacy will be responsible by user themselves.

3. Home Automation Framework

3.1. Design Concepts

The objective of this work is to develop a framework that enables the integration of DIY components for automating home activities. To achieve both functionality and usability, the following modifications to the IFFFT model are proposed.

1) There are three groups of stakeholders: homeowners, open-source developers, and web services/social networks.
2) The platform consists of three computer systems including a programmable gateway, a public cloud, and a user terminal (e.g., computer, smartphone).
3) Channels are provided by scripts in the programmable gateway and web services.
4) Both channel review and activation processes are handled by the homeowners themselves.

These modifications correspond to the nature of DIY home automation systems in which some scenarios may conflict with mandatory requirements of IFTTT or similar Internet of Things platforms. For example, homeowners may build their systems by purchasing and assembling off-the-shelf parts. Internet connection may not exist or not reliable. Almost use cases of home automation systems prefer restricted group access and unbalanced connectivity (*at-home* and *anywhere*) features.

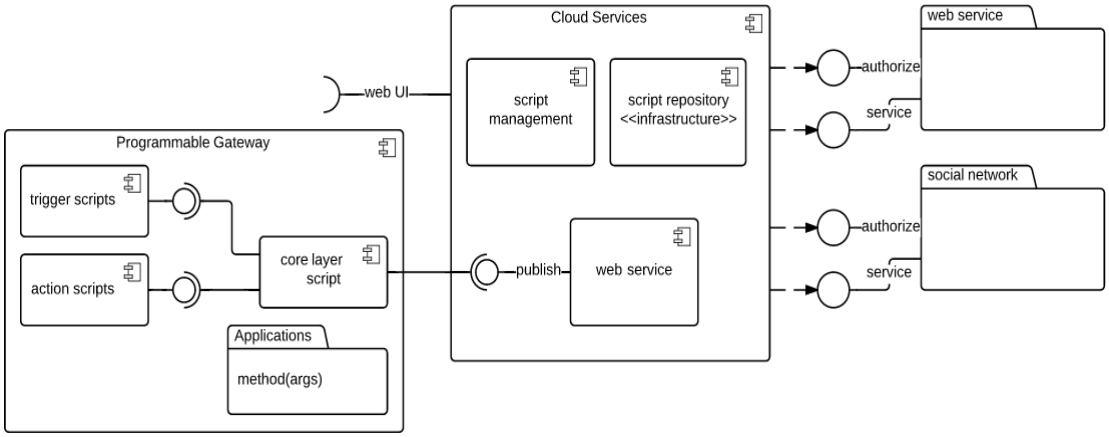

Fig. 5. Software architecture of our proposed framework.

The software system on the public cloud (see Fig. 5) acts as a code repository that allows developers to upload their code and provides custom scripts and configuration files for each user. Another service is to serve as a broker for the request of access tokens from web services/social networks that enforce the OAuth authorization flow. The software system on the programmable gateway will handle automated jobs according to the concept of "IF-This-Then-That" by running code expressed in configuration files. The user terminal will only be used for the authorization of our cloud services with web services/social networks.

3.2. Cloud Services

Our cloud services have been prototyped on the Google App Engine (GAE) platform under the URL "http://rpi-box.appspot.com". Web applications have been developed by utilizing the webapp2 framework, the ndb datastore, the jinja2 templating library, and the jQuery Mobile library. The service architecture on the cloud (see Fig. 6) can be separated into two groups: web applications for managing script/configuration and web services for bridging between programmable gateways and other services.

3.3. Programmable Gateway

There are three groups of software to be installed in the programmable gateway: a core script, software applications providing hardware/software features, and a collection of trigger/action scripts. Our proposed software architecture differs from existing open-source home automation frameworks in three aspects:

1) The core script is very small and invoked periodically as cron jobs.
2) A chain of automated "IF-This-Then-That" activities is expressed in the form of a JSON configuration file which is generated from the recipe configuration page.

3) Required trigger/action scripts and software applications as shown in the recipe configuration page have to be manually installed in the gateway.

As a consequence, homeowners are required to have at least basic skills of installation and configuration of target hardware platforms. Even inconvenience, the necessity of such manual installation/configuration is to encourage homeowners to review software components inside their home automation systems. This decision corresponds to one of merits of open-source software that allows source code to be reviewed. Note that this demand is not so troublesome since trigger/action scripts should be simple and short with respect to their single-purpose and straightforward operations.

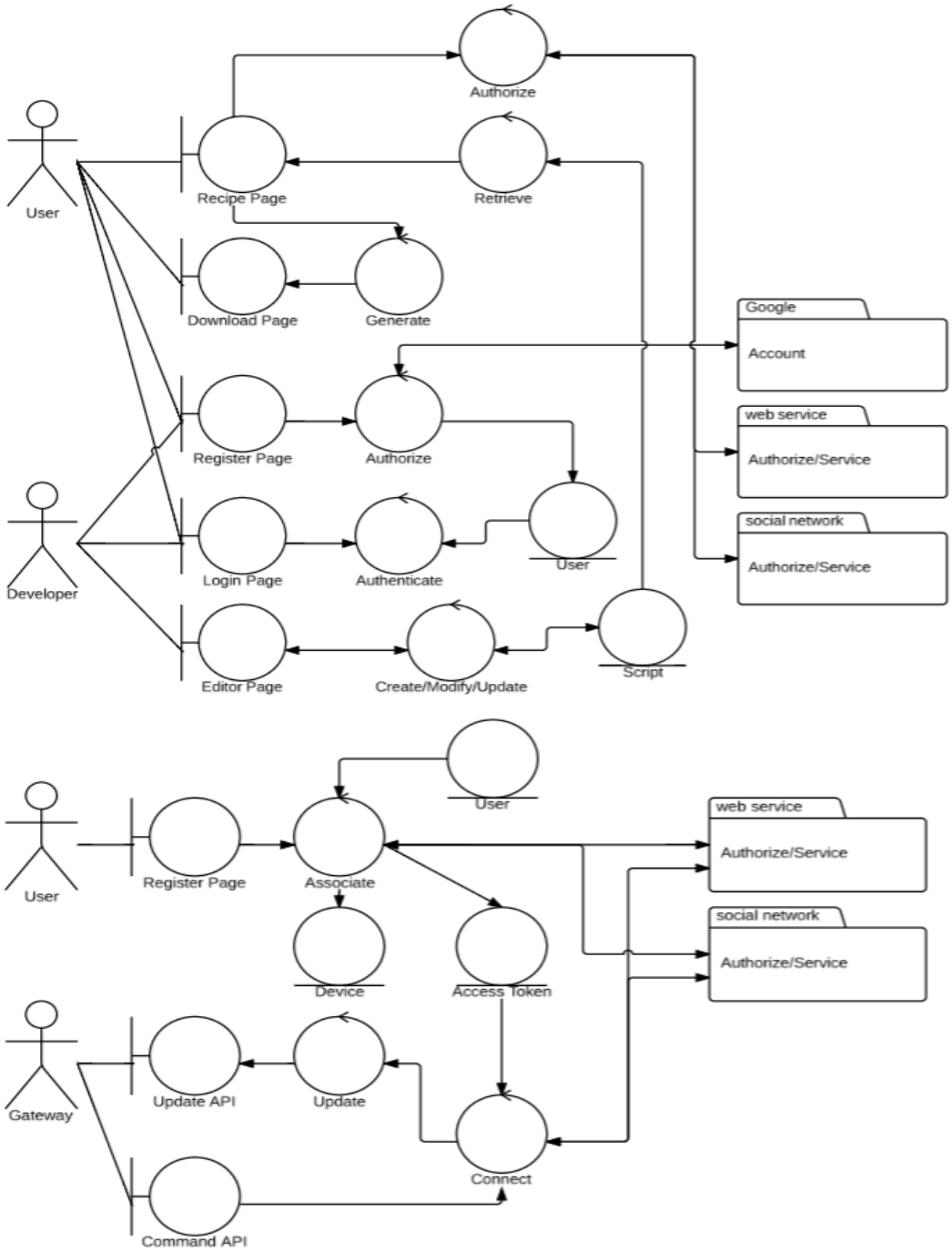

Fig. 6. Software architecture of user-oriented and device-oriented cloud services.

The Raspberry Pi model B board was chosen for prototyping the programmable gateway due to its

benefits for home automation applications, e.g., HDMI port for TV display, USB ports for I/O boards, and desktop Linux distribution.

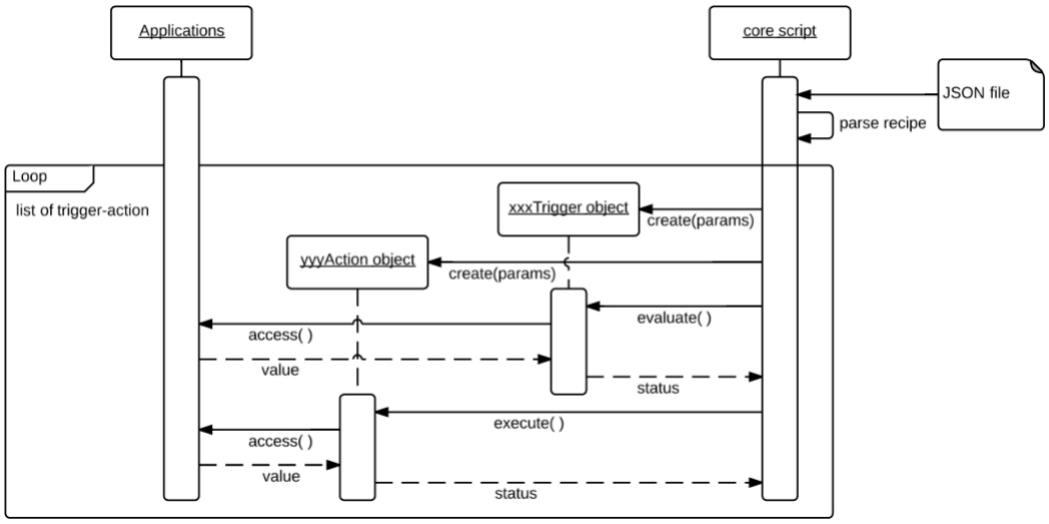

Fig. 7. Interaction within software stack.

Automated "IF-This-Then-That" activities (see Fig. 7) are realized by the core script to be scheduled as cron jobs. Three settings for the crontab file are recommended for running every minute, every hour, and every day. Based on the factory pattern, the pseudocode of the core script is outlined as follows.

1) Parse a given JSON configuration file.
2) Create *xxxTrigger* and *yyyAction* objects according to the class name as specified in "Trigger" and "Action" keys.
   a) Skip for all unknown class name.
3) Invoke the evaluate() method of each *xxxTrigger* object.
   b) If true, invoke the execute() method of corresponding *yyyAction* objects
4) If internet connection exists, connect to the cloud service to verify version of scripts.

The declaration of *xxxTriggerClass* and *yyyActionClass* is given in corresponding script files in the /root directory. The template of *xxxTriggerClass* and *yyyActionClass* is outlined as follows.

5) Extract key-value pairs from "params" key
6) Perform operations on parameters.
   a) If something wrong, return a dictionary of {"status":error code}.
7) Return a dictionary of {"status":success code}.

The above template represents the adapter pattern underlying trigger and action scripts which convert from key-value parameters into result and operations, respectively.

4. Conclusion

This paper has presented the development of a software framework for home automation systems based on modified concepts from the popular web service IFTTT. Our framework consists of two software systems: web applications as cloud services and software stack on home automation gateways. Our cloud services handle two major functions: repository of automated scripts and API bridges for web services/social networks. The software stack integrates the simplified core script, a collection of trigger/action scripts, and software applications providing features. The main advantage of our proposed framework is its simplified design while preserving extensibility and usability.

References

[1] Gomez, C., & Paradells, J. (2010). Wireless home automation networks: a survey of architectures and technologies. *IEEE Communications Magazine*, *48(6)*, 92-101.

[2] Pedrasa, M. A. A., Spooner, T. D., & MacGill, I. F. (2010). Coordinated scheduling of residential distributed energy resources to optimize smart home energy services. *IEEE Transactions on Smart Grid*, *1(2)*, 134-143.

[3] Harmo, P., Taipalus, T., Knuuttila, J., Vallet, J., & Halme, A. (2005). Needs and solutions — Home automation and service robots for the elderly and disabled. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3201-3206). Alberta, Canada.

[4] Zhu, J., Gao, X., Yang, Y., Li H., Ai, Z., & Cui, X. (2010). Developing a voice control system for ZigBee-based home automation networks. *Proceedings of the 2nd IEEE International Conference on Network Infrastructure and Digital Content* (pp. 737-741). Beijing, China.

[5] Elkamchouchi, H., & ElShafee, A. (2012). Design and prototype implementation of SMS based home automation system. *Proceedings of the IEEE IEEE International Conference on Electronics Design, Systems and Applications* (pp. 162-167). Kuala Lumpur, Malaysia.

[6] Ye, X., & Huang, J. (2011). A framework for cloud-based smart home. *Proceeding of International Conference on Computer Science and Network Technology* (pp. 894–897). Harbin, China.

[7] Riley, M. (2012). *Programming Your Home: Automate with Arduino, Android, and Your Computer*. The Pragmatic Programmers, LLC.

[8] Dennis, A. K. (2015). *Raspberry Pi Home Automation with Arduino* (2nd ed.). Packt Publishing.

Supachai Vorapojpisut received the B.Eng and M.Eng in the electrical engineering field from Chulalongkong University, Thailand in 1990 and 1995, respectively. Then he has continued his study at Tokyo Institute of Technology, Japan and received the D.Eng (control engineering) degree in 2000. At present, he works as an assistant professor in the Department of Electrical and Computer Engineering, Thammasat University. His research interests focus on embedded software development and wireless sensor network.