ATLANTIS PRESS

Advances in Engineering Research, volume 170

7th International Conference on Energy and Environmental Protection (ICEEP 2018)

Implementing a Demand Response Ready Smart Home Prototype on a System-on-Chip Platform

Hongmei Li[1, a], Yanli Chai[1,b]

[1] Jiangsu Normal University, Xuzhou, Jiangsu, China

[a]lhmjcn@163.com, [b] 513705657@qq.com

Keywords: Smart home energy management (SHEM), hardware design, demand response, real-time pricing, time-of-use tariff, load shifting, smart appliances, energy efficiency.

Abstract: To bridge the gap between undergraduate electrical engineering curriculum and the modernizing electric power grid, Smart Home Testbed projects are ideal for undergraduate student design and research programs. Observed the fact that Smart Home Testbed projects are small-scale and short-term, a three-layer design and its implementation on a Raspberry Pi system-on-chip (SoC) platform is proposed, aiming at fast prototyping and quick idea verification. Two Smart Home Energy Management (SHEM) prototypes for time-of-use and real time pricing demand response programs, respectively, are implemented. An example demonstration project, which consists of a hardware setup, Python-based SHEM modules and Web based user interface, is shown. The demonstration and case study verified the effectiveness of the proposed DR ready Smart Home Testbed on the SoC platform.

Introduction

With the increasing penetration level of renewable generations and electric vehicle loads, the electric power grid has seen significant changes in the recent years [1]. Undergraduate education in electric power system is also undergoing changes to incorporate the conventional power system analysis courses with Smart Grid related courses and designs. Smart Home Testbed is a set of hardware, software and physical appliances to emulate the power consumption and intelligent controls in a household. It is an experiment platform that could bridge the curriculum gap by facilitating undergraduate students in conducting Smart Grid related research and development.

One characteristic that distinguishes Smart Grid from the traditional grid is the more flexibility on the demand side. Demand response (DR) is a set of programs offered by utility companies or load serving entities (LSE) to directly or indirectly change the energy usage pattern [2]. DR is especially suitable for reducing peak hour load, so as to maintain system stability and keep generation costs low.

Although the nowadays DR resources are mainly large industrial or commercial load, residential DR also has a considerable potential. Meanwhile, the rapid development of information technology (IT) has also brought in opportunities for residential DR implementation, not only to save energy but also to improve household convenience. Small-sized, high-speed and low-cost System-on-Chip (SOC) solutions are becoming available for smart home applications [3]. The goals of a Smart Home Testbed are two-folds: reduce power consumption for the power grid as potential DR resources, and improve the experience of household appliances. The former one requires a comprehensive understanding of the present residential DR programs, while the latter one coordinates the household appliances with various functions.

Research has been carried out on smart home related architecture design, methods and algorithms. A Cloud based smart home architecture is proposed in [4] which focuses on service reliability of Web service and Peer-to-Peer technologies. A Simple Object Access Protocol based management system is proposed in [5] to solve the interoperation problem of various appliances. An energy conservation oriented smart home design is given in [6] which integrates different groups of smart appliances to deliver more value-added services for users' preferences. Moreover, digging into the function blocks into the architecture of SHEM, coordination and scheduling of the appliances are the basics of energy management [7]. A mixed-integer linear programming (MILP) based smart home appliance scheduling model is proposed in [8] to minimize household electricity cost. Predicted real-time

electricity price is considered in optimal residential load control method in [9], which also minimizes the appliance waiting time. Uncertainties from appliance operation time and renewable intermittency are taken into account in the scheduling method proposed in [7]. The intelligent aspect of smart home that tries to enhance the experience of home appliances and improve comfort level is also studied from both technology and sociology perspectives. Integration of network-based fire detection system is proposed in [10] for smart home automation. Wireless sensing network, biometric technology and voice control are proposed in [11] and [12]. Those reference provides insights for researchers, however, they lack one characteristic to be implemented in undergraduate education courses or projects.

A fact of the Smart Home projects is that they are generally small-scale and short-term, yet significant for undergraduate education and research. Therefore, the need for an architecture design that enables fast prototyping of functions and algorithms to verify sparks from undergraduate students and generate insights for the industry has not been well satisfied. This motivate fostered the idea of designing and implementing a layered Smart Home Testbed on a SoC platform with fast-prototyping scripting language support.

In this paper, a three-layer architecture design of a DR-ready smart home testbed is proposed in Section I to provide DR capability to the grid, conserve energy and improve household comfort level. More specific hardware and software designs are described to illustrate the proposed architecture in Section II. The proposed design is implemented on a Raspberry Pi 2 based SoC platform as a Undergraduate Student Project and the results are shown in Section III..

Framework Design

A. Design Principles The design of a Smart Home platform involves both hardware layout design and software architecture design, each has different requirements. On a higher level of abstraction, the designs should follow a couple of general principles:

1) Modularity. Functions modules to the platform core so that it can be easily added or removed.

2) Common data interface. All modules should follow the same predefined common data interface to send and receive messages. This is the basis of modularity and simplifies future enhancements.

In this section a layered architecture is presented, followed by two major module designs as examples.

B. Layered Architecture Design Fig. 1 shows the proposed three-layer architecture for Smart Home Testbed, which consists of a perception layer, an optimization layer and an actuation layer.

The fundamental basis of the Smart Home Testbed is the Smart Home Energy Management (SHEM) which coordinates the appliances and optimizes the scheduling for energy related objectives. The SHEM is the aggregation of all the function modules to provide DR capability and elevate comfort level. Although in the drawing more demand response related functions are shown, it's worth mentioning that intelligent home related functions that improves user's comfort level is of the same importance in SHEM.

On top of the SHEM is the perception layer that collects information from an LSE, ambient environment, connected appliances and the user. Specifically, information received from LSE is transferred through computer network, while the data from ambient environment are collected using sensors. Regardless of the different technique used for data collection, these modules share a common role in the Smart Home Testbed as *perceivers*. Any future modules that collects related information, for example a mobile APP based owner's location tracker, should be categorized into this layer. At the bottom of SHEM system is the actuation layer which receives control signals conducts the optimal control actions from the optimization layer. It generally consists of remote power outlet controller, configurable thermostats for HVAC systems and timers for other appliances.

Moreover, throughout the three layers there are user interfaces (UIs) that provide interactions with the Smart Home Testbed At the perception layer, the UI provides quick adjustments for operation

mode and appliances to achieve different objectives. In the actuation layer, the UI provides reports of well analyzed data and, more preferably, visualizations of energy consumptions and conservations.

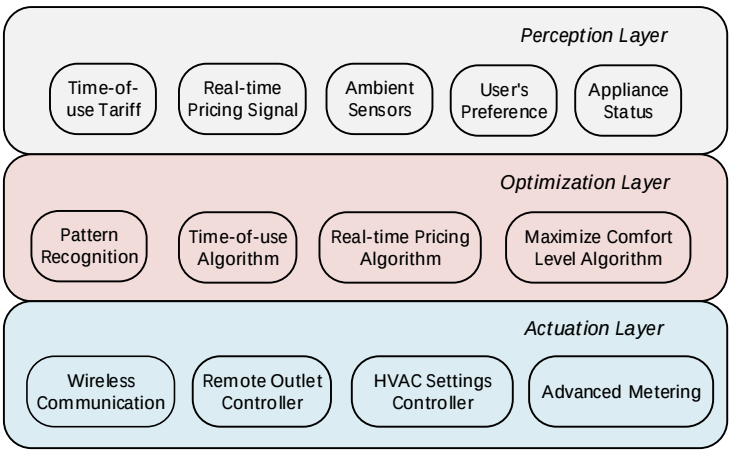

Fig. 1. Three-layer Design of Smart Home Testbed

C. Demand Response Capability Design The SHEM on the optimization layer contains a set of DR models that optimize the household appliance scheduling and consequently form the load profile. These models accept time-of-use (TOU) or real time price (RTP) signals from LSE as input, and, based on user's preferences and the availability of appliances, outputs the optimized states of appliances.

For a TOU based demand response program, users have the same step-wise electricity tariff across different days. Prices are generally higher at peak hours and lower at off-peak ones. A heuristic, delay-by-k hour algorithm for the TOU model is described in Table I. The basic idea is, for an appliance delayable by *e* hours being turned on at time *t*, only choose the hours with the lowest electricity price and turn it on; for a non-delayable appliance, turn it on immediately.

TABLE I. PROCEDURES OF DELAY-BY-K TOU ALGORITHM

Delay-by-k hour algorithm	
Inputs: 24-hour electricity price $P(t)$, appliance parameters	
1: **for** appliance h = 1 to n:	
2: if $U_h(t) - U_{h-1}(t) <= 1$:	# not turning on
3: $h = h+1$	
4: continue	
5: else:	# turning on
6: if h in {*delayables*}:	# is delayable
7: $k = \{P(k) \in \min_r\{P(t, t+1, \ldots, t+e+r-1)\}\}$	
8: $U_h(k) = 1$	# turn on when price low
9: else:	# otherwise turn on now
10: $U_h(t, t+1, \ldots, t+h-1) = 1$	

For an RTP model, the optimization is different due to the changing electricity price. However, with 5-minute-in-advance price signals, taking electric water heater (EWH) as an example, the appliance scheduling problem can be formulated as an optimization model to minimize the total electricity costs in (1) – (3).

$$\min \int_1^{24} RTP(t) \times P_{EWH} \times m(t) \tag{1}$$

$$\text{s.t.} \quad \frac{dX_T}{dt} = -a[X_T(t) - X_a(t)] - A(t)q(t) + P_{EWH} \times m(t) \tag{2}$$

$$T_{\min} \le X_T(t) \le T_{\max} \tag{3}$$

where RTP(t) is the last updated real time price at time t, P_{EWH} is the power consumption of EWH, $m(t)$ is a binary variable indicating the on or off state of EWH; X_T is the temperature of the water; $A(t)$ is the

rate of energy extraction when water is in demand at time t, $q(t)$ is the indicator of water in demand or not, a is the thermal resistance of tank walls. It's also worth pointing out that all optimizations are computed on a thin SoC client, hence a heuristic algorithm is employed to obtain a feasible solution instead of a global optimum.

Hardware and Software Implementation

A. Functional Requirements At the implementation level, the closed-loop hardware and software must meet the following functional requirements:

1) LSE to SHEM communication. Communication between the two are intended for electricity price signal broadcasting and, optionally, DR capacity feedback.

2) SHEM to actuation layer implementation. SHEM must be able to convert its control signals to the actual actions in the controlled home appliances, such as power outlets.

3) Appliance status storage. The on and off status of the household appliances must be stored somewhere in the SHEM securely and accessible with privilege granted.

4) User Interface. Energy saving mode and appliance status should be configurable in a friendly user interface.

5) SHEM optimization algorithms. This includes the DR ready models and algorithms discussed in Section III.C.

B. SHEM on Raspberry Pi The implementation of these functions largely depends on the hardware platform chosen for SHEM. For example, if an Arduino-compatible platform is chosen, those optimization models and algorithms must be written in Arduino's C-like language, which is more complicated than the prevailing scripting languages such as Python or Java. A comparison of some popular hardware platforms are given in Table. II.

TABLE II. COMPARISON OF POPULAR HARDWARE PLATFORMS

	I/O Pins	Operating System	USB Support	Display Ports
Arduino	Yes	No	No Native	No Native
Raspberry Pi	Yes	Yes	Yes	HDMI
TI DSP	Yes	No	No Native	Configurable

Among the features listed, operating system support carries the most weights because it allows the user to utilize the hardware resources more readily with interfaces provided by the system, as well as to take advantage of available software. The second important is the native support for USB, which enables plug and play of peripherals, such as wireless network dongle, sensors and audio devices.

Raspberry Pi offers support for Debian Linux operating system, which is ideal for SHEM function implementations. The lately released Raspberry Pi 2 model comes with a 900 MHz ARM Cortex-A7 CPU, 1 GB RAM, 4 USB 2.0 ports, full HDMI port and 40 GPIO pins. Running the customized Raspbian OS, it has native support for almost all hardware without having to any driver from scratch. The OS allows the developer to program in C language and meanwhile provides scripting language support like any other Linux system does. For the aforementioned reasons, Raspberry Pi is an ideal platform for Smart Home Testbed platform implementation.

Back to the required functions discussed in Section II.A, the following settings or configurations are done in preparation for the functional requirements. First, eight GPIO ports are connected to an external remote circuit board that controls the radio frequency wireless power outlets. Second, Python language is installed as a scripting language support. Third, an SQL database is setup to store the status of appliances, as well as the historical RTP data. Finally, an Apache HTTP server is set up for Web UI services. Note that the external remote circuit board mentioned is an actuator of the control signals. It is off the shelf and adopted for this project, as developing the radio-frequency communications is beyond the scope of the designated project.

C. Implementation Details Fig. 2 shows the implementation of the Smart Home Testbed platform on Raspberry Pi 2. To mimic the electricity market environment with RTP, the spot price in New York area, shown in Fig. 3, is retrieved from NY-ISO by LSE in order estimate RTP for the next 24 hours at a step size of 5 minutes. Then, the price data is broadcasted to SHEM in the households through the Internet.

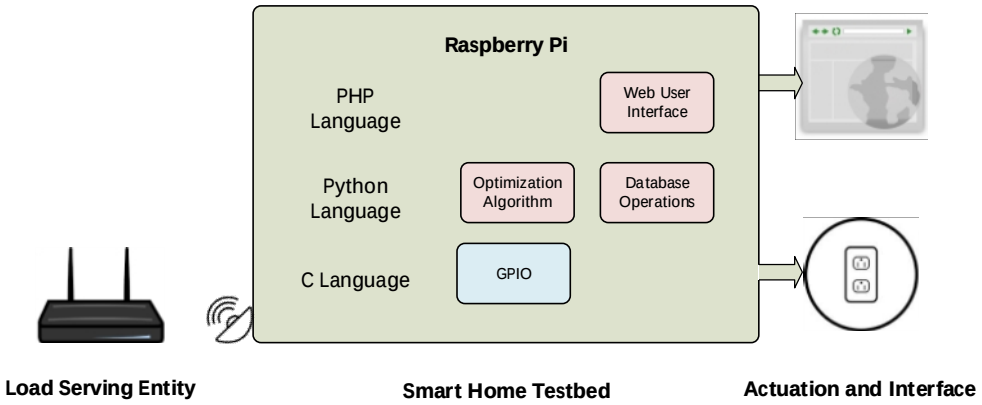

Fig. 2. Implementation of the Smart Home Testbed platform

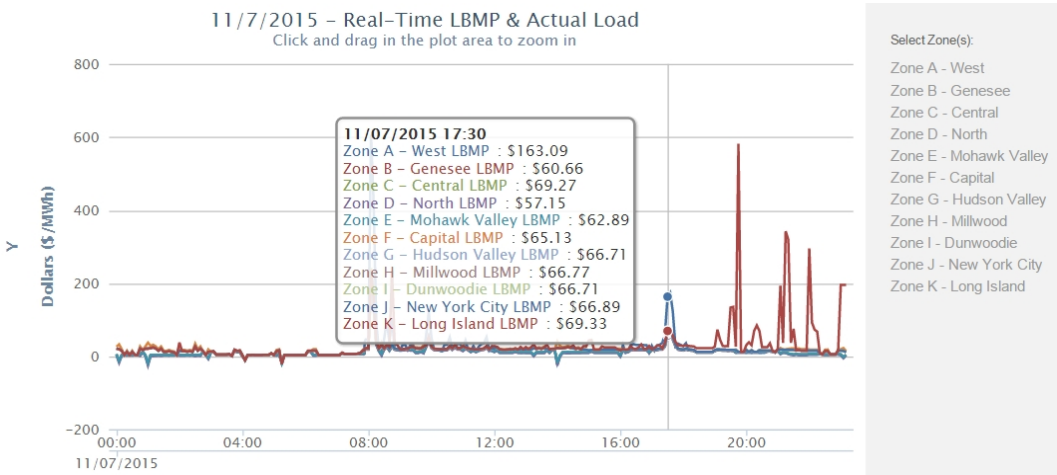

Fig. 3. Sample spot price curves from New York ISO

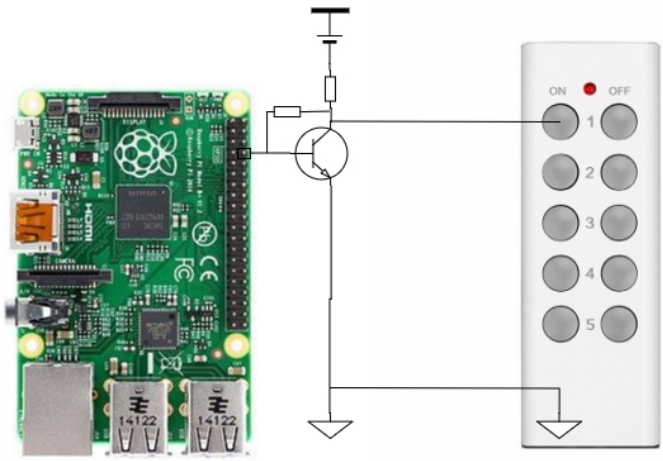

Fig. 4. The Smart Home hardware prototype wiring

On the Raspberry Pi platform, both C programs and Python programs are employed for different levels of functions. C programs are more straightforward for lower level hardware control, while Python programs are fast prototyping for higher level applications. Therefore, C programs are linked to an open source package called WiringPi to takes over the GPIO output control. Python programs

are written to implement the demand response algorithms. In the front-end, Web UI is programmed in the PHP language to provide convenient user setting adjustments.

Verification and Demonstration

The implemented of a Smart Home Testbed following the proposed designs is carried out as an Undergraduate Student project. A group of 6 senior-year undergraduates (2 from Electrical Engineering and 4 from Computer Science) constructed and set up this testbed in two months, mentored by 2 graduate students. In the process of the Testbed project, undergraduate students automatically split into three sub-groups, each having 2 people. The first group worked on hardware setup, the second group worked on DR related algorithm development in the SHEM, and the third one worked on Web UI development.

Fig. 4 shows the hardware wiring of the Raspberry Pi and the radio remote through transistor amplifiers. Owe to the peripheral supports, only transistors are used to amplify the signals from GPIOs pins. Fig. 5 shows the Web based UI for setting adjustments. In the 'User Settings' tab, an electricity price threshold can be entered and five preset modes are available to adjust their time of being turned on. In the 'Off Periods' tab, behaviors of appliances during the off-period can be configured.

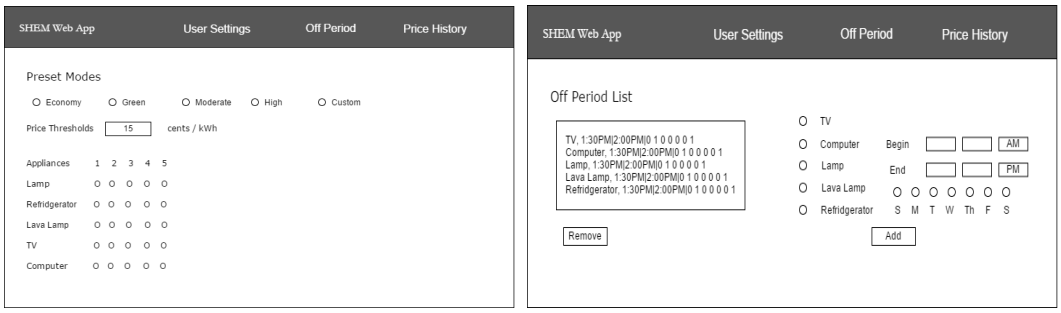

Fig. 5. Web UI based user interface

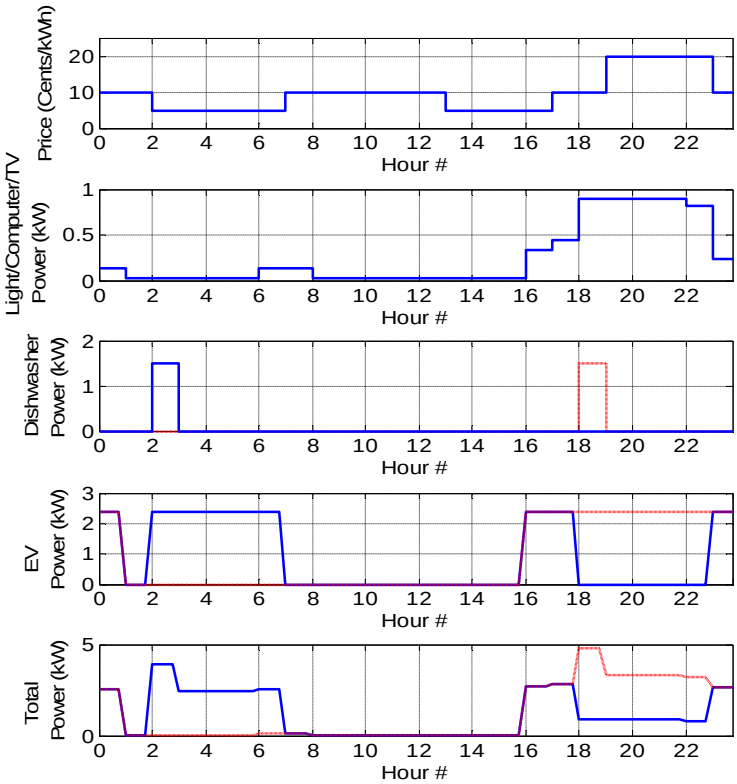

Fig. 6. Time-of-use appliance scheduling results

A. Verification of Time-of-Use Algorithm The delay-by-k hour algorithm for time-of-use demand response programs are verified on the testbed with 5 types of appliances: light, TV, computer, dish washer and electric vehicle. The first three types are configured as non-delayable, while dish washer and electric vehicles are delayable by a maximum of 8 hours. The input data contains the time-of-use tariff in the area, the power consumption and the turn-on time of each appliance. Case study result is given in Fig. 6, where dashed lines are the power consumption without the algorithm, while the solid ones are that with the algorithm.

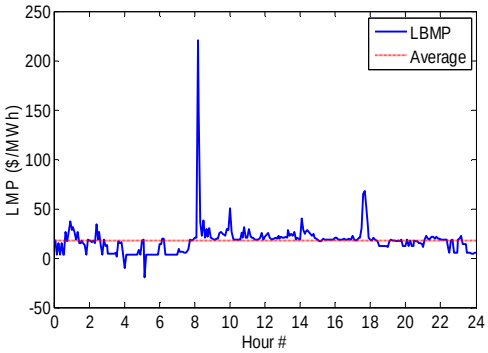

Fig. 7. Real time price signal on spot market

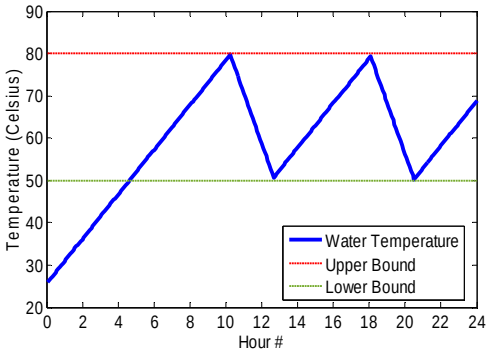

Fig. 8. Temperature in water heater without RTP

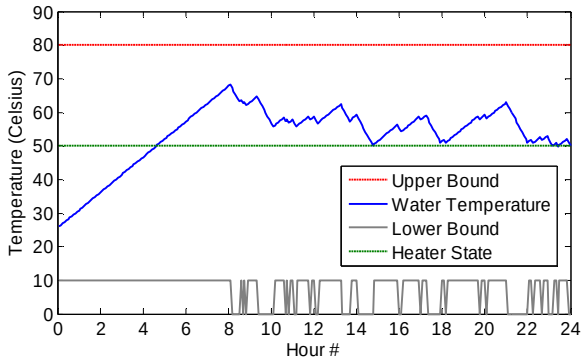

Fig. 9. Optimized temperature in water heater with RTP

From top down, the first figure plots the time-of-use electricity price which peaks at 7:00 PM – 11:00 PM. The second figure shows the aggregated power consumption of the non-delayable appliances whose peak power shows a concurrency with the electricity price. For the dishwasher being turned on a 6:00 PM, this algorithm compared the prices in the next 8 hours and found a lower price hour at 2:00 AM, thus delayed the dishwasher until then. A little different is the EV charging which takes 10 hours to finish. The algorithm found the lower electricity price hours for EV charging, which, as a result, is equivalent to avoiding the higher price hours. Finally, the total power consumption before and after the delay is compared in the last figure, where a noticeable peak avoiding from the time-of-use DR can be observed.

B. Verification of Real Time Price Response Algorithm An optimization of electric water heater directed by 5-minute ahead RTP is studied and shown in in Fig. 7 – Fig. 9. Fig. 7 shows the RTP of New York ISO on November 7, 2015. Fig. 8 shows a typical temperature in a water heater without any optimization, where water is heated up to the upper temperature limit and reheated when the lower temperature limit is reached. Shown in Fig. 9 is the optimized water heater on and off states, which reversely follows the price signal while maintaining the water temperature in the desired regions.

Conclusions

In this paper, a three-layer architecture design is proposed for a fast prototyping Smart Home Testbed is proposed. Implementation of the layered architecture on a Raspberry Pi SoC shows practicality for undergraduate students to test and verify ideas relevant to Smart Home designs. An

undergraduate student project example also verified that a SoC platform based Smart Home Testbed is suitable and effective for undergraduate level DR related model and algorithm implementation.

Acknowledgements

This work was financially supported by the College Natural Science Foundation of Jiangsu Province(16KJB4700004) and Doctoral Foundation of Jiangsu normal University(16XLR049) .

References

[1] H. Cui, R. Long, F. Li, X. Fang, and R. Long, "Distribution Network Reconfiguration with Aggregated Electric Vehicle Charging Strategy," in *IEEE Power and Energy Society General Meeting*, 2015, vol. 2015–Septe, no. 1, pp. 1–5.

[2] H. Cui, F. Li, Q. Hu, L. Bai, and X. Fang, "Day-ahead coordinated operation of utility-scale electricity and natural gas networks considering demand response based virtual power plants," *Appl. Energy*, vol. 176, no. 15, pp. 183–195, 2016.

[3] D.-M. Han and J.-H. Lim, "Design and implementation of smart home energy management systems based on zigbee," *IEEE Trans. Consum. Electron.*, vol. 56, no. 3, pp. 1417–1425, 2010.

[4] Z. Wei, S. Qin, D. Jia, and Y. Yang, "Research and design of cloud architecture for smart home," *Proc. 2010 IEEE Int. Conf. Softw. Eng. Serv. Sci. ICSESS 2010*, no. 60970, pp. 86–89, 2010.

[5] T. Perumal, a. R. Ramil, and C. Y. Leong, "Design and implementation of SOAP-based residential management for smart home systems," *IEEE Trans. Consum. Electron.*, vol. 54, no. 2, pp. 453–459, 2008.

[6] C. Y. Chen, Y. P. Tsou, S. C. Liao, and C. T. Lin, "Implementing the design of smart home and achieving energy conservation," *IEEE Int. Conf. Ind. Informatics*, pp. 273–276, 2009.

[7] X. Chen, T. Wei, and S. Hu, "Uncertainty-aware household appliance scheduling considering dynamic electricity pricing in smart home," *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 932–941, 2013.

[8] K. C. Sou, J. Weimer, H. Sandberg, and K. H. Johansson, "Scheduling smart home appliances using mixed integer linear programming," *IEEE Conf. Decis. Control Eur. Control Conf.*, pp. 5144–5149, 2011.

[9] A. H. Mohsenian-Rad and A. Leon-Garcia, "Optimal residential load control with price prediction in real-time electricity pricing environments," *IEEE Trans. Smart Grid*, vol. 1, no. 2, pp. 120–133, 2010.

[10] K. C. L. K. C. Lee and H.-H. L. H.-H. Lee, "Network-based fire-detection system via controller area network for smart home automation," *IEEE Trans. Consum. Electron.*, vol. 50, no. 4, pp. 1093–1100, 2004.

[11] B. El-Basioni, S. El-kader, and M. Abdelmonim, "Smart home design using wireless sensor network and biometric technologies," *Inf. Technol.*, vol. 2, no. 3, pp. 413–429, 2013.

[12] Q. Hu, F. Li, and C. Chen, "A Smart Home Test Bed for Undergraduate Education to Bridge the Curriculum Gap From Traditional Power Systems to Modernized Smart Grids," *Educ. IEEE Trans.*, vol. 58, no. 1, pp. 32–38, 2015.