Pers Ubiquit Comput (2014) 18:535–542
DOI 10.1007/s00779-013-0671-1

ORIGINAL ARTICLE

Implementation of a cost-effective home lighting control system on embedded Linux with OpenWrt

Cheong Ghil Kim · Kuinam J. Kim

Received: 21 December 2012 / Accepted: 15 March 2013 / Published online: 28 April 2013

Abstract In recent years, the significance of greenhouses has been increased greatly because the world has been facing serious problems with energy as its growing demand. At the same time, home automation systems have been steadily gaining popularity; growing toward smart home based on Cloud technology. This paper introduces a cost-effective home energy saving system based on a small embedded system with remote controlling feature. For this purpose, the system is composed of a wireless router based on embedded Linux for the platform to develop a low-cost energy control server and a smart phone for remote light control app. The prototype system was implemented by porting OpenWrt onto the wireless router which is connected with an interface board with LEDs attached. The remote access and GUI function were implemented by TCP/IP programming using Apple iPhone. The operation of the remote control system was verified by socket communication between the smart phone and the wireless router, and by USB communication between the wireless router and the interface board. The implementation result shows that an OpenWrt-based wireless router can give benefits of saving energy and safety through lighting control.

Keywords OpenWrt · Wireless router · Energy saving · Smart phone

C. G. Kim (✉)
Department of Computer Science, Namseoul University, Cheonan, Choongnam, South Korea
e-mail: cgkim@nsu.ac.kr

K. J. Kim
Department of Convergence Security, Kyonggi University, Suwon-si, Gyeonggi-do, South Korea
e-mail: harap123@hanmail.net

1 Introduction

In recent years, the significance of greenhouses has been increased greatly because the world has been facing serious problems with energy as its growing demand. In connection with this situation, home automation systems have been steadily gaining popularity, furthermore, growing toward smart home [1] technology in the smart grid [2]. In general, home automation systems have the functions of easy accessibility and convenient control of various appliances and devices in the home including monitoring all events happening at home both locally and remotely via wired and wireless networks. Recently, residential energy management has become an active topic and several researches have been i ntroduced [2–4].

In commercial buildings, lighting accounts for up to 40 % of total energy cost. Reducing this energy consumption has become a major goal. It is a well-known solution that to replace existing lights with more energy-efficient lighting sources such as LED is one of the ways to reduce this massive pool of energy use. An even greater level of energy reduction comes from turning off lights when they are not needed, optimizing light levels to suit worker needs and reducing overall demand for lighting energy. Improving system-wide control over lighting is the best way to ensure that lighting energy is automatically reduced as much as possible. Lighting control solutions, based on a variety of technologies, have been proven to reduce lighting energy consumption in commercial and industrial buildings by up to 70 %. These solutions have been limited in the past by cost, complexity and applicability, but new wireless technologies are providing ways to expand the capabilities of lighting control and offer them to a wider set of customers [5, 6]. Especially, recent advances in consumer wireless networking technologies and mobile devices such as smart phones and tablets are enabling users to have more intelligent and efficient smart home in which all controls of appliance and light are managed.

Nowadays, wireless networks are everywhere and even in home 802.11 wireless networks are popular, avoiding wiring costs and providing connectivity for all rooms. Therefore, we can find low-cost wireless router easily, and they can be utilized for low-cost embedded Linux platform using OpenWrt which targeted at the Linksys WRTG54 initially, but now targets many embedded wireless devices including equipment from Asus, D-Link, NetGear, Soekris, Viewsonic, and Linksys [7]. Primarily, OpenWrt is an operating system used on embedded devices to route network traffic. The main components are the Linux kernel, uClibc and busybox. All components have been optimized for size, to be small enough to fit the limited storage and memory available in home routers [8].

A typical home automation systems of today consist of a Wi-Fi router or an internet connection, a smart home gateway and multiple nodes (known as end devices). These systems can commonly be installed in standard home, making the homes smart. In this paper, a small residential wireless router is used as smart gateways incorporating power management features and the implementation result shows its feasibility of substantially reducing the power consumption of a home. For this purpose, we take advantage of the wireless router with OpenWrt as the home energy saver with remote control feature.

Over the years, as the number of electric and electronic devices increases dramatically at home and in buildings, it was difficult to manage the waste of energy due to the inefficient light control and illumination distribution [9]. In addition, it is not practical to rely on users to directly control the light switch to save energy. To this end, this paper introduces a remote LED control system which is based on OpenWrt. The prototype system was implemented by porting a wireless router to OpenWrt and by connecting an interface board to an LED. Also, the system communicated with a smart phone via socket communication to control LED.

The paper is structured as follows, in Sect. 2 we will provide detailed background information on home automation systems and OpenWrt and Sect. 3 will introduce information regarding the proposed home energy saving systems based on OpenWrt. In Sect. 4 we show the procedures of implementation and its result. Section 5 covers the conclusion.

2 Background

2.1 Home automation systems

The researches on home automation systems have been conducted for long times. However, it is true in some sense that these technologies have not been widely adopted with several barriers such as high cost of ownership, inflexibility, poor manageability and difficulty achieving security [10]. Even under this circumstance, researches in this field have extended the enhanced capabilities of the technology into areas such as remote monitoring and control, power management, tracking and security systems and disaster warning systems. For examples systems, Orange [11], Aware Home [12], House_n [13], eHome [14], and Web-based [15] have been built, and most of them have been implemented with technologies commercially available: Z-Wave [16], ZigBee [17] and X10 [18]. In now days, especially, power management has been of particular interest as it allows for a greener future and is added advantage to users as a cost-cutting measure [19, 20]. Other interesting research that broadens the areas of home automation may include the issues of elderly people, robotics and so on [21–25].

The core function of home automation systems in smart home has to allow users to access and control all the devices at home either locally or remotely via the internet. For this purpose, all devices at home have to have common interface and link it to systems and services remotely. Figure 1 shows a general block diagram of home automation systems consisting of with light, media, HVAC, security and outdoor controls. Here, a home gateway is networking all devices around home as the control tower of the system.

Basic functions of hardware devices may be summarized as bellowing [20].

2.1.1 *Home gateway*

The purpose of a gateway in a home automation system is to enable bridging between different technologies used. It also acts as a means to connect the home system to external services and vice versa with the help of a Wi-Fi router. Inter-connections between home gateway and home appliances are possible through wireless networks such as Bluetooth, WIFI or wired network, Home PNA and IEEEI394.

2.1.2 *Sensors*

Sensors can be used to relay information on certain aspects from within or outside the home to enable the system to adapt to the occurring changes. Sensors communicate directly with the home gateway and feed the system information with regards to light intensity inside a particular room, temperature inside and outside the home and motion sensing to name a few. Also, it should be able to automatically turn off devices when it detects no motion in the room for a brief period of time. Lighting systems allow

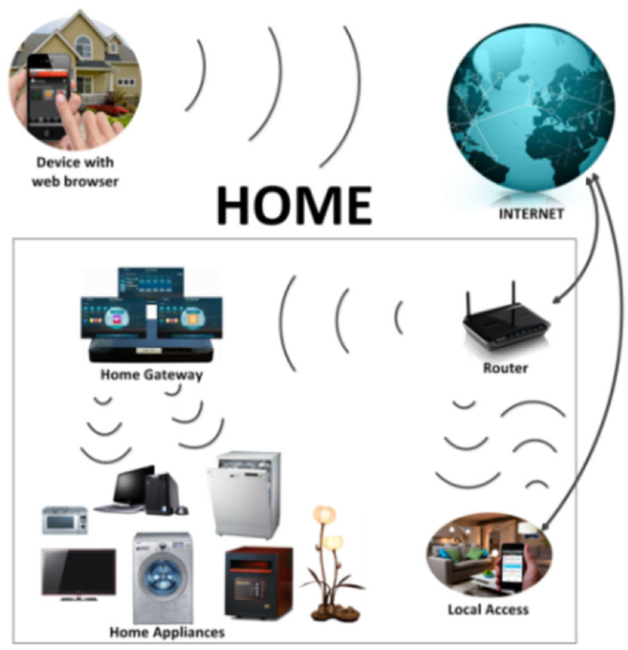


Fig. 1 General architecture of home automation systems

just enough light wherever and whenever the user requires it. This is done with the help of sensors that relay information regarding the intensity of the light inside and/or outside the house.

2.1.3 Standard appliance and devices

There are many devices scattered around a home. Depending on the size and number of occupants in a home, the number of devices found in homes can vary. It should also be noted that there tends to be more wastage in terms of electricity in homes with a higher number of occupants due to human negligence. They may require different amounts of power to run, and it is necessary to find specific control policy for each device.

2.2 OpenWrt

OpenWrt is a Linux-based embedded platform initially targeted at the Linksys WRTG54 [20], Wi-Fi capable residential gateway from Linksys, but now targets many embedded wireless devices including equipment from Asus, D-Link, NetGear, Soekris, Viewsonic and son on. Basically, any equipment with adequate flash memory and RAM that uses the Infeon ADM5120, TI AR7, Intel IXP4xx series or the Broadcom BCN63xx series can typically run OpenWrt. The functionality may be limited in some instances, depending on the equipment in the device [7].

Therefore, OpenWrt is a highly extensible GNU/Linux distribution for embedded devices. It is built from the ground up to be a full-featured, easily modifiable operating system for small network devices [8]. OpenWrt provides features similar to a modern package-based Linux distribution: a built-in web server with CGI support, an SSH server and most importantly, a package management tool, ipkg. Using ipkg, new applications, tools and kernel drivers can be added and removed at runtime, without requiring a reboot. OpenWrt is loaded onto the prospective router by using the standard firmware upgrade mechanism provided by the router. However, in the sense that it does not provide a complete desktop experience, it still provides much of the functionality expected from a modern Linux distribution [7].

Space on embedded devices is a very real constraint, because of this OpenWrt uses busybox as the shell environment. BusyBox embeds many of the common command line tools, so a distribution does not have to provide individual executables. This cuts down on the space requirements needed allowing OpenWrt to provide a near complete Linux experience on devices with limited space. With OpenWrt installed, the router can be configured as a wireless access point, a wireless client or a member of an ad hoc network. The latter option allows mesh networks to be created, requiring no networking infrastructure. This is useful for experimenting with emergent or flocking behaviors. OpenWrt has different approach to building a firmware, downloading, patching and compiling everything from scratch, including the cross-compiler.

OpenWrt does not contain any executables and only very few sources; it is an automated system for downloading the sources, patching them to work with the given platform and compiling them correctly for that platform. What this means is that just by changing the templates, you can change any step in the process [7].

Almost any standard non-GUI Linux/Unix program can be recompiled for use in OpenWrt, assuming it can fit in the smaller memory footprint of the target router. The cross-compilation and packaging techniques are simple and well documented. Installation of software uses the same ipkg mechanism as for system packages. Both the system and the third-party package libraries contain a growing list of standard Linux tools and device drivers [26].

The overview of OpenWrt including directory structure, packages and external repositories, toolchain, software architecture, is summarized as followings [27, 28].

2.2.1 Directory structure

There are four key directories in the base: tools, toolchain, package and target; tools and toolchain refer to common tools which will be used to build the firmware image, the compiler, and the C library.

2.2.2 *Packages and external repositories*

In an OpenWrt firmware, almost everything is an ipk, a software package which can be added to the firmware to provide new features or removed to save space. Note that packages are also maintained outside of the main trunk and can be obtained from subversion using the package feeds system. Those packages can be used to extend the functionality of the build system and need to be linked into the main trunk.

2.2.3 *Toolchain*

OpenWrt automates the toolchain creation for a particular architecture by passing the right arguments to the compiler during the cross-compilation process and using patches known to be working for the target architecture. It also allows users to switch between different combinations of gcc, binutils, kernel headers and uClibc, so that users can easily check regressions and programs compilation. Users can even customize the compiler flags passed to cross-compiler so users can test optimizations features of your cross-compiler or different locations for your cross-compiled libraries.

2.2.4 *Software architecture*

OpenWrt uses the common embedded Linux tools such as uClibc, busybox and shell interpreter and provides a hardware abstraction layer and package manager. Figure 2 shows the software stack that OpenWrt uses.

Every architecture uses a different Linux kernel allowing the user-space environment to be shared and consistent across devices. Therefore, users only need to recompile the uClibc and packages to match your target architecture to get the same programs running on a totally different embedded device.

2.2.5 *System and package configuration*

UCI which stands for Unified Configuration Interface was adopted with the extent of OpenWrt to other devices which did not have the NVRAM to store their settings into a separate flash partition. Since UCI is a C library, it can be easily integrated into an existing user-space application or to develop configuration storage that is OpenWrt compatible for new applications. Further developments for UCI include a web interface that uses UCI as a configuration file format as well as SNMP plugins to easily change the configuration and take actions on the embedded device.

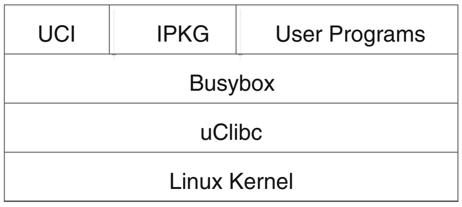


Fig. 2 Software architecture of OpenWrt

3 System components and configuration

In general, for home energy control, a home gateway has to equip with the functions of sensing, processing and networking. Figure 3 [29] shows a typical power management scheme in home automation system including all power sources around home in which intelligent agents are controlling different devices with different operation environments.

This paper introduces a cost-effective home energy saving system based on a small embedded system with remote controlling feature. For this purpose, the system is composed of a wireless router based on embedded Linux for the platform to develop a low-cost energy control server and a smart phone for remote light control app as shown in Fig. 4. The prototype system was implemented by porting OpenWrt onto the wireless router which is connected with an interface board with LEDs attached.

The system comprises of a smart phone to control a wireless router on OpenWrt, an interface board and LEDs, and a wireless router to communicate with the board. Here, the router is DIR-825 [30] wireless router of DLink and acts as a CPU board to control and manage the system. Figure 5 shows its hardware and software specifications are summarized in Tables 1 and 2, respectively. Figure 5 shows the router of DIR-825, and Fig. 6 depicts the link diagram between PC and OpenWrt router [31].

The smart phone, iPhone4 of Apple, communicates with the wireless router and allows the user interface (UI) to control the LED remotely through the communication with

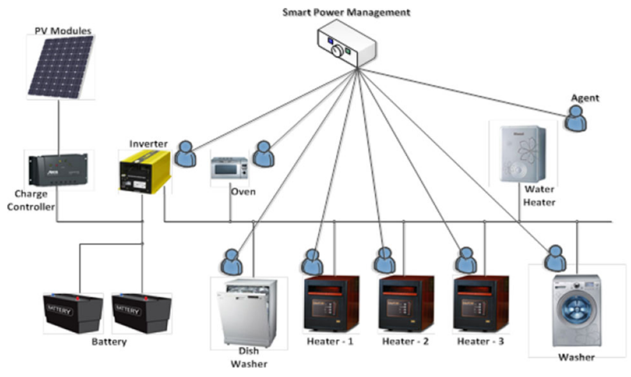


Fig. 3 Typical block diagram of smart power management systems of smart home

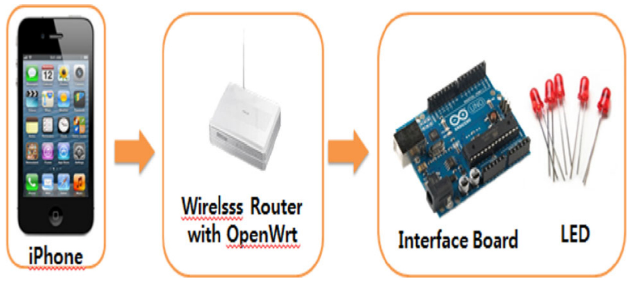


Fig. 4 System operating flow

Fig. 5 DIR-825

Table 1 Specification of wireless router

Items	Descriptions
Architecture	MIPS
Vendor	Atheros
Bootloader	U-Boot
System-on-chip	Atheros AR9132
CPU speed	400 MHz
Flash size	32 MB
RAM	64 MB
Network	4 × 1
Wireless	Atheros AR9160 BB/MAC and AR9103 2.4 GHz 3 × 3 MIMO radio b/g/n
Ethernet	RealTek RTL8366S Gigabit w/port-based VLAN support
USB	Yes
Serial	Yes

the interface board installed in its USB. For the interface board, we use Arduino UNO [28] of Arduino Lab. The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. Figure 7 shows the image of Arduino UNO interface board, and Table 3 shows the specification. We used DIR-825

Table 2 Installation environment

Host PCs OS	Window XP or later Linux Fedora 13	Required for firmware upload Kernel 2.6.36
Router	BUFFALO WZR-HP-G300NH	Required
Firmware	OpenWrt Backfire 10,03	The newest version: 10.03.1-rc6 Backfire porting
Package	TFTP	Firmware installation

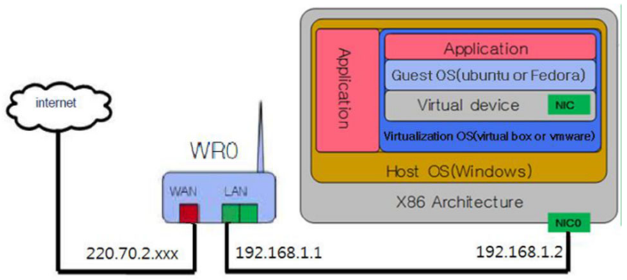


Fig. 6 OpenWrt router connection diagram

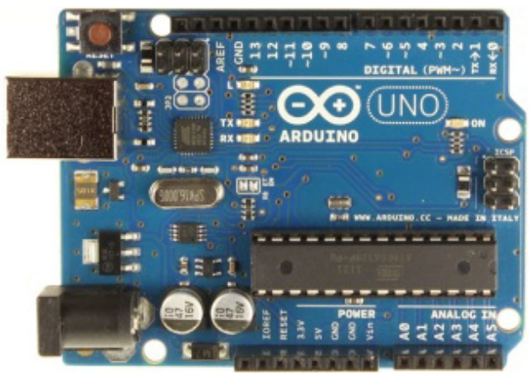


Fig. 7 Arduino UNO

wireless router of DLink and 5 ohm red LEDs. Figure 4 illustrates the overall structure of the system with the functionality of controlling the brightness of LEDs using iPhone remotely.

4 Implementation environments

4.1 Cross-development environment

In order to develop application using wireless routers, cross-development environment should be utilized. Here, a cross-compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is running. In general, cross-compiler tools are used to generate executables for embedded system or multiple

Table 3 Specification of Arduino UNO

Items	Specifications
Microcontroller	ATmega328
Operating voltage	5 V
Input voltage (recommended)	7 ~ 12 V
Input voltage (limits)	6 ~ 20 V
Digital I/O pins	14 (of which 6 provide PWM output)
Analog input pins	6
DC current per I/O pin	40 mA
DC current for 3.3 V pin	50 mA
Flash memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock speed	16 MHz

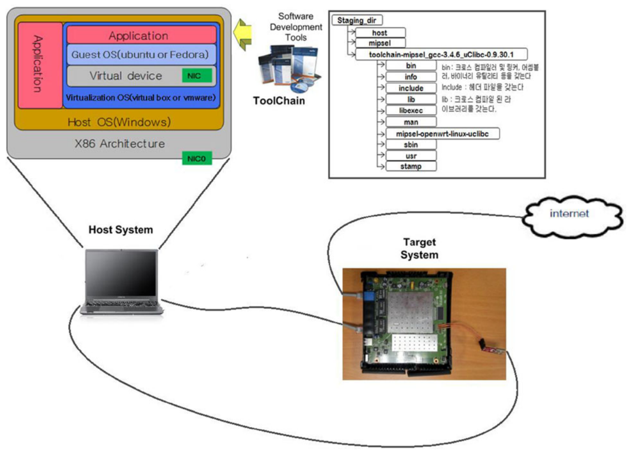

Application
Application
Guest OS(ubuntu or Fedora)
Virtual device
NIC
Virtualization OS(virtual box or vmware)
Host OS(Windows)
X86 Architecture
NIC0
Software Development Tools
ToolChain
Staging_dir
host
mipsel
toolchain-mipsel_gcc-3.4.6_uClibc-0.9.30.1
bin
info
include
lib
libexec
man
mipsel-openwrt-linux-uclibc
sbin
usr
stamp
bin : 크로스 컴파일러 및 링커, 어셈블러, 바이너리 유틸리티 등을 갖는다
Include : 헤더 파일을 갖는다
lib : 크로스 컴파일 된 라이브러리를 갖는다.
Host System
Target System
internet

Fig. 8 Cross-compiling

platforms. It is used to compile for a platform upon which it is not feasible to do the compiling, like microcontrollers that do not support an operating system. It has become more common to use this tool for paravirtualization where a system may have one or more platforms in use [32].

Figure 8 shows the cross-compiling environment using OpenWrt. The execution of cross-compiling is conducted by executing make after finishing coding in a director and assigning the path of cross-compiler.

4.2 Installation of OpenWrt

The first step to implementing the proposed system is to replace the AP firmware with OpenWrt. The easiest method is to use the built-in firmware upgrade web page on the AP to upload the OpenWrt firmware. However, not all devices support this function. The second method is to install the firmware via TFTP. When the AP boots, it runs a bootloader which validates and loads the actual firmware. The method to interrupt the bootloader depends on the AP model.

The procedure for installing OpenWrt on the router follows the bellowing sequence using TFTP.

- Connecting the router and PC
- Release window firewall and activate FTTP
- Download the newest OpenWrt firmware
- Initialize the IP of network adopter
- ARP route setting
- Firmware upload through TFTP
- Remove information of ARP route
- Change the IP of network adopter
- Change the root password of the router

Open WRT system can be configured in three different ways. The first is the configuration via graphical interface. To configure the system via a graphical interface one should use any web browser. Another way to configure the operating system is via configuring the command line (terminal). The third way is to edit OpenWrt system configuration files, which are stored in/etc./config/directory. To configure the system it is possible to use built-in editor.

OpenWrt system can be configured in three different ways. The first is the configuration via graphical interface. To configure the system via a graphical interface, one should use any web browser. Another way to configure the operating system is via configuring the command line (terminal). The third way is to edit OpenWrt system configuration files, which are stored in/etc./config/directory. To configure the system, it is possible to use built-in editor.

4.3 OpenWrt versions

Stable release versions have been made, and currently, three major versions of OpenWrt have been introduced.

4.3.1 Whiterussian

The legacy Whiterussian releases are still available, but are neither supported nor maintained. The firmware is based on Linux kernel 2.4 and mostly for the routers using Broadcom CPU.

4.3.2 Kamikaze

Kamikaze was introduced in order to support many CPUs. Substantial improvements to the build environment were made under the Buildroot-NG fork in 2006, and these were merged back into the main Kamikaze development branch in mid-October 2006 and became the first official Kamikaze release. There were several Kamikaze releases.

4.3.3 Backfire

First Backfire 10.03 release was in April 2010. The maintenance release 10.03.1 was released in December 2011. And then attitude adjustment 12.09 is the new stable release.

4.4 Execution

First of all, we installed OpenWrt in the router and then KMOD-USB-ACM package. Next, we connected it to Arduino in order to prepare communication between the board and router. The UI was applied to the smart phone and operated to enable socket communication through TCP/IP protocol by using programmed iOS SDK [33]. Then, we used the library of C and Linux for socket communication by TCP/IP protocol [34], and set the Baud ratio of the server program for control data transfer by USB port.

We selected a USB port and used a file descriptor to send data by 1 byte each [35], so that it fits to the OpenWrt system by doing cross-compile. The interface board of Arduino received data by each byte from the router through USB port. We programmed the interface board to deliver the control signal to the LED when all data were received [36]. Finally, we controlled the brightness of the LED by the MOSFET switch that depends on the control signal.

Figure 9 is the picture of the overall structure of the system. We created a breadboard with LED and then connected it to the main board with the USB port. We ran the OpenWrt-based software and then inserted command for starting as in Fig. 10. As a result, the smart phone became ready to communicate with the wireless router.

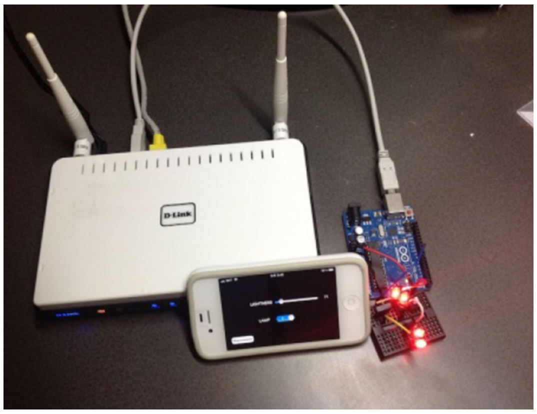
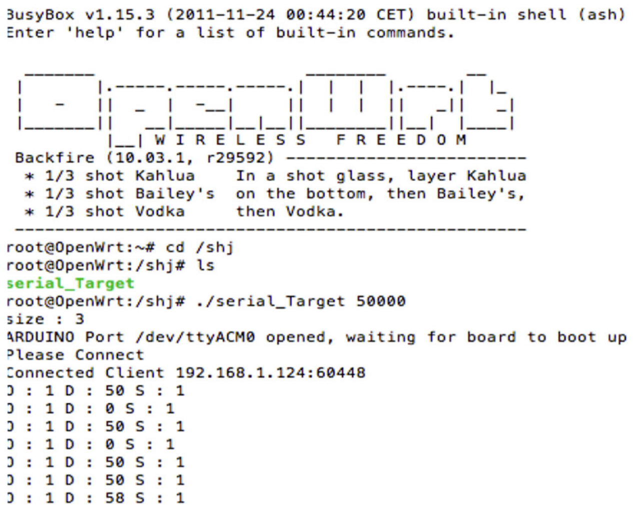


Fig. 9 Prototype implementation



```
BusyBox v1.15.3 (2011-11-24 00:44:20 CET) built-in shell (ash)
Enter 'help' for a list of built-in commands.

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 Backfire (10.03.1, r29592) --------------------------
  * 1/3 shot Kahlua    In a shot glass, layer Kahlua
  * 1/3 shot Bailey's  on the bottom, then Bailey's,
  * 1/3 shot Vodka     then Vodka.
 ---------------------------------------------------
root@OpenWrt:~# cd /shj
root@OpenWrt:/shj# ls
serial_Target
root@OpenWrt:/shj# ./serial_Target 50000
size : 3
ARDUINO Port /dev/ttyACM0 opened, waiting for board to boot up
Please Connect
Connected Client 192.168.1.124:60448
D : 1 D : 50 S : 1
D : 1 D : 0 S : 1
D : 1 D : 50 S : 1
D : 1 D : 0 S : 1
D : 1 D : 50 S : 1
D : 1 D : 50 S : 1
D : 1 D : 58 S : 1
```

Fig. 10 Program start

We loaded the UI and connected the wireless router and TCP socket communication from the smart phone as shown in Fig. 4. Then, we controlled the brightness of LEDs by data transfer. The brightness is controllable from light (255) to dark (50). Figure 9 shows the change of brightness of LED as changing brightness values through UI on smart phone.

Moreover, after changing the firewall settings of the router to allow external router access, not only local wireless router, but also control of 3G internet connection from external network became possible.

5 Conclusion

Nowadays, the significance of greenhouses has been increased greatly because the world has been facing serious problems with energy as its growing demand. Under this circumstance, this paper introduces a low-cost embedded system that can save home energy. The prototype implementation was made by control LED lights. For this purpose, we implemented a system that can control LED by using an interface board which was connected to a wireless router ported with OpenWrt by a smart phone. This enabled socket communication and link with peripheral device using a real wireless router, and utilization of a wireless router as an embedded system. Another application of the results of this paper may include control of remote household automation system via internet. If such system is utilized, it is expected that power consumption will be reduced, since we can turn off the lights in our homes via the internet even when we are on the opposite side of the globe. Furthermore, this system can contribute to saving power by enabling individuals to control the brightness of unnecessarily bright LEDs in a more customized fashion.

Acknowledgments Funding for this paper was provided by Namseoul University.

References

1. Haines V, Mitchell V, Cooper C, Maguire M (2007) Probing user values in the home environment within a technology driven smart home project. Pers Ubiquitous Comput 11(5):349–359
2. Melike EK (2011) Wireless sensor networks for cost-efficient residential energy management in the smart grid. Smart Grid, IEEE Trans 2:314–325
3. Dhiren T, Al-Kuwari AMAH, Potdar V (2011) Energy conservation in a smart home. In: Digital ecosystems and technologies conference (DEST). 2011 Proceedings of the 5th IEEE International Conference on, pp 241–246
4. Corno F, Razzak F (2012) Intelligent energy optimization for user intelligible. Smart Grid, IEEE Trans 3(4):2128–2135
5. Daintree Networks Inc (2010) The value of wireless lighting control.www.daintree.net. Accessed 17 Mar 2013
6. Rand P (2013) Strategic marketing wireless lighting control: the bright road ahead. Low power RF IEEE 802.15.4 and ZigBee products, Texas instruments incorporated (TI) http://www.eetimes.com/ContentEETimes/Documents/TI%20paper.pdf. Accessed 17 Mar 2013
7. SANS Institute, Murray J (2009) An inexpensive wireless IDS using Kismet and OpenWRT http://www.sans.org/reading_room/whitepapers/detection/inexpensive-wireless-ids-kismet-openwrt_33103. Accessed 17 Mar 2013
8. Wikipedia (2013) OpenWrt. http://en.wikipedia.org/wiki/OpenWrt. Accessed 17 Mar 2013
9. Lee J (2011) IEEE 1451 interface for smart grid. Dissertation, Hanyang University Graduate School
10. Brush AJB, Lee B, Mahajan R, Agarwal S, Saroiu S, Dixon C (2012) Home automation in the wild: challenges and opportunities. In: Sensing technology (ICST), 2012 sixth international conference on, pp142–145
11. Mozer MC (2005) Lessons from an adaptive house. In: Cook D, Das R (eds) Smart environments: technologies, protocols, and applications. Wiley, Hoboken, pp 273–294
12. Kietz J, Patel S, Jones B, Price E, Mynatt E, Abowd A (2008) The georgia tech aware home. Ext. Abstracts CHI, pp 3675–3680
13. Intille S (2002) Designing a home of the future. IEEE Pervasive Comput 1(2):80–86
14. Koskela T, Väänänen-Vaninio-Mattila K (2004) Evolution towards smart home environments: empirical evaluation of three user interfaces. Pers Ubiquitous Comput 8(3–4):234–240
15. Selçuk GH, Muharrem G (2010) Web based ZigBee enabled home automation system. In: 13th international conference on network-based information systems (NBiS) 2010, pp 290–296
16. Z Wave World, Inc (2007) Ask the expert. http://www.zwaveworld.com/ask/ask8.php. Accessed 17 Mar 2013
17. ZigBee alliance (2008) ZigBee specification. http://www.zigbee.org/Specifications.aspx. Accessed 17 Mar 2013
18. Rye D (2013) The X-10 POWERHOUSE power line interface model #PL513 and two-way power line interface model # TW523. ftp://ftp.x10.com/pub/manuals/technicalnote.pdf. Accessed 17 Mar 2013
19. Aman S, Simmhan Y, Prasanna VK (2013) Energy management systems: state of the art and emerging trends. Commun Mag, IEEE 51(1):114–119
20. Özden T, Okumu HI (2012) Designing a load agent for power management with a multi-agent home automation system. In: Innovations in intelligent systems and applications (INISTA), 2012 international symposium on, pp 1–5
21. Nazabal JA, Matias IR, Fernandez-Valdivielso C, Falcone F, Branchi P, Mukhopadhyay SC (2012) Home automation based sensor system for monitoring elderly people safety. In: The sixth international conference on sensing technology (ICST 2012), pp 142–145
22. Sa IK, Ahn HS, Yi KM, Choi JY (2009) Implementation of home automation system using a PDA based mobile robot. In: Industrial electronics, 2009. ISIE 2009. IEEE international symposium on, pp 1761–1766
23. Li N, Yan B, Chen G, Govindaswamy P, Wang J (2010) Design and implementation of a sensor-based wireless camera system for continuous monitoring in assistive environments. Pers Ubiquitous Comput 14(6):499–510
24. Chang YS, Wang WJ, Hung YS (2013) A near field communication-driven home automation framework. Pers Ubiquitous Comput 17(1):169–185
25. Tolmie P, Crabtree A, Egglestone S, Humble J, Greenhalgh C, Rodden T (2010) Digital plumbing: the mundane work of deploying UbiComp in the home. Pers Ubiquitous Comput 14(3):181–196
26. Kurt TE (2006) Low-cost on-board linux, vision, wi-fi, and more for the roomba robotics base http://www.aaai.org/Papers/Symposia/Spring/2007/SS-07-09/SS07-09-019.pdf. Accessed 17 Mar 2013
27. Fainelli F (2008) The OpenWrt embedded development framework. http://downloads.openwrt.org/people/florian/fosdem/fosdem.pdf. Accessed 17 Mar 2013
28. Arduino (2013) http://arduino.cc/en/Main/arduinoBoardUno. Accessed 17 Mar 2013
29. Wikipedia (2013) Linksys WRT54G series http://en.wikipedia.org/wiki/Linksys_WRT54G_series. Accessed 17 Mar 2013
30. OpenWrt: D-link DIR-825 (2013) http://wiki.openwrt.org/toh/d-link/dir-825. Accessed 17 Mar 2013
31. Jeong GS, Kim CG, Kwak HG, Jang H (2009) Embedded linux systems and applications using wireless router (ISBN-10: 8957271309). Green, Seoul, Korea
32. Wikipedia (2013) Paravirtualization. http://en.wikipedia.org/wiki/Paravirtualization. Accessed 17 Mar 2013
33. MarkD, Nutting J, LaMarche J (2010) Beginning iPhone 5 development: exploring the iOS 5 SDK. Wikibooks, pp 259–366
34. Yoon S (2007) TCP/IP programming. Orangemedia, pp 110–112
35. Arduino forum (2013) UNO serial latency. http://arduino.cc/forum/index.php?topic=96. Accessed 17 Mar 2013
36. Arduino playground (2013) Arduino MOTOR/LED control. http://arduino.cc/playground/MotorControlShieldV3/0. Accessed 17 Mar 2013