Research Article

Cross-Platform Android/iOS-Based Smart Switch Control Middleware in a Digital Home

Guo Jie,¹ Cheng Bo,² Zhao Shuai,² and Chen Junliang²

¹Beijing Liyun Technology Development Company, Beijing 100192, China

²State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Correspondence should be addressed to Cheng Bo; chengbo@bupt.edu.cn

Received 4 February 2015; Revised 29 April 2015; Accepted 8 June 2015

Academic Editor: Francesco Gringoli

Copyright © 2015 Guo Jie et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With technological and economic development, people's lives have been improved substantially, especially their home environments. One of the key aspects of these improvements is home intellectualization, whose core is the smart home control system. Furthermore, as smart phones have become increasingly popular, we can use them to control the home system through Wi-Fi, Bluetooth, and GSM. This means that control with phones is more convenient and fast and now becomes the primary terminal controller in the smart home. In this paper, we propose middleware for developing a cross-platform Android/iOS-based solution for smart switch control software, focus on the Wi-Fi based communication protocols between the cellphone and the smart switch, achieved a plugin-based smart switch function, defined and implemented the JavaScript interface, and then implemented the cross-platform Android/iOS-based smart switch control software; also the scenarios are illustrated. Finally, tests were performed after the completed realization of the smart switch control system.

1. Introduction

With the improvement of living standards and the changes in the concept of consumption, the smart home has quietly entered people's lives. The Internet of things makes it possible for objects to exchange information and control each other through wired or wireless communication networks, forming a massive network system that endows things with "wisdom" and therefore integrates human society with the system of objects. The appearance of the Internet of things provides a new opportunity for the smart home control system [1–5]. Normally, the following control plans are included in a smart home control system: controlling home appliances with an infrared remote controller, touch tablet, LCD monitors, keyboards, and other devices. The design of terminal control for the abovementioned systems is complicated and requires matched controllers, which leads to high user cost and does not support remote control. Currently, most smart home control systems adopt the PC as the control terminal, which clearly presents the problems of being bulky, inconvenient,

costly, and unportable, not meeting the needs of the modern smart home control system. As smart phones have become increasingly popular, we can use them to control the home system through Wi-Fi, Bluetooth, and GSM. This means that control with phones is more convenient and fast and has become the primary terminal controller of the smart home control system [6–13]. Using a mobile smart phone, people can control their home appliances at any time, changing the original static home environment into a dynamic smart digital home. However, the development of mobile applications is not simple; instead, it is a challenge that requires the developers to use multiple programming languages, such as SDK and API, on multiple smart cellphone operating systems. There are several smart phone operating systems that dominate the market, such as Android, iOS, and Windows Phone. However, in order to develop mobile smart switch applications to the majority of the smart phones, it is necessary to develop and deploy the smart switch application to the mainstream smart phone operating system. Therefore, there is a great demand to provide a cross-platform mobile home

control software solution, which can develop the application only once, and deploy it to several mobile operating systems. The contributions of this paper are as follows:

- (i) Proposing a cross-platform smart switch control software development framework for mobile smart phone, from the top to bottom view, which consists of UI layer, HTML Rendering Engine layer, Smart Switch Plugin layer, and Mobile OS layer.
- (ii) Adopting the plugin technology to encapsulate the public and user-defined API and realize the basic functions of the smart switch.
- (iii) Defining the JavaScript interface for smart switch, such as network configuration, device connection, device switch control, device time synchronization, device information reception, and time setting.
- (iv) Developing the smart switch and to utilize the smart phone to control the switch and the timer remotely and prove the possibility of developing cross-platform (Android/iOS) smart switch cellphone controlling software.

The rest of the paper is organized as follows. Section 2 briefly reviews related works. Section 3 describes the architecture of cross-platform Android/iOS-based switch control software middleware. Section 4 illustrates the scenarios. Section 5 is the performance measurements and analysis. Finally, Section 6 concludes the paper and the future work.

2. Related Works

When developing a smart phone application, there are two possible approaches to determine: the first approach is to develop a native application for each of the smart phone operating systems, and the second approach is to use a cross-platform development framework in order to develop the application only once and deploy it to several operating systems.

The native development approach is certain binary system executable programs installed in the equipment, included in the software development kit (SDK) and spread by the app store. Each mobile operating system has an SDK; unfortunately, different operating systems have different SDKs. For example, we must use iOS SDK and development tools to build an application for iOS and must use Objective-C programming language to code the application. However, an Android application must be developed using the Android SDK and coded in the Java language. Therefore, to build a mobile application, we have to know each SDK and use the corresponding programming language to code the application. Because the SDK of each platform has a steep learning curve, the development of cross-platform applications for smart phones is very complicated.

The cross-platform development approach means that the development of the software does not rely on any specific operating system [14–20]; instead, the coding set can be used in different operating systems without any modifications. The development approach to the cross-platform smart phone applications is geared to the need for multiple terminal

platforms and developing languages. Over the last few years, several cross-platform development technologies became popular with the purpose of deploying a single application to several smart phone operating systems aiming to reduce the amount of time. XMLVM [21] is introduced to be a solution for cross-compilation, which is a byte code level cross-compiler that is based on the Java byte code instructions. DragonRad [22] is a cross-platform mobile application development platform by Seregon Solutions Inc., which supports iPhone, Android, BlackBerry, and Windows mobile OS. RhoMobile [23] is another solution for the cross-compilation, which is a Ruby-based mobile development, which can be used across Linux, Mac, and Windows OS. PhoneGap [24] is an open-source mobile development framework that supports most heterogeneous platforms including iOS and Android. Emerging web standards especially as HTML5 are one possible solution; the difference between a web application and a native application is that the web applications are coded through web techniques (HTML, JavaScript, and CSS) and run in the kernel of the browser regardless of the operating system. Therefore, HTML5 based cross-platform development framework is going to be used to develop the mobile smart switch application to shorten the development period of time, which can write an application once, and deploy it to several operating systems. Cross-platform development approach allows implementing an application and its user interface using web technologies like HTML and CSS. Then the application can be built for several mobile platforms, such as iOS, Android, and Windows Phone.

3. Proposed System Architecture

The cross-platform smart switch cellphone control software is implemented using web techniques but packed as native applications. The applications are written with programming languages that the developers are familiar with and are applicable across multiple operating systems. Similar to native applications, cross-platform applications can be distributed and installed through the app store. The smart switch application can support both Android and iOS operating systems. The smart switch application can be invoked in the form of a plugin by the terminal developers. The framework of the developing middleware for the cross-platform smart switch cellphone control software is shown in Figure 1.

From the top of the framework down, the UI layer, HTML Rendering Engine layer, Wi-Fi Smart Switch Plugin layer, and Mobile OS layer are shown. The UI layer, which is the interface of the smart switch control software, is programmed with web techniques such as HTML, CSS, and JavaScript and is consistent across all smart phone platforms.

The HTML Rendering Engine layer is the key to the realization of the cross mobile OS. The HTML Rendering Engine layer is also called WebView and is actually the internal browser of the mobile smart phones. The existence of this layer is the foundation for the web application to be packed for the local OS. As it runs in WebView, we can use HTML and CSS to adjust the page layout conveniently. We extended and encapsulated WebView under the different operating

FIGURE 1: The middleware framework.

systems to a strong browser that can visit the local API of the equipment and is therefore able to visit the local API via JavaScript. In this case, we can visit the bottom API of the Wi-Fi based smart switch plugin using WebView.

The Wi-Fi Smart Switch Plugin layer is the communications function realization layer of the control software. It realizes the functions of configuring the network, connecting the smart switch, controlling the switch, time synchronization, accessing switch information, and setting timers. Those functions, programmed with native codes, must match corresponding JavaScript API functions. Thus, JavaScript API will expose the native functions to the JavaScript running in the browser, which enables the developer of the cross-platform applications to invoke the Wi-Fi smart switch control function by invoking the uniformed JavaScript interface regardless of the differences for mobile operating systems.

The Mobile OS layer is the basic API. In this control software, we used bottom API interface such as the shared network of Android and iOS, input and output.

3.1. Smart Switch Circuit Board. The smart switch utilizes the SOC (system on chip) power metering chip, which can achieve intellectual detection of the power net and the power usage of the electrical equipment such as LED energy saving lamps, air conditioners, refrigerators, and microwaves. The smart switch circuit board is shown in Figure 2.

The SOC chip is suitable for use in the home, rental house, office, and laboratory. We constructed a very brief wireless network with the CC3000MOD and MCU MSP430F5309 of TI. The CC3000MOD supports the IEEE 802.11 b/g agreement and has an IPv4 TCP/IP protocol stack in itself. The CC3000MOD can form an ideal wireless network with MCU (such as MSP430 series) that is slow, inexpensive, and power efficient. The SimpleLink proposal of TI has provided strong technical support and simplified the design of the hardware

FIGURE 2: The smart switch circuit board.

and software as well as the development of the application.

3.2. Communication Protocols. The communication is achieved using the master/slave model, which means the equipment (Wi-Fi switch) will not report information conditions or control information on its own initiative. The controlling device (cellphone) and the device controlled (switch) are in the same LAN, through which the data control frames are transferred using the UDP protocol. The control command frame can be divided into the master frame and the slave frame, and the format of each is as follows.

3.2.1. Master Frame. The master frame is the control command that can be sent to the devices. The format is as follows:

Starting byte: 68H.

Communication interface: COM.

Domain: CI.

```

Struct
{
    Unsigned char ip[4];    // IP address of the sender
    Unsigned int port;      // the listening PORT, for example, 8899
    Unsigned char Len;      // data length including the CI
    Device
}COM;

```

ALGORITHM 1

```

Struct
{
    unsigned char MAC[6];    // MAC address of the device being controlled
    unsigned int device type; // device type
}Device;

```

ALGORITHM 2

Device interface: Device.

Editable data block: Data.

Check: CS (1 byte).

End bit: 16H.

Communication Interface, COM. It includes mainly the IP address of the information sender and the listening port. Its structure is shown in Algorithm 1.

CI Is the Control Byte. For the highest bit, 0 means the frame comes from master data; 1 means it comes from slave data. The following seven bits represent the functions of the frame:

- 000 0000: search for devices.
- 100 0001: set the device switch.
- 100 0010: write the time for the timer.
- 100 0011: time synchronization.
- 111 1111: read all the data from the device.

Device. It is mainly the structure used to describe the slave device. The structure is as shown in Algorithm 2.

For this application, the device type is 0001H.

Editable Data Block. It usually includes the format, length, meaning, and unit of the data. It normally consists of data and DRH, which includes DIF, DIFE, and VIF.

3.2.2. The Slave Frame. The format of the slave frame data is the same as that of the master frame. There are certain differences in regard to the device interfaces. The slave device

data frame includes the hardware details of the slave device. Its structure is as shown in Algorithm 3.

3.3. The Realization of the Smart Switch Plugin Function. The plugin-based model is used to encapsulate the local API, including the public and the user-defined application with a unified plugin interface for the upper level to invoke the application module conveniently. The realization framework of the plugin-based smart switch function is shown in Figure 3.

Taking the device interface as an example, the public module integrates the common and general functions for invocation, which avoids repeated development and increases the efficiency for the programmer. These functions include small modules such as message receiving/sending, contacts, and camera, which can be realized by invoking the bottom API of each mobile OS. One certainty is that the invocation interfaces, realization languages, and development platforms of these modules are different on different platforms, in this case, the Android and iPhone systems. However, for the design of the user-defined module, taking the smart switch function interface as an example, the module requires repeated usage of the same function on different cellphone platforms targeting different projects to develop background. When the code volume is massive and the realization structure is complicated, the function will be encapsulated in the self-defined module so that the developer can invoke the module flexibly, decreasing the difficulty of the second development and increasing the development efficiency.

Based on the smart switch communication protocol and the SimpleLink Wi-Fi CC3000 solution provided by TI, we realized the basic functions of the smart switch, including network configuring, device connecting, switch control, time synchronization, device information receiving, and timer setting. We also encapsulated the functions of the switch

```

Struct
{
    unsigned char MAC[6]; // MAC address of the device
    unsigned int device_type; // device type
    unsigned int Manufr; // manufacturer logo, 2 bits
    unsigned char VER; // software version of the device
    unsigned char status; // status bit of the device being controlled
    unsigned char NO[6]; // BCD code of the device
    YYMMDDXXXXXX
}Device;

```

ALGORITHM 3

FIGURE 3: The realization framework.

into a plugin for the invocation of the uniformed JavaScript interface from the upper level, proving function extension for cross mobile OS. Taking the development of the Android OS as an example, the realization diagram of the smart switch function is shown in Figure 4.

It is observed from Figure 4 that the UDPPlugin is responsible for the key functions, including the following:

- (i) Obtain the current IP data from the smart switch.
- (ii) Send a set of static timer data.
- (iii) Send the information from the data switch.
- (iv) Deliver passcodes and accounts to the smart switch.

searchDevice: send out a UDP broadcast package that includes the IP address and the interface code of the mobile OS. The function will receive the UDP package sent back by the smart switch, which includes the IP address, the interface number, the MAC address, and the switch status of the smart switch. The searchDevice function will run in the background

without blocking the WebCore; other functions will block the UI path. When we click the searchDevice, the interface will notify the user that it is connecting and can be stopped by clicking. To avoid overtime, we can establish a time for UDP reception. For example, we set the timeout time to s.setTimeout(1000).

SetAC: it is to set the hotspot, broadcasting the account and passcode of Wi-Fi; invoke the TI CC3000 interface to search for available devices in the environment.

SetOF: it is to set the on/off status of the smart switch.

SetTime: it is to set the timer length of the smart switch.

BuildFrame series are mainly used to build the UDP frame to issue commands to the smart switch. The frame includes the communication interface COM, the control byte CI, and the editable data block. Each frame, including the search-device, device-info, device-time, and set-time frames, is in correspondence with its function realization.

CC3XWi-FiManager series provide the interface to receive the Wi-Fi details, including the SSID network parameters, and can be used to test whether the device has connected to Wi-Fi and to obtain the current IP and gateway address.

The CC3XACConfig series is the solution provided by TI. The series has the IPv4 TCP/IP protocol stack internally.

MyTimeutils, Utils, and DataTools are series designed for the format editing and transmission of the data under this system.

3.4. JavaScript Interface of the Smart Switch. The smart switch allows the smart OS to control the on/off status of the switch, locally or remotely, through a Wi-Fi network connection. With the wireless router and the terminal device in hand, the smart switch achieves the simple function of the remote switch in addition to other functions, such as network configuration, device connection, device switch control, device time synchronization, device information reception, and time setting. Each function was processed through the uniform interface encapsulation and under the invocation of a JavaScript interface. The JavaScript interface use the cordova.exec() function to invoke the code of the current device. When invoking the cordova.exec, the JavaScript may deliver it to the outcome processing function as well as the error process programming function and deliver a parameter

FIGURE 4: The function realization diagram of the smart switch.

```

varUDPClient=function() {
};
UDPClient.prototype.sendUDP=function(successs, fail,
action, args){
    cordova.exec(success, fail, 'UDPPlugin', action, args);
};
cordova.addConstructor(function(){
    if(!window.plugins){
        window.plugins={};
    }
    window.plugins.UDPClient=newUDPClient();
});

```

ALGORITHM 4

```

functionSwitchPlugin_sendPassword(password){
    var action="setAC";
    varargs=[password];
    window.plugins.UDPClient.sendUDP(
        function(data){
            alert(data);
        },
        Function(error){
            alert(error);
        },
        action,args);
}

```

ALGORITHM 5

set to the current device code. The invocation will also be sent to the current device name and current function name. Cordova will be responsible for communication between JavaScript and the current device (see Algorithm 4).

The smart switch plugin was named UDPPlugin. The cordova.exec accepts a success feedback function, a fail feedback function, a string reference of the current device code, a string reference of the function names to be invoked, and a data block of the parameters for delivery to the current device. Action corresponds to each functional realization, and the args is the parameter from the terminal. The definition of JavaScript interface for the smart switch function is shown in Figure 5. The following sections provide detailed illustrations of the realization of each function.

3.4.1. Configuring the Network Connection. Smart phones may choose any available Wi-Fi to connect and can send a broadcast frame to all available devices in the network through the broadcasting protocols, which includes the SSID and password of the Wi-Fi network, and the devices receive the broadcast frame and connect to the specified Wi-Fi network, and the corresponding JavaScript API is defined as sendPassword(); the variable action invokes the setAC function of the device; the variable args is the passcode entered by the client. The process is shown in Algorithm 5.

3.4.2. Connecting the Switch Device. After the smart phone and the device are connected to the same network, the smart phone terminal will send the UDP broadcast frame, which

FIGURE 5: The definition of JS API of the smart switch plugin.

includes the IP address and the code of the terminal. The device will return its own IP address and listening interface code to the smart phone. The smart phone then connects to the device. Under the `SwitchPlugin_getDevice()`, the variable `args` is an empty parameter, and variable `action` invokes the device function `searchDevice` (see Algorithm 6).

3.4.3. Control of the Device Switch. The smart phone controls the on/off status of the smart switch by sending unicast commands; the smart phone invokes the function `setOF` through the JS API `SwitchPlugin-switchState()` to control the switch (see Algorithm 7).

3.4.4. Synchronizing the Device Time. The smart phone sends a time synchronization command to the smart switch to synchronize the device time with the terminal in hand. JS API `SwitchPlugin_synTime()` is as shown in Algorithm 8.

3.4.5. Acquiring Device Information. The smart phone sends out the command to acquire the device information to

```
functionSwitchPlugin_getDevice(){
    var action="searchDevice";
    varargs=[];
    window.plugins.UDPCClient.sendUDP(
        Function(data){
            SwitchPlugin_callback_getDevice(data);
        },
        function(error){
            alert(error);
        },
        action,args);
}
```

ALGORITHM 6

the smart switch, and the device sends back the information regarding the on timer or off timer. The definition of

```
functionSwitchPlugin_switchState(){
    var action="setOF";
    varargs=[""];
    window.plugins.UDPCClient.sendUDP(
        function(data){
            SwitchPlugin_callback_switchState(data);
        },
        function(error) {
            alert(error);
        },
        action,args);
}
```

ALGORITHM 7

```
functionSwitchPlugin_getDeviceInfo(){
    var action="getDeviceInfo";
    varargs=[""];
    window.plugins.UDPCClient.sendUDP(
        function(data){
            SwitchPlugin_callback_getDeviceInfo(data)
        },
        function(error){
            alert(error);
        },
        action,args);
}
```

ALGORITHM 9

```
functionSwitchPlugin_synTime(){
    var action="synTime";
    varargs=[""];
    window.plugins.UDPCClient.sendUDP(
        function(data){
            SwitchPlugin_callback_synTime(data);
        },
        function(error){
            alert(error);
        },
        action,args);
}
```

ALGORITHM 8

the corresponding JS API SwitchPlugin_getDeviceInfo() is as shown in Algorithm 9.

3.5. The Interactive Process of the Smart Switch Plugin. The smart switch plugin achieved cross-platform isolation of the client interface to the Java Code of the Android OS and the Objective-C code of the iPhone OS using the HTML Rendering Engine (WebView) between the local code and browser and achieved data communication between the browser interface and the network socket interface. WebChromeClient provides an onJsPrompt method, which will be invoked when invoking the prompt function. We changed the onJsPrompt function to the interface of the Android exposed to the web. Therefore, every time the web attempts to invoke any function from the Android, the prompt will be invoked. The WeonJsPrompt would then analyze parameters to process the following actions. The plugin is then invoked to satisfy the demand from the web. The interactive process of the smart switch plugin is shown in Figure 6.

It is observed from Figure 6, the smart switch plugin initiates the commands in the JavaScript interface, invoking the JS API of each function module of the smart switch plugin. The JS API will invoke the JavaScript prompt function and trigger the onJsPrompt listen device in the DroidGap package (WebView). The PluginManager will then

be assigned to handle the process. The onJsPrompt method is in the GapClient series, which is inherited from and realized by the WebChromeClient series of the WebView. The JavaScript interfaces send the communication information and JavaScript action commands to the plugin manager, which is responsible for arranging function implementation, such as connecting the switch device and setting the timer.

After the implementation of the sentence Stringr = pluginManager.exec (service, action, callbackId, message, async) in onJsPrompt, the plugin manager PluginManager will distribute the commands to the smart switch plugin for implementation according to the parameters received. When the commands are distributed and the tasks of the smart switch plugin are finished, the system will send the upper level JavaScript feedback about the implementation results. There are two potential situations. In the case of synchronous invocation, that is, a simple invocation of a function without other operations in different sequence, the last step of the upper sequence diagram does not exist; there is no invocation of the CallbackServer; and the results of the plugin implementation will be sent back to JS through JsPromptResult.confirm and to the clients thereafter. If the invocation is nonsynchronous, the last step of the upper sequence diagram will be used and will involve the CallbackServer, and the CallbackServer will use the XHRServer, which is realized in the client terminal, and, in each instance after the nonsynchronous invocation, the results of the implementation outcome in the background will be saved in the link table of the CallbackServer. To be specific, the link table is the privateLinkedList<String>javascript in CallbackServer; the result is a string representing the JS function invocations. The behavior of the XHRServer is very simple: once there is a request, take the earliest data in the link table to the client terminal for further use; if there is no request, send back an empty reply every 10 seconds to maintain XHRServer.

4. Application Scenarios

We connect the smart phone and Wi-Fi smart switch in the same Wi-Fi network in order to utilize the smart phone to

FIGURE 6: The sequence diagram of the interactive process.

FIGURE 7: The smart switch system working scheme diagram.

control the switch and the timer remotely. We use the AES-128 to send encrypted information or controlling commands directly. The smart switch system working scheme is diagrammed in Figure 7.

The workflow of the interaction between the smart phone and the Wi-Fi smart switch is shown below:

- (1) The mobile client connects to Wi-Fi.
- (2) The mobile client can then acquire the gateway address of the Wi-Fi based on the connecting information.
- (3) The mobile client will calculate the broadcasting IP address based on the IP address, gateway address, subnet mask, and other information and will acquire the SSID and password of Wi-Fi from the user interface. By using the broadcasting protocol of TI CC3000, the client can broadcast the device character

information and the AES-128 encrypted SSID and password to the subnetwork.

- (4) The device that needs to connect to the Wi-Fi is manually set to debug mode. If the device has the same device character information, after receiving broadcasting frames, it can obtain the SSID and password and connect to the designated Wi-Fi.
- (5) After the device connects to the Wi-Fi, the mobile client attempts to broadcast UDP connection request frames via the LAN broadcast protocol. The frames carry the IP address and port number of the mobile client. Connection failure will occur in the event of connection time-out.
- (6) After receiving the broadcasting connection request frames, the device can resolve the IP address and port number of the mobile client and resend its own

FIGURE 8: Smart switch software control interface.

FIGURE 9: Actual test scenarios.

IP address and port number to the client via UDP frames.

- (7) The mobile client receives the connection confirmation frames sent by the device and can then resolve and record the IP address of the device in order to send control frames.
- (8) When the user sends a control order from the mobile client, such as controlling the device switch, synchronizing the device clock, setting the timer, or acquiring device information, the mobile client will construct comparable UDP frames and send them to the IP address and port number of the device.

The smart switch application is installed on the smart phone. The Android and iOS operating system share the same software control interface, as shown in Figures 8(a) and 8(b).

In this scenario, we use a night light as the electric appliance. The control command is sent by the smart phone. The final results are shown in Figure 9. The left picture shows that the night light is off, and the right picture shows turning on the switch.

5. Performance Analysis

First, we tested whether the functionalities for control actions of the smart switch are correct. Figure 10 shows the success rates of different functions of the smart switch.

FIGURE 10: Success rates of different functions of the smart switch.

From Figure 10, we can see that the success rates for set the passcode, set on/off, acquire information, and set the timer functions are below 100%, and because some experiments are done in the far distance, and there is a disconnection between the devices and the wireless access point. However, the system performance meets the requirements and can perform the corresponding functions in a right way.

FIGURE 11: Responding time of smart switch functions.

In the second experiment, we tested the sensibility of the control commands, which means the response time needed by the device to perform the corresponding function actions after a control command on the mobile control client is executed. Functions such as connecting to the device, setting the switch, synchronizing the clock, acquiring information, setting the timer, and others all require sending, receiving, and resolving data packages.

Different functions require different data bits, frames have different lengths, and the complexity of resolving data package information is different; therefore, the communication time delays of processing UDP are different. UDP communication and data resolving create processing time delays from the moment of creation of the DatagramSocket. After the processes of framing, creating data packages, sending data packages, and receiving data packages sent by the device, the time delay test will yield the moment at which useful data are resolved.

Insert the *Log.d* before creating DatagramSocket and after resolving data packages. Thereafter, we can send different function commands, export debugging information at the corresponding moment to *Logcat*, and calculate time delays. The response time of smart switch functions is shown in Figure 11.

Figure 11 shows that the time delays of all UDP communication and data resolving functions of the smart switch are less than 200 ms, indicating that the requirements are met.

In the third experiment, we focus on the time for a smart switch application to be developed and deployed, which can be used to measure the time to market for business goal. Here, we compare the development time for smart switch with the same UI on the Android and iOS smart phone using the “native” development and “cross-platform” development framework method, respectively, with one skilled programmer, and the development time is counted at 9:00 a.m. to 17:00 for each working day, respectively. Figure 12 shows the comparison result.

As is observed from Figure 12, we can see that the average development time for traditional native development

FIGURE 12: Comparison for development time to market.

approach is double more than the cross-platform approach in our practical experience. Because different operating systems have different SDKs with native development approach separately, however, cross-platform based approach can develop the application only once and deploy it to several operating systems, and such works can always be done in hours or couple of days.

6. Conclusion and Future Work

In this paper, we propose a cross-platform smart switch control software development framework for mobile smart phone, adopt the plugin technology to encapsulate the public and user-defined API, realize the basic functions of the smart switch, define JavaScript interface for smart switch, such as network configuration, device connection, device switch control, device time synchronization, device information reception, and time setting, also develop the smart switch, utilize the smart phone to control the switch and the timer remotely, and prove the possibility of developing cross-platform smart switch cellphone controlling software. Finally, the scenarios are illustrated, and the performance tests are analyzed. In the future work, we will focus on the networking of home appliances and the security of mobile control software in a digital home.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (no. 61132001); Program for New Century Excellent Talents in University (no. NCET-11-0592); National High-tech R&D Program of China (863 Program) (no. 2013AA102301); and Project of New Generation Broadband Wireless Network under Grant no. 2014ZX03006003.

References

- [1] M. Chan, D. Estève, C. Escriba, and E. Campo, "A review of smart homes—present state and future challenges," *Computer Methods and Programs in Biomedicine*, vol. 91, no. 1, pp. 55–81, 2008.
- [2] H. Wang, J. Saboune, and A. El Saddik, "Control your smart home with an autonomously mobile smartphone," in *Proceedings of the IEEE International Conference on Multimedia and Expo Workshops (ICMEW '13)*, pp. 1–6, IEEE, San Jose, Calif, USA, July 2013.
- [3] N. Komninos, E. Philippou, and A. Pitsillides, "Survey in smart grid and smart home security: issues, challenges and countermeasures," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1933–1954, 2014.
- [4] M. R. Alam, M. B. I. Reaz, and M. A. M. Ali, "A review of smart homes—past, present, and future," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 42, no. 6, pp. 1190–1203, 2012.
- [5] F. Guodong, Y. Yuebin, G. Xuemei, and W. Guoli, "A smart fiber floor for indoor target localization," *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 52–59, 2015.
- [6] R. Ferfús, O. Sallent, J. Pérez-Romero, and R. Agustí, "A solution framework to provide management services for wireless communications in the digital home," *IEEE Communications Magazine*, vol. 50, no. 11, pp. 132–141, 2012.
- [7] B. Mrazovac, M. Z. Bjelica, N. Teslic, and I. Papp, "Towards ubiquitous smart outlets for safety and energetic efficiency of home electric appliances," in *Proceedings of the 1st IEEE International Conference on Consumer Electronics (ICCE '11)*, pp. 322–326, Berlin, Germany, September 2011.
- [8] E. K. Hansen, G. G. H. Olesen, and M. Mullins, "Home smart home: a danish energy-positive home designed with daylight," *Proceedings of the IEEE*, vol. 101, no. 11, pp. 2436–2449, 2013.
- [9] F. Viani, F. Robol, A. Polo, P. Rocca, G. Oliveri, and A. Massa, "Wireless architectures for heterogeneous sensing in smart home applications: concepts and real implementation," *Proceedings of the IEEE*, vol. 101, no. 11, pp. 2381–2396, 2013.
- [10] S. Makonin, L. Bartram, and F. Popowich, "A smarter smart home: case studies of ambient intelligence," *IEEE Pervasive Computing*, vol. 12, no. 1, pp. 58–66, 2013.
- [11] N. Saito, "Ecological home network: an overview," *Proceedings of the IEEE*, vol. 101, no. 11, pp. 2428–2435, 2013.
- [12] R. Piyare and S. R. Lee, "Smart home-control and monitoring system using smart phone," in *Proceedings of the 1st International Conference on Convergence and Its Application*, pp. 83–86, 2013.
- [13] R. A. Ramlee, M. A. Othman, M. H. Leong, M. M. Ismail, and S. S. S. Ranjit, "Smart home system using android application," in *Proceedings of the International Conference of Information and Communication Technology (ICoICT '13)*, pp. 277–280, IEEE, Bandung, Indonesia, March 2013.
- [14] R. Oliveira, G. Pontes, J. Machado, and A. Abelha, "Analysis of cross-platform development frameworks for a smartphone pediatric application," in *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management (IEEM '13)*, pp. 1057–1061, December 2013.
- [15] J. Ohrt and V. Turau, "Cross-platform development tools for smartphone applications," *Computer*, vol. 45, no. 9, pp. 72–79, 2012.
- [16] M. Palmieri, I. Singh, and A. Cicchetti, "Comparison of cross-platform mobile development tools," in *Proceedings of the 16th International Conference on Intelligence in Next Generation Networks (ICIN '12)*, pp. 179–186, October 2012.
- [17] T.-M. Grønli, J. Hansen, G. Ghinea, and M. Younas, "Mobile application platform heterogeneity: android vs windows phone vs iOS vs Firefox OS," in *Proceedings of the 28th IEEE International Conference on Advanced Information Networking and Applications (AINA '14)*, pp. 635–641, May 2014.
- [18] M. Palmieri, I. Singh, and A. Cicchetti, "Comparison of cross-platform mobile development tools," in *Proceedings of the 16th International Conference on Intelligence in Next Generation Networks (ICIN '12)*, pp. 179–186, Berlin, Germany, October 2012.
- [19] I. Dalmaso, S. K. Datta, C. Bonnet, and N. Nikaein, "Survey, comparison and evaluation of cross platform mobile application development tools," in *Proceedings of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC '13)*, pp. 323–328, July 2013.
- [20] M. H. Ng, B. C. Liu, W. Y. Ting, and H. H. Mohamed, "Cross-platform mobile applications for android and iOS," in *Proceedings of the Wireless and Mobile Networking Conference*, pp. 1–4, 2013.
- [21] A. Puder and I. Yoon, "Smartphone cross-compilation framework for multiplayer online games," in *Proceedings of the 2nd International Conference on Mobile, Hybrid, and On-Line Learning (ELML '10)*, pp. 87–92, IEEE, February 2010.
- [22] C. Best, "Seregon, solutions announces support for blackberry playbook," 2011, <http://www.prweb.com/releases/DragonRAD/PlayBook/prweb4960674.htm>.
- [23] RhoMobile, "What is so special about the rhodes smartphone app framework?" 2010, <http://rhomobile.com/blog/whats-so-special-about-therhodes-smartphone-app-framework>.
- [24] PhoneGap, 2012, <http://phonegap.com>.