Android Application for Accessing KNX Devices via IP Connection

J. A. Nazabal, J. Gómez, F. Falcone, C. Fernández-Valdivielso,
P. E. Branchi and I. R. Matías

*Electrical and Electronic Engineering Department, Public University of Navarra,
Campus de Arrosadia 31006, Pamplona, Spain
juanantonio.nazabal@unavarra.es*

Abstract

In this work, an Android application programmed in Java for accessing KNX devices via IP packets using KNXNet/IP protocol is presented. The work consists of a custom programmed Java library that implements the client part of the KNXNet/IP protocol and an application that uses it for graphically controlling KNX devices.

Keywords: *KNX, Smartphone, Android, KNXNet/IP, Smart Homes, Home Automation*

1. Introduction

Nowadays we live in an interconnected society and given this fact, it is easy to see how technology is moving increasingly towards that environment. Internet has become part of our daily routine and IP protocol has become a global standard used for different networking technologies and purposes all over the world. Likewise, the KNX standard has followed this trend and different products have been launched to the global market devoted to the integration with the KNXNet/IP protocol [1, 2] and the home environment. The presence of home automation and environmental control systems in homes and in public buildings has been increasingly common in these last years. These devices enable to perform usual actions within the home environment in a more comfortable and protected way, and can dramatically improve quality of life. These systems are capable of automating a home through energy management, safety, welfare and communication elements allowing a more efficient energy use and therefore contributing to sustainable development of society.

In a similar manner as with home automation, mobile terminals [3, 5] have quickly become an indispensable element in our daily lives, leading to rapid changes in industry. Smartphones/PC Tablets have allowed a large number of different applications, increasingly powerful and more affordable. Additionally, a wide variety of networking solutions are available, being lighter and with greater autonomy. Therefore, developing a touch interface for a Smartphone can be a good solution for many people allowing greater independence, improving their social integration and professional development. Nowadays there are many different Smartphones/PC Tablets providers, with Android based systems experimenting strong adoption.

Due to the previous considerations, the aim of this project was the integration of a touch interface in a home environment by using the KNXNet/IP protocol over an Android operating system.

2. Description

The work that has been realized will be presented in the next four subsections, devoted to KNX, Android OS, Java Libraries and the resulting Android Application.

2.1. KNX

KNX technology is a worldwide standard for home and building control with different communications media. KNX devices are provided by a large number of manufacturers, offering more functionality within the home automation system. The underlying standardization provides assurance that different manufacturers' products may be connected together within the overall system. Therefore, there are no limitations to a single manufacturer and if a manufacturer ceases to trade or offer a particular product, the same or similar products are available from other manufacturers.

Bus devices can either be sensors or actuators needed for the control of building management equipment such as security systems, energy management, or heating and climate, just to mention a few. It is worth noting that KNX supports several communication media like twisted pair, Power Line, Radio frequency and IP/Ethernet and can be coupled to other systems like building automation systems, telephone networks, multimedia networks, etc… via “Ad Hoc” gateways.

2.2. Android Programming Environment

The Android Operating System is based on Linux. Over the kernel, the different C/C++ libraries used by components of the system and the Android runtime, with the core libraries and the Dalvik Virtual Machine can be found. Finally, and on top of the architecture, the Application Framework is located. It consist of a series of APIs (Application Programming Interface) used for developers for programming their own code. The native programming language for Android is Java but there is also the possibility of using other different languages like visual, C#, C, etc.

In order to develop applications it is mandatory to employ the Android SDK (Software Development Kit) and in order to use it with the Java language, an IDE (Integrated Development Environment) like Netbeans or Eclipse is used. Finally, the ADT plugin (Android Development Tools) must be installed and properly configured in the chosen IDE. It comes with an emulator for testing the developed applications before installing them. The result of compiling the java code of the application will be a file with the APK extension (android File Format) used to distribute and install application software and middleware onto Android operating system.

2.3. JKNXNetIP: Java KNXNet/IP Library

This library, programmed in Java, is an implementation of the client part of the KNXNet/IP tunneling protocol, developed for accessing KNX devices using UDP datagrams or TCP segments over IP packets (see Figure 1). The use of UDP protocol is mandatory in all the KNXNet/IP certified devices. However, the use of TCP is optional, which is the reason why the library has been programmed for using UDP datagrams. It consists in the encapsulation of cEMI (common EMI) frames within IP packets. The cEMI message format is a generic structure for medium independent KNX messages for information transportation and is not dependent on the frame structures of the different KNX media. For the development of this library, the access to the source code of the Calimero [3] library developed by the Vienna Technical University has been really useful and has been taken as a model.

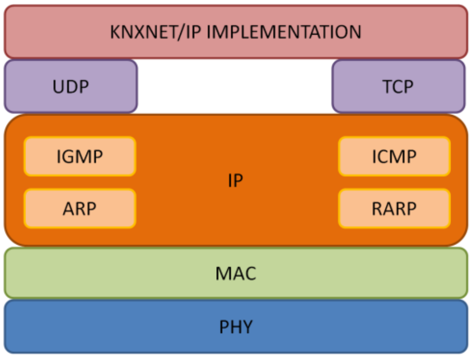


Figure 1. Layers Architecture

The KNXNet/IP protocol is based on client/server architecture as shown in Figure 2. Usually the server part is implemented in a standalone KNX device called IP Interface. This device is connected to the KNX bus and to an IP network also, usually via an Ethernet interface using a RJ-45 connector. This is the configuration of the device that is used in this work, an IP Interface model N148/21 by Siemens.

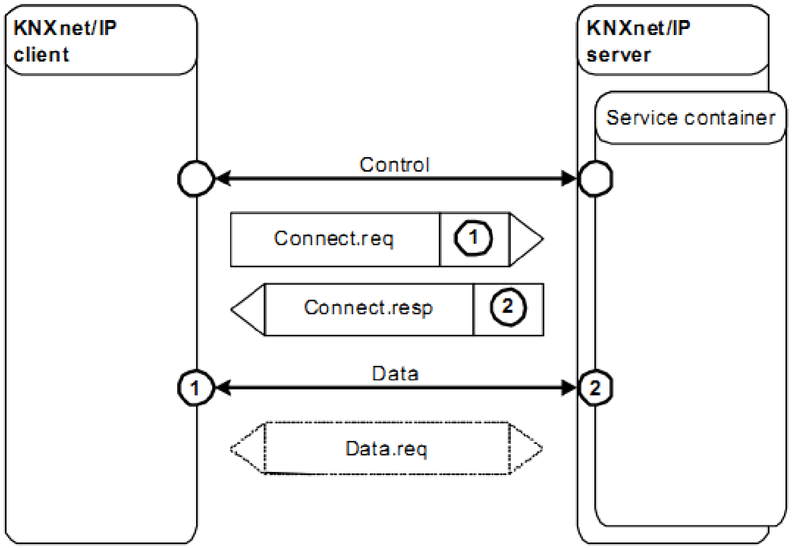


Figure 2. Client / Server Architecture [1]

For establishing the connection with the KNXNet/IP server, first a specific connection request frame must be sent. When the server receives this frame, it sends a connection response specific frame indicating the status of the connection and the communication channel associated that will be used the rest of the communication. The binary format of this frame is shown in Figure 3. As it can be seen in the figure, the client IP address and port number are included in the different HPAI (Host Protocol Address Information) structures and it will be the last time in all the connection that the server will ask for the client's IP address. The rest of the time the server will use the IP address associated to the communication channel obtained in the connection establishment. This issue is a relevant problem if the client ́s IP address changes in the connection time because the server will not notice it and will send the frames to an incorrect IP address.

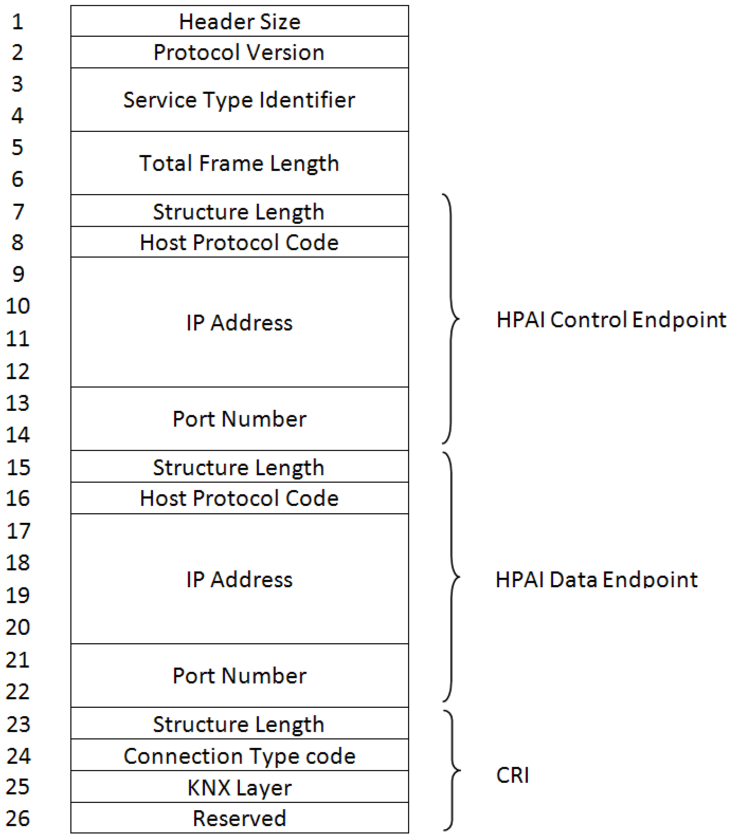


Figure 3. CONNECT_REQUEST Frame Binary Format

Another issue to take into account is to obtain the Smartphone ́s IP address. Usually the device is connected to Internet through a NAT (Network Address Translation) and retrieving its IP may prove difficult. One solution may be to send an http request to a webpage where the IP is display somewhere within the page, and retrieve it using regular expressions. When KNXnet/IP communication has to traverse across network routers using KNXnet/IP Client shall set the value of the IP address and the port number in the HPAI to zero to indicate NAT traversal to the receiving KNXnet/IP Server. This procedure will replace them with the IP package received info.

In Figure 4 is shown the scheme used in this work for requesting data to the KNX sensors using the JKNXNetIP library. The boxes with green background colour indicate objects and methods already implemented by the JKNXNetIP library and the ones with orange background, interfaces and method that must be implemented by the user of the library.

The first step is to implement the KNXInterface interface, putting custom code in both of its methods. Afterwards a KNXNetIPConnection object using the implemented interface as an input parameter has to be created. The next step consists of calling KNXNetIPConnection’s Connect method for sending a CONNECT_REQUEST frame to the KNXNet/IP server for establishing the connection. Finally, KNXNetIPConnection’s method SendCEMIFrame will send a request to a KNX sensor and this sensor will send a response, activating

KNXInterface's KNXNetIPDatagram method and executing implemented custom code that has been placed there in the implementation of the KNXInterface.

For disconnecting, KNXNetIPConnection's Disconnect method is used. It sends a DISCONNECT_REQUEST frame to the KNXNet/IP server, responding with a DISCONNECT_RESPONSE frame and then the KNXInterface's DisconnectRequest method is called with the result of the disconnect process.

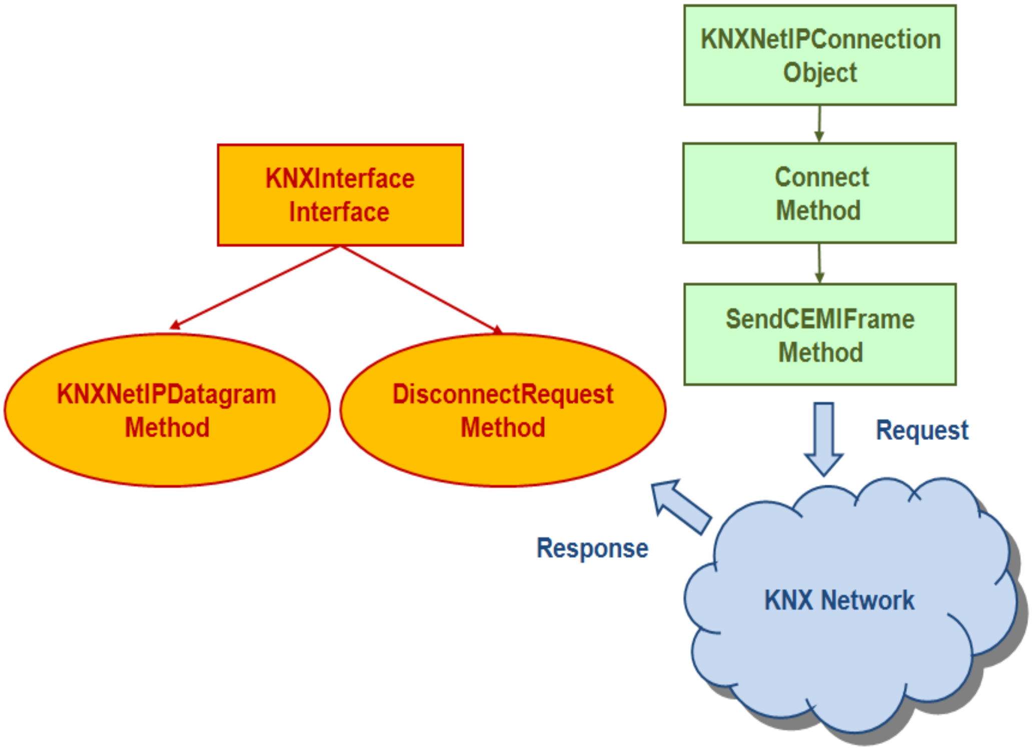


Figure 4. JKNXNetIP Library Operating Scheme

2.4. Android Application

The developed application, programmed for Android [6-9] is a user friendly graphical environment for interacting with KNX devices in a home installation using the JKNXNetIP library. The first step for using this application is to define the different rooms in the home and the KNX devices placed in each of them. In Figure 5(a) is shown the dialog box for adding, editing and removing the different home's rooms. Figure 5(b) depicts a screenshot of the dialog box used for managing and configuring the KNX devices present in each room. The application has been tested using a small KNX network installed in one of the Public University of Navarre's laboratories. The application has been configured for that specific scenario and as can be seen in Figure 5(b), the network only has installed a bulb, a blind and a thermostat.

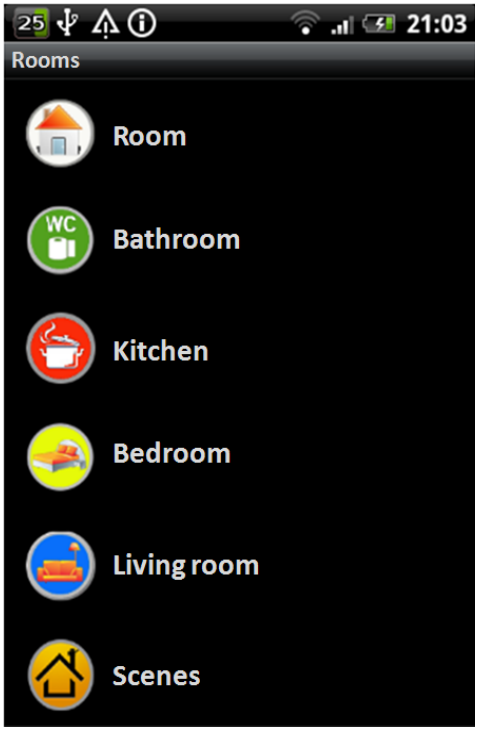


Figure 5. (a) Dialog Box for Managing Rooms

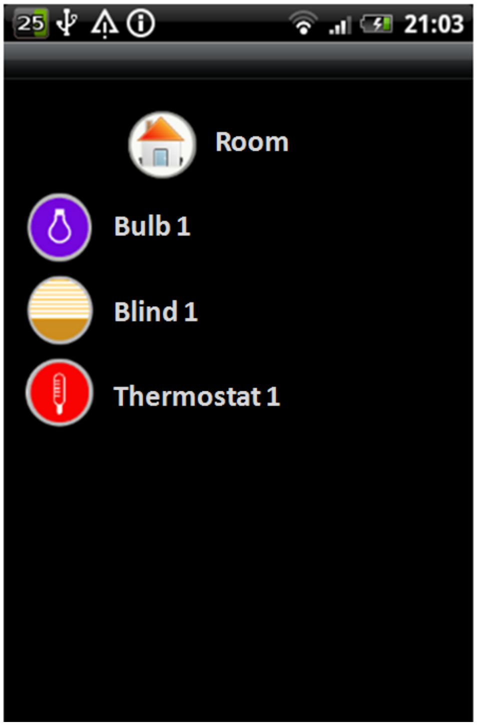
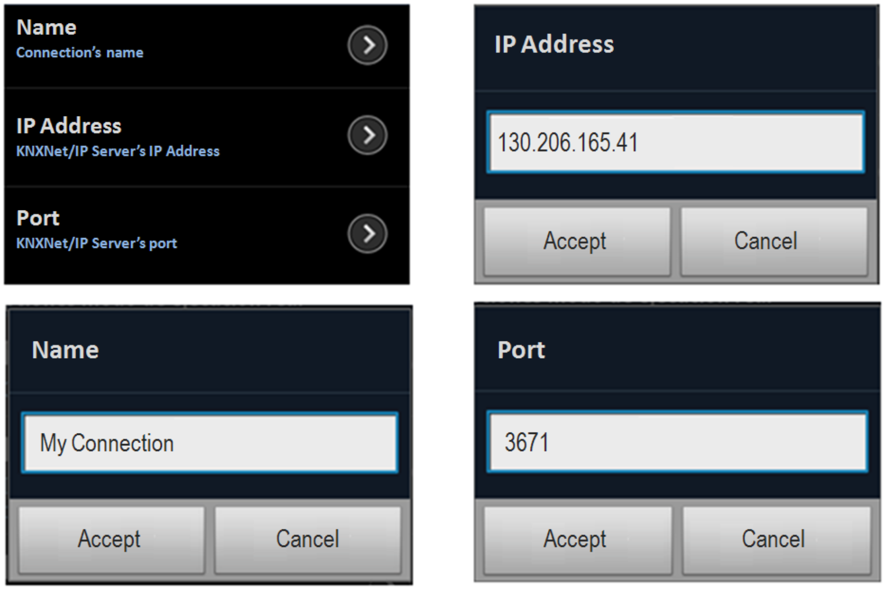


Figure 5. (b) Dialog Box for Managing Devices

The next step for using the application is to configure the connection. For connecting with the IP interface, the first step is to know its IP address and the used port. As shown in the different screenshots depicted Figure 6, the application has different dialog boxes for managing the connections and configuring the name, IP address and the port used.



IP Address: 130.206.165.41

Name: My Connection

Port: 3671

Figure 6. Managing and Configuring Connections

Finally, once the different rooms and KNX associated devices have been successfully configured and the connection has been made, the devices are available for interacting with them. There are different dialogs for the different types of devices considered in this application. For example, Figure 7 shows the dialog used for controlling the status of an electric bulb. The green and red buttons are used for turning On and Off the bulb and the status is shown by the bulb-shaped icon.

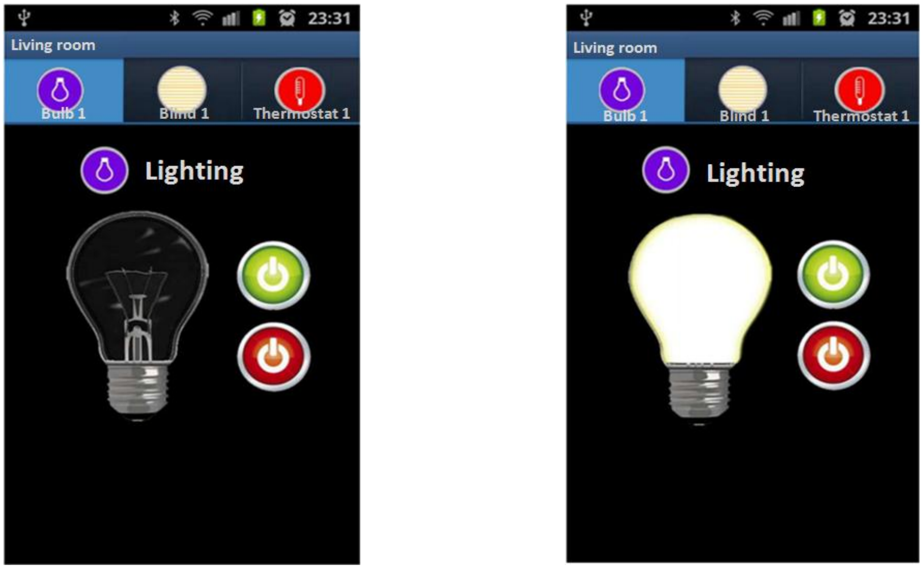


Figure 7. Managing Status of a Bulb

3. Conclusions

To our knowledge, this is the first time that an Android application for accessing KNX devices is presented. Basically, it consists of a custom programmed Java library using KNXNet/IP protocol. This is a first step to advance in the development of applications in home automation system. In this case, it is applied to a worldwide standard for home and building control, which is KNX.

Due the nature of the KNXNet/IP protocol, both the KNXNet/IP client and server IP addresses must remain static all the connection time. This is a relevant issue when using Internet 3G or GPRS connections, typical of Smartphones, due to the fact that the device’s IP address changes every few seconds, and this time period depends only of the ISP provider. When using Smartphone’s Wi-Fi connection instead, the same problem remains but it is less important, because the IP address changes within a matter of hours or even in days, depending also of the ISP provider. So, using Wi-Fi and for connection times within a matter of minutes, the application works properly without any problems.

The JKNXNetIP library implemented is an important initial step that needs to improve its usability and also to add the capability of using both UDP and TCP protocols. An interesting enhancement could be the translation of the library to other programming languages like C++.

Finally, in this work a specific scenario has been implemented in order to show the viability of the application: This application uses static structures for managing the different rooms and devices and another enhancement could be to replace them with dynamic structures, giving the application more flexibility, issues in which we are working at the present moment.

Acknowledgements

This work was supported by the Spanish Economy and Competitivity Ministry AIB2010NZ-00328.

References

[1] European standard EN 13321-2:2006, "Open Data Communication in Building Automation, Controls and Building Management-Home and Building Electronic Systems - Part 2: KNXnet/IP Communication", **(2006)**.

[2] European standard EN 50090, "Home and Building Electronic Systems", **(2005)**.

[3] R. Shahriyar, E. Hoque, S.M. Sohan, I. Naim, M. Akbar and M. Khan, "Remote Controlling of Home Appliances using Mobile Telephony", International Journal of Smart Home, vol. 2, no. 3, **(2008)** July, pp. 37-54.

[4] L. Zoref, D. Bregman and D. Dori, "Networking Mobile Devices and Computers in an Intelligent Home", International Journal of Smart Home, vol. 3, no. 4, **(2009)** October, pp. 15-22.

[5] R. D. Caytiles, B. J. Park, "Mobile IP-Based Architecture for Smart Homes", International Journal of Smart Home, vol. 6, no. 1, **(2012)** January, pp. 29-36.

[6] https://www.auto.tuwien.ac.at/a-lab/calimero.html.

[7] J. F. DiMarzio, "Android A Programmer's Guide", McGrawHill, United States of America, **(2008)**.

[8] R. Rogers, "Android Application Development", O'Reilly Series, United States of America, **(2009)**.

[9] R. Rogers, "Professional Android Application Development", Wrox, Canada, **(2009)**.