Freescale Semiconductor
Wireless Connectivity Community content

Document Revision: 1.0
Date: 03/2015

Connected Home Gateway for the Internet of Things (IoT) using Kinetis MCU Family and MQX RTOS

1 Introduction

This document describes the implementation of the so-called Connected Home Gateway for the Internet of Things (IoT) and its controller implemented in a Smart device (tablet) running Android OS. This system features a powerful yet low cost microcontroller (MCU) as a wireless sensor platform (ZigBee Protocol) which could be accessed via a smart device application through a commercial wireless router. This allows users to control home appliances wirelessly such as lights, switches, thermostats, etc. without needing ZigBee protocol implemented.

This solution is oriented to be used by any person with a smart device and with access to the router's wireless network in order to monitor and control the house appliances with a simple touch and without external devices or adapters. Furthermore, the MCU capabilities allow a future implementation of a second sensor network (i.e. Smart Energy, RF4CE, etc.) without adding any external hardware and with minimum efforts.

Contents

2 System Approach

The following sections describe the Connected Home Gateway system from the beginning of its meaning to the conception of it. It also covers a small introduction to the protocols used and building blocks forming the gateway.

2.1 Antecedents

Before going any further with system implementation details, it is important to mention the background highlights of this system design and the reasons of why it was decided to be done in such way.

First of all, implementing a gateway between Wi-Fi and ZigBee sensor networks might have been not that relevant few years ago. However, the recent tendency to have devices connected to each other has led to similar designs with the aim of providing a single solution to fulfill a multi-protocol solution needs. This is where the concept of Internet of Things comes to discussion.

2.1.1 Internet of Things (IoT)

Nowadays, with the different emerging technologies, things are tending to be connected between each other creating a massive network. Instead of considering a personal computer, a cell phone or even a house light as isolated appliances, now we can think of them as part of a same system, where they can communicate with each other in order to provide new services.

A good example would be a smart watch that monitors a person's heart rate and uploads all this information to a database in the *cloud*. This database could be monitored by specialists and therefore notify users if they might have a health problem or simply recommend them to change habits in order to avoid them.

Another example could be a smart refrigerator, which is tracking of expiration dates of food so it could inform the users. It could also have some statistics of food purchased lately and automatically order items from online grocery stores when they become scarce. Additionally, it could display any sort of offers that users might be interested in, etc.

In general, the Internet of Things is a concept of not only having everything connected to the same network, but also to use this data in order to provide new services. Some of the uses cases would be:

- Smart Connectivity
- Personalized Marketing
- Statistical behavior
- New Patterns search triggered by relevant events
- Others we can imagine and others we will discover

2.1.2 Building Blocks of the IoT

Several building IoT blocks could be inferred from the information mentioned before, from the edge nodes such as sensors or actuators to the big data servers storing and analyzing all this information.

The Smart Home Gateway implementation described in this document focuses just in both the user's front end (smart device application) and the gateway itself (multi-protocol system). There are obviously several end nodes used for demonstration purposes such as smart light bulbs, power outlets or emulated appliances, but those are not part of the gateway implementation and could be obtained in large stock stores.

Next figure illustrates the IoT building blocks and those covered in this document.

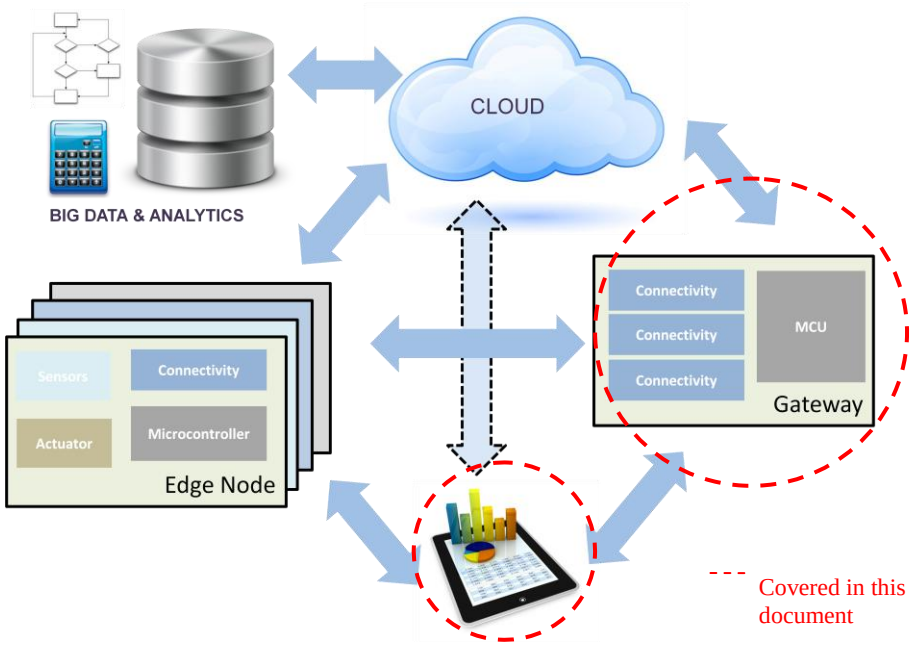


Figure 1. Internet of Things building Blocks and gateway coverage

As figure illustrates, there are several connectivity blocks in the gateway that basically translates to different protocols used to communicate with the different edge nodes. Such protocols include Bluetooth, WiFi, Ethernet, CAN, UART, ZigBee and many others.

While the gateway could have as many as the developer would like to include, this specific implementation focuses in ZigBee protocol, that due to its nature of low cost and low power solution for wireless sensor networks, it has been chosen to be used in several wireless-enabled home appliances.

2.1.3 ZigBee Protocol Overview

ZigBee is a low-cost, low-power, wireless mesh network standard global protocol based on IEEE 802.15.4 standard. It defines the network, security and application software layers and was created to be used specially in personal area networks such as wireless sensor networks.

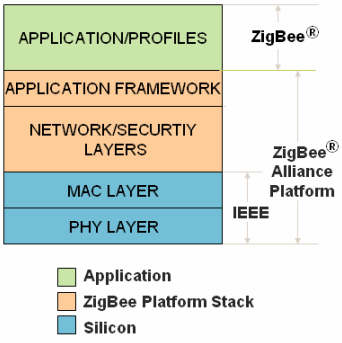


Figure 2. ZigBee Specification Software layers

ZigBee is used in applications that require a low data rate, long battery life, and secure networking. It meets key market needs such as reliability, scalability, security, long battery life, interoperability and low cost.

It operates in the industrial, scientific and medical (ISM) radio bands and 2.4 GHz in most jurisdictions worldwide. Data transmission rates vary from 20 kilobits/second in the 868 MHz frequency band to 250 kilobits/second in the 2.4 GHz frequency band, which makes it best suited for periodic or intermittent data or a single signal transmission from a sensor or input device.

The following figure illustrates a comparison between wireless technologies according to its data rate and range.

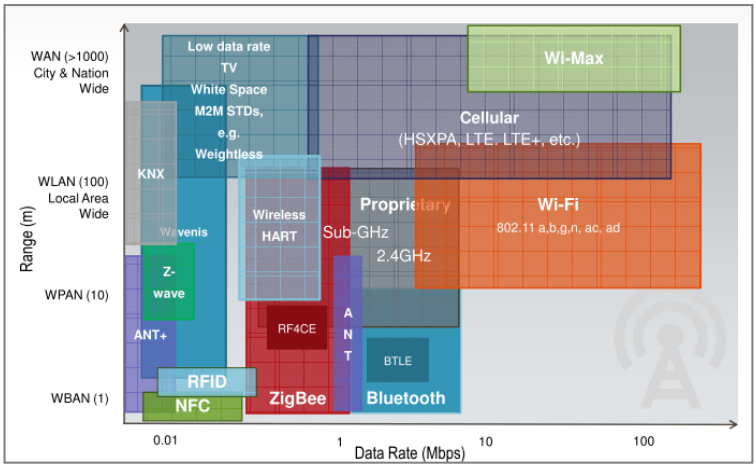


Figure 3. Wireless Technologies Comparison

In comparison to other wireless technologies, ZigBee is intended to be simpler and less expensive solution while keeping the low power consumption as priority. Applications may range from single light bulbs and switches, to complex smart meters.

NOTE

For more information about ZigBee Specification please go to www.zigbee.org or refer to the ZigBee Specification document.

2.2 System Overview

The Connected Home Gateway System, in few words, consists in few electronic boards and devices serially interfaced that in conjunction are used as a communication bridge between the Internet and a ZigBee Home Area Sensor network.

The core of the gateway System is a powerful 32-bit microcontroller (MCU) with Ethernet port communicating with a wireless router via a Wi-Fi transceiver. A second board is used to handle all ZigBee-related communications and it is controlled via serial commands. Additionally, the system includes an Android application to control the house appliances through a simple yet useful graphical interface whenever the local network is available.

Figure 4 provides a scenario of application where the main and secondary boards comprise the so-called Connected Home Gateway with an Android application to allow users control the house appliances over a smart device. Note that the gateway also has the capabilities to integrate a web-based user interface, but this is not covered by this phase of implementation.

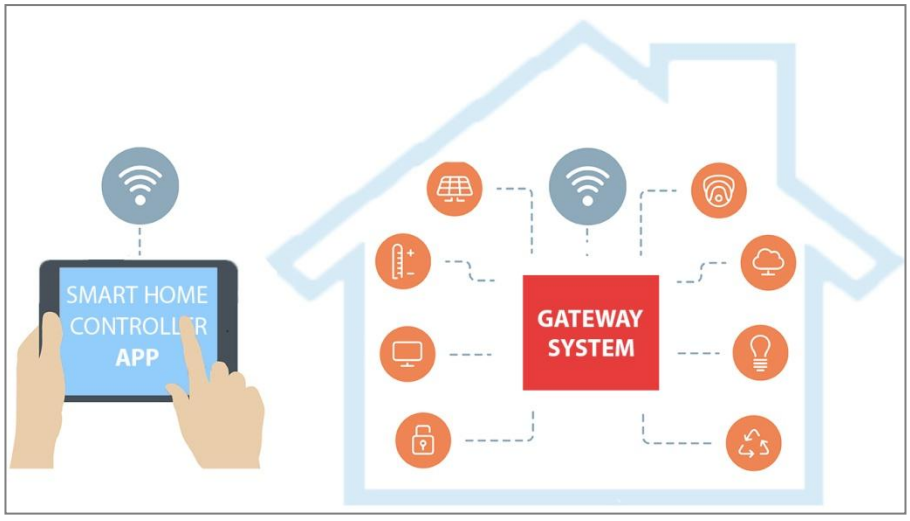


Figure 4. Connected Home Gateway System Overview

2.3 Objectives

The Connected Home Gateway system is intended to serve as a communication bridge between WiFi/Ethernet and ZigBee Protocol. This would make every ZigBee-enabled device accessible and controllable form any smart device with Wi-Fi capabilities such as a smart home or tablet. This will remove the need of having a ZigBee transceiver in every mobile device attempting to control the house appliances. In general, users will be able to:

- Remote control of Home Appliances using ZigBee protocol
- Any WiFi-enabled device could control the appliances without a ZigBee transceiver
- Achieve bi-directional communication between users and appliances

3 System Implementation

Real system implementation would require a powerful secondary MCU to host the whole ZigBee stack. The main MCU (MK60N512VMD100) is enough for the task. Also, a Wi-Fi transceiver will be used in order not to require an Ethernet connection, in case the system is not directly next to a router. Anyway, Ethernet connection could still be included if desired.

Due to its performance and peripherals available, the KW24D512 device was selected to act as host the ZigBee stack. This device was convenient since it was provided by Freescale by this specific demo, and there is a board in Tower-factor that allows using a single pair of tower elevators. Also, the TWR-WiFi-AR4100 was used as WiFi transceiver.

NOTE

For further details about KW2x devices and the development kits available, please visit http://www.freescale.com/kw2x

3.1 Block diagram

Figure 5 shows a block diagram of the actual system with the new selected boards including the end nodes such as light bulbs and development boards.

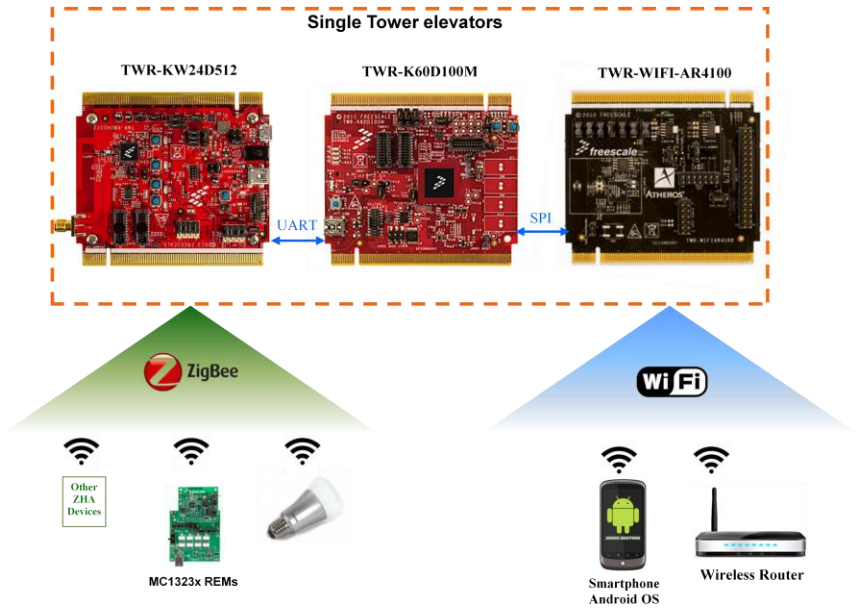


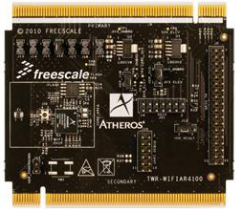
Figure 5. Connected Home Gateway minimum system

3.1.1 Hardware Requirements

The following items are required to assemble and use the gateway system illustrated before.

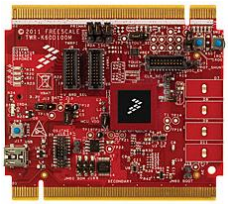
- A pair of Tower System Elevator Modules

- TWR-WiFi-AR4100 board featuring wireless transceiver



- [Optional] Serial TWR board to physically connect the wireless router to the MK60N512VMD100 Ethernet port

- A TWR-K60N512 to handle Ethernet-ZigBee communication bridge



- A TWR-KW24D512 hosting ZigBee HA1.2 Protocol and (optionally) second network's stack

- Commercial Wireless Router

- Android OS enabled device such as a Smartphone or tablet

- External ZHA-enabled appliances such as Bulbs, power outlets or Freescale's EVB boards

3.1.2 Software requirements

Beside the custom code required to run the demo, there are other software blocks needed before implementing the gateway system application.

- MQX RTOS with RTCS library (for Ethernet) for MK60N512VMD100
- ZigBee HA1.2 stack for KW24D512
- Android SDK to develop the application to be used in smart devices

3.1.3 System Assembly

The gateway, besides the wireless router, uses a single pair of TWR elevators so all boards could be assembled in a same platform as described in chapter 3.1.1 Hardware requirements. Next figure illustrates how assembled boards look like, conforming the Connected Home Gateway itself.

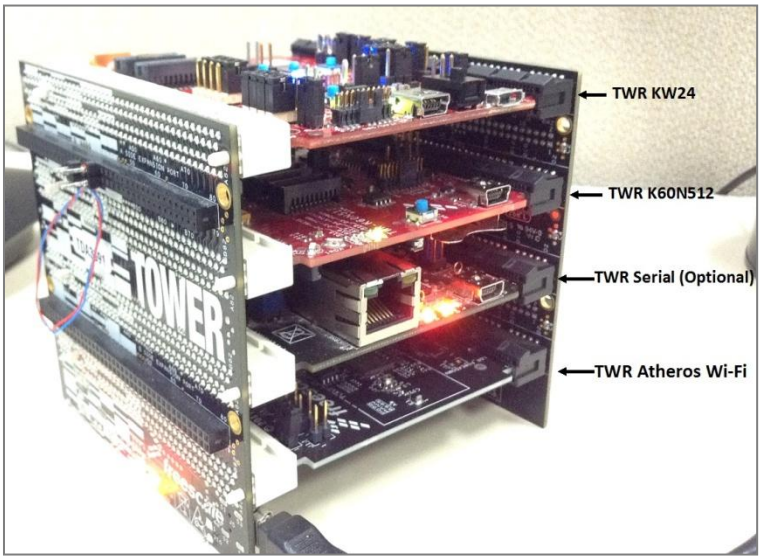


Figure 6. Smart Home Gateway assembled boards

3.2 Design Implementation

Once it is decided which hardware and software blocks are to be used, code could be now implemented to achieve the goals intended. First of all, main MCU firmware will be described.

3.2.1 Main Microcontroller

Summarizing the tasks of the microcontroller we have:

- Control both Ethernet port and WiFi transceiver (via SPI port) to communicate directly with the wireless router and parse the commands received from the android application.
- Send the parsed commands to the KW24D512 device via UART

To achieve the second bullet it is only needed to initialize the UART module and start sending the commands, which is pretty straight forward. The hard work remains in the server application implementation. This is where the RTCS library from MQX RTOS becomes important.

3.2.1.1 MQX RTOS with RTCS library

Freescale streamlines embedded design with a complimentary real-time operating system (RTOS) and software stacks. MQX RTOS includes a multitasking kernel with pre-emptive scheduling and fast interrupt response, extensive inter-task communication and synchronization facilities, and a file system. Its small, configurable size conserves memory space for embedded applications and can be configured as little as 6 KB of ROM, including kernel, interrupts, semaphores, queues and memory manager.

MQX RTOS includes a TCP/IP stack (RTCS) which is used in this implementation. It also includes embedded MS-DOS file system (MFS), USB host/device stack with personal health care device class (PHDC), as well as task-aware debugging. MQX RTOS board support packages are available for a number of platforms, including Kinetis and the TWR-K60N512 board used in this design.

NOTE

For further details and download of MQX RTOS please visit
http://www.freescale.com/MQX

3.2.1.2 Server application

In order to communicate with the wireless router either via Ethernet or WiFi, a server task using the mentioned RTCS library of MQX is required. At the end of the day, this TCP server task will wait for the commands sent by the Android OS application client. Main task would work as follows:

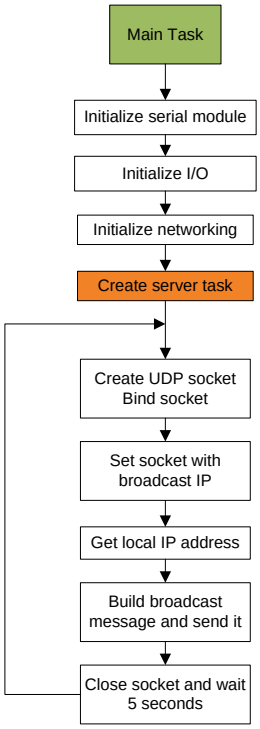


Figure 7. Main MCU application flowchart

Basically, the main application first initializes the peripherals that will be used (i.e. GPIOs or UART) and then proceed to initialize the network (Ethernet or WiFi). Once done that, a new task is created, which basically creates an UDP socket which will send broadcast messages with server IP in order to be detected by the android application or any other client. The server task is the one in charge to manage the connections.

The following flow chart illustrates how it basically works:

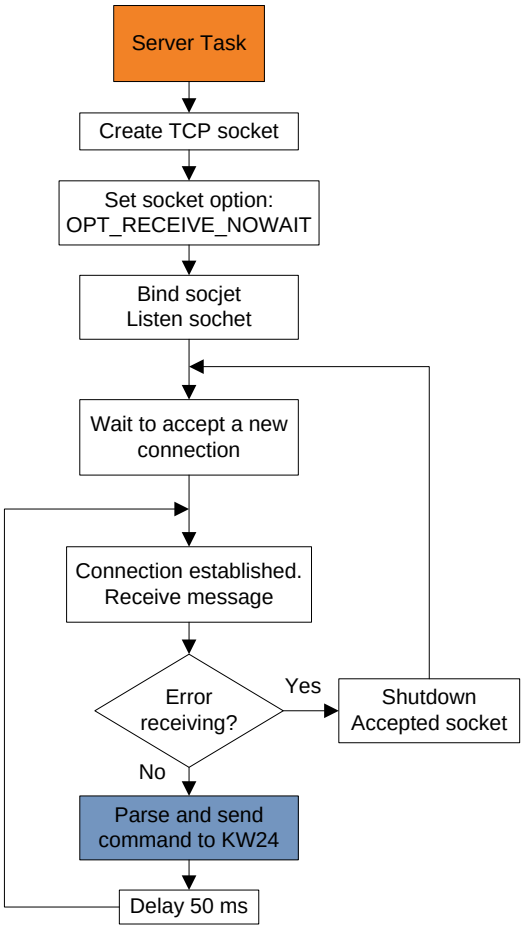


Figure 8. TCP server task flowchart

Also, since there are many example applications using MQX and RTCS library when installing the product, there was no need to start an application form scratch and re-use one of the existing to simply add the custom code that the server requires. For this reason the “shell” application was selected, which already starts the server and receive commands serially.

NOTE

If WiFi is desired to be included (implemented in the demo), users should acquire a TWR-WIFI-AR4100 board and install Atheros patch for MQX. The shell demo also supports Atheros board and it is transparent to the user.

3.2.1.3 Server Settings

The network settings are configurable at compilation time and user is able to change the server's IP assigned (or decide to use DHCP), the gateway mask, the network's SSID to join, the security type, passphrase, among other things. Following figure illustrates how this is done in *config.h* file of the project.

```
#ifdef ENET_DEVICE /* The target's IP address and netmask*/

#ifndef ENET_IPADDR
    #define ENET_IPADDR  IPADDR(192,168,1,95)
#endif

#ifndef ENET_IPMASK
    #define ENET_IPMASK  IPADDR(255,255,255,0)
#endif

#ifndef ENET_IPGATEWAY
        #define ENET_IPGATEWAY  IPADDR(192,168,1,2)
#endif

#endif  /* ENET_DEVICE */

#if DEMOCFG_USE_WIFI
   #include "iwcfg.h"

   #define DEMOCFG_SSID            "KW2X_DUALPAN"
   //Possible Values ENET_MEDIACTL_MODE_INFRA or ENET_MEDIACTL_MODE_ADHOC
   #define DEMOCFG_NW_MODE         "managed"
   /* Posible values.
    * 1. ENET_MEDIACTL_SECURITY_TYPE_WPA
    * 2. ENET_MEDIACTL_SECURITY_TYPE_WPA2
    * 3. ENET_MEDIACTL_SECURITY_TYPE_NONE
    * 4. ENET_MEDIACTL_SECURITY_TYPE_WEP
    */
   #define DEMOCFG_SECURITY        "wpa"
   #define DEMOCFG_PASSPHRASE      "12345678"
   #define DEMOCFG_WEP_KEY         "ABCDE"
   #define DEMOCFG_WEP_KEY_INDEX   1 //can be 1,2,3, or 4
   #define DEMOCFG_DEFAULT_DEVICE  1
   #define DEMOCFG_DEFAULT_WEP_MODE  "open"  //"shared"
   #define DEMOCFG_DEFAULT_U_CIPHER  "TKIP"          //"CCMP"
   #define DEMOCFG_DEFAULT_M_CIPHER  "TKIP"  //"CCMP"
#endif
```

Figure 9. TCP server settings

3.2.2 ZigBee Slave Device

As described in the first chapters of this document, a second MCU is required to act as slave and host the ZigBee stack so it could manage everything related to it and keep it transparent to the TCP server mounted in the host microcontroller.

This slave device is the MKW24D512, a powerful 32-bit MCU part of the Freescale Kinetis family with an 802.15.4 radio in-package and able to host the whole ZigBee stack, also available from Freescale without charge.

The KW24D512 is assembled in the same tower system as well as the MK60N512VMD100 and communicates serially via UART to receive the commands. The only external connection needed is the UART-to-UART signals (RX to TX and vice versa) as illustrated in **Figure 10**.

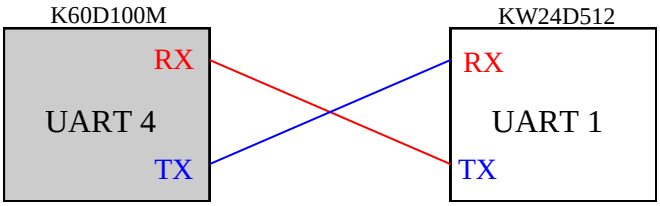


Figure 10. UART connections

4 Android Application

The application used for the specific purpose of interacting with the gateway was developed using Android SDK due to the documentation and tools available for free and because not any special license or tool needs to be purchased. Further details could be obtained in http://developer.android.com/index.html.

4.1 Application Flow chart

The application consists in two different user interfaces, the one launched at initialization that starts the connection with the gateway and the room layout showing the different appliances available.

Basically, the first layout waits for the user to press the connect button to start scanning for gateways available. Once detected, customer selects form a list and the second layout is displayed containing the house layout. Each of the layouts has different appliances to control.

The flowchart would be as follows:

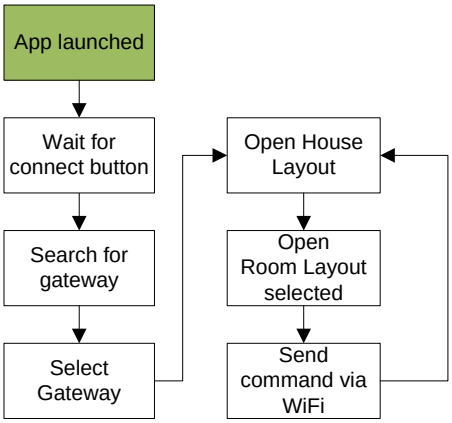


Figure 11. Android application flowchart

4.2 Application Design

As described above, the application running in the Android OS System is integrated with two user Interfaces (UI). The first UI is launched when the application starts and waits for the user to press the button to start searching for the gateway.

Figure 12. Android app initial UI

Once the button is pressed, the application scans for available MK60N512VMD100 devices in the WLAN. Once scanning time out is reached (6 seconds), a dialog box is displayed listing all gateway systems found. The user can then choose one of these devices and start a communication.

Figure 13. Android app gateway list

The second UI allows the user to interact with the Gateway System itself within the established TCP connection. The application uses an actual house layout to simulate a real house (same as the house mockup used with the demo).

Figure 14. Android app rooms layout

Each of the room layouts contain the different house appliances able to be controlled from the app. From a simple light bulb to power outlets and LEDs, the user could change their status with a simple touch.

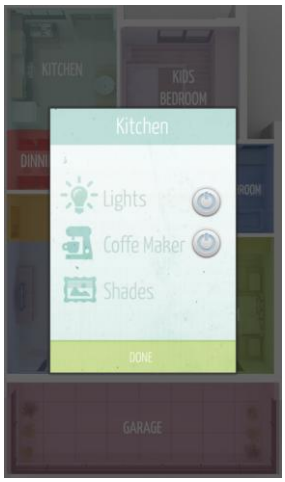


Figure 15. Android app initial UI

NOTE

Please note not all of the appliances listed in every room are available, therefore buttons are not shown. This only demonstrates a potential use case of the gateway and the app.

4.3 Server Connection

The initial layout waits for the user to press a "connect" button. Once pressed, this button initializes global variables, starts the server thread, and starts an asynchronous thread. This last thread is dedicated to show an information dialog in the application every time a new gateway system device is found in the WLAN. The scans last around the 6 seconds and then a dialog box appears listing the available devices.

The Server thread creates a UDP socket to listen for the broadcast messages sent by a gateway through the LAN. Every time a message is received, it is parsed and identified. If this message belongs to a gateway, it is registered in the list. This process continues until the scanning period times out. Following flowchart illustrates this process:

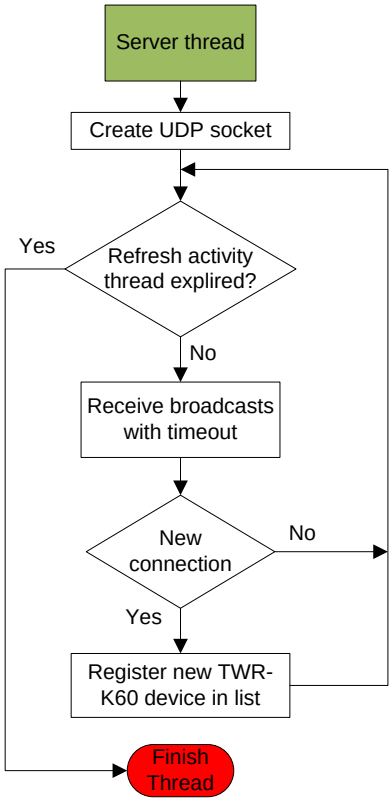


Figure 16. Android app server thread

Afterwards, when the user selects one of the devices in the dialog box, the event listener launches the rooms' layout UI. The process starts when an item from the list is selected. After the listener gets the desired item, an informative dialog displays which device was chosen.

The rooms' layout UI is the last stage. When the IP address is obtained from the devices, a TCP connection is established to send commands to the gateway system. Such commands are listed in **Table 1** of section Shell Commands.

5 Reproducibility

In order to reproduce this implementation, users should do the following:

1. Get the boards listed in Hardware requirements section.
2. Install latest MQX version with Atheros Patch (if WiFi is to be used) and use the shell example as base project.

3. Read Application Note AN4810 about how to communicate the K60 devices with an android application.
4. Implement their own parser for incoming commands from Android device, and translate it to serial commands for ZigBee-enabled MCU (MKW24D512).
5. Interface both boards (K60 and KW24) serially.
6. Implement the parser in KW24 for incoming serial commands from K60 host MCU and translate them to ZigBee commands.

   An option would be to use ZigBee Test Client (ZTC) commands. However, ZigBee may not be mandatory and a different 802.15.4-based stack could be used instead. It will depend in the application's needs.
7. Develop the Android application GUI as desired. Several examples in the web show how to do this.

6 References

- IEEE 802.15.4 - Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)
  http://standards.ieee.org/findstds/standard/802.15.4-2006.html

- ZigBee Specification (2007)
  http://zigbee.org/Specifications/ZigBee/Overview.aspx

- Freescale K60P144M100SF2 Reference manual
  http://cache.freescale.com/files/32bit/doc/ref_manual/K60P144M100SF2RM.pdf

- Freescale MKW24D512V data sheet
  http://cache.freescale.com/files/rf_if/doc/data_sheet/MKW22D512V.pdf

- AN4810 Communication between MQX™ RTOS and Android Operating System
  http://cache.freescale.com/files/microcontrollers/doc/app_note/AN4810.pdf

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com