VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Chenxi Ouyang

DESIGN AND IMPLEMENTATION OF A WIRELESS ZIGBEE MESH NETWORK

Technology and Communication

2014

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Chenxi Ouyang
Title	Design and Implementation of a Wireless ZigBee Mesh Network
Year	2014
Language	English
Pages	47
Name of Supervisor	Antti Virtanen

ZigBee is officially a wireless network protocol that is designed to be used with the low-data-rate sensor and control networks. The objective of this thesis was to implement a ZigBee mesh network with XBee 802.15.4 RF module and Raspberry Pi. The target was to make a mesh network with three nodes.

The project consists of two parts: using the X-CTU application to implement the ZigBee network with XBee RF modules, XBee 802.15.4 Starter Development Kit and XBee Adapter Board, and then the implementation of the ZigBee network with XBee RF modules and Raspberry Pi. This project was to implement a wireless mesh network with three XBee RF modules in the Linux programming environment of Raspberry Pi. Of course, this implementation could be improved in a large mesh network with lots of wireless modules in the future.

The thesis concentrates on how to build a mesh network. In a mesh network, more than two nodes can communicate with each other. First the wireless communication between two XBee RF modules was built, and then the third XBee RF module was added into this network. Finally, this mesh network was tested to see, how the mesh network works.

The result of this project brings a feedback of the mesh network. If a node is in unicast mode, it can provide a peer-to-peer communication. However, if a node is in broadcast mode, it can send data to all the other nodes in a same personal area network.

In conclusion, in a mesh network, there is at least one node is in a broadcast mode, and the DL address of other nodes should be same as MY address of the broadcast node. Otherwise a mesh network cannot be built.

Keywords: XBee, mesh network, Raspberry Pi

FOREWORD

This mesh network is designed for all the people, who are interested in wireless communication. This is my first completed wireless network design and implementation. I met lots of difficulties when I was developing the wireless network. Hereby, I want to give my thanks to all of my teachers and friends who have helped me to go through these problems.

First of all, I would like to give my deepest gratitude to my supervisor Antti Virtanen, thanks for guiding me and comforting me when I lost my direction or felt dispirited. I also want to thank Jukka Matila and my telecommunication teacher Chao Gao, they provided me a lot of help with my thesis.

Then, my thankfulness also goes to my dearest friend Sami Mahamoud Mahamoed, Chunqiu Lu and Han Lin, thanks for your encouragement when I was frustrated.

Chenxi Ouyang

Vaasa, 2nd, June, 2014

ABBREVIATIONS USED

API Mode	Application Programming Interface Mode
AT Mode	Application Transparent Mode
CSMA-CA	Carrier Sense Multiple Access Collision Avoidance
DH	Destination High Address
DL	Destination Low Address
GPIO	General Purpose Input/Output
HDMI	High-Definition Multimedia Interface
IEEE	Institute of Electrical and Electronics Engineers
ISM	Industrial Scientific Medical
MAC	Medium Access Control
NWK	Network
PAN	Personal Area Network
PHY	Physical Layer
Rx	Receive Port
RF	Radio Frequency
Tx	Transmit Port
UART	Universal Asynchronous Receiver/Transmitter

LIST OF FIGURES, TABLES AND LISTINGS

CONTENTS

1 INTRODUCTION

1.1 Background

Today, wireless communication is increasingly taking place where there used to be cables in communications. This technology is becoming more and more popular because of its low cost and ease-of-use. In many cases wireless communication is much cheaper than the wired communication. This technology allows people a faster and more convenient access to the outside world although its communication reliability is not good enough. /2/

ZigBee is one of the representative wireless modules using what is called Carrier Sense Multiple Access Collision Avoidance (CSMA-CA) to increase reliability /2/. It is a specification for a suite of high level communication protocols used to create personal area networks built from small, low-power digital radios /2/. ZigBee is a technology which enhances reliability through mesh networking, acknowledgements and use of the robust IEEE 802.15.4 standard /9/. XBee is a technology utilizing ZigBee, XBee provides the world with a variety of wireless applications and gradually becomes an industry standard in wireless communication /9/.

1.2 Objective of the Thesis

The aim of this project was to build a short range communication network which can be used for sending and receiving data between different XBee RF modules. The details of the implementation will be displayed on chapter 4. Chapter 5 will give the outcome of the project

1.3 Planning of the Project

The planning procedure for this project can be divided into three steps. The first step was to be familiar with the AT commands. It is the most basic requirement for the wireless network implementation.

Then, the second step was to know how to use XBee RF module with its Starter Development Kit. For example, how to use the AT commands to configure XBee RF module with the X-CTU software and start to communicate between two modules. The XBee RF modules employ a UART interface, which allows any microcontroller or microprocessor to immediately use the services of the ZigBee protocol /2/. The ZigBee hardware designer has to make sure that the host's logic-levels of serial-port should be compatible with the XBee's 2.8-3.4V logic levels /5/.

The last indispensable step was to replace the XBee 802.15.4 Starter Development Kit with Raspberry Pi. Raspberry Pi is like a microcontroller which uses the C programming to control the XBee serial communication in the Linux environment. Based on the Raspberry Pi programming, it is easy to mesh the network with three XBee RF modules.

This paper is organized as follows: Chapter 2 is a detailed description about XBee protocol and serial communication. The two XBee modes and mesh network theory is also mentioned here. The tools and implementation language are described in Chapter 3. In Chapter 4, the steps for configured Raspberry Pi and XBee RF modules are displayed. Chapter 5 is the outcome of the project. Finally, Chapter 6 will summarize the whole project.

2 THEORETICAL BASIS

The XBee RF module is the product based on ZigBee protocol. So it also uses IEEE 802.15.4 standards. It can support a wireless sensor network which requires low-cost and low-power. For the XBee RF module, it does not need too much power and it can provide a reliable way to communicate between two or more devices. It interfaces to a host device through a logic-level asynchronous serial port. Through its serial port, the module can communicate with any logic and voltage compatible UART; or through a level translator to any serial device (for example through an XBee Adapter). The controlling with the XBee RF module is very easy with the AT commands. The XBee RF module can be configured with the AT (Application Transparent) mode or API (Application Programming Interface) mode but in this thesis implementation, only the AT mode is used. So in the next chapters, this thesis concentrates on the AT commands. /2/, /5/, /13/

A key component of the ZigBee protocol is the ability to support mesh networking. In a mesh network, nodes are interconnected with other nodes so that multiple pathways connect to each node. In this project, three XBee RF modules were used to build a small mesh network. In this wireless network, all of them could communicate with each other with the suitable configuration.

2.1 ZigBee Protocol and Communication

ZigBee operates in the worldwide 2.4 GHz ISM (Industrial Scientific Medical) band. Proprietary radios are the primary competitor of ZigBee today. ZigBee is based on the IEEE 802.15.4 networking. The protocol stack can be divided into two parts: IEEE 802.15.4 and the ZigBee Alliance. The physical layer (PHY) and the MAC (Medium Access Control) layer are defined by the IEEE 802.15.4 specification. The ZigBee stack’s NWK (Network) layer of the ZigBee stack and all layers above it are under the control of the ZigBee Alliance specification. /2/

A typical ZigBee stack is organized in the fashion shown in Figure 1.

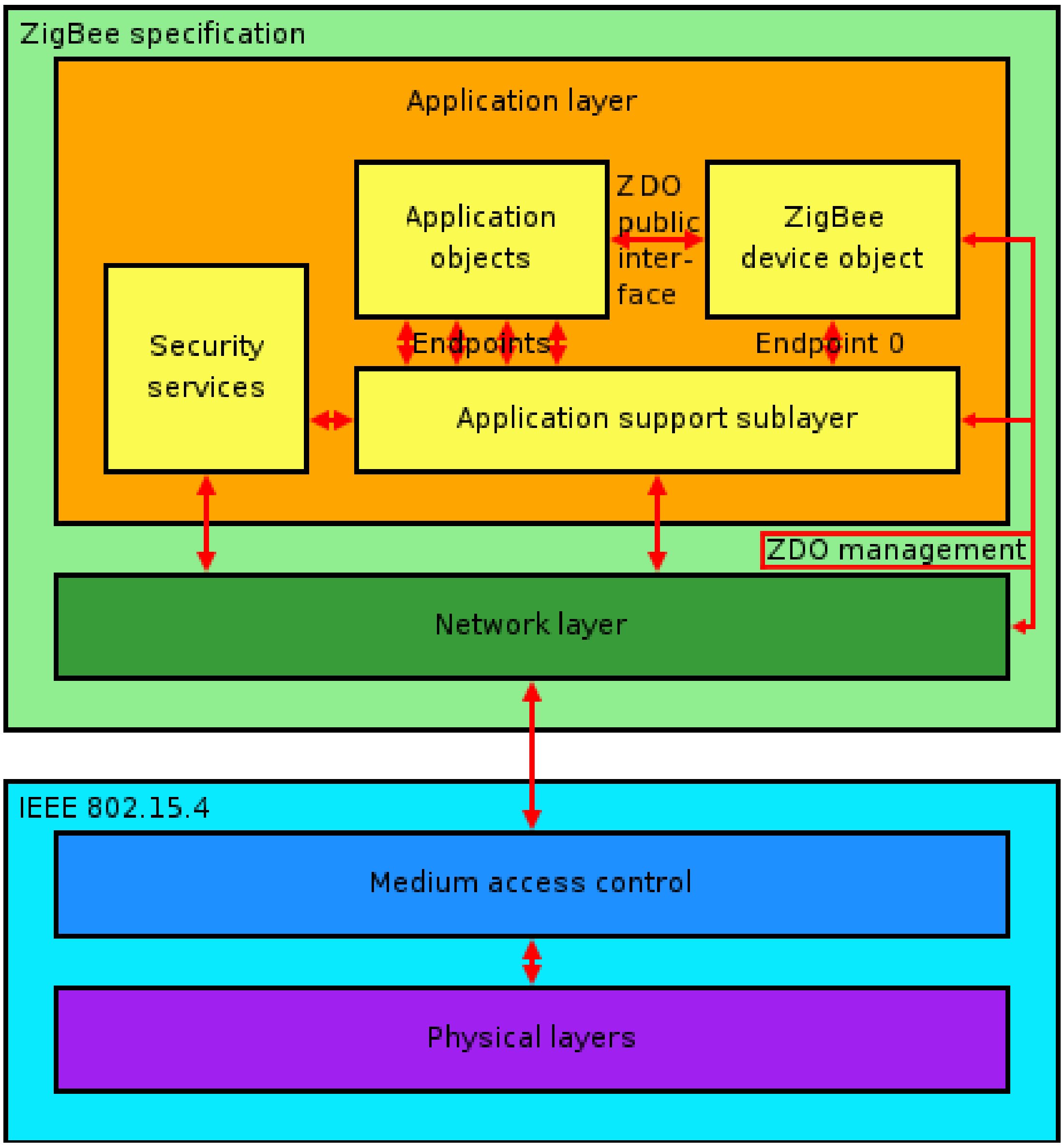


Figure 1. ZigBee Protocol Layer /15/

In a nutshell, the IEEE 802.15.4 standard defines the characteristics of the physical and MAC layers. ZigBee builds upon the IEEE 802.15.4 standard and defines the network layer specifications and provides a framework for application programming in the application layer /2/.

2.2 UART Serial Communication

The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. The UART takes bytes of data and transmits the individual bits in a sequential fashion /1/. At the destination, a second UART re-assembles the bits into complete bytes /1/. In this project, the communication between Raspberry Pi and XBee RF module is the UART serial communication. Raspberry Pi can transmit and receive bytes of data from XBee RF module.

2.3 Two XBee Networking

As shown in Figure 2, the coming data from remote module typically includes: source address, destination address, error checking values, and other pertinent information needed by the protocol /5/. The data which is sent by one node will encompass here with the other information, so that the other node will receive it.

In networking terms, the packaging of the data use point-to-point transmission. When the information comes from one microcontroller, it will first be encompassed by source address, destination address, error checking values, and so on. After these packaging, the remote XBee module will receive the data out of error. Then the microcontroller which connects to this module will follow the requirements of coming information. /5/

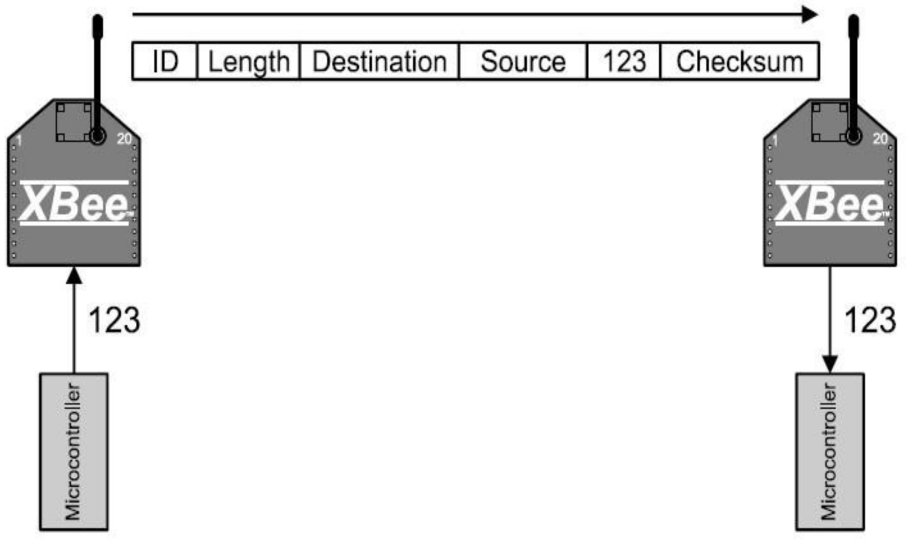


Figure 2. Two XBee Networking /5/

The XBee supports both an AT and an API (Application Programming Interface) Mode for sending and receiving data at the controller. Both have their own advantages.

2.3.1 AT Mode

In the AT Mode, also called Transparent Mode, just the message data itself is sent to the module and received by the remote controller. The protocol of wireless communication between these two XBee nodes is simple. As shown in Figure 2, the simple transmission and reception of serial data is allowed by this mode. The AT commands are used to configure the XBee RF modules, so that one node can communicate with its destination node. /5/

In chapter 4: the AT commands will be explained in details. In general, the implementation requires placing the XBee into Command Mode, sending AT codes for configuration, and exiting the Command Mode.

Although the transmission and reception is only data, the message itself is encapsulated with needed information, and then it can be passed between the nodes.

2.3.2 API Mode

In the API Mode, the programmer packages the data with needed information, such as destination address, types of packet, and checksum value. As the transmission and reception of data in the AT Mode, the command parameter is given one by one, and each command need a waiting time. However, the advantage of API mode is that all the command parameters can be used to configure an XBee RF module at one time. In other words, under suitable programming, the API Mode can provide users more reliability and flexibility in some cases and it can avoid unnecessary energy consuming. /5/

Although it seems a better choice to use the API Mode to build a mesh network, in this thesis project the AT Mode was chosen, because it is much easier than the

API Mode. However, this mode consumes energy when people change configuration on the destination address every time.

2.4 Mesh Networking

A mesh network is a local area network, it consist of at least three nodes. In these nodes, there is at least one coordinator and several routers. Figure 3 gives an example of a simple mesh network. For a mesh network, it is more flexible and reliable than a peer-to-peer network that is because if one node in this network is failed, the data can also send out with other nodes. The failed node will not cause a big problem in the process of communication in the wireless network /10/.

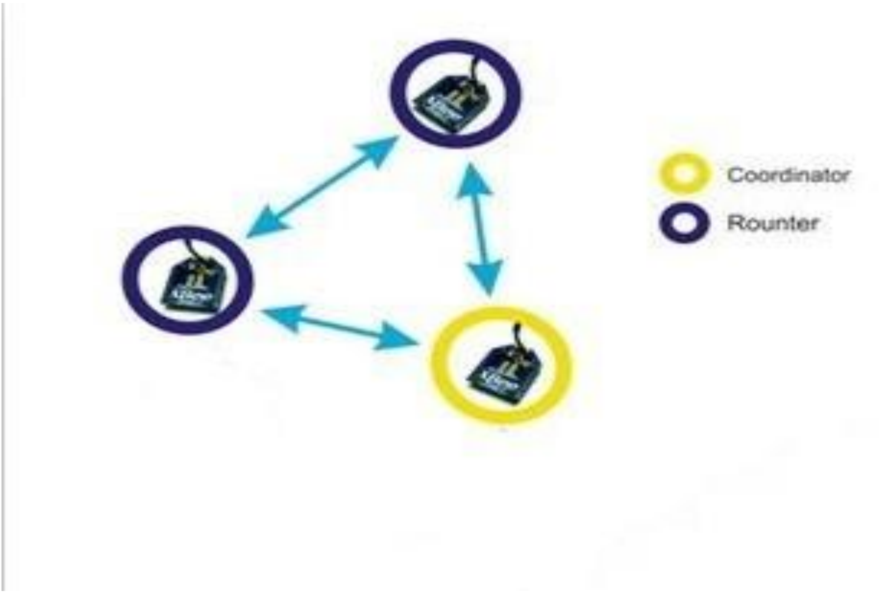


Figure 3. Mesh Network /6/

3 PROGRAMMING LANGUAGE AND TOOLS

In this chapter, the programming language and the tool for this project are introduced in details.

In general, this project is based on Raspberry Pi using the C programming to configure an XBee RF module and control it to communicate with remote XBee RF modules. PuTTY and X-CTU monitor the whole process in this project.

3.1 C Programming

The C programming is relatively simple, yet powerful and widely used. The C language is a very basic language. And this language can be used on many applications. In addition, the experience with the C language is useful for a better understanding of the Linux operating system. That is because their contents are largely written in C language.

3.2 Developing Environment and Tools

PuTTY and WinSCP are the tools to access Raspberry Pi. X-CTU is used by the XBee RF module when it connects to PC with XBee 802.15.4 Starter Development Kit or XBee 5V/3.3V Adapter Board.

3.2.1 X-CTU Application

This software is easy to use and allows MaxStream customers to test the radio modems in the actual environment with just a computer and the items include with the radio modems /14/. This application can configure XBee RF module as the requirement of users directly, such as destination address, coordinator/ end devices, AT/ API Mode, etc.

3.2.2 WinSCP

WinSCP (Windows Secure Copy) is a free and open-source SFTP, SCP and FTP client for Microsoft Windows. Its main function is to secure the file transfer between a local and a remote computer /11/. This application provides people an easy method to get into Raspberry Pi. With the correct IP address, username, password of a Raspberry Pi, people can remote logon the SD card of a Raspberry Pi, and then bring into or take away the files they want.

3.2.3 PuTTY

PuTTY is a free and open-source terminal emulator, serial console and network file transfer application. It supports several network protocols, including SCP, SSH, Telnet, rlogin, and raw socket connection. The name "PuTTY" has no definitive meaning, though "tty" is the name for a terminal in the Unix tradition, usually held to be short for Teletype /7/. This application provides people a way to logon the Linux operating system of the Raspberry Pi directly. With “gcc” and “./a.out” commands, it is easy to check and run a program in the SD card of Raspberry Pi.

3.2.4 XBee 802.15.4 Starter Development Kit

This kit is designed to make it easy to set up an XBee network, send data from one XBee RF module to another, and adjust the XBee RF module settings.

3.2.5 XBee 5V/3.3V Adapter Board

The adapter board is designed to make adding wireless point-to-point or mesh networking easy. For the XBee RF module, the space between pins is 2 mm. But the typical hole space of breadboard and solder board is 2.54mm. So it is hard to connect it with a normal device easily. /5/

On the adapter board:

- Two LED lights. Green one is blinking when this board powered. The frequency of the red one blinking shows the status of the XBee RF module (such as the coordinator blinks one time per second, the end device blinks two times per second)
- Ten pins on it used for different target.

In this thesis, this adapter board was chosen to provide the serial interfacing.

The XBee 5V/3.3V Adapter Board is shown in Figure 4.

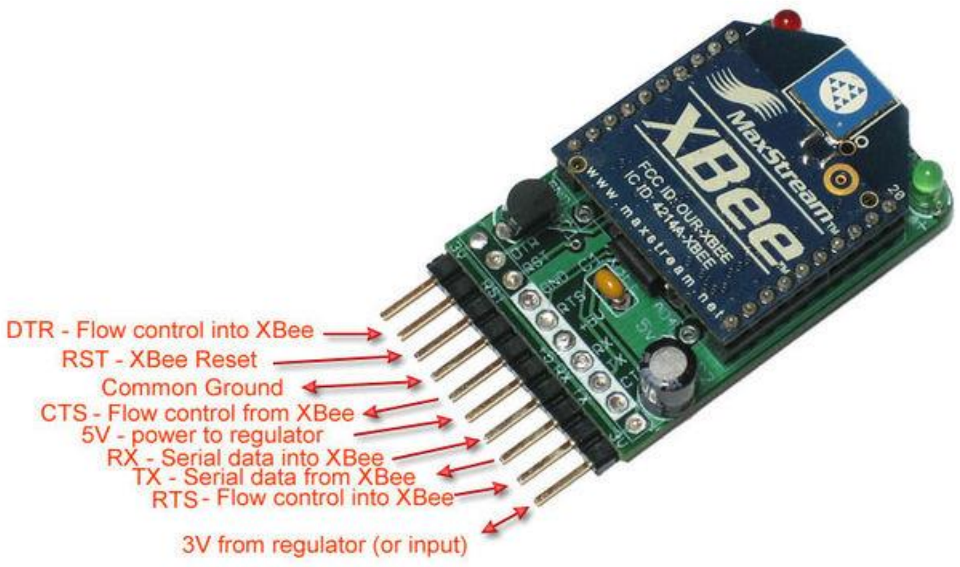

DTR - Flow control into XBee
RST - XBee Reset
Common Ground
CTS - Flow control from XBee
5V - power to regulator
RX - Serial data into XBee
TX - Serial data from XBee
RTS - Flow control into XBee
3V from regulator (or input)

Figure 4. XBee 5V/3.3V Adapter /12/

3.2.6 Raspberry Pi Board

The Raspberry Pi is a credit-card-sized single-board computer. It is manufactured in two board configurations through licensed deals with Newark element14 (Premier Farnell), RS Components and Egoman /3/. For Raspberry Pi, after it is installed in the Raspbian system, it primarily uses the Linux environment. Many devices can be controlled by it with a different port. In this project, we use the GPIO serial port of Raspberry Pi.

4 IMPLEMENTATION

This chapter will illustrate how to configure XBee RF module and Raspberry Pi as well as the details of hardware programming and programming.

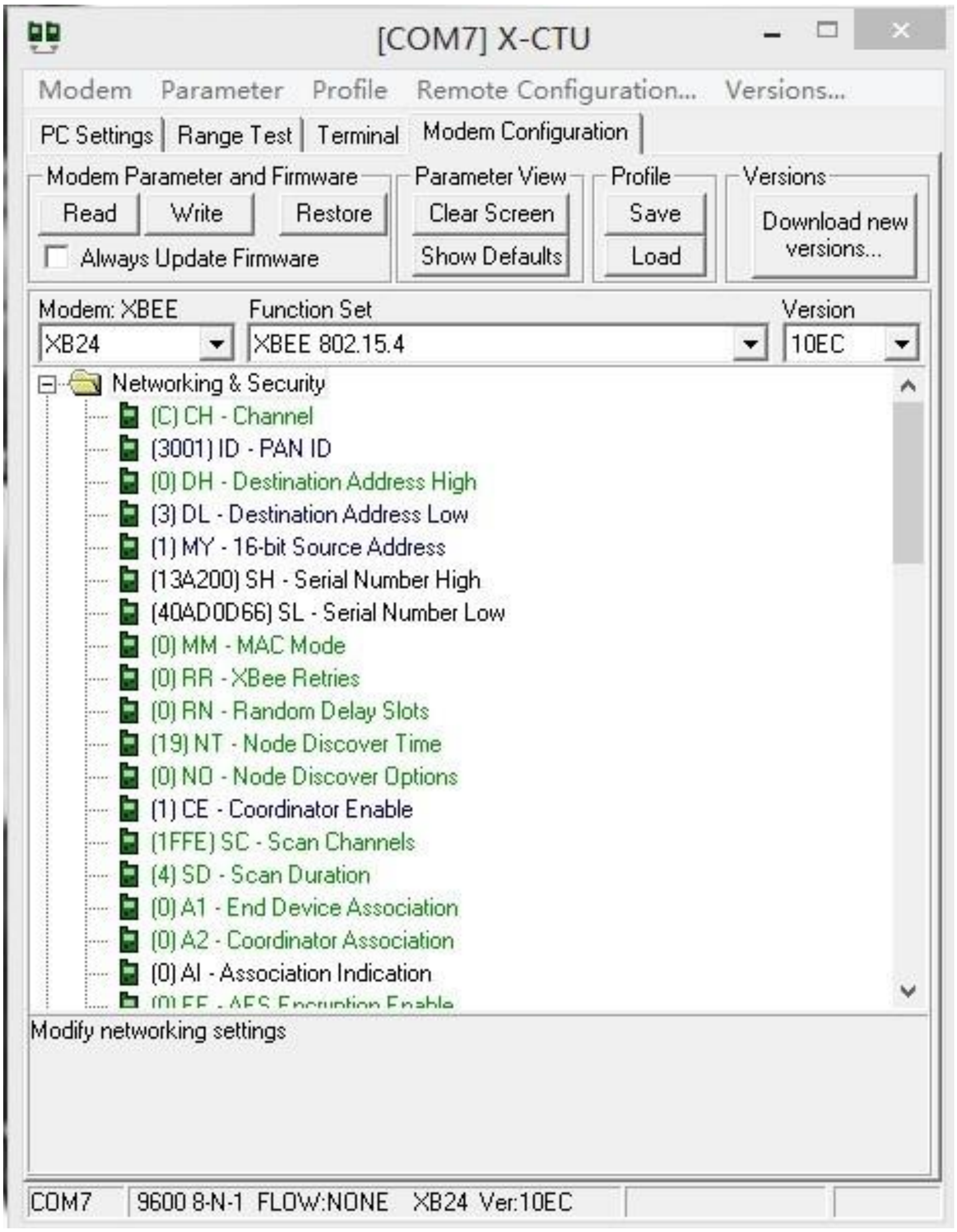
4.1 XBee RF Module Setting

Before controlling the XBee RF module by Raspberry Pi, the configuration of XBee RF module by X-CTU should be done first. In this project, the first step of implementation is to connect XBee RF module to the PC with XBee 802.15.4 Starter Development Kit. In the application of X-CTU, the XBee module can be configured as the requirement of users.

With the experience of daily life, a problem will come out when people want to talk with a specific person in a group. Who you talk with? How to make other people realize that you are talking to one of them? If there is only one person, there is no doubt that the person is the target that you try to communicate with. However, for example, in a classroom or a meeting room, if you cannot provide effective information to identify the person you want to talk to, others in this group will feel confused. When you can provide a valid information to identify a person directly (for instance: name), other people will know they are in no need to response nor perhaps use the data you send. /5/

That is the reason for why the XBee RF module should be configured first by the X-CTU application before using it. After the configuration, every XBee RF module can be distinguished easily. For instance, the PAN ID is like a small class which is used to separate nodes into groups. Then the MY address is like a personal identification for other people to recognize you. In addition, the DL address is like the information to identify other people you want to talk to. So after these configurations, a simple XBee communication network will be built.

The two XBee RF modules have been set as shown below. Figure 5 is setting for the first XBee RF module and the Figure 6 is setting for the second XBee RF module.



Modem: XBEE — XB24
Function Set — XBEE 802.15.4
Version — 10EC

Networking & Security
- (C) CH - Channel
- (3001) ID - PAN ID
- (0) DH - Destination Address High
- (3) DL - Destination Address Low
- (1) MY - 16-bit Source Address
- (13A200) SH - Serial Number High
- (40AD0D66) SL - Serial Number Low
- (0) MM - MAC Mode
- (0) RR - XBee Retries
- (0) RN - Random Delay Slots
- (19) NT - Node Discover Time
- (0) NO - Node Discover Options
- (1) CE - Coordinator Enable
- (1FFE) SC - Scan Channels
- (4) SD - Scan Duration
- (0) A1 - End Device Association
- (0) A2 - Coordinator Association
- (0) AI - Association Indication
- (0) EE - AES Encryption Enable

Modify networking settings

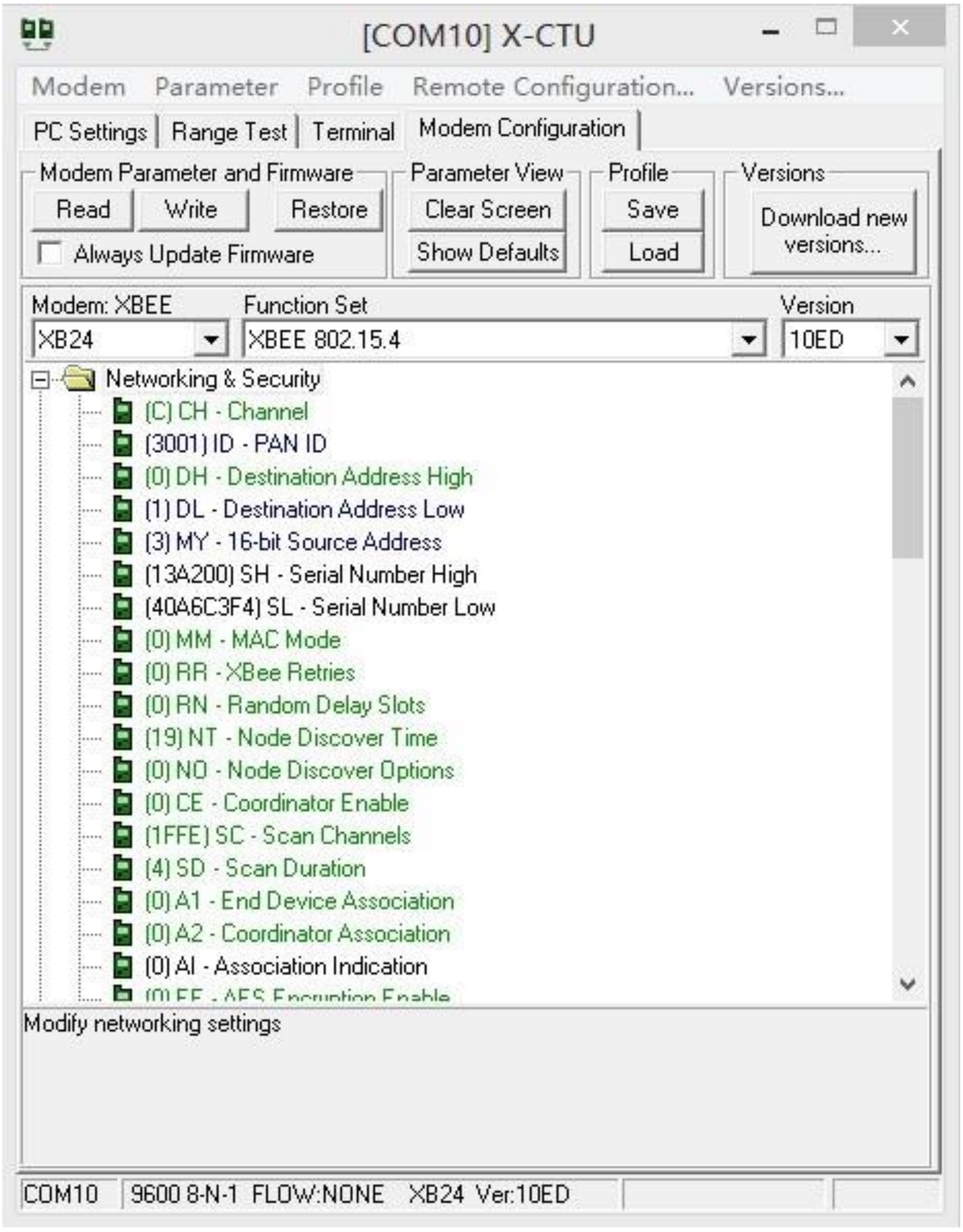
COM7 | 9600 8-N-1 FLOW:NONE | XB24 Ver:10EC

Figure 5. First XBee RF Module Setting

The setting of the first local XBee RF module is:

- ID : 3001
- DH : 0
- DL : 3
- MY : 1

- CE : 1(coordinator enable)



XB24 | XBEE 802.15.4 | 10ED

```
Networking & Security
  (C) CH - Channel
  (3001) ID - PAN ID
  (0) DH - Destination Address High
  (1) DL - Destination Address Low
  (3) MY - 16-bit Source Address
  (13A200) SH - Serial Number High
  (40A6C3F4) SL - Serial Number Low
  (0) MM - MAC Mode
  (0) RR - XBee Retries
  (0) RN - Random Delay Slots
  (19) NT - Node Discover Time
  (0) NO - Node Discover Options
  (0) CE - Coordinator Enable
  (1FFE) SC - Scan Channels
  (4) SD - Scan Duration
  (0) A1 - End Device Association
  (0) A2 - Coordinator Association
  (0) AI - Association Indication
  (0) EE - AES Encryption Enable
Modify networking settings

COM10 | 9600 8-N-1 FLOW:NONE | XB24 Ver:10ED
```

Figure 6. Second XBee RF Module Setting

The setting of the first local XBee RF module is:

- ID : 3001
- DH : 0
- DL : 1

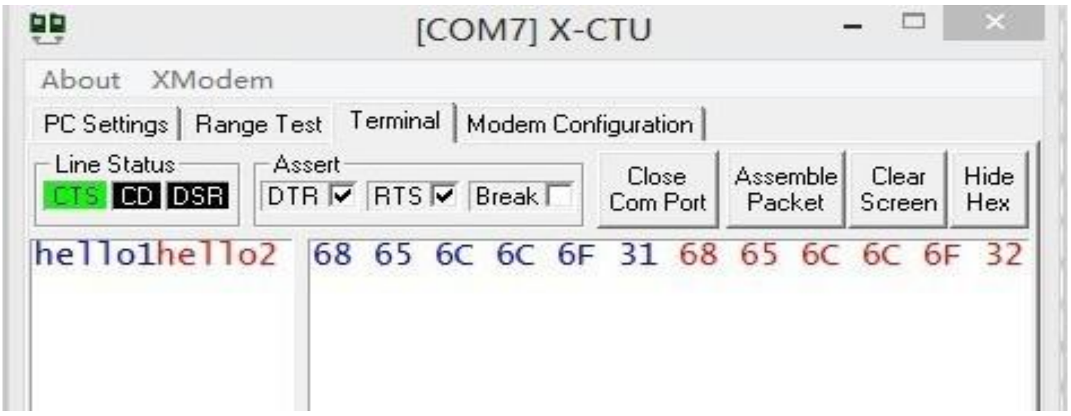
- MY : 3
- CE : 0

In the Modem Configuration window, it is easy to find that both of them are working in the same PAN ID, it is a personal area network and all of the nodes in the same PAN ID can communicate with others.

In addition, the DH, which means destination high address, is set to 0 in two of them. That is because the total bits of the address that two XBee RF modules use to communicate is 32, so 16 bits are used for high address and another 16 bits used for low address.

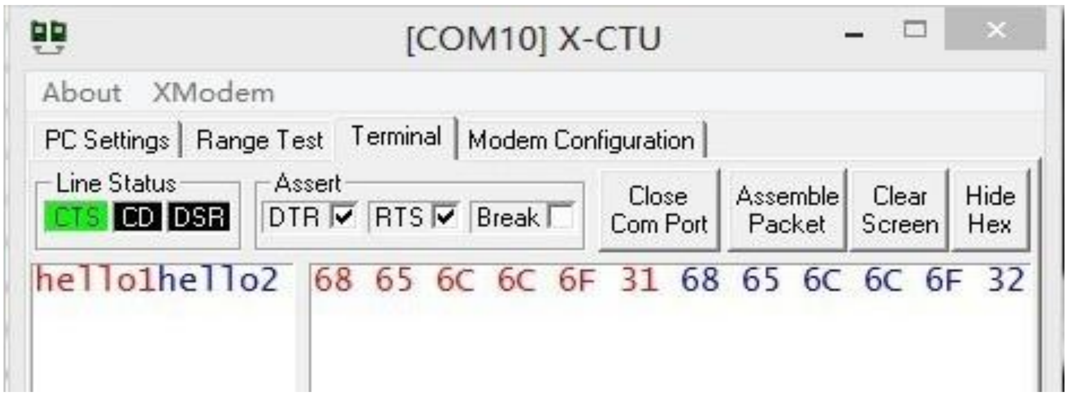
Then, the DL and MY are set to 3 and 1 respectively for the first XBee RF module, and 1 and 3 for the second one. The CE of the Xbee module is also set which connects to COM7 to Coordinator.

After these settings, it is the time turn to the Terminal window to check if the XBee configuration is right or not. Here are the results in Figure 7 and Figure 8. The first node sends “hello1” to the second node, and then receives “hello2” back from the second one.



```
hello1hello2    68 65 6C 6C 6F 31 68 65 6C 6C 6F 32
```

Figure 7. Node 1 send “hello1” to Node 2



```
hello1hello2  68 65 6C 6C 6F 31 68 65 6C 6C 6F 32
```

Figure 8. Node 2 send "hello2" to Node 1

In these figures, the string in red color is the sending part and the blue string is the receiving part. It is easy to see that the communication is built in a bi-direction.

4.2 Raspberry Pi Configuration

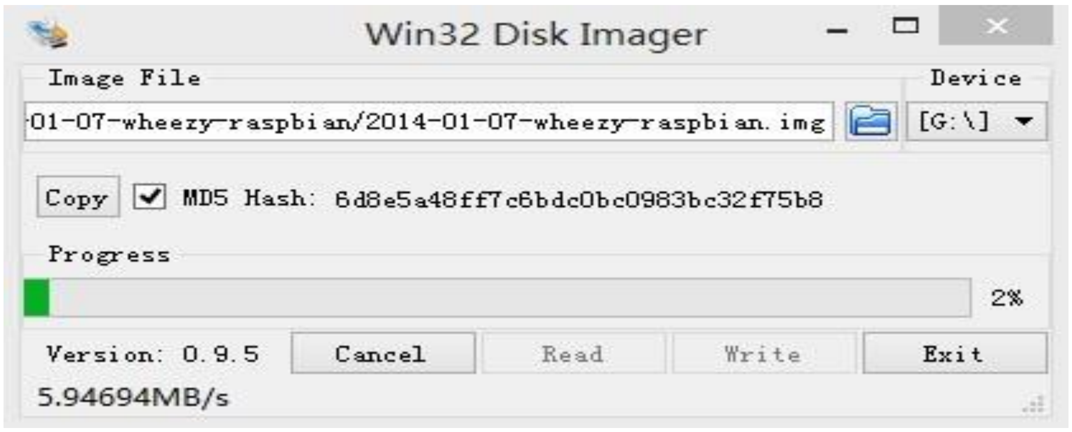
All of the data are stored in the SD card of Raspberry Pi. To use a Raspberry Pi for the first time, the SD card needs to be configured and initialized and some default settings enabled or disabled in the system of Raspberry Pi. After this, Raspberry Pi can be used easily.

4.2.1 SD Card Configuration

Before getting started with Raspberry Pi, the SD card first needs to be formatted. It can be formatted the following four steps:

- ✓ Plug the SD card into the PC, then download the newest image from the website. http://www.raspberrypi.org/downloads/
- ✓ Download the win 32 Image Writer from the website. https://launchpad.net/win32-image-writer
- ✓ Write the image with image writer to the SD card, and wait for a few minutes.
- ✓ Plug the SD card into Raspi, then connect it to computer with an HDMI cable.

Figure 9 shows the details of the SD card configuration. PC use Win32 Disk Imager to write image into SD card of Raspberry Pi.



```
01-07-wheezy-raspbian/2014-01-07-wheezy-raspbian.img    [G:\]
MD5 Hash: 6d8e5a48ff7c6bdc0bc0983bc32f75b8
2%
Version: 0.9.5
5.94694MB/s
```

Figure 9. Image Writer

4.2.2 Booting Raspberry Pi for the First Time

Raspi should be booted at the first time.

- ✓ On the first boot the Raspi-config window will appear on the screen.
- ✓ Make sure that the Raspbian system is already installed.
- ✓ Reboot the Raspi and remove the HDMI cable.
- ✓ Logon with the username: pi and the password: raspberry with PuTTY on your own PC.

4.2.3 UART Port Start

Many of the GPIO pins on the Pi have other special uses. The most useful of these are the serial port pins #8 and #10, which transmit and receive for an RS 232 serial port /4/. In this project, port pins #8 and #10 will connect to the reception and transmission pin on the XBee Adapter board. It should be noted that transmission pin connects to the reception pin and the reception pin connects to the transmission pin.

By default this port will output diagnostic messages during the boot and then provides a user login. The configuration is 8 bits, no parity, 1 stop bit, no hardware handshaking, at 115200 baud /4/. All of these configurations are the default settings by Raspberry Pi. So it should be changed to meet the requirements of users.

The device name is /dev/tty/AMA0. If taking complete control of the serial port is required, there are two configuration changes to make:

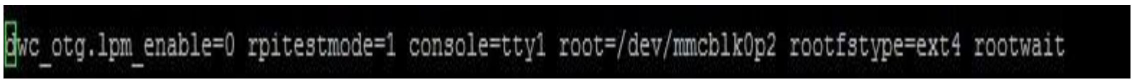
- First, disable the boot up and diagnostic output to the serial port.

  sudo nano /boot/cmdline.txt

  Remove the content below:

  console=ttyAMA0,115200 kgdboc=ttyAMA0,115200

  After removing, the /boot/cmdline.txt file will look as shown in Figure 10:



```
dwc_otg.lpm_enable=0 rpitestmode=1 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 rootwait
```

Figure 10. cmdline.txt Change

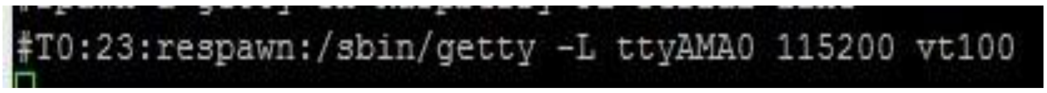
- Second, disable the login prompt.

  sudo nano /etc/inittab

  Find the line near the end.

  T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

  Comment this line, and the file after the change should look as in Figure 11:



```
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Figure 11. inittab Change

4.3 Serial Programming

This part will display the hardware implementation and the programming in a wireless network.

4.3.1 Hardware Implementation

After the settings for the UART port of Raspberry Pi Board is ready, it is time to connect the XBee RF module to the Raspberry Pi GPIO port. There are two steps in this process:

1. Plug XBee RF module into XBee 5V/3.3V Adapter Board.
2. Connect ground and 5V power pins of XBee 5V/3.3V Adapter and Raspberry Pi. Then connect Tx and Rx of XBee Adapter Board to Rx and Tx of Raspberry Pi board. The GPIO of Raspberry Pi is shown below:

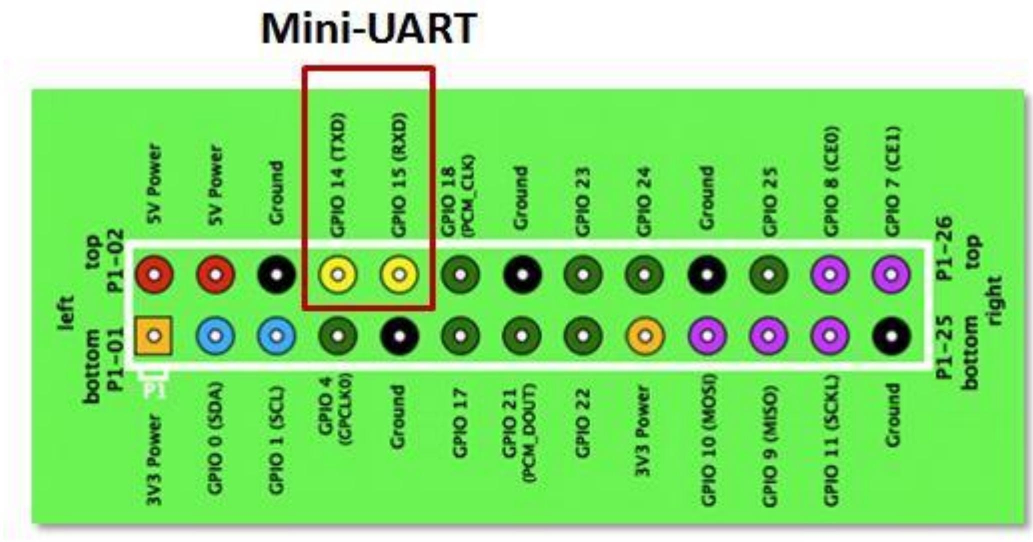


Figure 12. GPIO of Raspberry Pi /8/

4.3.2 Programming Flowchart

Figure 13 provides a general view of the programming in this project. In this flowchart, `init_com()` function is used for com port initializing. It has two values, 0 and -1. When `init_com()!= -1`, UART port on the Raspberry will open, then the parameters for XBee configurations will be set. After setting, `sendCommand()` will begin to these settings to XBee RF module, and then send data “hello” five times. When sending is finished, this program will turn to `receiveCommand()`, from this time, Raspberry Pi can receive the data come from the remote XBee RF module.

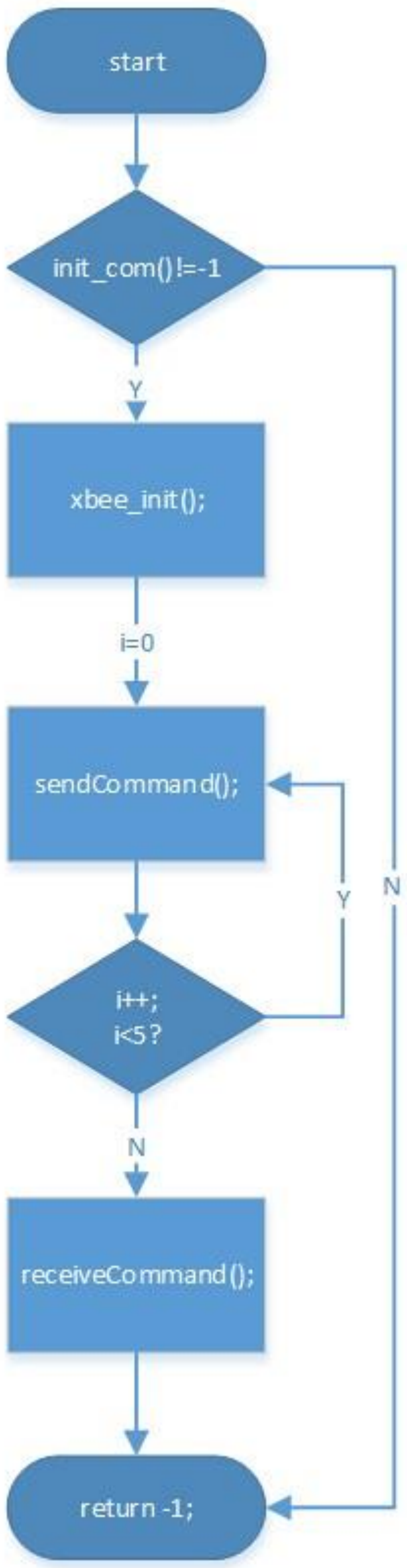


Figure 13 Flowchart

4.3.3 Programming Details

Listing 1 comes from `init_com()` function, it is used to open the UART port of Raspberry Pi, after opening `/dev/ttyAMA0`, and this file is not equals to 0, the serial communication can start from now on.

Listing 1. UART port start

```
uart0_filestream = open("/dev/ttyAMA0", O_RDWR | O_NOCTTY |
O_NDELAY); //Open in non blocking read/write mode
```

Before turning into the AT mode in `xbee_init()` function, there is no data coming for at least 2s. So in Listing 2, 2s delay is used to ensure that there is no data coming before entering into next three plus.

Listing 2. Ensure no data coming before sending "+++"

```
sendCommand("\n\n\n\r");
_delay_ms(2000);
```

Then, the Raspberry Pi will transmit three plus to the XBee RF module with 2s delay at the end. That is because "+++" is the command to start the AT Mode, and its protocol requires at least a 2s delay to wait for the XBee RF module response. Then the XBee RF module will come into the AT Mode. Listing 3 below displays how to start the AT Mode.

Listing 3. AT Mode start

```
sendCommand("+++");//send command to enter XBee Pro command mode
_delay_ms(2000);// waiting for finish sending and XBee respond
```

Then the next step is to initialize the XBee RF module before using it to communicate with the remote module. atmy1 will configure the MY address to 1, and atdl3 will configure the DL address to 3. atwr here is used to write the AT commands into the XBee RF module. However, between the configurations of two AT commands, a delay time is needed. So `_delay_ms()` function here is used to provide the XBee RF module the response time it needs. Listing 4 below used for XBee RF module initializing:

Listing 4. XBee RF module initializing

```
 sendCommand("atmy1");//send command to set Source address
_delay_ms(500);

sendCommand("atwr");// send "WR" (write)command to Xbee
_delay_ms(500);

sendCommand("atdl3");//  send  command  to  set  Destination  low
address
_delay_ms(500);

sendCommand("atwr");
_delay_ms(500);
```

After these settings, in Listing 5, atcn is used to exit the AT Mode to normal sending mode.

Listing 5. Exit AT Mode

```
sendCommand("atcn");// send command for Exit AT Command Mode
```

The setting of the XBee RF module which connects to Raspberry Pi is

- atmy : 1
- atdl : 3

It can talk to the remote XBee RF module whose setting is:

- atmy : 3
- atdl : 1

Then the `sendCommand()` function will send it out. The variable of “length” and “temp” here will record the length of message. If the length is bigger than 3, a return and an ending mark will be added to the end of data. The buffer of message and str are used for storing data. The message buffer will copy all of the contents inside of it to the str buffer one by one. In Listing 6, the data will be copied from the message buffer to the str buffer with a return and an ending mark.

Listing 6. Copy data from message to str with return and ending mark

```
int length=strlen(message),retn;
int temp=strlen(message);
char str[20];

if(length>3)
/*length is used for AT command length which we enter into, if
length greater than 3, that would be a return in ASCII after AT
command*/
           {


while(length--)
           {
               str[length]=message[length];
           }
           str[temp]= (char)13;
           str[temp+1]= '\0';
}
```

As seen in Listing 7, if the length is equals to 3, the program will copy all the characters and “+++” in the contents from the message to str directly.

Listing 7. Copy data from message to str directly

```
else
           {

strcpy(str, message); //else, copy all the characters from message
to str directly

}
```

Then, as shown in Listing 8, the program will print the content of the str and the length of str out.

Listing 8. Print str and the length of it out

```
printf("%s %d\n",str,strlen(str));
```

Next, the program will write the str content and the length of the content into `/dev/ttyAMA0`.

Listing 9. Write the str and the length of it into uart0_filestream file

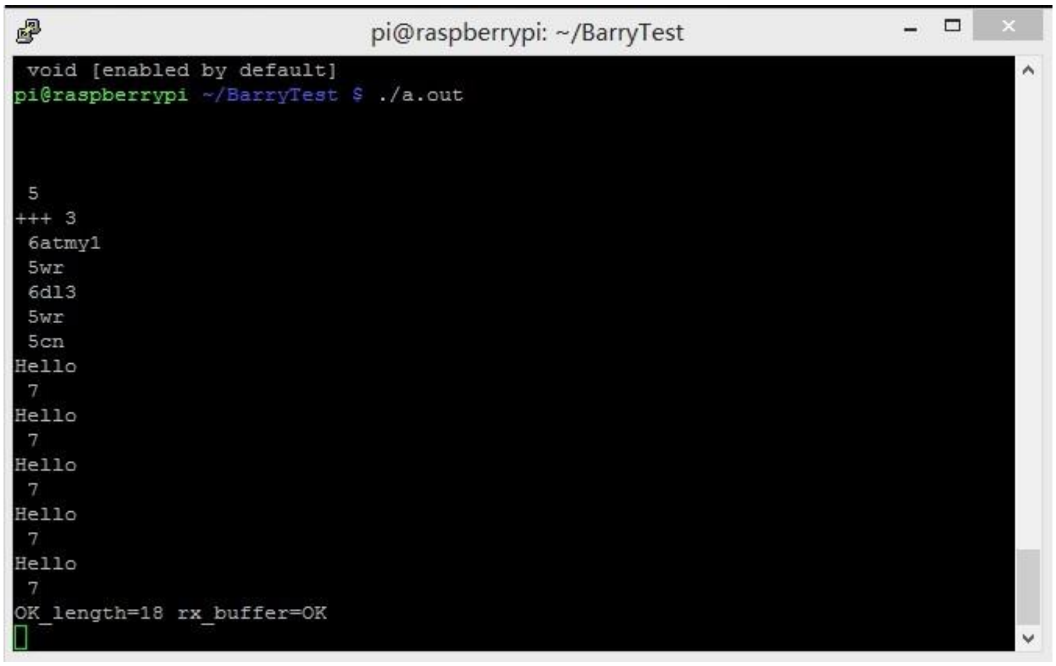
```
retn = write(uart0_filestream, str, strlen(str));
```

retn is used to save the byte write into the file, if retn is equal to 3, for example "+++" comes. It will print out the length. The Listing 10 below will show the details of print.

Listing 10. Print retn out which length is equal to 3

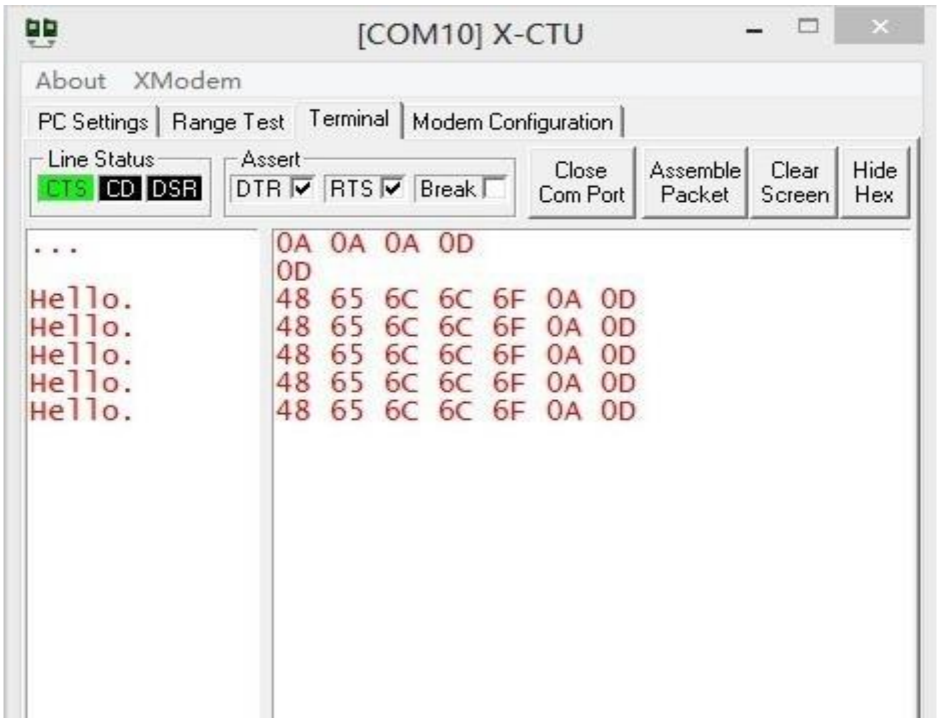
```
if(retn==3)
{
                printf("%d ", retn);
}
```

Figures 14, 15 show the details of communication. The data comes from Raspberry Pi after the XBee RF module has been configured. On the X-CTU terminal window, it displays which data it received.



```
pi@raspberrypi: ~/BarryTest
 void [enabled by default]
pi@raspberrypi ~/BarryTest $ ./a.out



 5
+++ 3
 6atmy1
 5wr
 6dl3
 5wr
 5cn
Hello
 7
Hello
 7
Hello
 7
Hello
 7
Hello
 7
OK_length=18 rx_buffer=OK
```

Figure 14. Raspberry Pi Send "Hello" Five Times to X-CTU



```
...                  0A 0A 0A 0D
                     0D
Hello.               48 65 6C 6C 6F 0A 0D
Hello.               48 65 6C 6C 6F 0A 0D
Hello.               48 65 6C 6C 6F 0A 0D
Hello.               48 65 6C 6C 6F 0A 0D
Hello.               48 65 6C 6C 6F 0A 0D
```

Figure 15. X-CTU Receive "Hello" Five Times from Raspberry Pi

After sending successfully, Raspberry Pi can also receive data with the `receiveCommand()` function.

In this part of code, `while(1)` is used to make sure that the receiving continues. `rx_buffer` is used here to store data, the size of it is 20, meaning 20 bytes is the maximum length to receive. Listing 11 will show the definition of rx_buffer.

Listing 11. rx_buffer definition

```
unsigned char rx_buffer[20];
```

`rx_length` is used to check the data length inside of `rx_buffer`. If the length is greater than 0, it means some data is coming. In Listing 12, the definition of rx_length will be displayed.

Listing 12. rx_length definition

```
int rx_length = read(uart0_filestream, (char*)rx_buffer, 19);
//Filestream, buffer to store in, number of bytes to read (max)
```

Then, `printf()` function can print the length and content in rx buffer out. It should be noted that before printing the contents in the buffer, the user should initialize the buffer first. The programming details will be shown in listing 13.

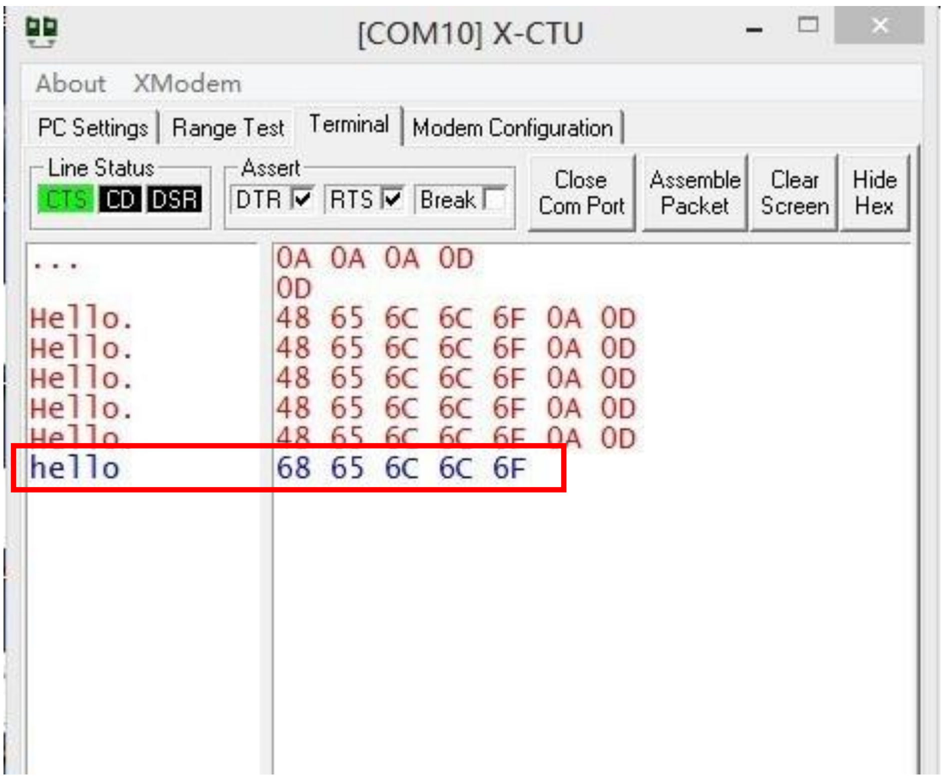
Listing 13. Print the length and content in rx buffer

```
rx_buffer[rx_length] = '\0';     //initialized the buffer
printf("rx_length=%d rx_buffer=%s\n", rx_length, rx_buffer);
```

If the `rx_length` equals to 0, it means there is no data coming, so this program will wait for the coming data until it received.

There are two ways to send data from X-CTU, the first way is to send characters one by one, the second way is to send a string.

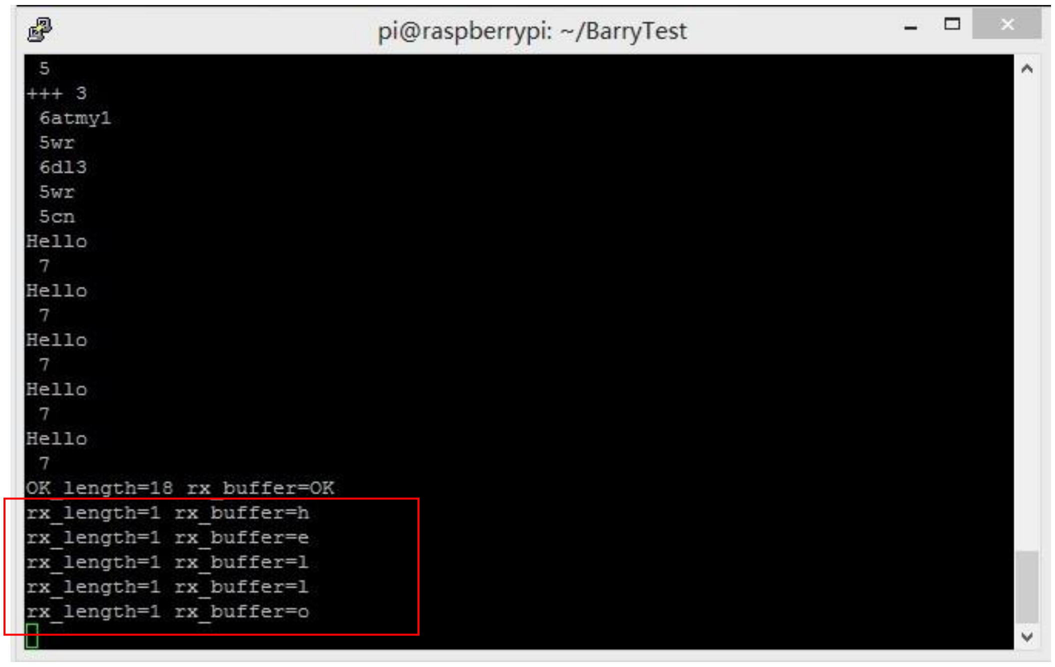
Figures 16, 17 show the details of sending characters one by one from X-CTU to Raspberry Pi. In the terminal window, red characters are the data which X-CTU received from Raspberry Pi before, blue characters are the data which it send to Raspberry Pi.



```
...                 0A 0A 0A 0D
                    0D
Hello.              48 65 6C 6C 6F 0A 0D
Hello.              48 65 6C 6C 6F 0A 0D
Hello.              48 65 6C 6C 6F 0A 0D
Hello.              48 65 6C 6C 6F 0A 0D
Hello               48 65 6C 6C 6F 0A 0D
hello               68 65 6C 6C 6F
```

Figure 16. X-CTU Send "hello" in characters to Raspberry Pi

In Figure 17, the highlight block diagram shows the details of receiving in Raspberry Pi. It is clear to see that Raspberry Pi received only one character at a time. Because `rx_length` is 1, and the content of `rx_buffer` is "h", "e", "l", "l", "o" separately.

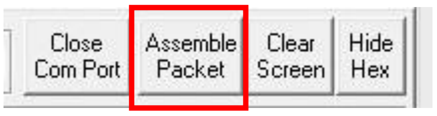


```
 5
+++ 3
 6atmy1
 5wr
 6dl3
 5wr
 5cn
Hello
 7
Hello
 7
Hello
 7
Hello
 7
Hello
 7
OK length=18 rx buffer=OK
rx_length=1 rx_buffer=h
rx_length=1 rx_buffer=e
rx_length=1 rx_buffer=l
rx_length=1 rx_buffer=l
rx_length=1 rx_buffer=o
```

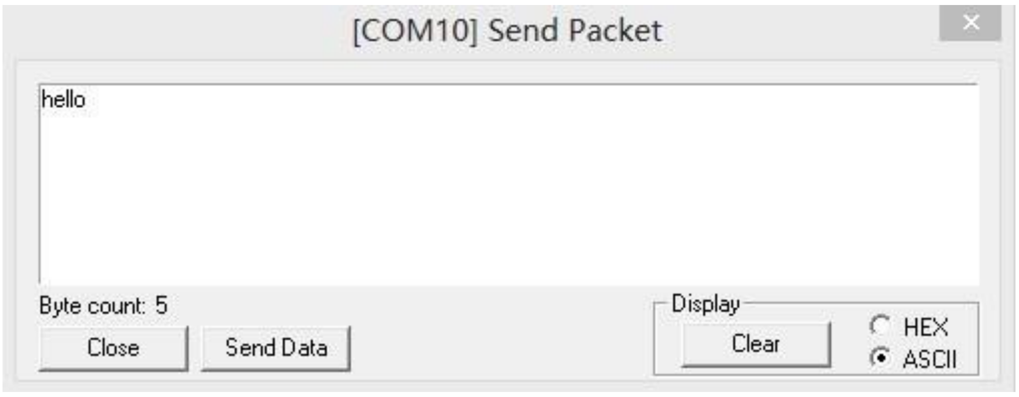
Figure 17. Raspberry Pi Receive Characters from X-CTU

Figure 18 shows how to send a string instead of one character.

First click the “AssemblePacket” button on the right top X-CTU



Then, a window will appear for users to type into a string



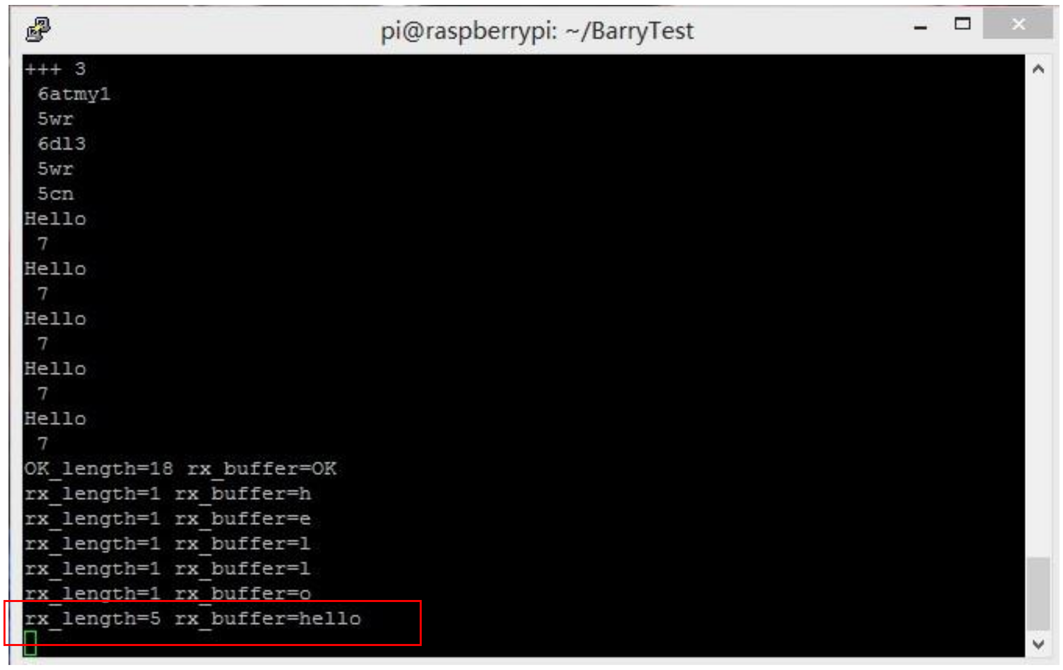
hello

Byte count: 5

HEX [✓ ASCII]

Figure 18. X-CTU Send String to Raspberry Pi

"hello" here is a five byte wait for sending by X-CTU, so click "Send Data" button, Figure 19 shows the result on PuTTY. This time, Raspberry Pi can receive the content as a string instead of characters.



```
+++ 3
 6atmy1
 5wr
 6dl3
 5wr
 5cn
Hello
 7
Hello
 7
Hello
 7
Hello
 7
Hello
 7
OK_length=18 rx_buffer=OK
rx_length=1 rx_buffer=h
rx_length=1 rx_buffer=e
rx_length=1 rx_buffer=l
rx_length=1 rx_buffer=l
rx_length=1 rx_buffer=o
rx_length=5 rx_buffer=hello
```

Figure 19. Raspberry Pi Receive String from X-CTU

Five bytes have been received and the content of the `rx_buffer` is "hello".

All results mean that the communication between the Raspberry Pi and the remote XBee RF module works very well, peer-to-peer communication has been built.

5 OUTCOME OF PROJECT

This chapter gives the result of a wireless mesh network.

5.1 Mesh Network Implementation and Testing

After all the preparations, it was time to build a mesh network with three XBee RF modules. In this project, there were four cases to implement a mesh network.

5.1.1 Mesh Network Implementation case 1

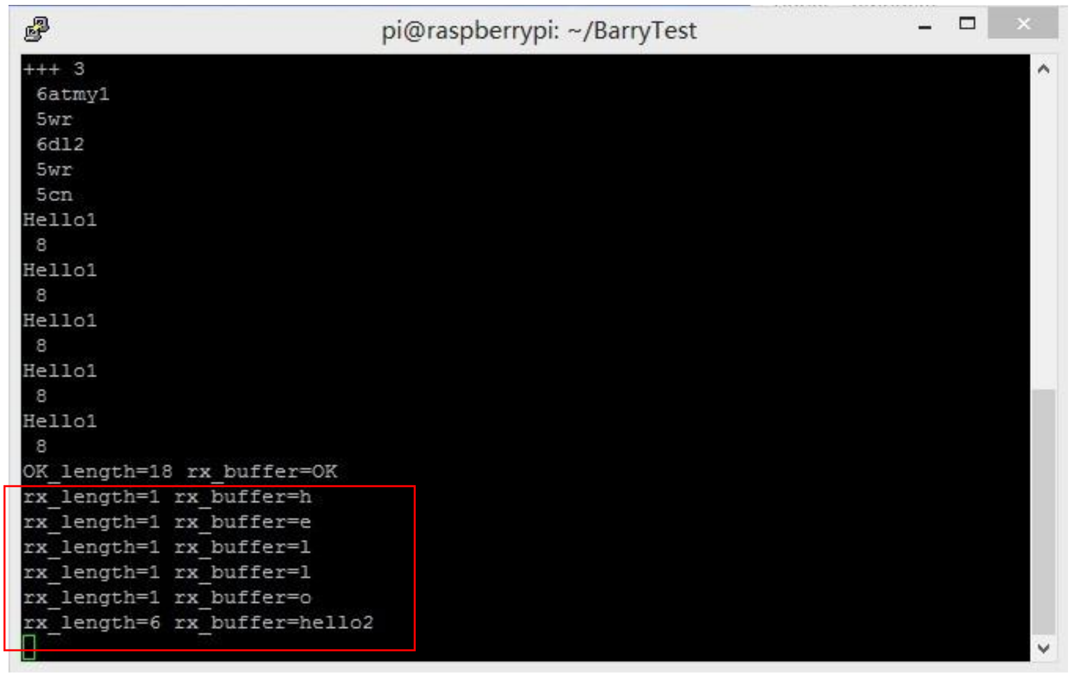
The configurations of three XBee RF modules are listed below:

The first XBee RF module which was connected to Raspberry Pi. Its CE was configured to the coordinator and its DH was configured to 0 before by XBee 802.15.4 Starter Development Kit. Table 1 below lists the configurations of the three XBee RF modules.

Table 1. Mesh Network Configuration Case 1

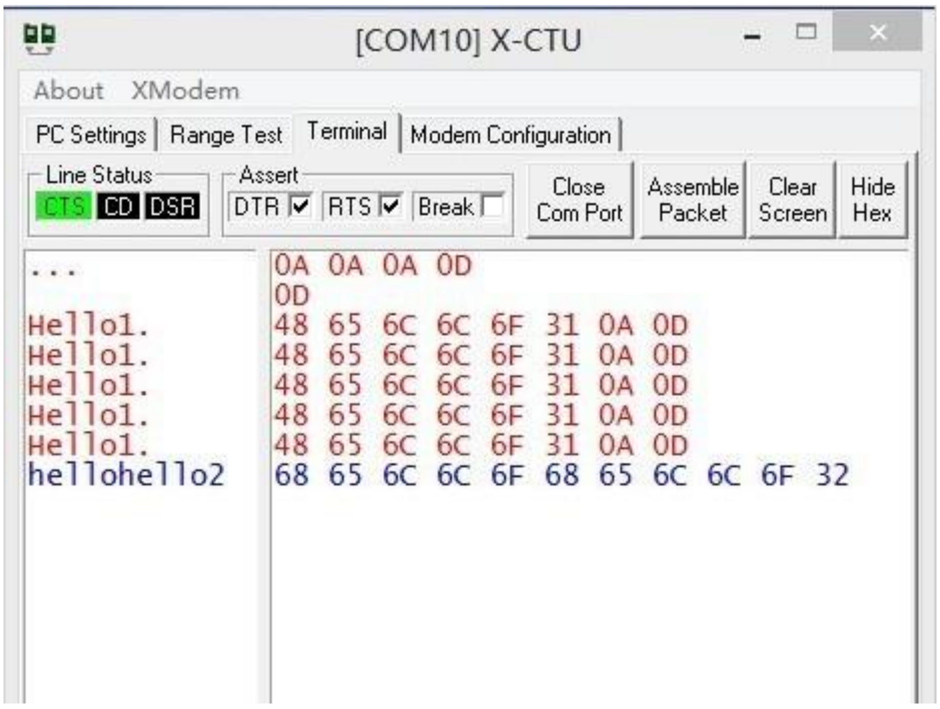
The First module	The Second module	The Third module
ID: 3001	ID: 3001	ID: 3001
DH: 0	DH: 0	DH: 0
DL: 2	DL: 1	DL: 4
MY: 1	MY: 2	MY: 3
Message sending: Hello1	Message sending: hello, hello2	Message sending: hello3

When the configuration is as above, the communication was successful between the first one and the second one, but failed in the last one. The communications between the three XBee RF modules are shown in Figure 20, Figure 21 and Figure 22. In this figure, the first node can transmit data to node 2 and receive from it also. For node 2, it sends message “hello” as characters and “hello2” as a string.



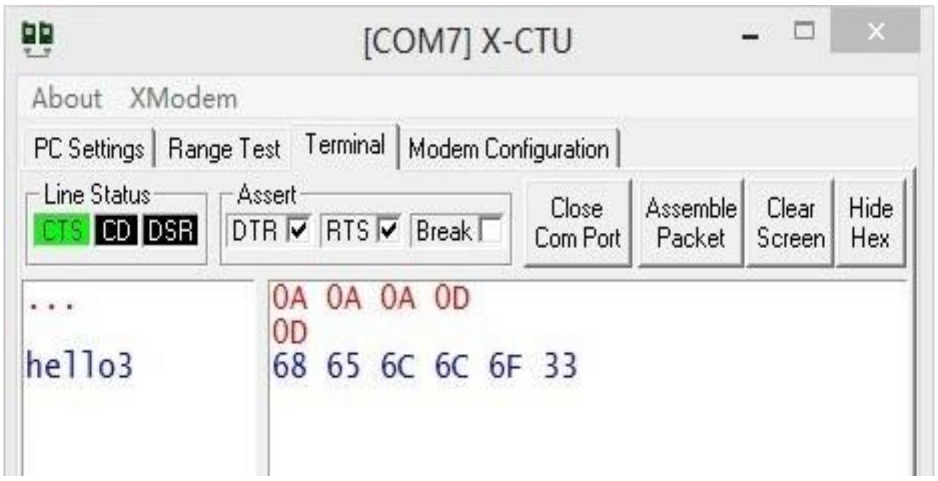
```
+++ 3
 6atmy1
 5wr
 6dl2
 5wr
 5cn
Hello1
 8
Hello1
 8
Hello1
 8
Hello1
 8
Hello1
 8
OK_length=18 rx_buffer=OK
rx_length=1 rx_buffer=h
rx_length=1 rx_buffer=e
rx_length=1 rx_buffer=l
rx_length=1 rx_buffer=l
rx_length=1 rx_buffer=o
rx_length=6 rx_buffer=hello2
```

Figure 20. Node 1 send “Hello1” and receive “hello”, “hello2”



```
...              0A 0A 0A 0D
                 0D
Hello1.          48 65 6C 6C 6F 31 0A 0D
Hello1.          48 65 6C 6C 6F 31 0A 0D
Hello1.          48 65 6C 6C 6F 31 0A 0D
Hello1.          48 65 6C 6C 6F 31 0A 0D
Hello1.          48 65 6C 6C 6F 31 0A 0D
hellohello2      68 65 6C 6C 6F 68 65 6C 6C 6F 32
```

Figure 21. Node 2 receive “Hello1” and send “hello”, “hello2”



```
...
hello3        0A 0A 0A 0D
              0D
              68 65 6C 6C 6F 33
```

Figure 22. Node 3 send "hello3"

This is the unicast Mode, in this mode, only support peer-to-peer communication is possible. If the first XBee RF module wants to communicate with the third node in the same personal area network, it needs to change its DL and MY address to:

- DL : 3
- MY : 4

5.1.2 Mesh Network Implementation case 2

Table 2 shows the configuration of XBee RF modules. In the case 2 configuration, the communication between the first module and the third module was built, but the communication between the first module and the second module failed.

Table 2. Mesh Network Configuration Case 2

The First module	The Second module	The Third module
ID: 3001	ID: 3001	ID: 3001
DH: 0	DH: 0	DH: 0
DL: 3	DL: 1	DL: 4
MY: 4	MY: 2	MY: 3
Message sending: Hello1	Message sending: hello2	Message sending: hello3

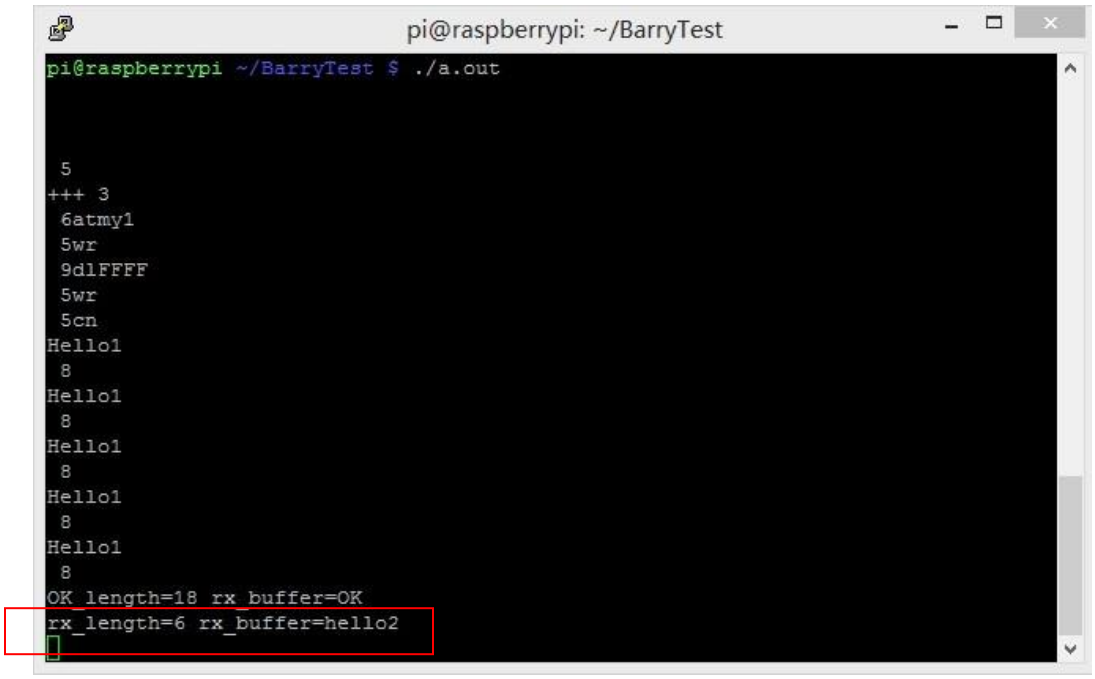
5.1.3 Mesh Network Implementation case 3

If the first node wants to communicate with all of the nodes in the same personal area network, the DL of the first node is changed to ”FFFF”, it is slightly different as before. ”FFFF” means the broadcast mode. In theory, if one node is in the broadcast mode, all of other nodes in the same personal area network can receive the data coming from the broadcast node. Table 3 below displays the configurations of three XBee RF modules in case 3.

Table 3. Mesh Network Configuration Case 3

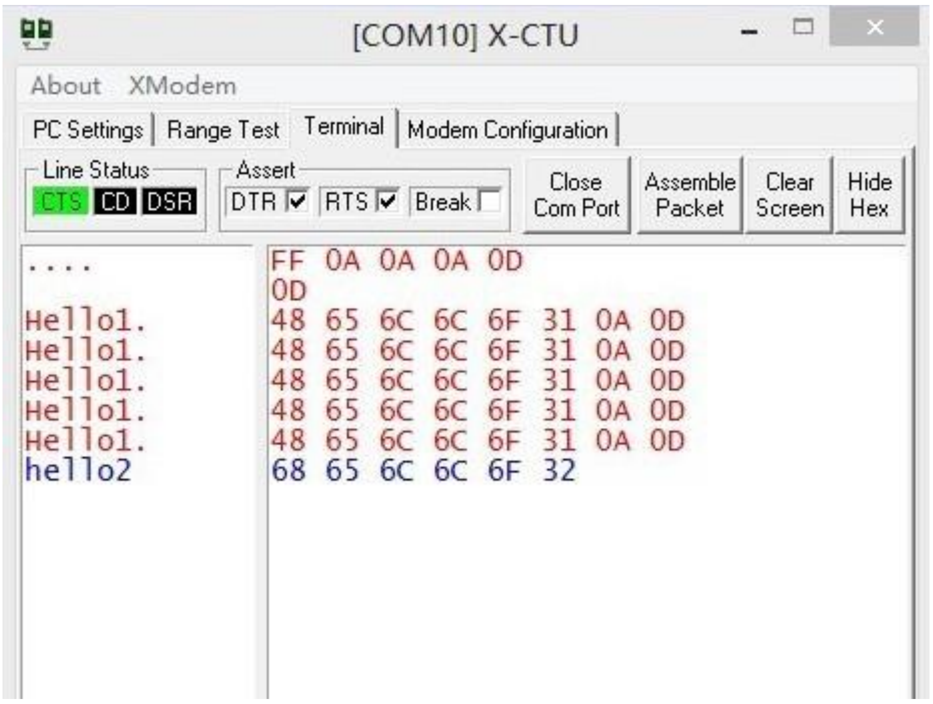
The First module	The Second module	The Third module
ID: 3001	ID: 3001	ID: 3001
DH: 0	DH: 0	DH: 0
DL: FFFF	DL: 1	DL: 4
MY: 1	MY: 2	MY: 3
Message sending: Hello1	Message sending: hello2	Message sending: hello3

Figure 23, Figure 24 and Figure 25 show the details of communication after this change. Raspberry Pi can transmit “Hello1” to node 2 and node 3, but it can receive “hello2” from node 2.



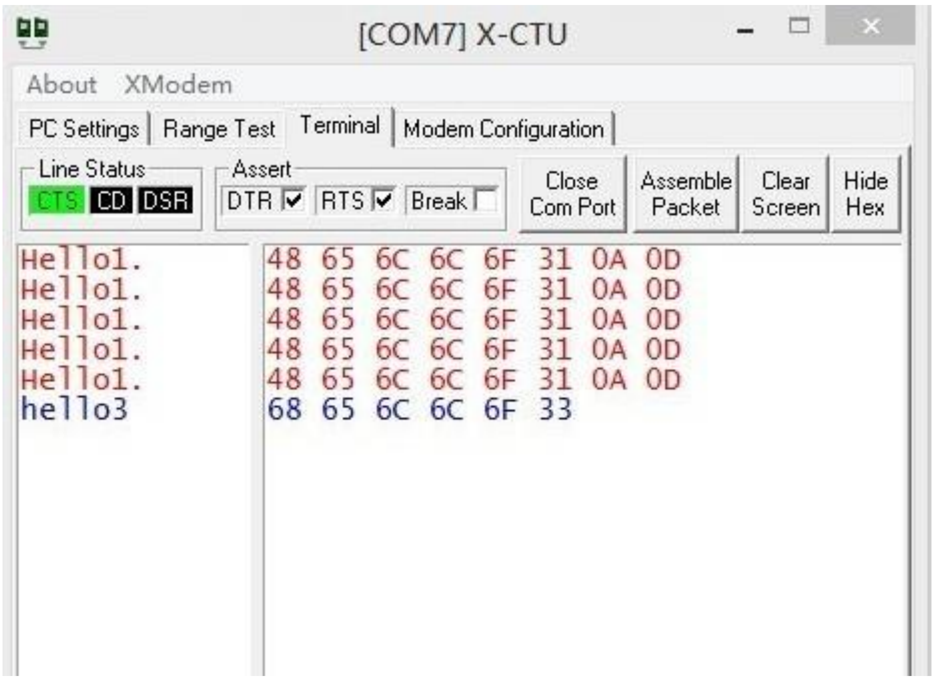
```
pi@raspberrypi ~/BarryTest $ ./a.out



 5
+++ 3
 6atmy1
 5wr
 9d1FFFF
 5wr
 5cn
Hello1
 8
Hello1
 8
Hello1
 8
Hello1
 8
Hello1
 8
OK_length=18 rx_buffer=OK
rx_length=6 rx_buffer=hello2
```

Figure 23. Node 1 send “Hello1” and receive “hello2”



```
....                FF 0A 0A 0A 0D
                    0D
Hello1.             48 65 6C 6C 6F 31 0A 0D
Hello1.             48 65 6C 6C 6F 31 0A 0D
Hello1.             48 65 6C 6C 6F 31 0A 0D
Hello1.             48 65 6C 6C 6F 31 0A 0D
Hello1.             48 65 6C 6C 6F 31 0A 0D
hello2              68 65 6C 6C 6F 32
```

Figure 24. Node 2 receive “Hello1” and send “hello2”



```
Hello1.     48 65 6C 6C 6F 31 0A 0D
Hello1.     48 65 6C 6C 6F 31 0A 0D
Hello1.     48 65 6C 6C 6F 31 0A 0D
Hello1.     48 65 6C 6C 6F 31 0A 0D
Hello1.     48 65 6C 6C 6F 31 0A 0D
hello3      68 65 6C 6C 6F 33
```

Figure 25. Node 3 receive "Hello1" and send "hello3"

Both the second and third node can receive "Hello1" from the first XBee RF module, but for the first node, it can only receive data coming from the second node. That is because the DL of the second one is the same as the MY address in the first one. In the AT protocol, one node can only send data to another one whose MY address is the same as the DL address of the sender.

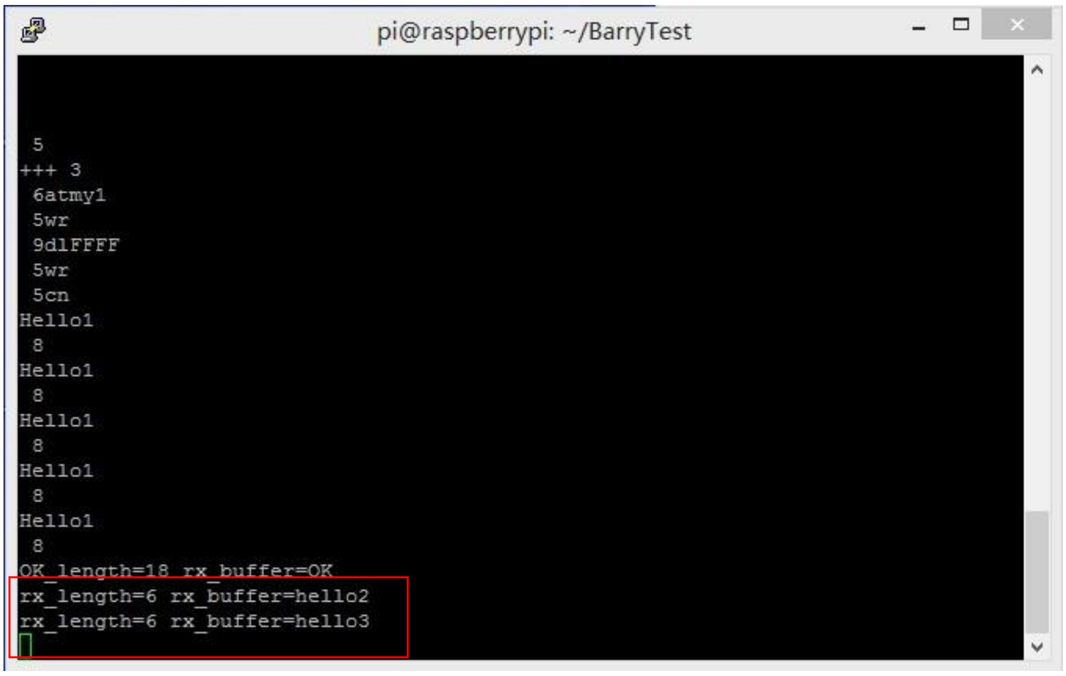
5.1.4 Mesh Network Implementation case 4

Finally, the DL of the third XBee RF module was changed to "1", and it was tested if it can be received by the first module. Table 4 below lists the configurations in case 4.

Table 4. Mesh Network Configuration Case 4

The First module	The Second module	The Third module
ID: 3001	ID: 3001	ID: 3001
DH: 0	DH: 0	DH: 0
DL: FFFF	DL: 1	DL: 1
MY: 1	MY: 2	MY: 3
Message sending: Hello1	Message sending: hello2	Message sending: hello3

The result of this configuration is shown in Figure 26. This time, Raspberry Pi can receive “hello2” and “hello3” from node 2 and node 3 respectively.



```
 5
+++ 3
 6atmy1
 5wr
 9dlFFFF
 5wr
 5cn
Hello1
 8
Hello1
 8
Hello1
 8
Hello1
 8
Hello1
 8
OK length=18 rx_buffer=OK
rx_length=6 rx_buffer=hello2
rx_length=6 rx_buffer=hello3
```

Figure 26. Node 1 receive “hello2” and “hello3”

This time, the first XBee RF module can receive data from the second one and the third one. Of course, both the second one and the third one can receive data coming from the first one.

6 CONCLUSIONS

In summary, the wireless mesh network was built successfully in a basic way. There are three nodes in this network and the communication between each two of them works very well. In the broadcast mode, the other two nodes can receive data from the first node. If their DL address is the same as the MY address of the node in the broadcast mode, both of them can transmit data to this broadcast node.

This project requires very good programming skill in the C language. Concentrating on the basic theory of UART serial communication is also very important.

The advantage of this project is that the wireless network communication between more than two nodes provides people more than one way to communicate with one another. If one of the nodes in this network is down, the communication still can be built with other nodes. So it reduces the possibility of interruption in wireless communication and improves the reliability of this network.

There are several aspects of future improvement for this application. Firstly, the AT Mode could be changed to the API Mode, the latter one is more suitable for a mesh network. In the unicast communication, if the AT Mode is used, more energy will be consumed. In addition, the code in Raspberry Pi could be changed so it would tell the user which node is sending instead of entering into different contents to distinguish different nodes.

REFERENCES

/1/ Durda IV, F. 1996. Serial and UART Tutorial. Accessed 04.05.2014. https://www.freebsd.org/doc/en/articles/serial-uart/

/2/ Eady, F. 2007. The XBee ZigBee Module. Hands-on ZigBee Implementing 802.15.4 with Microcontrollers. 2007 Elsevier Inc.

/3/ Gislason, D. 2008. ZigBee Is Highly Reliable. ZIGBEE WIRELESS NETWORKING. 2008 Elsevier Inc. on pp. 4-5.

/4/ Hale, T. B. 2012. The Raspberry Pi Hobbyist. Accessed 02.05.2014. URL:http://raspberrypihobbyist.blogspot.fi/2012/08/raspberry-pi-serial-port.html

/5/ Hebel, M., Bricker, G., Harris, D. Getting started with XBee RF modules. Accessed 02.05.2014. http://www.makershed.com/v/vspfiles/assets/images/122-32450-xbeetutorial-v1.0.1.pdf

/6/ Mesh with Xbee. Accessed 02.06.2014. http://xbee.wikispaces.com/Mesh+with+Xbee

/7/ PuTTY. Accessed 29.4.2014.

http://www.bu.edu/tech/support/desktop/software/windows/gsputty/

/8/ Raspberry Pi . Accessed 27.4.2014. http://www.raspberrypi.org/help/what-is-a-raspberry-pi/

/9/ Stevanovic, D. 2007. ZigBee IEEE 802.15.4 Standard. Accessed 23.3.2014. http://www.cse.yorku.ca/~dusan/Zigbee-Standard-Talk.pdf

/10/ Vachirapol M., Narisorn L., Thanachai P., Somsak K. Mesh Networking with XBee. Accessed 29.4.2014. http://www2.siit.tu.ac.th/somsak/pub/final_ZBNetwk_100328.pdf

/11/ WinSCP. Accessed 29.4.2014.http://winscp.net/eng/docs/introduction

/12/ XBee Adapter. Accessed 14.05.2014. http://www.instructables.com/id/XBee-adapter/

/13/ XBee/ XBee-PRO ZB RF Modules. 2012 Digi International, Inc. Accessed 02.06.2014. http://www.adafruit.com/datasheets/XBee%20ZB%20User%20Manual.pdf

/14/ X-CTU(XCTU) software. Accessed 29.4.2014.
http://www.digi.com/support/kbase/kbaseresultdetl?id=2125

/15/ ZigBee Protocol Layer. Accessed 14.05.2014.
http://commons.wikimedia.org/wiki/File:ZigBee_protocol_stack.png