Practical Guide For Implementing Thread Protocol

Anshul Kothari, anshulkothari96@gmail.com

June 30, 2017

Contents

1 Introduction

1.1 What is Internet of Things?

The Term Internet of Things is composed of two words Internet and Things, where Things means our daily routine items such as refrigerator, ceiling fans, Light Bulbs, Television, meter etc, given computing power i.e. the ability to make decisions and Internet refers to connectivity of this objects. Thus Internet of Things means connection of this smart objects.

Figure 1: Connection of Smart Devices

1.2 Applications of Internet of Things

There are myriads of application of Internet of Things, few of them are as follows:

1.2.1 Detection of Forest Fire

For example if we place large number of temperature sensors having computing power and wireless radio, at different locations in the forest, then if there is increase in the temperature in particular region, sensor at that location will send warning signal over the air, and thus massive destruction can be prevented.

1.2.2 Flood Detection

we can deploy wireless sensor network which detects weather conditions, rainfall, water level etc and sends the warning signal, so that preventive steps can be taken.

1.2.3 Smart Home

Connecting everything in your home and letting them talk to each other will bring happiness. For example what if your refrigerator can place order for the items that need to be purchased.

1.3 Comparison of Different Protocol

There are many protocols out there in the market such as Bluetooth, ZigBee, and Thread etc. The comparison of the protocols on the basis of the layer in which they operate is shown in the figure.

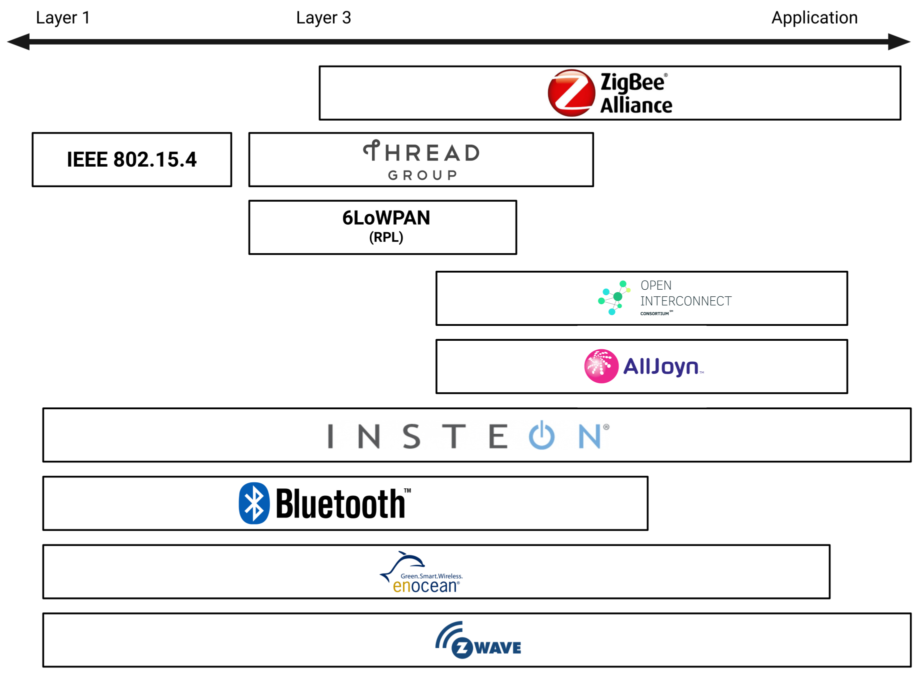
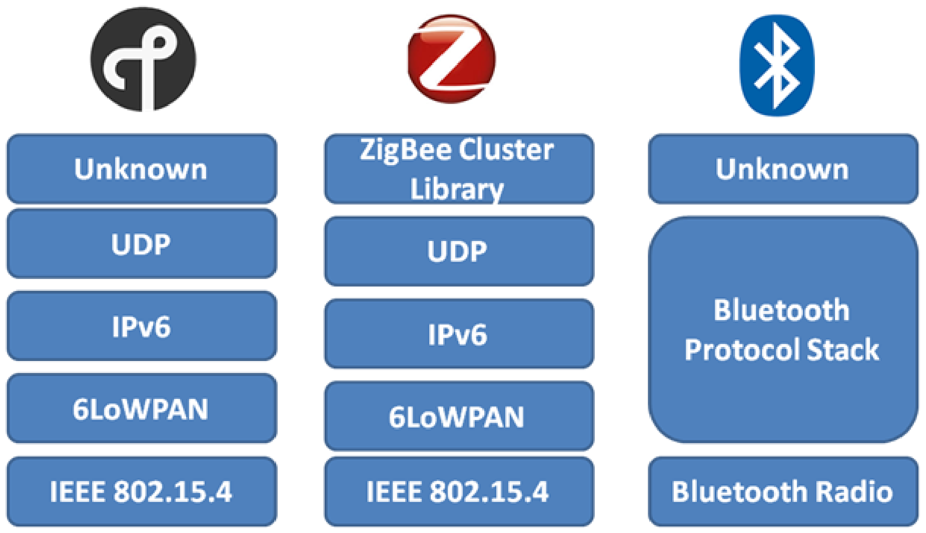


Figure 2: Comparison of Different Protocols

1.4 Why Thread Protocol?



It turns out that ZigBee and Thread protocol are similar in their architecture, both use IEEE 802.15.4 at Physical Layer/MAC Layer and 6LoWPAN at the networking Layer. But there is subtly differences between the two. Mechanisms which are NOT defined in 6LoWPAN standard:
1) Key exchange mechanism
2) Mesh Routing
3) Neighbour Discovery
4) Border Router Behaviour
Now though Thread and ZigBee both are implementing 6LoWPAN standard, both the protocols implement the aforementioned mechanisms differently. For example Thread uses MLE neighbour discovery protocol, while ZigBee uses RPL.

1.5 What is Thread Protocol?

1. Thread is the networking layer protocol for mesh network, specifically designed for Internet of Things.

2. It is based on IEEE 802.15.4

3. Efficient

4. Reliable and Secure

5. Clean and Less Complex Stack

1.6 Type of Devices in the Thread Network

There are basically two types of devices in Thread network:

1.6.1 Always Powered on Devices

Such as refrigerator, meter etc. They can be used as routers.

(a) Power Meter

(b) Refrigerator

1.6.2 Occasionally Powered on Devices

Such as light bulb, ceiling fan etc. Also known as Sleepy Devices Routers are further classified as Leader Router, Border Router based on their functionality.

(a) Toaster

(b) Light Bulb

(c) Ceiling Fan

1.7 Basic Thread Network Structure

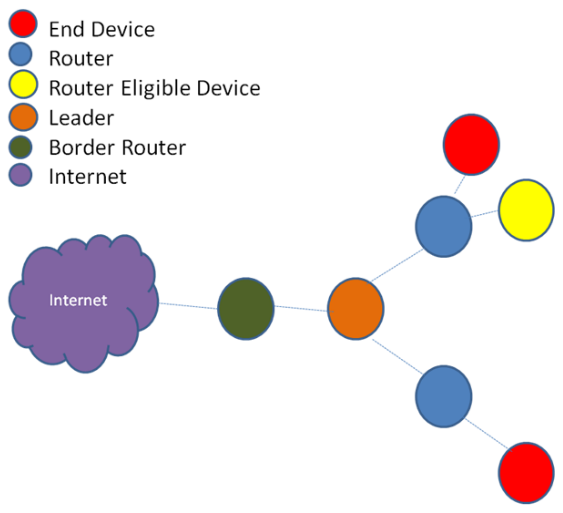


Figure 5: Structure of Thread Network

1.7.1 End Device

This device does not participate in the routing of packets through the Thread network. They communicate with other nodes in the Thread network through their parent router.

1.7.2 Active Router

This devices are responsible for routing packets in the Thread Network.

1.7.3 Router Eligible Devices (REED)

This devices are similar to end devices but can become routers when network needs them.

1.7.4 Leader

Leader Router is responsible for the following:

1. Assigning Address to the Routers
2. Decision Making in the Thread Network
3. Transferring Network Data

1.7.5 Border Router

Border Router is responsible for the following:

1. Connecting Thread network with Internet

2. External Commissioning

3. Assigning Global Unicast Address

1.8 Types of Address Assigned to the node in Thread network

1.8.1 Link Local Address

This address is used to communicate with directly connected node or node at single hop distance.

1.8.2 Mesh Local Address

This is address is used to communicate with any other node in the Thread network.

1.8.3 Local Multicast Address

This address is used to send multicast messages to the node directly connected sender node

1.8.4 Mesh Local Multicast Address

This address is used to send multicast messages to all the nodes in the Thread network.

1.8.5 Global Unicast Address

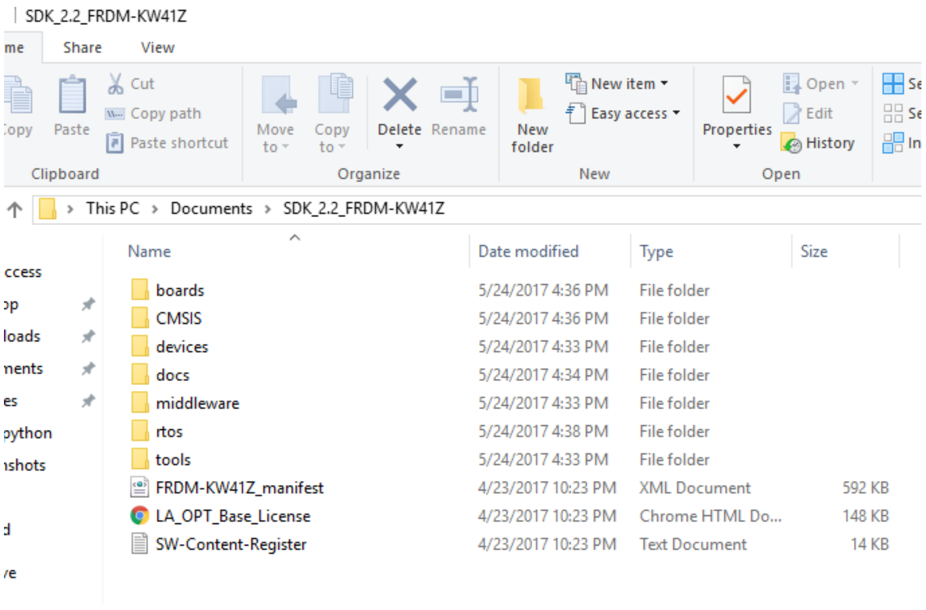
This is used to communicate with remote node (Node connected to the Internet)

2 Play With NXP FRDM KW41Z Boards

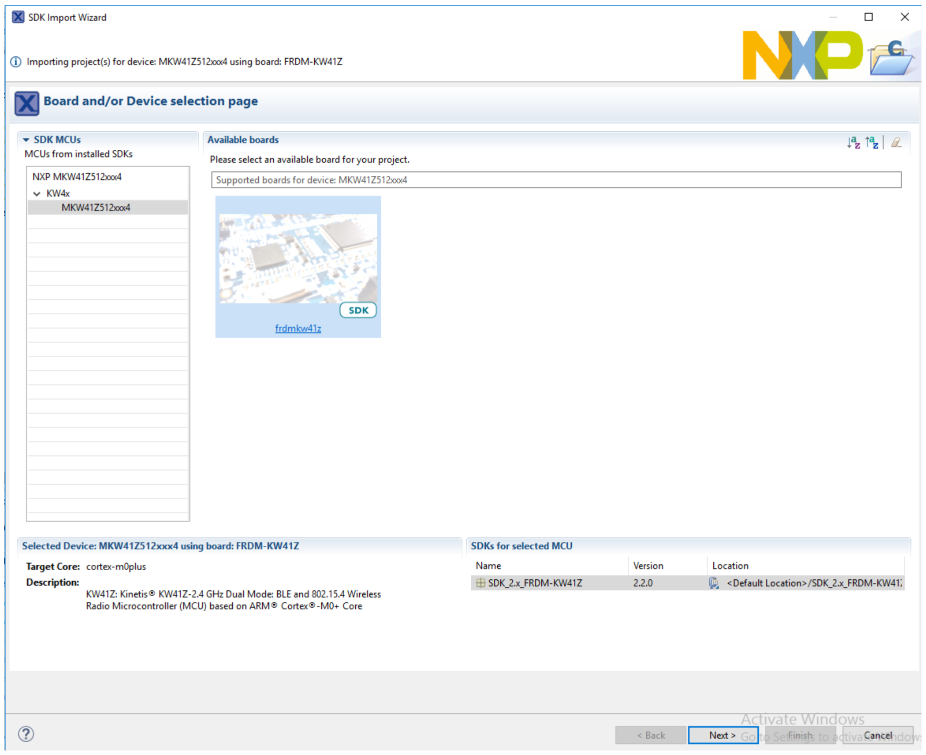
2.1 Environment Setup

Operating System Used: Windows 10
Steps to be followed:

1. Download and Install MCUXpresso IDE from http://www.nxp.com/products/software-and-tools/hardware-development-tools/freedom-development-boards/nxp-freedom-development-kit-for-kinetis-kw41z-31z-21z-mcus:FRDM-KW41Z?tab=In-Depth_Tab
2. Drag and Drop SDK folder in the MCUXpress IDE Installed SDK window as shown in the figure.
3. Download and Install Terminal Application such as Putty (for Windows) from https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html



Name	Date modified	Type	Size
boards	5/24/2017 4:36 PM	File folder	
CMSIS	5/24/2017 4:36 PM	File folder	
devices	5/24/2017 4:33 PM	File folder	
docs	5/24/2017 4:34 PM	File folder	
middleware	5/24/2017 4:33 PM	File folder	
rtos	5/24/2017 4:38 PM	File folder	
tools	5/24/2017 4:33 PM	File folder	
FRDM-KW41Z_manifest	4/23/2017 10:23 PM	XML Document	592 KB
LA_OPT_Base_License	4/23/2017 10:23 PM	Chrome HTML Do...	148 KB
SW-Content-Register	4/23/2017 10:23 PM	Text Document	14 KB



Importing project(s) for device: MKW41Z512xxx4 using board: FRDM-KW41Z

Board and/or Device selection page

SDK MCUs

MCUs from installed SDKs

NXP MKW41Z512xxx4
KW4x
MKW41Z512xxx4

Available boards

Please select an available board for your project.

Supported boards for device: MKW41Z512xxx4

frdmkw41z

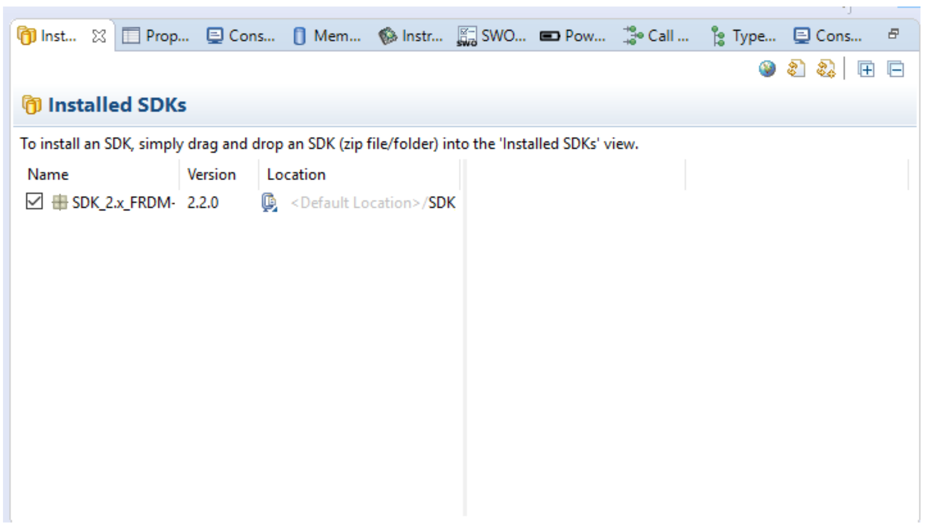
Selected Device: MKW41Z512xxx4 using board: FRDM-KW41Z

Target Core: cortex-m0plus

Description: KW41Z: Kinetis® KW41Z-2.4 GHz Dual Mode: BLE and 802.15.4 Wireless Radio Microcontroller (MCU) based on ARM® Cortex®-M0+ Core

SDKs for selected MCU

Name	Version	Location
SDK_2.x_FRDM-KW41Z	2.2.0	<Default Location>/SDK_2.x_FRDM-KW41

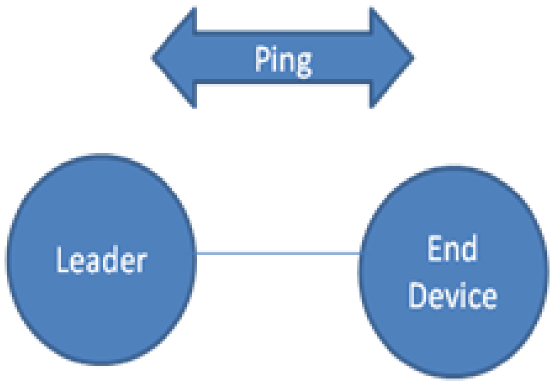


Installed SDKs

To install an SDK, simply drag and drop an SDK (zip file/folder) into the 'Installed SDKs' view.

Name	Version	Location
☑ SDK_2.x_FRDM-	2.2.0	<Default Location>/SDK

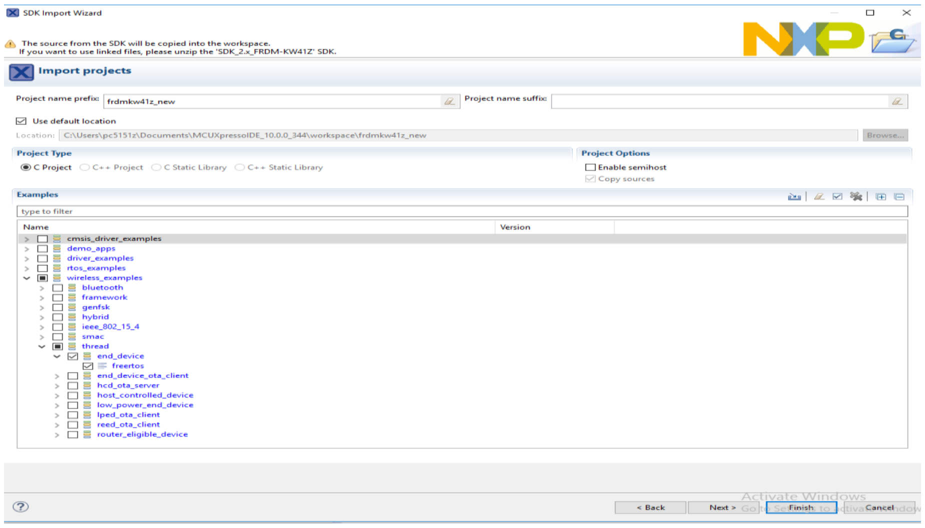
2.2 Creating Two Board Network



Steps to be followed:

1. Open MCUXpresso IDE
2. In the Quick Start panel click on import SDK examples
3. Select the Target as KW41Z, click next
4. Select wireless_examples− >thread− >end_device− >freetos, also uncheck the enable semi hosting check box
5. Click next
6. Uncheck the redirect printf check box
7. Right Click on the project and click build project
8. Select your project in project explorer view and click debug in the quick start panel
9. If you get an error then try to right click on the project and click clean project and build and debug project again.
10. Repeat the procedure for flashing Router Eligible Device Application on another board.
11. Open putty session for both of your board
12. Create the thread network from REED terminal thr create
13. Join the End device to the Thread network thr join
14. Use ifconfig on both the terminal to get the IP address of the nodes

15. Try to ping devices in the network Kudos! You have Established communication between two boards


SDK Import Wizard

The source from the SDK will be copied into the workspace.
If you want to use linked files, please unzip the 'SDK_2.x_FRDM-KW41Z' SDK.

Import projects

Project name prefix: frdmkw41z_new
Project name suffix:

☑ Use default location

Location: C:\Users\pc5151z\Documents\MCUXpressoIDE_10.0.0_344\workspace\frdmkw41z_new

Project Type

[✓ C Project] C++ Project C Static Library C++ Static Library

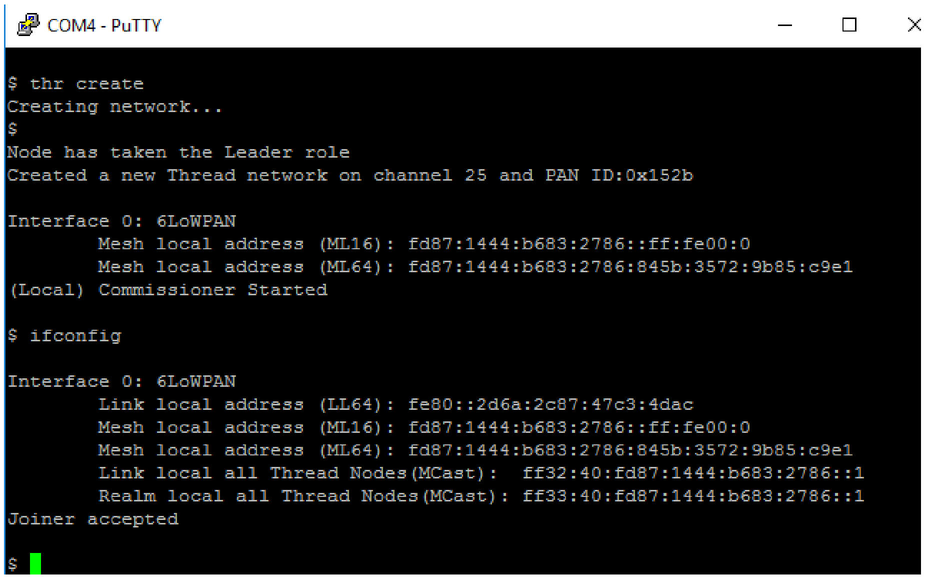
Project Options

☐ Enable semihost
☑ Copy sources

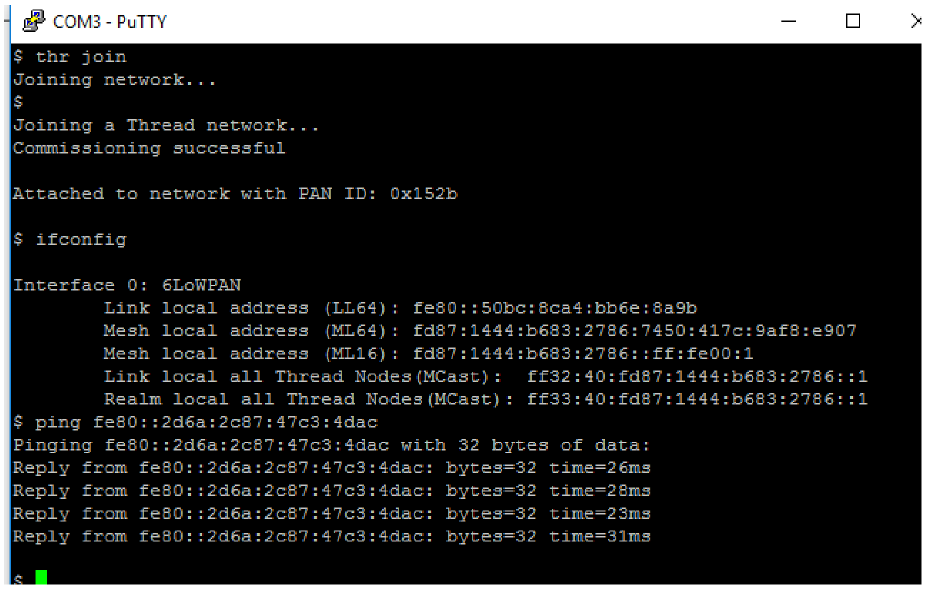
Examples

type to filter

Name	Version
☐ cmsis_driver_examples	
☐ demo_apps	
☐ driver_examples	
☐ rtos_examples	
▣ wireless_examples	
☐ bluetooth	
☐ framework	
☐ genfsk	
☐ hybrid	
☐ ieee_802_15_4	
☐ smac	
▣ thread	
☑ end_device	
☑ freertos	
☐ end_device_ota_client	
☐ hcd_ota_server	
☐ host_controlled_device	
☐ low_power_end_device	
☐ lped_ota_client	
☐ reed_ota_client	
☐ router_eligible_device	


COM4 - PuTTY

```
$ thr create
Creating network...
$
Node has taken the Leader role
Created a new Thread network on channel 25 and PAN ID:0x152b

Interface 0: 6LoWPAN
        Mesh local address (ML16): fd87:1444:b683:2786::ff:fe00:0
        Mesh local address (ML64): fd87:1444:b683:2786:845b:3572:9b85:c9e1
(Local) Commissioner Started

$ ifconfig

Interface 0: 6LoWPAN
        Link local address (LL64): fe80::2d6a:2c87:47c3:4dac
        Mesh local address (ML16): fd87:1444:b683:2786::ff:fe00:0
        Mesh local address (ML64): fd87:1444:b683:2786:845b:3572:9b85:c9e1
        Link local all Thread Nodes(MCast):  ff32:40:fd87:1444:b683:2786::1
        Realm local all Thread Nodes(MCast): ff33:40:fd87:1444:b683:2786::1
Joiner accepted

$
```

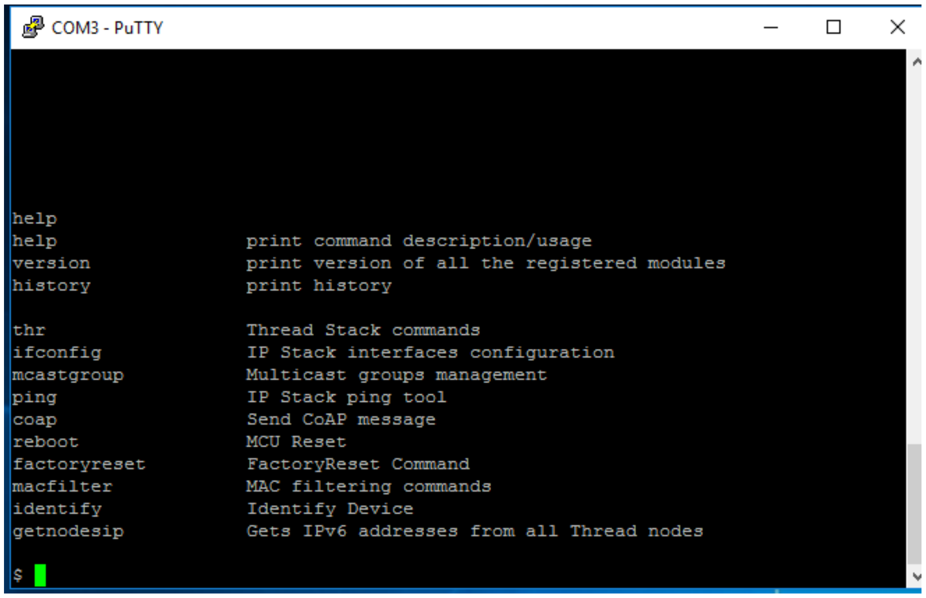


```
$ thr join
Joining network...
$
Joining a Thread network...
Commissioning successful

Attached to network with PAN ID: 0x152b

$ ifconfig

Interface 0: 6LoWPAN
        Link local address (LL64): fe80::50bc:8ca4:bb6e:8a9b
        Mesh local address (ML64): fd87:1444:b683:2786:7450:417c:9af8:e907
        Mesh local address (ML16): fd87:1444:b683:2786::ff:fe00:1
        Link local all Thread Nodes(MCast):  ff32:40:fd87:1444:b683:2786::1
        Realm local all Thread Nodes(MCast): ff33:40:fd87:1444:b683:2786::1
$ ping fe80::2d6a:2c87:47c3:4dac
Pinging fe80::2d6a:2c87:47c3:4dac with 32 bytes of data:
Reply from fe80::2d6a:2c87:47c3:4dac: bytes=32 time=26ms
Reply from fe80::2d6a:2c87:47c3:4dac: bytes=32 time=28ms
Reply from fe80::2d6a:2c87:47c3:4dac: bytes=32 time=23ms
Reply from fe80::2d6a:2c87:47c3:4dac: bytes=32 time=31ms

$
```

Example Explained

- The execution of the Application starts from main() in the file framework/OSAbstraction/Source/fsl_os_abstraction_free_rtos.c (Note: we are using FREE RTOS for real time scheduling) Main() − > startup_task() − > main_task() Main_task() is in nwk_ip/app/common/app_init.c
- Main_task() calls the respective functions to initialize different tasks such as LED, MAY, PHY , Key Board etc.
- Main_task() then calls APP_Init(), which initializes our application (that is create task and initialize its parameters)
- Main_task() then jumps into infinite loop and calls APP_Handler(), which handles any event occurred and calls the respective handler.
- In source/router_eligible_device_app.c there are different types of
- Event handlers such as Key Board handler, commissioning event handler, joining event handler etc (You can modify event handler according to your need, for example if you push button then what should happen?)
- Use pton() to convert the IP address string to ipAddr_t type o pton(AF_INET6, "fdc3:8ac1:b6ac:dffe:985d:fb14:f4ef:28ac",&gCoapDestAddress)
- Use shell_write("your_string) to print string on the shell, It is very useful to debug your application

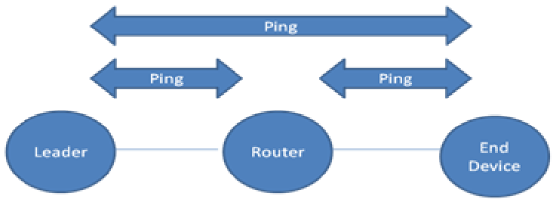
2.3 Using Shell (Command Line Interface)



```
help
help                    print command description/usage
version                 print version of all the registered modules
history                 print history

thr                     Thread Stack commands
ifconfig                IP Stack interfaces configuration
mcastgroup              Multicast groups management
ping                    IP Stack ping tool
coap                    Send CoAP message
reboot                  MCU Reset
factoryreset            FactoryReset Command
macfilter               MAC filtering commands
identify                Identify Device
getnodesip              Gets IPv6 addresses from all Thread nodes

$
```

Here is the brief description of the often used shell commands.
help: To get help for the commands available in the shell and their parameters
ifconfig: To get the IP address assigned to the node
coap: Used to send COAP messages between nodes in the thread network.
factoryreset: To reset the device
identify: Used to identify the device in the mesh network (The device on which you run the command will start blinking led)

2.4 Creating three or more Board Thread Network

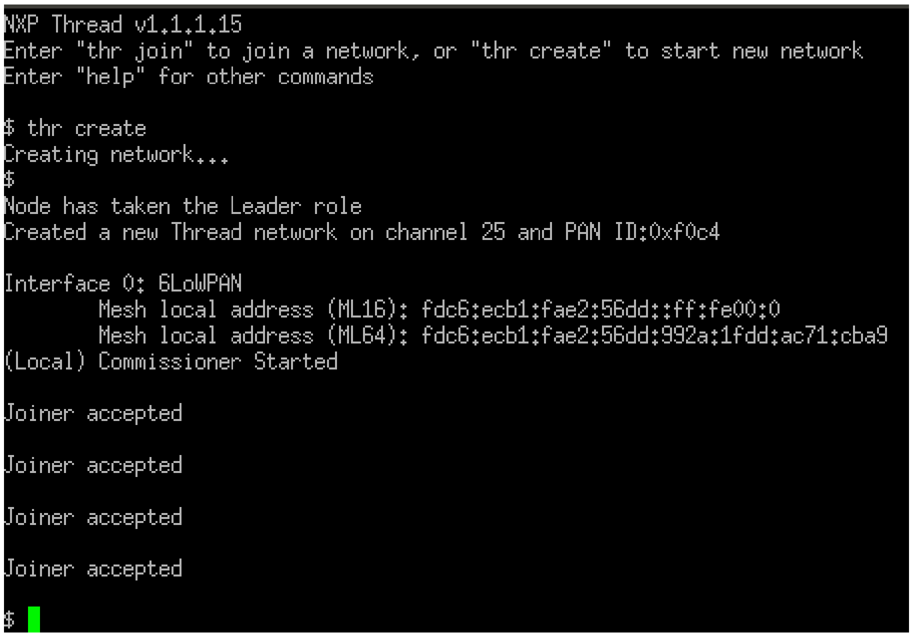


Steps to be followed:

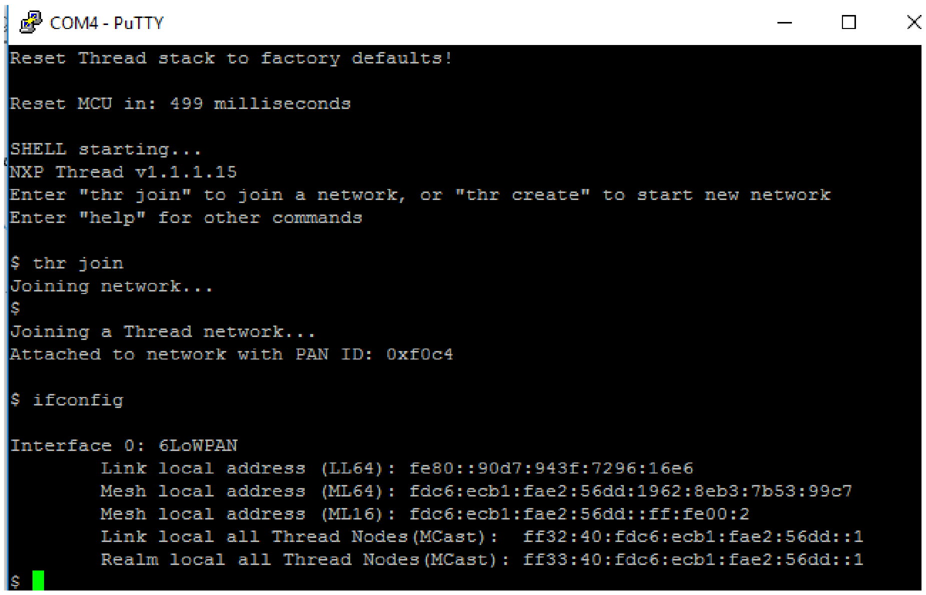
1. Connect board to the PC
2. Flash one by one the example application on three boards as mentioned in section 1.5 (End_Device Application on one board and REED Application on other two devices)

3. Check for the USB port assigned to your boards (In the device− > Device Manager − > Port) such as COM3 or COM8 etc
4. Open the Putty Application
5. Configure the Parameters and open three different sessions using all the three boards simultaneously
6. Keep the arrangement of the boards as shown in the figure (So that we can test for Range and routing capability of Thread Network)
7. Now Create thread network using shell of one of the REED
   thr create
8. Join other REED into thread network using shell
   thr join
9. Join the End Device to the Thread network using its shell
   thr join
10. Try to ping one device from another using ping command, use ifconfig command to get the ip address of the node
11. Try to remove Router (not leader router) from the network and see the effect! (End device will quickly gets connected to leader router)
12. Try to remove Leader from the network and see the effect (Network will self heal, Router will become Leader)

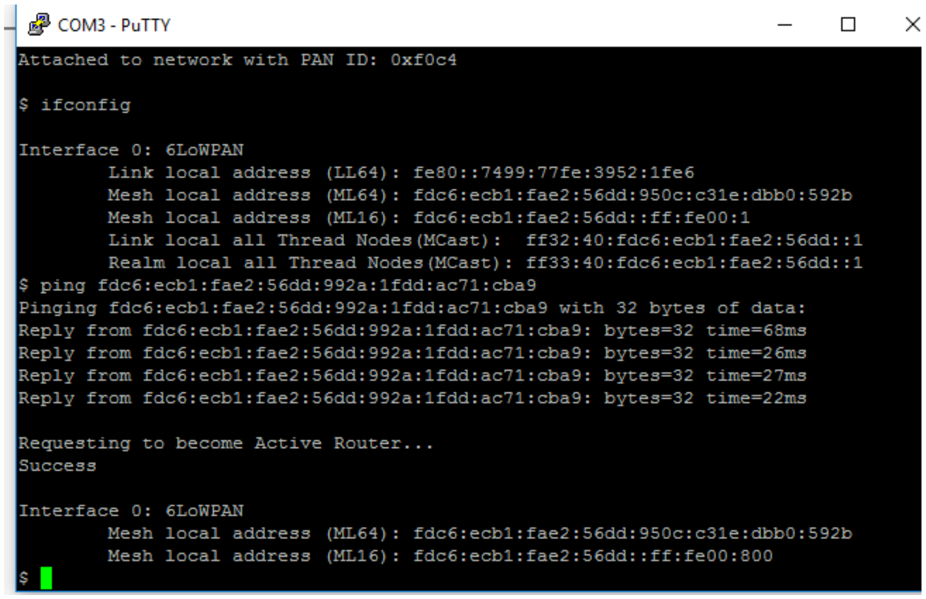
Kudos! You have tested and analyzed thread protocol on mesh network



```
NXP Thread v1.1.1.15
Enter "thr join" to join a network, or "thr create" to start new network
Enter "help" for other commands

$ thr create
Creating network...
$
Node has taken the Leader role
Created a new Thread network on channel 25 and PAN ID:0xf0c4

Interface 0: 6LoWPAN
        Mesh local address (ML16): fdc6:ecb1:fae2:56dd::ff:fe00:0
        Mesh local address (ML64): fdc6:ecb1:fae2:56dd:992a:1fdd:ac71:cba9
(Local) Commissioner Started

Joiner accepted

Joiner accepted

Joiner accepted

Joiner accepted

$
```

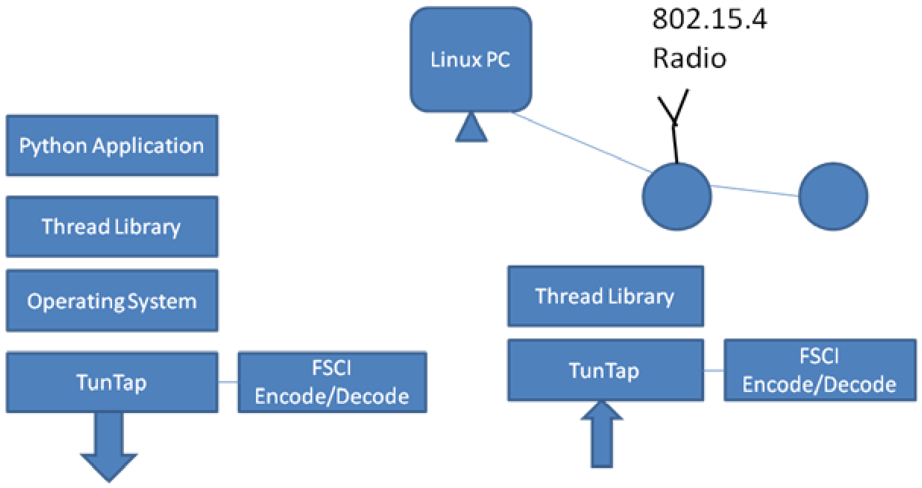


```
Reset Thread stack to factory defaults!

Reset MCU in: 499 milliseconds

SHELL starting...
NXP Thread v1.1.1.15
Enter "thr join" to join a network, or "thr create" to start new network
Enter "help" for other commands

$ thr join
Joining network...
$
Joining a Thread network...
Attached to network with PAN ID: 0xf0c4

$ ifconfig

Interface 0: 6LoWPAN
        Link local address (LL64): fe80::90d7:943f:7296:16e6
        Mesh local address (ML64): fdc6:ecb1:fae2:56dd:1962:8eb3:7b53:99c7
        Mesh local address (ML16): fdc6:ecb1:fae2:56dd::ff:fe00:2
        Link local all Thread Nodes(MCast):  ff32:40:fdc6:ecb1:fae2:56dd::1
        Realm local all Thread Nodes(MCast): ff33:40:fdc6:ecb1:fae2:56dd::1
$
```



```
Attached to network with PAN ID: 0xf0c4

$ ifconfig

Interface 0: 6LoWPAN
        Link local address (LL64): fe80::7499:77fe:3952:1fe6
        Mesh local address (ML64): fdc6:ecb1:fae2:56dd:950c:c31e:dbb0:592b
        Mesh local address (ML16): fdc6:ecb1:fae2:56dd::ff:fe00:1
        Link local all Thread Nodes(MCast):  ff32:40:fdc6:ecb1:fae2:56dd::1
        Realm local all Thread Nodes(MCast): ff33:40:fdc6:ecb1:fae2:56dd::1
$ ping fdc6:ecb1:fae2:56dd:992a:1fdd:ac71:cba9
Pinging fdc6:ecb1:fae2:56dd:992a:1fdd:ac71:cba9 with 32 bytes of data:
Reply from fdc6:ecb1:fae2:56dd:992a:1fdd:ac71:cba9: bytes=32 time=68ms
Reply from fdc6:ecb1:fae2:56dd:992a:1fdd:ac71:cba9: bytes=32 time=26ms
Reply from fdc6:ecb1:fae2:56dd:992a:1fdd:ac71:cba9: bytes=32 time=27ms
Reply from fdc6:ecb1:fae2:56dd:992a:1fdd:ac71:cba9: bytes=32 time=22ms

Requesting to become Active Router...
Success

Interface 0: 6LoWPAN
        Mesh local address (ML64): fdc6:ecb1:fae2:56dd:950c:c31e:dbb0:592b
        Mesh local address (ML16): fdc6:ecb1:fae2:56dd::ff:fe00:800
$
```

2.5 Implementing Border Router using Linux PC



Steps need to be followed:

1. Flash the host_controlled_device application into one of the NXP board (You need to configure macro, goto source– >config.h
#define THR_SERIAL_TUN_ROUTER 1

2. Copy the folder in windows PC /SDK_2.2_FRDM-KW41Z/tools/wireless/host_sdk to your linux PCs Desktop

3. Open new Terminal and chage directory to host_sdk folder
cd Desktop/host_sdk/hsdk

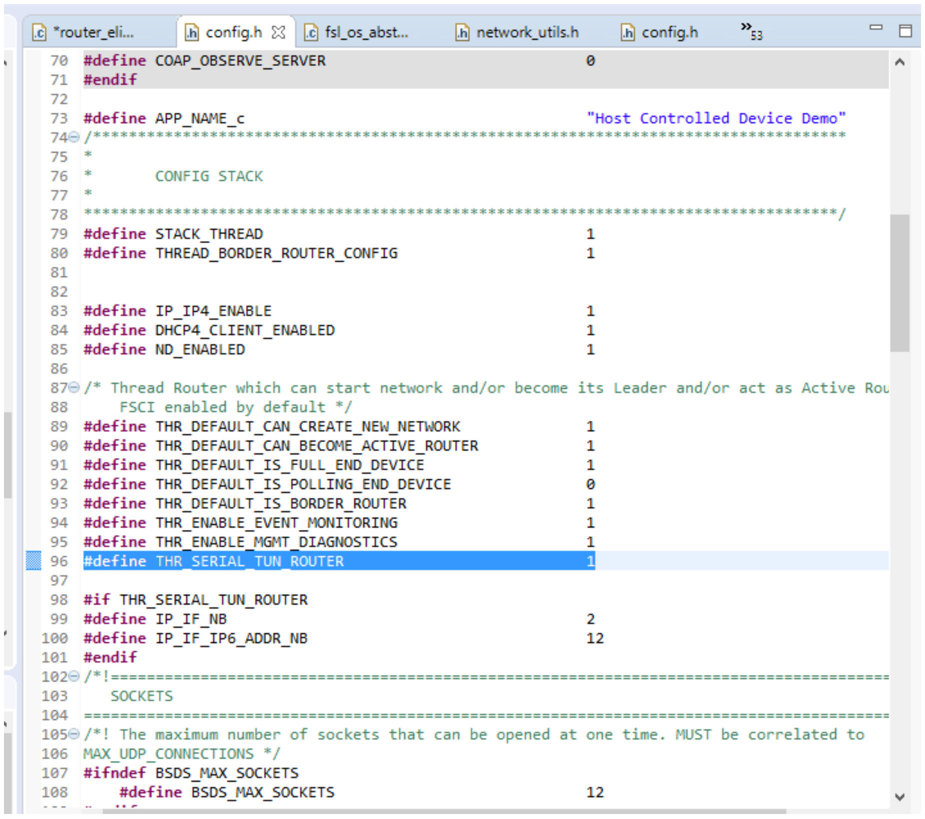
4. Install dependencies
sudo apt-get install build-essential
sudo apt-get install libudev-dev
sudo apt-get install libpcap-dev

5. Generate shared libraries, files will be generated in the host_sdk/hsdk/build directory
make

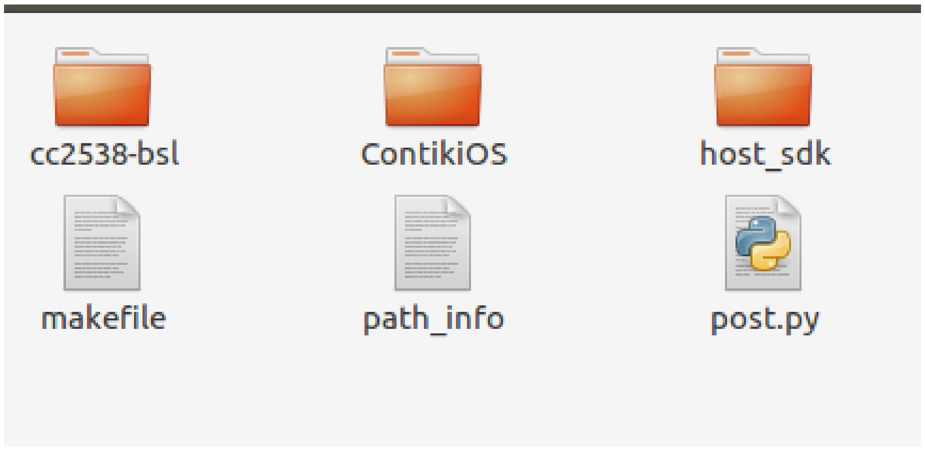
6. Move the files to /usr/local/lib
sudo make install PREFIX=/usr/local/lib

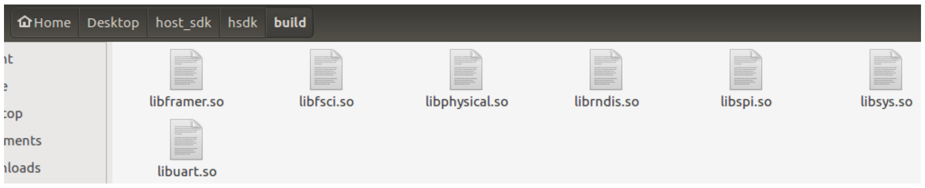
7. Change directory to demo
cd demo

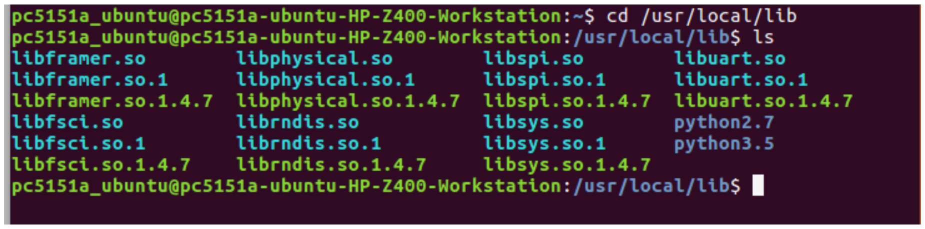
8. Compile the demo applications
   ```
   make
   make spi
   ```
9. Check the files in bin folder
   ```
   ls bin/
   ```
10. Test the GetKinetisDevices Application, Plug your device to linux PC
    ```
    cd bin
    ./GetKinetisDevices
    ```
11. Creating a virtual interface using tun/tap, open the new terminal and run the following command in the command prompt
    ```
    sudo ip tuntap Add mode tun fslthr0
    sudo ip -6 addr add fd01::2 dev fslthr0
    sudo ip -6 route add fd01::1 dev fslthr0
    sudo ip -6 route add fd01:0000:0000:3ead::/64 dev fslthr0
    sudo ip link set fslthr0 up
    sudo sysctl w net.ipv6.conf.all.forwarding=1
    sudo ip -6 route show
    ```
12. Open putty session on the end device and use `ifconfig` to get the IP address of the node
13. Try pinging the end device from Linux PC using the following commands
    ```
    cd hskd/demo/bin ./Thread_KW_Tun /dev/ttyACM0 fslthr0
    ```
14. Open another terminal in linux PC `ping6 c 1 fd01::3ead:7c16:37b5:e6ef:701a` (Change with your Unicast Global Address)
    Kudos! You have implemented your Border Router, now you can connect your node to internet, YeaH



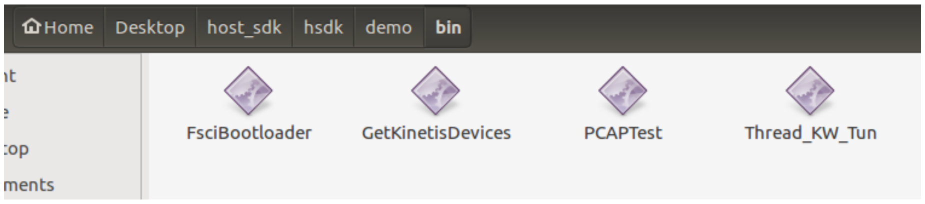
```
#define COAP_OBSERVE_SERVER                                        0
#endif

#define APP_NAME_c                                                 "Host Controlled Device Demo"
/*****************************************************************************************
*
*       CONFIG STACK
*
*****************************************************************************************/
#define STACK_THREAD                                               1
#define THREAD_BORDER_ROUTER_CONFIG                                1


#define IP_IP4_ENABLE                                              1
#define DHCP4_CLIENT_ENABLED                                       1
#define ND_ENABLED                                                 1

/* Thread Router which can start network and/or become its Leader and/or act as Active Rou
    FSCI enabled by default */
#define THR_DEFAULT_CAN_CREATE_NEW_NETWORK                         1
#define THR_DEFAULT_CAN_BECOME_ACTIVE_ROUTER                       1
#define THR_DEFAULT_IS_FULL_END_DEVICE                             1
#define THR_DEFAULT_IS_POLLING_END_DEVICE                          0
#define THR_DEFAULT_IS_BORDER_ROUTER                               1
#define THR_ENABLE_EVENT_MONITORING                                1
#define THR_ENABLE_MGMT_DIAGNOSTICS                                1
#define THR_SERIAL_TUN_ROUTER                                      1

#if THR_SERIAL_TUN_ROUTER
#define IP_IF_NB                                                   2
#define IP_IF_IP6_ADDR_NB                                          12
#endif
/*!==========================================================================================
   SOCKETS
===========================================================================================
/*! The maximum number of sockets that can be opened at one time. MUST be correlated to
MAX_UDP_CONNECTIONS */
#ifndef BSDS_MAX_SOCKETS
    #define BSDS_MAX_SOCKETS                                       12
```

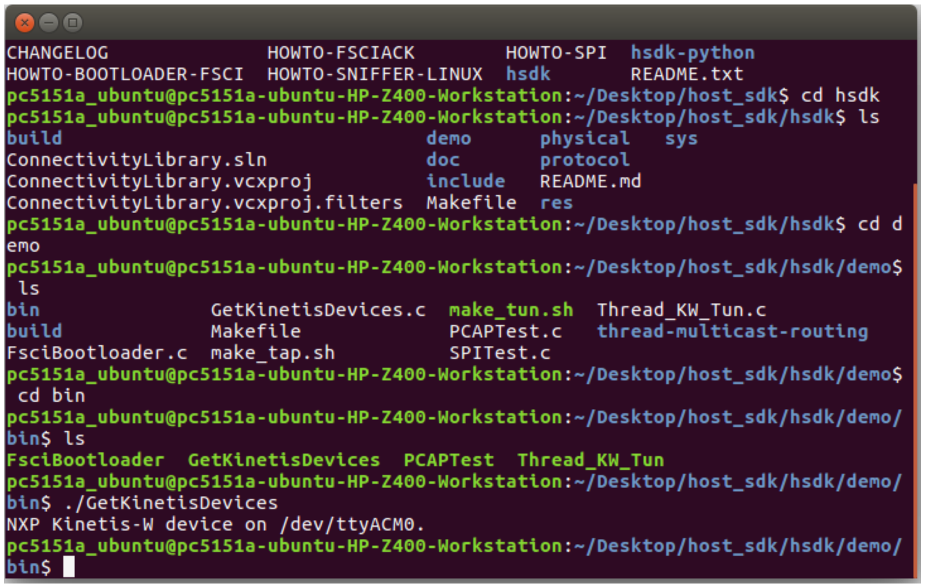




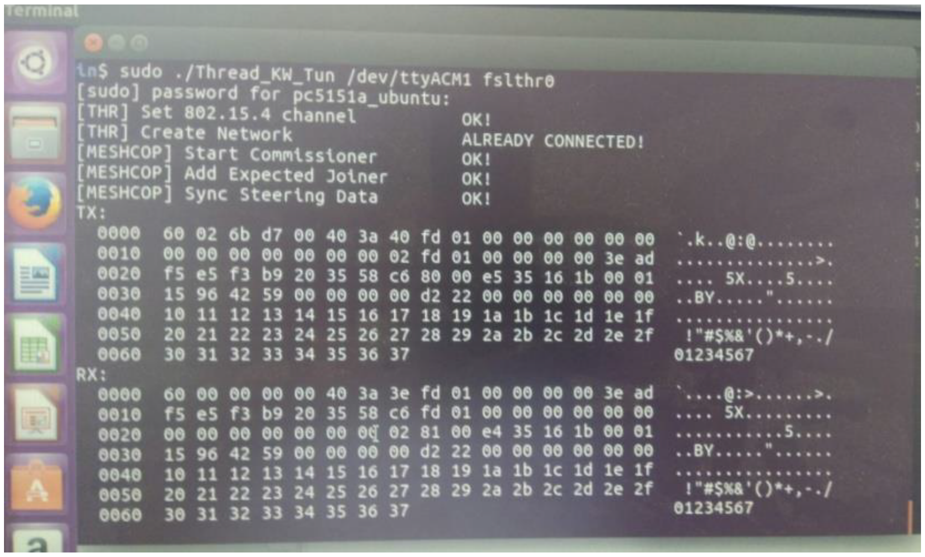


```
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~$ cd /usr/local/lib
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:/usr/local/lib$ ls
libframer.so        libphysical.so        libspi.so        libuart.so
libframer.so.1      libphysical.so.1      libspi.so.1      libuart.so.1
libframer.so.1.4.7  libphysical.so.1.4.7  libspi.so.1.4.7  libuart.so.1.4.7
libfsci.so          librndis.so           libsys.so        python2.7
libfsci.so.1        librndis.so.1         libsys.so.1      python3.5
libfsci.so.1.4.7    librndis.so.1.4.7     libsys.so.1.4.7
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:/usr/local/lib$ 
```





```
CHANGELOG              HOWTO-FSCIACK        HOWTO-SPI  hsdk-python
HOWTO-BOOTLOADER-FSCI  HOWTO-SNIFFER-LINUX  hsdk       README.txt
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Desktop/host_sdk$ cd hsdk
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Desktop/host_sdk/hsdk$ ls
build                                 demo     physical  sys
ConnectivityLibrary.sln               doc      protocol
ConnectivityLibrary.vcxproj           include  README.md
ConnectivityLibrary.vcxproj.filters   Makefile res
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Desktop/host_sdk/hsdk$ cd d
emo
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Desktop/host_sdk/hsdk/demo$
 ls
bin               GetKinetisDevices.c  make_tun.sh  Thread_KW_Tun.c
build             Makefile             PCAPTest.c   thread-multicast-routing
FsciBootloader.c  make_tap.sh          SPITest.c
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Desktop/host_sdk/hsdk/demo$
 cd bin
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Desktop/host_sdk/hsdk/demo/
bin$ ls
FsciBootloader  GetKinetisDevices  PCAPTest  Thread_KW_Tun
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Desktop/host_sdk/hsdk/demo/
bin$ ./GetKinetisDevices
NXP Kinetis-W device on /dev/ttyACM0.
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Desktop/host_sdk/hsdk/demo/
bin$ 
```



```
in$ sudo ./Thread_KW_Tun /dev/ttyACM1 fslthr0
[sudo] password for pc5151a_ubuntu:
[THR] Set 802.15.4 channel              OK!
[THR] Create Network                    ALREADY CONNECTED!
[MESHCOP] Start Commissioner            OK!
[MESHCOP] Add Expected Joiner           OK!
[MESHCOP] Sync Steering Data            OK!
TX:
  0000   60 02 6b d7 00 40 3a 40 fd 01 00 00 00 00 00 00   `.k..@:@........
  0010   00 00 00 00 00 00 00 02 fd 01 00 00 00 00 3e ad   ..............>.
  0020   f5 e5 f3 b9 20 35 58 c6 80 00 e5 35 16 1b 00 01   .... 5X....5....
  0030   15 96 42 59 00 00 00 00 d2 22 00 00 00 00 00 00   ..BY.....".......
  0040   10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
  0050   20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f    !"#$%&'()*+,-./
  0060   30 31 32 33 34 35 36 37                           01234567
RX:
  0000   60 00 00 00 00 40 3a 3e fd 01 00 00 00 00 3e ad   `....@:>......>.
  0010   f5 e5 f3 b9 20 35 58 c6 fd 01 00 00 00 00 00 00   .... 5X.........
  0020   00 00 00 00 00 00 00 02 81 00 e4 35 16 1b 00 01   ...........5....
  0030   15 96 42 59 00 00 00 00 d2 22 00 00 00 00 00 00   ..BY.....".......
  0040   10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
  0050   20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f    !"#$%&'()*+,-./
  0060   30 31 32 33 34 35 36 37                           01234567
```

2.6 Building Application to Test Border Router Functionality

Steps to be followed:

1. Every time you want to run an application using Border router, you need to start your FSCI encoder/decoder cd hsdk/demo/bin ./Thread_KW_Tun /dev/ttyACM0 fslthr0

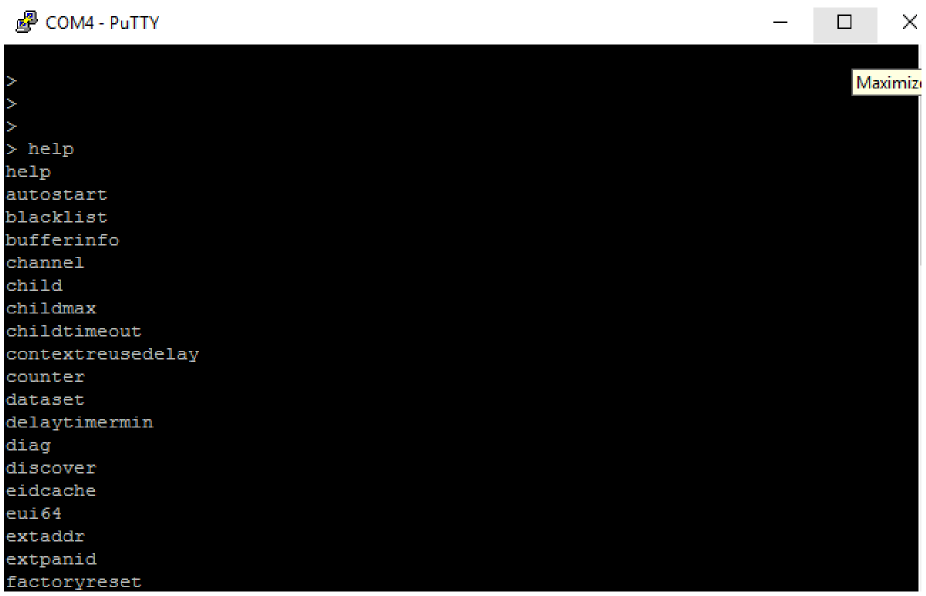
2. Open new terminal in the Linux shell cd host-sdk/hsdk-pyhton/src/com/nxp/woreless_connectivity/test sudo python rgb_led.py fd01::3ead:7ccc:0000:8855:0005(Change the address according to your boards Global unicast address) Kudos! You will now see led flashing on the end device.

3. Exercise Test with temp.py file in the same folder and try to send the data to the server using Ethernet interface (Hint: Collect the data you received on the Tun interface and then send the data to the server script such as PHP, which will store the data to the database).

2.7 Replacing NXP Thread Stack with Open Thread Stack

Note, you have to build your own applications such as LED On/Off using CoAP as Application Layer protocol when using openthread stack, because it is just Networking layer which can be used by IP enable Application layer such as CoAP to send data over 802.15.4 . Operating System Used: Ubuntu 16.04 LTS Steps to be followed:

1. Download and Install GNU toolchain, if not already installed from https://launchpad.net/gcc-arm-embedded
2. Download and install NXP Test Tool fromhttp://www.nxp.com/webapp/sps/download/license.jsp?colCode=TESTTOOL_SETUP
3. Change the directory to Desktop/openthread-master
4. Complie the demo application using
   textcolorMaroon ./bootstrap make f examples/Makefile-kw41z
5. If the complilation is successful, you will get the .elf files in openthread-master/output/kw41z/bin
6. Convert the .elf file to pure binary file using cd output/kw41z/bin arm-none-eabi-objcopy O binary ot-cli-ftd ot-cli-ftd.bin
7. Flash the binary file to the board using NXP Test Tool,
8. Open the Test Tool
9. Click on Firmware Loader$-$ > Kinetis Frimware Loader$-$ > Browse
10. Browse the binary file (Note you need to transfer the files from Ubuntu PC to Linux PC, because Test Tool works only in Windows)
11. Select Upload
12. Open the Putty terminal session using the port assigned to your board and with proper configuration
13. Press Enter, and enter help, you will see list of commands available to you.
14. Flash the same binary file to other NXP board
15. Use ifconfig to get the IP address of the nodes
16. Try to ping one node form other. Kudos! You have finished Open Thread Implementation on NXP Board.



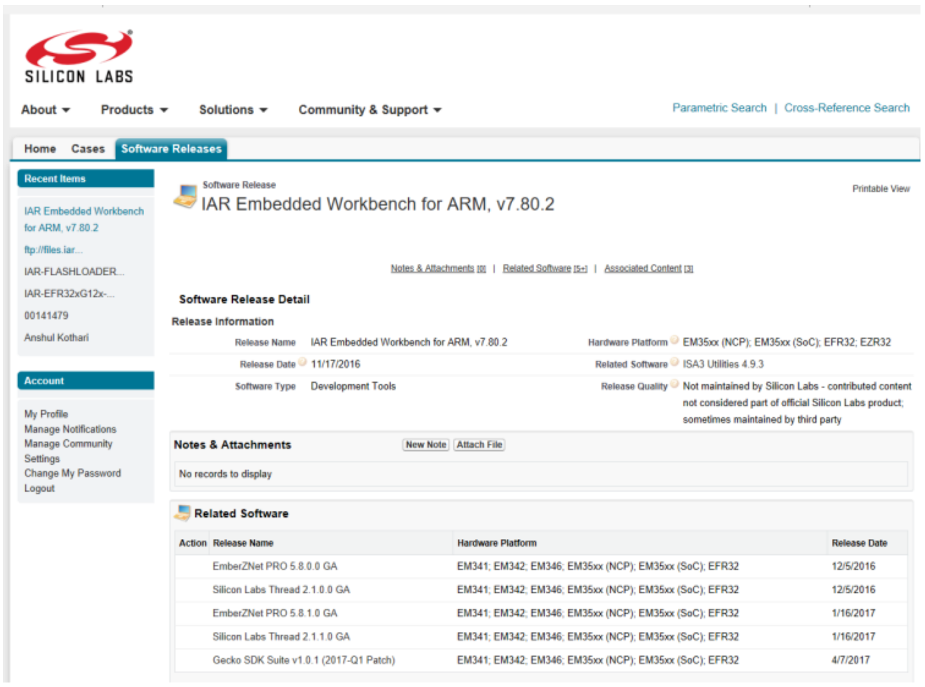
```
>
>
>
> help
help
autostart
blacklist
bufferinfo
channel
child
childmax
childtimeout
contextreusedelay
counter
dataset
delaytimermin
diag
discover
eidcache
eui64
extaddr
extpanid
factoryreset
```

3 Play with Silicon Lab EFR32MG KIT

3.1 Environment Setup

Important Note: Use Public WiFi or LAN connection, when using Simplicity Studio. It will not work, when you are connected to ESIEE network, which uses proxy server.
Steps to be followed:

1. Plug your boards to PC via USB cable
2. Download and install the Simplicity studio from`http://www.silabs.com/support/getting-started/mesh-networking/thread-getting-started/get-started-with-thread-and-mighty-gecko`
3. Sign up, if you are new user
4. You will be prompted for installation of the device drivers, click OK
5. Register your Kit, if not already registered
6. Download Thread SDK as follows
7. Click on the package manager (green downward arrow) and check the Thread SDK (2.3.0.0) and uncheck other packages
8. Wait for the complete installation
9. Your devices will be shown in the Device panel
10. Double Click on the device
11. Click on the Install, next to Adapter Firmware version (if update is available)
12. Install IAR workbench to compile your source files using
13. Click on Resource$-$ >Technical Support$-$ > Scroll Down to Email Support Request $-$ > Select Software Releases
14. Select EFR32MG Software and click on Go
15. Click on IAR Embedded Workbench for ARM, v7.80.2
16. Click on ftp://files_.exe link
17. Click on Open in the Top left corner, to download IAR workbench
18. Install the IAR workbench using default setting, generate temporary key if needed



Software Release

IAR Embedded Workbench for ARM, v7.80.2

Software Release Detail

Release Information

Release Name	IAR Embedded Workbench for ARM, v7.80.2	Hardware Platform	EM35xx (NCP); EM35xx (SoC); EFR32; EZR32
Release Date	11/17/2016	Related Software	ISA3 Utilities 4.9.3
Software Type	Development Tools	Release Quality	Not maintained by Silicon Labs - contributed content not considered part of official Silicon Labs product; sometimes maintained by third party

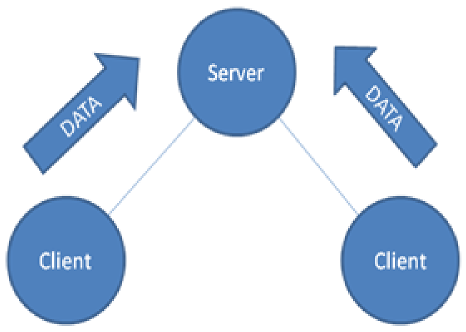
Notes & Attachments

No records to display

Related Software

Action	Release Name	Hardware Platform	Release Date
	EmberZNet PRO 5.8.0.0 GA	EM341; EM342; EM346; EM35xx (NCP); EM35xx (SoC); EFR32	12/5/2016
	Silicon Labs Thread 2.1.0.0 GA	EM341; EM342; EM346; EM35xx (NCP); EM35xx (SoC); EFR32	12/5/2016
	EmberZNet PRO 5.8.1.0 GA	EM341; EM342; EM346; EM35xx (NCP); EM35xx (SoC); EFR32	1/16/2017
	Silicon Labs Thread 2.1.1.0 GA	EM341; EM342; EM346; EM35xx (NCP); EM35xx (SoC); EFR32	1/16/2017
	Gecko SDK Suite v1.0.1 (2017-Q1 Patch)	EM341; EM342; EM346; EM35xx (NCP); EM35xx (SoC); EFR32	4/7/2017

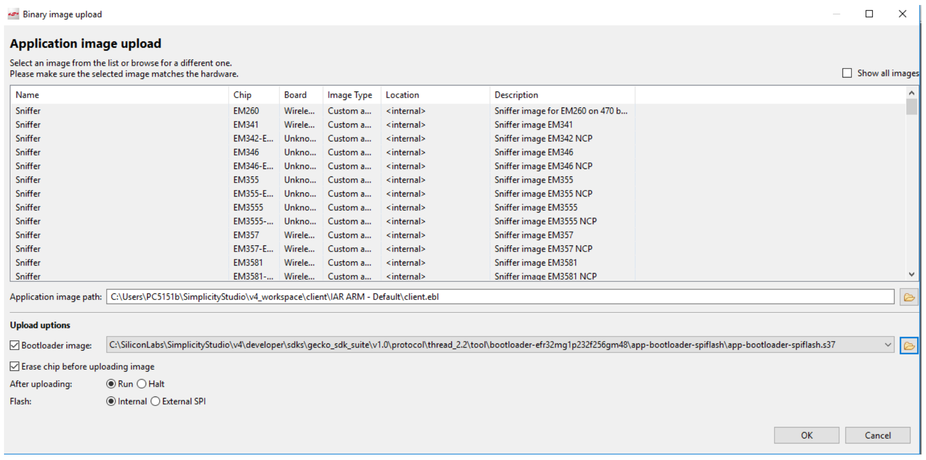
3.2 Server Client Application



Steps need to be followed:

1. Double click on one of the Device in the Device Panel
2. Then select Client under the Getting Started -¿ Silicon Lab Thread examples
3. Click OK to Create Example Project

4. Click Generate to generate the Application source files (Change the generation directory if needed)
5. Right click on your project in the project explorer view and click build project (Note that you have IAR workbench working)
6. Right click on the device in the device panel and select Upload Application
7. Choose the files as shown in the figure, note the path of the files
8. Click OK
9. Right click on the device in the device panel and select launch console, tab on serial 1 and press enter
10. Join the client in the network using command
    join
    Kudos! you have your client application up and running
    Follow the similar procedure for the server (Note you need to install server application on one of the board and client on other two)
11. Launch the console for server and choose serial 1 tab
12. Add the client by using the following command
    expect join-key (join-key, which you will get when you type join in the client shell)
13. Analyze the packets in the air using Network Analyzer
14. Right click on the device on which you want to analyze the traffic, in the device panel and select start capture, you can visualize the sent and received packets over the air
15. In the Launcher Click on the Compatible Tools -¿ Energy Profiler
16. Choose the device in the current device dialog, for which you want to measure the power consumption, YeaH Now you can measure power consumption, average voltage, average current of each device in the thread network. Kudos! Now you are free to explore the Silicon Lab Boards and create awesome applications.



Binary image upload

Application image upload

Select an image from the list or browse for a different one.
Please make sure the selected image matches the hardware.

☐ Show all images

Name	Chip	Board	Image Type	Location	Description
Sniffer	EM260	Wirele...	Custom a...	<internal>	Sniffer image for EM260 on 470 b...
Sniffer	EM341	Wirele...	Custom a...	<internal>	Sniffer image EM341
Sniffer	EM342-E...	Unkno...	Custom a...	<internal>	Sniffer image EM342 NCP
Sniffer	EM346	Unkno...	Custom a...	<internal>	Sniffer image EM346
Sniffer	EM346-E...	Unkno...	Custom a...	<internal>	Sniffer image EM346 NCP
Sniffer	EM355	Unkno...	Custom a...	<internal>	Sniffer image EM355
Sniffer	EM355-E...	Unkno...	Custom a...	<internal>	Sniffer image EM355 NCP
Sniffer	EM3555	Unkno...	Custom a...	<internal>	Sniffer image EM3555
Sniffer	EM3555-...	Unkno...	Custom a...	<internal>	Sniffer image EM3555 NCP
Sniffer	EM357	Wirele...	Custom a...	<internal>	Sniffer image EM357
Sniffer	EM357-E...	Wirele...	Custom a...	<internal>	Sniffer image EM357 NCP
Sniffer	EM3581	Wirele...	Custom a...	<internal>	Sniffer image EM3581
Sniffer	EM3581-...	Wirele...	Custom a...	<internal>	Sniffer image EM3581 NCP

Application image path: C:\Users\PC5151b\SimplicityStudio\v4_workspace\client\IAR ARM - Default\client.ebl

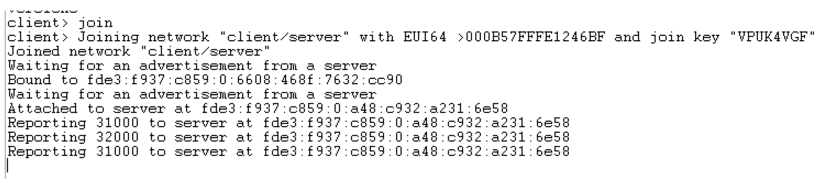
Upload uptions

☑ Bootloader image: C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v1.0\protocol\thread_2.2\tool\bootloader-efr32mg1p232f256gm48\app-bootloader-spiflash\app-bootloader-spiflash.s37

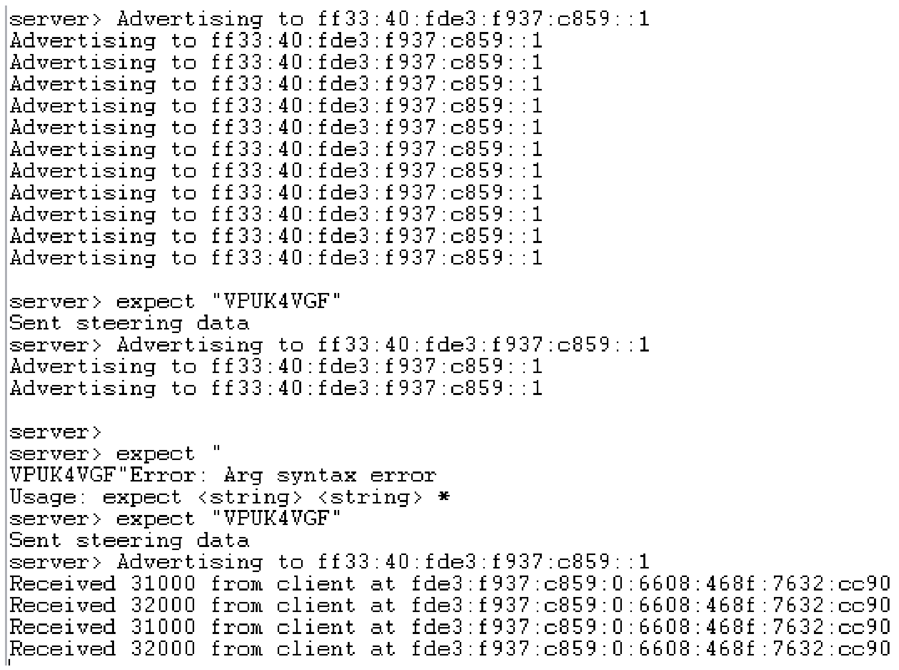
☑ Erase chip before uploading image

After uploading: [✓ Run] Halt

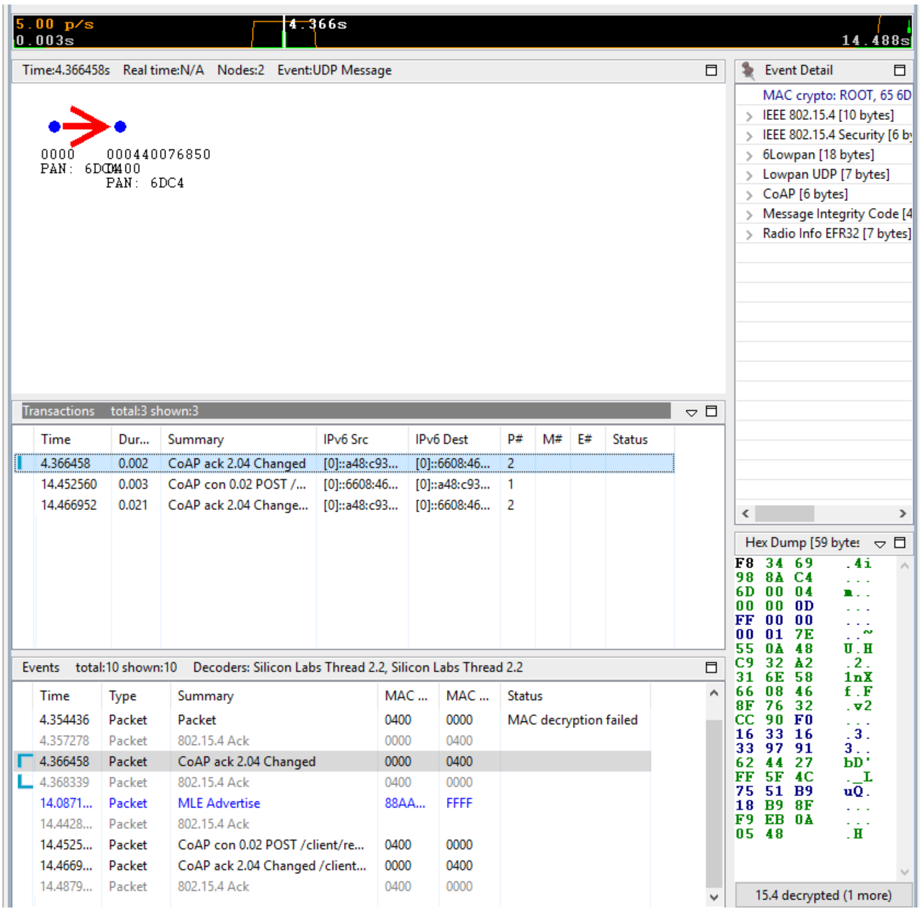
Flash: [✓ Internal] External SPI



```
client> join
client> Joining network "client/server" with EUI64 >000B57FFFE1246BF and join key "VPUK4VGF"
Joined network "client/server"
Waiting for an advertisement from a server
Bound to fde3:f937:c859:0:6608:468f:7632:cc90
Waiting for an advertisement from a server
Attached to server at fde3:f937:c859:0:a48:c932:a231:6e58
Reporting 31000 to server at fde3:f937:c859:0:a48:c932:a231:6e58
Reporting 32000 to server at fde3:f937:c859:0:a48:c932:a231:6e58
Reporting 31000 to server at fde3:f937:c859:0:a48:c932:a231:6e58
```



```
server> Advertising to ff33:40:fde3:f937:c859::1
Advertising to ff33:40:fde3:f937:c859::1
Advertising to ff33:40:fde3:f937:c859::1
Advertising to ff33:40:fde3:f937:c859::1
Advertising to ff33:40:fde3:f937:c859::1
Advertising to ff33:40:fde3:f937:c859::1
Advertising to ff33:40:fde3:f937:c859::1
Advertising to ff33:40:fde3:f937:c859::1
Advertising to ff33:40:fde3:f937:c859::1
Advertising to ff33:40:fde3:f937:c859::1
Advertising to ff33:40:fde3:f937:c859::1
Advertising to ff33:40:fde3:f937:c859::1

server> expect "VPUK4VGF"
Sent steering data
server> Advertising to ff33:40:fde3:f937:c859::1
Advertising to ff33:40:fde3:f937:c859::1
Advertising to ff33:40:fde3:f937:c859::1

server>
server> expect "
VPUK4VGF"Error: Arg syntax error
Usage: expect <string> <string> *
server> expect "VPUK4VGF"
Sent steering data
server> Advertising to ff33:40:fde3:f937:c859::1
Received 31000 from client at fde3:f937:c859:0:6608:468f:7632:cc90
Received 32000 from client at fde3:f937:c859:0:6608:468f:7632:cc90
Received 31000 from client at fde3:f937:c859:0:6608:468f:7632:cc90
Received 32000 from client at fde3:f937:c859:0:6608:468f:7632:cc90
```



Time:4.366458s Real time:N/A Nodes:2 Event:UDP Message

```
0000      000440076850
PAN: 6DC4 0400
          PAN: 6DC4
```

Event Detail

```
MAC crypto: ROOT, 65 6D
> IEEE 802.15.4 [10 bytes]
> IEEE 802.15.4 Security [6 by
> 6Lowpan [18 bytes]
> Lowpan UDP [7 bytes]
> CoAP [6 bytes]
> Message Integrity Code [4
> Radio Info EFR32 [7 bytes]
```

Transactions total:3 shown:3

Time	Dur...	Summary	IPv6 Src	IPv6 Dest	P#	M#	E#	Status
4.366458	0.002	CoAP ack 2.04 Changed	[0]::a48:c93...	[0]::6608:46...	2			
14.452560	0.003	CoAP con 0.02 POST /...	[0]::6608:46...	[0]::a48:c93...	1			
14.466952	0.021	CoAP ack 2.04 Change...	[0]::a48:c93...	[0]::6608:46...	2			

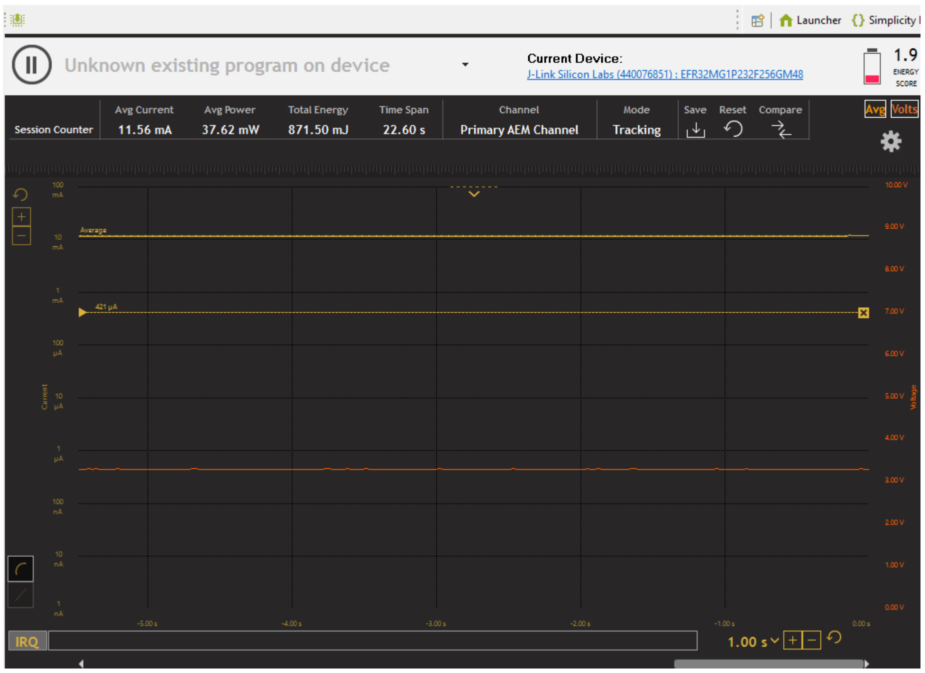
Hex Dump [59 bytes

```
F8 34 69   .4i
98 8A C4   ...
6D 00 04   m..
00 00 0D   ...
FF 00 00   ...
00 01 7E   ..~
55 0A 48   U.H
C9 32 A2   .2.
31 6E 58   1nX
66 08 46   f.F
8F 76 32   .v2
CC 90 F0   ...
16 33 16   .3.
33 97 91   3..
62 44 27   bD'
FF 5F 4C   ._L
75 51 B9   uQ.
18 B9 8F   ...
F9 EB 0A   ...
05 48      .H
```

15.4 decrypted (1 more)

Events total:10 shown:10 Decoders: Silicon Labs Thread 2.2, Silicon Labs Thread 2.2

Time	Type	Summary	MAC ...	MAC ...	Status
4.354436	Packet	Packet	0400	0000	MAC decryption failed
4.357278	Packet	802.15.4 Ack	0000	0400	
4.366458	Packet	CoAP ack 2.04 Changed	0000	0400	
4.368339	Packet	802.15.4 Ack	0400	0000	
14.0871...	Packet	MLE Advertise	88AA...	FFFF	
14.4428...	Packet	802.15.4 Ack			
14.4525...	Packet	CoAP con 0.02 POST /client/re...	0400	0000	
14.4669...	Packet	CoAP ack 2.04 Changed /client...	0000	0400	
14.4879...	Packet	802.15.4 Ack	0400	0000	



Unknown existing program on device

Current Device:
J-Link Silicon Labs (440076851) : EFR32MG1P232F256GM48

1.9
ENERGY
SCORE

Session Counter	Avg Current	Avg Power	Total Energy	Time Span	Channel	Mode
	11.56 mA	37.62 mW	871.50 mJ	22.60 s	Primary AEM Channel	Tracking

Save Reset Compare

4 Play with Zolertia Re-Mote

4.1 Environment Setup

Operating System Used: Ubuntu 16.04 LTS
Steps to be followed:

1. Get your RE-Mote Board (Hardware platform from Zolertia) and the USB cable

2. Plug the board to the PC using USB cable

3. Open the terminal (Clt + Alt + T)

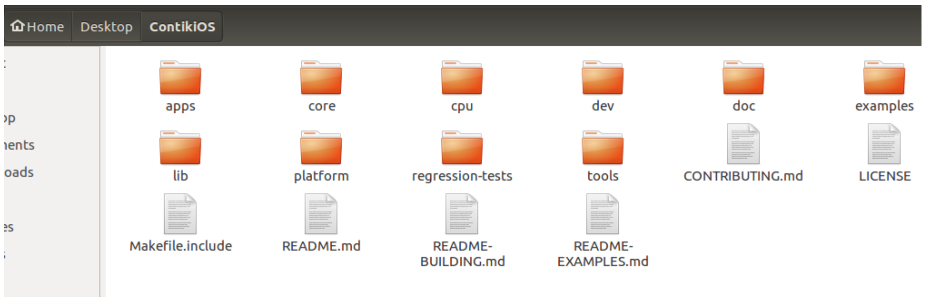
4. Install the toolchain (Compiler and debugger), if not installed
   sudo apt-get update
   sudo apt-get install gcc-arm-none-eabi gdb-arm-none-eabi

5. Change directory, where you want to install Contiki operating system cd Desktop git clone https://github.com/contiki-os/contiki

6. Update the repository with dependencies
   cd contikiOS git submodule update init –recursive

4.2 Contiki Operating System

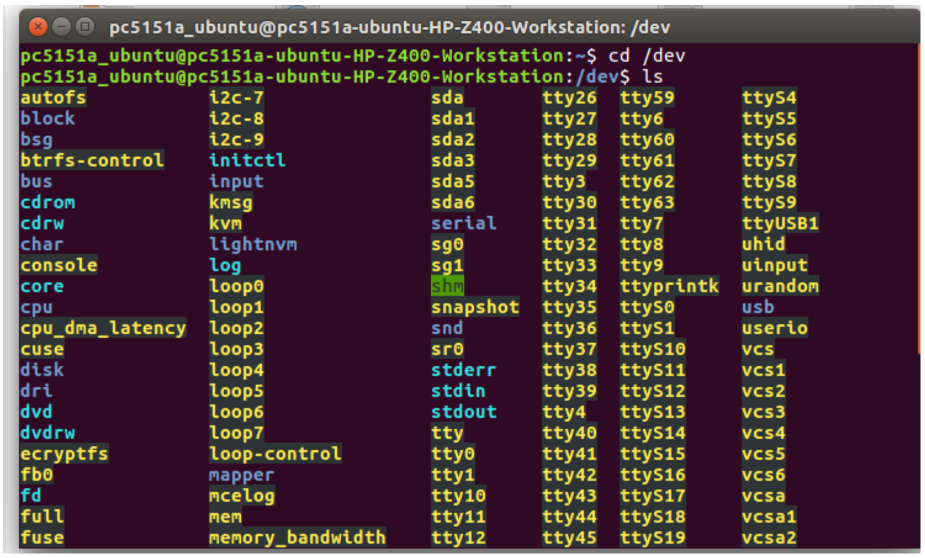
It is open source OS , specifically designed for Internet of Things. Directory structure for Contiki Operaring System is as follows:
Apps : It contains application files such as email, MQTT, CoAP etc.
Core: It contains OS files which are platform independent.
Examples: It contains many examples to test the functionality of sensors attached.
Platform: It contains platform specific files such as pin configuration etc
Tools: It contains files for tools such as Simulator etc.



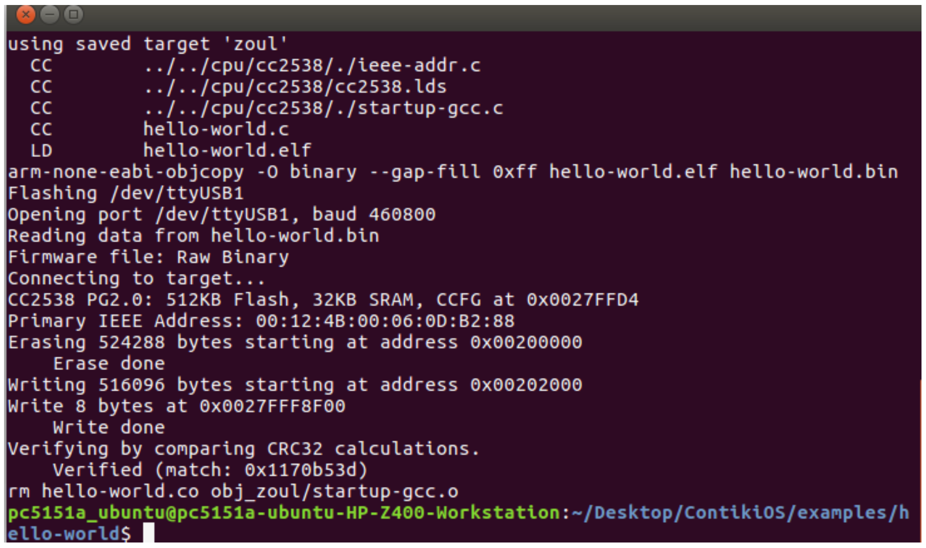
4.3 Building Hello World Application

Steps to be followed:

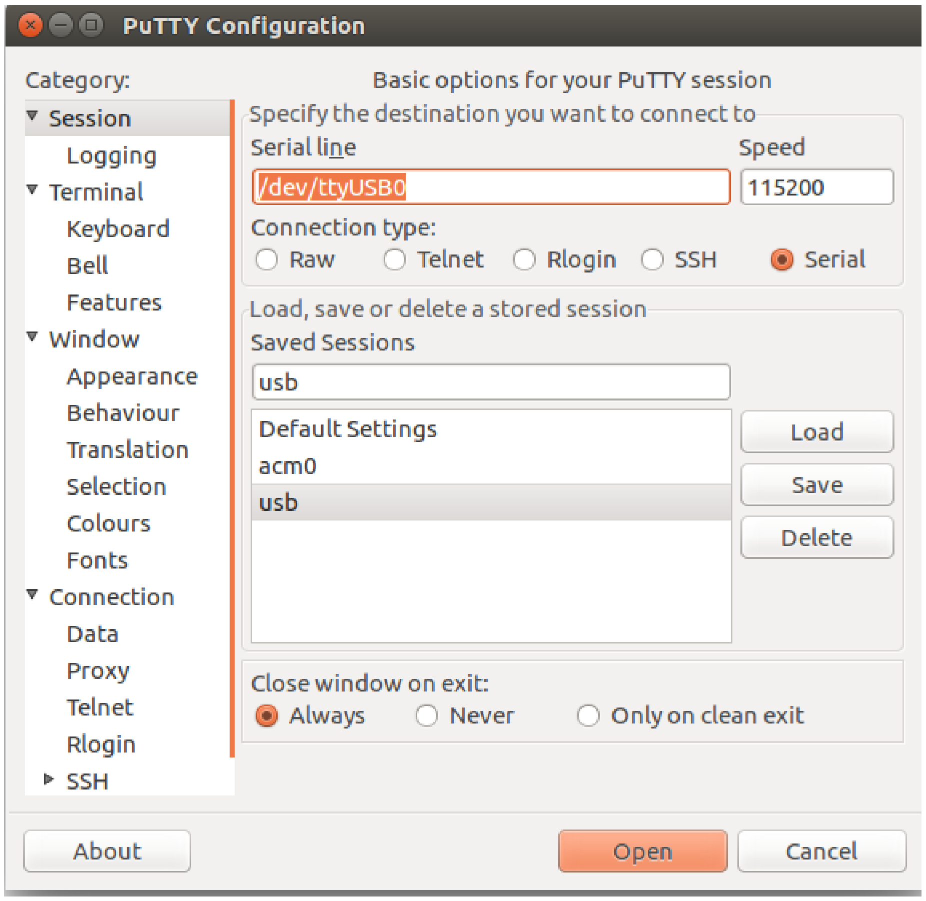
1. Change directory
   cd ContikiOS/examples/hello-world
2. Definition of Make file: Make files are used to compile many source files together and efficiently
3. Save the Target platform in the Makefile.target
   make TARGET=zoul
4. Open new terminal change directory
   cd /dev
5. Search for the ttyUSB0 or any other ttyUSB file assigned to your board and change its permission (you have to do every time you log in to your system)
   sudo chmod 777 ttyUSB0
6. Compile and flash the binary file to the board
   make hello-world.upload (It should create 4 files with extensions .bin, .elf, .csc, .map)
7. Open Putty terminal with the following configurations as shown in the image Yeah! You will see hello world printed to the console (Note that you will get only single hello world written on the console, if you have used default hello-world.c file)



```
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~$ cd /dev
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:/dev$ ls
autofs           i2c-7               sda       tty26  tty59      ttyS4
block            i2c-8               sda1      tty27  tty6       ttyS5
bsg              i2c-9               sda2      tty28  tty60      ttyS6
btrfs-control    initctl             sda3      tty29  tty61      ttyS7
bus              input               sda5      tty3   tty62      ttyS8
cdrom            kmsg                sda6      tty30  tty63      ttyS9
cdrw             kvm                 serial    tty31  tty7       ttyUSB1
char             lightnvm            sg0       tty32  tty8       uhid
console          log                 sg1       tty33  tty9       uinput
core             loop0               shm       tty34  ttyprintk  urandom
cpu              loop1               snapshot  tty35  ttyS0      usb
cpu_dma_latency  loop2               snd       tty36  ttyS1      userio
cuse             loop3               sr0       tty37  ttyS10     vcs
disk             loop4               stderr    tty38  ttyS11     vcs1
dri              loop5               stdin     tty39  ttyS12     vcs2
dvd              loop6               stdout    tty4   ttyS13     vcs3
dvdrw            loop7               tty       tty40  ttyS14     vcs4
ecryptfs         loop-control        tty0      tty41  ttyS15     vcs5
fb0              mapper              tty1      tty42  ttyS16     vcs6
fd               mcelog              tty10     tty43  ttyS17     vcsa
full             mem                 tty11     tty44  ttyS18     vcsa1
fuse             memory_bandwidth    tty12     tty45  ttyS19     vcsa2
```



```
using saved target 'zoul'
  CC        ../../cpu/cc2538/./ieee-addr.c
  CC        ../../cpu/cc2538/cc2538.lds
  CC        ../../cpu/cc2538/./startup-gcc.c
  CC        hello-world.c
  LD        hello-world.elf
arm-none-eabi-objcopy -O binary --gap-fill 0xff hello-world.elf hello-world.bin
Flashing /dev/ttyUSB1
Opening port /dev/ttyUSB1, baud 460800
Reading data from hello-world.bin
Firmware file: Raw Binary
Connecting to target...
CC2538 PG2.0: 512KB Flash, 32KB SRAM, CCFG at 0x0027FFD4
Primary IEEE Address: 00:12:4B:00:06:0D:B2:88
Erasing 524288 bytes starting at address 0x00200000
    Erase done
Writing 516096 bytes starting at address 0x00202000
Write 8 bytes at 0x0027FFF8F00
    Write done
Verifying by comparing CRC32 calculations.
    Verified (match: 0x1170b53d)
rm hello-world.co obj_zoul/startup-gcc.o
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Desktop/ContikiOS/examples/h
ello-world$ 
```



Specify the destination you want to connect to

Serial line: /dev/ttyUSB0

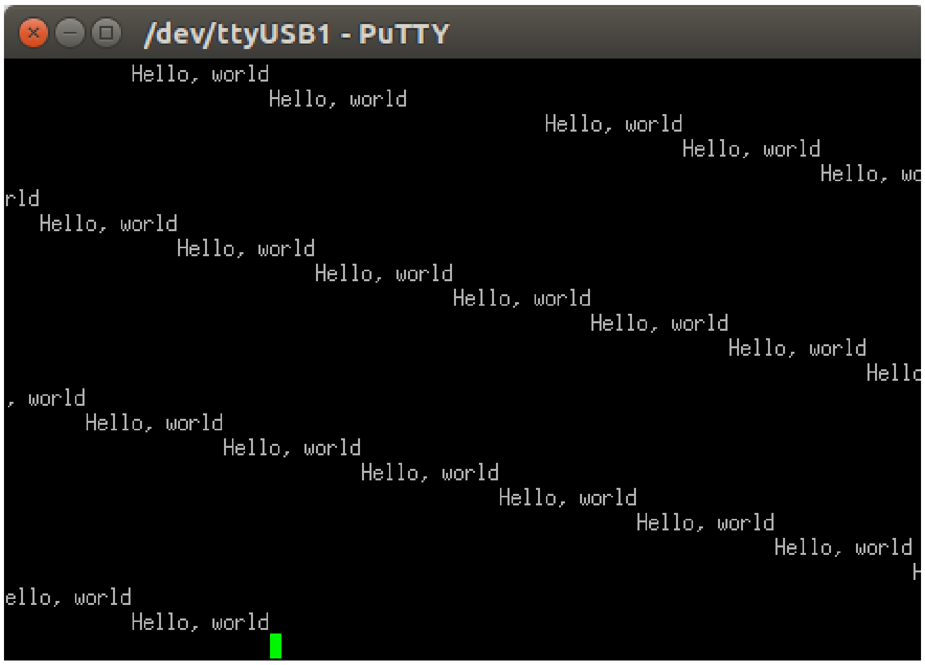
Speed: 115200

Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Saved Sessions: usb

Default Settings
acm0
usb

Close window on exit: [✓ Always] Never Only on clean exit



```
           Hello, world
                       Hello, world
                                               Hello, world
                                                           Hello, world
                                                                       Hello, wo
rld
   Hello, world
               Hello, world
                           Hello, world
                                       Hello, world
                                                   Hello, world
                                                               Hello, world
                                                                           Hello
, world
       Hello, world
                   Hello, world
                               Hello, world
                                           Hello, world
                                                       Hello, world
                                                                   Hello, world
                                                                               H
ello, world
           Hello, world
```

4.4 Hello World Application Explanation

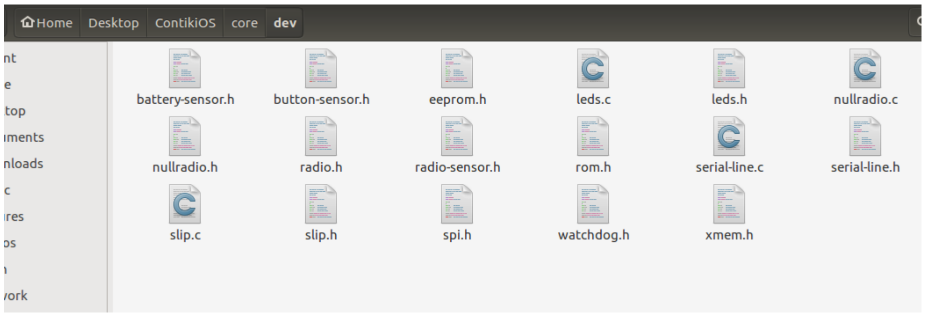
Open the file hello-world.c in the current directory with gedit or nano text editor. Below is the line wise description of the file.

- Include the core operating system files
- Includes the standard library of C
- Give the name to your process, one that will be used internally and one that you want to be printed on the console
- It instructs the OS to start the process as soon as the OS boots.
- Syntax for defining your process, ev is event and data is event data
- Your actually function logic should be between process_begin() and process_end()

4.5 Building Led On/Off Application

For every sensor or actuator you have driver in the ContikiOS, which provides abstraction. So you can just use API without concerning about the underlying hardware configuration.
Steps to be followed:

1. Create test-led.c file and its make file similar to the make file for hello-world
2. #include /dev/led.h, (LED Driver) which contains led specific functions declarations
3. Check for the available functions in led.h and use them accordingly.
4. Create the test-led.c similar to hello-world.c file and modify the function logic between Process_Begin() and Process_End()
5. Complie and Run the file similar to hello-world example.
6. Exercise: Try to create application for displaying current temperature using temperature sensor Driver API. (Hint you can find the API in ContikiOS/platform/zoul/dev)



4.6 Implementing Open Thread Stack

Steps to be followed:

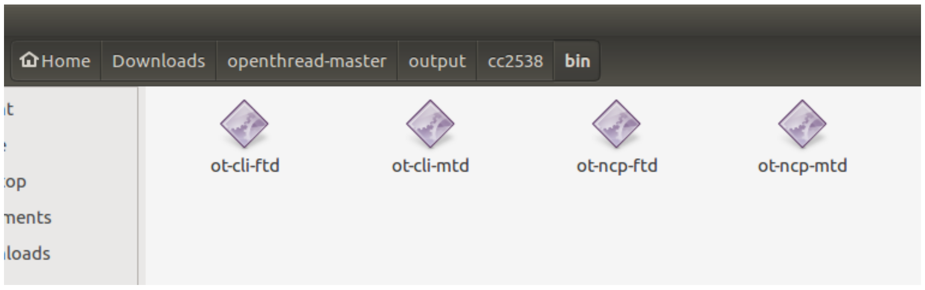
1. If not installed, install toolchain
   sudo apt-get update
   sudo apt-get install gcc-arm-none-eabi gdb-arm-none-eabi
2. Install the dependencies
   sudo apt-get install python-pip
   pip install intelhex
   pip install python-magic
3. Clone the openthread repository (change the directory, where you want to clone the repository to)
   cd Desktop
   git clone recursive https://github.com/openthread/openthread

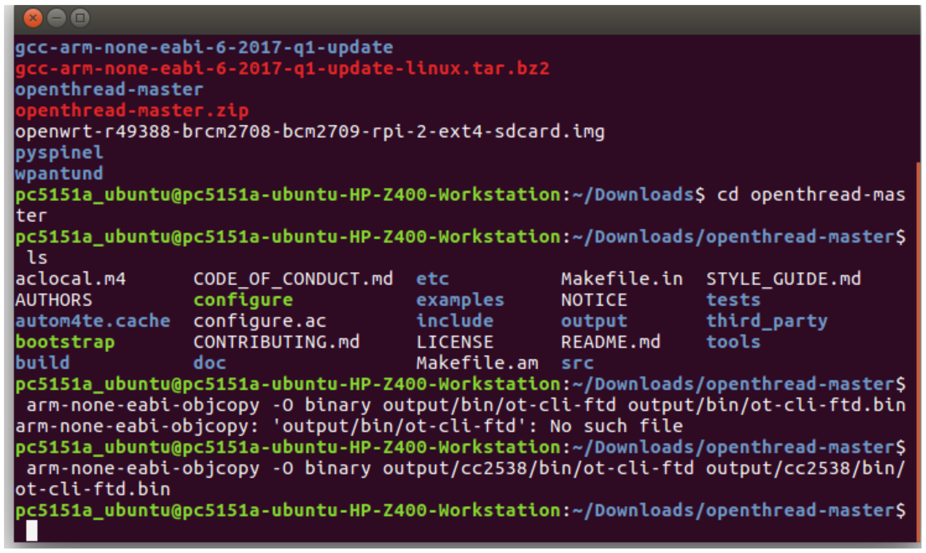
4. Compile the Source files
   ./bootstrap
   make f examples/Makefile-cc2538 clean && make f examples/Makefile-cc2538
   (4 files will be created in the directory openthread-master/output/cc2538/bin)

5. Convert the files to pure binary
   arm-none-eabi-objcopy O binary output/cc2538/bin/ot-cli-ftd output/cc2538/bin/ot-cli-ftd.bin

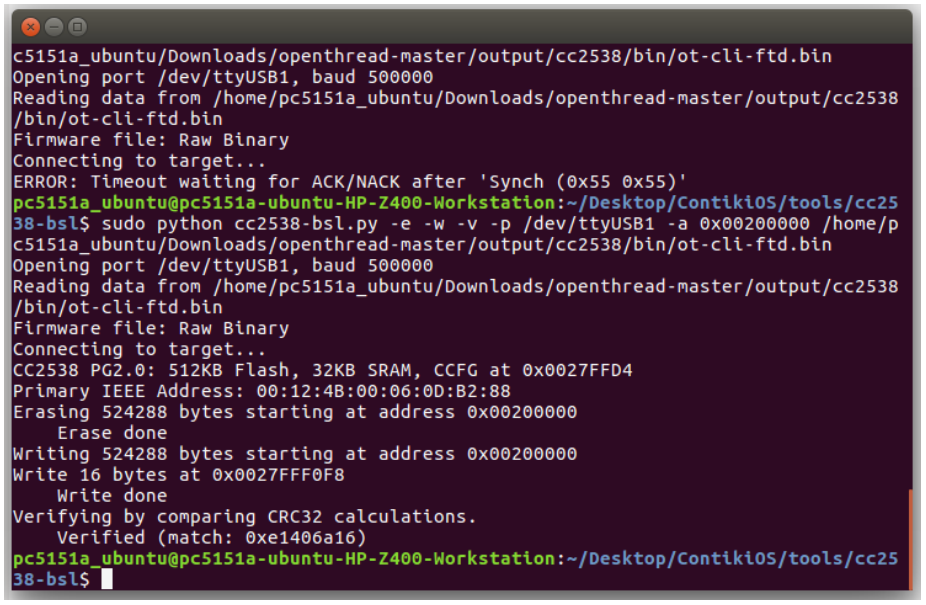
6. Flash the binary files to the board using cc2538-bsl (Flash program) , open new linux terminal
   cd ContikiOS/tools/cc2538-bsl
   sudo python cc2538-bsl.py e w v p /dev/ttyUSB0 a 0x00200000 home/Desktop/openthread-master/output/cc2538/bin/filename.bin (Complete file path) Note: You may need to change the device file according to the port your board is assigned. (To check the port run the following command in the linux shell dmseg — grep ttyUSB)

7. Open Putty terminal session with the configurations mentioned in earlier exercise. Kudos! You have your open thread command line interface running on your board.

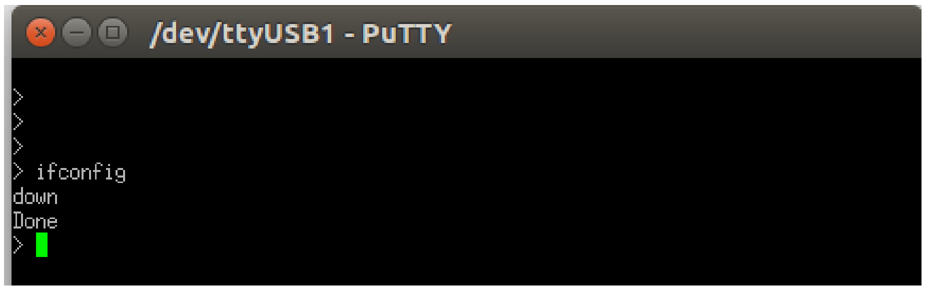




```
gcc-arm-none-eabi-6-2017-q1-update
gcc-arm-none-eabi-6-2017-q1-update-linux.tar.bz2
openthread-master
openthread-master.zip
openwrt-r49388-brcm2708-bcm2709-rpi-2-ext4-sdcard.img
pyspinel
wpantund
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Downloads$ cd openthread-mas
ter
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Downloads/openthread-master$
 ls
aclocal.m4      CODE_OF_CONDUCT.md  etc          Makefile.in  STYLE_GUIDE.md
AUTHORS         configure           examples     NOTICE       tests
autom4te.cache  configure.ac        include      output       third_party
bootstrap       CONTRIBUTING.md     LICENSE      README.md    tools
build           doc                 Makefile.am  src
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Downloads/openthread-master$
 arm-none-eabi-objcopy -O binary output/bin/ot-cli-ftd output/bin/ot-cli-ftd.bin
arm-none-eabi-objcopy: 'output/bin/ot-cli-ftd': No such file
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Downloads/openthread-master$
 arm-none-eabi-objcopy -O binary output/cc2538/bin/ot-cli-ftd output/cc2538/bin/
ot-cli-ftd.bin
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Downloads/openthread-master$
```



```
c5151a_ubuntu/Downloads/openthread-master/output/cc2538/bin/ot-cli-ftd.bin
Opening port /dev/ttyUSB1, baud 500000
Reading data from /home/pc5151a_ubuntu/Downloads/openthread-master/output/cc2538
/bin/ot-cli-ftd.bin
Firmware file: Raw Binary
Connecting to target...
ERROR: Timeout waiting for ACK/NACK after 'Synch (0x55 0x55)'
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Desktop/ContikiOS/tools/cc25
38-bsl$ sudo python cc2538-bsl.py -e -w -v -p /dev/ttyUSB1 -a 0x00200000 /home/p
c5151a_ubuntu/Downloads/openthread-master/output/cc2538/bin/ot-cli-ftd.bin
Opening port /dev/ttyUSB1, baud 500000
Reading data from /home/pc5151a_ubuntu/Downloads/openthread-master/output/cc2538
/bin/ot-cli-ftd.bin
Firmware file: Raw Binary
Connecting to target...
CC2538 PG2.0: 512KB Flash, 32KB SRAM, CCFG at 0x0027FFD4
Primary IEEE Address: 00:12:4B:00:06:0D:B2:88
Erasing 524288 bytes starting at address 0x00200000
    Erase done
Writing 524288 bytes starting at address 0x00200000
Write 16 bytes at 0x0027FFF0F8
    Write done
Verifying by comparing CRC32 calculations.
    Verified (match: 0xe1406a16)
pc5151a_ubuntu@pc5151a-ubuntu-HP-Z400-Workstation:~/Desktop/ContikiOS/tools/cc25
38-bsl$
```

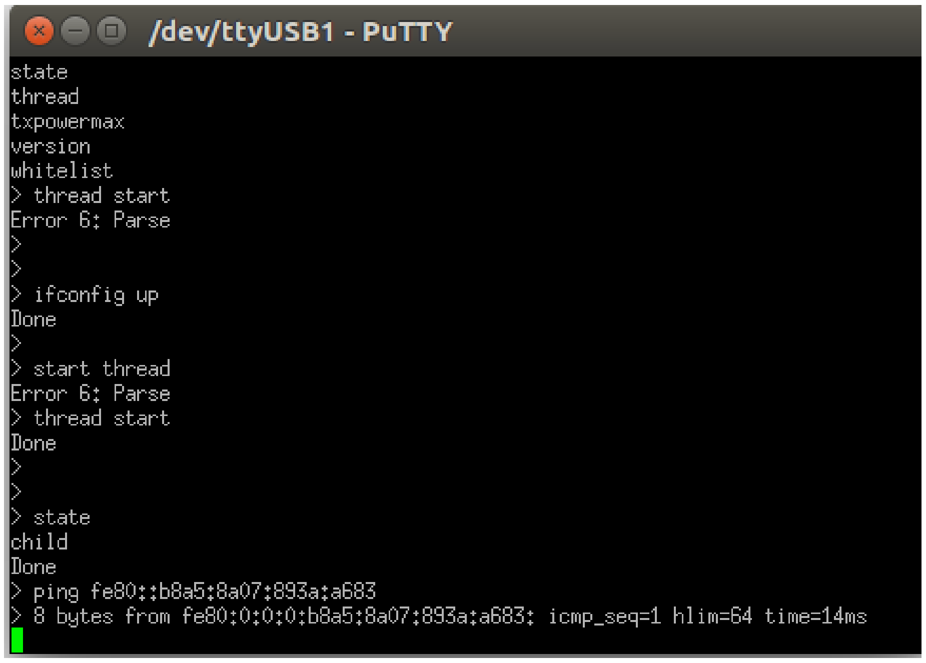


```
>
>
>
> ifconfig
down
Done
> 
```

5 Creating Heterogeneous Network with NXP FRDM-KW41Z and Zolertia Boards

Steps to be followed:

1. Configure one of your NXP board as mentioned in the section 1.5 and Zolertia board as mentioned in the section 5.1
2. Connect both the boards via USB to your PC (Windows or Linux)
3. On each one open Putty terminal session on the connected USB port (Check for the port in devices-¿ devices manager-¿Ports) such as COM3 or COM8 etc
4. Create thread network using NXP boards shell
   thr create
5. Get the channel in NXP board
   thr get channel
6. Set the channel in the Zolertia board
   channel xyz
7. Get the PANID of the thread network in NXP board
   thr get panid
   0x abc
8. Set the panid for the Zolertia board
   panid 0xabc
9. Join the zolertia board in the network
   ifconfig up
   thread start
10. On each one Try to ping other device in the network (use ifconfig command to get the IP address of the device) Kudos! You have established the communication between different types of board.



```
/dev/ttyUSB1 - PuTTY
state
thread
txpowermax
version
whitelist
> thread start
Error 6: Parse
>
>
> ifconfig up
Done
>
> start thread
Error 6: Parse
> thread start
Done
>
>
> state
child
Done
> ping fe80::b8a5:8a07:893a:a683
> 8 bytes from fe80:0:0:0:b8a5:8a07:893a:a683: icmp_seq=1 hlim=64 time=14ms
```

6 Work Need To Done

- Establish a GIANT mesh network with all the three types of boards i.e. FRDM KW41Z, EFR32MG, Zolertia Re-Mote (using NXP thread stack on NXP board, silicon Lab thread stack on silicon Lab board and open thread stack on Zolertia board)
- Testing with External Commissioning (i.e. using Border router as relay agent between joiner router and external commissioner)

7 References

I am listing some of the references from where you can get the help when you are stuck somewhere or need detailed explaination of the concepts.

1. If you want to get the detailed overview of Thread Protocol then click
   https://threadgroup.org/About

2. Link to getting started guide for NXP Board
   http://www.nxp.com/products/software-and-tools/hardware-development-tools/freedom-development-boards/nxp-freedom-development-kit-for-kinetis-kw41z-31z-21z-mcus:FRDM-KW41Z

3. Link to Schematics of FRDM KW41Z
   http://www.nxp.com/downloads/en/schematics/FRDM-KW41Z-SCH.pdf

4. Link to open thread repository and directions for configurations
   https://github.com/openthread/openthread

5. Link for configuring openthread on NXP board
   https://github.com/openthread/openthread/tree/master/examples/platforms/kw41z

6. Link for configuring openthread on Zolertia board
   https://github.com/openthread/openthread/tree/master/examples/platforms/cc2538

7. Link to getting started guide to Zolertia board
   https://github.com/Zolertia/Resources/wiki/Getting-Started-with-Zolertia-products

8. Link to the book for understanding the Internet of Things concepts and get the practical experience with zolertia board
   http://wireless.ictp.it/school_2016/book/IoT_in_five_days-v1.0.pdf

9. Link to quick start guide from Silicon Labs
   https://www.silabs.com/documents/public/quick-start-guides/qsg113-efr32-thread.pdf