Meshymesh Update

There are three parts:Device application,Bootloader,Android application.

Device application

Name:Meshymesh
Git:git@git.thoughtworks.net:thoughtfulmeshmakers/meshymesh.git,at the dfu_support branch.

In the update process,this part is used for communicating with the Android application.It will check every message whether the client has right,return the version of device software,reboot the device into dfu mode.

Message Struct

The message structs are define in “message.h”.

This is a sample.

```
typedef struct {
        BleMessageHead head; //All request message must declare a head like this at the first of
struct,the the distribute logic can distribute this message.
     …………    // Must define other field behind the head
}__attribute__ ((packed))  BleMessageQueryVersionReq;  // Must declare __attribute__ ((packed))
to make the compiler don't pack this struct,else the Android application maybe can't decode this
message or it can't decode message from Android application.
```

```
typedef struct {
        BleMessageResult result; //All response must declare a result like this at the first of
struct.then the Android application can decode even if it's error.
        uint8_t versionMain;
        uint8_t versionSub;
}__attribute__ ((packed))  BleMessageQueryVersionRsp;
```

The head declare as this

```
typedef struct {
        BleMessageType messageType;     // Message Type is used to identify a request
        uint8_t sessionID;              // This is for a message which need response.the
client can pair request and response by this id.
        uint8_t password[6];            // Used for checking customer's role,whether the customer
can config the device.
}__attribute__ ((packed)) BleMessageHead;
```

Message Logic

The message logic is implemented at void handle_write_event(ble_evt_t * bleEvent) in gap.c
The password is not checked now,It should be implemented latter,and can be improved by certification.

You can use write_rsp(connectionHandle, &result, sizeof(result)) to send response message.

Question

1.The function handle_write_event can handle message of all characteristics,so maybe handle some unexpect message,and give no response.

2.Now the version information is queried on scanning device and selecting device.It will be better to add this version in advertsing package.It will be more easy to use but maybe it's not securety.

Bootloader

Name:Bootloader
Git:git@git.thoughtworks.net:thoughtfulmeshmakers/meshymesh.git,at the dfu_support branch.
There is a directory boot loader,all code is in this directory.we can compile it by makefile at this directory.

The boot loader is copied from meshymesh/deploy/sdk/nrf_sdk_9_0/examples/dfu/bootloader and change some file.
the main.c is changed ,this line is commentated // ble_stack_init(!app_reset);

dfu_transport_ble.c
change the device name

There is no install script for the boot loader can only installed by nRFgoStudio.

Android application

Name:FirmwareUpdate
Git:git@git.thoughtworks.net:thoughtfulmeshmakers/over-the-air-device-firmware-update.git

In the update process,this part is the customer' ui.The customer can update device by this application.

There are 4 packages.

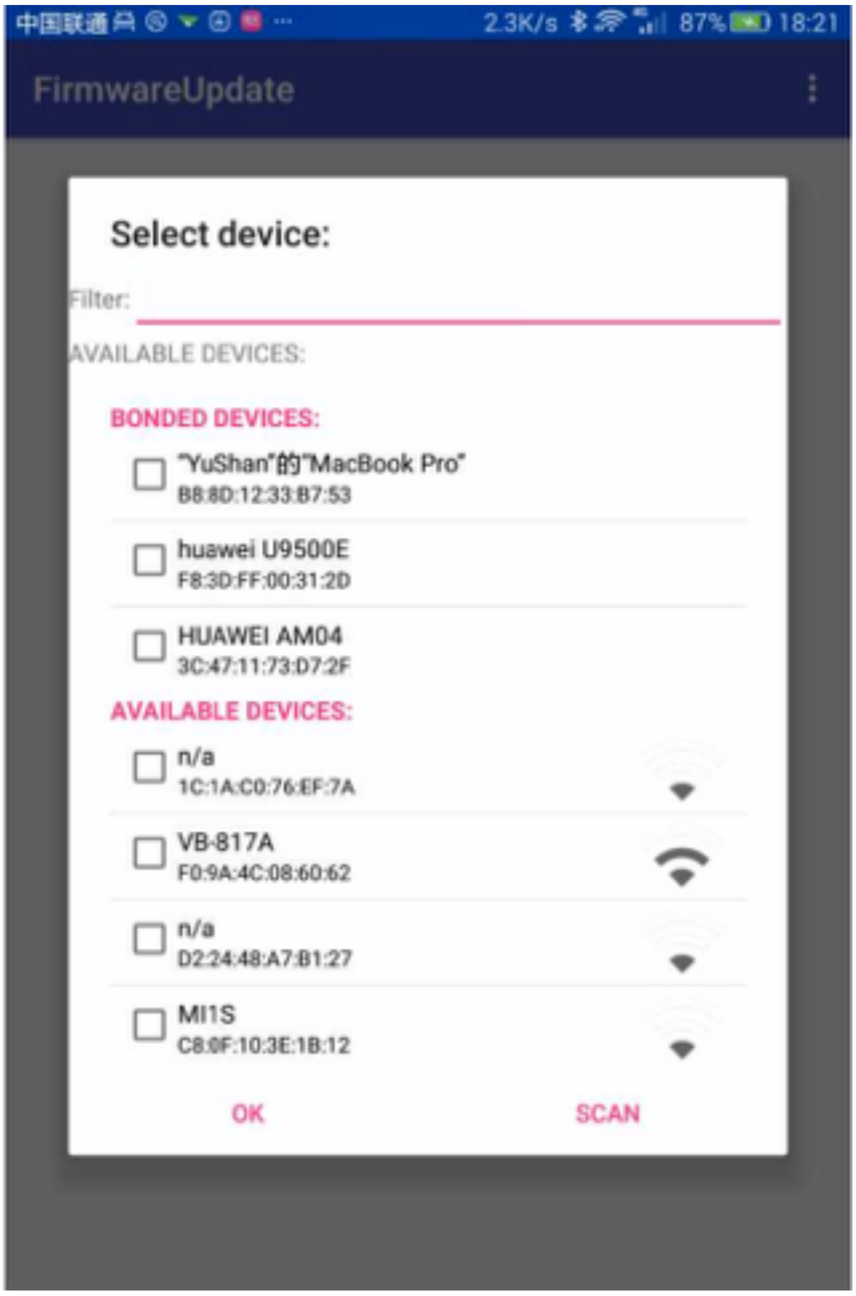
com.tw.firmwareupdate.msg

This package define all the message struct in java code,and has a class BleMessageConnection that can send and receive message from device.All message must have the same struct as message in c code.and we must implement the decode and encode method.

Please use method BleMessageRsp.readUint8 to read a 8 bits number,don't cast the byte to int.The result is that you expect when it's bigger then 127.

com.tw.firmwareupdate.scan

The code for scan device.



中国联通 2.3K/s 87% 18:21

FirmwareUpdate

Select device:

Filter:

AVAILABLE DEVICES:

BONDED DEVICES:

- "YuShan"的"MacBook Pro" B8:8D:12:33:B7:53
- huawei U9500E F8:3D:FF:00:31:2D
- HUAWEI AM04 3C:47:11:73:D7:2F

AVAILABLE DEVICES:

- n/a 1C:1A:C0:76:EF:7A
- VB-817A F0:9A:4C:08:60:62
- n/a D2:24:48:A7:B1:27
- MI1S C8:0F:10:3E:1B:12

OK SCAN

com.tw.firmwareupdate.update

This package is just extend from sdk used for updating device.

com.tw.firmwareupdate.util

This package contains some common class.

Update Logic

The update logic is in com.tw.firmwareupdate.MainActivity.updateSingleDevice

1.Scan the device.
When we connect to the device,we use the address,but the address is not the same when running application and running bootloader.and the device in device list to be update,maybe added at anytime,so it maybe not in the same mode when update.We get the real address by scanning again.
2.Connect to the device.
When connect,we can check the service,then we known the device is in dfu mode or application mode.

3.Send restart command
If the device is in the application mode,we will send a command to let the device reboot to dfu mode.
It will scan again to get the address in dfu mode and wait it reboot.

4.Update
This is done by the nordic sdk,It will upload zip file to device.

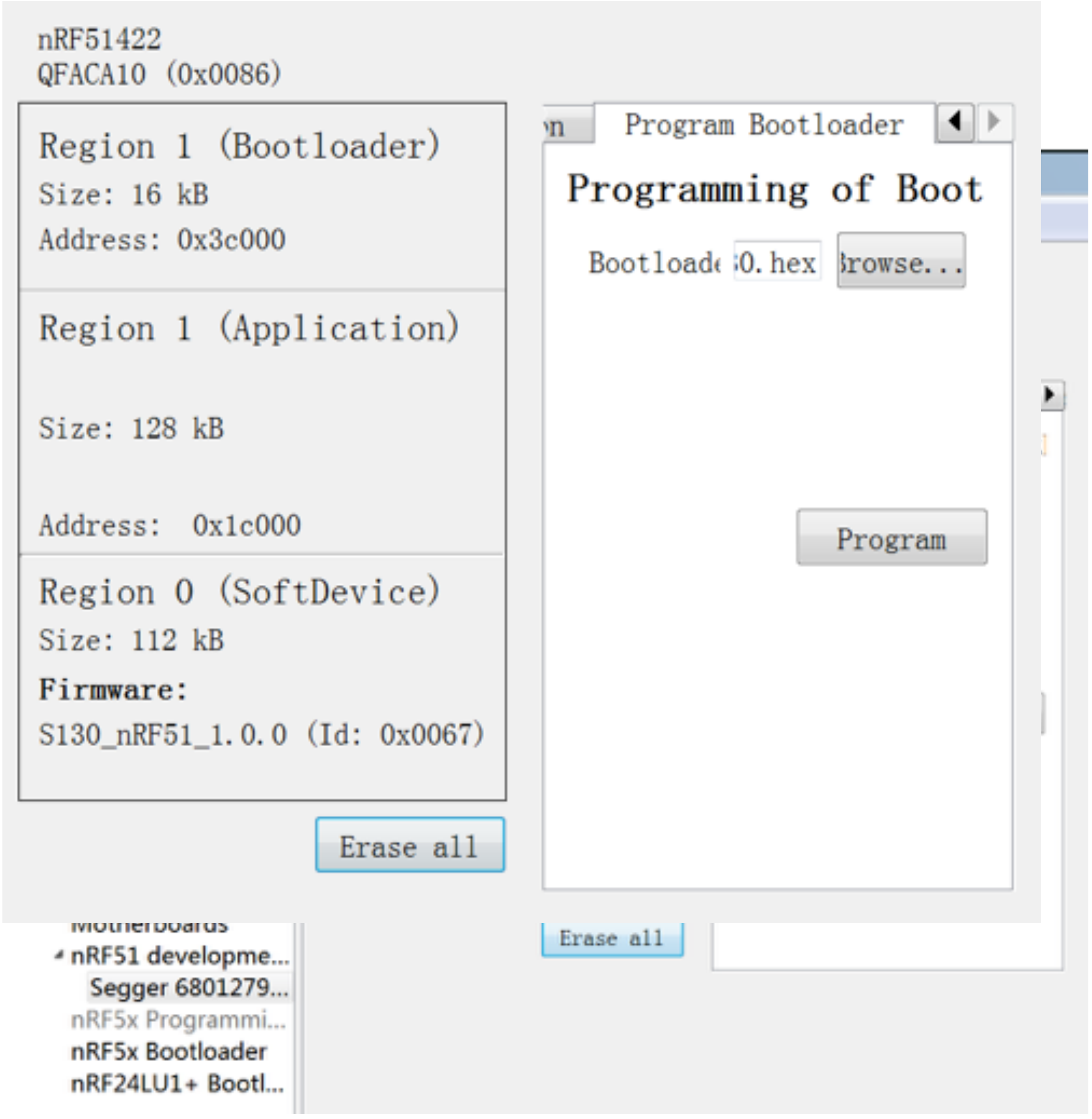
How to update

1.Erase the device
Click the Erase all button,all data in device will be erased.The boot loader can't be programed when there is boot loader installed,so we should erase first before program bootloder.
2.Install soft device
Select softdevice for s130,then click program button.
3.Install bootloader
Select boot loader hex file,you can build from meshymesh/bootloader.click program button.

4.Prepare deivce application to be updated.
You can compile meshy mesh from git@git.thoughtworks.net:thoughtfulmeshmakers/meshymesh.git
compile program with command "./go c"
make update file with command "make_update.cmd", It's a windows script.
Then we get a update failed "update.zip",we can copy this file to an android phone.

5.Install the android application

Get code from git@git.thoughtworks.net:thoughtfulmeshmakers/over-the-air-device-firmware-update.git
Use the Android Studio to open this project,and compile it.Connect you phone to computer with a USB line.The Android Studio can install and run this program on you phone.


nRF51422
QFACA10 (0x0086)

Region 1 (Bootloader)
Size: 16 kB
Address: 0x3c000

Region 1 (Application)
Size: 128 kB
Address: 0x1c000

Region 0 (SoftDevice)
Size: 112 kB
Firmware:
S130_nRF51_1.0.0 (Id: 0x0067)

6.Start the application FirmwareUpdate.

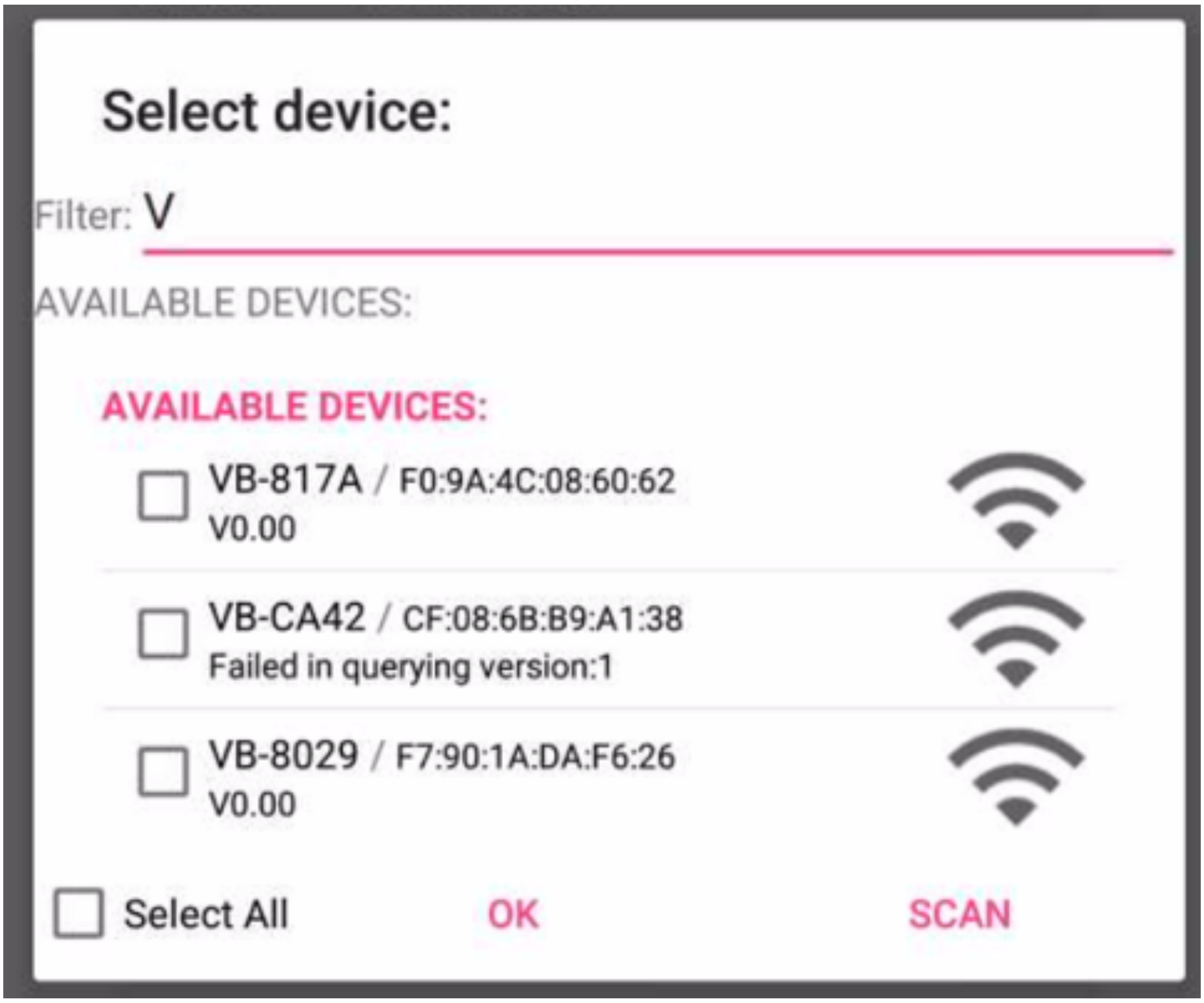
7.Select Firmware to update.
Click the button “Select File”.Choose the zip file.Now we can only update zip file with application.

8.Select Devices.
Click the button “Select Device”
You can click the check box before the device which you want to update.and then click button OK.
If the device is not here you can click button “SCAN” to scan again.

To select many device at once quickly, you can part of name in Filter,such like “VB”,then only the device which name contains “VB” will show in the list.Then click the check box “Select All”,then all device will be selected,Then click “OK”

In this frame you can see the device version,the Android application will query the version one by one.

9.Update


Select device:

Filter: V

AVAILABLE DEVICES:

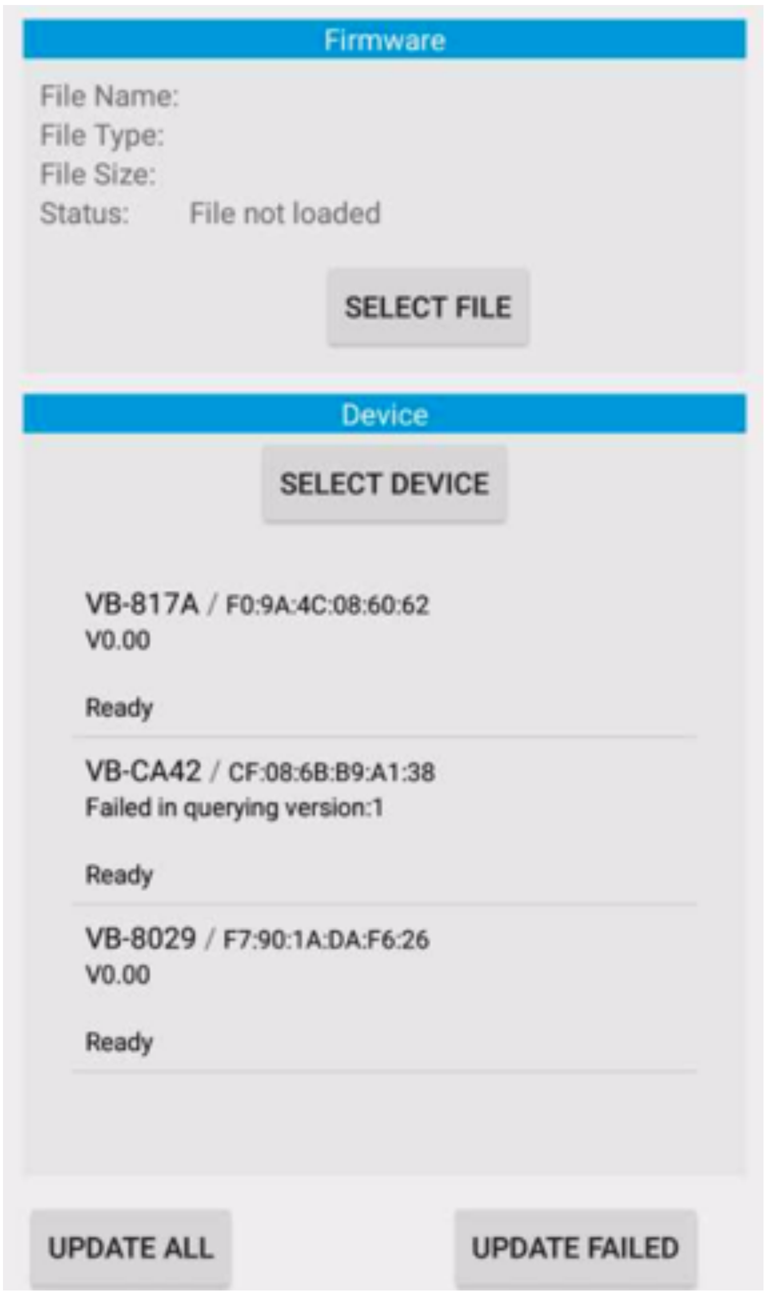
AVAILABLE DEVICES:

VB-817A / F0:9A:4C:08:60:62
V0.00

VB-CA42 / CF:08:6B:B9:A1:38
Failed in querying version:1

VB-8029 / F7:90:1A:DA:F6:26
V0.00

Just click the button “Update all”,all device selected will be update one by one.We want to update quickly,But the nordic sdk can’t update parallel,When the sdks supports update parallel,We can change this number in MainActivity,It’s the number how many device will be update parallel.
public static final int PARALLEL_UPDATE_COUNT = 1;
We can click a device at list to query the version again.
The update process is very complex,It maybe failed.We can use the button “UPDATE FAILED” to update the failed device only.



File Name:
File Type:
File Size:
Status: File not loaded

VB-817A / F0:9A:4C:08:60:62
V0.00

Ready

VB-CA42 / CF:08:6B:B9:A1:38
Failed in querying version:1

Ready

VB-8029 / F7:90:1A:DA:F6:26
V0.00

Ready