Developers

File Structure

This is a list and description of the most important folders in the repository.

```
├── config/                 configuration header files
├── └── featuresets/        different build configurations
├── inc/                    main include directory for C++ headers
├── ├── c/                  includes for c code
├── └── nrf52/              specific nrf52 includes
├── inc_c/                  include directory for C headers
├── linker/                 linker scripts
├── sdk/                    modified nRF SDKs
├── softdevice/             softdevice hex files
├── src/
├── ├── base/               wrappers for BLE stack functionality
├── ├── c/                  drivers and other c code
├── ├── mesh/               mesh functionality
├── ├── modules/            functionality wrapped in modules
├── ├── utility/            helper classes
├── ├── Boardconfig.cpp     runtime pin and board configuration
├── ├── Config.cpp          configuration
├── ├── FruityMesh.cpp      bootup and main functionality
├── ├── GlobalState.cpp     holds references to basic classes
├── └── Main.cpp            Startup Code
├── util/                   tools and utilities
└── src_examples/           code templates and examples
```

Functionality that is not part of the meshing algorithm should be placed in the `fruitymesh/modules/` folder and the module class extended. A module receives all events that it is interested in and can save and load its configuration from flash memory.

Have a look at Class Structure for some more detailed explanations on how the code is structured. Also keep in mind Instances when implementing new functionalities.

There are some utilities in the `/util` folder, for example a python script that opens a putty terminal for all connected serial devices and a Wireshark dissector that can be used together with the nRF Sniffer to inspect the advertising packets.

Configuring FruityMesh

Most settings are found in the `fruitymesh/src/` folder in `Config.h` but should be configured in the Featuresets. Module specific settings are kept within the module's header and cpp file.

Feature Sets

FruityMesh uses so called featuresets for creating different distributions for different use-cases. A featureset defines the compile time and run time configuration for a node. The cmake build process can be configured differently for a featureset, a number of different defines or macros can be used during compile time and different code is used during runtime. This allows us to tailor the firmware functionality and size to each use-case and also allows FruityMesh to compile for different chipsets from the same source. You can specify the featureset by calling cmake with `cmake --build . --target featureset_name`.

A featureset can contain a number of Board Configurations which allows you to flash and run the binary of the featureset on a number of different boards. The correct board configuration such as the pin configuration is then loaded by FruityMesh at runtime depending on the boardId stored in the UICR. If no boardId is stored in the UICR, the default boards will be loaded from `Boardconfig.cpp`

UICR

The UICR is used to store unique settings for each chip at flashing time. These are stored in an immutable persistent region of the chip. If this data is not present, default values from the code will be used. For production nodes, this data should be filled. The structure of this region is explained in our Specification.

It is possible to use the srec_cat (http://srecord.sourceforge.net/) utility to modify the generated .hex file with the necessary UICR data. This assumes that you have been using VsCode to compile the github_nrf52 featureset. Other tools using our CMake build will work as well, only the paths will be different.

- First, make sure that you have installed srec_cat (http://srecord.sourceforge.net/) and put it into your PATH so you can access it from everywhere
- Make sure that you are not overwriting the UICR settings in the `github_nrf52.cpp` featureset in the method `SetFeaturesetConfiguration_github_nrf52`
  - You should remove the part where the `NODE` settings are overwritten
- Next, go to the folder where your compiled binary is. E.g. `C:\projects\fruitymesh\_build\vscode`
- Open a command prompt and execute the following exemplary srec_cat command for adding UICR data to a .hex file before flashing.

> This is only valid for the NRF52 chipset family (Customer UICR Data is located at 0x10001080) and you must change the address offset to match your chipset when working with a different chip.

Exemplary UICR creation for node FFBBB

```C++
//Use the srec_cat utility
srec_cat
//Write the Magic Number so that the Mesh Node knows that there is data in the UICR
-generate 0x10001080 0x10001084 -constant-l-e 0x00F07700 4
//Use Boardid 4 for the NRF52-DK (PCA10040)
-generate 0x10001084 0x10001088 -constant-l-e 4 4
//Deprecated Field should be filled with FFFF....FFFF
-generate 0x10001088 0x10001090 -repeat-string %FF%FF%FF%FF%FF%FF%FF%FF
//Set a randomly generated unique NodeKey, in this example: 00:01:02:03:04:05:06:07:08:09:0A:0B:0C:0D:0E:0F
-generate 0x10001090 0x100010A0 -repeat-string %00%01%02%03%04%05%06%07%08%09%0A%0B%0C%0D%0E%0F
//Use Manufacturer Id 0x024D from M-Way Solutions (BLE SIG Company Identifier)
-generate 0x100010A0 0x100010A4 -constant-l-e 0x024D 4
//Put the node into unenrolled state by default
-generate 0x100010A4 0x100010A8 -constant-l-e 0 4
//Use a default nodeId of 1 in the unenrolled state
-generate 0x100010A8 0x100010AC -constant-l-e 1 4
//Use Device Type STATIC
-generate 0x100010AC 0x100010B0 -constant-l-e 1 4
//Serial Number Index for FMBBB (Make sure to read our Specification about Serial Numbers!)
-generate 0x100010B0 0x100010B4 -constant-l-e 2673000 4
//NetworkKey 11:11:11:11:11:11:11:11:11:11:11:11:11:11:11:11
-generate 0x100010B4 0x100010C4 -repeat-string %11%11%11%11%11%11%11%11%11%11%11%11%11%11%11%11
//Create a new file in intel-hex format that contains all the changes
github_nrf52_merged.hex -intel -output github_nrf52_merged_node_FMBBB.hex -intel -output_block_size 16

//Here ist the full command without any comments so that you can paste it into a terminal:
srec_cat -generate 0x10001080 0x10001084 -constant-l-e 0x00F07700 4 -generate 0x10001084 0x10001088 -constant-
l-e 4 4 -generate 0x10001088 0x10001090 -repeat-string %FF%FF%FF%FF%FF%FF%FF%FF -generate 0x10001090
0x100010A0 -repeat-string %00%01%02%03%04%05%06%07%08%09%0A%0B%0C%0D%0E%0F -generate 0x100010A0 0x100010A4 -
constant-l-e 0x024D 4 -generate 0x100010A4 0x100010A8 -constant-l-e 0 4 -generate 0x100010A8 0x100010AC -
constant-l-e 1 4 -generate 0x100010AC 0x100010B0 -constant-l-e 1 4 -generate 0x100010B0 0x100010B4 -constant-
l-e 2673000 4 -generate 0x100010B4 0x100010C4 -repeat-string %11%11%11%11%11%11%11%11%11%11%11%11%11%11%11%11
github_nrf52_merged.hex -intel -output github_nrf52_merged_node_FMBBB.hex -intel -output_block_size 16
```

You can check the generated file with the following command. Afterwards you can open the generated .txt file to see the binary data. Scroll down to the bottom to see the UICR data.

Converting an intel hex file (.hex) to a hex dump

```C++
srec_cat github_nrf52_merged_node_FMBBB.hex -intel -output github_nrf52_merged_node_FMBBB.txt -hex_dump
```

Afterwards, you can flash the created .hex file by using the `nrfjprog` utility. Remember that this .hex file is only intended for a single mesh node and that you must create other .hex files for each of your chips.

Flashing the created .hex file to a node

```C++
nrfjprog --chiperase --program github_nrf52_merged_node_FMBBB.hex --reset
```

Memory Requirements

FruityMesh doesn't run on devices with only 16kb of RAM. It may be possible to optimize the build and resize some buffers, but this is currently not supported. The binary of FruityMesh is around 50kb depending on the configuration and will easily fit on devices with 256kb flash together with the softdevice and still be updatable using dual bank updates.

Want To Contribute?

All kinds of contributions are welcome. Before you start coding, please contact us to align the development process.

About Questions

If you have a general question, the best way is to open a new issue and label it with "question". This way, a knowledge base of questions and answers is maintained for easy access in the future. If it is a commit-specific comment or question, you can just comment under the commit.

About Forking

We'd love to develop the FruityMesh protocol as an interoperable protocol that works across devices from different developers. If you want to make any changes to the protocol itself, please contact us first so that we can work out a mutual agreement. Every implementation that is compatible with the current official release of FruityMesh is welcome to use the M-Way Solutions Company identifier (0x024D) in the manufacturer specific data along with the current mesh identifier. Be sure to read the Specification for some basics. This is only very basic documentation, we try to continually improve the specification and add more details. In the meantime, do not hesitate to contact us or have a look in the implementation.

About Documentation

When adding documentation for a module, make sure to check the Module Documentation Template.

About Contributions

The implementation is written in C++. This makes it easy to implement new functionality and separate it from other parts in a clean way. Refactoring or refinement tips are welcome. If you contribute, please comment your code thorougly and keep the implementation as readable as possible. This will help other contributors understand the code quickly. If you have documentation to add, please post a pull request as well.

Licence

FruityMesh is published under the GPLv3 version, which is available in the repository.