Installing the toolchain

To build the example applications, you need a toolchain based on either CMake or SEGGER Embedded Studio. Install instructions are provided for Windows and Debian/Ubuntu. The steps should be similar for other platforms.

Table of contents

Important note about Python
Python is *not* required to build the mesh stack and examples. The nRF5 SDK for Mesh uses both Python 3 <https://www.python.org/downloads/> and Python 2.7 <https://www.python.org/downloads/>.

Tasks that require Python 3:

- working with Interactive PyACI script Mesh tool
- generating SEGGER Embedded Studio projects
- building documentation

Tasks that require Python 2:

- starting DFU transfer with nrfutil <https://github.com/NordicSemiconductor/pc-nrfutil/tree/mesh_dfu>. This is a legacy serial tool written for py2.

Remember to install and add to PATH <https://datascience.com.co/how-to-install-python-2-7-and-3-6-in-windows-10-add-python-path-281e7eae62a > both versions of Python on Windows.

Build environment based on SEGGER Embedded Studio

To use SEGGER Embedded Studio, download the installer from the SEGGER website <https://www.segger.com/downloads/embedded-studio/> and follow the installation instructions. You will find project files for each of the examples in their respective folders.

Moreover, you must install the following tools.

Download link	Recommended *minimum* version	Installation notes
SEGGER J-Link Software Pack <https://www.segger.com/downloads/jlink/>	6.16a	
Python 2.7 <https://www.python.org/downloads/>	2.7	Required for DFU.
Python 3.5 <https://www.python.org/downloads/>	3.5.1	Must be 32-bit for `nrfjprog` DLL to work. Ensure that `pip` is installed and that Python 3 is added to `PATH`.
nRF5 SDK <https://developer.nordicsemi.com/nRF5_SDK/nRF5_SDK_v15.x.x/>	15.2.0	Required for building with SEGGER Embedded Studio. See Downloading nRF5 SDK.

Build environment based on CMake

As an alternative to SEGGER Embedded Studio, CMake <https://cmake.org/> is a build management system used for managing an environment that is independent of the compiler and build system used. Version 3.6 or above is required by the mesh stack.

Required tools depend on your operating system:

- Installing CMake on Windows
- Installing CMake on Debian/Ubuntu

Additionally, if you want to build:

- documentation: install additional tools for building documentation;
- unit tests: install optional, additional tools for building unit tests.

Installing CMake on Windows

The following tools are required if you want to work with the nRF5 SDK for Mesh using CMake on Windows.

Download link	Recommended *minimum* version	Installation notes
nRF5x Command Line Tools <https://www.nordicsemi.com/DocLib/Content/User_Guides/nrf5x_cltools/latest/UG/cltools/nrf5x_installation>	9.5.0	Ensure that all command line tools are available in a folder referenced bythe system path (for example, the `PATH` environment variable).
SEGGER J-Link Software Pack <https://www.segger.com/downloads/jlink/>	6.16a	
Python 2.7 <https://www.python.org/downloads/>	2.7	Required for DFU.
Python 3.5 <https://www.python.org/downloads/>	3.5.1	Must be 32-bit for `nrfjprog` DLL to work. Ensure that `pip` is installed and that Python 3 is added to `PATH`.
CMake <https://cmake.org/download/>	3.9.0	Download the latest installer and follow the installation instructions.
Ninja <https://github.com/ninja-build/ninja/releases>	1.7.2	Preferred build system on Windows. Download the binary from the ninja-build release page <https://github.com/ninja-build/ninja/releases> and place it in a suitable folder.
GNU ARM Embedded Toolchain <https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads>	6-2017-q2-update (6.3.1)	One of two alternative build systems available on Windows. Download the arm-none-eabi-gcc <https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads> installer and follow the installation instructions. **Warning** Do not use the `8-2018-q4-major` version released on December 20, 2018, as it contains a bug that breaks the CMake building process on Windows.
ARM Compiler Version 5	5	The other alternative build system available on Windows. Follow the instructions provided for armcc v5 <https://developer.arm.com/products/software-development-tools/compilers/arm-compiler/downloads/version-5>. The armcc v5 toolchain is also provided by Keil and comes bundled with the Keil uVision IDE.
nRF5 SDK <https://developer.nordicsemi.com/nRF5_SDK/nRF5_SDK_v15.x.x/>	15.2.0	Required for building with CMake. See Downloading nRF5 SDK with CMake.

You can also install optional, additional tools for building unit tests.

Installing CMake on Debian/Ubuntu

For Debian/Ubuntu, most tools are available from the system package manager `apt`.

The following tools are required if you want to work with the nRF5 SDK for Mesh using CMake on Debian/Ubuntu.

Download link	Recommended *minimum* version	Installation notes

nRF5x Command Line Tools <https://www.nordicsemi.com/DocLib/Content/User_Guides/nrf5x_cltools/latest/UG/cltools/nrf5x_installation>	9.5.0	Reload the udev rules after installing the nRF5x Command Line Tools with the following commands:
		
		`sudo udevadm control --reload`
		`sudo udevadm trigger --action=add`
		
		Ensure that all command line tools are available in a folder referenced bythe system path (for example, the `PATH` environment variable).

SEGGER J-Link Software Pack <https://www.segger.com/downloads/jlink/>	6.16a	
Python 2.7 <https://www.python.org/downloads/>	2.7	Required for DFU. See the Installing Python on Debian/Ubuntu section below.
Python 3.5 <https://www.python.org/downloads/>	3.5.1	Ensure that `pip` is installed and that Python 3 is added to `PATH`. See the Installing Python on Debian/Ubuntu section below.
CMake <https://cmake.org/download/>	3.9.0	For Ubuntu versions older than `zesty`, a manual installation of CMake is required as the version available in the package manager is older than 3.6. Visit CMake <https://cmake.org/> to download the latest release and follow the installation instructions. - Install CMake with the following command: `sudo apt-get install cmake cmake-curses-gui` - Ensure that your CMake version is at least 3.6 with the

		following command: `cmake --version`

GNU ARM Embedded Toolchain <https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads>	6-2017-q2-update (6.3.1)	As the version usually found in the Debian package manager is quite old (4.9.3), install the toolchain in the following way (alongside GDB, the GNU Debugger for ARM):
		
		`sudo add-apt-repository ppa:team-gcc-arm-embedded/ppa`
		`sudo apt-get update`
		`sudo apt-get install gcc-arm-embedded`

make	-	Default build system on Debian/Ubuntu. Usually comes with the distribution. As an alternative, you can use Ninja.
Ninja <https://github.com/ninja-build/ninja/releases>	1.7.2	Alternative build system on Debian/Ubuntu. You can install it with the following command: `sudo apt-get install ninja-build`
nRF5 SDK <https://developer.nordicsemi.com/nRF5_SDK/nRF5_SDK_v15.x.x/>	15.2.0	Required for building with CMake. See Downloading nRF5 SDK with CMake.

You can also install optional, additional tools for building unit tests.

Installing Python on Debian/Ubuntu

The default Python version that comes with most Linux distributions is Python 2.7, but the nRF5 SDK for Mesh requires Python 3.5. It is recommended to use `virtualenv` to manage Python versions. It makes managing Python settings across different projects easy.

To install Python:

1. Run the following command `$ sudo apt-get install virtualenv`
2. Make a directory to keep your virtual environments in and create a new environment for mesh development:
   ```
   $ mkdir virtualenvs
   $ virtualenv -p python3 virtualenvs/mesh
   ```
3. Activate the environment:
   ```
   $ source virtualenvs/mesh/bin/activate
   ...
   ```

```
$ which python
/home/<user-name>/virtualenvs/mesh/bin/python
$ which pip
/home/<user-name>/virtualenvs/mesh/bin/pip
```

This will set the `python` and `pip` commands to point to the version within the given environment. All packages installed through `pip` will be local to the active environment.

You can deactivate the environment with the command `$ deactivate`. The environment will only be set for the active shell session.

To make this virtual environment the default when starting a new shell, add the following to your `~/.bashrc` file:

```
source virtualenvs/mesh/bin/activate
```

Additional tools for building documentation

If you want to build the documentation, make sure that the following tools are installed and available from the command line:

- Doxygen <https://www.stack.nl/~dimitri/doxygen/>
- Graphviz
- Mscgen

Optional: Additional tools for building unit tests (host)

The nRF5 SDK for Mesh contains a set of unit tests that verify module behavior. These unit tests run on the host system (PC, not the nRF5 device), and are built with GCC.

The following tools are required for building unit tests.

Download link	Windows or Debian/Ubuntu	Installation notes
Git <https://git-scm.com>	Both	Required for the installation of CMock and Unity. On Debian/Ubuntu, you can install it with: `$ sudo apt-get install git`

CMock <https://github.com/ThrowTheSwitch/CMock>	Both	Used by the unit tests to generate mocks.
		Make sure to clone the CMock repository recursively in the same directory as the nRF5 SDK for Mesh:
		`git clone https://github.com/ThrowTheSwitch/CMock.git <https://github.com/ThrowTheSwitch/CMock.git> --recursive cmock`
		The directory structure should look like this:
		`. +-- cmock/ +-- nrf5_sdk_for_mesh/`

Ruby <https://www.ruby-lang.org/>	Both	Required by CMock. On Debian/Ubuntu, you can install it with the following command: `sudo apt-get install ruby`
Unity <https://github.com/ThrowTheSwitch/Unity>	Both	Unit testing framework that is used for running the tests. CMock bundles Unity as a submodule, but you can also use a different version.
GCC compiler	Both	**Windows**: Available through MinGW. **Debian/Ubuntu**: Available in the distribution by default.
MinGW <https://sourceforge.net/projects/mingw/files/>	Windows	Required to use the standard GCC compiler on Windows. Install the `mingw-base` and ensure that the 32-bit version is installed or that 32-bit libraries are available.
`libpthread`	Windows	Needed for the multithreaded test. Install it using `mingw-get.exe`. From the command line, call the following command: `mingw-get install libpthread`
`gcc-multilib`	Debian/Ubuntu	**Optional.** Required to enable compilation for a 32-bit architecture on a 64-bit system (`-m32`). Install it with the following command: `sudo apt-get install gcc-multilib`
`lcov`	Debian/Ubuntu	**Optional.** Required if you want to generate code coverage report. Install it with the following command: `sudo apt-get install lcov`

Downloading nRF5 SDK

The nRF5 SDK for Mesh now *requires* the nRF5 SDK to compile. By default, the nRF5 SDK is expected to be stored next to the nRF5 SDK for Mesh, in a directory structure that looks like this:

```
.
+-- nrf5_sdk_for_mesh/
+-- nRF5_SDK_15.2.0_9412b96/
```

You can get the correct SDK either manually or using a custom CMake target.

Downloading nRF5 SDK manually

Download the nRF5 SDK version 15.2.0 from the Nordic Semiconductor Developer website <http://developer.nordicsemi.com/nRF5_SDK/nRF5_SDK_v15.x.x/>. Extract the package in the same folder as the nRF5 SDK for Mesh to match the folder structure above.

Downloading nRF5 SDK using a custom CMake target

1. Generate CMake build files:

```
nrf5_sdk_for_mesh $ mkdir build
nrf5_sdk_for_mesh $ cd build
build $ cmake -GNinja ..
```

You will get a warning that the nRF5 SDK is not found.

2. Run the `nRF5_SDK` target:

```
build $ ninja nRF5_SDK
```

This command downloads and extracts the correct nRF5 SDK in the folder next to the nRF5 SDK for Mesh.

3. Re-run CMake and it will pick up the correct path:

```
build $ cmake ..
```