# Contributing to NodeMCU

:+1::tada: First off, thanks for taking the time to contribute! :tada::+1:

The following is a set of guidelines for contributing to NodeMCU on GitHub. These are just guidelines, not rules, use your best judgment and feel free to propose changes to this document in a pull request.

It is appreciated if you raise an issue \_before\_ you start changing NodeMCU, discussing the proposed change; emphasizing that you are proposing to develop the patch yourself, and outlining the strategy for implementation. This type of discussion is what we should be doing on the issues list and it is better to do this before or in parallel to developing the patch rather than having "you should have done it this way" type of feedback on the PR itself.

### Table Of Contents

\* [General remarks](#general-remarks)

\* [Development environment setup](#development-environment-setup)

\* [Writing Documentation](#writing-documentation)

\* [Working with Git and GitHub](#working-with-git-and-github)

\* [General flow](#general-flow)

\* [Keeping your fork in sync](#keeping-your-fork-in-sync)

\* [Commit messages](#commit-messages)

\* [For collaborators](#for-collaborators)

\* [Handling releases](#handling-releases)

## General remarks

We are a friendly and welcoming community and look forward to your contributions. Once your contribution is integrated into this repository we feel responsible for it. Therefore, be prepared for constructive feedback. Before we merge anything we need to ensure that it fits in and is consistent with the rest of NodeMCU.

If you made something really cool but won't spend the time to integrate it into this upstream project please still share it in your fork on GitHub. If you mention it in an issue we'll take a look at it anyway.

## Development environment setup

Use the platform and tools you feel most comfortable with. There are no constraints imposed by this project. You have (at least) two options to set up the toolchain to build the NodeMCU firmware:

- [Full-fledged Linux environment](http://www.esp8266.com/wiki/doku.php?id=toolchain#how\_to\_setup\_a\_vm\_to\_host\_your\_toolchain), either physical or virtual.

- [Docker image](https://hub.docker.com/r/marcelstoer/nodemcu-build/) which allows running the build inside the container as if you were running a build script on your local machine.

## Writing Lua Code

A great resource about writing Lua for NodeMCU can be found in [Lua Developer FAQ](https://nodemcu.readthedocs.io/en/latest/lua-developer-faq/) - make sure to read it! When you're writing your Lua code and it's not working as it should you can test it with `luacheck` tool that can help you find various types of bugs. To install it you have to install [luarocks](https://luarocks.org/) and use command `sudo luarocks install luacheck` to install the tool. Now you're ready to go! By using this command (assuming you're in `nodemcu-firmware` directory):

`luacheck --config tools/luacheck\_config.lua <your file to check>`

you can look for bugs and problems within the code!

## Writing Documentation

The NodeMCU documentation is maintained within the same repository as the code. The primary reason is to keep the two in sync more easily. It's thus trivial for the NodeMCU team to verify that a PR includes the necessary documentation. Furthermore, the documentation is merged automatically with the code if it moves from branch X to Y.

The documentation consists of a collection of Markdown files (see note on Markdown syntax at end of chapter) stored in the [`/docs`](docs) directory. With every commit, a human readable and browsable version is automatically built with [Read the Docs](https://readthedocs.io/) (RTD). The public NodeMCU documentation can be found at [nodemcu.readthedocs.io](http://nodemcu.readthedocs.io/).

There are essentially only two things to keep in mind if you're contributing a PR:

- If you add functions to or change functions of an \*existing module\* you should modify the module's `.md` file in [`/docs/en/modules`](docs/en/modules). Adhere to the existing documentation structure and keep functions in alphabetical order.

- If you add a \*new module\* you should, in addition to the above, also add a reference for the new `.md` file to [`mkdocs.yml`](./mkdocs.yml) (lines 32+). Note that modules are ordered alphabetically here as well.

If you also want to verify that all is well with your Markdown files you can install Python-based [MkDocs](http://www.mkdocs.org/), which is used by RTD to build the static HTML files, and run `mkdocs serve` in the root of your NodeMCU firmware directory.

A note on Markdown \*syntax\*. As MkDocs is Python-based it's no surprise it uses a [Python Markdown implementation](https://pythonhosted.org/Markdown/). The good news is that it sticks pretty closely to John Gruber's Markdown and also [supports tables and fenced code blocks](http://www.mkdocs.org/user-guide/writing-your-docs/#markdown-extensions) just like GitHub does.

A collection of doc-writing hints and tips is maintained on the [wiki](https://github.com/nodemcu/nodemcu-firmware/wiki/Notes-about-writing-docs).

If you're interested in some NodeMCU history you're welcome to read [issue #774](https://github.com/nodemcu/nodemcu-firmware/issues/774)

## Working with Git and GitHub

\*Pull requests for new features and fixes should be opened against the `dev` branch.\*

Avoid intermediate merge commits. [Rebase](https://www.atlassian.com/git/tutorials/merging-vs-rebasing) your feature branch onto `dev` to pull updates and verify your local changes against them before placing the pull request.

### General flow

1. [Fork](https://help.github.com/articles/fork-a-repo) the NodeMCU repo on GitHub.

1. [Create a branch](https://help.github.com/articles/creating-and-deleting-branches-within-your-repository/#creating-a-branch) in your fork on GitHub \*\*based on the `dev` branch\*\*.

1. Clone the fork on your machine with `git clone https://github.com/<your-account>/<nodemcu-fork>.git`

1. `cd <nodemcu-fork>` then run `git remote add upstream https://github.com/nodemcu/nodemcu-firmware.git`

1. `git checkout <branch-name>`

1. Make changes to the code base and commit them using e.g. `git commit -a -m 'Look ma, I did it'`

1. When you're done:

1. Think about [squashing (some of) your commits](http://www.andrewconnell.com/blog/squash-multiple-git-commits-into-one). There are [several ways](http://stackoverflow.com/a/5201642/131929) to do this. There's no need to squash everything into a single commit as GitHub offers to do this when we merge your changes. However, you might want to trim your commit history to relevant chunks.

1. Bring your fork up-to-date with the NodeMCU upstream repo ([see below](#keeping-your-fork-in-sync)). Then rebase your branch on `dev` running `git rebase dev`.

1. `git push`

1. [Create a pull request](https://help.github.com/articles/creating-a-pull-request/) (PR) on GitHub.

This is just one way of doing things. If you're proficient in Git matters you're free to choose your own. If you want to read more then the [GitHub chapter in the Git book](http://git-scm.com/book/en/v2/GitHub-Contributing-to-a-Project#The-GitHub-Flow) is a way to start. [GitHub's own documentation](https://help.github.com/categories/collaborating-with-issues-and-pull-requests/) contains a wealth of information as well.

As a Windows or Mac user you could also resort to [GitHub Desktop](https://desktop.github.com/). It's a mature GUI application that supports most of the tasks outlined above.

### Keeping your fork in sync

You need to sync your fork with the NodeMCU upstream repository from time to time, latest before you rebase (see flow above).

1. `git fetch upstream`

1. `git checkout dev` but you may do this for `master` as well

1. `git merge upstream/dev`

### Commit messages

From: [http://git-scm.com/book/ch5-2.html](http://git-scm.com/book/ch5-2.html)

<pre>

Short (50 chars or less) summary of changes

More detailed explanatory text, if necessary. Wrap it to about 72

characters or so. In some contexts, the first line is treated as the

subject of an email and the rest of the text as the body. The blank

line separating the summary from the body is critical (unless you omit

the body entirely); tools like rebase can get confused if you run the

two together.

Further paragraphs come after blank lines.

- Bullet points are okay, too

- Typically a hyphen or asterisk is used for the bullet, preceded by a

single space, with blank lines in between, but conventions vary here

</pre>

Don't forget to [reference affected issues](https://help.github.com/articles/closing-issues-via-commit-messages/) in the commit message to have them closed automatically on GitHub.

[Amend](https://help.github.com/articles/changing-a-commit-message/) your commit messages if necessary to make sure what the world sees on GitHub is as expressive and meaningful as possible.

## For collaborators

### Handling releases

- Create a [milestone](https://github.com/nodemcu/nodemcu-firmware/milestones) right after you cut a new release. Give it a meaningful name if you already have an idea what the scope of the upcoming release is going to be. Set the due date to ~2 months in the future.

- Add this milestone to every PR before you merge it. Also, add the milestone to PRs you want to see land in this milestone.

- Add notes to the description of the milestone in the course of the ~2 months it lives.

- Be careful and reluctant to merge PRs once we're past the 6-weeks mark of a milestone. Ideally, we don't merge anything in the last 2 weeks.

- Cutting a release

- Create a PR for the `master` branch for collaborators to approve.

- Once approved merge it. :exclamation::boom::exclamation: Make sure you do NOT "squash and merge" but make a regular merge commit!

- Fetch the changes into your local clone and create an annotated tag like so: `git tag -a <SDK-version>-master\_<yyyyMMdd> -m ""`, `git push --tags`

- Create a new [release](https://github.com/nodemcu/nodemcu-firmware/releases) based on the tag you just pushed. The version name is the same as the tag name.

- Write release notes. Mention breaking changes explicitly. Since every PR that went into this release is linked to from the milestone it should be fairly easy to include important changes in the release notes.