# Contributing to ThingPulse OLED SSD1306

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

The following is a set of guidelines for contributing to the ThingPulse OLED SSD1306 library 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 the code, 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)

\* [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)

## 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 code.

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.

## Writing Documentation

ThingPulse maintains documentation for its products at [https://github.com/thingpulse/docs/](https://github.com/thingpulse/docs/). If you contribute features for this project that require altering the respective product guide then we ask you to prepare a pull request with the necessary documentation changes as well.

## Working with Git and GitHub

Avoid intermediate merge commits. [Rebase](https://www.atlassian.com/git/tutorials/merging-vs-rebasing) your feature branch onto `master` 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) this repository 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 `master` branch\*\*.

1. Clone the fork on your machine with `git clone https://github.com/<your-account>/<esp8266-oled-ssd1306>.git`

1. `cd <weather-station-fork>` then run `git remote add upstream https://github.com/ThingPulse/esp8266-oled-ssd1306`

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

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/) contains a wealth of information as well.

### Keeping your fork in sync

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

1. `git fetch upstream`

1. `git checkout master`

1. `git merge upstream/master`

### 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.