Software Workflow

The purpose of this document is to spell out the general workflow and procedures that shall be followed by software developers on the CS team (and microcontroller developers on the EE team). This workflow is a slimmed-down variation of the Agile workflow. We primarily use Github's issue tracking and pull request features to keep track of our work. All software tasks that need to be performed are created into issues on Github. Collectively these issues are a list of all work that needs to get done, is in progress, or completed. Issues are usually created by the team lead. The word “issue” and “task” are used interchangeably for the rest of this document. For git, we'll be using the rebase model as described in the “Git Policies” documentation.

Milestones: Sprints and the Backlog

Issues on github can be assigned to something called a “milestone” in github. When a task is first created, it typically will be added to the “Backlog” milestone on Github. When we write software, we will perform 1-3 week (TBD) “sprints”. Before the sprint begins, tasks are created or pulled from the backlog, and then assigned to developers. The goal is that all issues are completed by the end of the sprint, therefore we try to be realistic when decided what tasks will be in the sprint. Any tasks not completed by the end of the sprint typically are placed in the next sprint, but again, we try to make sure we never have any uncompleted tasks.

Using Github Issues

When you click on an issue in Github, you'll see a new window with more information about the issue. Here you can have an extended description of an issue, and people add comments to the issue so a discussion can occur about the task. These comments can reference specific lines of code on a specific commit (https://help.github.com/articles/getting-permanent-links-to-files/), which is great for showing what you're talking about.

Overall Workflow

1. Grab an issue from the current sprint that is assigned to you. If no issues are assigned to you, you can pick out one that is not assigned to anyone, or talk to the team lead, who can help you find a task.
2. Check out your own branch on git, work on the code, make commits, rebase on dev often, make comments on the issue on github as necessary.
3. When you think the work is completed, create a pull request on Github
4. A code review is performed via Github's pull request framework
5. Perform any changes resulting from the code review
6. After the review is completed, the git manager will fast-forward merge our dev branch into your branch, and the pull request and issue can both be closed.

7. Go back to step 1!

Code Review/Pull Request Policy

The team lead will be the head of all code reviews, but anybody can participate. The more people looking at the code, the more likely we are to discover bugs before we do a pool test. All code should undergo review before being integrated to the dev branch Exceptions are allowed at the discretion of the team lead for small, quick fixes where the overhead of a review is not necessary.

Labels

Issues can also be flagged with labels that we can use for filtering issues. One such label is “impeded”, which should get flagged if the primary developer is stuck for some reason and needs help.