This is a reference guide for programmers that contains the necessary information for daily work on code.

This guide was intended to be more of a reference guide than a tutorial (but the tutorial parts are still in the back). There is probably too much information to comprehend in one go, but much of the stuff here will become second nature by simply working on actual code. I recommend reading though this once (or just skim it), and then referring to it occasionally to refresh your mind. By this time, I’m quite sure someone has made a Prezi/Power Point that introduces these concepts much more effectively (hint, hint).

General steps to making changes:

1. Start with no Labview projects open
2. Make a new branch or switch to a previous branch
3. Open project in Labview
4. **Do not switch branches** while project is open
5. Make changes to code
6. Save everything including the Labview project
7. Confirm you are on the correct branch
8. Choose which files to commit carefully
   1. It is an excellent idea to keep notes on what changes you made and in what file so you can know which files to commit and write a good commit message
   2. However, if you are unsure or in a hurry, committing all changes is not the end of the world. Just remember doing this will result in many headaches (possibly for other programmers) when it comes time to merge the branch.
9. Commit your selected files
10. Sync if branch already existed or publish the new branch if you just created it
11. When ready for release, make a pull request
12. One of two possibilities depending if pull request is:
    1. Accepted: sub-branch is merged into master by an admin
    2. Rejected: development continues on sub-branch
       1. Can make further commits after making pull request to fix issues since pull request is automatically updated.

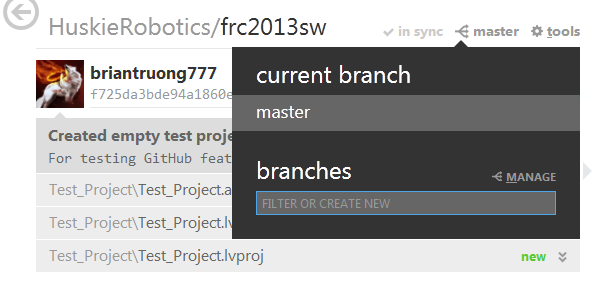
Work Flow:

* Master branch should always be ready-for-release
  + “Ready-for-release” means every commit on master should be runnable, competition-ready code
  + The only commits made to master should be:
    - Merges from sub-branches
    - Universal changes affecting all sub-branches
    - Preparation for future branches
  + Only admins should make commits to master
  + All code added to master must be thoroughly tested
  + At important commits, set up tags to easily find such commits. Examples:
    - New major versions of code with significant new features
    - Snapshot after testing extensively
    - Snapshot before trying experimental things
    - Code freeze before major competition
  + Only tag commits that are on the master branch
* Make sub-branches whenever any work needs to be done. Examples:
  + Sub-branches can be based on a sub-assembly (autonomous, drive, killer tennis ball of doom) and consist of any work relating to that sub-assembly. Would be continuously merged into master until all work on sub-assembly is done.
  + Sub-branches can be based on fixing a bug (autonomous timing is off, left drive PID is too jumpy, robot can’t stop dancing) and promptly merged and forgotten as soon as work is done.
  + Sub-branches can be based on a task (picking up an object, climbing only half-way up, doing my homework) and also promptly merged and forgotten as soon as work is done.
  + Sub-branches can be based on an experimental track to test new possibilities (PID drive system, vision-based distance, time machine).
  + Sub-branches **should not** be based on a single person or group. This should be avoided because it gives them an implied freewill to alter anything in the project resulting in painful merging down the road.
* Sub-branches should have these implied rules. These rules can be broken at the possible cost of merging issues down the road. I’m only calling them rules because breaking them should make you feel bad even if there is no other way.
  + A branch can alter only a certain set of files relating to its purpose
  + Such files are not in common among other branches
  + Branches are only merged after making a pull request no matter how insignificant
  + Any work relating to a branch’s purpose should be done in that branch
  + Branches are not owned by specific people
  + Branches can only be deleted after being merged into the parent branch (unless it is an experimental branch)
  + Branches can be kept as long as they are still useful
* Each branch can test its own code regardless of development on other branches
* When code has been tested, a pull request is made to merge a sub-branch into its parent branch
  + Follow with online discussion on GitHub if necessary
  + Admin will then handle merging (or not if request is rejected)
* Any changes that will affect multiple branches must be made in master or other parent branches
  + This will be done by an admin
  + These changes will be distributed by merging from master to all sub-branches if needed

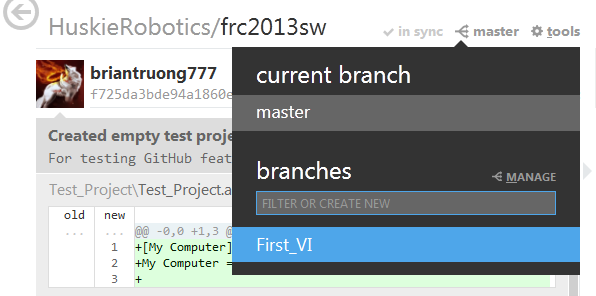
The following are tutorials that walk through how to do the various things mentioned above. Know that Github frequently changes their interface and some of the pictures will probably be out-of-date by the time you read this. The steps should still remain accurate.

Making a new branch:

1. Click the branch icon, ensure you are on the branch you want to branch off from, and type in a new branch name (don’t forget to publish the branch later)

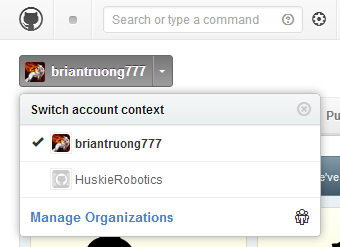


Switching to previous branch:

1. Click on the branch icon, and click on the particular branch

Making a pull request:

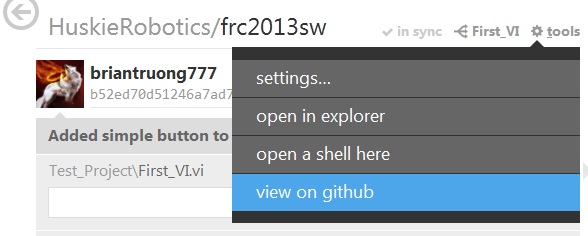
1. Go to github.com
   1. In the top right of the screen, click on your username to switch to the HuskieRobotics organization



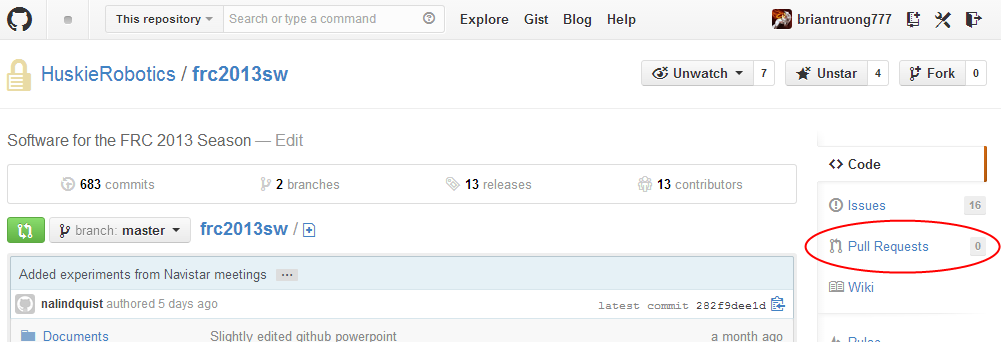
* 1. On the right side of the HuskieRobotics organization page, click on the repository holding the season’s code



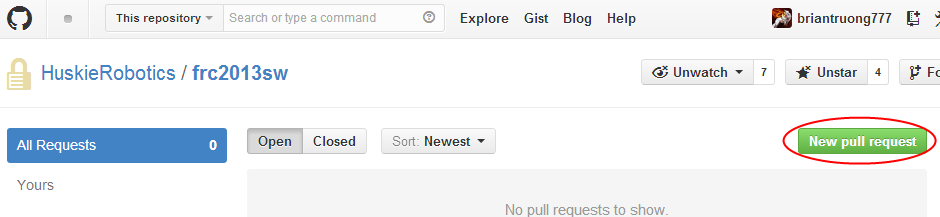
1. Alternatively in GitHub for Windows, click on tools, then “view on github” to go directly to the repository page



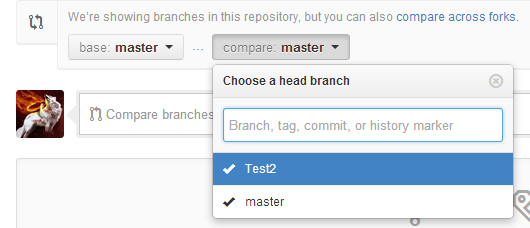
1. Click on the pull request button on the right side of the repository page



1. Click on the green “New Pull Request” button



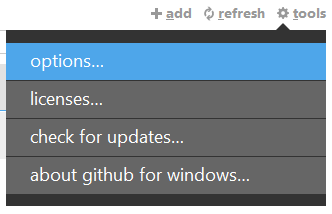
1. Select your sub-branch in the right-side dropdown and parent branch in the left-side dropdown (Merge from right-side branch to left-side branch)



1. Click on the “Click to create a pull request for this comparison” area to create a pull request for this comparison
2. Give the pull request a title and write a comment to describe overall changes since the creation of the branch

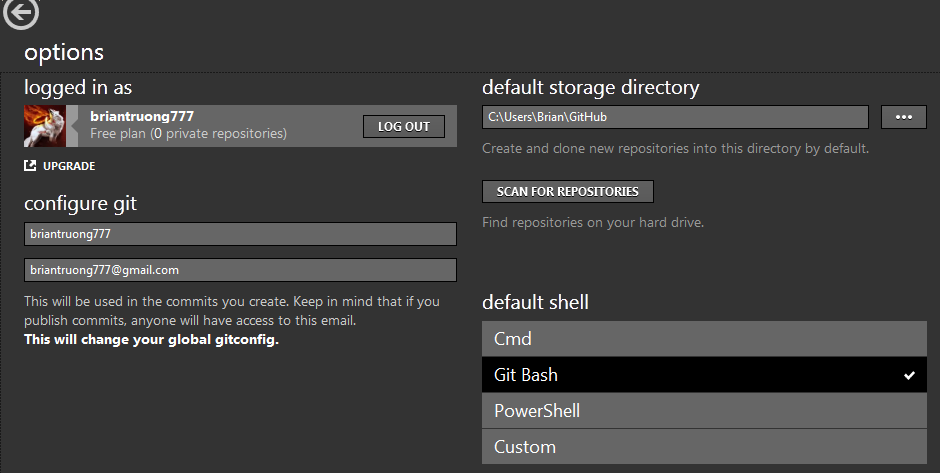
Setting up GitHub for Windows:

1. Download and install it from [windows.github.com](http://windows.github.com/)
   1. You do need an internet connection for the installer
2. Login with your GitHub user name and password
   1. You must login in order to adjust the following settings
3. Click on tools->options…

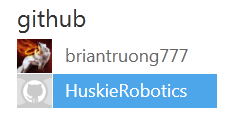


* 1. Make sure you are not currently in a repository
  2. You can also login here if you didn’t already

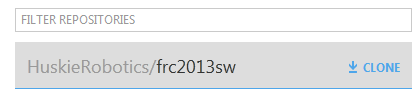
1. Configure a name and email to be attached to every commit you make
2. Select your default shell
   1. You can choose whatever you are familiar with
   2. Otherwise, I recommend Git Bash
3. Select default storage directory
   1. This is where all your local repos will go
   2. It’s a good idea to have a shortcut to this location for quick access
4. Click the update button
   1. Example of a correct setup:



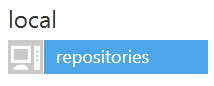
1. Select the remote repo “HuskieRobotics” from the list of local repos under “github”



1. Click on the clone button to create a local repo

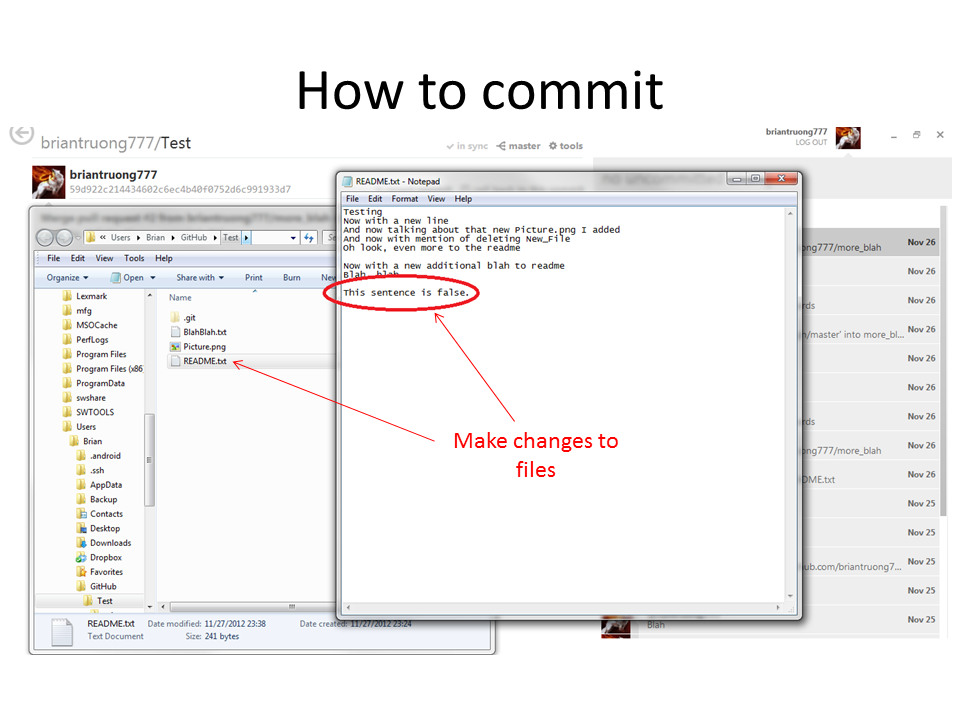


1. Click on the local repositories list to see your newly clone repo

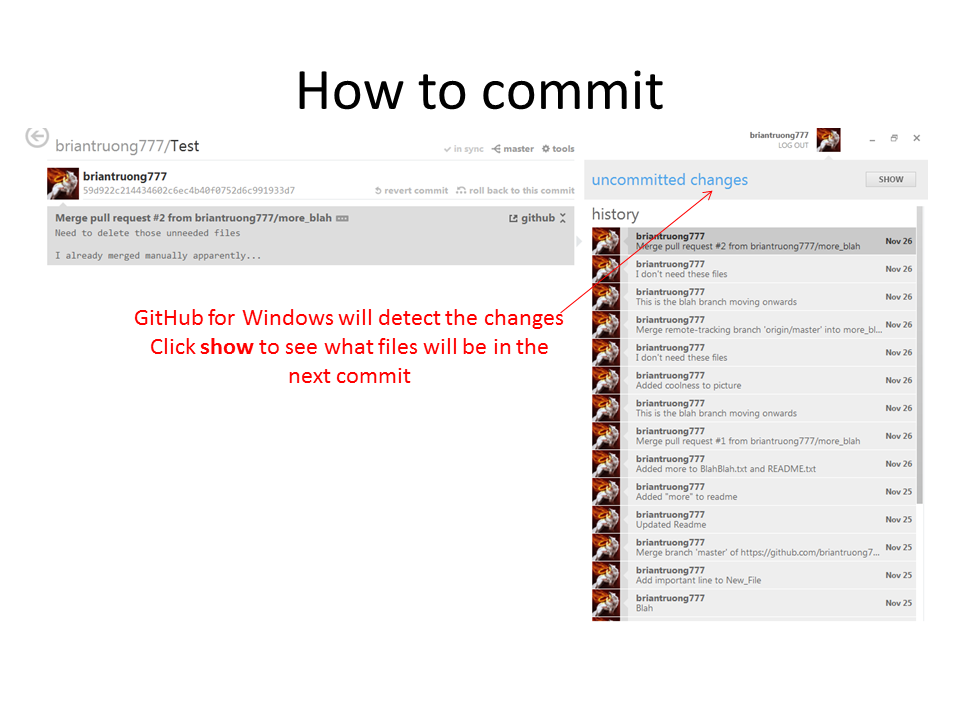


Making a commit

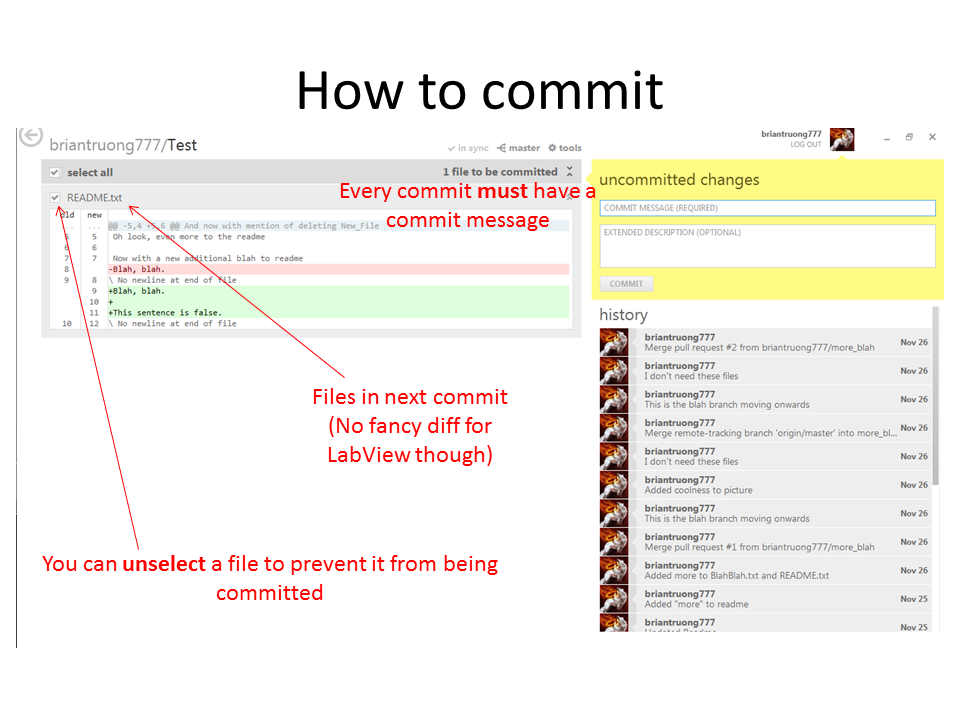
1. Make changes



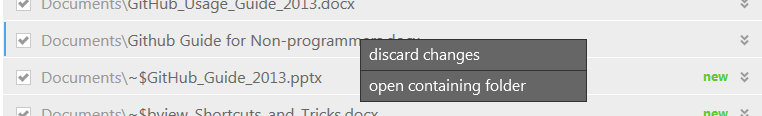
1. Go to Github for Windows and click on the “show” button



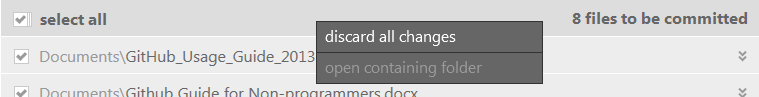
1. Select files to be committed and write up a commit message. Click commit.



1. Either do some more work and then make another commit, or continue to the next step when you are ready to push your commits to the server
2. Discard all files that you changed but didn’t commit. All files that show up after making a commit should belong in this category.
   1. If these files must be kept, save them outside of the repository or ask a knowledgeable programmer on how to use the command line to help you.
   2. Right click on a file, and then click on “discard changes” to discard changes to that file



* 1. You can also discard all changes by right clicking on the bar on top of all the files



1. Click on the sync/publish button.

