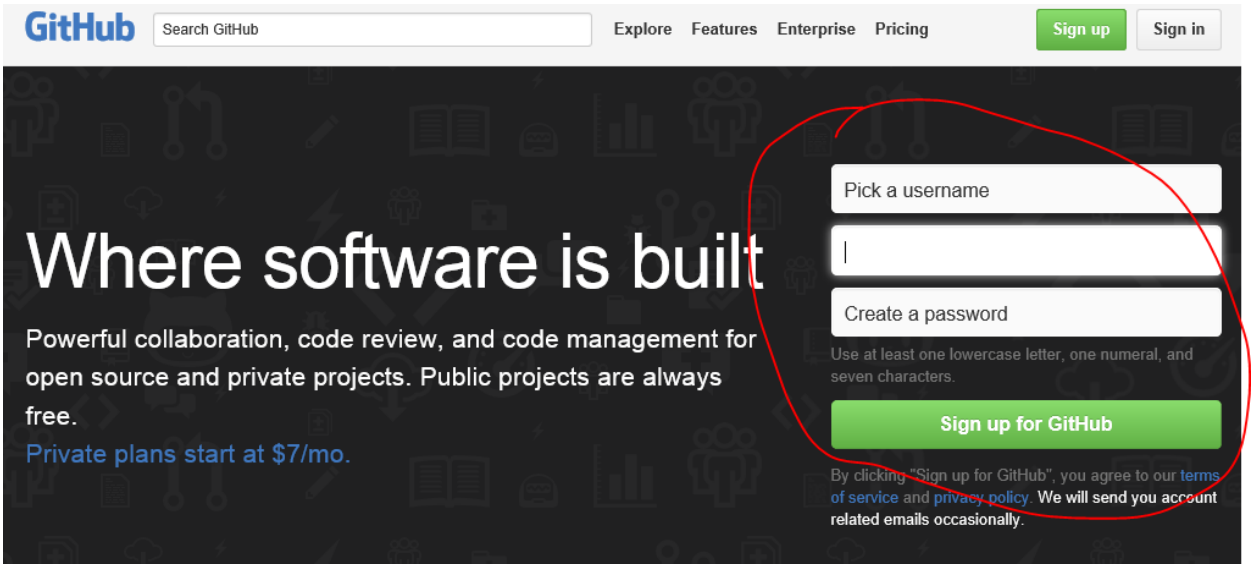
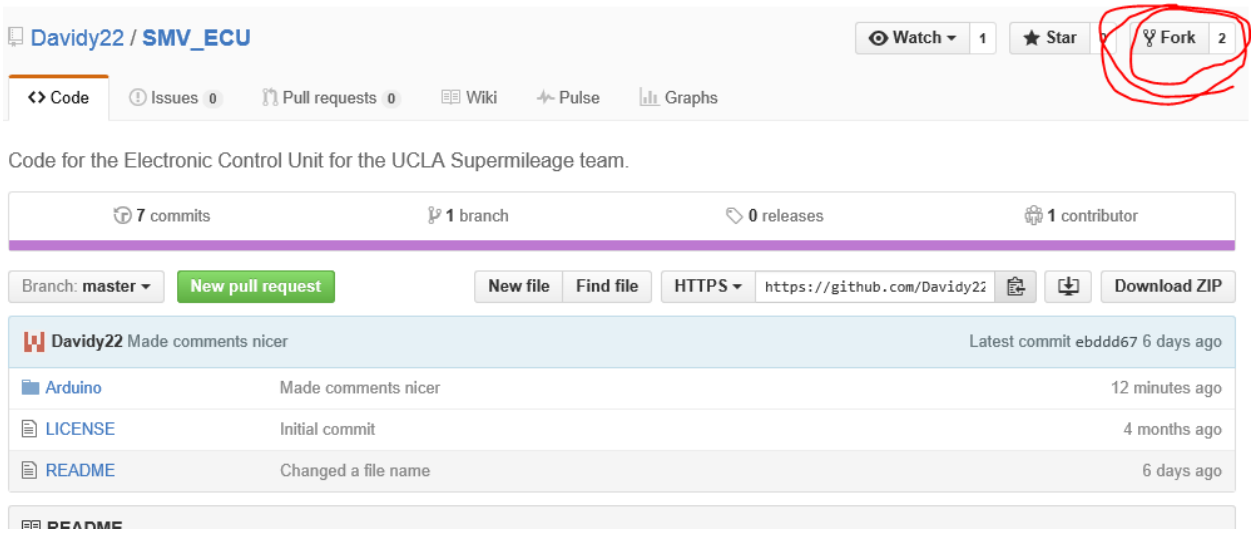
If you use a Mac or Linux and want to be l33t and use Git on the command line to contribute to this code, you can read up the overview of Git here: https://git-scm.com/book/en/v2/Distributed-Git-Contributing-to-a-Project. There's also a detailed description of how Git works in the overview, if you're interested. If you want to use smartgit instead because you want shiny buttons to point and click at, keep reading.

First, we're going to make a github account. Go to github.com, fill in the form on their front page and verify your email address with them.

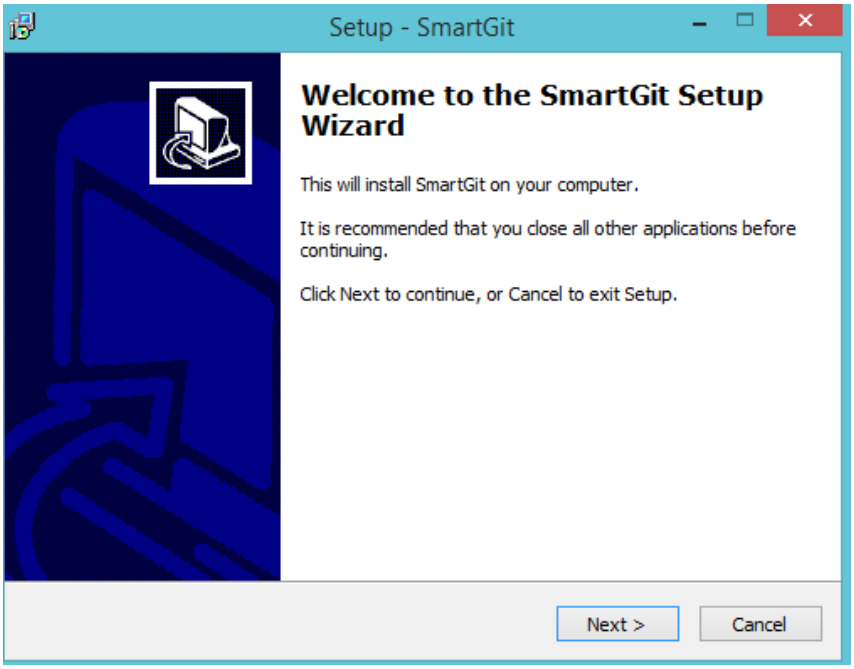


Then, go to https://github.com/Davidy22/SMV_ECU and click the fork button to make your own copy of the code.

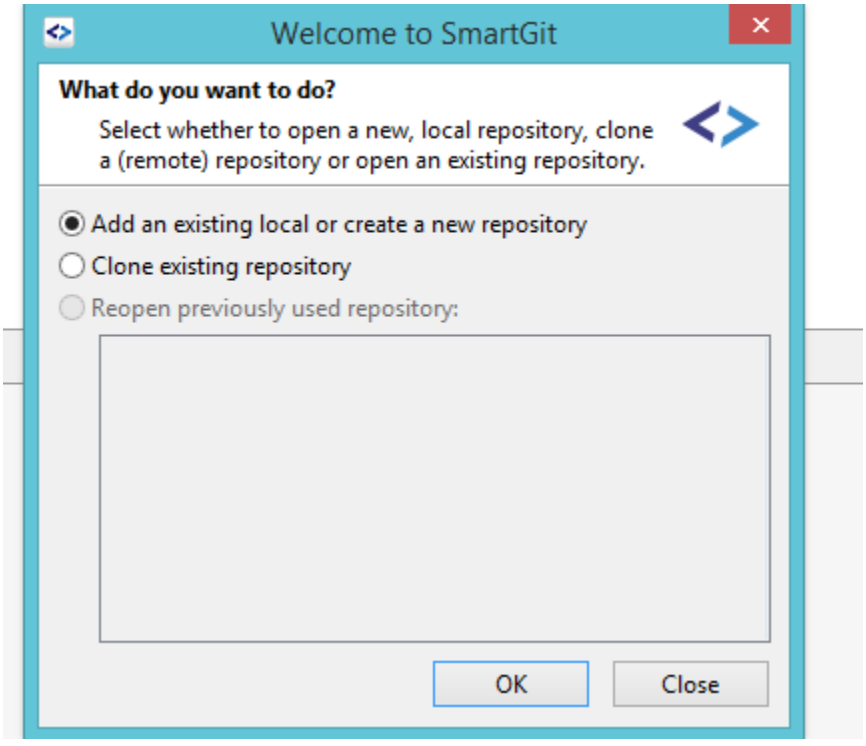


You now have your own copy of the source code. Next, we need to install git.

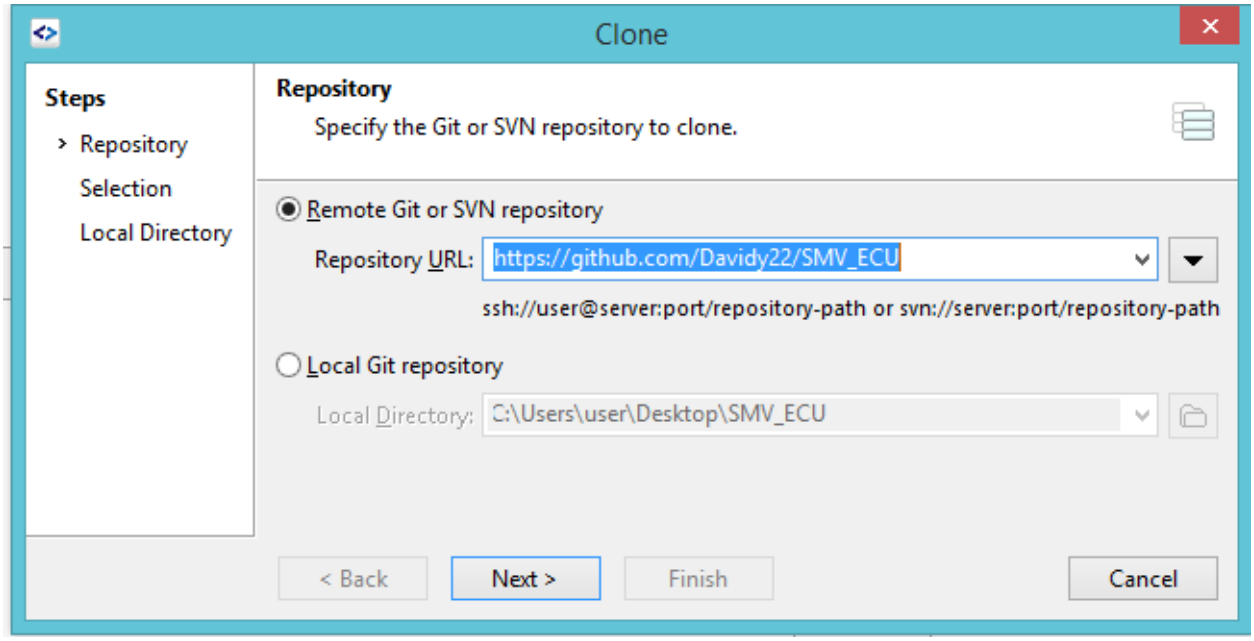
Download smartgit at http://www.syntevo.com/smartgit/. Unzip the file and run the .exe file.



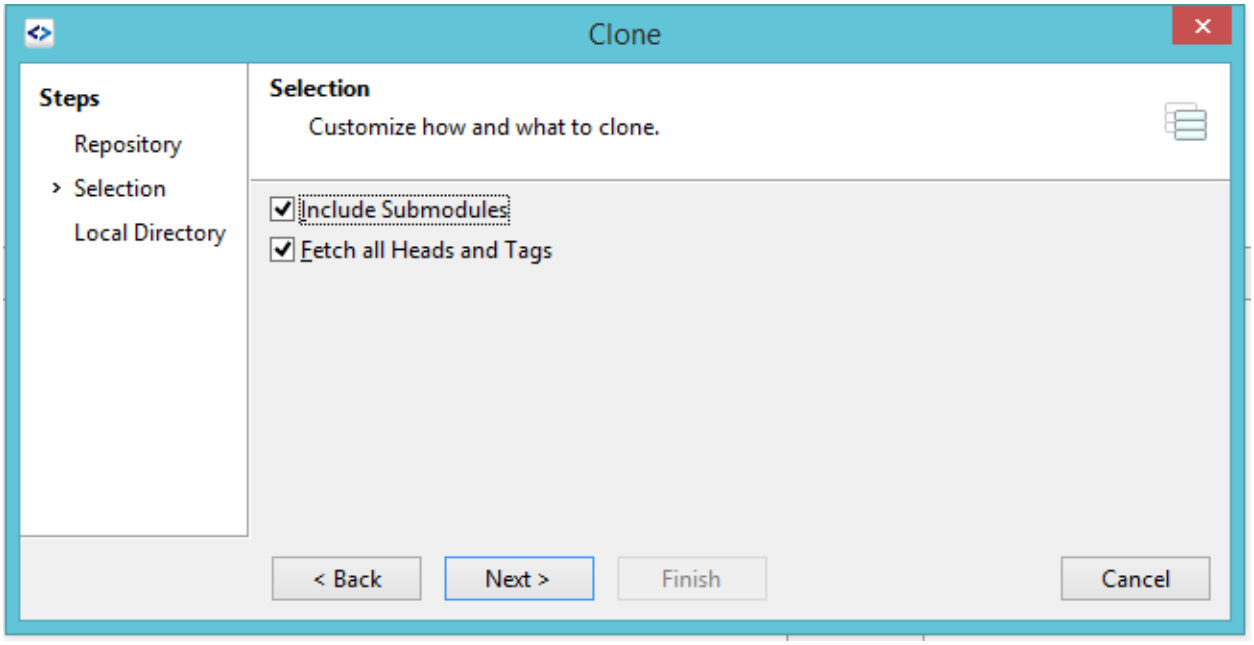
Click through this, then open smartgit when it finishes installing.



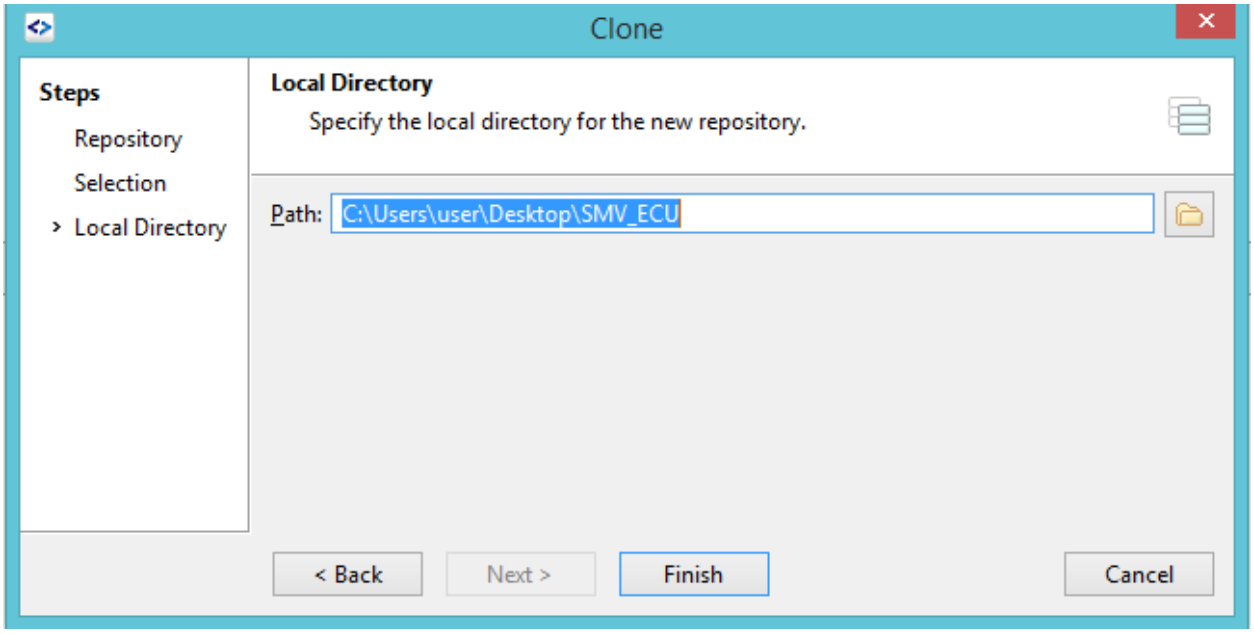
You want to clone a repository.



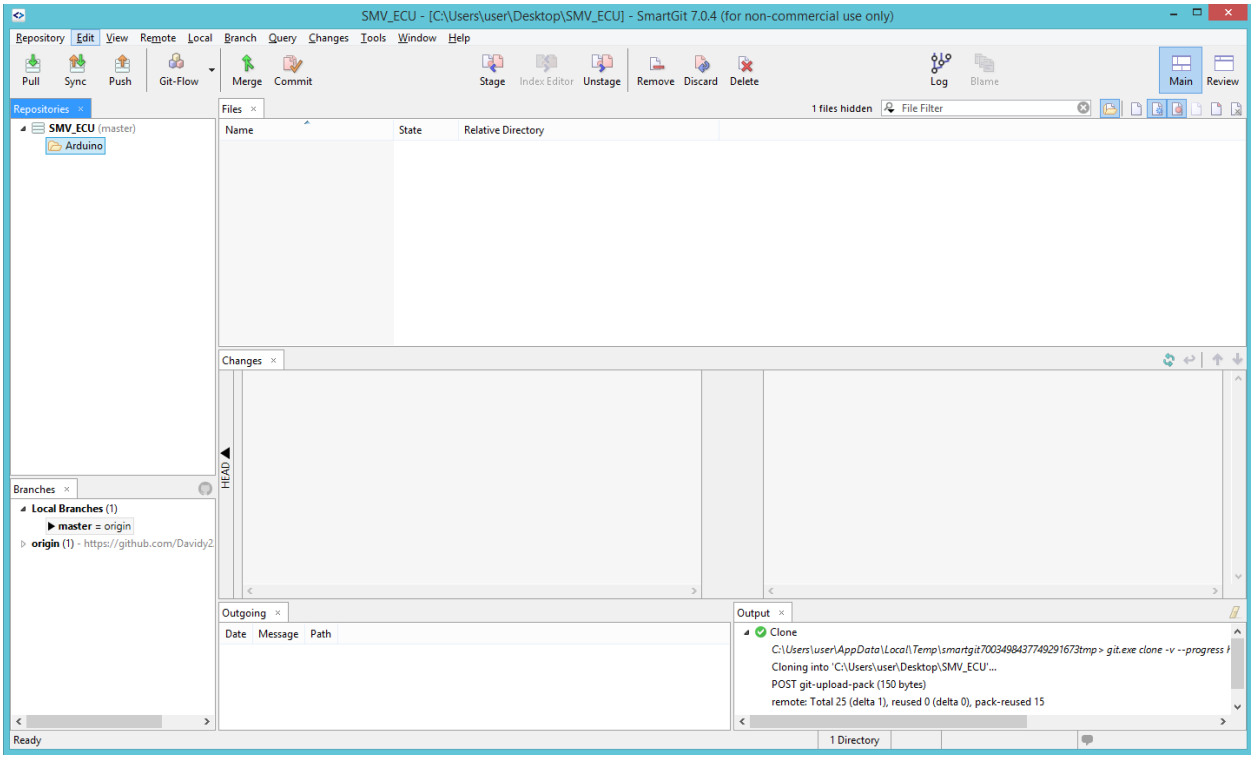
Make sure the window looks like this. Make sure the repository URL field contains the URL of your own fork of the source code.



Check both of these boxes.



Pick where you want the code to be stored. Hit finish and it'll start downloading the code.

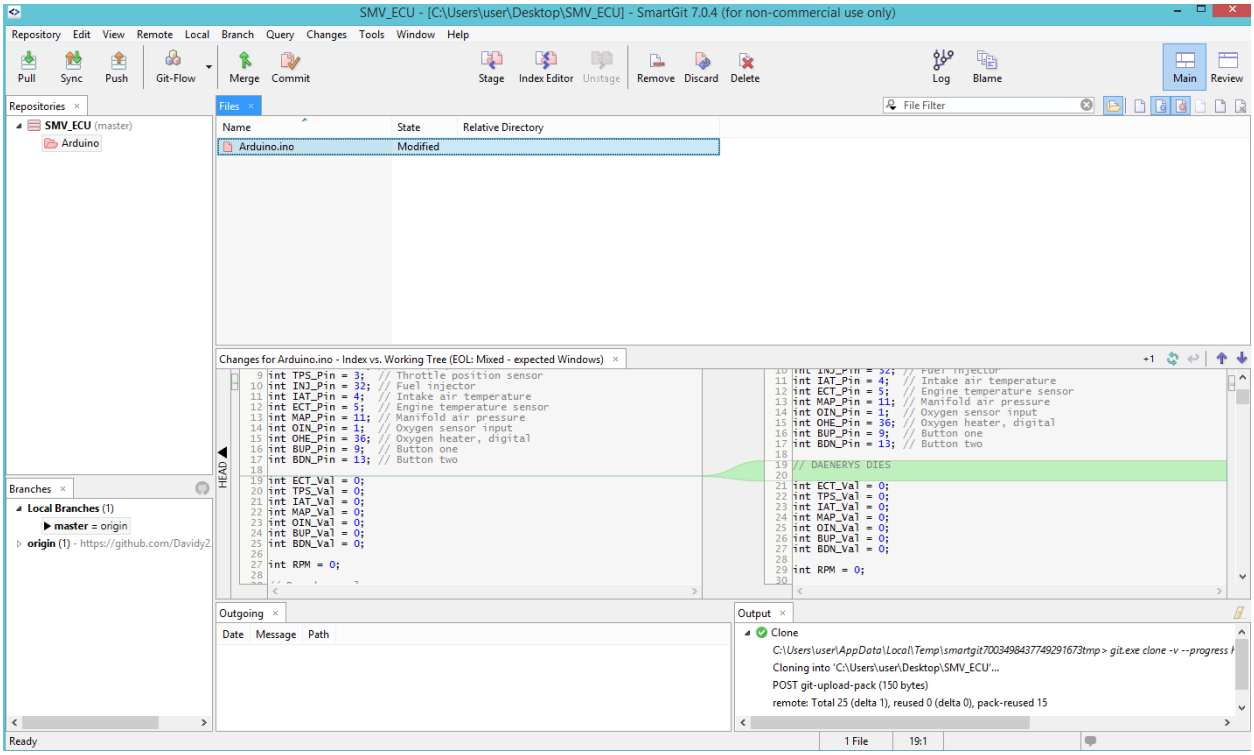


Your window should look something like this now. The code should also be where you chose to save it. The files area shows what files have been changed, and it should be blank. Let's make a change now.

```
#include <Wire.h>
#include <LiquidCrystal.h>
#include <math.h>
int incoming = 0;

LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
int HES_Pin = 2;  // Hall effect sensor
int PMP_Pin = 48; // Fuel pump
int TPS_Pin = 3;  // Throttle position sensor
int INJ_Pin = 32; // Fuel injector
int IAT_Pin = 4;  // Intake air temperature
int ECT_Pin = 5;  // Engine temperature sensor
int MAP_Pin = 11; // Manifold air pressure
int OIN_Pin = 1;  // Oxygen sensor input
int OHE_Pin = 36; // Oxygen heater, digital
int BUP_Pin = 9;  // Button one
int BDN_Pin = 13; // Button two

int ECT_Val = 0;
int TPS_Val = 0;
int IAT_Val = 0;
int MAP_Val = 0;
int OIN_Val = 0;
int BUP_Val = 0;
int BDN_Val = 0;
```

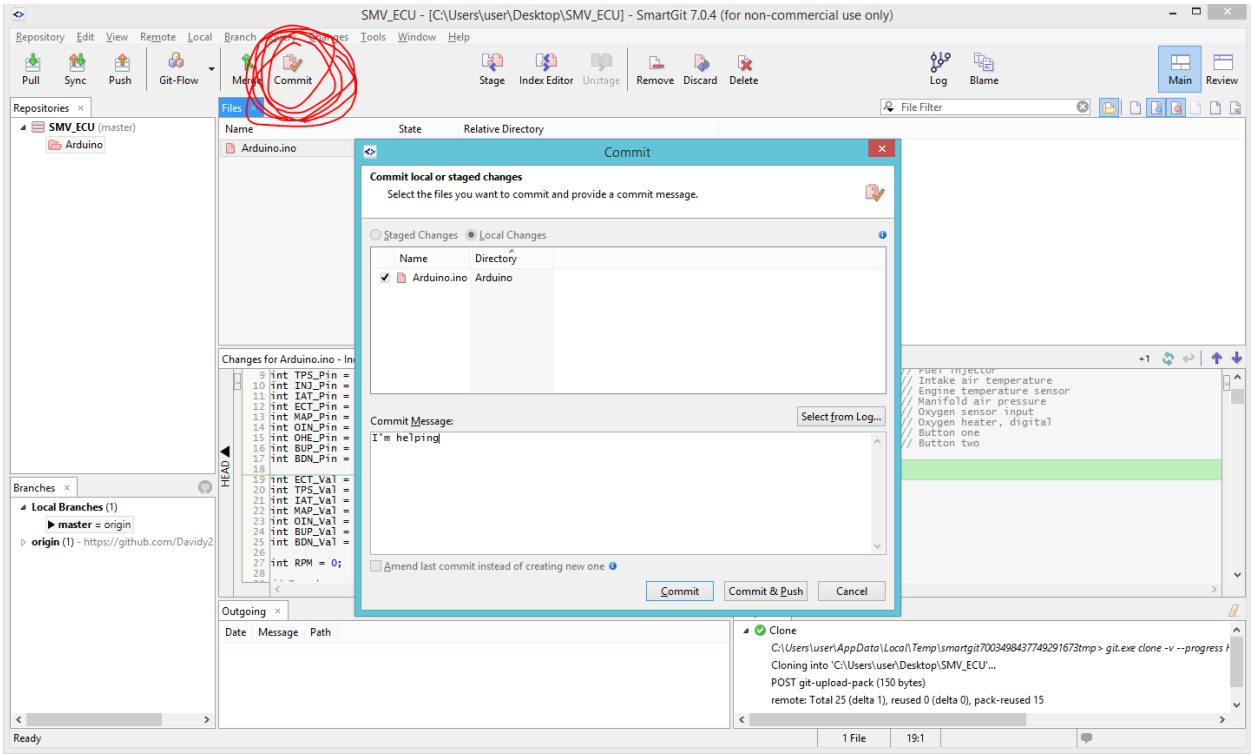
```
#include <Wire.h>
#include <LiquidCrystal.h>
#include <math.h>
int incoming = 0;

LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
int HES_Pin = 2;  // Hall effect sensor
int PMP_Pin = 48; // Fuel pump
int TPS_Pin = 3;  // Throttle position sensor
int INJ_Pin = 32; // Fuel injector
int IAT_Pin = 4;  // Intake air temperature
int ECT_Pin = 5;  // Engine temperature sensor
int MAP_Pin = 11; // Manifold air pressure
int OIN_Pin = 1;  // Oxygen sensor input
int OHE_Pin = 36; // Oxygen heater, digital
int BUP_Pin = 9;  // Button one
int BDN_Pin = 13; // Button two

// DAENERYS DIES

int ECT_Val = 0;
int TPS_Val = 0;
int IAT_Val = 0;
int MAP_Val = 0;
int OIN_Val = 0;
int BUP_Val = 0;
```

If we go back to the smartgit window, we see that Git can see our change.

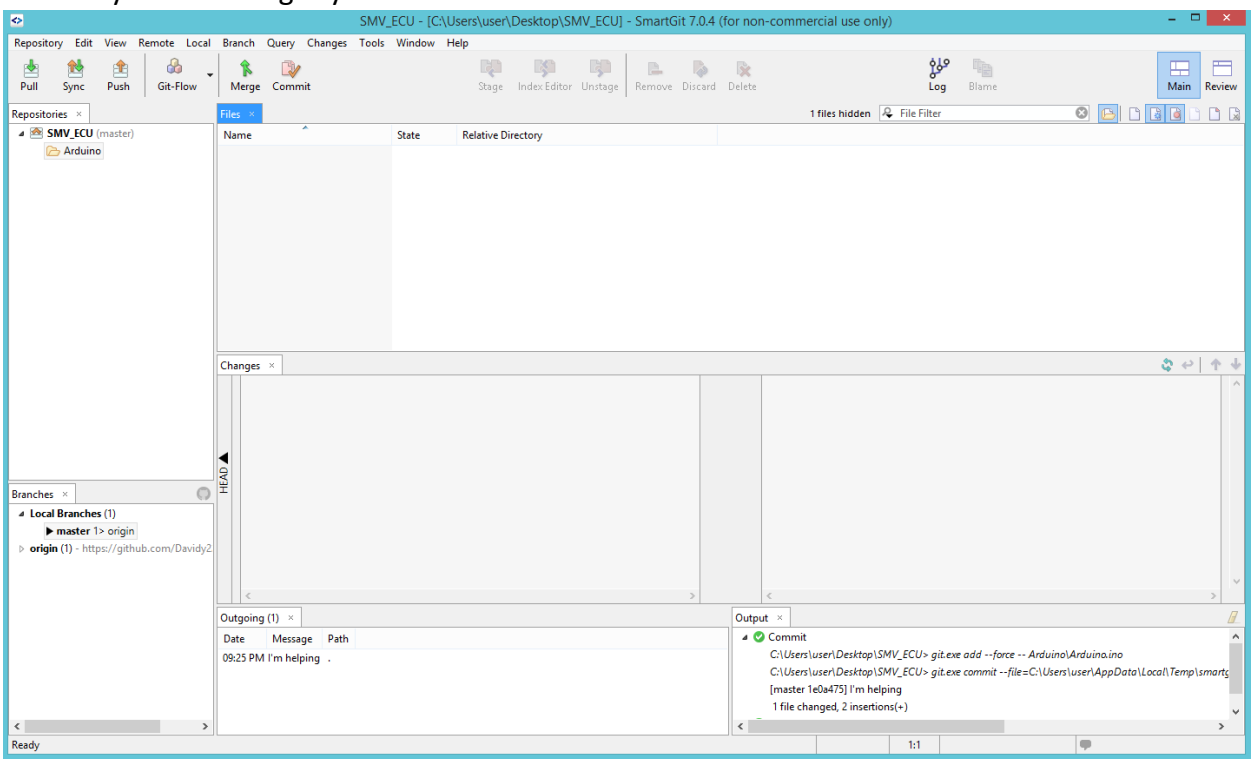


Git lets you save the current state of your code in the form of changes since the last time you saved and return to it later on if you want to undo code or switch between two different versions of your code for testing. Saving a state is called “committing.” Click the commit button

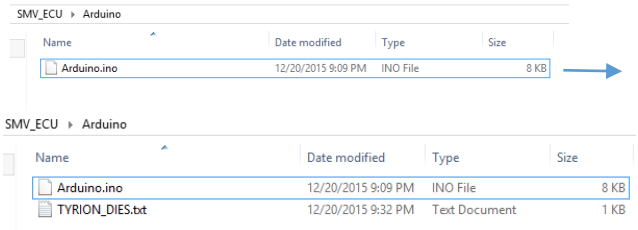
to save the valuable addition you've just made.



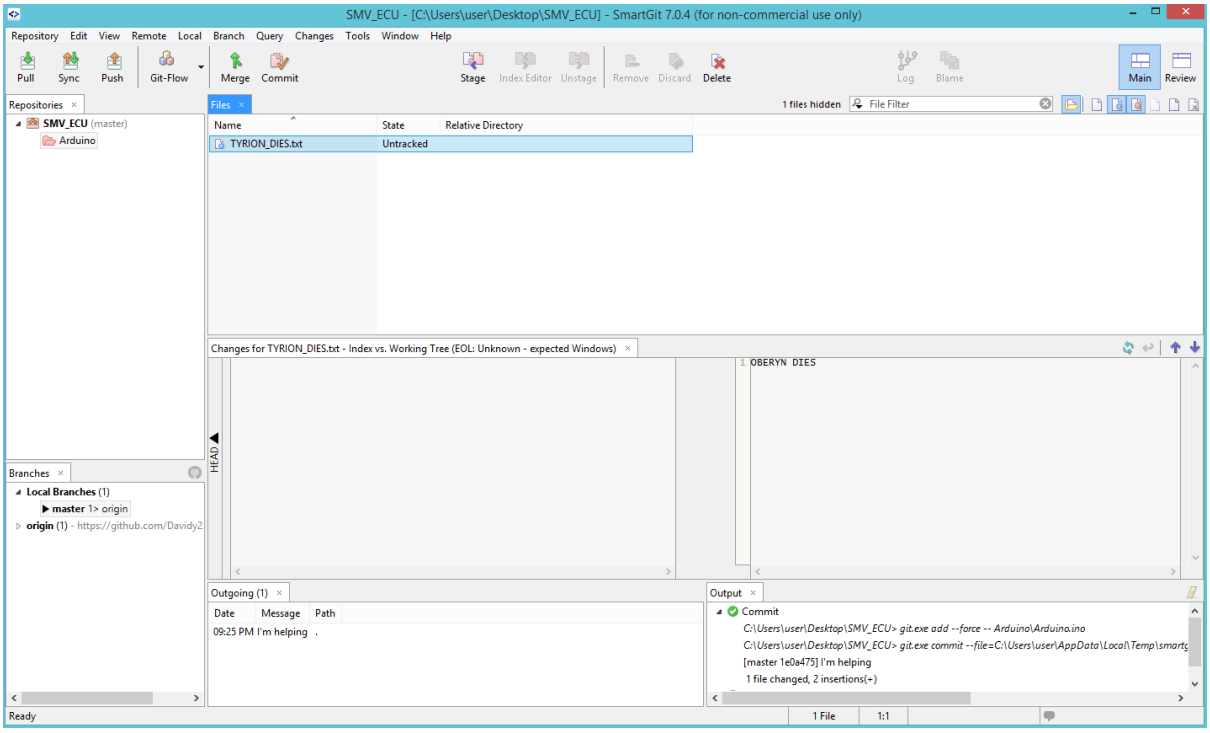
Tick all the files that you've changed that you want to save changes for and write a descriptive summary of the changes you made. Then click commit.



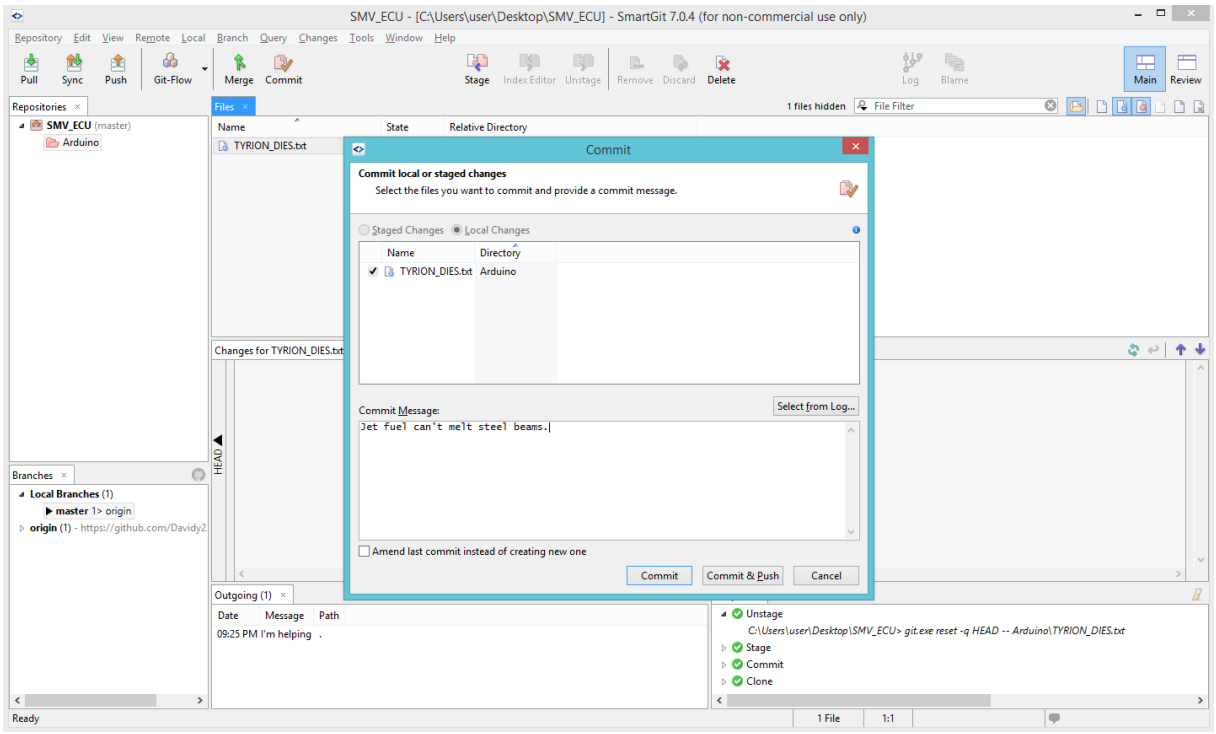
The files area is now blank again, because there are no more files that have been changed since your last commit. You can also add new files too.



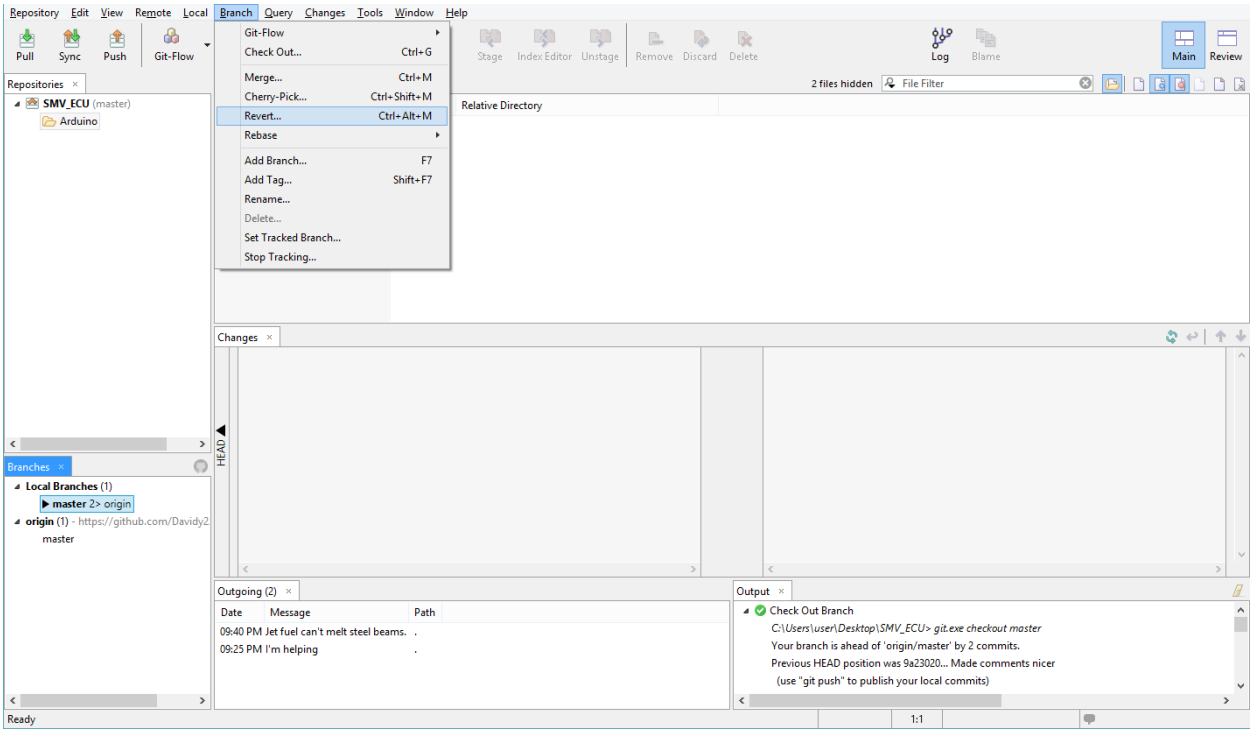
This new file will show up in smartgit too.

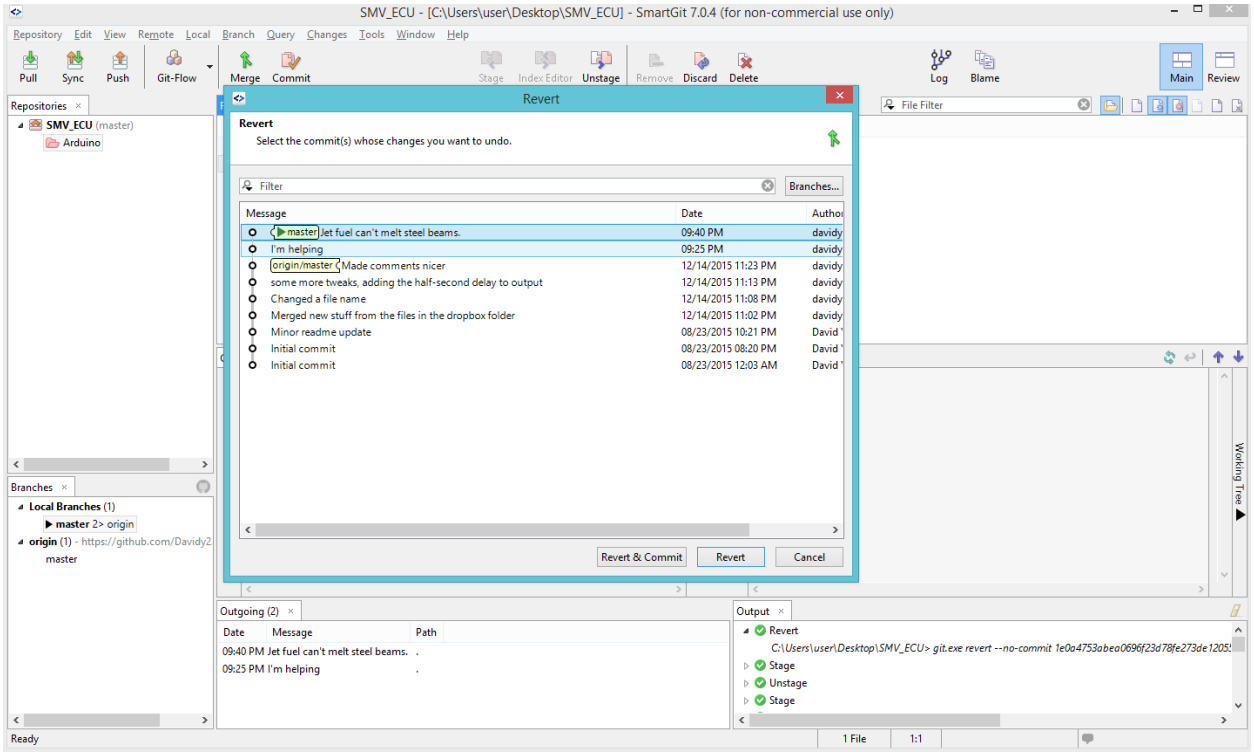


You can commit this new file the same way you committed the changes to an existing file.

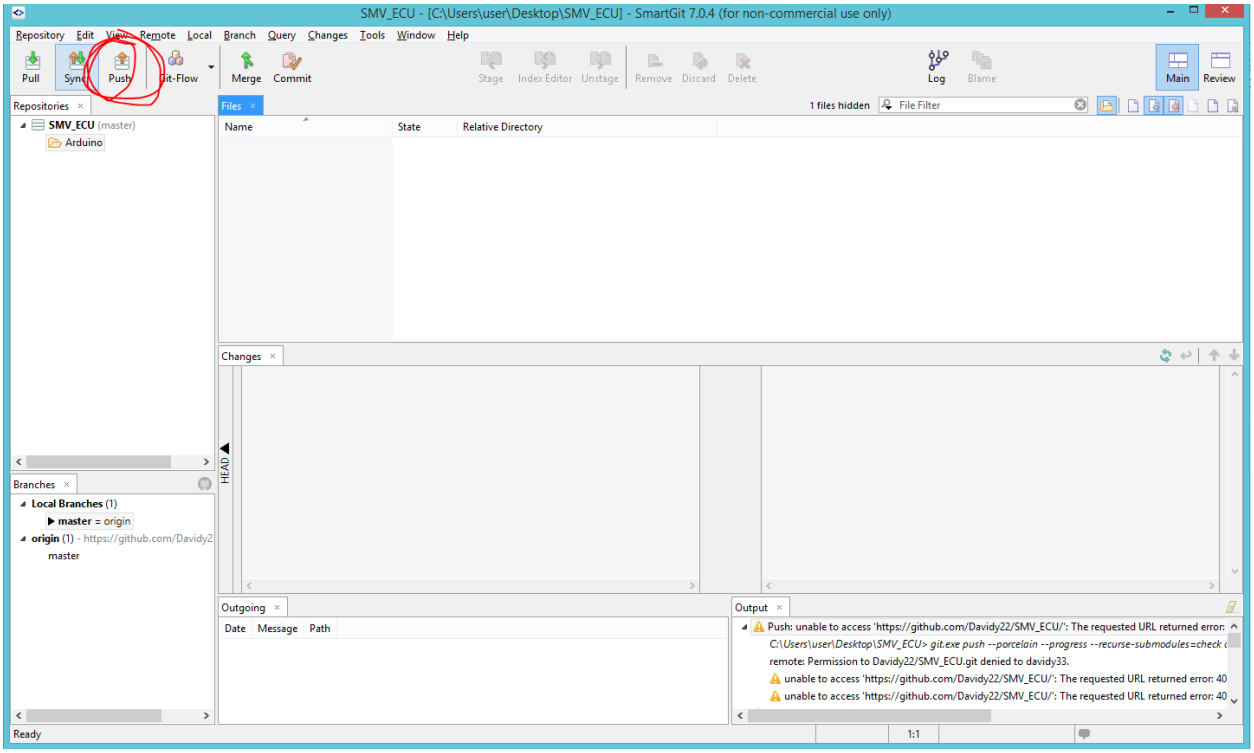


Now, some idiot is putting Game of Thrones spoilers in our repository, and we want to undo them. We can do this with the revert tool.

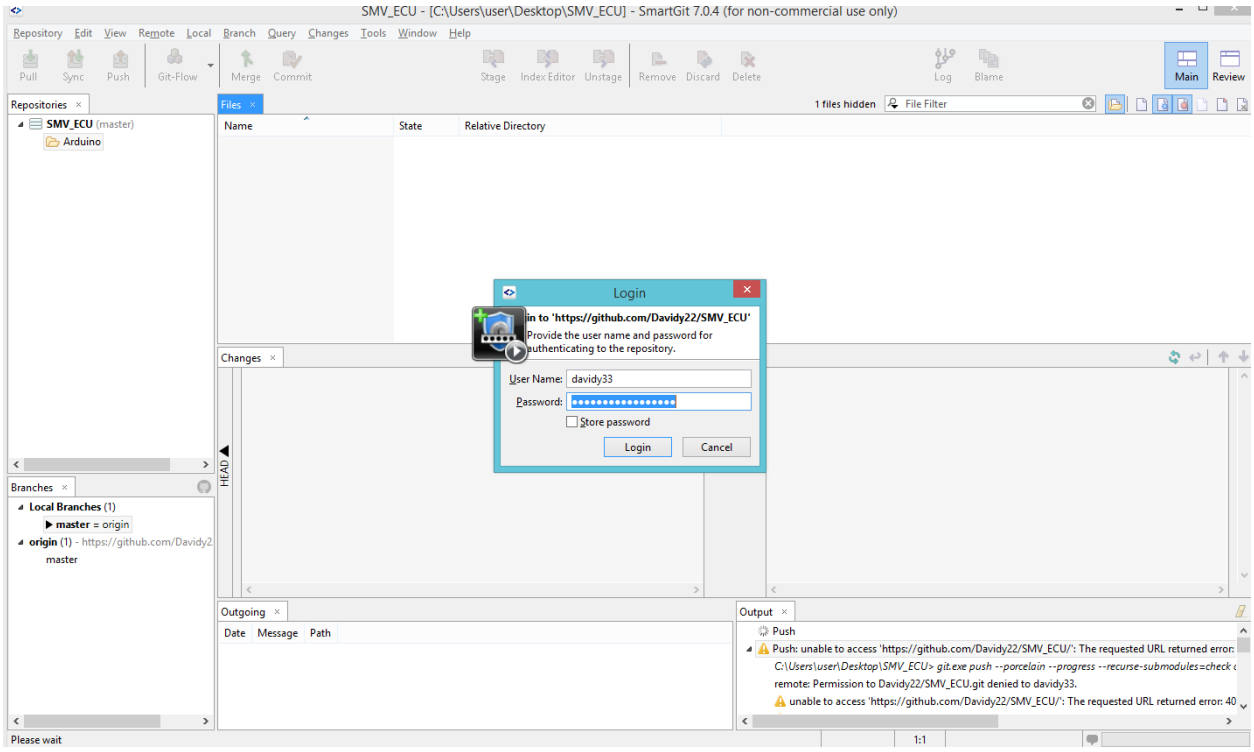




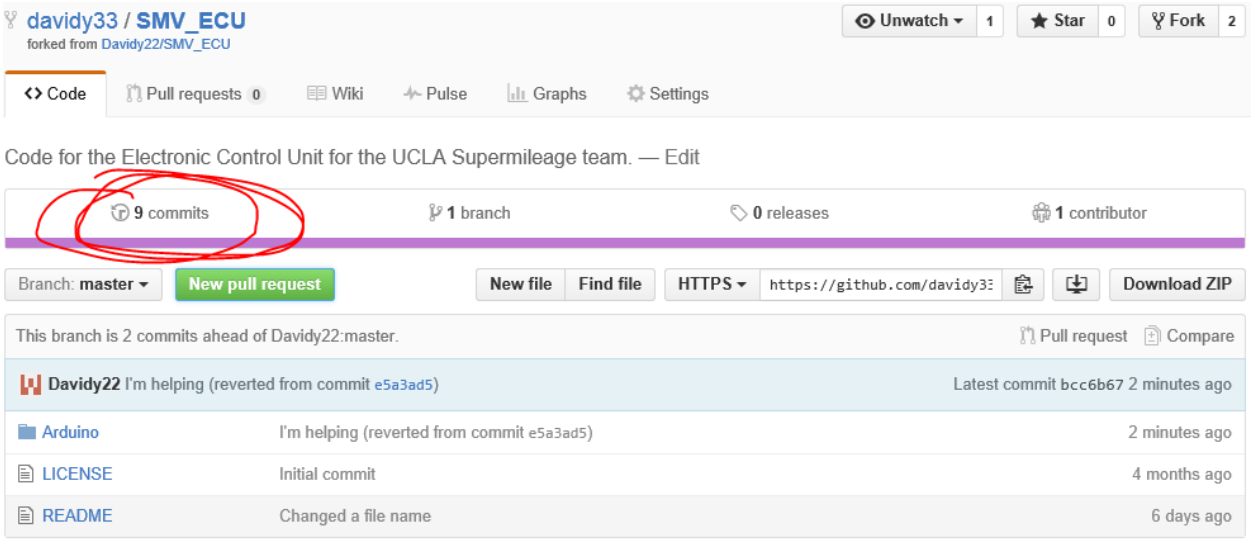
Double click on the offending commit to undo the changes that are saved in it. This will undo the changes made in that commit. When you feel like you like the changes you've made, you can push them online to github using the push button.

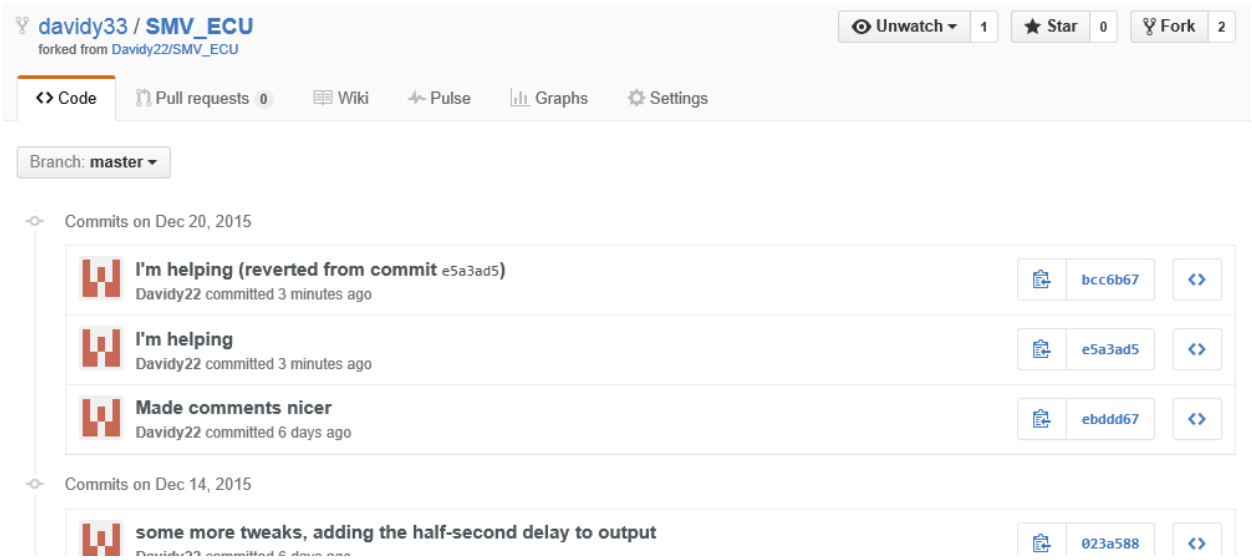


Just push to the current branch. The first time you push, you'll be asked to enter a username and password. Use your github login.

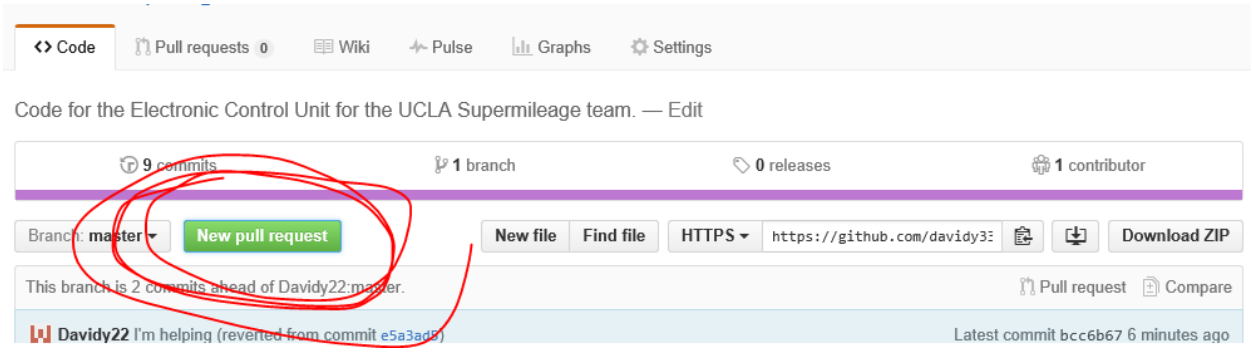


Your changes are now saved online. You can examine your commits by heading over to your github repository and clicking the commits button.

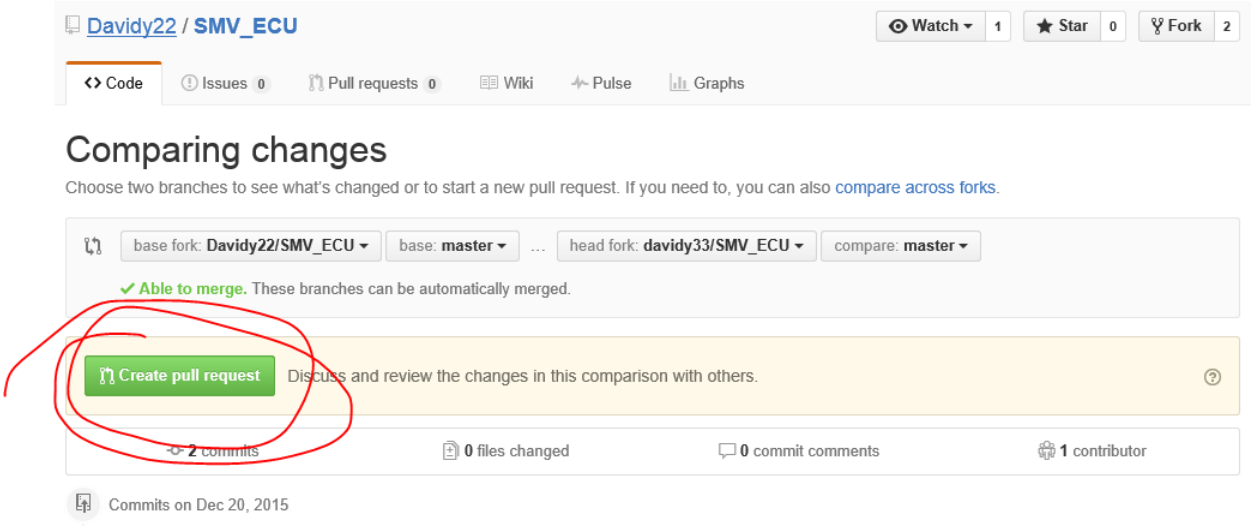




All your changes remain confined within your own repository initially. To send your changes to the main repository, click the “New pull request” button.

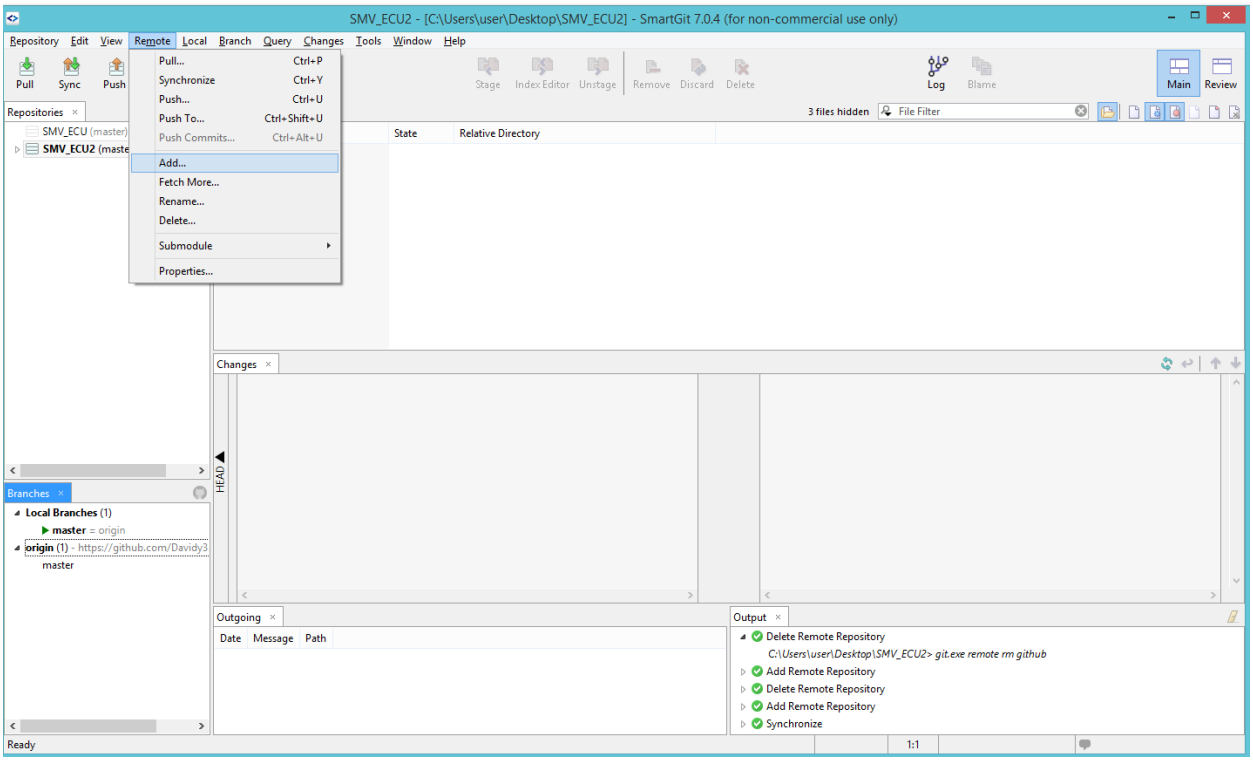


This will take you to a page where you can check to make sure you're sending the changes to the right place. It will default to sending your changes to the repository you forked from. Click the "Create pull request" button to request that the owner of the repository include your changes in their version of the code.

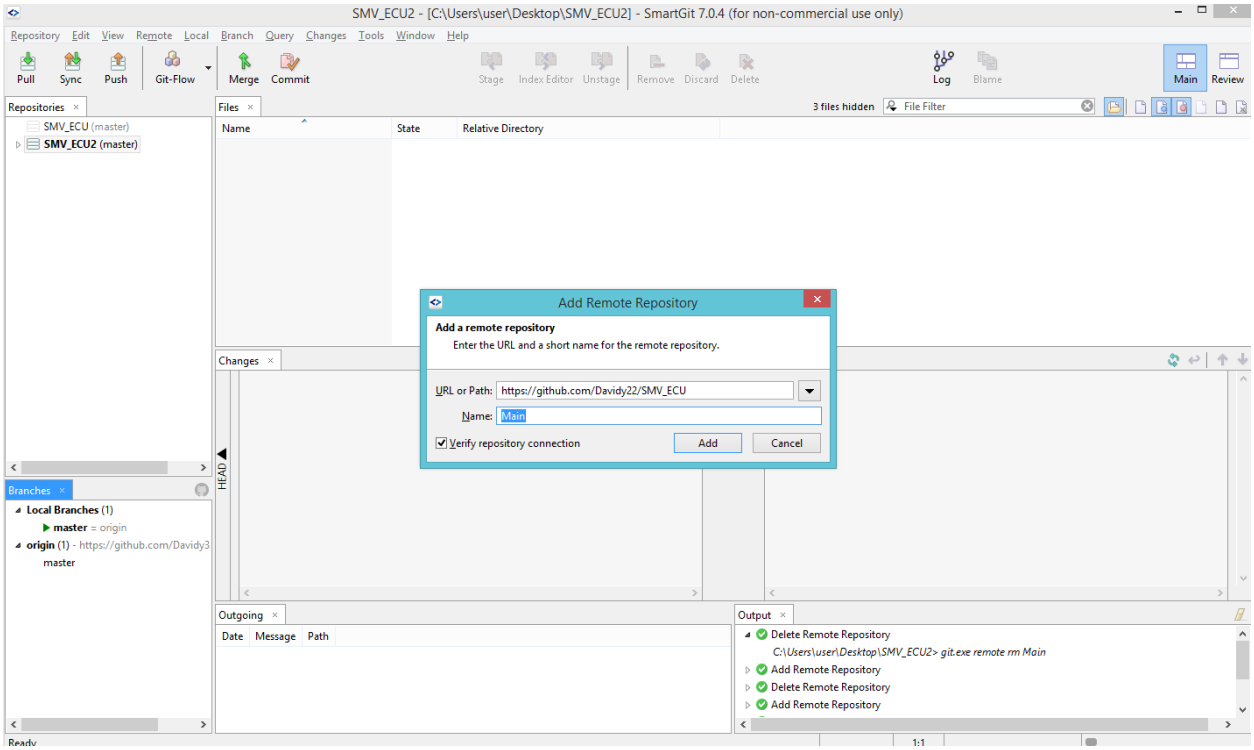


When you do this, a project lead will look at your changes, make sure you're not some guy from USC trying to sabotage our code, read through to make sure you were sober when you wrote the code and decide whether or not to fold your changes into the main repository.

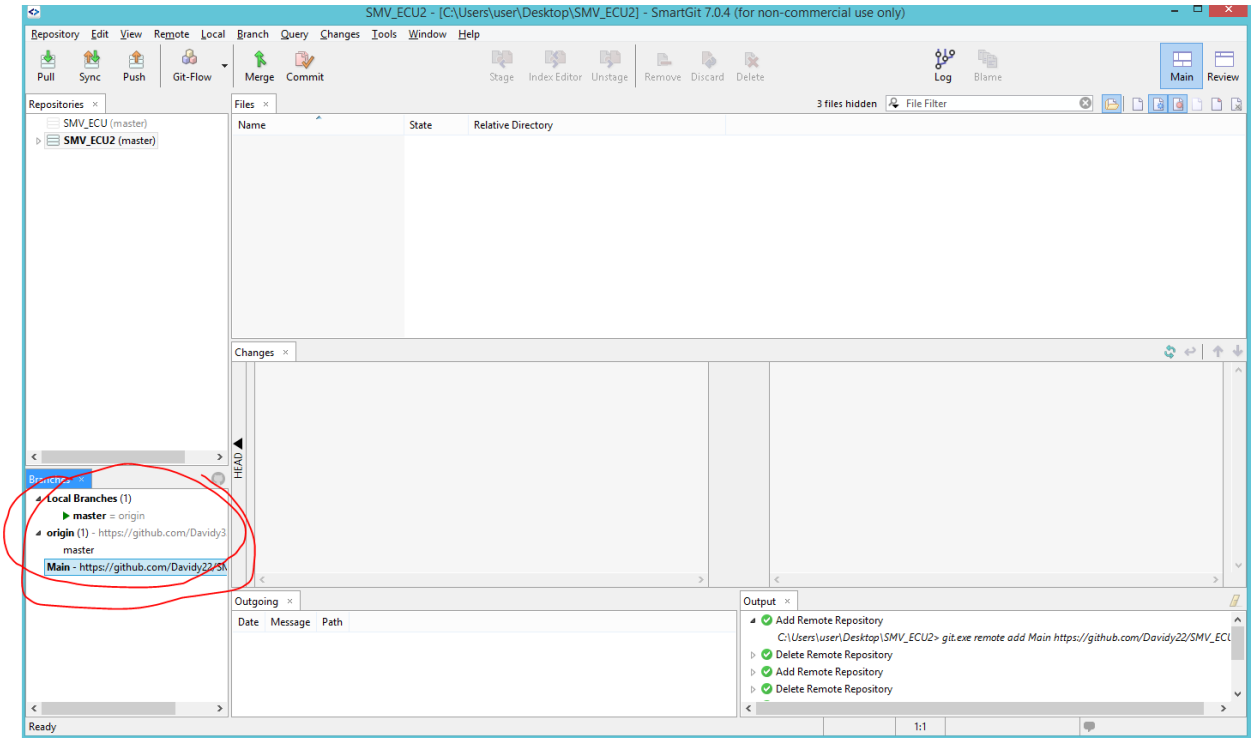
If someone else makes changes to the main repository, you need to keep up with them. Before you start coding, sync up your local copy of the code with the main repository's changes. First, you need to tell git what repository you want to get changes from.



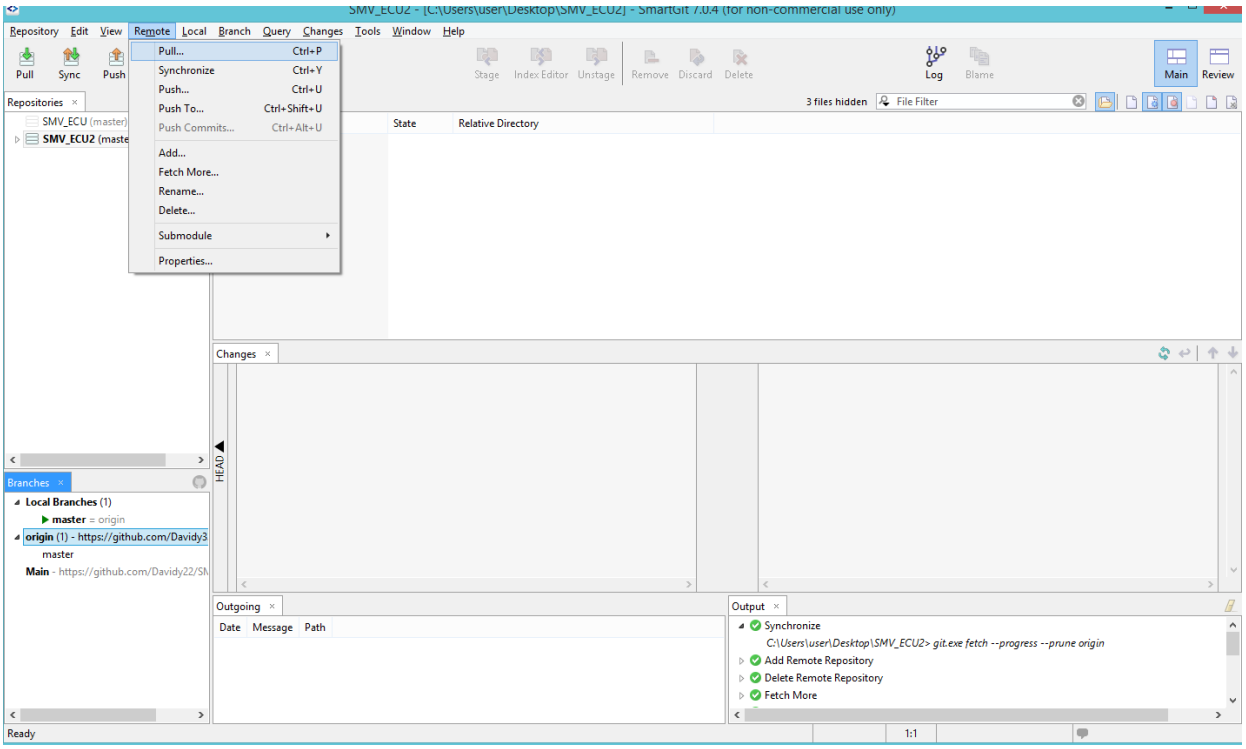
Enter the URL of the main repository and give it a memorable name.



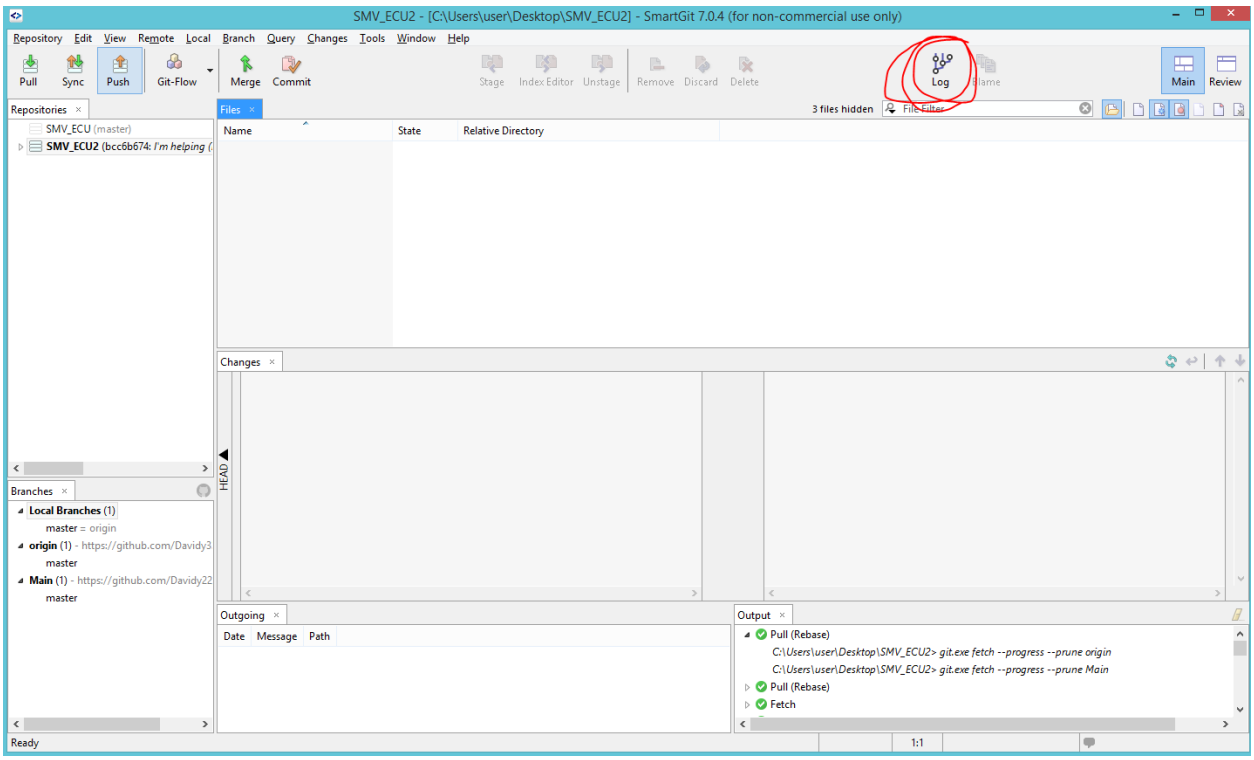
Git now has a link to the main repository.

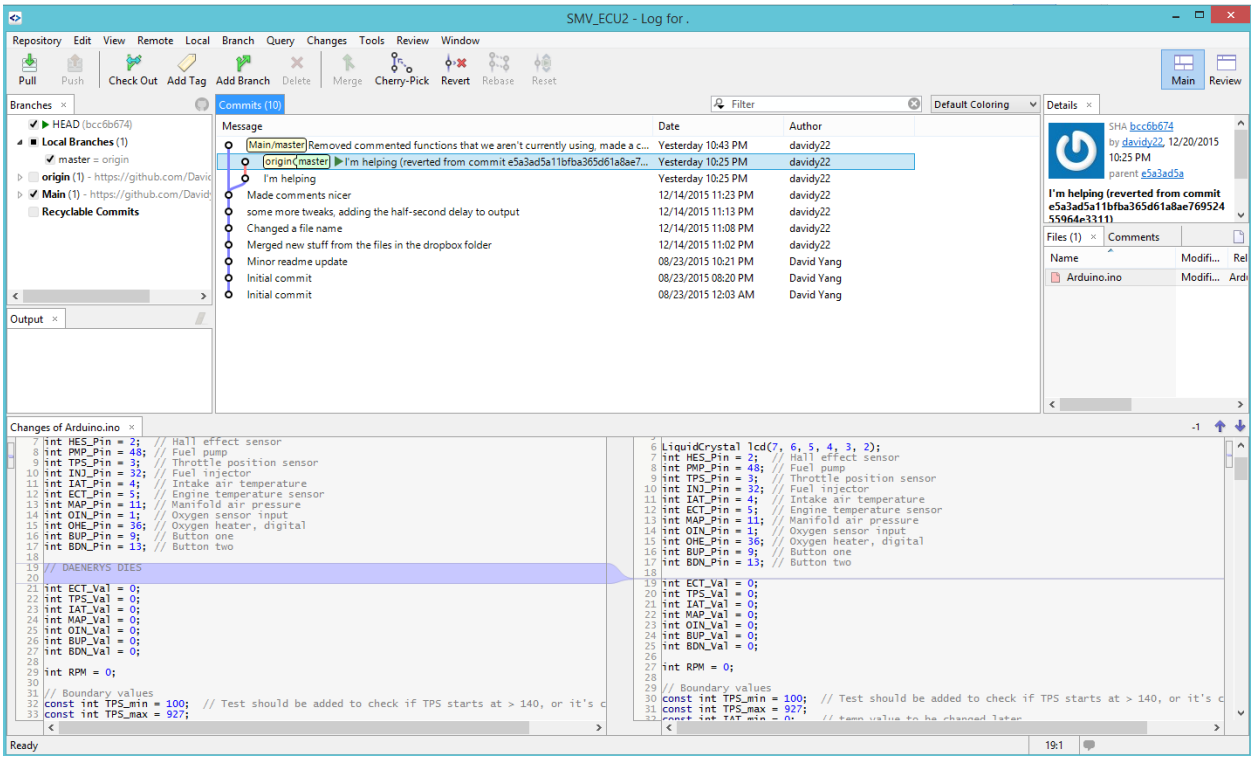


Click on the main repository's link to select it, then grab all the commits that have been saved there that aren't in your copy of the code with the pull function.

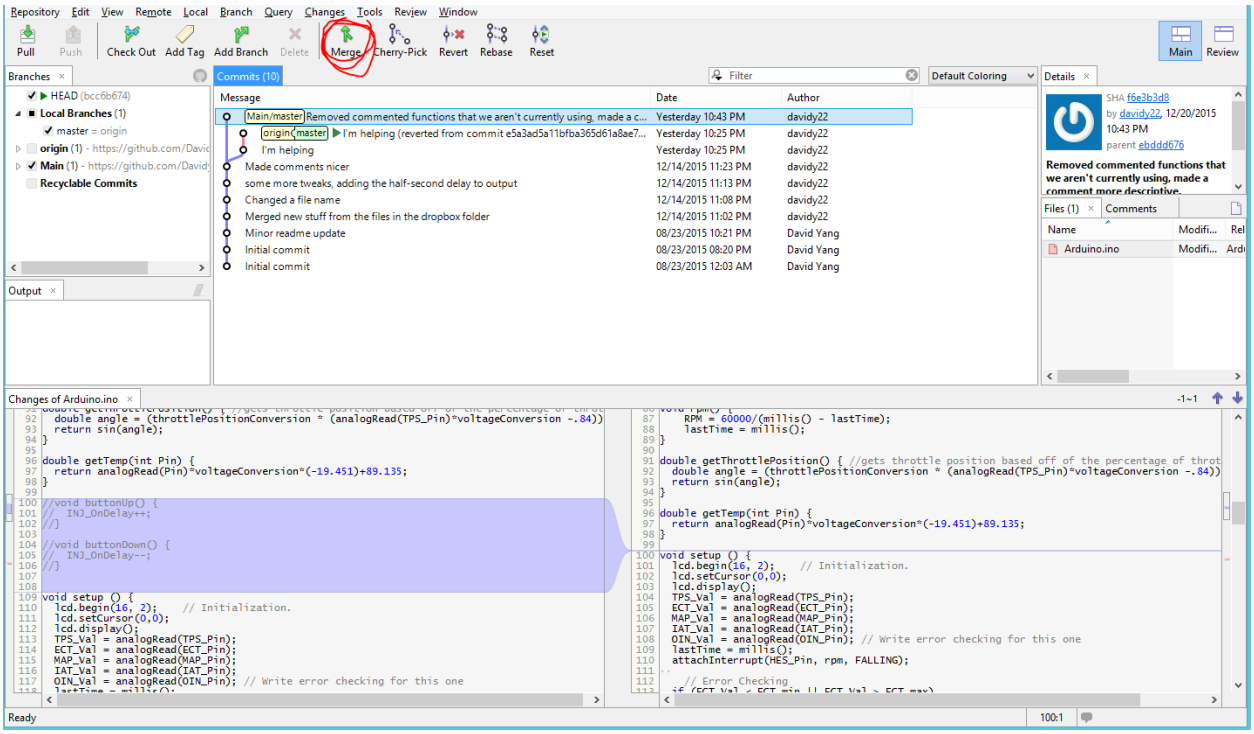


You should now have all the foreign commits that were made in the branch you pulled from. These commits all contain the changes that were made in them, but none of those changes have actually been applied to your code yet. We can look at the changes we've downloaded in more detail in the log.

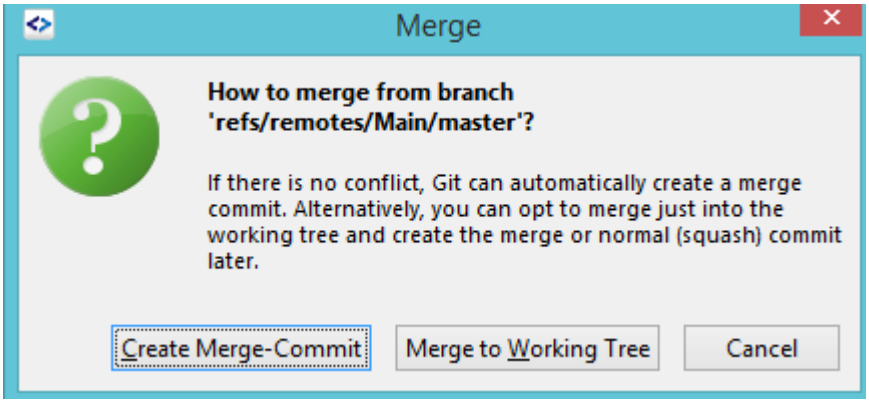




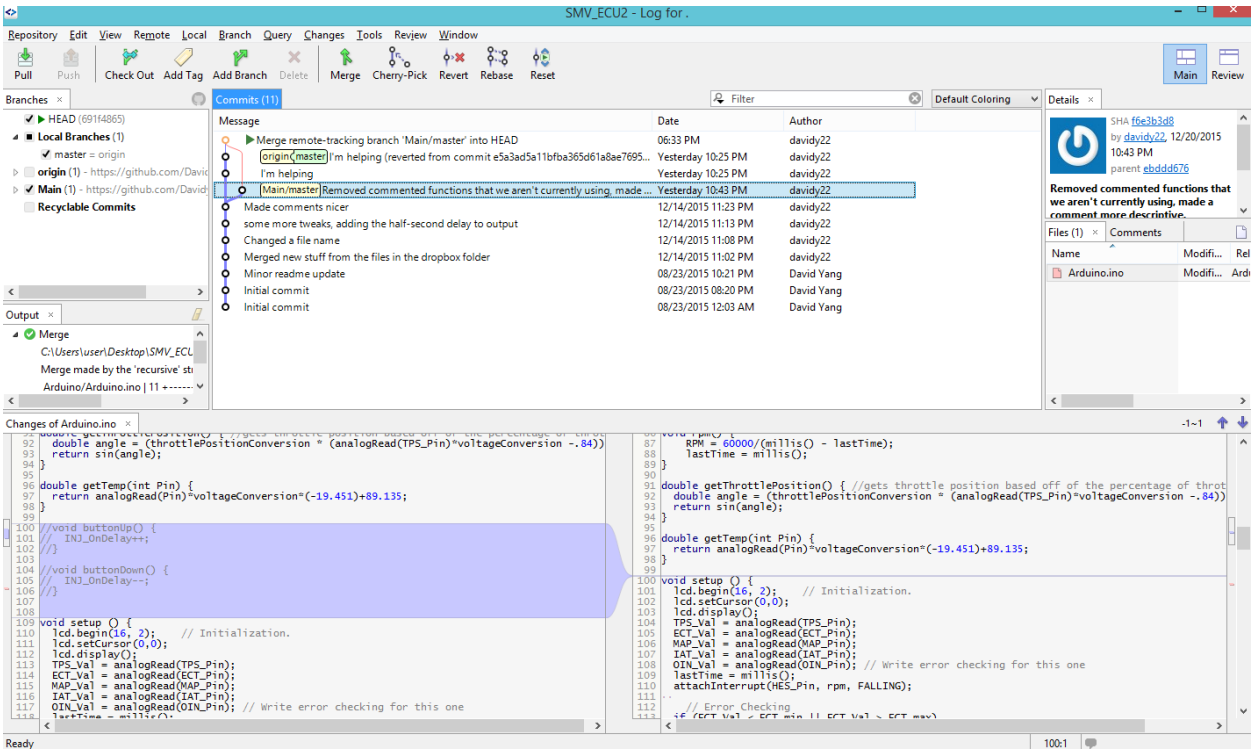
This is the log of all changes made in each repository. The yellow tags are the names of each repository; origin is the default name for our own repository, and Main/master is the repository that we just added and pulled a change from. Notice how the two repositories share changes up to “Made comments nicer,” before deviating. The branch that our origin goes down contains our vandalism and the revert of that vandalism, and main has a change that was made after we made a copy of it, that we want to merge into our own code. To merge these changes, click on a commit that’s in the main branch and click “Merge.”



Git will ask you if you want to merge the changes automatically and make a commit out of it, or if you just want to make the changes and commit the changes later. You'll usually want the first option.



After you select "Create Merge-Commit," your log should something like this:



Notice how the two branches are now joined again. We're nice and up-to-date again.

This should be all you need for your regular day-to-day usage of git. If you want to learn more, there's a video of the creator of Git describing how it works and why it's good. There are also some nice simple videos made by the Git project on getting started with Git.

Creator's pitch: https://www.youtube.com/watch?v=4XpnKHJAok8

Simple videos: https://git-scm.com/videos