

A brief introduction to GitHub, Git and Git Desktop

So, as we mentioned in the first lab, all of the code, projects, and lab sheets for the Embedded Computer Networks module are available on GitHub at:

<https://github.com/al3xsh/embedded-computer-networks>

I have chosen to do this for a few reasons:

1. It is the easiest way to ensure that you have access to any changes I make to the code, projects, or libraries
2. Being able to use Git will enhance your CV and is a vital skill for any software / embedded development role
3. It is a convenient way to provide external access to the code base

This document is a brief guide to getting access to the code, using Git for version control, and keeping your copy of the embedded computer networks repository up-to-date with my original one.

Getting the code

1. Create an account on GitHub / sign in to the account you have already created. Go to my version of the embedded-computer-networks repository (see link above), and click the “Fork” button (as shown in Figure 1). This will create a copy of my version of the embedded-computer-networks repository in your GitHub account.

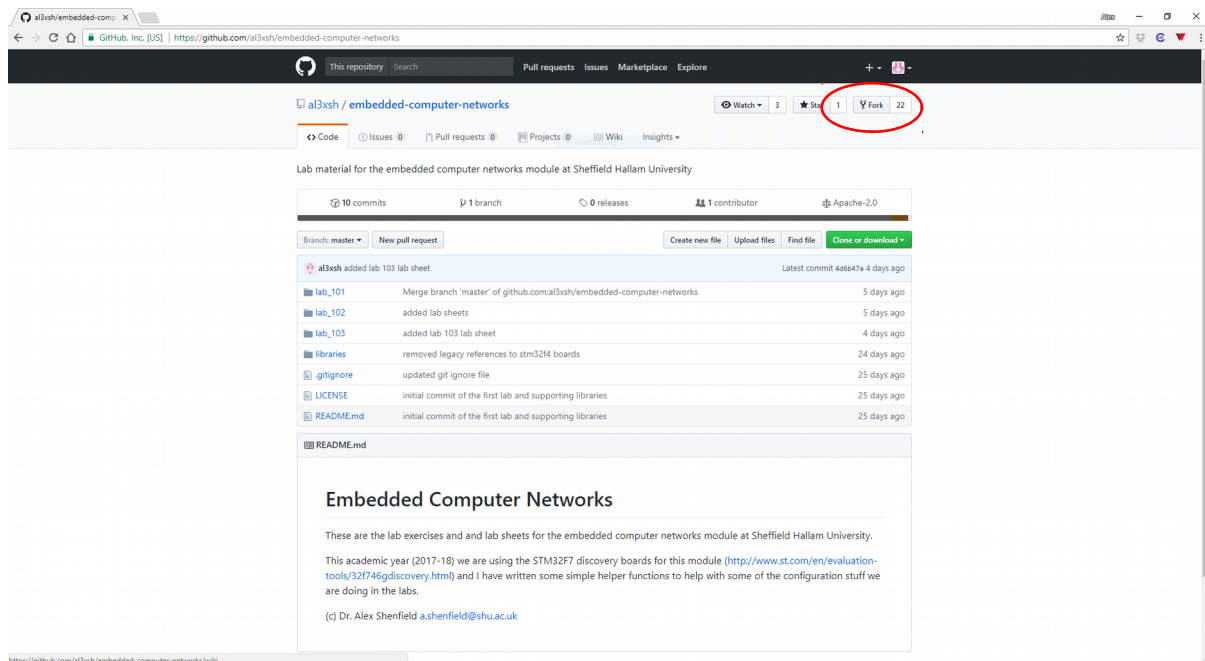


Figure 1 – Fork the repository

2. Then download and install git desktop from <https://desktop.github.com/> (or use a command line client).
3. Sign in to your GitHub account (as in Figure 2).

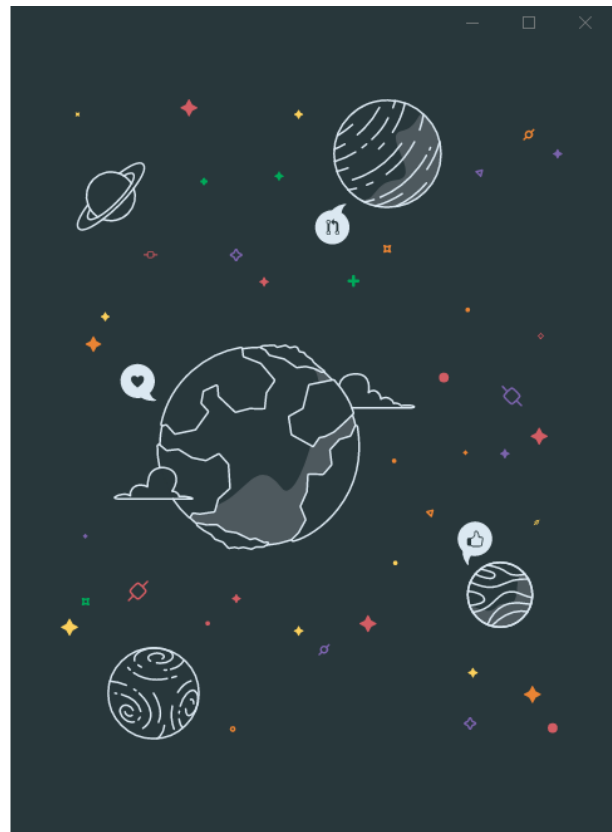
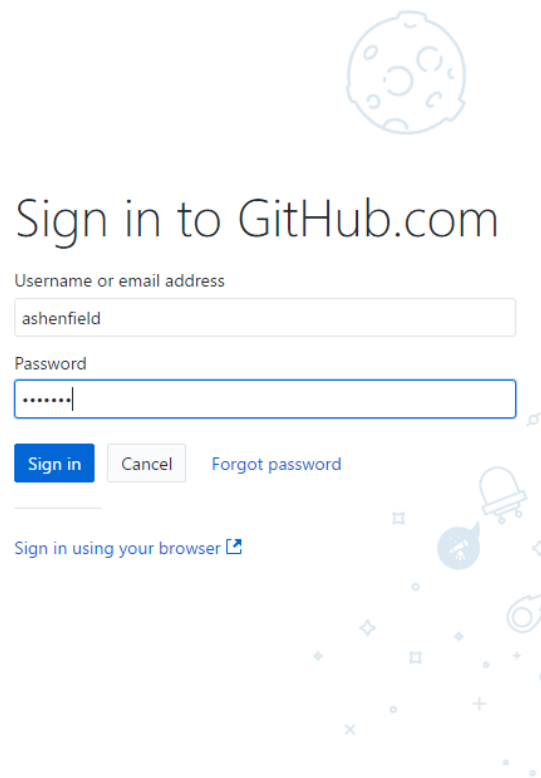


Figure 2 – Signing into your GitHub account using Git Desktop

4. From the “clone a repository menu on the right hand side” (see Figure 3) you will be taken to a menu where you should be able to see the repository you forked.

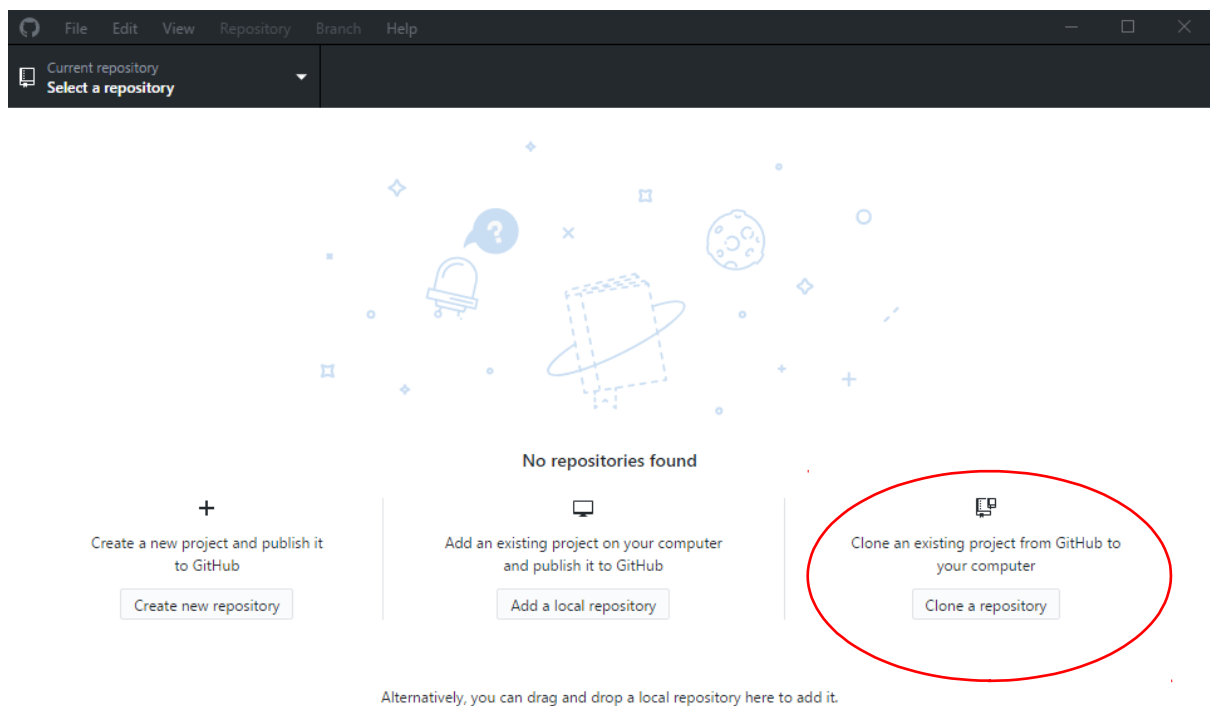


Figure 3 – Cloning a repository

5. Select the repository and choose a local path to clone to (see Figure 4). I would suggest that (unless you have a decent amount of space available on your home drive) you either use D: or a USB 3.0 memory stick plugged into the new USB 3.0 ports on the lab machines.

If you use D:, make sure to back up your work – pushing your changes to GitHub will do this (but they will be publicly available) – and delete the working copy on D:

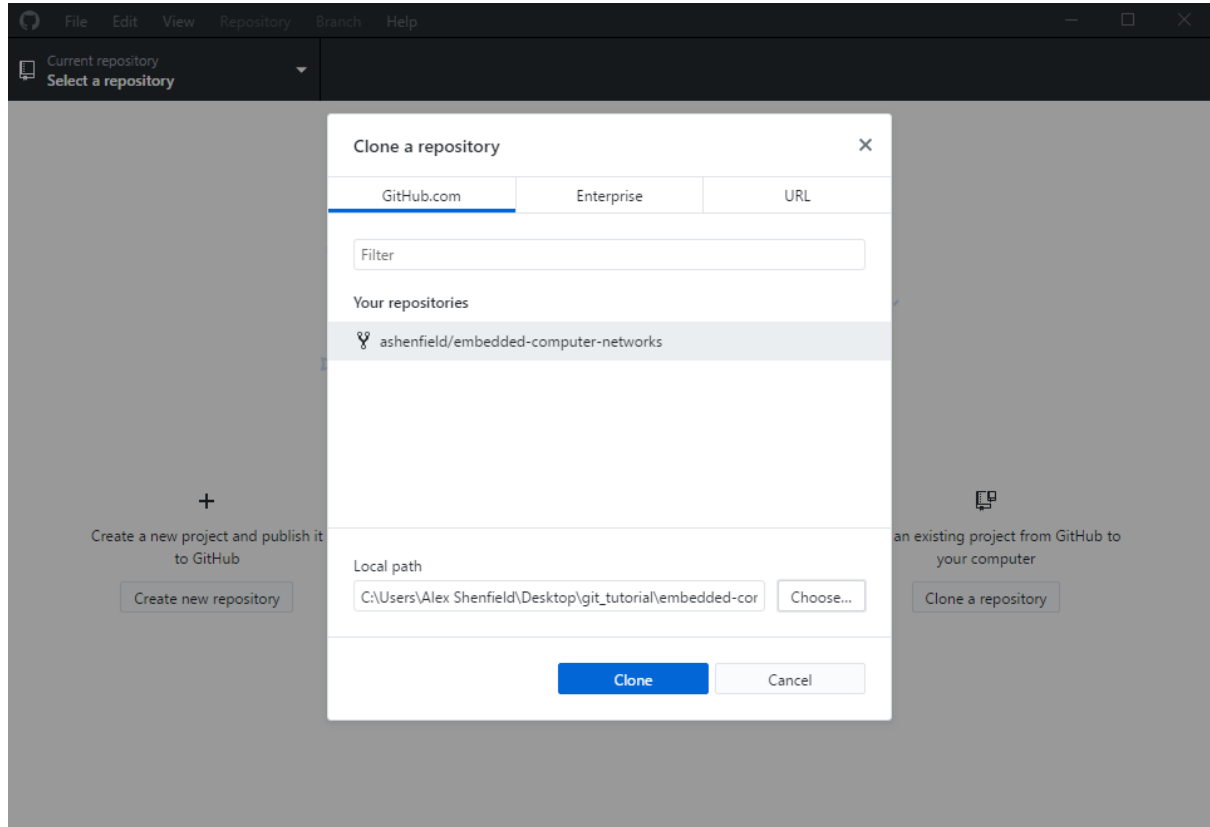


Figure 4 – Cloning the forked repository

6. You will see a message that looks something like Figure 5.

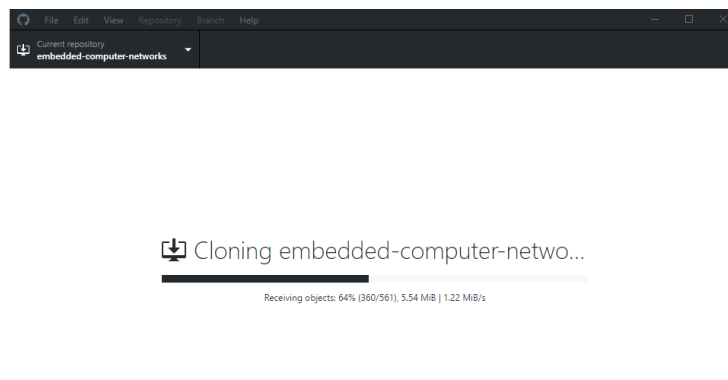


Figure 5 – Cloning in progress

- Click “open this repository in windows explorer” (see Figure 6) to get to the code for the embedded computer networks labs (see Figure 7).

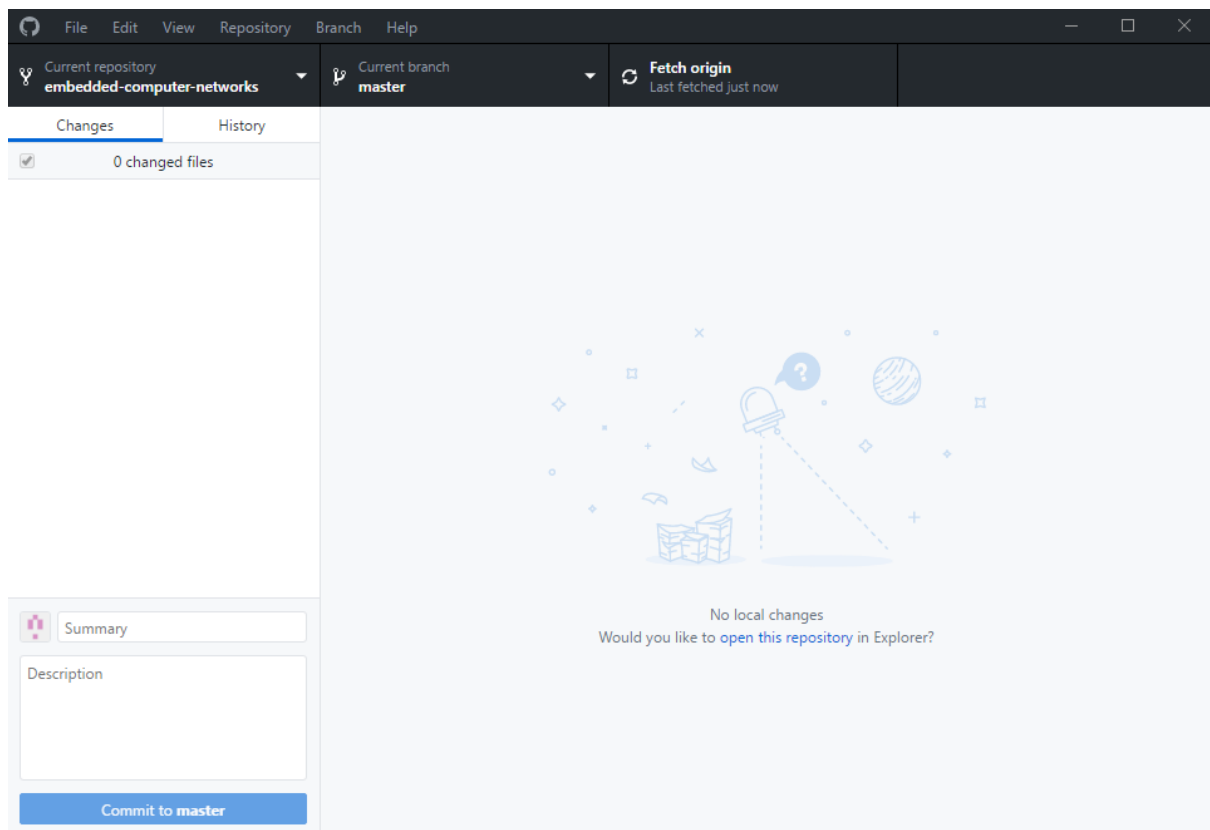


Figure 6 – Open the repository

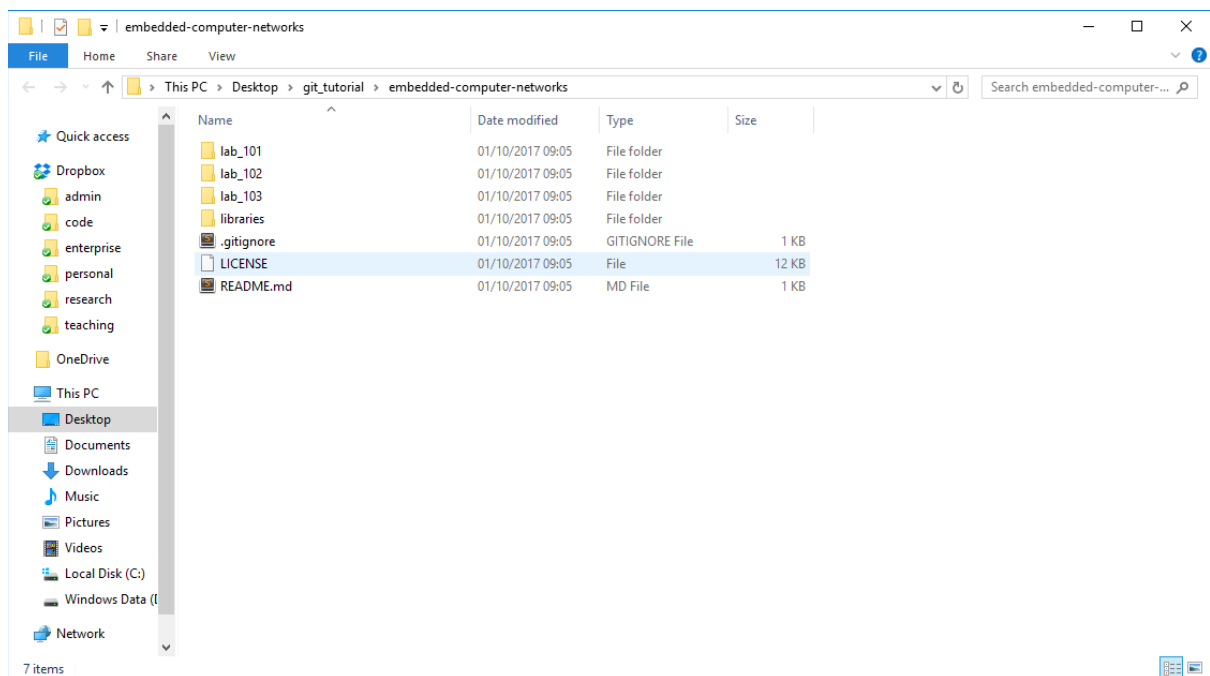


Figure 7 – The root directory for the embedded computer networks code

- Now you can work on the labs!

Making and synchronising changes

Once you have made and saved some changes to some of the code the Git Desktop application will show you exactly what changes have been made (see Figure 8).

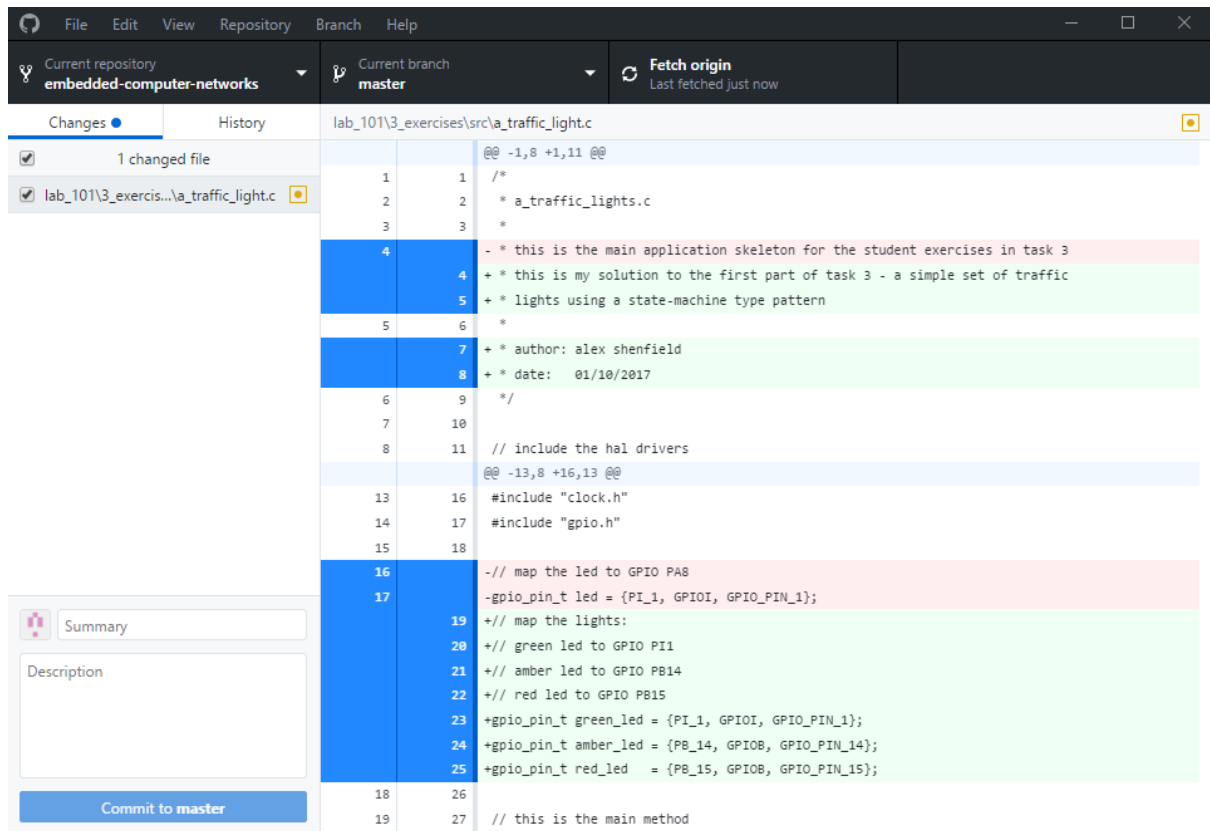


Figure 8 – Changes made to the file

Once you have tested and are happy with the changes you have made you can commit them to the local repository. Ideally, you should only commit changes you have tested and are happy with. Whilst it is possible to revert back to the time before you made a change (this is the point of version control after all), it makes a bit of a mess of your commit history ...

To commit changes you have to write a short description of what you changed (see Figure 9).

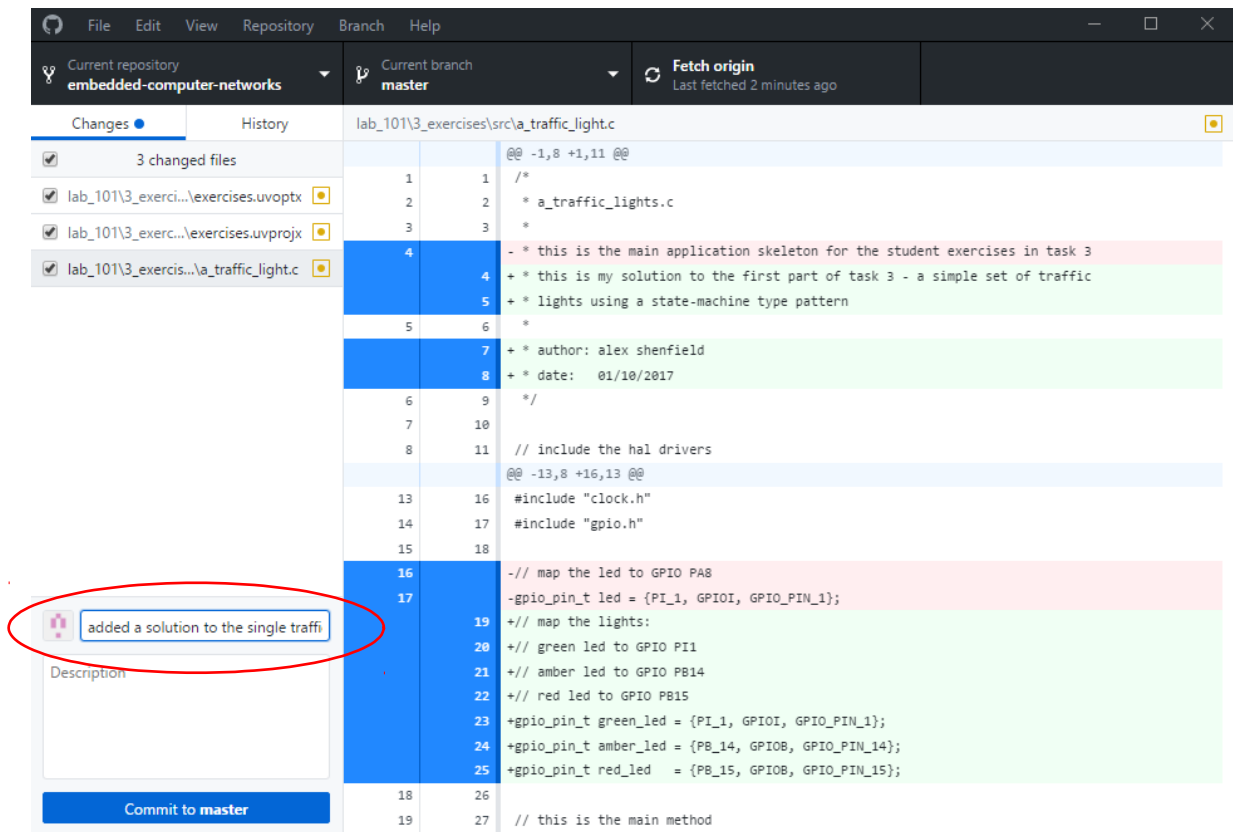


Figure 9 – You need a short commit message before you can commit changes

If you click “Commit to master” (shown bottom left in Figure 9 above), your changes will then be written to the local repository. However, you will still need to push your changes to the remote repository using the “Push origin” button into the toolbar (see Figure 10).

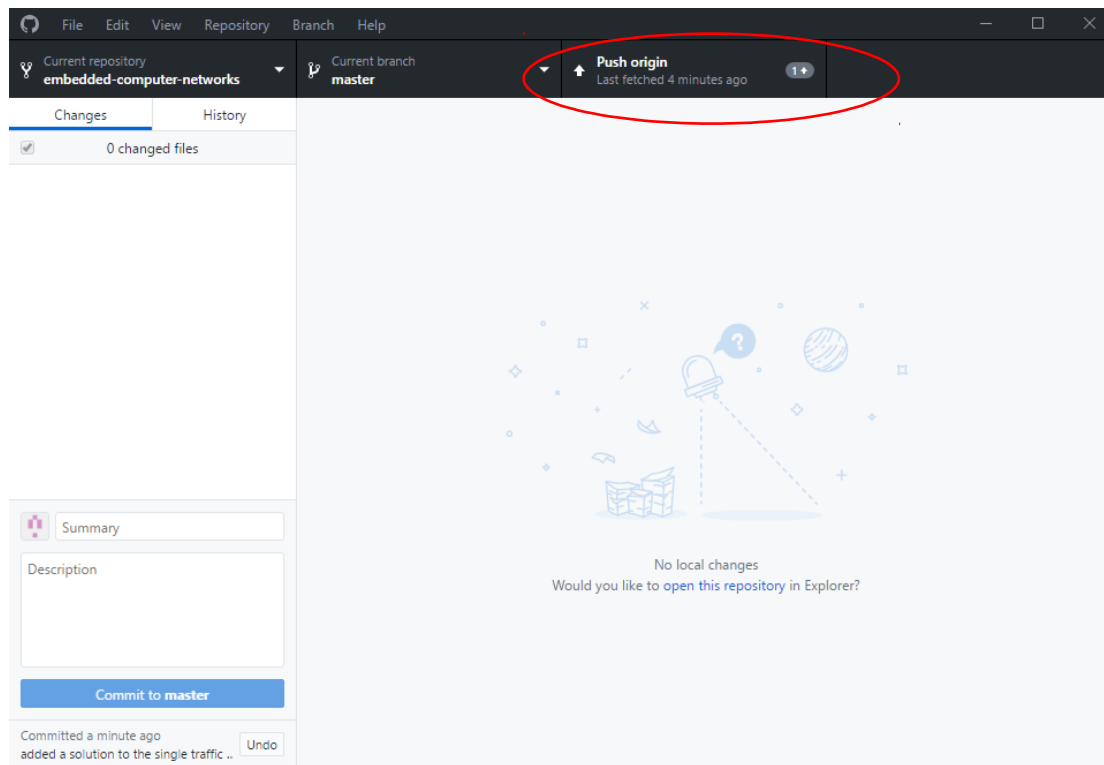


Figure 10 – Pushing changes to the remote repository

Once you have done this you should see your changes reflected on the repository page on GitHub (see Figure 11).

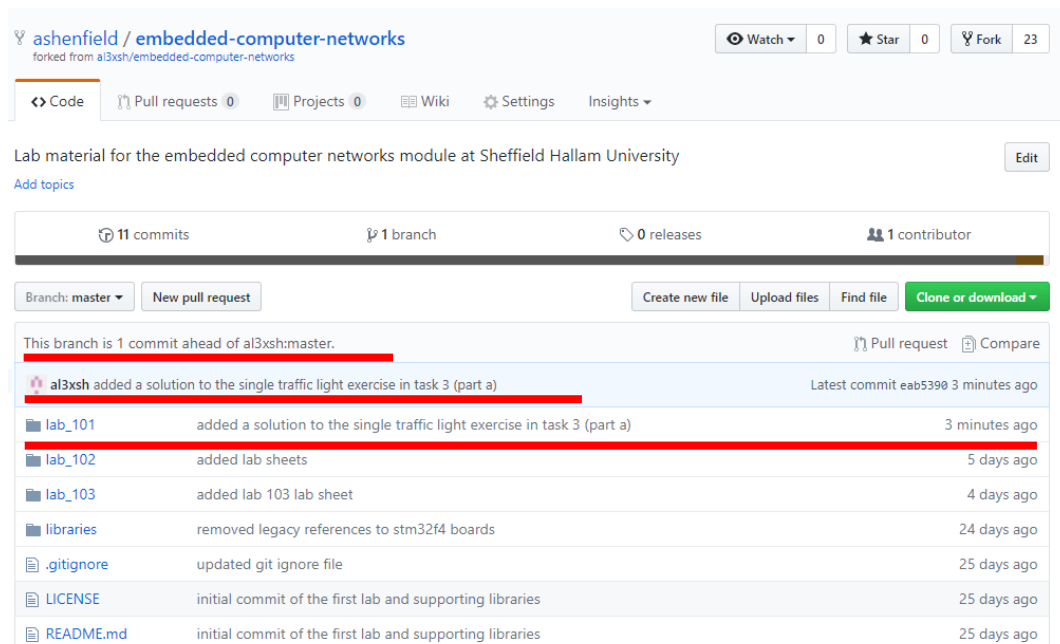


Figure 11 – Details of the pushed commit

Pulling changes from my original repository

As you can see from Figure 12, when I push updates to the original repository, GitHub will let you know that there is a commit in my repository that is not present in your fork.

The screenshot shows the GitHub interface for a repository named 'embedded-computer-networks' by user 'ashenfield'. The repository is a fork of 'al3xsh/embedded-computer-networks'. The page shows 11 commits, 1 branch, 0 releases, and 1 contributor. The current branch is 'master'. A message indicates the fork is '1 commit ahead, 1 commit behind al3xsh:master'. A list of commits is shown, with the latest commit 'al3xsh added a solution to the single traffic light exercise in task 3 (part a)' highlighted. The commit list includes files like 'lab_101', 'lab_102', 'lab_103', 'libraries', '.gitignore', 'LICENSE', and 'README.md'.

Commit	Message	Time
al3xsh	added a solution to the single traffic light exercise in task 3 (part a)	12 minutes ago
	added lab sheets	5 days ago
	added lab 103 lab sheet	4 days ago
	removed legacy references to stm32f4 boards	24 days ago
	updated git ignore file	25 days ago
	initial commit of the first lab and supporting libraries	25 days ago
	initial commit of the first lab and supporting libraries	25 days ago

Figure 12 – The forked repository is one commit behind my original repository

Before we make any changes, it is a good idea to take a backup of the entire local code base you are working on (either to USB or copy the entire set of labs to the desktop) in case things go wrong. It is also essential to make sure your remote repository and local repositories are up-to-date by pulling or pushing any changes.

Unfortunately there is no easy way to pull these changes from my original repository in the Git Desktop client ☹ .

However, it is very straightforward from the command line. This is accessible from the Repository menu (shown in Figure 13).

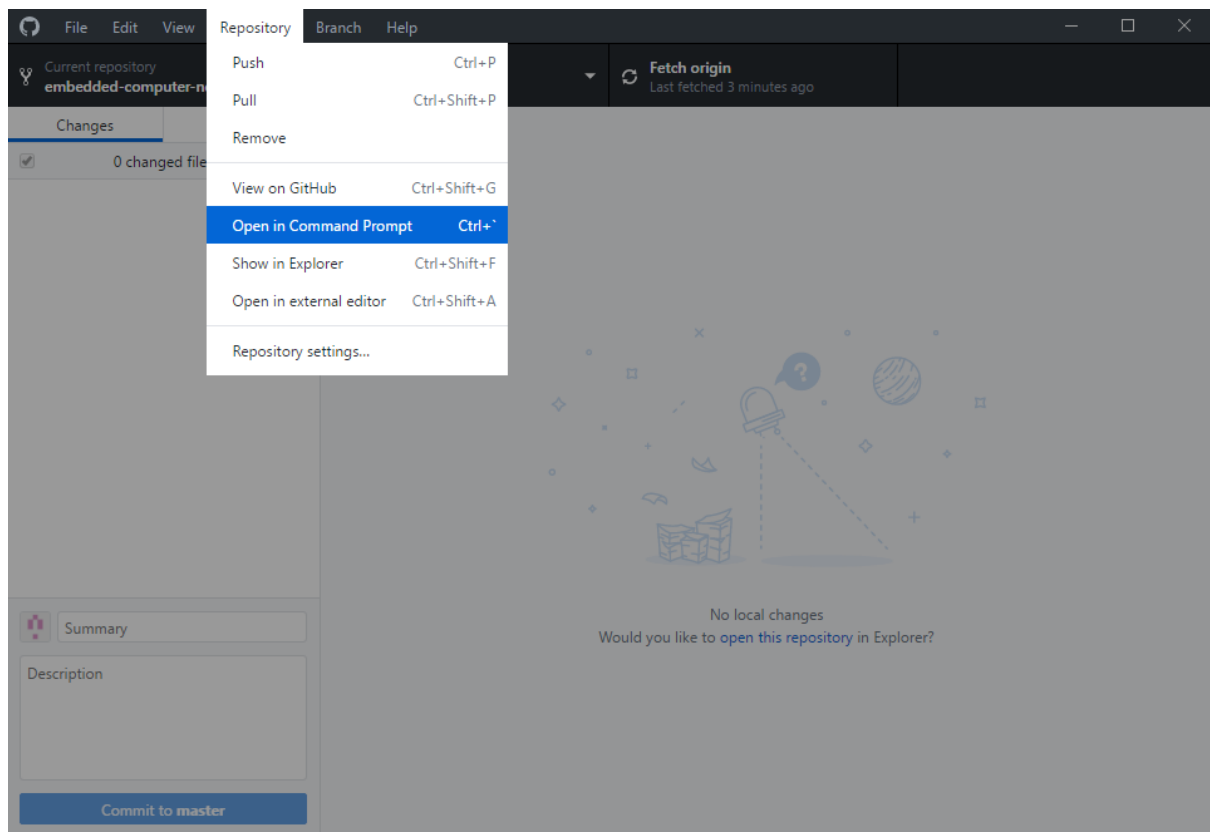


Figure 13 – Open repository in command window

This will give you a command window (see Figure 14).

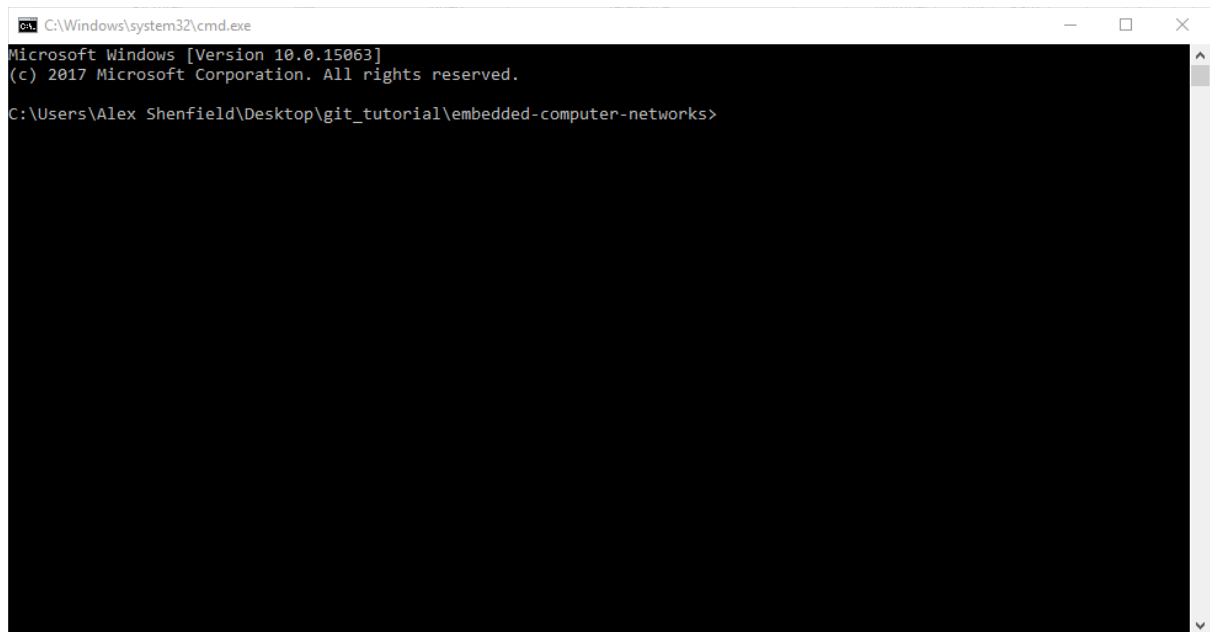
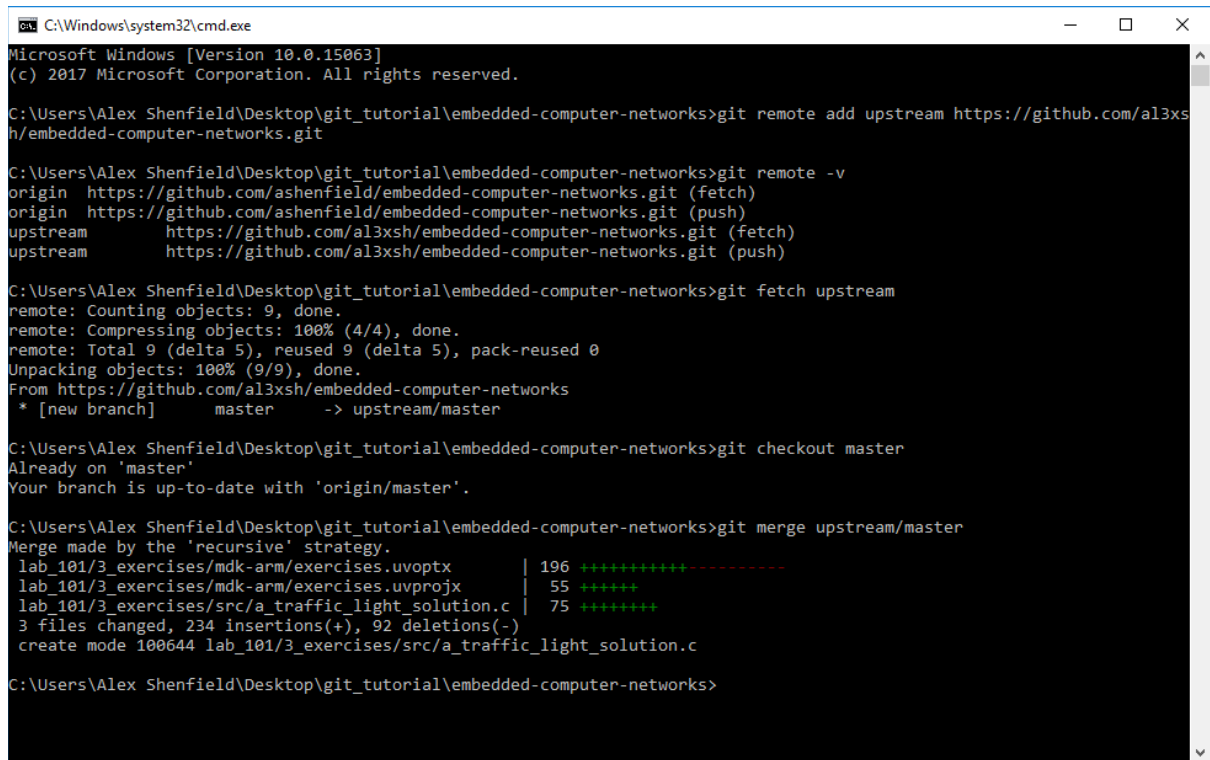


Figure 14 – Git command window

Then we need to type the following commands¹ (see Figure 15 for the output):

```
>> git remote add upstream https://github.com/al3xsh/embedded-computer-  
networks.git  
>> git remote -v  
>> git fetch upstream  
>> git checkout master  
>> git merge upstream/master
```



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Alex Shenfield\Desktop\git_tutorial\embedded-computer-networks>git remote add upstream https://github.com/al3xsh/embedded-computer-networks.git

C:\Users\Alex Shenfield\Desktop\git_tutorial\embedded-computer-networks>git remote -v
origin  https://github.com/ashenfield/embedded-computer-networks.git (fetch)
origin  https://github.com/ashenfield/embedded-computer-networks.git (push)
upstream https://github.com/al3xsh/embedded-computer-networks.git (fetch)
upstream https://github.com/al3xsh/embedded-computer-networks.git (push)

C:\Users\Alex Shenfield\Desktop\git_tutorial\embedded-computer-networks>git fetch upstream
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 9 (delta 5), reused 9 (delta 5), pack-reused 0
Unpacking objects: 100% (9/9), done.
From https://github.com/al3xsh/embedded-computer-networks
 * [new branch]      master      -> upstream/master

C:\Users\Alex Shenfield\Desktop\git_tutorial\embedded-computer-networks>git checkout master
Already on 'master'
Your branch is up-to-date with 'origin/master'.

C:\Users\Alex Shenfield\Desktop\git_tutorial\embedded-computer-networks>git merge upstream/master
Merge made by the 'recursive' strategy.
 lab_101/3_exercises/mdk-arm/exercises.uvoptx | 196 ++++++++
 lab_101/3_exercises/mdk-arm/exercises.uvprojx | 55 +++++
 lab_101/3_exercises/src/a_traffic_light_solution.c | 75 +++++
 3 files changed, 234 insertions(+), 92 deletions(-)
 create mode 100644 lab_101/3_exercises/src/a_traffic_light_solution.c

C:\Users\Alex Shenfield\Desktop\git_tutorial\embedded-computer-networks>
```

Figure 15 – Command window output

What this does is basically to pull all the changes in the original repository into a separate area (called a branch) and then merge them into your local repository. Note, you will still need to push these changes to GitHub if you want them to be stored remotely – see the “Push origin” button in Figure 10.

¹ For those that are interested, what these commands actually do are:

- i. Adds my original repository as the upstream branch
- ii. Lists the remote repositories attached to this local repository
- iii. Fetches my original repository
- iv. Switches back to the main working area (if you are not already there)
- v. Combines the changes made in your local repository with the changes made in my original repository

If you then go to your local copy of the repository in windows explorer, you can see the files that have been updated (see Figure 16 – where there is now a file “a_traffic_light_solution.c”).

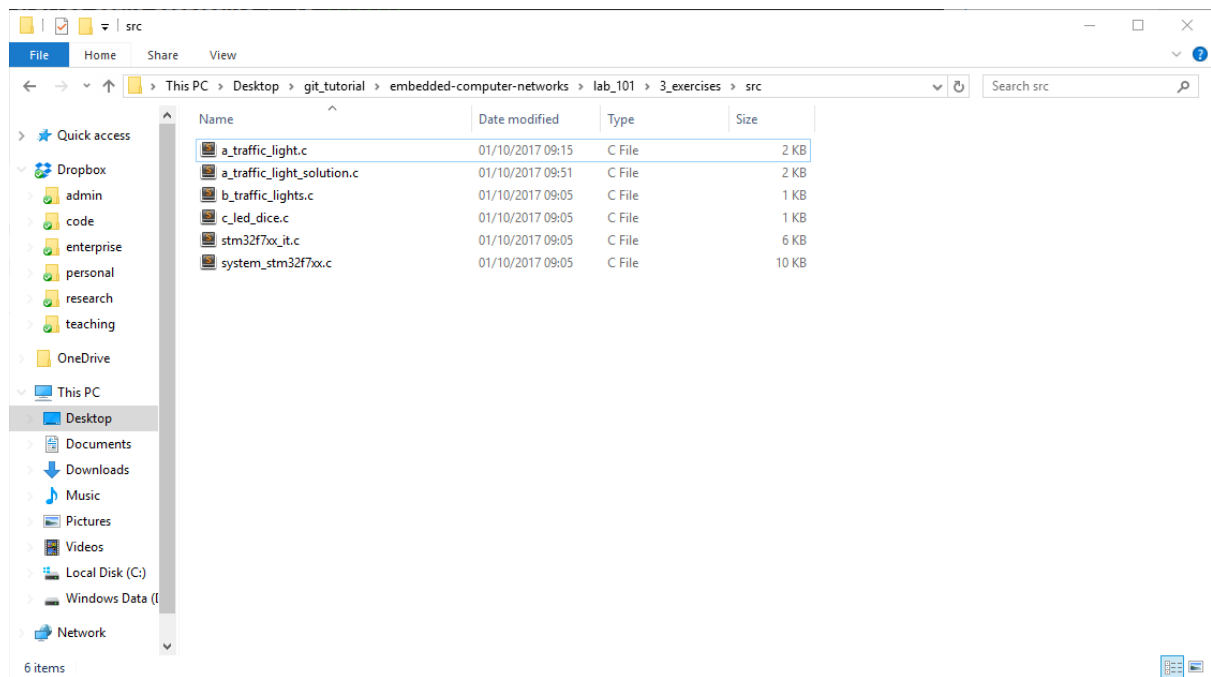


Figure 16 – Merged changes into the repository