ORGA SOFTWARE BATM

E.Chanrion 27/10/2017

1 Introduction

This document explains how the development of shared-software is done in the lab. For example, we have in common a complete framework to run an experiment from instructions to analysis GUI. In order to continue to enjoy the benefits from those joint efforts, we explain here a proper set of rules and good habits.

Our goal is to enhance this "collaborative development", so than:

- no one will ever lose time re-coding something already done by someone else in the lab,
- we keep track of each development and improvement of the different software
- we share the whole process of certification to ensure that the software runs without bugs

Notably, all the rules set in the following are good habits to keep even when you are developing something just for yourself. First, we introduce GitHub which is the main tool used to manage the development process. Then, we explain how things are done in the lab, and rules to follow.

2 GitHub

(Most part of this section is copy-past from https://www.howtogeek.com/180167/)

2.1 The "Git" in GitHub

Git is a version control system, but what does that mean? When developers create something (an app, for example), they make constant changes to the code, releasing new versions up to and after the first official (non-beta) release.

Version control systems keep these revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute. Similarly, people who have nothing to do with the development of a project can still download the files and use them.

Git is the preferred version control system of most developers, since it has multiple advantages over the other systems available. It stores file changes more efficiently and ensures file integrity better.

2.2 The "Hub" in GitHub

You do not want to use "Git" by itself. It's too geeky: you run it only through command-line tools; there is no safety to ensure that you are not blowing the whole thing, ... That's why you use GitHub!

GitHub gives you all the safety that you need through either a web interface or a cool desktop application. Both are easy to use, as we will see. Plus, GitHub gives you tool to manage a shared-project: "Who can participate? How?" But first, you need to know the vocabulary used in this collaborative-world:

Repository

A repository (usually abbreviated to “repo”) is a location where all the files for a particular project are stored. Each project has its own repo, and you can access it with a unique URL.

Forking a Repo

“Forking” is when you create a new project based off of another project that already exists. This is an amazing feature that vastly encourages the further development of programs and other projects. If you find a project on GitHub that you’d like to contribute to, you can fork the repo, make the changes you’d like, and release the revised project as a new repo. If the original repository that you forked to create your new project gets updated, you can easily add those updates to your current fork.

Pull Requests

You’ve forked a repository, made a great revision to the project, and want it to be recognized by the original developers—maybe even included in the official project/repository. You can do so by creating a pull request. The authors of the original repository can see your work, and then choose whether or not to accept it into the official project. Whenever you issue a pull request, GitHub provides a perfect medium for you and the main project’s maintainer to communicate.

Changelogs

When multiple people collaborate on a project, it’s hard to keep track revisions—who changed what, when, and where those files are stored. GitHub takes care of this problem by keeping track of all the changes that have been pushed to the repository.

3 How things are done in the lab ?

3.1 Our GitHub framework

We set up an "Organization" called **OrgaSoftwareBatM**. This organization is not linked to a specific GitHub account and should contains all project developed among the group. Once you created a GitHub account, ask someone to send you an invitation to join OrgaSoftwareBatM.

3.2 User rules

1. Create 1 development fork per person (for example Baptiste-Dev, Everton-Dev or Sionludi-Dev) or one per main project (New-FPGA-Dev).
2. Direct commits to the master branch are disabled, only commit to your own branch. Update master branch via pull request from your personal fork, and only after proper tests and debugging.
3. Someone else should be responsible for checking the added parts (and the debugging) and accepting the pull request.
4. Updating the code while running an experiment is not mandatory, but if you develop something new you need to either start from the last "master" version or make sure it will be compatible.
5. Please comment your commits.
6. The gitignore file prevents .pyc files from being committed. Please make sure not to upload useless (or setup-dependant) files.
7. We should keep a clear version number for both the Python and the Labview part. Ideally with a clear changelog that matches the 2.
8. The "Projects" tab is here to share your on-going and future development ideas.
9. If you modify something in your code and don't want to commit it, please at least write the changes in the "Projects" tab, instead of trusting oral transmission.
10. As soon as you start adding an instrument, please register the new number (on the "Projects" tab for example). This will make merging easier.