**Projects Protocol / Standards**

This document describes the protocol for developing projects in the biorobotics lab. It includes information on how to structure the projects on git, where to host them, how to provide documentation, what information should be included, which format to follow on the documentation, and more. **Note that this are general rules and might not apply to all projects.**

1. **Operating System**

All projects must be able to run at least under the linux PCs in the lab unless otherwise stated.

1. **Access to the labs PCs**

To use the PCs in the lab you’ll need to get a username and password from one of the current lab admins.

1. **Git usage**

Unless the project involves confidential information (such as solutions for a class project), all projects must be sourced on the lab’s github ( <https://github.com/biorobaw> ). Either if you are starting a new project or if you are picking up from someone else’s you’ll have to talk with one of the current lab admins to create either a new repository or branch and to get contributor permissions.

Usage requirement:

* Get familiar with what git is used for and its good practices.
* You must commit to your branch at least once a week to keep track of the changes/progress made.
* All commits should have relevant messages explaining the changes made.
* Do not use git for what it is not intended. Examples:
  + Do not create branches for having different versions of the same file (such as different configurations)
* Use the files “.gitignore” to tell git to ignore unwanted files such as IDE config files, generated videos, logs, etc.

1. **Project Folder Structure**

All github projects must have the following structure (an empty sample project might be found in the lab’s github). Small changes may be made to the structure, but the overall distribution and concepts must be followed.

Project folder

| TODO.txt #file describing future work

| readme.md #describes project

| Makefile #file to make project

|

+---src #folder containing source code

|

+---libs #folder containing all req libs

|

+---build #folder containing compiled code

|

+---bin #folder containing build program or lib

|

+---config #folder containing all config files

|

+---docs #folder containing all documentation

| Instructions.pdf #installation instructions pdf format

| InstallInstructions.doc #installation inst.for future editing

| UsageInstructions.pdf #instructions on how to run the program

| UsageInstructions.doc #RunInstructions editable

| Architecture.pdf #File describing the code arechitecture

| Architecture.doc #Editable Architecture file

| VideoLinks.txt #File containing the links to all videos

|

+---Reports #folder containing all reports

| week0.pdf

| week1.pdf

| …

| weekN.pdf

| FinalReport.pdf

NOTE: The Installatio, instructions, architecture documents and video links may be

1. **TODO file**

This file should contain the next steps to do, bugs which were not fixed, etc. It is required when submitting the final state of the project.

1. **readme.md**

This file contains the information read on the project’s repository. It should at least provide a description of what the project is about, its extent, and what should a new user need to know to start using the project.

1. **Makefile**

File for allowing easy command line compilation of the project. It doesn’t necessarily have to be a makefile, it can be any equivalent tool.

1. **src folder**

This is the folder where all the code should be placed. The folder substructure should group source files into functional modules and follow all conventions of the language/framework being used. Folder naming should be representative of the functionality provided by each module.

1. **libs folder**

This folder should contain all code related to libraries.

1. **build folder**

This folder should contain all the compiled files.

1. **bin folder**

This folder should contain the final output of the makefile utility such as generated libraries or executable files.

1. **config folder**

This folder should contain all configuration files for the project. Configuration files should not be part of the source code to avoid recompiling for each configuration.

1. **docs folder**

This folder should contain all the project documentation. All files that might be extended in the future should provide an editable copy.

1. **Install Instructions**

At the end of the semester, the project should include a file providing a step by step guide on how to install/compile the project. The file should include all the following:

* Summary of the extent of the file
* Index
* Project requirements / dependencies
* Tested environments: Include under which OS and version was the project tested. Also include the versions used of all dependencies and development environment (such as IDE and so on).
* Step by step instructions ideally using images.
* Expected output if successful.

1. **Usage Instructions**

At the end of the semester, the project should include a file providing instructions on how to use the generated program/output. The file should include:

* Summary of the extent of the file
* Index
* List of all functionalities implemented by the program
* Step by step instruction of how to run the program
* Step by step instructions of how to use all features provided by the program.
* Use case examples

1. **Architecture file**

At the end of the semester, the project should include a file explaining what the code does and how it is organized. The file should include:

* Summary of the extent of the file
* Index
* List / diagram of all logical modules and how do they relate to each other.
* For each module explain what the module does.
* List of all project files/classes specifying what they do.
* For each file / class provide a list and description of each function.
* Any other information that allows further development of the project.

1. **Video list file**

At the end of the semester you are **(might be?)** required to upload youtube videos showing how to install, run or use the project. All videos must be uploaded to the lab’s youtube channel. To do so:

* Decidir Protocolo: Decidir si playlist con contribución (problema de ownership) u otro mecanismo
* When recording a program, do not use a camera, instead use a screen capture utility such as OBS studio: <https://obsproject.com/>
* For each video, add a description of what it shows.

1. **Weekly Reports**

Each week at least 2 days before the weekly meetings, a report must be uploaded to git hub (see the git folder structure on section 4). The report must include all the following:

* Description of the work done during the previous week (except for week 0).
* Estimation of the time spent on each task during the previous week (except for week 0)
* Description of current work
* Description of goals for the following week
* Issues encountered if any

1. **Final Report and project presentation**

At the end of the semester, a project presentation must be performed, and an accompanying report uploaded to github. The report must be submitted at least 2 days prior to the presentation. Both the report and presentation must include:

* Description of the project goals
* Description of the project outcome.
* Description of how the outcome achieves each goal.
* Description of relevant decisions.
* List of unmet goals and explanations of why they were unmet.
* Description of the project limits.
* Future work describing how could the software be improve suggesting future lines of work.
* Provide any insight learned in the project that might be useful for future students.

During the presentation, you will also be required to do a live show of the final product of your project working (meaning that videos won’t be accepted unless otherwise stated).

1. **Quality standards for all documentation files**

The following is a checklist of quality standards for submitting documentation:

* No documents, images or diagrams can be hand written.
* All numbers are required to have adequate units
* At the start of any document, a brief description should be given stating the purpose and the extent of the document.
* When explaining how to configure something, provide numbered step by step explanations using images. On each image highlight the regions where the reader should make focus on.
* When explaining configuration files or talking about parameters, explain what the parameter control, which are the possible values, to what were they set, why, and how would you choose new values if necessary.
* Whenever possible, provide references to where can more information be found.
* If you had to learn what something was, don’t assume other people will know about it. Explain it or provide a reference.
* Whenever talking about a robot or a piece of hardware, provide details to identify it such as links to the item, IDs or pictures.
* When developing a robot, fundament its design. Explain the reasons for choosing each piece of HW, other alternatives considered, why was each component placed where it was, etc.
* Do not upload videos to github. Instead, all videos should be uploaded to youtube and links should be provided.