

LpzRobots and GoRobotos

An installation manual

Frank Hesse, Christoph Rauterberg

April 25, 2012

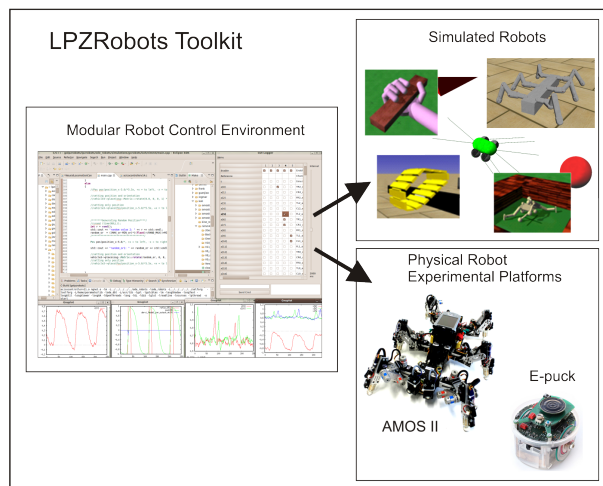


GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN



This installation manual shall first provide some general knowledge about the gorobots-Project and the architecture we use. Then this guide will explain step-by-step how to set up your Eclipse, so that you can work with the LpzRobots-Simulation and the gorobots-Project.

In the future, we would like to complete this installation manual in a way, that it will also include manuals for setting up the software needed for AMOSII (Hexapod) and other robots.



Contents

1	Introduction	3
2	Setting up Eclipse and LpzRobots	5
2.1	Running setUpGoRobots.sh	5
2.1.1	Forking a repository	5
2.2	Setting up Eclipse	5
2.2.1	Installing Tool-Kits within Eclipse	5
2.2.2	Importing the Repositories	6
2.2.3	Code-Style withing Eclipse	6
2.3	Troubleshooting	6
3	Project Structure	7
4	Working with GIT	9
4.1	Overview of GIT-commands	9
4.2	Setting up your own branch with GIT	11
4.3	Forking a Repository	12
4.4	Merging	12
4.5	How to update your Master Branch	12
5	Installing LpzRobots by Hand	15
6	FAQs	16
6.1	Encountered errors	16
6.2	Errors using setUpGoRotobots.sh	16
6.3	Problems with GIT	16
6.4	Problems within the Script	17
6.4.1	Error with git clone	17

1 Introduction

The LpzRobots-Simulation is a robot simulation programmed at the university of Leipzig. Its main features include the [ode_robots](#), which is a 3D robot simulator, that is physically correct, and the so called [selforg](#), that is a framework for controller implementation.

The important parts of the software architecture are shown in Figure: 1:

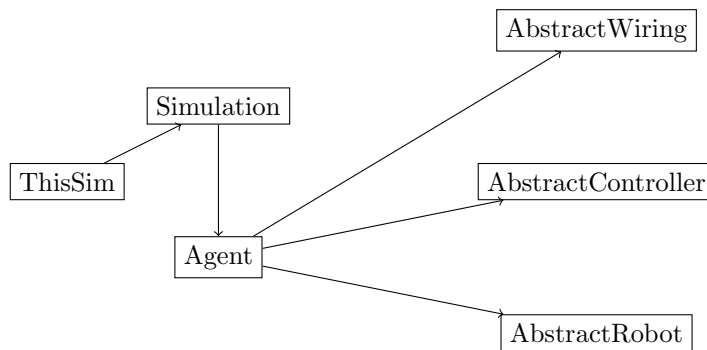


Figure 1: Software Architecture for LpzRobots and GoRobotos

[ThisSim](#) will, during a simulation, integrate all elements of this very simulation, which means controlling the environment, the robot, as well as setting initial parameters and plotting or logging data.

[Agent](#) will integrate all elements of an agent by using the shown classes, one can for example add sensory preprocessing using a child class of [AbstractWiring](#).

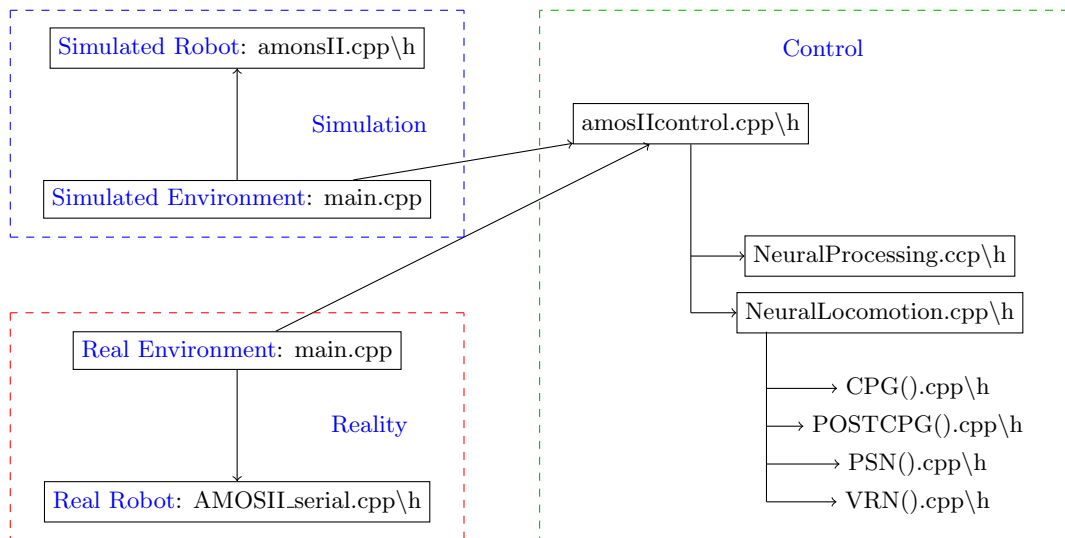


Figure 2: Need name for this picture

For working with the LpzRobots-Simulation, you will need a couple of things:

1. A (preferably up-to-date) UNIX-based Operating System (state of the art: Ubuntu 11.10 or Debian 6.0)
2. The Eclipse-Software combined with the following packages:
 - C/C++ SDK
 - The EGit-Tool-Kit
3. Access to the Assembla-Repository (contact your supervisor for access)
4. The setUpGoRobots.zip file (contact your supervisor for the file)

2 Setting up Eclipse and LpzRobots

2.1 Running setUpGoRobots.sh

From your supervisor, you will receive the .zip-File [setUpGoRobots.zip](#) - or you can find it at [gorobots/docs/install.script/](#). To begin with, you need to extract the files and the script [setUpGoRobots.sh](#) within. This script will:

1. Install the required packages on your computer
2. Include important settings to your [.bashrc](#)
3. Fetch the repositories [LpzRobots](#) and [GoRobots](#)
4. Import the project settings file
5. Compile the files

Please note, that you need to be [root](#) to install the packages via this script!. After extracting, you can run the script, by typing [./setUpGoRobots.sh](#) in the corresponding directory.

2.1.1 Forking a repository

The script will ask you for the URL of your forked repository. Please read about how to fork a network with Assembla in Section 4.3.

2.2 Setting up Eclipse

2.2.1 Installing Tool-Kits within Eclipse

Before importing the repositories, you need two tool-kits for your Eclipse:

1. To install a tool-kit, go to [Help](#)→[Install Software](#)
2. The first tool-kit you need is the C++ Development Kit. Work with the following link:
<http://download.eclipse.org/tools/cdt/releases/indigo>
and install the so called [CDT](#)-Tool-kits.
3. The second tool-kit is EGIT for accessing GIT within Eclipse. You can download it working with this link:
<http://download.eclipse.org/egit/updates>. Whilst doing so, you have to [check](#) the box for [Eclipse Git Team Provider](#) and [uncheck](#) the box for [EGit Mylyn](#).

2.2.2 Importing the Repositories

Once you have finished running the script and have installed the tool-kits, you are ready to import the repositories into your Eclipse.

To do so, first switch into the [Git Repositories - View](#) within Eclipse. You can do this by clicking: [Window](#)→[Show View](#)→[Other](#)→[Git](#)→[Git-Repositories](#) and hit [OK](#). In this view, you can choose [Add an existing local GIT repository](#). The script will have placed the files at [/home/yourlogin/workspace](#).

2.2.3 Code-Style withing Eclipse

To adapt the Code-Style, go [Window](#)→[Preferences](#)→[C/C++](#)→[Code-Style](#)→[Import](#). Now you choose the file: `workspace/lpzrobots/codeStyleEclipse.xml` and hit [apply](#).

2.3 Troubleshooting

If you encounter any problems while using the script, please contact Frank or me (c.rauterberg@gmx.de) and send us error output, solutions you found, etc. if possible. Find the install-by-hand-instructions at the end of this manual.

3 Project Structure

We use, as already mentioned, two GIT repositories, LpzRobots and GoRobots. A rough overview of the structure is given below in Figure 4:

Later, the controllers for each robot will be implemented within [GoRobots](#), accessing the robots, which are located in [LpzRobots](#). The folder [DEMO](#) will later contain demos of the different robots. Another visualisation of the two repositories and where which file belongs is given in Figure 3.

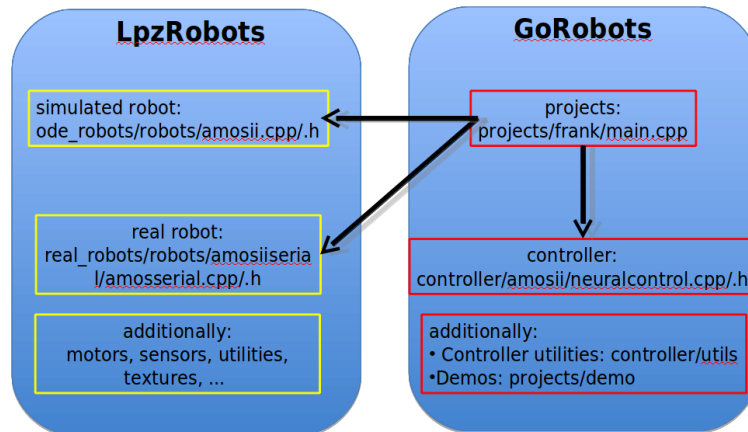


Figure 3: Structure of the two repositories, LpzRobots and GoRobots

You will later work with your own copy of the two repositories.

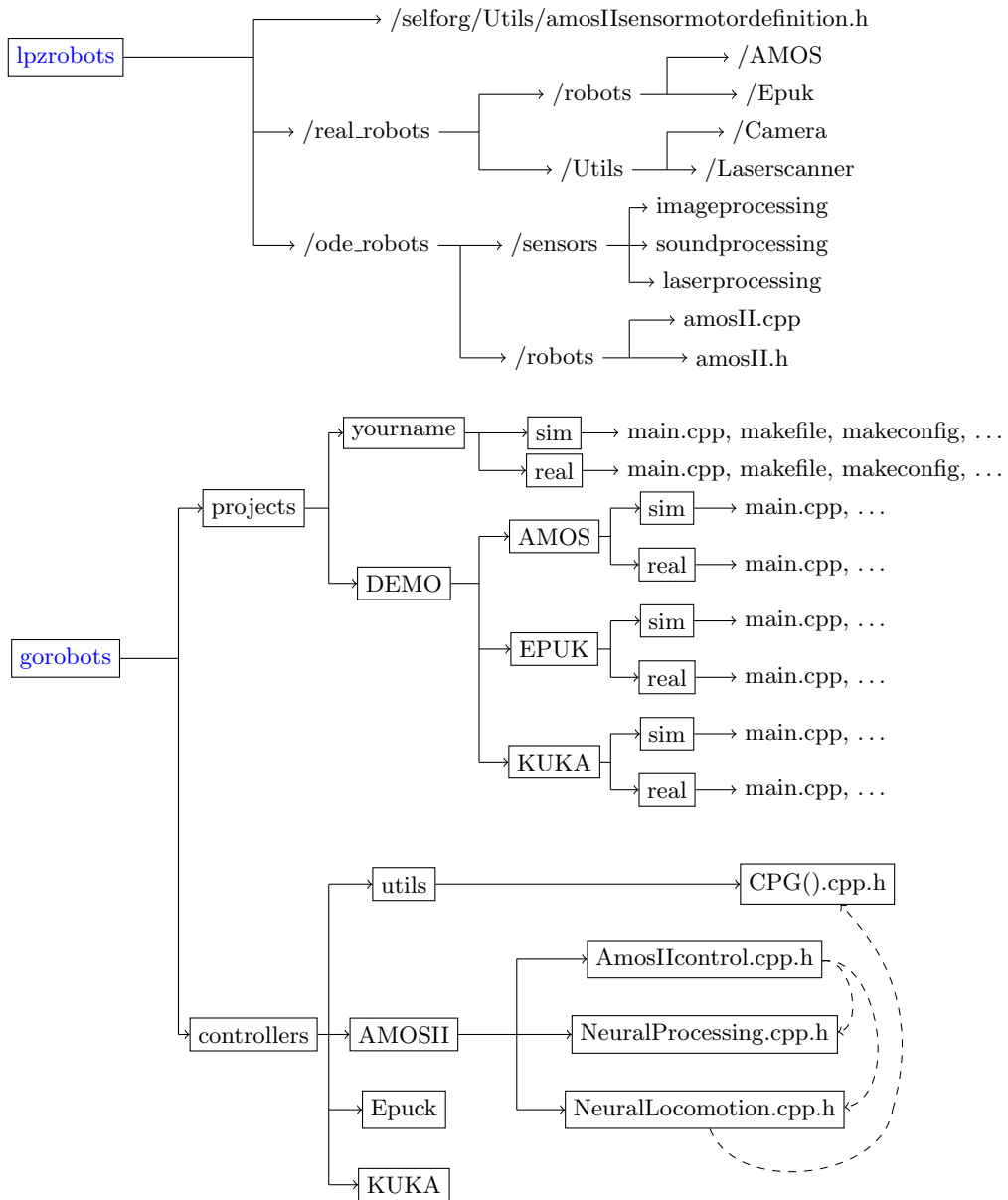


Figure 4: Structure of the two repositories, LpzRobots and GoRobots

4 Working with GIT

4.1 Overview of GIT-commands

The following picture will give a good overview over the workflow and the commands in GIT:

The main advantage in GIT is, that every host has got his own local repository, which

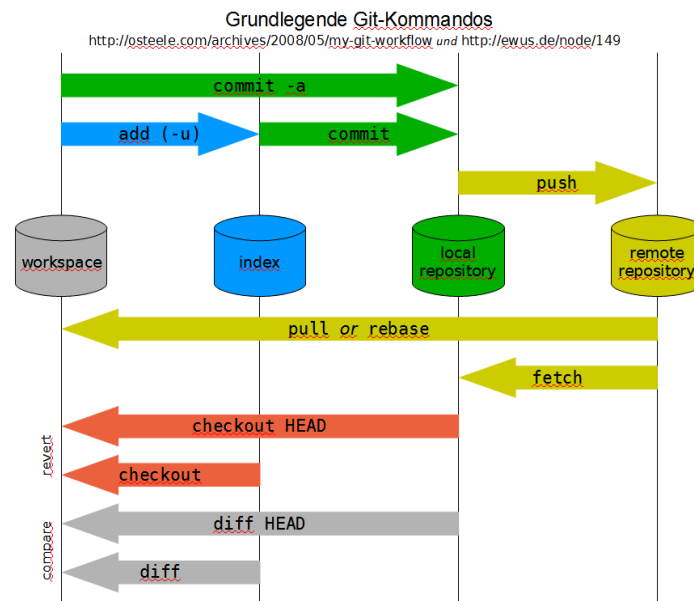


Figure 5: An Overview over GIT commands

in itself is an external repository to another host.

I will give an example, in which I will try, to use most of the commands GIT gives us:

- You create a new file in your workspace in your own branch, that was not under version control before
- Now you first have to add it to the [index](#), which for GIT is just a list of files, that it has to monitor. You can do this using [add](#)
- The next step is adding the actual file to the local repository. You can do this via [commit](#)
- Now the file is under version control. You can now could get the file from the repository using [checkout HEAD](#)
- If you also want to add the file to the remote repository, you can use [push](#)
- Now another person can get your whole work using [pull](#), which will provide him with everything, that is currently listed in your [remote repository](#)

Please note: Be careful whilst merging, Eclipse will offer you to “overwrite”, which is not a good idea, as it does exactly what it promises instead of merging. We will support this warning with a screenshot if possible! To provide a full overview over the commands, I will list all the commands, that Eduard did show us in his presentation:

- [git add](#): adds file changes in your workspace to your index
- [git commit](#): Commits all the changes listed in the index to the local repository
- [git push](#): Pushes local branch to the remote repository
- [git fetch](#): Fetches all files from the remote repository, that are not in the local repository
- [git merge](#): Merges one or more branches into your current branch
- [git pull](#): Fetches files from remote repository and merges them with local files (equal to [git fetch](#); [git merge](#))
- [git rm](#): Removes a file from your repository

4.2 Setting up your own branch with GIT

Within your repository, you should still create branches for each feature you develop. To create a new branch you simple do:

1. In Eclipse, right-click on [name_of_your_fork](#)
2. Select **Team**→**Switch to**→**New Branch**
3. As shown in the picture bellow, select the source:
[refs/remotes/origin/goettingen_master](#)¹
4. Choose a name following the scheme yourname_yourfeature
5. Activate the checkbox for checking out the new branch
6. Work away

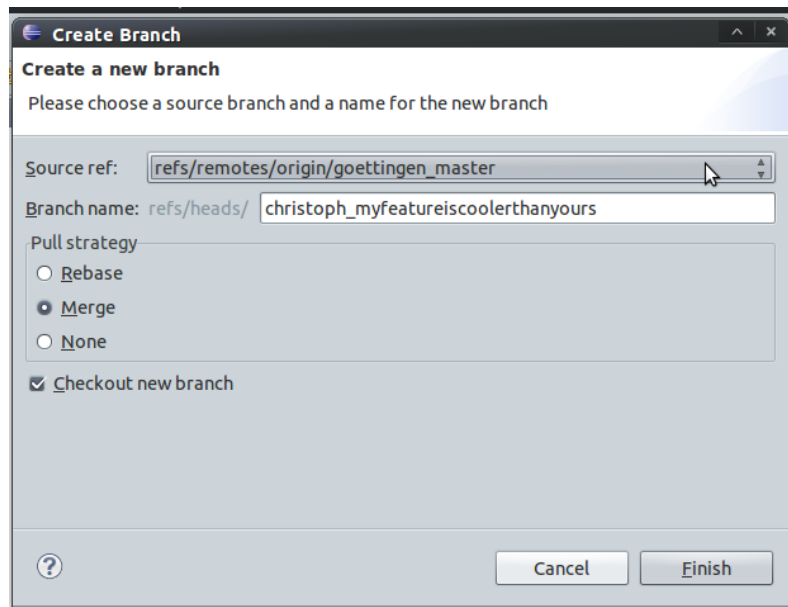


Figure 6: How to create a new branch within Eclipse

¹By choosing this, you make sure, that you have the latest online version

4.3 Forking a Repository

To create a fork of a repository, you need access to the assemblea-website. Once you have logged in, you can choose a repository and then click on the button [fork network](#). One click on the button [fork](#) will then create a fork of the repository for you, as shown in Figure 7. You will furthermore be asked for a name of this copy of the repository, remember this name, as you will need it for the setup!

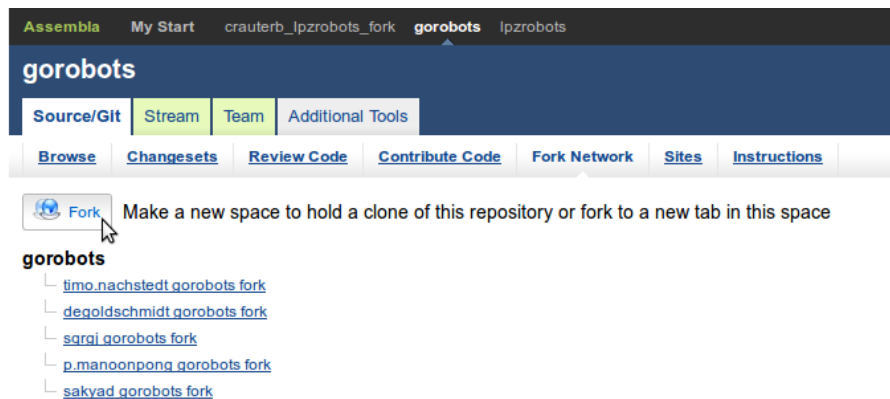


Figure 7: How to fork a repository at the Assembla website

4.4 Merging

Please check with your supervisor, before merging back into [master](#). If you want to merge a branch of yours, you will have to send a merge request via the assemblea-webpage. A supervisor will then have a look at your code. **BUT** he is only able to do so, if you have given him or her the rights to read your code. You can do so by, on the assemblea-webpage in your forked repository, select [team](#) and then, on the right, select [Invite people from other teams](#). No you can choose which one you want to invite, and then you can choose which rights you want to give him or her.

4.5 How to update your Master Branch

Make sure that you are in your master branch by typing:

```
~/workspace/YOURLOGIN-gorobots-fork $ git branch
```

The output should be something like this:

```
goplus
feature_branch_1
feature_branch_2
feature_branch_3
* master
```

The star indicates, in which branch you are working right now. If you are not working in the [master](#)-branch, switch to the branch by typing:

```
~/workspace/YOURLOGIN-gorobots-fork $ git checkout master
```

You can use [git branch](#) again, to check if you are in the right branch now.

If you are in the master branch type

```
~/workspace/YOURLOGIN-gorobots-fork$ git remote -v
```

The output should be something like this (if you did not add a remote before):

```
origin https://YOURLOGIN@git.assembla.com/YOURLOGIN-gorobots-fork.git (fetch)
origin https://YOURLOGIN@git.assembla.com/YOURLOGIN-gorobots-fork.git (push)
```

Now you have to add a stable repository as additional remote type:

```
git remote add stable https://YOURLOGIN@git.assembla.com/gorobots.git
```

[git remote -v](#) should now show something like:

```
~/workspace/YOURLOGIN-gorobots-fork $ git remote -v
origin https://YOURLOGIN@git.assembla.com/YOURLOGIN-gorobots-fork.git (fetch)
origin https://YOURLOGIN@git.assembla.com/YOURLOGIN-gorobots-fork.git (push)
stable https://YOURLOGIN@git.assembla.com/gorobots.git (fetch)
stable https://YOURLOGIN@git.assembla.com/gorobots.git (push)
```

This means you are able to connect to the stable repository (referred to as stable) from your local repository.

Now update your local repositories information about the stable repository by typing:

```
~/workspace/YOURLOGIN-gorobots-fork $ git fetch stable
```

After typing your password you see an output like the following:

```
From https://git.assembla.com/gorobots
* [new branch]      master    -> stable/master
```

Your local repository knows now what branches are available in the stable repository. Now merge the changes of the stable master in your workspace (which currently contains your local master branch) by typing:

```
~/workspace/YOURLOGIN-gorobots-fork $ git merge stable/master
```

If there were now changes, the Output would be something like:

Already up-to-date.

An example of an output including changes would look like:

Updating 1054d9a..330d652

Fast-forward

```
docs/README | 6 ++++++
docs/install_manual/README | 5 +++++
docs/install_script/README | 6 ++++++
3 files changed, 17 insertions(+), 0 deletions(-)
create mode 100644 docs/README
create mode 100644 docs/install_manual/README
create mode 100644 docs/install_script/README
```

Now you can use [git commit](#), to commit changes to your local repository, and [git push](#), to commit changes to the remote repository.

Now your master branch is up to date with our stable version and you can continue with the next section.

5 Installing LpzRobots by Hand

Before you install LpzRobots, you first need to install a couple of additional packages. You can do this using the following commands:

```
sudo apt-get install\  
g++ make automake libtool xutils-dev m4 libreadline-dev libgsl0-dev\  
libglu-dev libgl1-mesa-dev freeglut3-dev libopencscenagraph-dev\  
libqt4-dev libqt4-opengl libqt4-opengl-dev qt4-qmake libqt4-qt3support gnuplot2
```

Also, you will need this package:

```
sudo apt-get install binutils-gold3
```

Furthermore, you will need to add some lines to your `.bashrc`:

```
# definitions for lpzrobots  
export CPATH="$HOME/include"  
export LIBRARY_PATH="$HOME/lib"  
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$HOME/lib:/usr/lib/osgPlugins2.8.1  
export PATH=${PATH}:$HOME/bin
```

Now you are ready to install the LpzRobot-Simulation.

As Eclipse has now synchronized its workspace with the GIT-Repository, you can find the setup-files in your workspace.

To install LpzRobots, go to `workspace→lpzrobots` and run `make (all)`.

Install it to `/home/YOURLOGIN` and choose `install as developer`, which is the shortcut `d`

²You can also find this list of dependencies in the repository [lpzrobots](#)

³You may need this only for newer Ubuntu-Versions (≥ 11.10), as the linker does not link anymore

6 FAQs

6.1 Encountered errors

The error messages was:

```
> ./start
> ./start: error while loading shared libraries: libode\_dbl.so.1:
  cannot open shared object file: No such file or directory
```

The solution was:

```
> source ~/.bashrc
```

6.2 Errors using setUpGoRobots.sh

I encountered the problem whilst cloning the git repositories:

```
Cloning into crauterb-lpzrobots-fork...
Password:
remote: Counting objects: 25478, done.
remote: Compressing objects: 100\% (6030/6030), done.
remote: Total 25478 (delta 19211), reused 25478 (delta 19211)
Receiving objects: 100\% (25478/25478), 19.97 MiB | 716 KiB/s, done.
Resolving deltas: 100\% (19211/19211), done.
warning: remote HEAD refers to nonexistent ref, unable to checkout.
```

The solution was simply, that the reference in the `git clone` command was set to `master`, but for lpzrobots, there is no master, so I had to change the command to:

```
git clone https://$crauterb@git.assembla.com/$crauterb-lpzrobots-fork.git -b goettingen_master
```

6.3 Problems with GIT

In some cases, the instructions on how to set up the GIT-repositories within Eclipse did not work.

Here is a different approach, that worked for me:

1. Import the repositories into the GIT-view of Eclipse, just as described before
2. Instead of importing over the GIT-View, you now go onto `File → Import → General → Existing Projects into Workspace` and you then choose the two repositories
3. After Eclipse has imported the files, you can `right-click` on the Project, and then select `Team → Share`
4. No, just select `GIT` and the two GIT-repository-adresses should appear
5. `Apply`

6.4 Problems within the Script

6.4.1 Error with `git clone`

Whilst cloning the repositories, I encountered the following error:

```
git clone https://crauterb@git.assembla.com/lpzrobots.git -b goettingen-master
Cloning into crauterb-lpzrobots-fork ...
Password:
remote: Counting objects: 25478, done.
remote: Compressing objects: 100% (6030/6030), done.
remote: Total 25478 (delta 19211), reused 25478 (delta 19211)
Receiving objects: 100% (25478/25478), 19.97 MiB | 527 KiB/s, done.
Resolving deltas: 100% (19211/19211), done.
warning: Remote branch master not found in upstream origin, using HEAD instead
warning: remote HEAD refers to nonexistent ref, unable to checkout.
```

The problem is, that we switched from the branch name `goettingen-master` to “just” `master`. In my case, I forked the repository Lpzrobots when there was no branch `master`, and for some reason, it was not added later and did not appear anywear. The solution was simple: Delete the forked version and create a new one - if possible. Worked for me.