LpzRobots and GoRobotos

An installation manual

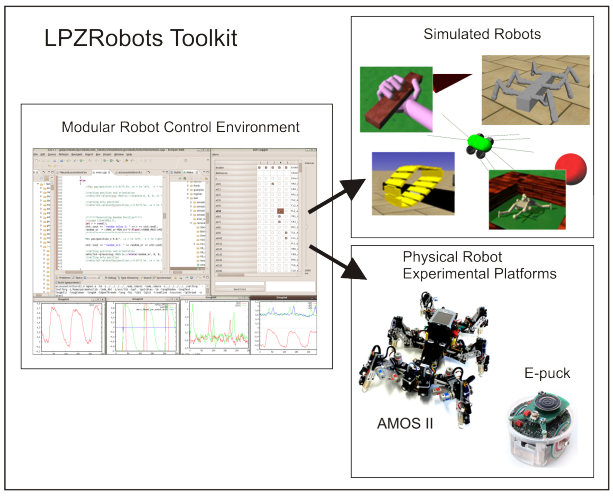
Frank Hesse, Christoph Rauterberg, Poramate Manoonpong

June 16, 2014

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

bccn 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

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:

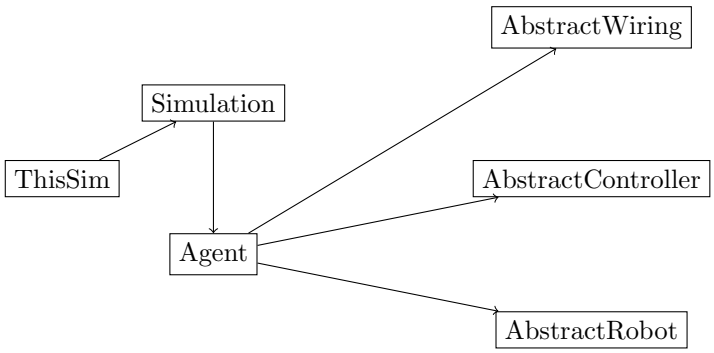


Figure 1: Software Architecture for LpzRobots and GoRobotos

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.

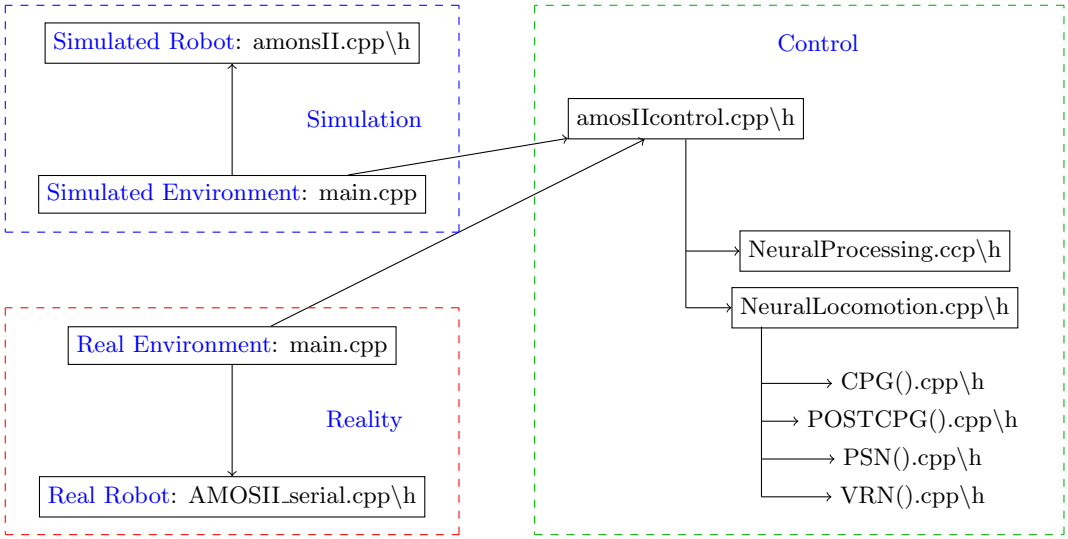


Figure 2: Need name for this picture

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

After extracting, you can run the script, by typing ./setUpGoRobots.sh in the corresponding directory. However, you might have to change permission "chmod u+x setUpGoRobots.sh" if you cannot execute.

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 Eclispe

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.

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.

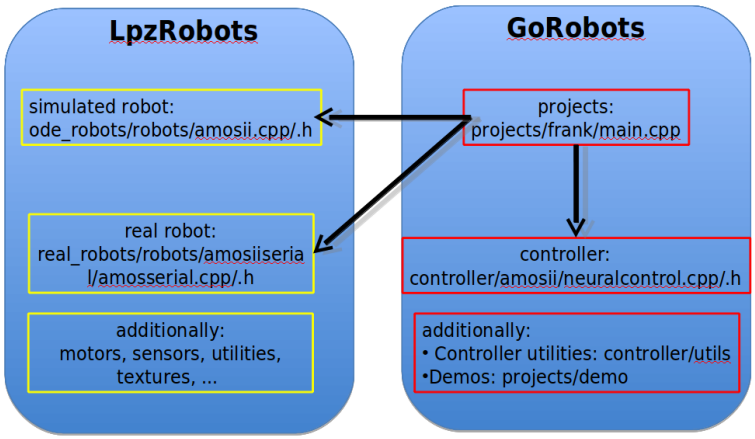


Figure 3: Structure of the two repositories, LpzRobots and GoRobots

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

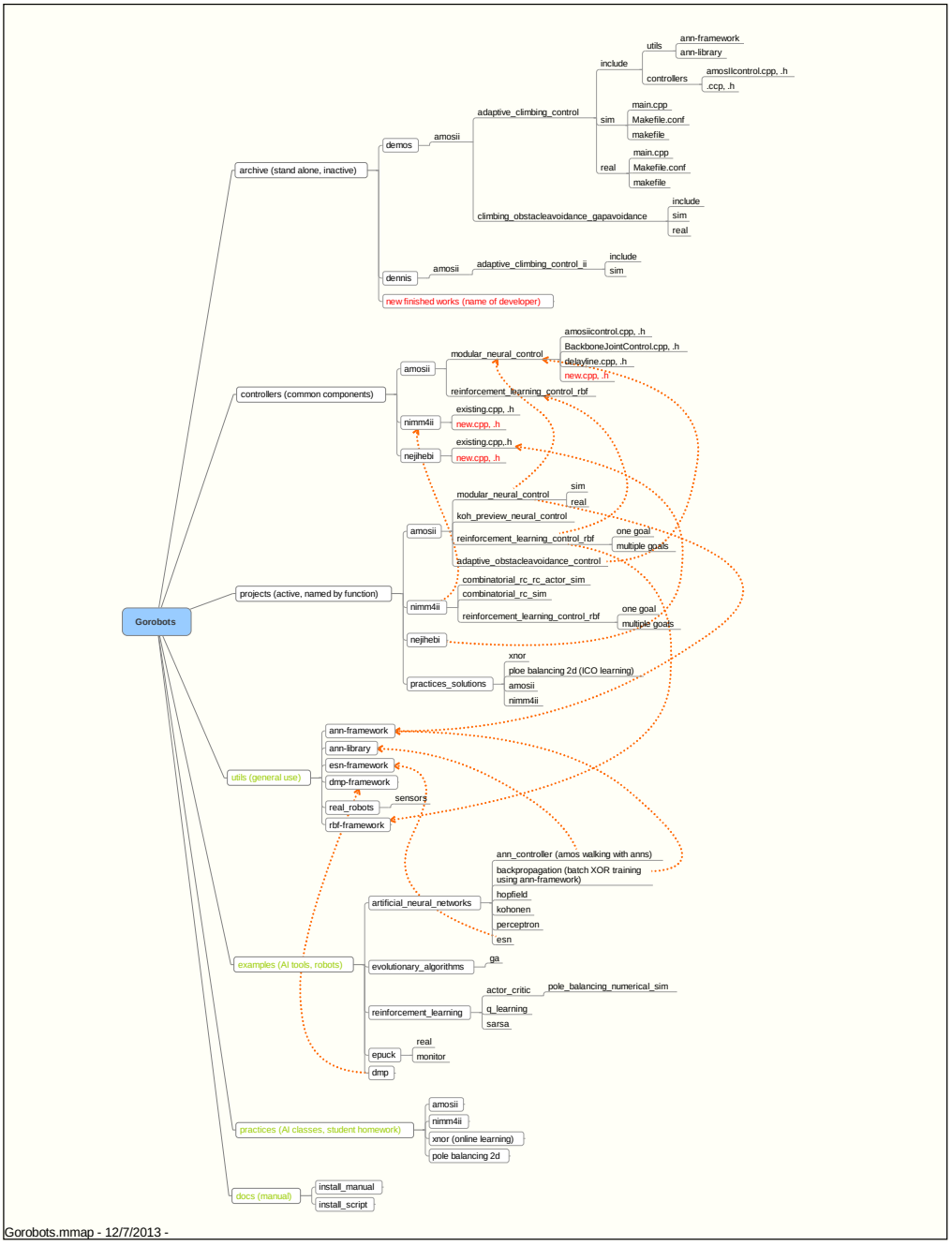


Figure 4: Structure of GoRobots. Green parts belong to GoRobotsEdu

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

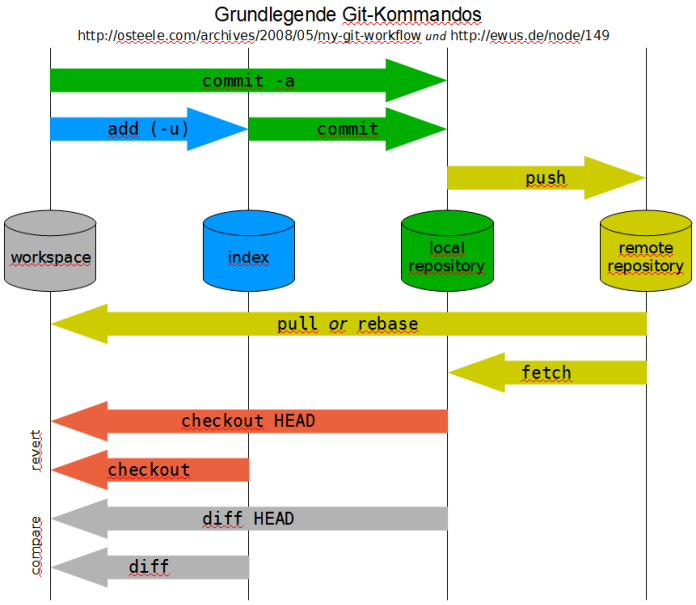


Figure 5: An Overview over GIT commands

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 promisses 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

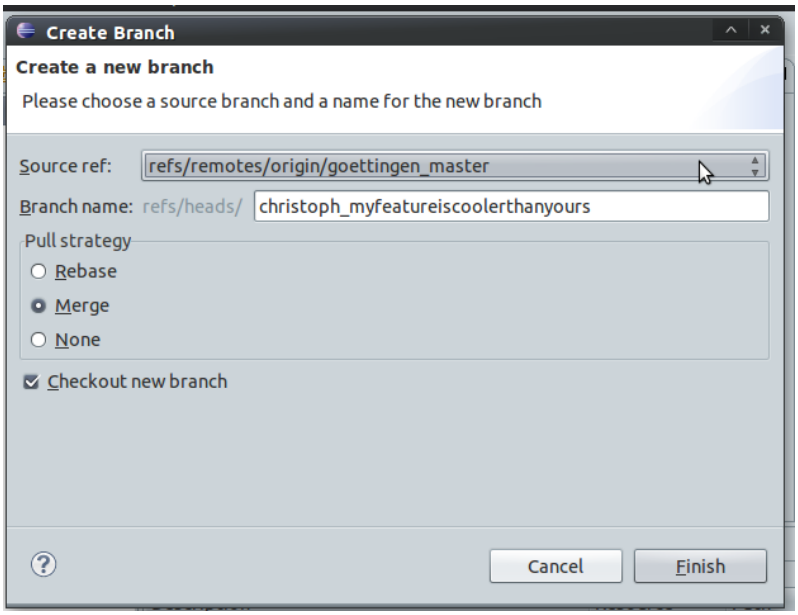


Figure 6: How to create a new branch within Eclipse

[1] By 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 assembla-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!

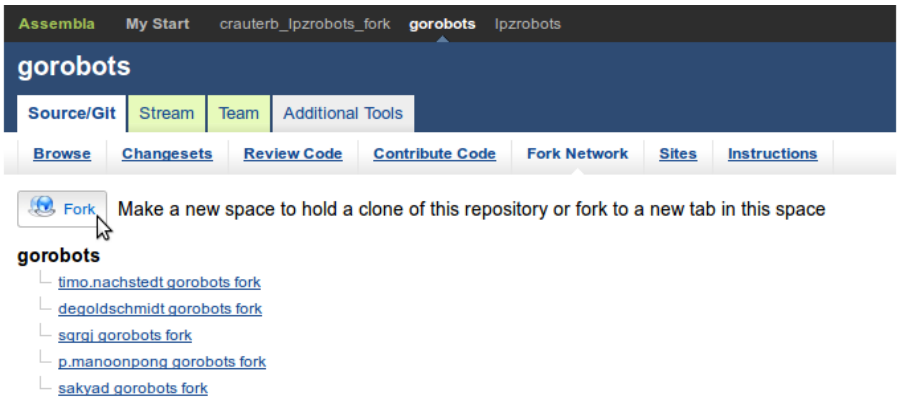


Figure 7: How to fork a repository at the Assembla website

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 assembla-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 assembla-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 of 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 libopenscenegraph-dev\
libqt4-dev libqt4-opengl libqt4-opengl-dev qt4-qmake libqt4-qt3support gnuplot[2]
Also, you will need this package:
sudo apt-get install binutils-gold[3]
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

[2] You can also find this list of dependencies in the repository lpzrobots
[3] You may need this only for newer Ubuntu-Versions (>= 11.10), as the linker does not link anymore

6 Using the Google Test Framework for Unit Testing

We are trying to maintain our code with the help of a unit testing framework, namely googletest.

6.1 Installing the framework

To be able to use this feature and develop new test cases, you first have to install the googletest framework. Here are the necessary steps:

Start by downloading the source code of the most recent googletest version. It is available under `https://code.google.com/p/googletest/downloads`. Decompress the archive and enter it in a terminal:

```
$ cd gtest-1.7.0
```

Inside the decompressed folder, create a new directory called "build" and enter it:

```
$ mkdir build
$ cd build
```

Use cmake to create a Makefile and use this Makefile to build googletest:

```
$ cmake ..
$ make
```

You'll find a libgtest.a and libgtest_main.a file inside the build directory. Copy these files to some path that is registered in the LD_LIBRARY_PATH environment variable. Typically, during the installation of lpzrobots, you add the lib folder in your home directory to the LD_LIBRARY_PATH. To copy the files there, type:

```
$ cp libgtest.a libgtest_main.a ~/lib
```

Furthermore, you have to copy the folder gtest inside the include directory located in the googletest root folder to some path that you include into the search path while compiling gorobots simulations. You can use the include folder created during the lpzrobots installation (typically in your home folder). Starting in the build directory, type the following:

```
$ cd ../include
$ cp -r gtest ~/include/
```

This finishes the installation of the googletest framework. You may now enter the gorobots folder and build the test cases:

```
$ cd ~/workspace/gorobots
$ make test
```

You should also try to run the tests:

```
$ ./build/run_test
```

6.2 Creating test cases

There is a nice tutorial about the basic ideas and concepts of the googletest framework at the webpage of the project. You should definitely read it before going on: `https://code.google.com/p/googletest/wiki/V1_7_Primer`

Within the gorobots project, we want to use the googletest framework for real unit testing as well as for something that is more like integration testing. Unit testing refers to the testing of as small as possible subunits of code. For instance, there is a test for the ann-framework that checks whether the activity property of the Neuron class works as expected:

```
TEST(NeuronTest, activity) {
  Neuron neuron;
  neuron.setActivity(1.2);
  ASSERT_EQ(1.2, neuron.getActivity());
}
```

This is as primitive as it can get. Observe that "NeuronTest" is the name of the test case and "activity" the name of the actual test. You should make sure your test case names are unique within gorobots.

For creating a test for a given unit of code, create a new file in the "tests" folder within the gorobots repository. The path of the file within the tests folder should resemble the path of the tested unit in the gorobots repository. Furthermore, the name of the test file should be the name of the unit plus "_test". If, for instance, the files containing the actual code are

```
utils/ann_framework/neuron.h
utils/ann_framework/neuron.cpp
```

the correct path and name of the corresponding test file is

```
tests/utils/ann_framework/neuron_test.cpp
```

Within your new test file, you have to include the header of the googletest framework and the headers necessary to perform your tests. For instance:

```
#include "gtest/gtest.h"
#include "utils/ann-framework/neuron.h"
```

Afterwards, just start to define your tests.

When it comes to testing of projects, thins get a little more trickier. This is due to our standard of defining the simulation class within the main.cpp. Unfortunately, every main.cpp typically also contains one main method and we cant't link multiple objects that contain methods with the same name. The solution for this is a little hackish: Before including the main.cpp, which is necessary here, add two define statements to override the name of the main method and the ThisSim class. The latter is neccessary to avoid identically named classes in different object files. The results might look like this:

```
#include "gtest/gtest.h"

// mask names of objects in global scope
#define main projects_nejihebi_example_main
#define ThisSim ProjectNejihebiExampleMainSim

#include "projects/nejihebi/example/sim/main.cpp"
```

Within the actual test of your project, you should run your simulation for an as short as possible time interval and check afterwards whether the relevant parameters of the robot have the expected values. This might require too add some additional methods to the ThisSim class. Check out the test of the Nejihebi example project to get an idea how this might look like:

```
tests/projects/nejihebi_example_test.cpp
```

7 Additional (real) Hardware: Camera and Laserscanner

7.1 Step by step to install: Camera, LpzRobots and Eclipse, Laser scanner

1) Linux: Install Linux e.g., Ubuntu 12.04
2) Camera: Install OpenCV and AprilTag for Camera (see How to Use Camera.pdf) OR follow the steps below

1. Use Synaptic Package Manager to install (OpenCV): libopencv-dev
2. Install necessary packages of AprilTag:

   ```
   See http://april.eecs.umich.edu/wiki/index.php/Download_and_Installation
   ```

   OR

   ```
   sudo apt-get install emacs git-core ant subversion gtk-doc-tools
   libglib2.0-dev libusb-1.0-0-dev gv libncurses-dev openjdk-6-jdk
   autopoint libgl1-mesa-dev
   ```

3. Install LCM of AprilTag:

   ```
   cd $HOME
   svn checkout http://lcm.googlecode.com/svn/trunk lcm
   cd lcm
   ./bootstrap.sh
   ./configure
   make
   sudo make install
   ```

4. Install libdc1394of AprilTag:

   ```
   cd $HOME
   svn co https://libdc1394.svn.sourceforge.net/svnroot/libdc1394/trunk/libdc1394/
   cd libdc1394
   autoreconf -i -s
   ./configure
   make
   sudo make install
   ```

   OR

   ```
   Note that I met with a problem when I install libdc1394.
   So, another way to do this is:
   At terminal: aptitude search libdc1394
   then you can see several files there.
   Choose libdc1394-22 and libdc1394-22-dev to install, like:
   sudo aptitude install libdc1394-22
   sudo aptitude install libdc1394-22-dev
   ```

5. Set up environment variables of AprilTag by adding the following lines to .bashrc

```
export CLASSPATH=$CLASSPATH:/usr/share/java/gluegen-rt.jar:/usr/
local/share/java/lcm.jar:$HOME/april/java/april.jar:./
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/april/lib
alias java='java -ea -server'
```

6. Then reload your bash settings (you won't need to do this the next time you log in):

```
source ~/.bashrc
```

7. Install April Toolkit:

```
cd $HOME
git clone git://april.eecs.umich.edu/home/git/april.git
cd april/java
ant
```

8. Test AprilTag -> You can test the 3D visualization environment with:

```
java april.vis.VisTest

Or try out the camera acquisition software with:

java april.jcam.JCamView

Or try to run camera test program at:
\camera\ToSteffen
Make sure that "cvCaptureFromCAM(-1) or cvCaptureFromCAM(0)"
in test_client.cpp

Terminal 1:
java j_server_test

Terminal 2:
make clean
make
./test_client
```

3) Lpzrobots and Eclipse (see Setting up Eclipse and LpzRobots section)

```
If you want to use Laser scanner then make sure that in
./setUpGoRobots.sh

These ones are not installed:
#echo "sudo apt-get install binutils-gold"
#sudo apt-get install binutils-gold

>After finish this:
>Start: Eclipses>>Select: Help>Install Software

1. Work with: http://download.eclipse.org/tools/cdt/releases/indigo
```

```
Select: C/C++ Development Tools

2. Work with: http://download.eclipse.org/egit/updates
Select: Eclipse EGIT


Due to Ubuntu 12.04, you need to type sudo make all
then you might get the error messages:

make[2]: Leaving directory '/home/poramate/workspace/pmanoonpong-lpzrobots-fork'
sudo make install_ode_robots
[sudo] password for poramate:


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

The solution was:
When installing Lpzrobots make sure that you install it in
/home/yourlogin, e.g., /home/poramate

If you made this mistake, you could change this by:
From terminal, go to your lpzrobots,
e.g. poramate@ubuntu:~/workspace/pmanoonpong-lpzrobots-fork$

make clean
make clean-all
make conf

Then: set new installing path: /home/yourlogin, e.g., /home/poramate

make all
```

4) Laser scanner

```
utilities/urg
1) sudo apt-get install libsdl-net1.2
2) sudo apt-get install libsdl-net1.2-dev
3) sudo apt-get install libsdl1.2-dev

4) sudo sh configure
5) sudo make
6) sudo make install

poramate@ubuntu:~/Documents/utilities/urg


Go to example:
```

```
poramate@ubuntu:~/Documents/utilities/urg/samples/cpp


Running program:

UrgDevice::connect: /dev/ttyACM0: No such file or directory
poramate@ubuntu:~/Documents/utilities/urg/samples/cpp: sudo ./mdScan
UrgDevice::connect: /dev/ttyACM0: No such file or directory
poramate@ubuntu:~/Documents/utilities/urg/samples/cpp:
```

7.2 Starting Laser scanner demo with simulated and real robots

```
1) Check out "preview_laserscanner"  branch from Lpzrobots repository
2) in the Lpzrobot directory: e.g., ~/worskspace/ yourname-lpzrobots-forks: make conf ,
select option = 1 for using laserscanner,
select option = 0 for not using laserscanner,
3) then make all
4) go to ~/worskspace/ yourname-lpzrobots-forks/oderobots/
simulations/template_laserscanner
5) type make and then ./start
```

8 FAQs

8.1 Installation errors in general

8.1.1 Symbol lookup error

After installing the "binutils-gold" package, I get the following error after recompiling my source code: "./test: symbol lookup error: /usr/local/lib/liburg.so.0: undefined symbol: _ZTIN3qrk10CoordinateE" (Note: liburg is a library for the Hokuyo laser scanners from here or via packet manager).

As "binutils-gold" contains a different linker for C++ files, which is optimized for speed, it is likely that this new linker does not link the laser scanner libraries correctly (probably due to "bad" programming within the drivers). However, this new linker is not necessary for compiling and running lpzrobots, it just speeds up the compilation time. So the solution to the problem is simple: Uninstall "binutils-gold" and everything works.

8.2 Errors using setUpGoRotobots.sh

8.2.1 Error with git clone

Whilst cloning the repositories, I encountered the following error:

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

8.3 Problems with GIT

8.3.1 Setting up Repositories in Eclipse

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. Now, just select GIT and the two GIT-repository-adresses should appear
5. Apply

8.3.2 No Connection to GIT Server: The remote end hung up unexpectedly.

If you defined stable as usual (check with`git remote -v`):
`stable https://wbj@git.assembla.com/lpzrobots.git(fetch)`
`stable https://wbj@git.assembla.com/lpzrobots.git(push)`

but when typing `git fetch origin` you get the following error message:

```
Password for 'https://wbj@git.assembla.com':
error:  RPC failed; result=22, HTTP code = 401
fatal:  The remote end hung up unexpectedly
```

and also defining the origin with http only:
`stable_http http://wbj@git.assembla.com/lpzrobots.git(fetch)`
does not help, try to use the git protocol:
`stable_git2 git@git.assembla.com:lpzrobots.git(fetch)`
`stable_git2 git@git.assembla.com:lpzrobots.git(push)`

8.4 Compile Errors

8.4.1 OpenCV highgui missing

You are using gorobots as a library build ("new makefile system") and you are experiencing the following error while trying to compile, you are missing the highgui library development files of the opencv package:

```
utils/real_robots/sensors/camera/CamTagPositionSensor.h:11:39:
fatal error: opencv2/highgui/highgui.hpp: No such file or directory
```

Install the missing package by typing:

```
$ sudo apt-get install libopencv-highgui-dev
```

8.4.2 C++11 unknown

If you are trying to compile and obtain errors related to the flag

```
-std=c++11
```

it means that you finally will have to upgrade your machine as your version of gcc does not support the C++11 flag.

8.4.3 osg::Geometry::BIND_PER_PRIMITIVE unknown

If your version of Open Scene Graph it too recent, you may observe the following error while trying to compile lpzrobots:

```
osg/osgprimitive.cpp:668:32: error:
  'BIND_PER_PRIMITIVE' is not a member of 'osg::Geometry'
```

One quick fix for this is to just comment out line 668 in osgosgprimitive.cpp. Updating to a more recent version of lpzrobots would resolve the problem as well.

8.4.4 cannot find -ltinyxml2

With the recent version of gorobots where unit test is embedded, you might have this problem after "make" in gorobots:

```
/usr/bin/ld: cannot find -ltinyxml2
collect2: error: ld returned 1 exit status
make: *** [libgorobots.so] Error 1
```

You need to install -ltinyxml2. For installation, you could use synaptics and search for tinyxml2 and then select "libtinyxml2-dev and litinyxml2-0.0.0".

8.5 Errors when starting the simulation

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
```

```
if you use Ubuntu version 12.4 LTS, you might see this problem:
```

```
> ./start
> ./start: symbol lookup error: /usr/lib/libgsl.so.0: undefined symbol: cblas_dnrm2
```

The solution was:

```
 > sudo apt-get remove binutils-gold
```

8.6 Using Lpzrobots

```
> after start simulation , press 1 to fixed camera view
> Ctrl r = record movie
> .\start -f = record log file
> .\start = start program
> .\start -g 1 = display GUI
```

8.7 Using EPuck Monitor

8.7.1 When trying to run qmake, I obtain "unknown QT: widgets"

EPuck monitor is optimized for Qt5. If you obtain the "unknown QT: widgets" warning you are most likely using Qt4 by default on your system. Find out how to enable Qt5 by default on your system. If you are working with Ubuntu 12.04 LTS, the following terminal commands are what you are looking for. They will install Qt5 and enable it as default:

```
$ sudo apt-add-repository ppa:ubuntu-sdk-team/ppa
$ sudo apt-get update
$ sudo apt-get install qtdeclarative5-dev qt5-default
```