BUZZ and ARGOS

Argos:  
  
**Installation:**

Github: <https://github.com/ilpincy/argos3>

Clone the repository.

* Compiling Argos Simulator for PC

$ cd argos3

$ mkdir build\_simulator

$ cd build\_simulator

$ cmake ../src

$ make

* Compiling Argos on Real Robot (not for PC)

$ cd argos3

$ mkdir build\_myrobot

$ cd build\_myrobot

$ cmake -DARGOS\_BUILD\_FOR=myrobot ../src

$ make

* Compiling Documentation

$ cd argos3

$ cd build\_simulator # or 'cd build\_myrobot'

$ make doc

$ sudo make package

**Examples:**

You can download Argos3 examples from the following link:

https://github.com/ilpincy/argos3-examples

**Problems:**

1. argos3 -c command may not be recognised due to the installation not being global. To solve this problem run this in the argos3 main directory.

$ sudo make install

1. GALIB not found: Ignore
2. Lua52 not found:

Lua5.2 cannot be installed properly separately so install ROS, which is a prerequisite for Google Cartographer, which contains Lua52. Run the commands from the following links:

<http://wiki.ros.org/kinetic/Installation/Ubuntu>

<https://google-cartographer-ros.readthedocs.io/en/latest/compilation.html>

1. argos3 command not working due to the error “shared library not accessible”. To rectify this, add the following lines to bashrc:

export LD\_LIBRARY\_PATH=/usr/local/lib/argos3/

1. There may be a problem in running argos3 simulation examples as it may not be able to find the ‘build’ directory in usr/local/lib/argos3 due to which it won’t be able to initialise the controllers.

To resolve this:

Create a link **(soft)** directory “build” from:

usr/local/lib/argos3/build\*

to:

/home/<your\_pc\_name>/argos3-examples/build/

\* indicates linking directory

To create link directory run the following:

$ cd /usr/local/lib/argos3

$ sudo ln –s /home/<your\_pc\_name>/argos3-examples/build/ build

1. buzz\_controller\_footbot error **\*\*Very Complicated, Read Carefully\*\***
2. Note that on doing the above process, the argos3 looks for controllers in the corresponding “controllers” folder in argos3-examples directory.

1. But “buzz\_controller\_footbot” directory is not present there. Create a new directory with this name over there and add subdirectory statement in the CMakeLists.txt file for “buzz\_controller\_footbot”.
2. The actual controller files are in buzz/src/buzz/argos.
3. Create a linking directory with the name “buzz\_controller\_footbot” in the controllers folder such that it opens up in buzz/src/buzz/argos.

$ cd argos3-examples/controllers

$ ln –s buzz/src/buzz/argos/ buzz\_controller\_footbot

1. Error Solved? NO!
2. Open the CMakeLists.txt in the buzz/src/buzz/argos/ and you will find that it needs some files from its parent directory to generate an executable. But that parent directory cannot be directly accessed from the link folder that we just created.

What to do now?

Copy the files in buzz/src/buzz and paste them into a new directory buzz/src/argos/buzz\_files.

Update the CMakeLists.txt file in buzz/src/buzz/argos accordingly.

1. Make Sure there is a “library” attribute with the controller tag in the .argos file. Library attribute has its value as the path of the .so file of the controller

Buzz:

**Installation:**

Github: <https://github.com/MISTLab/Buzz>

* Compilation

$ cd buzz/build

$ sudo make install

* Installation

$ cd buzz/build

$ sudo make install

$ sudo ldconfig <- The last command is only for Linux OS

Sample Run:

1. Download the buzz and argos scripts
2. Compile the buzz scripts using bzzc command
3. Add the .bo and .bdb files generated after compilation to the <params> tag in the .argos file
4. Run the argos file using the command:

argos3 -c <your\_file\_name>.argos

**Note:**

If you get the .argos file from any other source than the one we have uploaded alongwith, make sure to check for the media tag as it is necessary for swarming. Media refers to the communication channel and an incomplete media tag can lead to errors during execution

**Additional Points I found useful to inform:**

1. /usr/local/include/buzz/argos/ (Location of buzz\_controller\_footbot.h to refer to functions)
2. There is no setleds function as used in examples. It is set\_leds.. refer to the .cpp file in above path for better details
3. There is no debug function as used in examples. Debug is a library and print is the function. So use debug.print(<value\_you\_want\_to\_check>
4. To include libraries, type at the top of the buzz file,

include “<file\_with\_complete\_path>”

1. Location of the library files:

/usr/local/share/buzz/include/

1. Location of buzz\_controller\_footbot.cpp:

/usr/local/lib/argos3/build/controllers/buzz\_controller\_footbot