# SETUP THE ENVIRONMENT

# Linux on BeagleBone Blue

Useful information about the board can be found under [Beagle Bone Blue](https://wiki.ntb.ch/infoportal/embedded_systems/ti_sitara_am335x/beaglebone_blue/start). More about working with the board can be found under <https://eewiki.net/display/linuxonarm/BeagleBone+Blue>. FAQs oft he Beagle Bone Blue <https://github.com/beagleboard/beaglebone-blue/wiki/Frequently-Asked-Questions-(FAQ)>

## EEROS on BeagleBoneBlue

Fetch example application and build scripts on host.

$ git clone https://github.com/ntb-ch/BeagleBoneBlue.git

Get EEROS and the hardware wrapper library.

$ cd BeagleBoneBlue

$ ./clone.sh

The library for the roboticscape must be manually compiled. Though the library is already on the target, we must also have it on the host, in order to be able to link an application. Change into:

$ cd robotics\_cape\_installer/libraries

and edit the Makefile therein. Change the following to lines from

CC := gcc

LINKER := gcc

to

CC := arm-linux-gnueabihf-gcc-4.9

LINKER := arm-linux-gnueabihf-gcc-4.9

This is necessary because we work on the host and must make sure to choose the right cross compiler. You must install this compiler with

$ sudo apt-get install g++-4.9-arm-linux-gnueabihf

Checkout Version 0.3.4 from the robotics cape library

$ git checkout v0.3.4

Now you can create the roboticscape library with

$ make

Change back into your project directory and start the compilation of EEROS, the hardware wrapper library, and the application with

$ ./make.sh

Finally, load the executables onto the target with

$ ./deploy.sh

The password is „temppwd“. Edit the deploy.txt file to define which files must be downloaded.   
ssh into the target and change into /opt/eeros/bin. Start the demo application with

$ sudo ./myApp -c HwConfigBBBlue.json

If the libraries have not been copied to the target, copy them by hand to the right place.

$ scp ./build-armhf/libbbblueeeros.so [debian@192.168.7.2:/opt/eeros/lib](mailto:debian@192.168.7.2:/opt/eeros/lib)

$ scp ./build-armhf/libeeros.so.0.0.0.0 [debian@192.168.7.2:/opt/eeros/lib](mailto:debian@192.168.7.2:/opt/eeros/lib)

$ scp ./build-armhf/myApp/myApp [debian@192.168.7.2:/opt/eeros/bin](mailto:debian@192.168.7.2:/opt/eeros/bin)

$ scp ./myApp/HwConfigBBBlue.json [debian@192.168.7.2:/opt/eeros/bin](mailto:debian@192.168.7.2:/opt/eeros/bin)

Make sure that the user „debian“ has rights for this order. If not, on the target enter

$ sudo chmod +777 /opt/eeros

Also make sure that your libraries are linked to the /usr/lib

$ sudo ln -s /opt/eeros/bin/libeeros.so.0.0.0.0 /usr/lib/libeeros.so.0.0.0.0

$ sudo ln -s /opt/eeros/bin/libeeros.so.0.0.0.0 /usr/lib/libeeros.so.0.0

$ sudo ln -s /opt/eeros/bin/libeeros.so.0.0.0.0 /usr/lib/libeeros.so

$ sudo ln -s /opt/eeros/bin/libbbblueeeros.so /usr/lib/libbbblueeeros.so

## RT-Kernel for Beagle Bone Blue

You can also run the Beagle Bone Blue with a real-time Kernel.

Check if there is already a RT-Kernel: (not necessary)

uname -r

If not, download:

sudo apt-get install linux-image-4.4.49-ti-rt-r89

Do not use a newer one. Robotics Cape won't work anymore (ERROR: missing PINMUX driver).

Reboot and delete the old kernel:

sudo apt remove --purge linux-image-4.4.54-ti-r93

## Get your IDE

Install KDevelop

sudo apt-get install kdevelop

# Use prebuilt test programs

<http://wiki.eeros.org/getting_started/tutorials/start>

There are already all tutorials from <http://wiki.eeros.org/getting_started/tutorials/start> compiled on your Beagle Bone Blue EEROS installation. (see setup eeros for beagle bone blue).

To run this applications on your target, navigate to the path where you cloned the git repository.

$ cd /path/to/working/dir/BeagleBoneBlue

In this folder there is a file named „deploy.txt“. There you can specify the files which should be uploaded to the target, e.g.;

//remove comments when copy paste

Install-armhf/lib/libeeros.so.0.0.0.0 // the eeros library for the target

Install-armhf/lib/libbbblueeeros.so // the beagle bone blue wrapper forh the eeros library

myApp/HwConfigBBBlue.json // the hardware configuration file

build-armhf/myApp/myApp // the application

Change or add the files you want to have on your target and run

$ ./deploy.sh

Now you can ssh on your target and run

$sudo ./opt/eeros/bin/myApp -c HwConfigBBBlue.json

# Write your own appliction

Start KDevelop

Create a new C++ Project

Project -> New from Template

Choose Standard/Terminal as project type and type a project name (e.g. test-project)

Click on Next and then Finish

Create the build configuration. It is recommendet to put the target architecture in the name oft he build folder. Put the location where EEROS is installed in the „Installation Prefix“ field. Important: make sure that this is the location where you have installed EEROS.

## Test Program

1. Copy the following code into “main.cpp”
2. #include <iostream>
3. #include <eeros/logger/Logger.hpp>
4. #include <eeros/logger/StreamLogWriter.hpp>
6. int main() {
7. **using** **namespace** eeros::logger;
9. StreamLogWriter w(std::cout);
10. Logger log;
11. log.set(w);
13. log.info() << "Hello, EEROS";
15. return 0;

}

1. Open CMakeLists.txt file, delete the text written on it and copy/paste the following into it:
2. cmake\_minimum\_required(VERSION 2.8)
4. project(test-project)
6. find\_package(EEROS REQUIRED)
7. include\_directories(${EEROS\_INCLUDE\_DIR};${EEROS\_LIBUCL\_INCLUDE\_DIR})
8. link\_directories(${EEROS\_LIB\_DIR};${EEROS\_LIBUCL\_LINK\_DIR})
10. set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} -std=c++11")
12. add\_executable(test-project main.cpp)

target\_link\_libraries(test-project eeros ucl ${CMAKE\_DL\_LIBS})

(Note 1: “test-project” is the name of the dummy-project created for the example. Put your project name where “test-project” is. Note 2: This file is in the project folder. e.g. /home/abajric/work/test-project)

1. Build the project by clicking on the “Build” button up, on the left.

* DEPLOY.sh

# Delta Tutorial, Build your own Controller

Delta’s accessories

Power Supply

(Xbox controller with USB connector)

(Mouse with scroll wheel left/right function)

Micro USB cable

# Preparations

## VM

* Install Oracle VM Virtual Box Manager [VM](https://www.virtualbox.org/) or from the [Kiosk](https://intranet.ntb.ch/ntb/kiosk/) (NTB students)
* 64-bit
* 8GB RAM
* Hard disk VDI (VirtualBox Disk Image) dynamic allocated, 50GB ROM

## Linux

* Any 64-Bit Linux Version.
* This tutorial works with Linux Mint 17.2 Rafaela 64Bit [mint](http://www.linuxmint.com/)
* Remark for VM beginners: its easier to work if you allow copy/paste from host to target and vice versa. [copy/paste](http://www.liberiangeek.net/2013/09/copy-paste-virtualbox-host-guest-machines/).
* Set your computer name to **delta**. You can choose another name, but you have to adapt paths during this tutorial.
* Remember your password!
* Remember the snapshot function of Virtual Box? Make a snapshot after the clean Linux installation.

## EEROS for Beagle Bone Blue

* Ensure you have installed eeros for your Beagle Bone Blue 🡪 SEE HERE

## EEDURO

* Download and install EEDURO on your VM from [EEDURO](https://github.com/eeduro/eeduro-scripts) and follow the README.md
* As described in README.md, build for Host and Target. **Don't execute copy2robot, clean or distclean now!**

## Putty

* Download and install Putty on your host, not in your VM. [putty](http://www.putty.org/)

## Credentials of delta

* Ask your responsible advisor for username, password and IP- Address of the delta.

# Connect to delta

Plug in the power supply, the USB hub with mouse and XBox. Then start the delta with the ON button. Wait till the BeagleBoard has booted and the white light has turned off. The connection procedure depends the knowledge of the IP-Address. If you know it, you're lucky and you can proceed to [Connect to console](http://hw.eeros.org/eeduro/delta/build_own_controller/start#connect_to_console).

## Get delta's IP-Address

* Connect the mini USB with the PC.
* Then open Putty and enter the IP address 192.168.7.2 and choose as Connection type SSH (see image). You will get a security warning, click yes. Then follow the instructions under [login](http://hw.eeros.org/eeduro/delta/build_own_controller/start#login).

## Login

The first time you do this you will see a security alert in the putty console - press “Yes”. Then the following appears on the putty console:

"login as:"

Write the username „debian“ and hit enter. Then enter the password „temppwd“ and confirm with enter. Don't be alarmed if the console does not response while you type your password. If you logged in successfully, the console will show up like this:

Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.8.13-bone63 armv7l)

\* Documentation: https://help.ubuntu.com/

Last login: Wed Oct 7 11:34:45 2015 from 146.136.53.139

debian@arm:~#

# Test programms

The delta has test programms, which allow you to test several things. They are a perfect entry point. Open your VM and [connect](http://hw.eeros.org/eeduro/delta/build_own_controller/start#connect_from_vm) to the delta. Navigate to the folder /opt/eeduro/bin with:

$ cd /opt/eeduro/bin/

List the files in this folder with

$ ls

The following will be shown:

button eeduro-ctrl eeduro-delta encoder led magnet voltage

The green color means that the files are executable, see more under [colors linux files](http://pthakkar.com/2013/03/linux-color-files-ls-command/).  
To execute these test programms, you have to stop the actual program. Which starts every time you start delta.

1. Type:

$ screen -r eeduro

1. The console will write a lot of text. To stop it, press (Ctrl) + © for about 2-3 seconds.
2. The programm is stopped if you can
3. screen is terminating]

root@arm:/opt/eeduro/bin\# ^C

on your console.

Now load the button test program. Type:

$ ./button

If everything works fine, you can press the buttons and you get a log on the console:

button 1: down

button 1: up

You can see the source code of the button test programm here: /home/delta/eeduro-project/eeduro/test/button.cpp

You can also try out the other programms too.

Remark: Its not possible anymore to halt delta by pressing the button for 5 seconds. Instead type “halt” in the console and then turn the power off.

## Problems

If your testprogram does not work, check README.md.

# Toolchain

Now you want to write you own programm. But first you have to set up the toolchain.

1. Start KDevelop. If you still have the last session from ["Hello World"](http://hw.eeros.org/eeduro/delta/build_own_controller/start#eeros) open, click: Session, Start New Session.
2. Now click: Project, Open / Import Project.
3. Navigate to: /home/delta/eeduro-project and select CMakeLists.txt. Press Next.
4. Name the new Project “eeduro-project”, click Finish.
5. Change Build Directory to /home/delta/eeduro-project/build-armhf/
6. With the following settings, the code is compiled for the delta roboter. Put the following to the extra arguments input field:**-DCMAKE\_TOOLCHAIN\_FILE=/home/delta/eeduro-project/linaro-tc/toolchain.cmake -DADDITIONAL\_INCLUDE\_DIRS=“/home/delta/eeduro-project/eeduro/eeduro/include/;/home/delta/eeduro-project/eeros/includes/” -DADDITIONAL\_LINK\_DIRS=“/home/delta/eeduro-project/build-armhf/eeros/src/;/home/delta/eeduro-project/build-armhf/eeduro/eeduro/“** It's no typo, one folder is actually called include and the other include**s**.   
   ATTENTION: be sure to have the Linux Double Quotes, copy and paste can modify them!
7. Build.
8. If you open the project view in KDevelop, you can unfold the eeduro-project and open copy2robot.sh.
9. Change the setc remote\_host “192.168.7.2” to your IP-Address and save.
10. Execute the copy2robot script with the console.
11. Now you have to go trough the chapter [test programms](http://hw.eeros.org/eeduro/delta/build_own_controller/start#test_programms) again. If it works, you have done everything right.

Remark: Now is a good time to make another snapshot of your VM.

# Your own testprogramm

Now create a new testprogramm. The LED's inside the buttons should glow while they are pressed.  
Go to the folder /home/delta/eeduro-project/eeduro/test and create a new file button\_led.cpp.

Copy the following code into this file and program the missing fragment.

#include <iostream>

#include <unistd.h>

#include <eeduro/Board.hpp>

int main(int argc, char \*argv[])

{

eeduro::Board board;

if (!board.open("/dev/spidev1.0")) {

std::cerr << "cannot open SPI device " << std::endl;

return 1;

}

bool button[3] = { };

while (true) {

board.run();

for (int i = 0; i < 3; i++) {

do it yourself!

}

usleep(30000);

}

board.close();

return 0;

}

Note: if you are not able to write the code in on your own, the source code is in the [appendix](http://hw.eeros.org/eeduro/start#button_ledcpp).

Then open CMakeList which is in the same folder. Add the following code to the end of the file:

add\_executable(button\_led button\_led.cpp)

target\_link\_libraries(button\_led eeduro eeros)

Now go to copy2robot.sh and past the following line under the line with voltage:

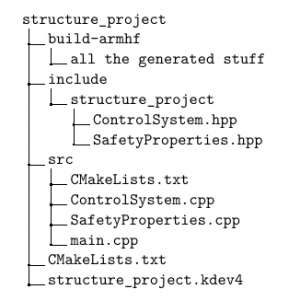
cp build-armhf/eeduro/test/button\_led $tmpdir/bin

Execute copy2robot.sh. ([Like here](http://hw.eeros.org/eeduro/delta/build_own_controller/start#test_programms)), [connect](http://hw.eeros.org/eeduro/start#connect_to_delta) to delta and [check](http://hw.eeros.org/eeduro/delta/build_own_controller/start#test_programms) if button\_led was successfully copied into /opt/eeduro/bin.

Now try to run the new test programm, as described [here](http://hw.eeros.org/eeduro/delta/build_own_controller/start#test_programms).

# Hello Structure

Readers with experience in cmake can skip this chapter.  
  
Bigger projects need to be well structured, so you have to organize your code in folders. Lets create a project with this structure:

[](http://hw.eeros.org/eeduro/_detail/delta/build_own_controller/dirtree1.png?id=delta%3Abuild_own_controller%3Astart)

Create a new project in KDevelop, as usual, with Projects, Create New Project, Standard Terminal and enter the name structure\_project in the location /home/delta/projects. Set the same Extra Arguments as [before](http://hw.eeros.org/eeduro/delta/build_own_controller/start#toolchain). The EEDURO include is useless now, but you may want to extend this project later.

Now navigate to the new folder structure. As you can see, KDevelop has already created CMakeLists.txt(structure), main.cpp and structure.kdev4. Create two folders one called “src” and the other called “include”. Move main.cpp into src, then open src. Create the following empty files: CMakeLists.txt, ControlSystem.cpp and SafetyProperties.cpp. Open the folder “include” and create another folder called “structure\_project”. Here you create ControlSystem.hpp and SafetyProperties.hpp. The .cpp and .hpp (except main.cpp) have no functionality but are a good basis when you want to extend this project.

Open the CMakeLists.txt(src) and paste the following into it:

add\_executable(structure main.cpp ControlSystem.cpp SafetyProperties.cpp)

target\_link\_libraries(structure eeros eeduro)

ControlSystem.cpp and SafetyProperties.cpp in add\_executable are just written here that you see how you add them when you need these classes. “structure” is the name of the executable that will be builded. Once again, the target\_link\_libraries eeduro is useless, you don't need it now. Open the CMakeLists.txt(structure\_project) and paste the following into it:

cmake\_minimum\_required(VERSION 2.8)

project(structure)

include\_directories(${ADDITIONAL\_INCLUDE\_DIRS})

link\_directories(${ADDITIONAL\_LINK\_DIRS})

set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} -std=c++11")

add\_subdirectory(src)

With add\_subdirectory, you specifiy which folders should be included. They need to have their own CMakeLists.txt.

Open the main.cpp and paste the same code as in Hello World into it:

#include <iostream>

#include <eeros/logger/Logger.hpp>

#include <eeros/logger/StreamLogWriter.hpp>

int main() {

using namespace eeros::logger;

StreamLogWriter w(std::cout);

Logger<LogWriter>::setDefaultWriter(&w);

Logger<LogWriter> log;

log.info() << "Hello, EEROS";

return 0;

}

Build and try to execute. Don't remember how? See [hello world](http://wiki.eeros.org/getting_started/tutorials/helloworld).

## Control System

INSERT CONTROL SYSTEM IMAGE HERE

## Safety System

INSERT SAFETY SYSTEM IMAGE HERE

## Sequencer

INSERT SEQUENCER IMAGE HERE

## HAL

INSERT HAL IMAGES HERE

## Adapterprint

INSERT PRINT SCHEMATIC HERE

INSERT DATASHEETS OF MOTOR / ENCODER / GEARS HERE

BEAGLE BONE SCHEMATIC https://github.com/beagleboard/beaglebone-blue/blob/master/BeagleBone\_Blue\_sch.pdf

The Beagle Bone Blue can be operated with the 12V DC Power supply, 8.4V DC from a battery or with the 5V from the USB. You can’t run the motors when only powered over USB, so you have to connect an 8.4V battery or the 12V Power supply. You also need 12V on the adapterprint fort he electromagnet.

To get the +5V to the adapterprint you can simply get it from the PWR port from the Beagle Bone Blue. You’ll need this 5V for both types of encoders used on the Delta. The encoder inputs oft he Beagle Bone Blue are only 3.3V capable. To get the signals from 5V downto 3.3V a voltage divider is used for each encoder channel.

You can get the 3.3V also from the PWR port of the Beagle Bone Blue. This is used for the pullups (HIER FEHLT NOCH WAS). (BESCHREIBUNG BUTTONS LEDS).

To power up and down the electromagnet we use a transistor and a diode. (BESCHREIBUNG HIER NOCH)

LAYOUT HIER EINFÜGEN