Walki Software Platform - User manual

Romain Baud (romain.baud@epfl.ch)

May 26, 2021

Contents

1 Introduction **3**

2 Environment installation **3**
2.1 Source code access . . . 3
2.2 BeagleBone board setup . . . 4
2.2.1 Initial Linux setup . . . 4
2.2.2 Troubleshooting . . . 7
2.3 BeagleBone software development . . . 7
2.3.1 Qt Creator IDE . . . 7
2.3.2 Compiler toolchain . . . 7
2.3.3 Create the configuration file . . . 8
2.3.4 Cross-compilation setup on Qt Creator . . . 8
2.3.5 Source code organization . . . 9
2.4 Motorboard firmware development . . . 9
2.5 MATLAB scripts . . . 10

3 Typical workflows **11**
3.1 Typical exoskeleton controller development workflow . . . 11
3.2 Motorboard test setup . . . 12

4 Geometric and anatomic conventions **12**

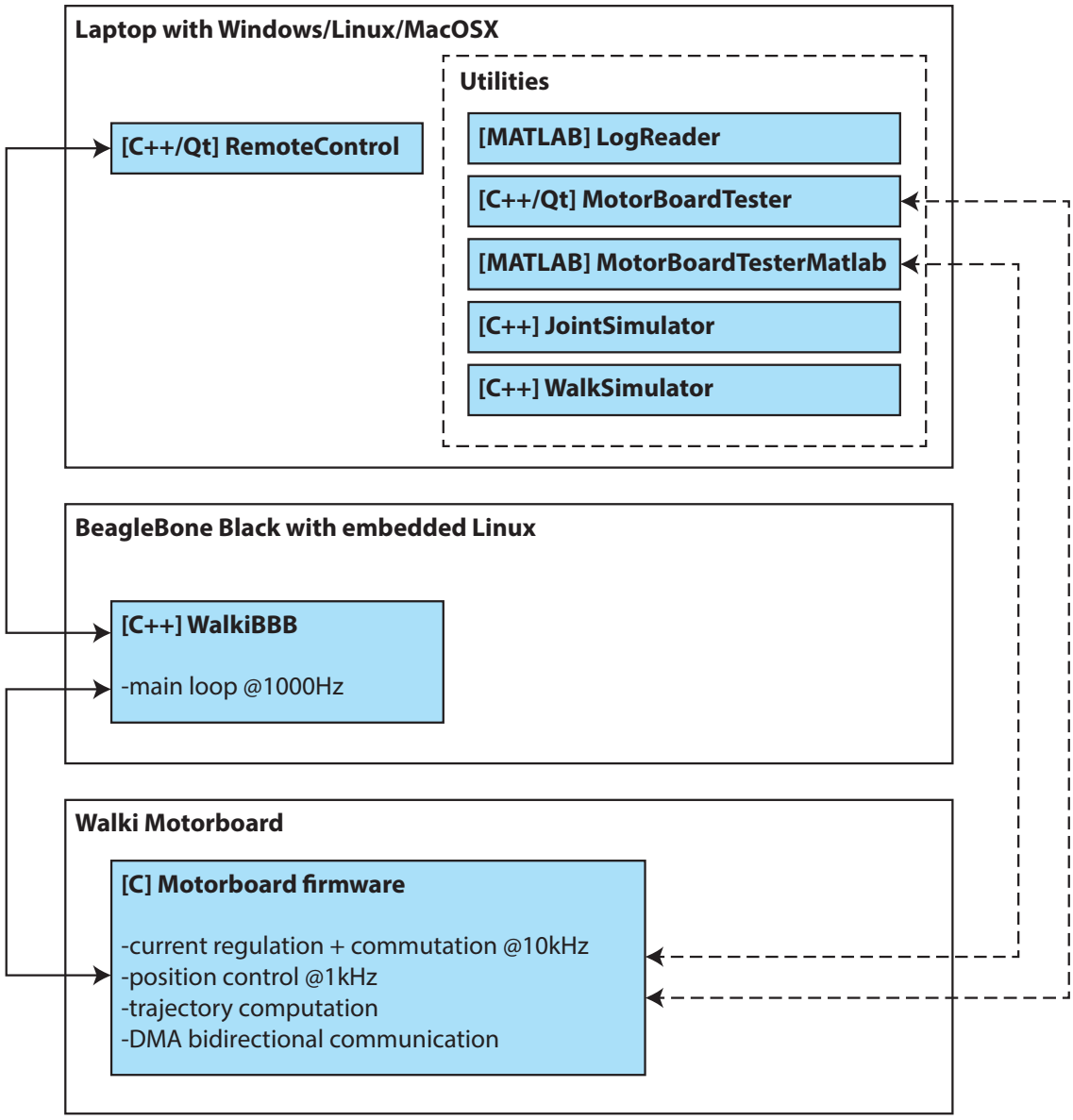

Figure 1: Walki software architecture overview

1 Introduction

The goal of this document is to give an overview of the Walki software platform, and how to get started.

2 Environment installation

2.1 Source code access

All the source code is located on a single Git repository. Ask romain.baud@epfl.ch to get access to it.

```
git clone https://git.epfl.ch/repo/walki-software.git
```

2.2 BeagleBone board setup

2.2.1 Initial Linux setup

Download the latest version of Debian for the BBB (http://beagleboard.org/latest-images), and copy it on the BeagleBone micro SD card using dd (Linux), Win32DiskImager[1] (Windows), or Etcher[2] (Windows/Linux).

Keep pushed the BeagleBone S2 button (to boot from the SD card), connect the BeagleBone to the PC with a USB cable, and wait a few seconds while it boots. You can release the button after the 4 status LEDs are all lit, then starting blinking.
Then, connect to it via SSH, using the login debian / temppwd. You can either use the USB-ethernet link (192.168.7.2), or the serial interface (COMX, /dev/ttyX).

It is advised to connect the BeagleBone to the DHCP network with an ethernet cable. If this is not possible, it is possible to share the laptop thanks to a "Ethernet over USB":

```
/sbin/route add default gw 192.168.7.1
```

Then add a DNS server, by adding the line `nameserver 8.8.8.8` at the end of `/etc/resolv.conf`.
On the laptop side, setup the BeagleBone network interface to use the IP 192.168.7.1, and subnet mask 255.255.255.0.

Setup the bootloader

```
sudo mkdir /mnt/boot
sudo mount /dev/mmcblk0p1 /mnt/boot
sudo cp /mnt/boot/bbb-uEnv.txt /mnt/boot/uEnv.txt
```

From now on, pressing on S2 is not required anymore when powering on the board.

Upgrade the software packages, and install the missing ones

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install wireless-tools usbutils crda firmware-atheros build-essential git
    psmisc cpufrequtils gdb-minimal gdbserver bluez
```

A reboot is probably necessary, depending on what was updated.

Enable the root account

```
sudo su
root_password=$(cat /etc/shadow | grep root | awk -F ':' '{print $2}')
sed -i -e 's:'$root_password'::g' /etc/shadow
sed -i -e 's:#PermitEmptyPasswords no:PermitEmptyPasswords yes:g' /etc/ssh/sshd_config
sed -i -e 's:UsePAM yes:UsePAM no:g' /etc/ssh/sshd_config
sed -i -e 's:#PermitRootLogin prohibit-password:PermitRootLogin yes:g' /etc/ssh/sshd_config
service ssh restart
```

Logout, then log in as root (no password).

Extend SD partition

```
cd /opt/scripts/tools/
git pull
./grow_partition.sh
reboot
```

Setup the Wi-Fi Access point Install and setup the software required to make a access-point: hostapd will setup a Wi-Fi access point, while dnsmasq will distribute the IP addresses.

[1] https://sourceforge.net/projects/win32diskimager/
[2] https://etcher.io/

hostapd setup for standard 2.4 GHz dongle (e.g. TL-WN722N)

```
apt-get install hostapd dnsmasq
```

Setup hostapd. Create the file `/etc/hostapd/hostapd.conf` with the following content:

```
ctrl_interface=/var/run/hostapd
ieee80211n=1
ctrl_interface_group=0
beacon_int=100
interface=wlan0
ssid=Walki_AP_1
hw_mode=g
channel=6
auth_algs=1
wmm_enabled=1
eap_reauth_period=360000000
macaddr_acl=0
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=myownexo
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Note: the digit of the Wi-Fi SSID (`Walki_AP_1`) should be replaced by the mainboard index (any number from 1 to 99).

hostpad setup for 5 GHz dongle (e.g. EW-7811UTC) For this dongle, it is necessary to compile and install the driver, and install a modified version of hostapd.

```
apt-get install linux-headers-‘uname -r‘
git clone https://github.com/gnab/rtl8812au.git
cd rtl8812au/
```

In the file `Makefile`, edit the values of the parameters as follows:

```
CONFIG_PLATFORM_I386_PC = n
CONFIG_PLATFORM_ARM_RPI = y
```

Then, perform the following steps:

```
make
make install
modprobe 8812au

cd ~
apt-get remove hostapd
wget https://github.com/jenssegers/RTL8188-hostapd/archive/v2.0.tar.gz
tar -zxvf v2.0.tar.gz
cd RTL8188-hostapd-2.0/hostapd
make
make install
```

In the file `/etc/hostapd/hostapd.conf`, edit the values of the parameters as follows:

```
ssid=Walki_AP_1
channel=48
wpa_passphrase=myownexo
hw_mode=a
```

Note: the digit of the Wi-Fi SSID (`Walki_AP_1`) should be replaced by the mainboard index (any number from 1 to 99).
In the 5 Ghz band, the following channels are available and compatible with most 5 GHz devices: 8, 12, 16, 32, 36, 40, 44, 48. Higher channels numbers (52, 56, 60, 64, 68) may work with recent devices (e.g. the Nexus 4

cannot use them).
It is also possible to use this dongle with the 2.4 GHz band, by setting "channel" to a number between 1 and 11, and "hw_mode" to "g". This is useful to connect to devices that do not support the 5 GHz band, such as most smartwatches (e.g. the Misfit Vapor 1 and 2) or old laptops.

hostapd common steps and dnsmasq setup Then, in `/etc/default/hostapd`, find the line "`#DAEMON_CONF=""`" and replace it with:

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Setup dnsmasq. Add the following lines at the bottom of the file `/etc/dnsmasq.conf`:

```
interface=wlan0
dhcp-range=192.168.200.100,192.168.200.200,255.255.255.0,12h
```

Setup the Wi-Fi network interface. Edit the wlan0 paragraph in `/etc/network/interfaces`:

```
allow-hotplug wlan0
iface wlan0 inet static
    post-up service hostapd restart
    post-up service dnsmasq restart
    address 192.168.200.1
    netmask 255.255.255.0
    network 192.168.200.0
    broadcast 192.168.200.255
```

Note: the last digit of the address (`192.168.200.1`) should be replaced by the mainboard index.
Enable hostapd and dnsmasq, and reboot.

```
service dnsmasq start
service hostapd start

update-rc.d dnsmasq enable
update-rc.d hostapd enable
reboot
```

The clients can now connect to the network "Walki_AP", with the passphrase "myownexo".
On the BeagleBone side, use the following command to list the clients connected:

```
arp --device wlan0 --numeric
```

Install a real-time Linux kernel

```
cd /opt/scripts/tools/
git pull
./update_kernel.sh --ti-rt-channel --lts-4_9 # You can replace "--ti-rt-channel" by
    "--bone-kernel" if you experience kernel crashes.
reboot

git clone https://github.com/beagleboard/bb.org-overlays
cd bb.org-overlays
./dtc-overlay.sh
./install.sh
reboot

git clone https://github.com/RobertCNelson/dtb-rebuilder.git
cd dtb-rebuilder
git checkout 4.9-ti
```

Edit the file `src/arm/am335x-boneblack.dts` to remove all blocks related to mcasp, otherwise the SPI will not work. For reference, the corresponding edited file can be found in `BeagleBone/scripts/am335x-boneblack.dts`.

```
make all
cp /boot/dtbs/$(uname -r)/am335x-boneblack.dtb /boot/dtbs/$(uname -r)/am335x-boneblack.dtb.bak
cp src/arm/am335x-boneblack.dtb /boot/dtbs/$(uname -r)/
```

In `/boot/uEnv.txt`, edit the "#dtb=" line to "dtb=am335x-boneblack-overlay.dtb".
Reboot.

Enable the SPI1 Copy the SPI DTS file in the Git (`BeagleBone/scripts/BB-SPI1-01-00A0.dts`) to the BeagleBone, using SCP or Filezilla. Then, execute the following commands:

```
dtc -O dtb -o BB-SPI1-01-00A0.dtbo -b 0 -@ BB-SPI1-01-00A0.dts
cp BB-SPI1-01-00A0.dtbo /lib/firmware/
```

To test that the installation worked:

```
echo BB-SPI1-01 > /sys/devices/platform/bone_capemgr/slots
ls /dev/spidev1.*
```

`/dev/spidev1.0` should appear.

Enable the PWM The procedure is very similar to the one to setup the SPI1.
Copy the SPI DTS file in the Git (`BeagleBone/scripts/BB-EHRPWM12-00A0.dts`) to the BeagleBone, using SCP or Filezilla. Then, execute the following commands:

```
dtc -O dtb -o BB-EHRPWM12-00A0.dtbo -b 0 -@ BB-EHRPWM12-00A0.dts
cp BB-EHRPWM12-00A0.dtbo /lib/firmware/
```

To test that the installation worked:

```
echo BB-EHRPWM12 > /sys/devices/platform/bone_capemgr/slots
ls /sys/class/pwm/
```

`pwmchip0  pwmchip2` should appear.

Improve the boot time By default, the booting time is very long (1min30), because of the networking service. Edit the file `/lib/systemd/system/networking.service` and set the parameter `TimeoutStartSec` to `5sec`.

2.2.2 Troubleshooting

SSH session opening time is very long Add `UseDNS no` at the end of the file `/etc/ssh/sshd_config`.

The Wi-Fi access point does not start anymore The Wi-Fi access point will not start if the adapter name (wlan0, wlan1, etc.) has changed. This can happen if the Wi-Fi dongle has been changed. To fix the problem, connect to the BeagleBone via USB, connect to the serial COM port (115200 baud), then open the file `/etc/udev/rules.d/70-persistent-net.rules`, and delete all the lines containing "wlan*". Finally, reboot.

2.3 BeagleBone software development

2.3.1 Qt Creator IDE

This IDE allows code editing, compilation, debugging and remote deployment (http://www.qt.io/download-open-source/).

When executing the setup package, you will need to install the following elements:

- The latest version of the core Qt library (Qt 5.13 MinGW 7.3.0 64 bit, at the time of writing).
- Android ARMv7 (only required to compile the Android applications).
- Qt Charts (not used yet, but will be in the future).
- Qt Data Visualization (not used yet, but will be in the future).
- The latest version of the MinGW compiler, if not already installed.

2.3.2 Compiler toolchain

The cross-compiling toolchain arm-linux-gnueabihf-g++ is required. On most Linux distributions, a ready-to-use packet with this name exists.

On Windows, the Linaro toolchain gcc-linaro-*-i686-mingw32_arm-linux-gnueabihf can be downloaded from `https://releases.linaro.org/components/toolchain/binaries/` [3]. Extract it in the directory of your choice (e.g. `C:\`). Make sure your extracting program runs in administrator mode, otherwise the libraries symbolic links cannot be created.

[3] At the time of writing, the latest version is `https://releases.linaro.org/components/toolchain/binaries/7.3-2018.05/arm-linux-gnueabihf/gcc-linaro-7.3.1-2018.05-i686-mingw32_arm-linux-gnueabihf.tar.xz`

2.3.3 Create the configuration file

Duplicate the file `BeagleBone/config/config_template.h` to `config.h`.

2.3.4 Cross-compilation setup on Qt Creator

It is necessary to create a new mkspec to use the compiler previously installed. To do so, go to `/usr/lib/i386-linux-gnu/qt5/mkspecs/` (Linux 32 bits), `/usr/lib/x86_64-linux-gnu/qt5/mkspecs/` (Linux 64 bits), or `C:/Qt/5.13/mingw73_64/mkspecs/` (Windows, actual path depends on the Qt version) and duplicate the directory `linux-arm-gnueabi-g++` to `linux-arm-gnueabihf-g++`. Then, enter this new directory, and modify the file `qmake.conf` by replacing all occurence of "gnueabi" by "gnueabihf".

Start Qt Creator, go to Tools > Options > Kits > Compilers, and add a new item of type MinGW/C++:

Name: GCC ARM

Compiler path: `/usr/bin/arm-linux-gnueabihf-g++-7.3` (Linux) or `C:\gcc-linaro-7.3.1-2018.05-i686-mingw32_arm-linux-gnueabihf\bin\arm-linux-gnueabihf-g++.exe` (Windows).

Platform codegen flags: `-std=c++1z -lpthread -march=armv7-a -mtune=cortex-a8 -mfloat-abi=hard -mfpu=neon -ffast-math -O3`

Platform linker flags: `-std=c++1z -lpthread -march=armv7-a -mtune=cortex-a8 -mfloat-abi=hard -mfpu=neon -ffast-math -O3`

ABI: `arm-linux-generic-elf-32bit`

Then, go to the "Debuggers" panel, and add a new item:

Name: GDB ARM

Path: `/usr/bin/gdb-multiarch` (Linux) or `C:\gcc-linaro-7.3.1-2018.05-i686-mingw32_arm-linux-gnueabihf\bin\arm-linux-gnueabihf-gdb.exe` (Windows).

Then, go to "Devices" category > "Devices" tab, and add a new item. The assistant will ask for the following parameters:

Type: Generic Linux Device

Name: BeagleBone Wi-Fi

Host name: 192.168.200.X (should match the actual IP address of the BeagleBone).

Username: root

The "Key Deployment" step can be skipped (just click "Next" and "Finish").
The remaining parameters can be left to the default value:

Authentication type: Default (password)

SSH port: 22

Free ports: 10000-10100

Timeout: 10s

Password: [empty]

Private key file: [empty]

GDB server executable: [empty]

Then, go back to the "Kits" category > "Kits" tab, and add a new item:

Name: BeagleBone Cross compile

Device type: Generic Linux Device

Device: BeagleBone Wi-Fi

Compiler: GCC ARM

Debugger: GDB ARM

Qt version: Qt 5.*.* MinGW (replace by the Qt version that is actually installed, the actual version is not important, since only QMake will be used).

Qt mkspec: `linux-arm-gnueabihf-g++`

The kit will show an error, since the version of Qt does not match the compiler, but it can be ignored, since we are actually using only the build tools (qmake) and not the library itself[4].

On Windows, before compiling, make sure that GNU Make is in the PATH. It does not ship with Linaro, but you can use the one from another MinGW installation (it is not target-platform-specific). In this case, one possibility is to copy-paste `C:\Qt\Tools\mingw*\bin\mingw32-make.exe` to `C:\gcc-linaro-7.3.1-2018.05-i686-mingw32_arm-linux-gnueabihf\bin\`.
In case of deployment or remote debugging issues with Windows, check the firewall and the antivirus software, because they may block the SSH connection or the debugger.

2.3.5 Source code organization

Source code documentation The source code is documented using Doxygen. The Doxyfile to compile is in `BeagleBone/doc/`.

2.4 Motorboard firmware development

Tools installation: download and install:

- STM32CubeMX (http://www.st.com/en/development-tools/stm32cubemx.html).
- System Workbench for STM32 (http://www.openstm32.org/System+Workbench+for+STM32).

Install the STM32Cube library Start STM32CubeMX, go to Help > Install New Libraries, and install the latest release of the STM32CubeF7 (firmware package for family STM32F7).
Go to Help > Updater Settings > Updater Settings, to know what the "repository folder" directory is. Go to this directory, and remember the path of the recently installed STM32CubeF7 library (typically `C:/Users/Username/STM32Cube/Repository/STM32Cube_FW_F7_VX.X.X` (on Windows) or `/home/username/STM32Cube/Repository/STM32Cube_FW_F7_VX.X.X` (on Linux)).
Create an environment variable[5] named `CUBEF7_LIB_PATH`, with the STM32CubeF7 library path as value, without a trailing slash. On Windows, slashes should be used instead of backslashes.
Start System Workbench (if it was already open, close and restart it), and go to Window > Preferences > General > Workspace > Linked Resources, and create a path variable named `CUBEF7_LIB_PATH`, with the same value as the environment variable.

Create the configuration file Duplicate the file `MotorBoard/src/config/config_template.h` to `config.h`.

[4] To avoid this error, it would be necessary to manually compile Qt for this target, but this is time-consuming and unnecessary.
[5] On Windows, go to Control Panel > System > Advanced System Settings > Advanced > Environment variables > System variables > New. On Linux (under Debian with the Bash shell), append the command `export CUBEF7_LIB_PATH="path_to_your_cube_library"` to the file `~/.profile`, and logout and login again.

SW4 project setup: This step should normally not be needed, since the project is ready to use. These steps are only useful in case the eclipse project is fully regenerated from STM32CubeMX.

- Import the project into Eclipse.
- Set up the compilation settings.
  - C/C++ Build > Settings > Tool Settings > MCU GCC Compiler > Preprocessor > Defined Symbols
    - `ARM_MATH_CM7`
    - `BOARD_REVISION=X` (replace X by the actual version of the motorboard).
  - C/C++ Build > Settings > Tool Settings > MCU GCC Compiler > Includes > Include paths
    - `PATH_TO_YOUR_F7_INSTALL/Drivers/CMSIS/DSP/Include`
  - C/C++ Build > Settings > Tool Settings > MCU GCC Linker > Libraries
    - `m`
    - `arm_cortexM7lfsp_math`.
  - C/C++ Build > Settings > Tool Settings > MCU GCC Linker > Library search path
    - `${env_var:CUBEF7_LIB_PATH}/Drivers/CMSIS/Lib/GCC`
  - C/C++ Build > Settings > Tool Settings > MCU GCC Linker > Miscellaneous > Linker flags
    - `-u _printf_float -u _scanf_float`
- Modify the generated `Application/User/main.c` file to call the motorboardMain() main function:

```
/* USER CODE BEGIN Includes */
#include "../../src/motorboard_main.h"
/* USER CODE END Includes */

[...]

/* USER CODE BEGIN 2 */
motorboardMain();
/* USER CODE END 2 */
```

- Add the source code.
  - Application > Import > File System
  - Add "Software/MotorBoard/src". Check "Create top-level folder" and "Create links in workspace".
- Build.
- Run.

Improve the startup reliability: In case the 3.3V voltage that powers the microcontroller rises too slowly, there may be random issues. To improve the robustness, the STM32 can be set to remain in reset state until the voltage has reached the "brownout voltage" threshold. To do so, open STM32 ST-LINK Utility, go to Target > Option Bytes, set BOR Level to "Level 3", and press Apply.

Source code documentation The source code is documented using Doxygen. The Doxyfile to compile is in `Motorboard/doc/`.

2.5 MATLAB scripts

MATLAB 2015b or newer is required to run the scripts.
The scripts in `Utilities/LogReaderMATLAB/` are to display and analyze logfiles:

wkv_load.m: loads a WKV logfile generated by the BeagleBone application.

wkv_plot.m: plots the selected variables of a structure generated by `load_wkv()`.

wkmain_logread.m: loads and plots a WKL logfile, generated by the RemoteControl application.

wkmot_commutlogread.m: loads and plot a "phases currents" WKMOTLOG logfile, generated by the MotorboardTester application. Such logs contains the phases currents at maximum rate (10 kHz), which is useful to get evaluate the current regulation while commuting.

wkmot_currlogread.m: loads and plot a "motor current" WKMOTLOG logfile, generated by the MotorboardTester application. Such logs contains the motor current at maximum rate (10 kHz), which is useful to get evaluate the current regulation.

wkmot_logread.m: loads and plot a "position" WKMOTLOG logfile, generated by the MotorboardTester application. Such logs contains the board voltage and current, joints target and actual positions, motor currents, at the rate of the position loop (1 kHz).

The scripts in `Utilities/MiscMATLAB/` are dedicated to specific projects:

hibso_kinematic_simplification.m: computes the reduction ratio of HiBSO as the function of the joint angle, as well as the function that converts between joint and motor angle. It then generates the corresponding look-up tables.

make_c_lut.m: generates C-code for look-up tables to be used by the `utils_ReadLut()` function of the motorboard firmware.

walki_spline_traj_interpolation.m: generate C-code for piecewise polynomials approximation of gait trajectories, that can be evaluated using the Utils::evalPiecewisePolynomial() method of the BeagleBone application.

Finally, the `MotorboardTesterMATLAB/WalkiMotorboard.m` provides a MATLAB class to communicate and control a motorboard through a serial COM port. This allows quick and convenient scripting.

3 Typical workflows

The goal of this section is to give an overview of what is the purpose of each piece of software.

3.1 Typical exoskeleton controller development workflow

1. Power on the BeagleBone Black, wait approximately one minute, until the Wi-Fi network "Walki_APX" appears. Connect to it.

2. Implement your controller in the WalkiBBB source code, in the Qt Creator editor. If a variable in the code should be logged, or remote-controlled, create a "SyncVar" object linked to this variable. Finally, compile it and send it to the BeagleBone, using the "Deploy" command. It is also possible to remotely debug the program.

3. Open a remote Linux session using SSH (with Putty on Windows). Start the WalkiBBB application. The output of the application can be observed, and it can be cleanly closed with Ctrl+C if necessary.

4. Connect to WalkiBBB using the RemoteControl PC application, or the Android app.
   If using the RemoteController PC application, any "SyncVar" can be written, read and plotted, depending on the access rights. This application also generates a logfile recording the SyncVars values over time, if they are read.
   At the end of the session, close the WalkiBBB application.

5. The RemoteControl logfile can be opened and plotted using the wkmain_logread() MATLAB function. This rarely used, because the sampling frequency is the one from the communication (around 50 Hz), and much lower than the real controller rate (around 1 kHz). The logfile generated by WalkiBBB on the micro SD card is usually more informative. This may still be useful to get a trace, in case the exoskeleton logfile cannot be retrieved afterward (e.g. if the micro SD card is destroyed because of fire in the control box, which would be very unfortunate).

6. Download the logfiles on the computer, using scp or Filezilla.

7. Use the MATLAB function wkv_load() to load the desired logfile in a MATLAB structure. Then, display the content of this structure, and note the indices of the relevant variables. The selected variables can then be plotted using wkv_plot().
   It is recommended to save this structure in a MAT file, to load data quicker next time.

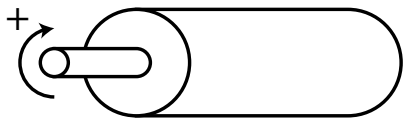

Figure 2: Motor shaft rotation convention

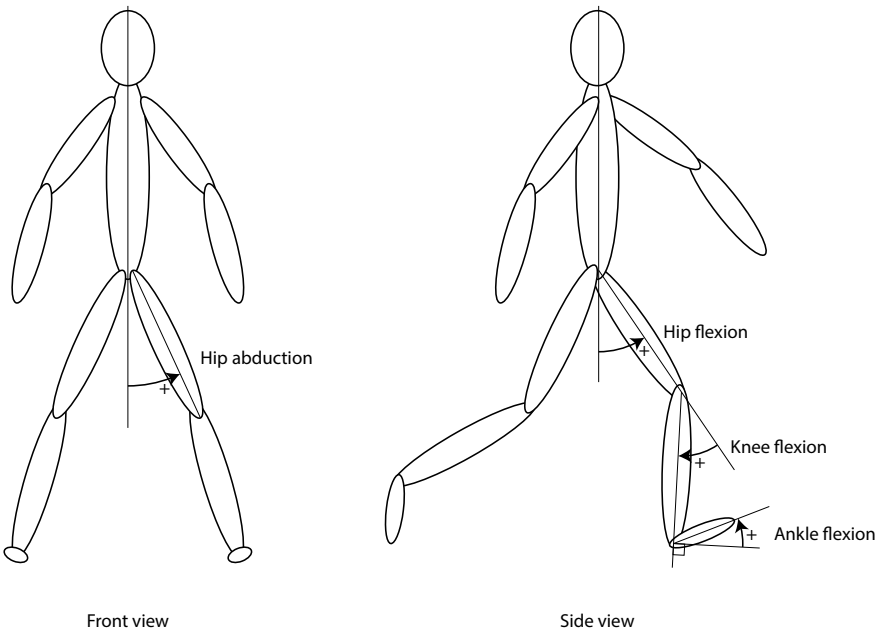

Figure 3: Joints direction convention

3.2 Motorboard test setup

The simplest way to test the motorboard is to connect it directly to a standard PC, using a USB-to-serial adapter connected to the RX and TX jumpers of the motorboard. This makes communicating easier, and allows direct acquisition of the board data, such as the phases currents.

1. Develop the firmware with System Workbench. Send the code to the board, and optionally use the debugger to test features that are not time-critical. In config.h, select between the regular (position) logging, the full-speed current logging or the full-speed phases current logging.
2. Control the board using the MotorboardTester application. A logfile will be generated during the execution. Then, close MotorboardTester.
3. Open the generated logfile with the wkmot_logread()/wkmot_currlogread()/wkmot_commutlogread() MATLAB function, depending on the motorboard configuration.

4 Geometric and anatomic conventions

All the software follow the following conventions.

Motor shaft rotation The positive direction is clockwise, when looking from the side the shaft is protruding (see fig. 2).

Joints rotation The flexion is positive, and the zero angle is defined by the angles when the person stands-up vertically, feets laying flat on the floor (see fig. 3).

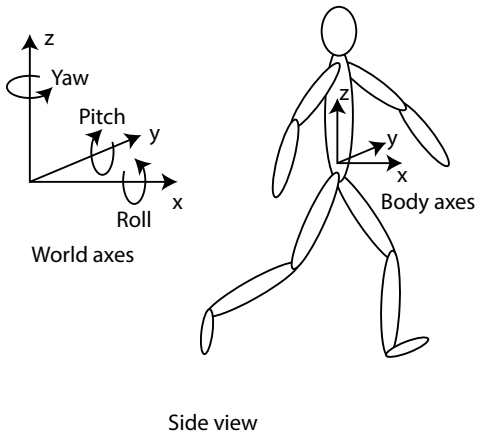

Figure 4: Axes convention

Axes The body axes are defined by the x pointing toward the front, y pointing at the left and z pointing at the top of the body. The world axes are the reference.
The body orientation is defined by the three successive rotations around the world axes, in the following order: roll (x), pitch (y), yaw (z). See fig. 4.