Hexapod User Manual

Dan Thilderkvist Sebastian Svensson

June 16, 2015

Contents

1 Introduction

This documentation contains an overview of the Simulink files and hardware of the Phantom Hexapod Mark II. In some of the Simulink files there are more detailed comments.

2 Requierd toolboxes

The following toolboxes and support package are required to run the model.

- Embedded Coder
- Simulink Coder
- Embedded Coder Support Package for BeagleBone Black Hardware
- Matlab
- SimMechanics
- Simscape
- Simulink
- Instrument Control Toolbox

3 Run Instructions

Using the thesis work might not be totally self explanatory. Here follows a couple of walk-through sections on how to use the different parts. First thing to do in every scenario is to run the init_script.m. This ensures that folders are added to the MATLAB workpath.

3.1 Running hexapod Model

For running the hexapod as a model the file Control/DevelopmentModel.slx can be used. This file contains a Model of the hexapod built up by the different library files and a controller built up similarly. Running the combination of the two is done by basically pressing play and SimMechanics should provide visual results. If there is a desire to change the hard-coded input to the controller one can change this inside the controller block 2 levels down. All changes to main controller and IK should be avoided inside this file. Those changes are better done in the library files.

3.2 Running Code Generation to the hexapod

First assemble the power to the hexapod. Connect the battery to the battery voltage monitor device to be able to monitor battery voltage. The monitor device will beep temporary when connected. When battery voltage is to low the device will start beeping continuously (charge the battery). Also connect the battery to the connection bringing power to BeagleBone Black, ArbotiX and servos. The BeagleBone gets powered directly but in order to power ArbotiX and servos the switch on the back of the hexapod has to be switched.

Start the file Control/CodeGenerationSetup.slx. This files provide two options, run in external mode or deploy to hardware. Running in external mode (play button) provides the ability to monitor signals with scopes in Simulink but needs the BeagleBone connected to the computer by USB. Deploy to hardware (blue deploy button) allows for no wire connection but do not supply scope data.

Power on the remote first after the code has started running to avoid communication being off-sync. When deployed to hardware the script StartBB and StopBB can be used to start/stop the program without having to recompile code.

3.3 Using the remote

A picture of the remote can be seen in Figure 1. Description of each button is described in Table 1.

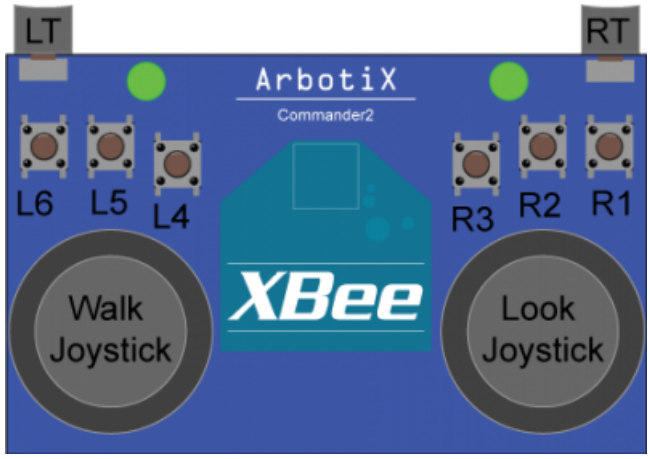

Figure 1: Location on the different controls on the remote.

Name	Description
WalkH ↔	Strafe left/right
WalkV ↕	Move forward/back
LookH ↔	Rotate left/right
LookV ↕	N/A
LT	Lower the body
RT	Rise the body
L6	Normal walking
L5	Experimental walk patterns with constraints
L4	N/A
R3	N/A
R2	N/A
R1	Balancing mode

Table 1: Description of the remote control.

4 Control

The control-structure is described in the master thesis report. Here most of the control files will be described.

The full system is triggered by a trigger-system in order to keep control of execution order. The trigger system is saved as a library and linked to from the various implementations. Trigger system library can be found under Simulink_Lib/Control. Here is also the rest of the control libraries located. IK, MainController, coordinate translater and a test sequence exists. The most advanced one of these is the main controller. If development of the controller is to be done it is inside this it can be performed. The controller implementation utilizes written m-files. These are located at Control/MatlabFunctions and are explained by comments in the code.

There is a folder located as Control/LegConstriants. The script inside is used to get a visualization of the constraint boundaries that can be used for the legs. Script is hopefully self explanatory.

5 Documents

This folder contains documentation for the different electrical components and other important parts of the hexapod.

ArbotiX-M documentation the arduino card ArbotiX-M

Beaglebone black documentation the BeagleBone Black

IMU documentation for MPU9150

Movies contains movies of Simulink models running on hardware and in simulation

Servo manual contains documentation on the AX12A servos

ThesisManual contains .tex files to expand and rebuild this manual

ThesisDocuments contains popular summary,thesis report and a presentation

6 Measurements

This folder contains some code for running step responses on the AX12A servos. To use this files, load `StepTest.ino` onto the ArbotiX card using Arduino IDE. Open the file `TestSetup.slx` and select the com port to which the ArbotiX card is connected. Finally set parameters for the step test in `testSetupRun.m` and run the script. Data will be logged by the ArbotiX card and then saved in a folder structure on the PC.

7 Model

7.1 CAD/Hexapod

The folder `Cad export` contains the file produced by SimMechanics Link and are used for visualization of the hexapod inside of Simulink.

The folder `SolidWorks` contains the SolidWork parts and assembly files to construct the hexapod. `Common leg parts` contains parts to assemble the three leg parts (coxa, femur and tibia). The two folders `Left leg` and `Right leg` contains coxa, femur and tibia part for each leg. The difference between those files are the placement of the servos.

To update mass of the parts first open the part file and check were the extra coordinate system are located. Then open the corresponding assembly file and perform the update needed. Choose Edit-¿rebuild Goto File-¿Save as-¿.part and select All components. Open the part file and insert the coordinate system. Finally use SimMechanics link to export the updated file to SimMechanics. If only the mass of the part needs to be changed this can be done directly in SimMechanics.

The file Hexapod.sldasm contains the complete hexapod to export into SimMechanics. The legs and joints are added when the model have been imported into SimMechanics.

7.2 Contact_force

This folder contains work done to model contact forces in SimMechanics. `Left_LegGCfriction.slx` shows a demonstration of contact with floor for one hexapod leg using SimMechanics contact force library. `hexapod_with_floor.slx` contains the complete hexapod with six feet contact points with the floor. the contact point are placed at the center of each foot `F_GC`. Other approaches for modelling contact forces are presented in the folder `Other approaches`. Some useful links to use when modelling contact forces can be found at `http://www.mathworks.com/matlabcentral/fileexchange/49374-rolling-ball-on-plane` and `http://www.mathworks.com/matlabcentral/fileexchange/47417-simmechanics-contact-forces-library`.

8 Simulink_Lib

This folder contains library files to construct the hexapod inside Simulink and SimMechanics. The file `Custom_buses.mat` contains definitions for the different buses used in the Simulink models.

8.1 BeagleBoneBlack

IMU_DMP contains documentation and code for using the DMP on MPU9150

Test contains files for testing the various S-Function builder blocks

Working Code contains the code generated for the S-Function Builder blocks

BeagleBoneBlackCom.slx contains all S-Function Builder blocks for use on the BeagleBone Black

ServoCommand.slx contains blocks to build up a message to send position references to the servos using the write to servo block found in `BeagleBoneBlackCom.slx`

8.2 How to build a S-Function

Set the working folder in MATLAB to were the generated code ends up e.g. to build the commander block select `WorkingCode\Commander`.

1. Open the S-Function Builder goto Build Info tab-¿ Additional methods select Terminate.
2. Press build in the upper right corner.
3. Open the generated `*_wrapper.c` goto the end of the file and copy the line `close(xbee_fd);`.
4. Open the *SFunctionName*.c and insert the line inside the `static void mdlTerminate(SimStruct *S)` function.

When building the IMU_DMP make sure to update `Libraries tab` with correct path to include existing source code.

8.3 Note on using UART

To make the UART work the following changes have to be made to `uEnv.txt` found in `BeagleBone Getting Started (F:)` on the BeagleBone Black. Before:

```
17 ##WIP: v3.13+ capes..
18 #cape=lcd4-01
19 #cape=
20
21 ##note: the eMMC flasher script relies on the next line
22 mmcroot=UUID=5338fca0-6bf1-4297-8feb-a7a909cad7a2 ro
23 mmcrootfstype=ext4 rootwait fixrtc
```

After:

```
17 ##WIP: v3.13+ capes..
18 #cape=lcd4-01
19 #cape=
20
21 #Setup serial ports
22 optargs=capemgr.enable_partno=BB-UART1,BB-UART2,BB-UART3,
       BB-UART4,BB-UART5
23
24
25 ##note: the eMMC flasher script relies on the next line
26 mmcroot=UUID=5338fca0-6bf1-4297-8feb-a7a909cad7a2 ro
27 mmcrootfstype=ext4 rootwait fixrtc
```

8.4 Control

This files are described in 4.

8.5 Model

Completemodels.slx contains model which are controlled by either torque or position. Also contains a experimental model for use ballonfloor contact force model see `http://www.mathworks.com/matlabcentral/fileexchange/49374-rolling-ball-on-plane`.

Servo model contains models for the AX12A servos. Two models are present one for torque and one for position actuation.

HexapodLib contains models of the different parts of the hexapod. Two models are present one for torque and one for position actuation. The interface blocks are used to connect the input signal to the correct servo id. Check the simulation model to get a example.

SM_Contact_Forces_Lib

A download of the contact forces library found at `http://www.mathworks.com/matlabcentral/fileexchange/47417-simmechanics-contact-forces-library`.

9 Software

Arduino_Phoenix_Parts-master A more advanced program for running the hexapod. More information can be found at `http://learn.trossenrobotics.com/10-interbotix/crawlers/phantomx-hexapod/68-phoenix-code.html`

HexapodMKIICommander-master contains the code delivered with the hexapod

PosePrograms contains a program for recording and playing back different poses for the hexapod. More information can be found at `http://learn.trossenrobotics.com/8-arbotix/131-dynapose-dynamixel-arbotix-pose-tool.html`

VirtualCommander A program that emulates the remote to the hexapod. More information can be found at `http://learn.trossenrobotics.com/projects/145-virtual-commander-generic-software-robot-control.html`.