Humanoid Robot Control Architecture

Pedro Teodoro, Mário Marques, Jorge Martins, Carlos Cardeira, Miguel Ayala-Botto, Limor Schweitzer, José Sá da Costa

***Abstract*— In this paper we present the current development status at the Humanoids Robotics Lab of the Department of Mechanical Engineering at Instituto Superior Técnico in collaboration with Robosavvy Ltd. The developments we present include the software development for interfacing the Matlab real time workshop toolbox with the humanoid robot controllers, hardware development towards wireless communication between the local robot controller and the remote PC, the identification of the internal and external dynamic parameter of the humanoid servos and structure respectively, the dynamics modeling and simulation using simMechanics and virtual reality toolbox. Our aim is the development of a humanoid robot able to make complex motions like walking, running and jumping.**

I. Introduction

Current commercially available humanoid robots are designed to perform motions using open-loop control providing the users a simple paradigm to create pre-orchestrate multi-DOF walking gaits. These robots are usually not able to move on uneven terrain and it is difficult or impossible to get them to perform movements that require instantaneous reaction to momentary instability. A popular way to compensate for these predicaments is to over-capacitate servo torques and to incorporate large foot soles, low center-of-mass and better shock absorption, resulting in humanoid robots with little resemblance to the human physique. Our long term objectives are to allow affordable humanoid robots to run, skateboard, jump and in general to react in a human-like physical way in dynamically unstable situations and uneven terrain.

We would like to achieve these goals by applying closed-loop control techniques to the humanoid robot servos. The input data stream should consist of multitude of sensors including servo position and torque, acceleration and inertial moment. The closed-loop control cycle should actuate the servos at rates of at least 50Hz which would give good responsiveness in a dynamic environment. Our development environment includes Simulink (tm) running on a Windows PC. Simulink can compile and offload control algorithms to various real-time hardware systems. We wanted the control loop to be able to run both on-board the humanoid and also on the PC. In both scenarios the sensory data and servo actuation commands should be streamed back and forth with the humanoid servos.

Given the above requirements, we looked for a humanoid robot for our development environment. We finally selected the Bioloid from Korean manufacturer Robotis.com. This was our humanoid kit of choice due to its well designed servo controllers that provide current, voltage, position and temperature sensing, well documented open controller board and its well documented servo control protocol.

Other humanoids platforms we considered included: (a) KHR1HV / KHR1HV / Manoi / Robonova - These are affordable humanoid kits whose servos provide position and current sensing. Their weight to torque ratio are probably better suited for running and skateboarding than Bioloid.

However, documentation was lacking at the time we evaluated this option. (b) Custom Humanoid - many of the RoboCup teams and robot researchers build their own humanoid model using Aluminum and Fibreglass brackets and high-torque RC servos. A popular choice is the Robotis high-power RX and DX servos to actuate the robot. However, due to the limited time and human resources, we decided to go with a kit.

II. Architecture and Developments

A. Hardware Architecture

Figure 1 shows the existing humanoid architecture and the control architecture we are using.

The humanoid controller named CM5 is connected to the controllers of the servos through a RS485 bus. The usual approach to teach the robot is the use of the humanoid proprietary software that connects to a PC through the RS232 serial line.

The CM5 has however an Atmega128 microcontroller with a bootloader which allows users to change the code and access directly to the servo controllers parameters.

Pedro Teodoro, Mário Marques, Jorge Martins, Carlos Cardeira, Miguel Ayala-Botto, José Sá da Costa are with the IDMEC lab at Instituto Superior Técnico, Technical University of Lisbon, Avenida Rovisco Pais, 1049-001 Lisboa, Portugal, {martins@dem., carlos.cardeira@, ayalabotto@, sadacosta@dem.} ist.utl.pt.

Limor Schweitzer, is with RoboSavvy Ltd, 37 Broadhust Gardens, London, United Kingdom, limor@robosavvy.com.

We developed a small program in the CM5 controller to implement a protocol for transmitting/receiving thought the serial port the data from/to the servos.

B. *Software Architecture*

In the computer running Simulink / RTW / Real Time Windows Target we developed the device drivers to send/receive data to/from the servos using the protocol we defined. For doing this, we created a C-MEX S-function written in C to communicate with the CM-5 throughout UART (universal asynchronous receiver / transmitter). Hereafter, it was necessary to establish a protocol for the serial communication between PC and the CM-5. Finally, we wrote a little C program for Atmega128 for completing the serial communication bridge. Hence we implemented this architecture that transparently maps Simulink variables into servos motion.

We have now a way to identify the parameters of the Humanoid models making online experiments, as MATLAB/SIMULINK® is a unique tool, widely used, for System Identification and control. However we still have to check if the delays introduced by the serial line, CM5 and RS485 are compatible with the sampling time of the controller.

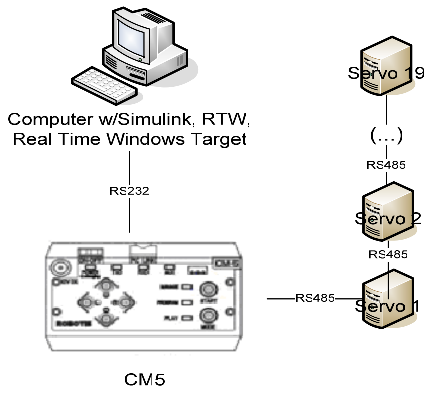

Figure 1 Humanoid Control Architecture

C. *Real Time Issues and Communication delays*

It was built up a program in C++ to measure all the current positions of the servos and therefore to verify the real frequency in a real application. In this last case, we used 19 real servos and one artificially for resynchronization.

The servo position is a 10 bit value that has to be divided in a high and low bit in order to be sent throughout the RS232 to the PC. Hence, the theoretical value, without latency, is:

$$f = \frac{57600}{(19+1)x2x10} = 144\text{Hz},$$

Where:

- 57600bps is Baud Rate
- 19+1 means 19 servos plus 1 artificially for resynchronization.
- 2 stands for 2 byte (the encoder number of each servo is 10 bits of resolution)
- 10 is the number of bits (8bits for the data +1 parity for +1 stop bit)

The measured frequency was not far away from the theoretical value, being just 0.6Hz bellow, 143.4 Hz. This means that the Latency is responsibly for just ~0.42% delay for all the process. Testing the latency between an order given by PC, read it and send it back to the PC by the CM5 is around 1.29ms, just shown after in a running test of 1000 samples.

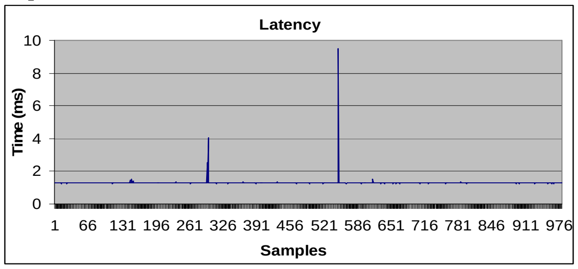

Figure 2: Communication Delays

Thus, it was decided to use a sample rate of 0.01 seconds (100 Hz) for all future tests.

III. System Identification and Control

A. *Description of the measured signals from the servos*

A set of output signals can be retrieved from the Humanoid Robot servos. These signals provide information regarding the actual servos angular position, angular velocity, D/C current, temperature and voltage. The angular position, temperature and voltage signals are sampled at 100 Hz, while the angular velocity and load are sampled at a rate 10 times slower.

B. *Close loop position control*

By default the servos are configured for position control. In fact, all servos have an internal feedback position control loop. This characteristic can be easily confirmed by the simple experiment shown in Figure 3. The servo tries to follow the desired time varying sinusoidal reference position by changing its actual D/C current (load) charge through time, even in the presence of an external torque applied at time instant t=6 sec.

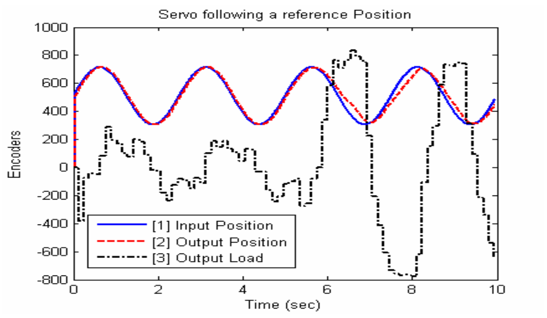

Figure 3: Servo following a position (close loop control)

C. *Open loop velocity control*

In contrast, experiments suggest that the servos do not have internally any angular velocity feedback control. This can be experimentally confirmed by applying an external torque to the servo while in constant rotating velocity. From Figure 4 it can be concluded that the current consumption is not able to respond accordingly after the fifth second, when a torque is applied, so the reference angular position cannot be followed.

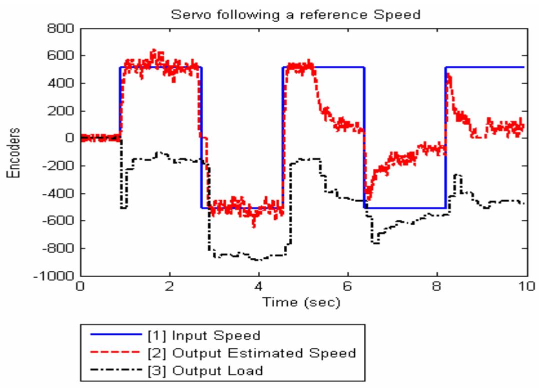

Figure 4: Servo following a reference velocity with open loop velocity control

D. *Stiction*

Stiction is a physical phenomenon that is present in almost any system with moving components. Therefore, its characterization is essential for obtaining an accurate dynamic model of the servos. A simple way to quantify stiction can be made through the following experiment: starting with the servo rotating at a constant speed in one direction, progressively slowing it down until it stops, and then slowly increase its rotating speed in the opposite direction. With this experiment it should be possible to identify the typical dead-zone effect due to stiction. In our case this was clearly quantified to be around 7-10% of the full range in case no load is applied to the servo, as can be seen from Figure 5.

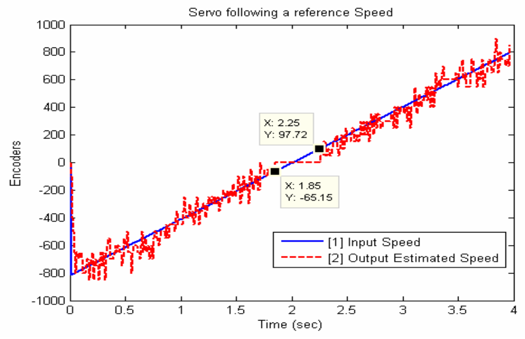

Figure 5: Stiction dead-zone

E. *Voltage supply*

Another parameter with relevance to the behavior of the system is the voltage supplied to the servos. Experiments show that the output estimated velocity error is proportional to the voltage supplied to the servo. In fact, good output velocity estimation is achieved only if the battery is charged around 10 V, as can be seen from Figure 6.

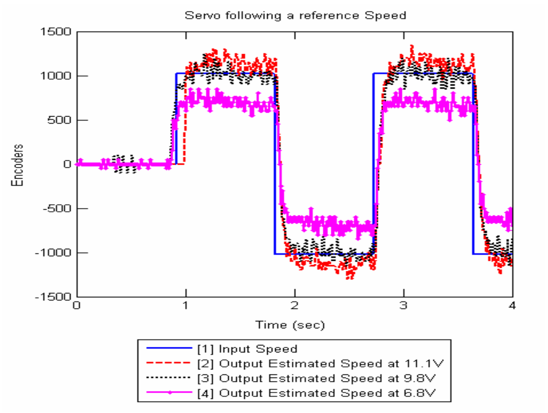

Figure 6: Effects of the supplied voltage to the servos in the outputs velocity response

F. *Humanoid identification*

In order to capture the static and the dynamic properties of the Humanoid Robot, both mechanical properties of all its components, such as mass and inertia, as well as its servos dynamic responses, must be known to a certain degree of accuracy. These dynamic properties will be used in Simulink and simMechanics in order to get an accurate simulator for the real Humanoid Robot aiming at a good control strategy.

1) *Mechanical properties identification*

An accurate static model of the Humanoid Robot can be obtained based on the physical properties of their components. Typically, by knowing the mass, center of mass and the inertia tensor of each element of the HR it is possible to get a quite reliable model that can be further used in simulation and control. For quantifying the masses of each element, a precision scale with a resolution of 0.05 grams was used. The centroid of each mass was then found by using the SolidWorks® software package, after the detailed elements of all the pieces involved were drawn in this 3D CAD software. It was assumed here that, except for the servos, all the pieces are of isotropic nature. A simple experiment has shown that the maximum error obtained for the geometric position of the centroid is of 0.5 mm on each Cartesian direction. Finally, the inertia tensor of each element was determined through the SolidWorks® software.

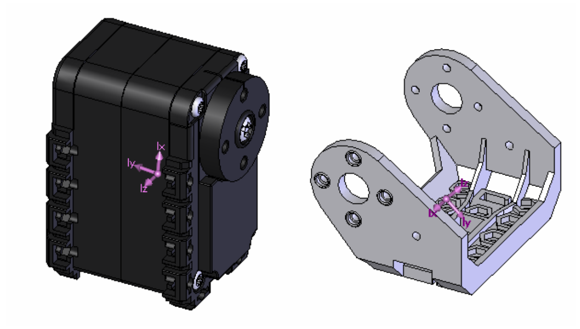

Figure 7: 3D models of a servo and a component of the Humanoide robot showing the centroid and the principal axes of the moment of inertia tensor.

2) *Dynamic properties identification*

For the identification of the dynamic behavior of the servos it was considered the relation between the reference input velocity and the correspondent estimated velocity obtained through the following equation:

$$\hat{\dot{\theta}}(t) = \frac{\theta(t) - \theta(t-1)}{Ts} \qquad (1)$$

Where

$\hat{\dot{\theta}}(t)$ is the estimated velocity at time instant t,

$\theta(t)$ is the angular position at time instant t

$\theta(t-1)$ is the angular position in the previous time instant

Ts=0.01 sec. is the sampling period.

The classical prediction error method was used for the identification of the servo dynamic model [3], using the identification data shown in Figure 8.

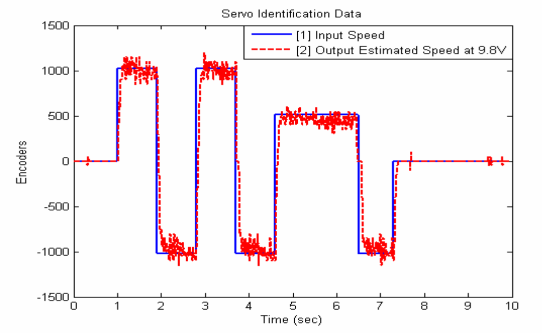

Figure 8: Servo Identification Data

After testing several tentative models with different orders, a BJ(2,1,2,1) was found to best approximate the desired dynamical behavior of the servo. The BJ model that results in the best data fit is the following:

$$TF = \frac{0.06217z}{z^2 - 1.469z + 0.5544} \qquad (2)$$

Figure 9 compares the real output of the servo with the one estimated by the BJ model for the validation data. It can be concluded that the dynamic characteristics of the servo are well captured by the BJ model.

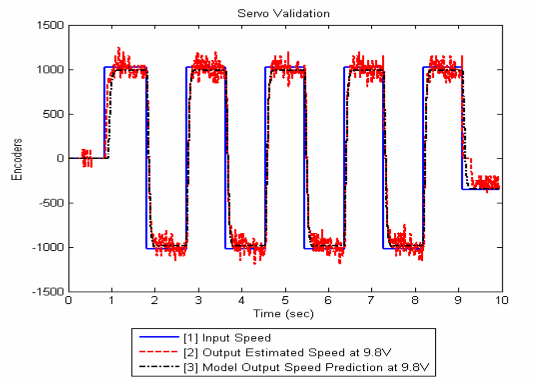

Figure 9: Servo Validation Data

IV. Virtual Reality Animation

Two models of the Humanoid Robot were built in Solidworks. While the first model was used for system identification in section III, the second model was exported to the wrml format to be used in the virtual reality toolbox of Matlab. In the second model, contrarily to the first one, the pieces were drawn with precise measurements but with few details in order to get a lighter animation running in real time.

The purpose of the virtual reality model using the Virtual Reality Matlab toolbox is twofold. First, and as described in chapter V, it will be used alongside with simMechanichs, a mechanical simulation platform of Matlab, in order to have a visualization of the Humanoid dynamics model. The second purpose, as already used, is to have an animation doing in real time exactly what its twin is doing in reality.

Two versions of the model have been built. The first one is just the normal Humanoid used for walking motion and the second one is Humanoid hanging from a Gymnastics high-bar.

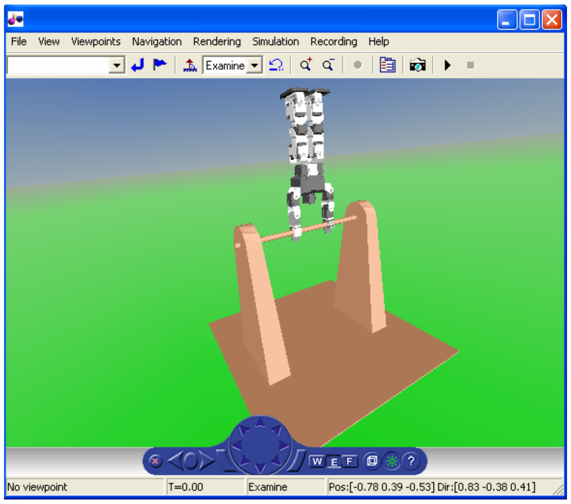
File View Viewpoints Navigation Rendering Simulation Recording Help
Examine
No viewpoint
T=0.00
Examine
Pos:[-0.78 0.39 -0.53] Dir:[0.83 -0.38 0.41]

Figure 10: Virtual Reality of the Humanoid doing a handstand on a high-bar.

V. Ongoing Work

A. New hardware architecture

Due to the tasks involved in this project the CM5 controller board contained in the initial configuration of the humanoid robot, will not be capable of undertaking all the computation required to the desired control functions. This CM5 controller board is based in an Atmel ATMega 128 microcontroller, with a 16MHz processor and 128Kb of flash memory. For this project we choose a new board from Gumstix Inc. this board has a 400MHz processor with 64Mb RAM and 16Mb of flash memory. Figure 11 shows the new architecture.

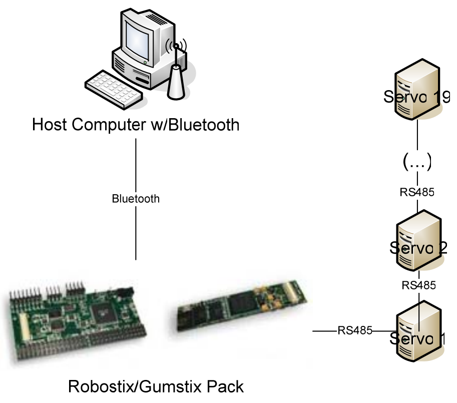

Figure 11: New hardware Architecture

Gumstix Connex runs an uClib Linux, in an ARM architecture processor. For data acquisition it is used an expansion board, Robostix, with the same base processor as the CM5 controller board. The communication between Gumstix and the Robostix is made through i2c.

The evolution in this new setup is that with the Gumstix the humanoid robot gains more processing power, as the low-level control functions are made in the humanoid robot, and not in an external computer, giving it more computation autonomy.
Therefore, with this setup, the humanoid robot can manage all the servos control (through RS485), and just release to the upper level computer the necessary data to the high-level control.

B. Wireless communication

One of the main improvements that had to be made was the way to connect the computer to the humanoid robot.
The physical connection through serial port goes against one of the philosophies on this project, that is, to create a more autonomous humanoid robot. Hence we used a Bluetooth adaptor for remote communication with the robot. For this it was made a TCP/IP connection with static IP addresses, with the computer acting as Host, and the Gumstix board as Client. The connection as already been tested in terms of throughput and latency, and revealed satisfactory results for the purpose of this connection, that is to make the interface between the high and low level control tasks.
For the latency tests the results obtained were, as shown in Figure 12.

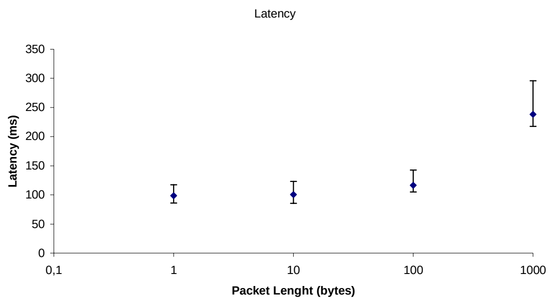

Figure 12: Wireless Communication latency

The tests were made by sending 200 packets of 1, 10, 100 and 1000 bytes, plus the 28 bytes of the beginning and the end of the TCP connection.
The throughput test has given the average result of 11,000 bytes/sec. These values vary with the distance between Host and Client, and with the interferences in the medium.
As we can conclude the wireless loop implements a loosely coupled connection [2] that will not be able to run close loop control of tasks involving small sampling periods. The fast control loops involving stability will autonomously run inside Gumstix which has a tightly coupled connection to the servos. The remote computer will give directions and targets to the humanoid but all the control and stability will run autonomously inside the robot.

C. *New Software Architecture*

In order to take full advantage of the new, more powerful, Robostix/Gumstix hardware architecture within the Matlab/Simulink environment, a custom target must be built for Real-Time Workshop. To this end, we start from the Linux Soft Real-Time Target in [4], and adapt it to the uClib Linux kernel of the Gumstix.

The target uses the POSIX real-time clocks to generate periodic signals to wake up the model process at every time step. Furthermore, the process is changed to real-time highest priority defined by the scheduler. The outcome is a soft real-time architecture rather than a hard real-time architecture since the Linux kernel itself is not preempted by the scheduler, and therefore the model execution can occasionally be delayed. The delays involved in this approach will be further analyzed, and compared against the sampling times used for humanoid control.

The resulting seamless working environment linking all the Maltab/Simulink tools and the (almost) real-time controller implementation in Gumstix renders this approach to be ideal for the objectives of humanoid control.

D. *Multibody Dynamics*

The process of modeling the multibody structure of a humanoid robot, either for the purpose of doing a hand-stand on a high bar or for generating a stable walking motion, is a very complex one. One must deal with the formulation and solution of highly nonlinear dynamics equations of a very large size; a standard humanoid has 18 servos. Furthermore, the problems imposed by transient mechanical constraints and by impact forces must be dealt with, and stabilizing controllers must be developed.

To aid in this development, and following in the line of thought of a seamless development tool within Matlab/Simulink, we use the SimMechanics toolbox. Within SimMechanics, modeling and simulation of the humanoid may be easily performed without getting into the analytical and numerical complexity of model formulation [5]. Furthermore, model linearization, reduction and automatic code generation for the Linux Soft Real-Time Target are straightforward to implement. During the whole process, one may also interface the outputs SimMechanics into the Virtual Reality Toolbox as shown in Section IV and visualize the behavior of the humanoid robot.

E. *Control objectives: High-Bar Hand-Stand and Walking Motion*

The two control objectives under development are intended to explore linear and non-linear control methodologies. For the high-bar hand-stand, the humanoid is treated as a two or three body serial chain in an inverted pendulum configuration. The system is under actuated, being the motion of the legs prescribed in order to stabilize the torso of the humanoid above the high bar. Linear control methodologies will be exploited in this problem for both the swing-up phase and equilibrium phase.

The problem of stabilizing a walking motion is much more complex than that of stabilizing a hand-stand on a high-bar. At each step there are impact forces and transient mechanical constraints, and the high model size reduction of the former case may no longer be performed. Thus, non-linear control approaches must be explored.

A stable walking gait controller is the basis for more complex motions such as running and jumping, and this constitutes the main path for our current and future work.

VI. Conclusions

In this paper we presented the current development status at the Humanoids Robotics Lab of the Department of Mechanical Engineering at Instituto Superior Técnico in collaboration with Robosavvy Ltd.
For the last semester we have worked on:

- Software Development for interfacing the Matlab real time workshop toolbox with the humanoid robot controllers.
- Hardware Development towards wireless communication between the local robot controller and the remote PC.
- Internal and external dynamic parameter identification of the humanoid servos and structure respectively.
- Dynamics modeling and simulation using simMechanics and virtual reality toolbox

The previous points aim at the development of:

  - Linear control methodologies for the swing-up phase and equilibrium phase of a hand stand on a high bar.
  - Non linear control methodologies for a stable walking gait for more complex motions such as running and jumping.

References

[1] Qiang Huang and Yoshihiko Nakamura. Sensory Reflex Control for Humanoid Walking, IEEE Transactions on Robotics, Vol. 21, No. 5, October 2005, pp. 977-984

[2] C. Cardeira, A. W. Colombo, R. Schoop, “Wireless solutions for automation requirements”, in ATP International – Automation Technology in Practice, IFAC-affiliated journal, Vol. 2, September 2006, pp 51-58.

[3] Lennart Ljung, System Identification: theory for the user, Prentice-Hall, 1987

[4] Bhanderi, D., “Linux Soft Real-Time Target V2.2”, http://www.control.auc.dk/~danji/downloads/, March 2007.

[5] Ledin, J., Dickens, M. and J. Sharp, "Single Modeling Environment for Constructing High-Fidelity Plant and Controller Models", AIAA Modeling and Simulation Technologies Conference and Exhibit, August 2003, Austin, Texas, USA.