

# REAL-TIME REMOTE NETWORK CONTROL OF AN INVERTED PENDULUM USING ST-RTL

R. Murillo Garcia<sup>1</sup>, F. Wornle<sup>1</sup>, B. G. Stewart<sup>1</sup>, D. K. Harrison<sup>1</sup>

**Abstract** - This paper describes the use of Simulink Target for Real-Time Linux (ST-RTL) to control over a network a horizontal driven inverted pendulum. ST-RTL is an application developed to provide a cost-effective alternative to the expensive real-time applications that require specialised software and hardware. The application allows performing a number of laboratory scale hard real-time control experiments, which can be remotely monitored and tuned at run-time from a second computer using a TCP/IP connection. The software packages Matlab, Simulink and Real-Time Workshop (RTW), widely used among educational institutions, are used as a controller algorithm development tool. A performance comparison with the commercial Windows based version Real-Time Windows Target has also been carried out. The conclusions will show that ST-RTL is a preferable tool since not only does it perform as well as RTWT, but it also includes remote networking capabilities to the system.

**Index Terms** - Inverted pendulum, Real-time Control, RT-Linux, Real-Time Workshop, Simulink.

## 1 INTRODUCTION

Real-time systems are those that can support the execution of applications with time constraints. There are two classifications of real-time systems based on their properties: soft and hard real-time systems. In hard real-time systems lateness is not accepted. If a deadline is not met, the controlled device may suffer a failure. On the other hand, soft real-time systems are less demanding, allowing for a grade of lateness. Although the performance suffers, the controlled device is not affected severely nor is destroyed when the system does not meet certain response time constraints. The interest of this investigation and experiment is focused on the former type of control.

Hard real-time control requires a platform that guarantees certain time constraints. Mainstream operating systems such as Windows 95, 98 and NT use protective mechanisms that make them incapable of hard real-time processing. Thus, hard real-time control experiments often require expensive equipment that may be beyond the financial capabilities of educational institutions such as small universities and schools.

Simulink Target for Real-Time Linux (ST-RTL) presents an alternative to the expensive and specialized software and hardware required for hard real-time control. This application enhances RTW by adding real-time features that allow the controller algorithm to run deterministically under RT-Linux. Since RT-Linux (<http://www.rtlinux.org>) and ST-RTL (<http://www.fst.gcal.ac.uk/esd/raulm/ST-RTL.htm>) can be downloaded free of charge off the Internet, an inexpensive hard real-time control can therefore be implemented. In addition ST-RTL offers the capability of remote monitoring and control using a client-server model. The target machine (RT-Linux), where the controller algorithm is executed, and the host machine (Simulink) from where the system can be monitored using the Simulink block diagram, exchange data through a TCP/IP connection.

Generally, nowadays, it is important to demonstrate the ability of networked controlled real-time applications. Thus, the importance of understanding and developing real-time techniques using familiar software platforms such as Simulink and Matlab can be extremely beneficial to the students learning and application experience.

This paper presents an example of control using ST-RTL and a comparison with RTWT in order to show the viability, the advantages and the performance of ST-RTL for real-time networked applications. ST-RTL has been successfully used in the past at the Control System laboratory of Glasgow Caledonian University. It was possible to control the velocity and position of DC-motors and the level and flow of liquid in a coupled-tank. Thus, students can take advantage of this tool to implement networked real-time control of classic control experiments. The work presented here will not focus on the dynamics of the pendulum nor will it propose to implement an ideal controller. Rather, this paper is intended to prove the networked viability, usability and performance for educational purposes of ST-RTL for the control of experiments such as the inverted pendulum.

This paper is divided into three sections, which are as follows: a brief description of the software used to implement ST-RTL, a brief description of the dynamic system of the inverted pendulum and the associated experimental set up, and finally, a general comparison between results obtained using RTWT and ST-RTL.

<sup>1</sup> School of Engineering Science and Design, Glasgow Caledonian University, G4 0BA Glasgow, Scotland, UK.

## 2 SOFTWARE DESCRIPTION

Simulink [1] is an easy-to-use block diagram based software package for use with Matlab for modelling, simulating, and analyzing dynamic systems. Scope blocks allow the monitoring of the system by displaying signal values in the time domain on a unique window, which resembles an oscilloscope. Real-Time Workshop (RTW) [2] translates the block diagram representation into platform-invariant ANSI C code files for a specific platform (target). ST-RTL has been developed to target the real-time operating system RT-Linux, thereby offering hard real-time control.

RT-Linux [3] is a real-time extension of the Linux kernel. By adding a new layer of interrupt handling between the original Linux kernel and the interrupt controller the Linux kernel is made pre-emptive. Since Linux is given the lowest priority, a Linux process only runs when no real-time task is scheduled. If a RT process is currently running the dispatching of an interrupt to Linux is delayed until the RT process is completed. RT-Linux has been successfully used with many applications [4] [5].

ST-RTL is a modification of the RTW target Generic Real-Time Target used for Unix based computers to adapt it to the RT-Linux environment. Following is a brief outline of the implementation of this application. A real-time process runs as a module in the kernel space, thus having access to the real-time facilities provided by the RT-Linux kernel. A real-time thread is created within this process and executed periodically. When this thread is invoked, it runs a controller algorithm cycle. In order to overcome the real-time kernel lack of network support (TCP/IP), a second process running in user space has been created to allow communication between the host and the target machines. Both processes exchange data using two real-time FIFOs and a block of shared memory. The FIFOs are used for exchanging commands whenever an exception occurs in either process and an action needs to be carried out. On the other hand, a ring buffer has been implemented in shared memory. Whilst the real-time process running in the kernel writes signal values into the shared memory ring buffer, the network process running in user space reads these values and uploads them to the host machine over the network using a TCP/IP connection. In turn, the host machine (Simulink) displays the signal values received through the network on the correspondent Scope block. The CIO-DAS08/JR-AO data acquisition card has been used with this experiment to access external inputs and outputs. To make this card available to ST-RTL a driver has been written and two Simulink blocks created to represent the inputs and outputs within a block diagram. The CIODAS08-JR/AO has been protected against overvoltages and short circuits by a protection board with opto-isolators and buffers.

RTWT [6] has also been used to control the inverted pendulum to carry a performance comparison with RT-Linux Target. RTWT ensures a real-time application to run in real time by installing a small real-time kernel. The real-

time kernel runs at CPU ring zero (privileged or kernel mode) and uses the built-in PC clock as its primary source of timing. It intercepts the interrupts from the PC clock and uses them to trigger the execution of the real-time program. By blocking any calls to the Windows operating system the kernel gives the highest priority to the real-time application. If an interruption triggers a non real-time task, the real-time kernel passes the control over to Windows, which will then handle the task. Shared memory is used to upload data from the real-time process to the Simulink application. RTWT and ST-RTL share certain similarities. However, in the former case host and target machine are the same, whereas in the latter case host and target are different machines.

## 3 INVERTED PENDULUM CONTROL

The inverted pendulum has been chosen for this experiment because it is a well documented classic example of control [7, 8] commonly used in the teaching of control systems engineering. The laboratory for which ST-RTL has been developed disposes of the commercial Bytronic Pendulum Control System (PCS1), described in Section 4. This system is inherently unstable since the pendulum fails to remain in the upright position when no action is applied. Neglecting the dynamics of the carriage servo, i. e. assuming that its response time is very fast in comparison with the pendulum, the inverted pendulum can be modelled by the following transfer function given by (1):

$$G(s) = \frac{1}{1 - \frac{s^2}{\omega_n^2}} \quad (1)$$

where  $\omega_n$  is the natural frequency, which can be found by swinging the non-inverted pendulum and recording the angle oscillation. The period for the experimental pendulum in question is found to be approximately 0.97 seconds, which is a natural frequency of 6.48 rad/sec. Thus, the transfer function of the system is given by (2).

$$G(s) = \frac{1}{1 - \frac{s^2}{6.48^2}} = \frac{1}{1 - 0.0238 s^2} \quad (2)$$

Figure 1 (a) shows the pendulum-controller system dynamics using the Matlab root locus tool and Figure 1 (b) the corresponding step response. As expected, this system is unstable due to the positive closed-loop pole at +9.167.

To stabilize the system a lead compensator with a gain of -1.3 has been used as a controller. Lead compensators introduce phase advance, thus improving the stability margin without reducing the gain-crossover frequency and speed of response [6]. Equation (3) represents the transfer function of the lead compensator. Figure 2 (a) and Figure 2 (b) show the root locus and a unit step response of the compensated

system. The unstable closed-loop pole has migrated from the positive to the negative half of the complex  $s$ -plane, giving a stable system.

$$G_c(s) = \frac{0.1 \cdot s + 1}{0.03 \cdot s + 1} \quad (3)$$

#### 4 EXPERIMENT SET UP

The inverted pendulum used is the commercial Bytronic Pendulum Control System (PCS1), which consists of a carriage module and a control unit. The former includes a cart with a pivoted rod and mass, which can be driven along a 500 mm track by a DC servomotor with integral tachometer and a measurement system (potentiometers) for determining the cart position and attitude of the rod/mass assembly. The latter provides a position control loop with adjustable gain and velocity feedback for the carriage and the analogue lead compensator with adjustable gain, pole and zero. The position of the cart depends on the voltage applied to the dc servomotor. Thus, 0 V corresponds to the center and  $\pm 10$  V correspond to both ends of the track.

A PC (133 MHz, 128 MB RAM) running RT-Linux installed is used as the target workstation. The host workstation is a second PC (400 MHz, 32 MB RAM) running Matlab, Simulink and RTW.

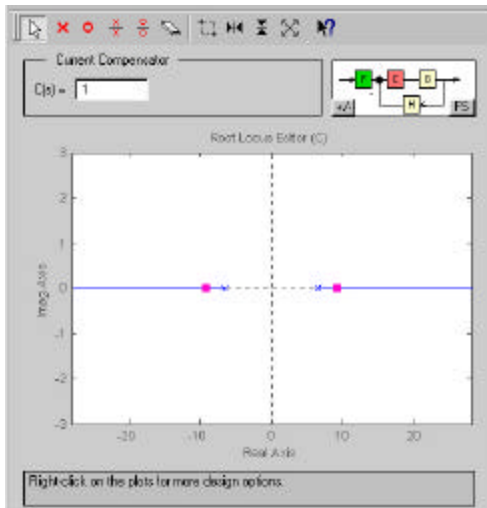
The digital controller has been implemented as shown in the Simulink block diagram in Figure 3. The *Feedback* block provides the readings from a selectable analogue input channel of the data acquisition card. This input, filtered to reduce noise (*Filter* block), is the feedback control unit consisting of the weighted sum of the cart position and the angle of the pendulum with a gain of  $-1.3$ . The lead compensator transfer function (*Controller Transfer Function* block) in the forward branch form the controller itself. Finally, the *output* block provides the interface to the analogue output of the data acquisition card, which is connected to the signal input of the control unit through the protection board to drive the cart. The Feedback signal and the Output signal are monitored on the *Scope* block. The sampling rate specified for this block diagram is 400 Hz, specified in the Simulink block diagram parameter "Fixed Step Size", under the *Simulation@Parameters* menu. The voltages of the data acquisition card have a range of  $\pm 5$ V. As the control unit voltage has a range of  $\pm 10$  V, an external amplifier with a gain of 2 is required. Furthermore, the protection board divides all analogue input voltages by a factor four and the driver normalises the converted voltages to  $\pm 1$ . A total compensator factor of 20 ( $4 \cdot 5$ ) is therefore applied to the inputs. The output normalisation factor  $1/10$  reduces the signal range to  $\pm 1$ , as required by the driver block *output*.

After designing the block diagram the automatic build process of RTW is executed with ST-RTL as target. A set of model dependent ANSI C source files is generated from the block diagram. These files are transferred to the RT-Linux

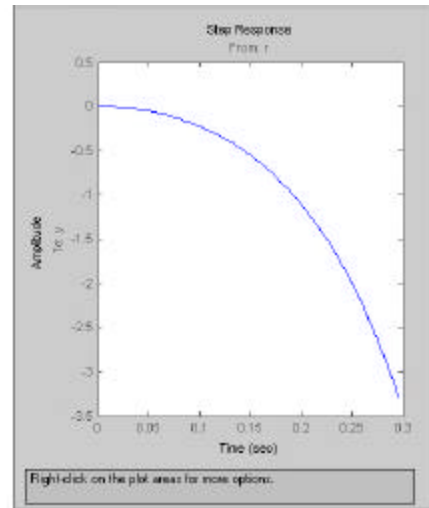
machine, where they are compiled and linked with a number of model independent source files to generate the user space process and the kernel process. After inserting the modules in the kernel the user space process is launched; the model is now running and the pendulum is controlled by the target machine. The model independent source files are a set of ANSI C source files that do not vary if the model is modified and provide the engine for running the model, data logging and networking.

To monitor the signals of the system, the Simulink block diagram on the host machine can be connected to the running model on the target machine, which starts uploading signal values. The feedback and output system signals can therefore be observed on the scope block of the host. Furthermore, all target model parameters can be modified at run-time. Figures 4 and 5 show the results obtained when the system is started and following a small disturbance due to a slight push applied to the pendulum, respectively.

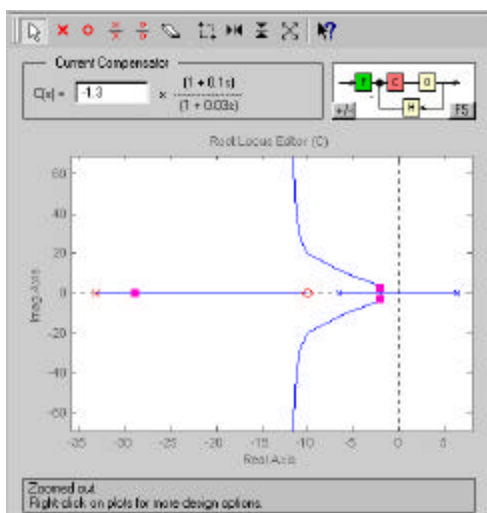
To provide a performance comparison with a similar commercial product, MathWorks' Real-Time Windows Target (RTWT) has also been targeted. The process involves fewer steps. The input and output blocks have been replaced for those specified by RTWT for the data acquisition card CIO-DAS08/JR-AO. As before, the RTW build process is executed to create the model dependent source files. Since host and target are actually the same machine, the model is generated automatically and compiled with the correspondent source files to create an executable file. The controller is started when Simulink connects to the model. At this point, the pendulum start being controlled and the system signals displayed on the scope block. Figure 6 depicts the signals observed when the control process is launched; Figure 7 shows the signals following a small disturbance. Figures 4, 5, 6 and 7 show that the pendulum stabilizes after a short transient. Both output signal and feedback signal are displayed.



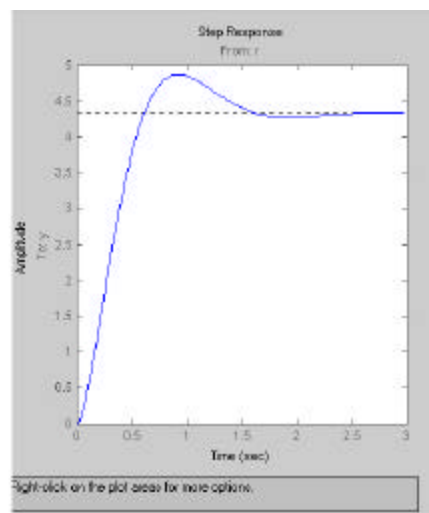
**FIGURE 1 (A)**  
 ROOT LOCUS OF THE SYSTEM. THE POSITIVE CLOSED-LOOP POLE (SMALL SQUARE, POSITIVE REAL AXIS) DENOTES THE SYSTEM INSTABILITY .



**FIGURE 1 (B)**  
 UNIT STEP RESPONSE OF THE UNCOMPENSATED SYSTEM (INSTABILITY).



**FIGURE 2 (A)**  
 ROOT LOCUS OF THE COMPENSATED SYSTEM. ALL CLOSED-LOOP POLES (SMALL SQUARES) ARE NOW AT THE LEFT OF THE IMAGINARY AXIS, RESULTING IN A STABLE SYSTEM



**FIGURE 2 (B)**  
 UNIT STEP RESPONSE OF THE COMPENSATED, WHICH IS NOW STABLE.

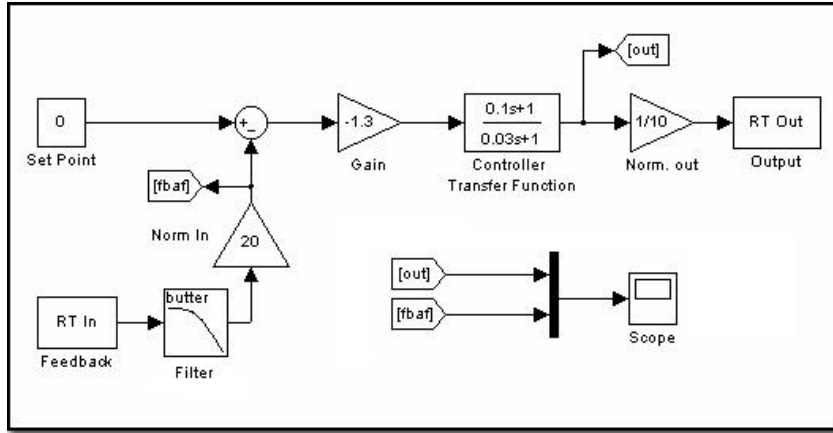


FIGURE 3  
SIMULINK BLOCK DIAGRAM USED TO CONTROL THE PENDULUM.

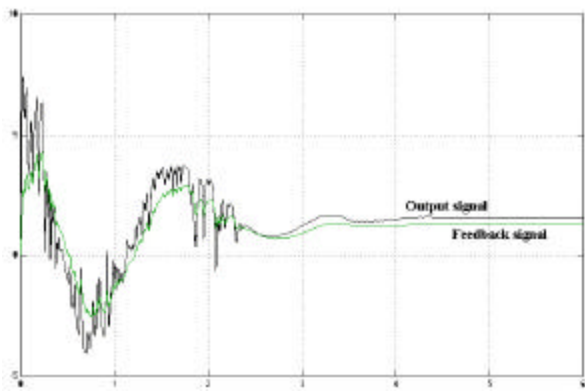


FIGURE 4  
SCOPE DISPLAYING THE OUTPUT SIGNAL AND THE FEEDBACK SIGNAL WHEN THE MODEL IS STARTED USING ST-RTL.

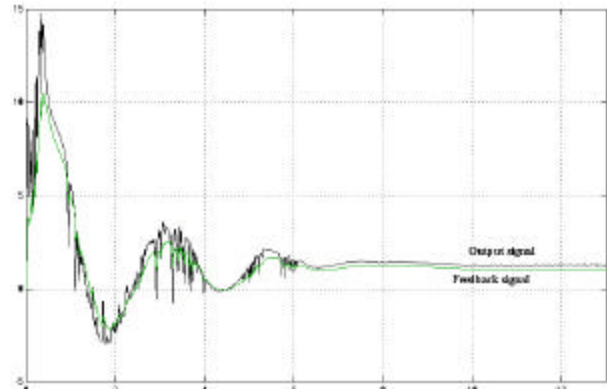


FIGURE 5  
SCOPE DISPLAYING THE OUTPUT SIGNAL AND THE FEEDBACK SIGNAL WHEN THE MODEL IS STARTED USING RTWT.

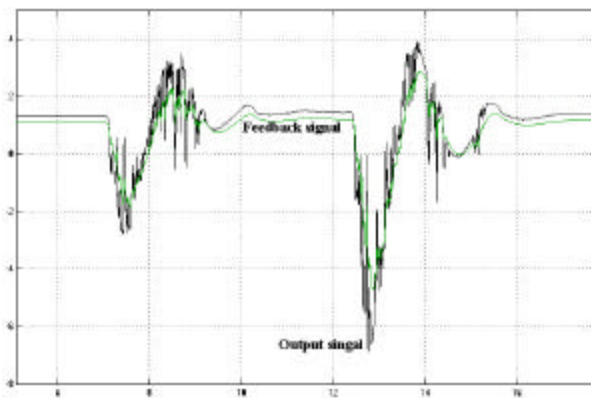


FIGURE 6  
SCOPE DISPLAYING THE OUTPUT SIGNAL AND THE FEEDBACK SIGNAL WHEN THE MODEL IS DISTURBED USING ST-RTL.

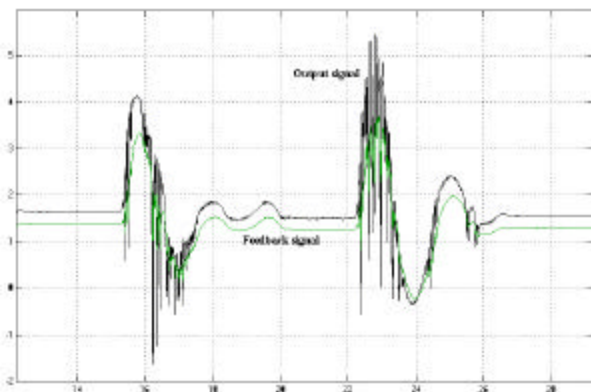


FIGURE 7  
SCOPE DISPLAYING THE OUTPUT SIGNAL AND THE FEEDBACK SIGNAL WHEN THE MODEL IS DISTURBED USING RTWT.

## 5 RESULTS

RT-Linux Target was found to perform successfully throughout all tests. The system can be remotely started, disconnected (without the model being stopped), terminated and monitored using the “External Mode Control Panel” under the Simulink menu “Tools”. In addition, the block diagram parameters can be tuned “on-the-fly”. This option is of great use as it allows a direct visualisation of parameter changes such as the gain and time constants of a Proportionate-Integral-Derivative (PID) controller. For example, when the gain of the lead compensator is increased beyond a critical value, the pendulum enters an unstable state. As expected, all these actions also work successfully with RTWT.

A summary of the similarities, major differences and advantages of ST-RTL over against RTWT may be outlined as follows:

- The use of ST-RTL and RTWT both yield successful results in the inverted pendulum control. This result was expected, as the code that represents the block diagram is platform-invariant, that is the same controller algorithm code has been used in both applications.
- For this particular experiment RTWT allows for a maximum sample rate of 8,000 Hz, whilst RT-Linux Target achieves sample rates of up to 5,000 Hz. The difference is due to the network processing needed in the latter application. Many laboratory scale experiments, like the inverted pendulum, have dynamics of a relatively low bandwidth and therefore do not require very high sample rates. However, higher sample rates can be easily achieved with a faster target machine. Note that most off-the-shelf PCs are now significantly more powerful than the ones used in our experiments. The bottleneck for the maximum useful sample rate is then given by the speed with which the analog to digital conversions can be carried out in the data acquisition card. This is independent of the application used.
- In RTWT the host and the target are the same machine and therefore remote monitoring is not available. This is the main disadvantage of RTWT. ST-RTL overcomes this problem by using a network connection between target and host workstations through which data can flow. Thus, remote plant observation and control can be carried out.
- ST-RTL is shown to provide remote networked control of a normally difficult to control mechanical dynamic system, thus showing its power and applicability to real-life networked control systems.
- In both RTWT and ST-RTL applications a delay in the real-time monitoring is observed as the sample rate is increased. The reason lies in more signal values being written into the shared memory than signal

values being read. In RT-Linux Target, however, the delay can be larger if the network traffic is high. Signal values cannot be sent to the host machine at the same rate as they are generated and are therefore queued in the ring buffer shared memory.

## 6 CONCLUSION

This paper has presented a case study which focuses on the real-time control of a horizontally driven inverted pendulum using ST-RTL. It has successfully been shown that ST-RTL can be used to implement remotely controlled applications, using Matlab and Simulink software environment. It performs as well as the commercial version RTWT, but with the added bonus of remote monitoring and tuning of the experiment. ST-RTL can be downloaded free of charge off the Internet. All this allows researchers and students to put into practice in a simple and cost effective way, remotely controlled real-time applications, making it possible to experiment with the theory as studied in class.

## REFERENCES

- [1] The MathWorks Inc., (1996), “SIMULINK Dynamic System Simulation for MATLAB”. The MathWorks Inc.
- [2] The MathWorks Inc., (1999), “Real-time Workshop User’s Guide, for use with SIMULINK”. Third printing for Version 3.0 (Release 11).
- [3] Yodakien, V. and Barabanov, M. “RTLinux Version Two” <http://www.rtlinux.org> [Accessed September 2001].
- [4] Wilshire, P. (2000) Real Time Linux: Testing and Evaluating [online], *Second Real-Time Workshop*, Available from: <http://www.linuxdevices.com/articles/AT2718102884.html> [Accessed September 2001].
- [5] Humphrey, M., Hilton, E. and Allaire, P., (1999), “Experiences using RT-Linux to implement a controller for a high speed magnetic bearing system”, *Proceedings of the Fifth IEEE Real-Time Technology and Applications Symposium*, pp. 121-130, ISSN 1080-1812.
- [6] The MathWorks Inc., (1999), “Real-time Windows Target User’s Guide, for user with Real-time Workshop”, The MathWorks Inc.
- [7] Golten, J. and Verwer, Andy, (1991), *Control System Design and Simulation*, McGraw-Hill International (UK), Cambridge, ISBN 0-07-707412-2, pp. 138-147.
- [8] Ogata, K., *Modern Control Engineering*, (1997), Prentice Hall, USA, ISBN: 0-13-261389-1, pp. 803-812.