Proceedings of the 2008 IEEE
International Conference on Robotics and Biomimetics
Bangkok, Thailand, February 21 - 26, 2009

Towards an Open-Source Integrated Development and Real-Time Control Platform for Robots

Shuhei Kume[1], Yoshikazu Kanamiya[2] and Daisuke Sato[2]
Musashi Institute of Technology (Tokyo City University, after April 2009)
1-28-1 Tamazutsumi, Setagaya-ku, Tokyo 158-8557, Japan
[1]*kume@rls.mes.musashi-tech.ac.jp,* [2]*{nenchev,sato}@ieee.org*

***Abstract*—The paper proposes an open-source platform for developing kinematic and dynamic models, simulation and control design of robots, based on MaTX [1]. We show that the developed programs can be easily transfered for direct execution in real-time, making use of improvements in the recent Linux kernel toward hard real-time capability. We show that, in combination with the real-time preemption patch of Ingo Molnar [2], there is the potential for real-time control of a robot arm. This is confirmed through experiments for real-time control of a Mitsubishi Heavy Industries seven-DOF PA-10 robot [3].**

***Index Terms*—Real-time robot control, Linux RT-Preemption patch, open-source platform, MaTX**

I. Introduction

Research and development in the field of robotics requires the choice of suitable platforms for (1) modeling and simulation, (2) development of new control algorithms and (3) real-time control. A variety of commercial and open-source platforms are available. In many cases, these three processes are handled separately, under different platforms. In the worst case scenario, the developed program will have to be entirely rewritten in a programming language suitable for real-time control. In the best case scenario, special tools will be available to automatically translate the development code into one suitable for real-time control. Both scenarios have their drawbacks, either leading to a time-consuming solution or to a cost-demanding one. Examples in the open-source field include ROBOOP [4] and RT-MBDYN [5] for modeling, Coin3D [6] and BLENDER [7] for simulation, Matlab-like Octave [8], Scilab/Scicos [9] for control algorithms, and RTOS's like VxWorks [10], μITRON [11], RTLinux [12], ART-Linux [13], RTAI [14], Xenomai [15] and others. Recently, efforts are underway to develop open-source platforms with seamless integration of the three processes, e.g. the European OROCOS project [16] and the Japanese RT-middleware project [17].

In our past work, we have been using the open-source platform MaTX [1] for modeling and control algorithm development, in combination with Coin3D for simulation and with RTLinux for real-time control. This is a heterogeneous environment, requiring efforts first to master, and then to work with the different components. There is also the problem that noncommercial RTLinux is available for Linux kernel versions up to 2.4.x only.

Recently, efforts are underway to implement real-time behavior into the Linux OS. A POSIX compatible patch for kernels 2.6.x, called RT preemption patch or RT-Preempt for short [2], [18], is available now. The patch allows nearly all of the kernel to be preempted. It further incorporates high resolution timers, referred to as hrtimers. The preemption capabilities of the patch together with the hrtimers, ensure hard real-time behavior of the Linux OS. This behavior has been tested in multimedia applications [19], with a painting robot [20], and with a wearable computer [21].

In this paper we describe our integrated modeling/simulation/control design and real-time control environment based on MaTX, running under Linux with the RT-Preempt patch. MaTX is a multi-platform, high-level programming platform that has an interpreter and a compiler [1], [22]. It is embedded with a variety of matrix/vector operations, math and other functions from control theory. MaTX has the advantage of being C-based. Hence, MaTX programs run usually "fast" and are easily linkable with other programs written in C, resulting in compact executable code. Because of this feature, it is possible to obtain real-time executable code directly from the development program, and to run it in real-time. We have confirmed this experimentally with a seven-DOF robot PA-10 (Mitsubishi Heavy Industries) [3].

The paper is organized as follows. In Section II the integrated environment based on MaTX and the RT-Preempt patch, and the experimental system are described. In Section III results from experiments are presented for various cases, including graphical and statistical jitter data. Section IV gives jitter analysis for the integrated environment, when the inverse kinematics for the experimental robot is solved in real-time. Finally, Section V presents the conclusions.

II. Integrated Environment for Robot Control with RT preempt and MaTX

A. The RT preempt kernel patch

It is well known that robot control requires generation of control inputs in a time-critical manner. This is known also as hard real-time. The standard Linux kernel does only meet soft-realtime requirements: it provides basic POSIX operations for userspace time handling but has no guarantees for hard timing deadlines. Nevertheless, efforts are underway to improve constantly the real-time capabilities of Linux kernels. Linux 2.6.x includes, for example, a number of significant improvements over Linux 2.4.x. There is a new scheduler algorithm that yields superior performance, especially under higher loads and on multiprocessor systems. There is also an improved threading model with in-kernel support for Native Posix Threading Library (NPTL) that increases the performance of threading operations and provides support for more threads.

In addition, there is now a kernel preemption patch. Traditionally, the Linux kernel will allow one process to preempt another only under certain circumstances. If kernel code is executing when some event takes place that requires a high-priority thread to start executing, the high-priority thread can not preempt the running kernel code, until the kernel code explicitly yields control. It is known that, in the worst case, the latency could potentially be hundreds milliseconds or more. So, before kernel 2.6.x, a user application could not preempt a task operating in kernel mode. With 2.6.x, preemption is possible resulting in lower latencies for user interactive applications. In addition, with Ingo Molnar's RT preemption patch [2] and Thomas Gleixner's generic clock event layer with high-resolution support [23], the kernel gains hard real-time capabilities. The patch becomes more and more usable and significant parts of it are leaking into the Linux kernel.

The RT-Preempt patch converts Linux into a fully preemptible kernel. The is done by means of [2]:

- making in-kernel locking-primitives (using spinlocks) preemptible though reimplementation with rtmutexes;
- making critical protected sections preemptible[1];
- implementing priority inheritance for in-kernel spinlocks and semaphores;
- converting interrupt handlers into preemptible kernel threads;
- converting the old Linux timer API into separate infrastructures for high-resolution kernel timers plus one for timeouts, leading to userspace POSIX timers with high resolution.

B. MaTX

So far, we have used MaTX for modeling and simulation, solving the direct kinematics, inverse kinematics and dynamics of robots. In addition, with MaTX, various types of control algorithms can be easily designed and implemented. In fact, the main purpose of the MaTX platform, developed by Masanobu Koga in 1989, is analysis, design and simulation of control systems.

The main features of MaTX are described as follows [1], [22]:

The interpreter (matx)

- matrices of an arbitrary size can be handled, and there is no need of variable type declaration;
- the four arithmetic operations can be used both on scalars and on matrices;
- when a matrix is defined, the variables can be treated as an element of the matrix.
- the same flow control and the same relational operators as in the C language are used.

The compiler (matc)

- a MaTX program can be easily converted to a C language program;
- a function needs to be defined just once. The respective values and definitions are allocated at the time of definition, so there is no need of multiple function definitions.

The libraries (matlib)

- there is a high-level library (automatic operation of memory allocation) and a low-level library (memory allocation by user);
- depending on the function, a real matrix is automatically distinguished from a complex matrix.

The matrix editor (mated)

- arbitrary size matrices can be handled, and in addition, the size can be reduced/expanded arbitrarily.
- The elements of a matrix can be deleted, moved, and copied.

In addition to the standard MaTX, there is a version for real-time control called RtMaTX [24]–[29]. The control program can be written in MaTX and the same functions as in MaTX can be used for real-time control without change. Unfortunately, RtMaTX relies on standard timer operations and is available only for MS-DOS (IBM PC compatible machines with DJGPP 2.21 compiler, IBM PC compatible machines/PC9801 machines with Borland C++ 3.1 compiler) [1]. These are outdated, so we prefer to use the new Linux RT-Preempt capabilities in combination with a Linux version of standard MaTX (MaTX-5.3.37 for Linux, glibc-2.3).

C. Experimental system hardware

Our experimental system is shown in Fig. 1. The robot is a Mitsubishi Heavy Industries seven-DOF PA-10 arm with respective controller PA-10-CNT. Communication with the control PC is via ARCNET [30]. Four ISA-bus ARCNET boards ARC-EVK/AT (Systemmicro) are used, each one capable of controlling up to two servo axes. We access directly the servo drivers in the controller, without using the internal control board. In this case, the sampling time is 2 ms. The four boards are inserted into the control PC — a control box with ISA bus backplane and a single-board computer PCA-6773 (Advantech). The specs are given in Table I.

The Development PC is a DELL DIMENSION 3100C desktop machine, with specs given in Table II. MaTX is installed and used for modeling, control law design and simulations. The developed program is then downloaded to the Control PC.

[1]It is still possible to create non-preemptible sections (in kernel), if necessary.

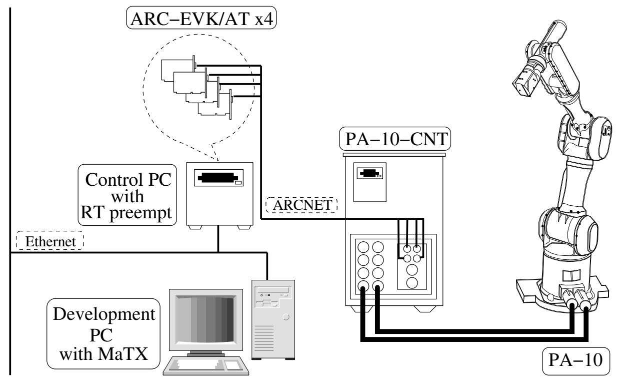


Fig. 1. Experimental system hardware.

TABLE I
SPECS. OF THE CONTROL PC (PCA-6773)

CPU	Intel Celeron ULV 650 MHz
System Chipset	VIA VT8606 "TwisterT"
Memory	SODIMM DDR 256 MB
OS	CentOS 5 (kernel 2.6.23-rt1)
Bus	ISA

TABLE II
SPECS. OF THE DEVELOPMENT PC (DELL DIMENSION 3100C)

CPU	Pentium 4 3GHz
Memory	DDR2 1.0 GB
OS	Ubuntu 7.10 (kernel 2.6.22-generic)

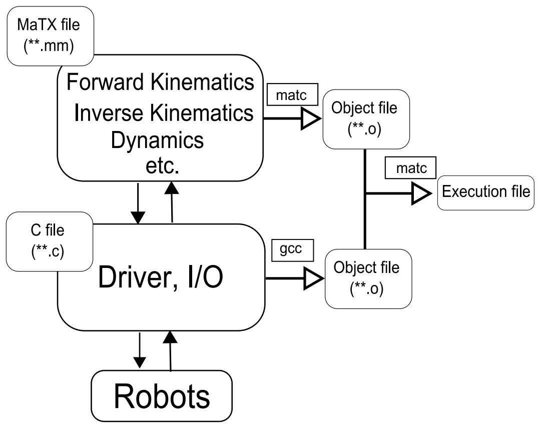
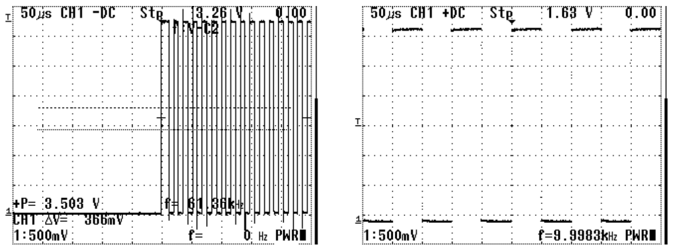


Fig. 2. Software components of the experimental system.



(a) Non real-time (b) Real-time

Fig. 3. Square wave with load.

D. Experimental system software

The software components used in our experimental system are shown in Fig. 2. The matc compiler is used to generate object files from the developed program. Servo driver access over ARCNET programs are written directly in C, and compiled via the gcc compiler. At the final stage, the object files are linked with the help of the matc compiler, and an executable file is produced. We note that it is quite easy to ensure linking of the MaTX - C programs with the matc compiler. The following command line ensures the generation of the executable code:

```
$ matc -v main.mm pa10.c -lrt
```

- matc: MaTX compiler/linker used to generate the executable code;
- main.mm: the main file of the development program, written in MaTX;
- pa10.c: the C program for real-time control (ARCNET driver access, 2 ms sampling time setup, memory allocation, task priority allocation);
- -v: matc compiler/linker option;
- -lrt: linker option to include the real-time library.

III. JITTER VERIFICATION

We verified the hard-real time capabilities of our system for different cases introduced below.

A. RT preempt only

First, we verified the hard-real time capabilities with the help of a C program accessing the parallel port to generate a square wave of 100 μs period. The program is available at [2].

The priority of a real-time program can be set between 0 (lowest) and 99 (highest) priority. In our example, the priority was set to 90. To ensure the desired square wave period, we use the standard POSIX function `clock_nanosleep()`. Two test were performed by executing the program, first without real time, and then under real time. Thereby, the CPU load was artificially increased via the command:

```
# cat /dev/zero > delete_this_file
```

The jitter was measured with the help of an oscilloscope. The results are shown in Fig.3.

From the left figure (a) it is clearly seen that in the case of non real-time, the generated signal is non periodic. On the other hand, from the right figure (b), it becomes apparent that under real-time, the generated signal represents a stable square wave. The average frequency is 9.9983 kHz, meaning that the average jitter is 0.017 μs. Having in mind that with a standard kernel, the guaranteed period is just 1 ms [31], we can conclude that with the RT-patch, it is possible to obtain highly accurate hard real-time capability.

B. PA-10 control under RT-Preempt only

We wrote a C program for controlling the robot arm under RT-Preempt, with sampling time of 2 ms. The desired input was rotation of the sixth joint axis from 0 to 45 degrees. Intermittent angular values and speeds were generated via a spline. Motion completes for 15 s, thereafter the desired angle remains 45 deg. The jitter is measured during the motion and thereafter, for a total of 300 s. The result is shown in Fig. 4. The values for the maximum jitter, average jitter and standard deviation are shown in Table. III. From these data we can conclude that hard real-time capability can be ensured.

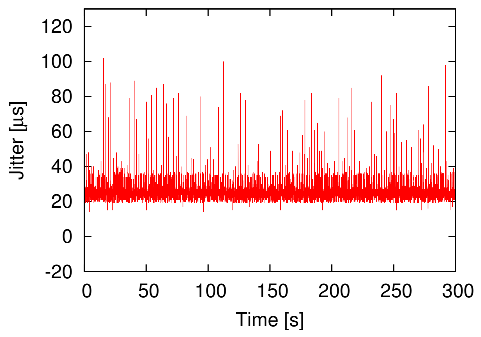
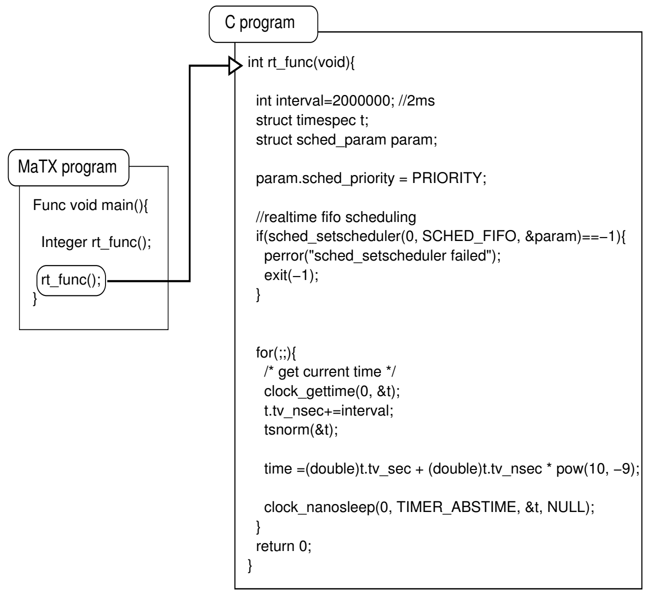


Fig. 4. Jitter for motion control under RT-Preempt.

TABLE III
STATISTICAL JITTER DATA FOR MOTION CONTROL UNDER RT-PREEMPT

Max	Average	Standard deviation
102 μs	25.1 μs	7.57 μs

C. Simple MaTX program under RT-Preempt

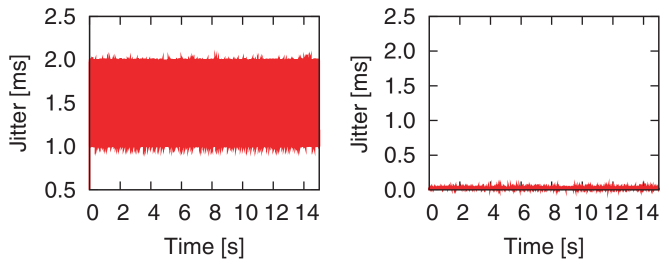


```
Func void main(){

 Integer rt_func();

 rt_func();
}
```

```
int rt_func(void){

 int interval=2000000; //2ms
 struct timespec t;
 struct sched_param param;

 param.sched_priority = PRIORITY;

 //realtime fifo scheduling
 if(sched_setscheduler(0, SCHED_FIFO, &param)==-1){
   perror("sched_setscheduler failed");
   exit(-1);
 }


 for(;;){
   /* get current time */
   clock_gettime(0, &t);
   t.tv_nsec+=interval;
   tsnorm(&t);

   time =(double)t.tv_sec + (double)t.tv_nsec * pow(10, -9);

   clock_nanosleep(0, TIMER_ABSTIME, &t, NULL);
 }
 return 0;
}
```

Fig. 5. Excerption from the simple verification program.

MaTX was installed on the Control PC (cf. Fig. 1). We wrote a simple MaTX program that makes just a call to a C program implementing the RT-Preempt capability (see Fig. 5). The two programs were compiled and linked together, as already explained in the previous section. The executable file was run again for two cases: non real-time and real-time. The sampling time was set to 2 ms to match the sampling time of the PA-10 arm. The motion execution time was 15 s. The results are shown in Fig. 6. The figure on the left (a) shows that without real-time, the jitter may reach 2 ms for most of the time. On the other hand, from the figure on the right (b) it can be seen that under real-time, jitter is very small. Zooming into the last data (see Fig. 7), we can see that this is true indeed. The values for the maximum jitter, average jitter and standard deviation are shown in Table IV. From these data, we can conclude that hard real-time capability can be ensured also for this case.



(a) Non real-time (b) Real-time

Fig. 6. Jitter for a simple MaTX program under RT-Preempt.

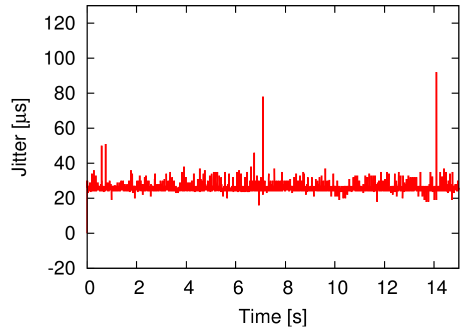


Fig. 7. Zoom in Fig. 6 (b) data.

TABLE IV
STATISTICAL JITTER DATA FOR RUNNING A SIMPLE MATX PROGRAM UNDER RT-PREEMPT

	Max	Average	Standard deviation
Non real-time	2.06×10^3 μs	1.45×10^3 μs	0.49×10^3 μs
Real-time	92.0 μs	25.5 μs	1.82 μs

D. PA-10 control with MaTX program

The MaTX program makes again just a call to a low-level C program that incorporates desired path generation for the sixth joint via a spline and calls to the ARCNET drivers, in addition to the RT-Preempt related settings, i.e. to the same C program used in the experiment described in III-B. The jitter graphs and the statistics values are shown in Fig. 8 and Table V, respectively. From these data, it is apparent the maximum jitter is 115 μs. This result is similar to that of the experiment in III-B. On the other hand, when compared with the result of III-C, it is seen that the jitter here is larger. We can make the conclusion then that calls to the ARCNET drivers lead to larger latencies and hence, somewhat increased jitter. This increase however does not diminish the hard real-time capability of the system.

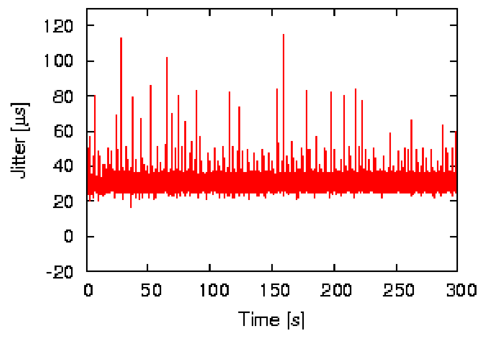


Fig. 8. PA-10 control with MaTX under RT-Preempt.

TABLE V
STATISTICAL JITTER DATA FOR PA-10 CONTROL WITH MATX UNDER RT-PREEMPT

Max	Average	Standard deviation
115 μs	27.0 μs	1.92 μs

IV. PA-10 CONTROL UNDER INTEGRATED ENVIRONMENT

We made an inverse kinematics program for the PA-10 arm in MaTX, to be used both for simulation and real-time control. This program generates desired inputs for the joint angles and the joint velocities and calls the low-level control C program, used in the last experiment. The new MaTX program and the C program were compiled and linked as usual. The executable program was used to control the robot. It turned out, however, that the jitter was too large, and the hard real-time capability could not be guaranteed. Closer examination revealed that the inverse kinematics program requires slightly over 1 ms time for the calculations. This, in addition to the time needed for ARCNET access (also slightly over 1 ms) lead to overall time requirement over the sampling time, set in this case to 2 ms.

We changed then the sampling time to 3 ms, and executed the same program. Jitter and statistics data are shown in Fig. 9 and Table VI, respectively. From the data, it is seen that the values are fully acceptable, and we can conclude that the hard real-time capability could be restored. This result demonstrates the potential of the integrated environment.

Finally, we checked performance by running two dummy load processes, similar to that described in Section III-A, in

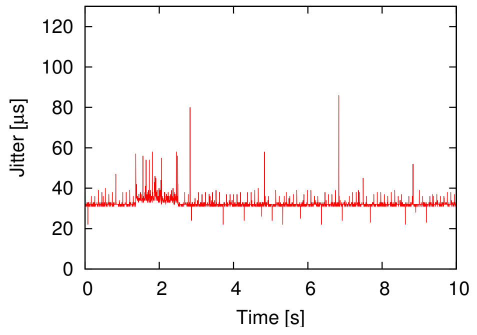


Fig. 9. Jitter for PA-10 inverse kinematics control in integrated environment.

TABLE VI
STATISTICAL JITTER DATA FOR PA-10 INVERSE KINEMATICS CONTROL IN INTEGRATED ENVIRONMENT

Max	Average	Standard deviation
86.0 μs	32.5 μs	2.58 μs

addition to the robot control process via MaTX. The sampling time was set again to 3 ms. The jitter graph is shown in Fig. 10, while the respective statistical data are given in Table VII. It is apparent that though the standard deviation increased, the maximum and average jitter remained of the same order. This demonstrates the capability of the system to deal also with multiprocess loads.

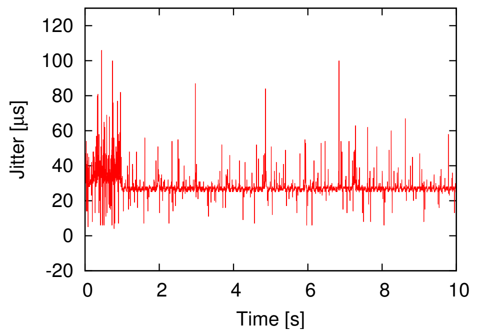


Fig. 10. Jitter for PA-10 inverse kinematics control in integrated environment with two loads.

TABLE VII
STATISTICAL JITTER DATA FOR PA-10 INVERSE KINEMATICS CONTROL IN INTEGRATED ENVIRONMENT WITH TWO LOADS

Max	Average	Standard deviation
106 μs	28.5 μs	6.29 μs

V. CONCLUSIONS

We examined the potential for creating an integrated environment for robot control, based on the high-level programming platform MaTX for modeling, simulation and control design, running under Linux with the real-time preemptive patch developed by Ingo Molnar. We have shown that it is possible to run the high-level inverse kinematics MaTX program for the PA-10 arm directly under hard real-time, albeit with some increase in the sampling time. We note that the hardware we used (ISA bus) was somewhat outdated, which is one of the reasons why the sampling time had to be increased.

In a future work we plan to examine other hardware platforms for the MaTX — Linux + RT-Preempt combination to reduce the sampling time and to allow for implementing more advanced control algorithms in hard real-time, including dynamic control of multi-DOF robots.

REFERENCES

[1] MaTX. [Online]. Available: http://www.matx.org/

[2] Real-Time Linux Wiki. [Online]. Available: http://rt.wiki.kernel.org/index.php/Main_Page

[3] Mitsubishi Heavy Industries, Ltd. [Online]. Available: http://www.mhi.co.jp/kobe/mhikobe/products/mechatronic/index.html

[4] ROBOOP - A robotics object oriented package in C++. [Online]. Available: http://www.cours.polymtl.ca/roboop/

[5] RT-MBDYN - MultiBody Dynamics Analysis Software on Real Time Distributed Systems. [Online]. Available: http://www.aero.polimi.it/ mbdyn/mbdyn-rt/index.html

[6] 3D Graphics Development Tools. [Online]. Available: http://www.coin3d.org/

[7] blender.org. [Online]. Available: http://www.blender.org/

[8] Octave. [Online]. Available: http://www.gnu.org/software/octave/

[9] Scicos Homepage. [Online]. Available: http://www-rocq.inria.fr/scicos/

[10] Wind River. [Online]. Available: http://www.windriver.com/

[11] ITRON Project Archive. [Online]. Available: http://www.ertl.jp/ITRON/home-j.html

[12] Wind River: RTLinuxFree. [Online]. Available: http://www.rtlinuxfree.com/

[13] ART-Linux. [Online]. Available: http://www.dh.aist.go.jp/research/humanoid/art-linux.html/

[14] RTAI - the RealTime Application Interface for Linux from DIAPM. [Online]. Available: https://www.rtai.org/

[15] Xenomai: Real-Time Framework for Linux. [Online]. Available: http://www.xenomai.org/index.php/Main_Page

[16] The OROCOS Project Smarter control in robotics & automation!. [Online]. Available: http://www.orocos.org/

[17] OpenRTM-aist Official Web Site. [Online]. Available: http://www.is.aist.go.jp/rt/OpenRTM-aist/html /FrontPage.html

[18] Index of /pub/linux/kernel/projects/rt. [Online]. Available: http://www.kernel.org/pub/linux/kernel/projects/rt/

[19] Daniel Walker, "The Evolution of Real-Time Linux," presented at 7th RTL Workshop [Online]. Available: ftp://ftp.realtimelinuxfoundation.org/pub/events/ rtlws-2005/SvenThorstenDietrich.pdf

[20] Morten Mossige, Pradyumna Sampath and Rachana G Rao, "Evaluation of Linux RT-Preempt for embedded industrial devices for Automation and Power Technologies - A Case Study," presented at 9th RTL Workshop [Online]. Available: http://www.linuxdevices.com/files/article081/Sampath.pdf

[21] Dongwook Kang, Woojoong Lee and Chanik Park, "Kernel Thread Scheduling in Real-Time Linux for Wearable Computers," *ETRI Journal*, vol. 29, no. 3, pp. 270–280, June 2007.

[22] Katsuhisa Furuta, Masanobu Koga, "A New Interactive CAD based on Window System," *The 29th Annual Conference of The Institute of Electrical Engineers of Japan*, Tokyo, Japan, July 1990, pp. 467–468.

[23] The high-resolution timer API [Online]. Available: http://lwn.net/Articles/167897/

[24] Masanobu Koga, "MaTX - A High-Performance Programing Language for Scientific and Engineering Computation," *Journal of The Robotics Society of Japan*, vol.14, No.6, pp. 300–303, 1996.

[25] Masanobu Koga, "An Integrated Environment for Simulation and Real-Time Control of Control Systems," *Journal of The Institute of Electrical Engineers of Japan*, vol. 118-C, no. 4, pp. 551–557, 1998.

[26] Masanobu Koga, Hiromitsu Kiyota, Naoto Nakayama and Mitsuji Sampei, "A Computer Environment for Design of Control System using MaTX," *JSME Centennial Grand Congress D&D'97 Conference*, Tokyo, Japan, July 1997, vol. A, pp. 82–85.

[27] Masanobu Koga, Hiroaki Toriumi, Mitsuji Sampei, "Real-Time CAD of Control Systems Achieving On-line Parameter Tuning," *the 35th Annual Conference of The Institute of Electrical Engineers of Japan*, 1996, pp. 181–182.

[28] Masanobu Koga, Hiroaki Toriumi and Mitsuji Sampei, "Real-time CAD of Control Systems Achieving Cooperation of Modeling and Design of Controllers," *Proceeding of the 1996 IEEE International Symposium on Computer-Aided Control System Design*, Ritz-Carlton, USA, Sept. 1996, pp. 457–462.

[29] Masanobu Koga, Hiroaki Toriumi, Mitsuji Sampei, "An Integrated Software Environment for Design and Real-Time Implementation of Control Systems," *11th IFAC Symposium on System Identification (SYSID'97)*, Kitakyusyu, Japan, July 1997, pp. 1603–1609.

[30] ARCNET Trade Association. [Online]. Available: http://www.arcnet.com/

[31] Masaaki Kumagai, "Robot Control Using Common Linux With KNOPPIX," *Proceeding of 2008 JSME Conference on Robotics and Mechatronics (ROBOMEC 2008)*, Tokyo, Japan, May 2006, 1P1-C40.