International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307, Volume-2, Issue-6, January, 2013

Real-Time Speed Control of a DC Motor using Open Source Code Tools

Ujjwal Mondal, Parthasarathi Satvaya, Sourav Kumar Das

Abstract*— The presented work envisaged to explore the possibility of developing ultra-low cost experimental setup for teaching and learning Real-Time systems. The presented work demonstrates, in steps, the development of a real-time control system with free open source code softwares. The free suite utilized and experimented within the present work composed by Linux operating system and the Real Time Application Interface (RTAI) add-on, the Scilab Computer Aided Control System Design (CACSD) software and the Control & Measurement Device Interface (COMEDI) drivers. Scilab/Scicos, a free scientific software package for numerical computations and control system simulation is used with RTAI to provide hard real-time extensions in to Linux environment. The development and deployment platform are the same and consisted of the (i) Linux, (ii) Scilab/ Scicos (iii) RTAI and (iv) COMEDI drivers running in a PC. The investment is reduced to the hardware as well as in software cost, which consists of a standard PC, dc motor and a COMEDI compatible acquisition board. The most obvious advantage of the proposed solution is that all the software or codes are free & available in the web. The whole idea is demonstrated by real time speed control of a dc motor using Pulse Width Modulation (PWM).

***Index Terms*— RTAI, CACSD, COMEDI, SCILAB/ SCICOS, PWM.**

I. INTRODUCTION

Rapid Control Prototyping (RCP) requires two components [1, 2] Viz. Computer Aided Control System Design (CACSD) software and a dedicated hardware with a hard real-time operating environment. Popular & widespread RCP environments are based on the commercial software Matlab/Simulink/Realtime-Workshop (RTW) - Real Time Windows Target (RTWT) CACSD software or LABVIEW which can be used to generate and compile codes for different targets. The main disadvantage of this solution is the cost of the software. The proposed solution overcomes the said problem as it can be freely downloaded from the web. Development system is based on Scilab/Scicos and Linux RTAI, a hard real-time extension of the Linux operating system [3]. This environment allows to quickly creating real-time controllers for real plants by generating and compiling the full control application directly from the Scicos scheme.

A real time system must respond to a signal, event or request fast enough to satisfy some time constraints with extreme reliability.

In order to get a real-time eration a standard kernel must be configured in a Linux base and before this configuration it will include the patching of Hardware Abstraction Layer (HAL) or Adaptive Domain Environment for Operating Systems (ADEOS) with the kernel. After patching and configuring the kernel (to make it real time compatible), installation of the RTAI package must be carried out including rtai-lab and comedi. After this whole process, a set of kernel modules are created in the user specified directory ("/usr/realtime"). Loading these modules, the real-time functionality is obtained [4].

In this stage keeping the entire previous configuration we should include COMEDI support over RTAI. The RTAI package with rtai-lab and COMEDI can be access through Scilab, when RTAI with COMEDI add-ons to Scilab and loads COMEDI modules. Scilab/Scicos gives the GUI to make RT simulation and as well as to generate codes and executable for RT operation [5]. In our experiments a COMEDI supported DAQ card is taken to set the RT target. Running the created RT executable in a Linux terminal we can observe the RT simulated signal through a CRO. Farther work may be the generating of RT control signal for a small plant.

II. DEVELOPMENT SYSTEMS

A. *Hardware*

1. A P4 or equivalent processor
2. Minimum 512MB RAM
3. DC motor
4. Driver electronics circuit
5. Data Acquisition Card (DAQ)

B. *Softwares or Codes*

1. Operating System: Functional GNU/Linux environment, better with a Debian or a Debian-like (e.g. Ubuntu) distribution.
2. A Kernel: it is necessary to ensure the best when the kernel version of the Linux-OS is as close as possible to the kernel we are going to compile and to merge with the RTAI.
3. RTAI source code
4. Scilab source code
5. COMEDI and COMEDI-LIB source

Another two supporting source codes are required to install. First one is "Mesa 3D" graphical library from and second one is the "EFLTK" graphic widgets library. Some software packages may have to upgrade and those are Automake, autoconf bison (for comedi) cpp, ftgl-dev (for efltk), gcc, g77, g++, gtk, libbind, libglu1-mesa-dev, libglut-dev, libfltk, libgtk-dev, libdrm-dev, libncurses, libperl-dev,mesa (related all packages),tcl8.4, tk8.4, tcl-8.4-dev, tk8.4-dev, tcllib-1.9, x11-proto.

III. DEVELOPMENT PROCESS

A. *Software development process in steps*

1. Operating System: Functional GNU/Linux environment, (experimented with Ubuntu 6.06)

Manuscript received on January, 2013.

Ujjwal Mondal, Department of Applied Electronics & Instrumentation Engineering, RCC Institute of Information Technology, Kolkata, India.
Parthasarathi Satvaya, Department of Electrical Engineering, Haldia Institute of Technology, Haldia, India.
Sourav Kumar Das, Department of Electrical Engineering, Haldia Institute of Technology, Haldia, India.

2. Unpacking of kernel and RTAI source codes in the directory "/usr/src" in the installed Linux. To ensure the best performance, the kernel version we are going to compile and to merge with the RTAI, should be as close as possible to the kernel version of the Linux-OS.
3. Patching of the HAL or ADEOS over the kernel under configuration.
4. Configuring the kernel for real time applications.
5. Compilation and Installation of the newly configured kernel.
6. Updating of the boot loader to access newly installed kernel.
7. Mesa and EFLTK installation
8. Installation of COMEDI and COMEDI-LIB
9. Configuration, compilation and Installation of RTAI.
10. Installation of Scilab & RTAI add-on to it [6].
11. Creating shared memory inodes for the activation of RTAI and COMEDI.
12. Loading RTAI, COMEDI and DAQ modules.

B. *Hardware development process*

1) *DC Motor Specifications*

A dc motor is taken for real time experimentation purpose with following details:

Model name: RF-500TB-12560

Voltage: Operating range: 6 volts to 12 volts.

Nominal=12 volts constant.

At No load:

Speed=5600 rpm.

Current=0.03 amp.

At maximum efficiency:

Speed=4653 rpm.

Current=0.11 amp.

Torque=18 g-cm.(1.76 mN-m)

Efficiency=67%

At stall:

Current=0.6A

Torque=12 g-cm(11.76 mN-m).

2) *DC Motor Driver Electronics*

As a digital driver of the dc motor model RF-500TB-12560, we have taken L293D push-pull four channel driver. A little modification is done in the input section. For the safety of data acquisition card and PC, Optocoupler PC817 is used in the input section to keep Data Acquisition Card and dc motor driver circuitry optically coupled or isolated from direct contact (Fig. 1).

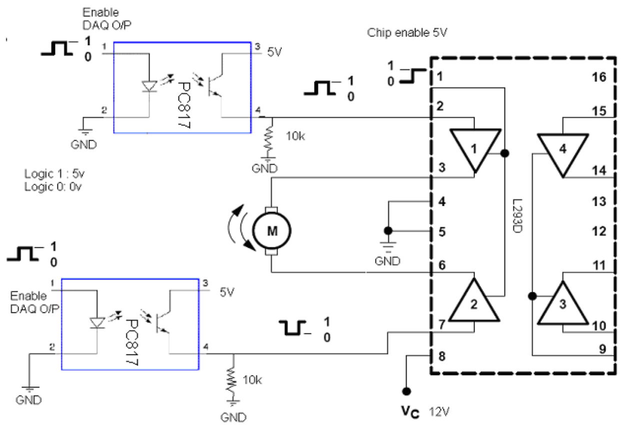


Fig. 1 dc motor driver electronics

3) *Data Acquisition Card*

A data acquisition card plugs directly into a personal computer's bus. All the power required for the A/D converter and associated interface components on the data acquisition card is obtained directly from the PC bus. For the presented work, RTAI with COMEDI provides a built-in graphical tools and libraries for data acquisition and analysis. "Advantech PCI-1711" data acquisition card is used which is a powerful and multifunction cards for the PCI bus and supported by COMEDI library.

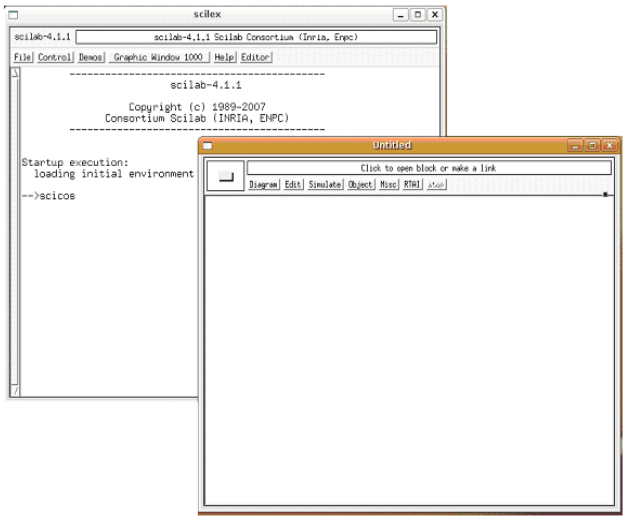
IV. EXPERIMENTATION

A. *Creating block diagram for Square wave generation*

Open the TERMINAL and type "scilab", it will open scilab window and in the scilab window type 'scicos' and it will open untitled window shown in Fig. 2. Then open menu "edit" and Select palettes. In the Palettes, select Sources at the top of the pop-up window [7, 8]. This will open a window with a group of source blocks as shown in Fig. 3. Take the red clock on the Scicos diagram page. Open the RTAI-Lib palette in a similar way as before and it will look like Fig. 4. From the RTAI-Lib palette, take the "Square" block, "Scope" block & "COMEDI D/A" block and place it in the main Scicos window. Connect those blocks. After drawing the Block diagram, we should make the "super block". So we should go to menu "Diagram" and select "Region to super block". Cover all the blocks excluding the Clock and dragging the mouse i.e. we must draw an elastic frame around all the blocks as in Fig. 5 and it will make the required super block as shown in Fig. 6. Double clicking on the super block we can again open those basic blocks to set parameters as shown in Fig. 7.

- Set parameters of Super-blocks:
  Square block- "Val[0]/amplitude=1", "Val[1]/time period=1" and "Val[2]/On time=0.5"
  & leave other parameter to default value.
  Comedi block- Keep default value (channel 0)
  Scope block- Keep default value.
- Close the window and set clock parameter.

Clock: Set "Period =0.001" and "Init Time=0" Connect the analog output (Channel 0) and analog ground of the signal acquisition card to a real oscilloscope. For example: with the "advantech PCI-1711" DAQ card, connect pins 58 (DAC0OUT) and 57 (AOGND).



```
scilab-4.1.1 Scilab Consortium (Inria, Enpc)
-------------------------------------------
              scilab-4.1.1

         Copyright (c) 1989-2007
       Consortium Scilab (INRIA, ENPC)
-------------------------------------------

Startup execution:
  loading initial environment

-->scicos
```

Fig.2 Scicos Interface

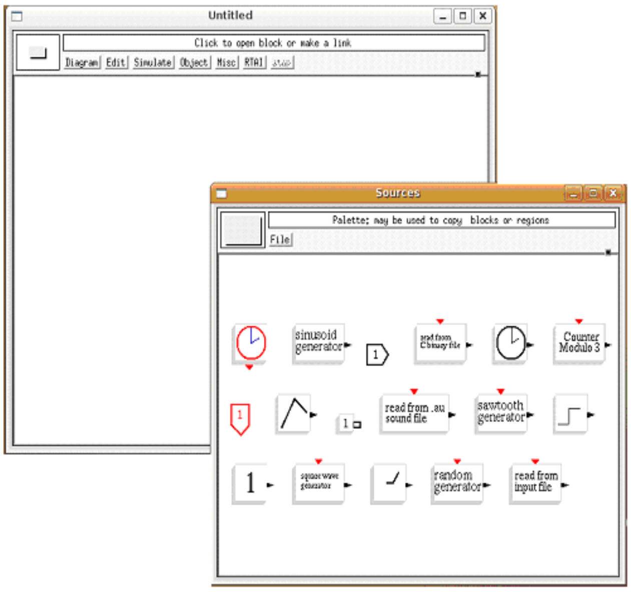


Fig.3 Scicos Source Blocks

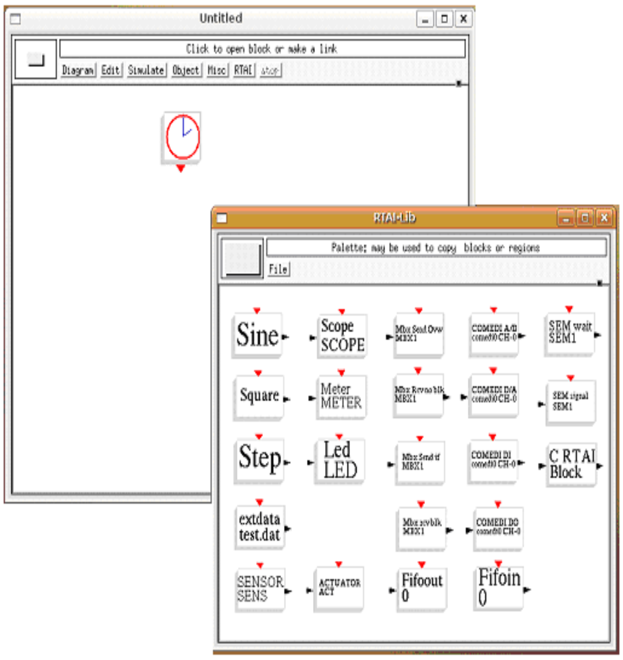


Fig.4 RTAI-Lib palette

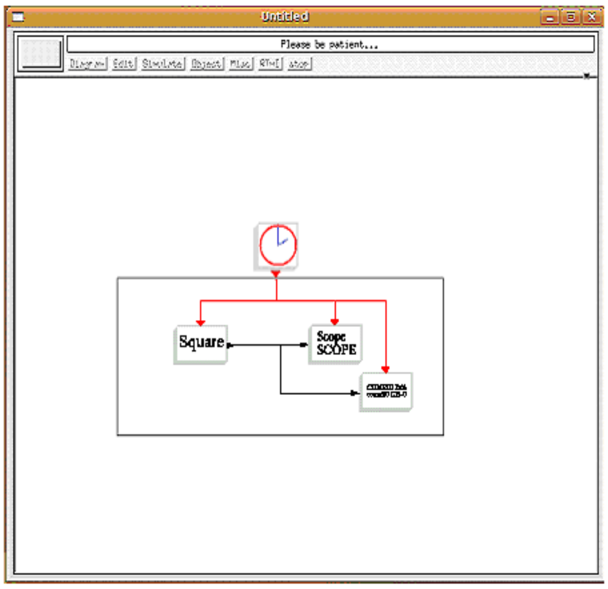
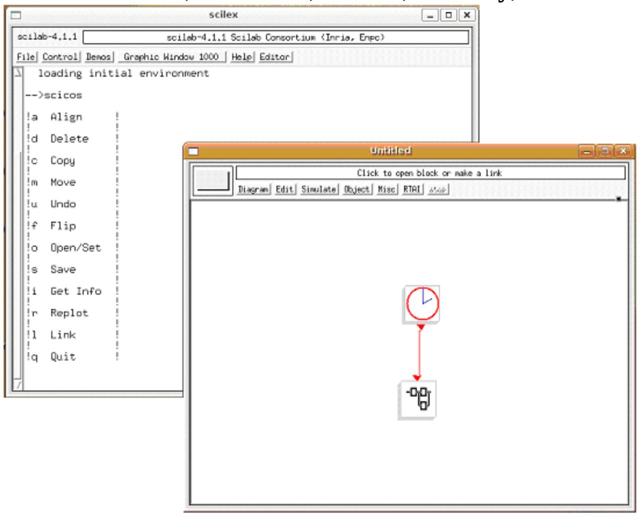
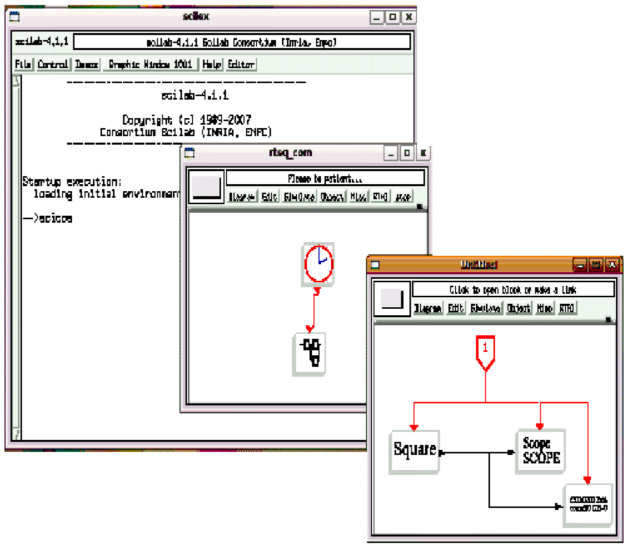


Fig.5 Making of RTAI Super block


scilex
scilab-4.1.1
scilab-4.1.1 Scilab Consortium (Inria, Enpc)
File Control Demos Graphic Window 1000 Help Editor
Untitled
Click to open block or make a link
Diagram Edit Simulate Object Misc RTAI stop

```
  loading initial environment

-->scicos

!a  Align      !
!d  Delete     !
!c  Copy       !
!m  Move       !
!u  Undo       !
!f  Flip       !
!o  Open/Set   !
!s  Save       !
!i  Get Info   !
!r  Replot     !
!l  Link       !
!q  Quit       !
```

Fig.6 RTAI Super block

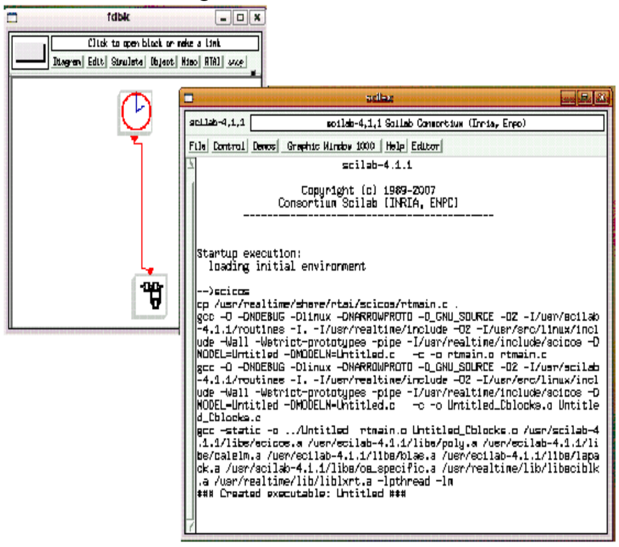

scilex
scilab-4.1.1
scilab-4.1.1 Scilab Consortium (Inria, Enpc)
File Control Demos Graphic Window 1001 Help Editor
rtsq_com
Please be patient...
Untitled
Click to open block or make a link
Square
Scope SCOPE

```
            scilab-4.1.1

       Copyright (c) 1989-2007
    Consortium Scilab (INRIA, ENPC)

Startup execution:
  loading initial environment

-->scicos
```

Fig.7 Inside of Super block

B. Checkout of RT signal through X-rtailab and Oscilloscope

Now going to the "RTAI" menu select "Set Target" and click over super block. Now we have to compile using "RTAI-Code gen" again through menu "RTAI". If compilation is properly done then on the scilab prompt a group of information will come with the lat line "Created Executable" as in Fig. 8.


fdbk
Click to open block or make a link
Diagram Edit Simulate Object Misc RTAI stop
scilex
scilab-4.1.1
scilab-4.1.1 Scilab Consortium (Inria, Enpc)
File Control Demos Graphic Window 1000 Help Editor

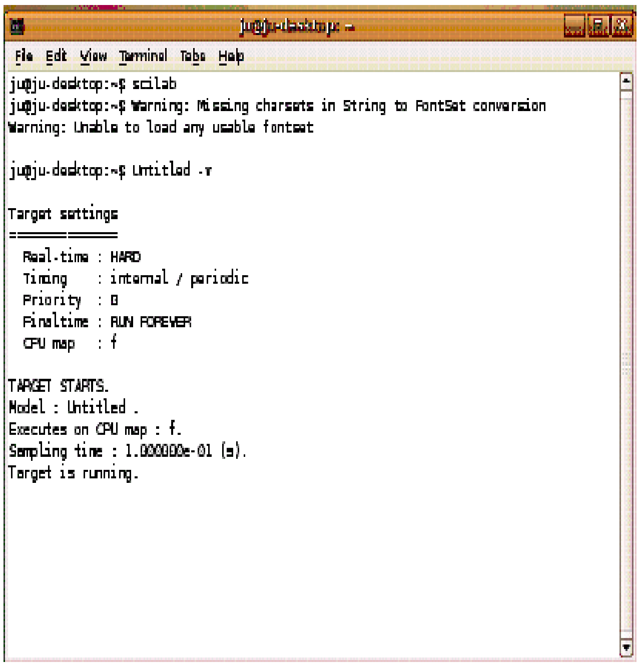
```
                scilab-4.1.1

            Copyright (c) 1989-2007
         Consortium Scilab (INRIA, ENPC)
    ----------------------------------------


Startup execution:
  loading initial environment

-->scicos
cp /usr/realtime/share/rtai/scicos/rtmain.c .
gcc -O -DNDEBUG -Dlinux -DNARROWPROTO -D_GNU_SOURCE -O2 -I/usr/scilab
-4.1.1/routines -I. -I/usr/realtime/include -O2 -I/usr/src/linux/incl
ude -Wall -Wstrict-prototypes -pipe -I/usr/realtime/include/scicos -D
MODEL=Untitled -DMODELN=Untitled.c   -c -o rtmain.o rtmain.c
gcc -O -DNDEBUG -Dlinux -DNARROWPROTO -D_GNU_SOURCE -O2 -I/usr/scilab
-4.1.1/routines -I. -I/usr/realtime/include -O2 -I/usr/src/linux/incl
ude -Wall -Wstrict-prototypes -pipe -I/usr/realtime/include/scicos -D
MODEL=Untitled -DMODELN=Untitled.c   -c -o Untitled_Cblocks.o Untitle
d_Cblocks.c
gcc -static -o ../Untitled  rtmain.o Untitled_Cblocks.o /usr/scilab-4
.1.1/libs/scicos.a /usr/scilab-4.1.1/libs/poly.a /usr/scilab-4.1.1/li
bs/calelm.a /usr/scilab-4.1.1/libs/blas.a /usr/scilab-4.1.1/libs/lapa
ck.a /usr/scilab-4.1.1/libs/os_specific.a /usr/realtime/lib/libsciblk
.a /usr/realtime/lib/liblxrt.a -lpthread -lm
### Created executable: Untitled ###
```

Fig.8 Compilation Information

Let the new executable is renamed as "rt_square" and saved in the current directory.

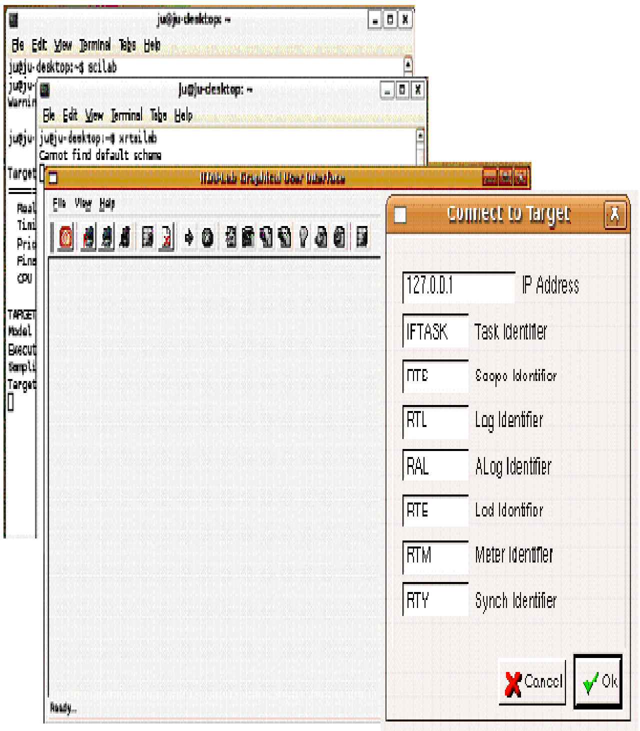
- In one terminal type: "rt_square –v" to run the executable in Hard RTS mode with verbose output as in Fig. 9.
- In another terminal type: "xrtailab" to open a GUI & from "File" menu select "Connect" and it will give the option to set the target. Click on "OK" as shown in Fig. 10.
- A square wavelike wave form can be seen on the Oscilloscope.

In xrtailab going to "View" select "parameters" and "scope". Now visualization parameters can be adjusted in the "xrtailab" to see the square wave properly in to the oscilloscope as shown in Fig. 11, 12.



```
ju@ju-desktop:~$ scilab
ju@ju-desktop:~$ Warning: Missing charsets in String to FontSet conversion
Warning: Unable to load any usable fontset

ju@ju-desktop:~$ Untitled -v

Target settings
===============
  Real-time : HARD
  Timing    : internal / periodic
  Priority  : 0
  Finaltime : RUN FOREVER
  CPU map   : f

TARGET STARTS.
Model : Untitled .
Executes on CPU map : f.
Sampling time : 1.000000e-01 (s).
Target is running.
```

Fig.9 Running the Executable



```
ju@ju-desktop:~$ scilab
ju@ju-desktop:~$ xrtailab
Cannot find default schema
```

Connect to Target

127.0.0.1	IP Address
IFTASK	Task Identifier
RTS	Scope Identifier
RTL	Log Identifier
RAL	ALog Identifier
RTE	Led Identifier
RTM	Meter Identifier
RTY	Synch Identifier

Fig.10 xrtailab Interface

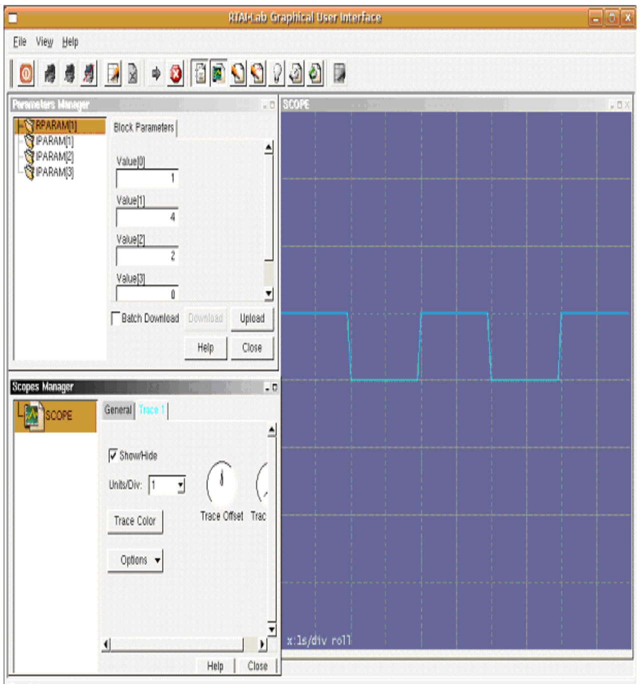


Fig.11 Square wave in the scope of xrtailab

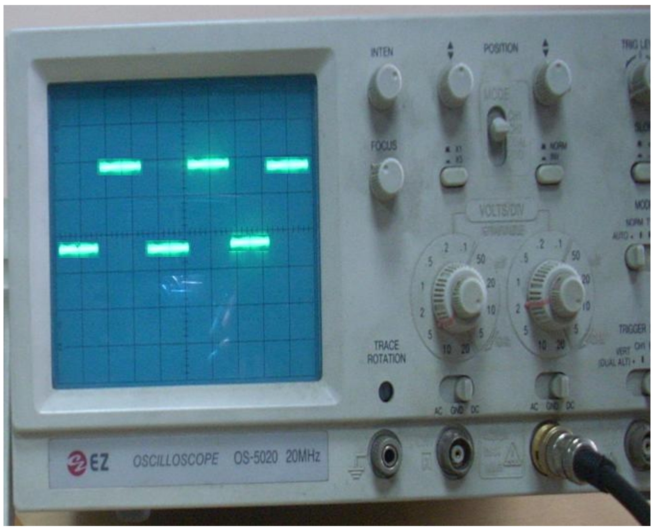


Fig.12 Square wave in the Oscilloscope

C. Real-time speed control of dc motor using PWM

Pulse-width modulation (PWM) or duty-cycle variation methods are commonly used in speed control of DC motors. The duty cycle is defined as the percentage of digital 'high' to digital 'low' plus digital 'high' pulse-width during a PWM period i.e. PWM the output voltage is the average of the supplied voltage over ON/OFF time.

$$V_{av} = V_s * D = (T_{on} * V_s)/(T_{on} + T_{off})$$

When V_{av}=average voltage, D= duty cycle, V_s =Supply Voltage, T_{on}=On time of the signal, T_{off}=Off time of the Signal. Controlling the period ($T_{on}+T_{off}$) and on time (T_{on}) of input pulses, the speed of the dc motor can be controlled as the dc motor speed varies with the variation of the average amplitude of input voltage to it. Steps to control the dc motor are as follows:

- Generate a real time square wave (Sec-IV).
- Take that output signal from DAQ.
- Fed that signal to the input section of a DC motor driver (Sec. III).
- Connect the DC motor driver output to the input of DC motor (SEC. III).

- Change the parameters on the fly in "xrtailab" interface (Fig. 11).
- Choose the "source" block parameter and set values as:
  Val (0) = 1 (amplitude of pulses)
  Val (1) = 0.0009 ($T_{on} + T_{off}$ = Period of pulse)
  Val (2) = 0.0002 (T_{on})

Now varying Val (2) = 0.0002 to 0.0009 (Pulse Width Modulation) on the fly, speed of the motor can be changed. It has been successfully experimented and variation in speed of the motor is observed. The dc motor with following details is connected to the driving circuit.

Supply voltage to driver circuit= 5V.
Voltage across motor=5 volts.
Current=0.03 amps
Short circuit current=0.5 amps.
A set of results is tabulated in Table 1.

TABLE 1

Sl. No.	Duty cycle (%)	Voltage across motor (V)	Current through motor (A)
1	22.22	1.11	0.007
2	33.33	1.67	0.010
3	44.44	2.20	0.014
4	55.55	2.78	0.017
5	66.66	3.34	0.021
6	77.77	3.91	0.025
7	88.89	4.44	0.028
8	99.99	4.91	0.031

Table 1: Parameters reading during RT control of dc motor

V. CONCLUSIONS

Successful implementation of the real-time system development and deployment were demonstrated by Speed Control of a DC Motor using Pulse Width Modulation.

The advantage of the proposed solution is that all the softwares are freely available on the web. However, unlike the (costly) commercial packages, the information available about these free softwares is scanty or sometimes confusing.

The contribution of this work is the attempt to remove some of the difficulties by tracing through the development steps and pitfalls.

In conclusion, this paper shows that with some adjustments and moderate additional effort, control system designing tools Scilab/Scicos and RTAI with COMEDI can successfully replace the costly commercial alternatives for teaching and learning Real Time Systems.

REFERENCES

[1] R. Bucher and L. Dozio, Paolo Mantegazza, "Rapid Control Prototyping with Scilab/Scicos and Linux RTAI", The Vlsi Journal (2004) pp. 739-744
[2] J. Jang, C. K. Ahn, S. Han, and W. H. Kwon, "Rapid Control Prototyping for Robot Soccer System using SIMTool," *in ProcSICE-ICASE International Joint Conference 2006*, Busan, Korea, Oct. 2006, vol. 2, pp. 3035–3039.
[3] R. Bucher and L. Dozio, "CACSD with Linux RTAI and RTAI-Lab," in Real Time Linux Workshop, Valencia, 2003.
[4] Giovanni Racciu and Paolo Mantegazza. RTAI 3.3 User Manual, 2006. URL www.rtai.org.
[5] "RTAI-Lab Tutorial" by Roberto Bucher, Simone Mannori and Thomas Netter, 2006
[6] Ramine Nikoukhah and Serge Steer, SCICOS - A Dynamic System Builder and Simulator, User's Guide, 1998. http://www.scilabsoft.inria.fr/doc/scicos/scicos.htm
[7] Stephen L. Campbell, Jean-Philippe Chancelier, and Ramine Nikoukhah. Modeling and Simulation in Scilab/Scicos. Springer, Berlin, Germany, 2006. URL www.scicos.org
[8] Roberto Bucher and Silvano Balemi "Scilab/Scicos and Linux RTAI –A unified approach" 2005 IEEE Conference on Control Applications Toronto, Canada, August 28-31, 2005

Ujjwal Mondal (b.1977) received his B.Tech. degree in Electronics & Instrumentation Engineering from University of Kalyani, West Bengal, India, in 2005, the M.E. degree in Control Systems from Jadavpur University, West Bengal, India, in 2008. His research interests include Real Time Systems, Electronic Instrumentation; Wavelet based system analysis & Repetitive Control. At present, He is an assistant professor, Applied Electronics & Instrumentation Engineering, RCC Institute of Information Technology, Kolkata, and West Bengal, India.

Parthasarathi Satvaya (b.1985) received his B.Tech. degree in Electronics & Instrumentation Engineering from Bankura Unnayani Institute of Engineering, West Bengal University of Technology, West Bengal, India, in 2006, the M.E. degree in Illumination Engineering from Jadavpur University, West Bengal, India, in 2008. His research interests include Smart Lighting, Real Time Systems, Electronic Instrumentation. At present, He is an Assistant Professor, Department of Electrical Engineering, Haldia Institute Technology, Haldia, and West Bengal, India.

Sourav Kumar Das (b.1984) received his B.Tech. degree in Electronics & Communication Engineering from Dumkal Institute of Engineering & Technology under West Bengal University of Technology, West Bengal, India, in 2006, the M.E. degree in Control Systems from Bengal Engineering & Science University, Shibpur, West Bengal, India, in 2009. His research interests include Real Time Systems, Power Electronics, DSP & Advanced Control Systems. At present, He is an assistant professor, Electrical Engineering, Haldia Institute Technology, Haldia, and West Bengal, India.