Journal of Education and Vocational Research
Vol. 4, No. 2, pp. 24-29, Feb 2013 (ISSN 2221-2590)

An Easy-to-use and Economic Experimental Apparatus for Motion Control Education

Manh-Tuan Ha, Chul-Goo Kang*
Konkuk University, Seoul, Korea
*cgkang@konkuk.ac.kr

Abstract: Motion control education in engineering requires hands-on experiments to catch the concept of feedback control, but experimental apparatus for motion control experiments is expensive, complex in mechanism and is hard to use it in general. In this paper, we develop an easy-to-use and economic experimental apparatus for motion control education using an open-source free operation system LINUX and a free real-time kernel RTAI. The developed apparatus is reliable in mechanical structure, and fast enough to realize 100 microsecond sampling time for real-time control purpose, and so it can test conventional PID control logics, various input shaping control logics and advanced motion control logics with cheap prices. The validity of the developed motion control apparatus is demonstrated by testing P, PD control logics and ZV input shapers experimentally.

Keywords: *Experimental apparatus, Motion control, Real-time control, Engineering education, Control education*

1. Introduction

In recent years, open-source LINUX operating systems which are significantly developed for graphics user interface such as UBUNTU, FEDORA etc are available in public for LINUX users. These LINUX operating systems are convenient for programming since they include the open source libraries which have various control modules that are useful for control engineers. Even if LINUX is multi-tasking operating systems, it is not true real-time operating system. For real-time control purpose, we can import free real-time kernel such as RTAI into LINUX operating system. The usual way in industry for real-time control is to use the real-time operating system (RTOS), but this is very expensive in general due to high cost of RTOS. There are many research works that use RTAI and LINUX for the development of a laboratory model (Anju, 2008), an electromechanical system (Matej & Martin, 2008), and a real-time control system (Michele et al., 2005), and a motion control system (Kwak & Kang, 2010). Motion control education in engineering requires hands-on experiments to catch the concept of feedback control, but experimental apparatus for motion control experiments is expensive, complex in mechanism and is hard to use it in general. In this paper, we develop an easy-to-use and economic experimental apparatus for motion control education using an open-source free operation system LINUX and a free real-time kernel RTAI. We use open-source operating system UBUNTU 11.04LTS, RTAI version 3.8, COMEDI library and NI PCI 6221 card to make a real-time apparatus. The controlled object is an AC servo motor connected with ball screw. The developed apparatus is reliable in mechanical structure, and fast enough to realize 100 microsecond sampling time for real-time control purpose, and so it can test conventional PID control logics, various input shaping control logics and advanced motion control logics with cheap prices. The validity of the developed motion control apparatus is demonstrated by testing P, PD control logics and ZV input shapers experimentally.

2. RTAI and COMEDI software

After UBUNTU 11.04LTS operating system has been installed, we can choose appropriate module RTAI to install. The setup file and detail instruction can be found in website www.rtai.org. Before installing COMEDI library, we have to confirm that RTAI is successfully installed. We can use the command below to test RTAI.

```
cd /usr/realtime/testsuite/kernel/latency ./run
```

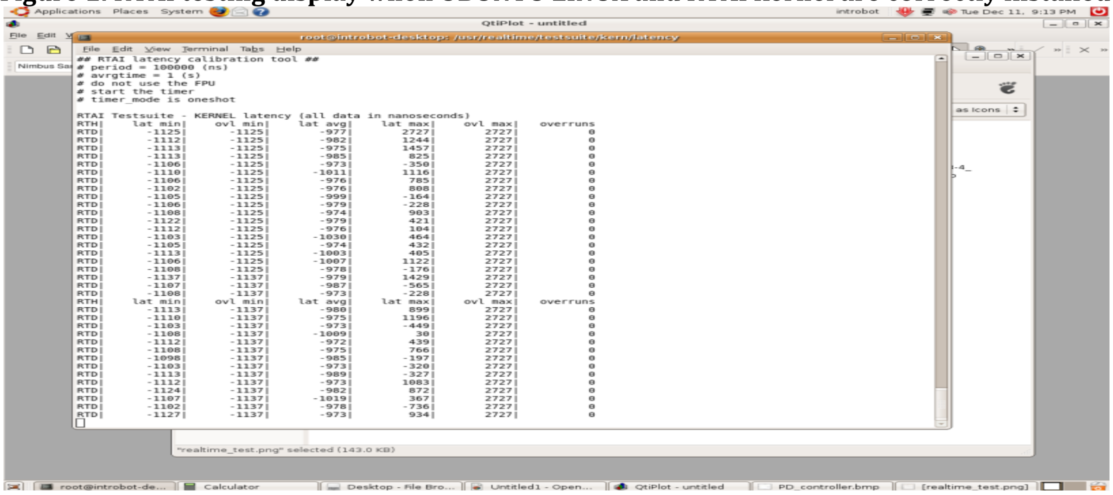
When the UBUNTU LINUX and RTAI kernel are correctly installed, we see the window as shown in Figure 1 which displays RTAI testing results. To communicate with and control hardwares in real-time applications, COMEDI library provides drivers of hardwares and call feature functions of hardwares. The setup files can be downloaded by using commands below.

git clone git://comedi.org/git/comedi/comedi.git
git clone git://comedi.org/git/comedilib.git
git clone git://comedi.org/git/comedi_calibrate.git
git clone git://comedi.org/git/comedi-nonfree-firmware.git

To get more details about setup procedure, we can use the RTAI installation guide (Joan, 2008). After finishing installations of RTAI and COMEDI, we have to connect them to hardware NI PCI 6221 card. The instructions below show that how we can connect them together.

insmod /usr/realtime/modules/rtai_hal.ko
insmod /usr/realtime/modules/rtai_lxrt.ko
modprobe ni_pcimion
comedi_config /dev/comedi0 ni_pcimio

Figure 1: RTAI testing display when UBUNTU LINUX and RTAI kernel are correctly installed



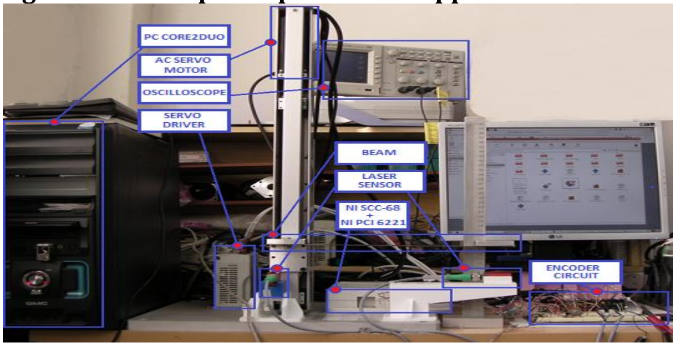
```
## RTAI latency calibration tool ##
# period = 100000 (ns)
# avrgtime = 1 (s)
# do not use the FPU
# start the timer
# timer_mode is oneshot

RTAI Testsuite - KERNEL latency (all data in nanoseconds)
RTH|    lat min|    ovl min|    lat avg|    lat max|    ovl max|   overruns
RTD|      -1125|      -1125|       -977|       2727|       2727|          0
RTD|      -1112|      -1125|       -982|       1244|       2727|          0
RTD|      -1113|      -1125|       -975|       1457|       2727|          0
RTD|      -1113|      -1125|       -985|        825|       2727|          0
RTD|      -1106|      -1125|       -973|       -350|       2727|          0
RTD|      -1110|      -1125|      -1011|       1116|       2727|          0
RTD|      -1106|      -1125|       -976|        785|       2727|          0
RTD|      -1102|      -1125|       -976|        808|       2727|          0
RTD|      -1105|      -1125|       -999|       -164|       2727|          0
RTD|      -1106|      -1125|       -979|       -228|       2727|          0
RTD|      -1108|      -1125|       -974|        903|       2727|          0
RTD|      -1122|      -1125|       -979|        421|       2727|          0
RTD|      -1112|      -1125|       -976|        104|       2727|          0
RTD|      -1103|      -1125|      -1030|        464|       2727|          0
RTD|      -1105|      -1125|       -974|        432|       2727|          0
RTD|      -1113|      -1125|      -1003|        405|       2727|          0
RTD|      -1106|      -1125|      -1007|       1122|       2727|          0
RTD|      -1108|      -1125|       -978|       -176|       2727|          0
RTD|      -1137|      -1137|       -979|       1429|       2727|          0
RTD|      -1107|      -1137|       -987|       -565|       2727|          0
RTD|      -1108|      -1137|       -973|       -228|       2727|          0
RTH|    lat min|    ovl min|    lat avg|    lat max|    ovl max|   overruns
RTD|      -1113|      -1137|       -980|        899|       2727|          0
RTD|      -1110|      -1137|       -975|       1196|       2727|          0
RTD|      -1103|      -1137|       -973|       -449|       2727|          0
RTD|      -1108|      -1137|      -1009|         30|       2727|          0
RTD|      -1112|      -1137|       -972|        439|       2727|          0
RTD|      -1108|      -1137|       -975|        766|       2727|          0
RTD|      -1098|      -1137|       -985|       -197|       2727|          0
RTD|      -1103|      -1137|       -973|       -320|       2727|          0
RTD|      -1113|      -1137|       -989|       -327|       2727|          0
RTD|      -1112|      -1137|       -973|       1083|       2727|          0
RTD|      -1124|      -1137|       -982|        872|       2727|          0
RTD|      -1107|      -1137|      -1019|        367|       2727|          0
RTD|      -1102|      -1137|       -978|       -736|       2727|          0
RTD|      -1127|      -1137|       -973|        934|       2727|          0
```

"realtime_test.png" selected (143.0 KB)

Experimental apparatus: For hands-on experiences for motion control experiments, we developed an easy-to-use and economic experimental apparatus with one degree-of-motion. The developed experimental system is easy to use since it is simple in mechanical structure and it is using a standard operating system, and is economic since it uses free softwares such as UBUNTU LINUX, RTAI and COMEDI. The developed system includes a servomotor and driver, a ball-screw, an NI card, an encoder circuit, two laser sensors, a desktop PC with Core2duo processor, UBUNTU 11.04LTS operating system, RTAI 3.8 module and COMEDI softwares as shown in Figure 2.

Figure 2: Developed experimental apparatus for motion control education



NI data acquisition card is plugged in PC mainboard and connected to NI connector. Analog A0 channel of the NI card is used to transfer analog signal to servo driver. The encoder feedback signal from the AC servo motor is processed with quadrature signal by encoder circuit before transferring to NI card. To detect motion and vibration of the moving parts of the system, two laser sensors are used to measure the displacements of the moving parts.

3. Experimental Demonstrations

To verify the validity of the developed motion control apparatus, we inserts program codes of our control logics into the program file. We only need to change the control logic part of the whole program, and thus you don't need to understand the whole program codes. Our experimental codes are written in C language using RTAI functions and COMEDI functions. To use RTAI and COMEDI functions, we have to add some header files at the top of the codes.

```
#include <rtai.h>
#include <rtai_sched.h>
#include <rtai_lxrt.h>
#include <rtai_msg.h>
#include <comedilib.h>
```

The following code is for generating 0.1 ms sampling time using RTAI for real-time motion control purpose.

```
mlockall(MCL_CURRENT|MCL_FUTURE);
rt_set_periodic_mode();
rt_make_hard_real_time();
start_rt_timer(nano2count(100000));
rt_task_make_periodic( g_stask1, rt_get_time(), nano2count(100000));
```

The first demonstration is P control (proportional control) of the developed motion control apparatus as shown in Figure 3. The plant dynamics is approximately a second-order system with a small damping, but the closed-loop system with a small proportional gain K_p becomes an overdamped system as shown in Figure 4. If we increase the proportional gain K_p further, then it becomes a critically damped system and then underdamped system, which are consistent with our intuitions. The P controller is given by

$$u(t) = K_p e(t), \quad e(t) = r(t) - y(t) \tag{1}$$

where $e(t)$ is an error signal, $r(t)$ is a position command, and $y(t)$ is an actual displacement of the moving part of the up-down motion system. Figure 4 shows step responses for 1 sec when the position command is 5 cm and the proportional gains are 0.5, 2 and 10. The actual code in the program for the P controller (1) is as follows

```
double Kp = 2;
.
.
.
err = ref – yout;
u = Kp*err;
```

Figure 3: Block diagram of the closed-loop control system with a P controller

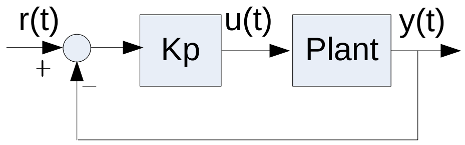
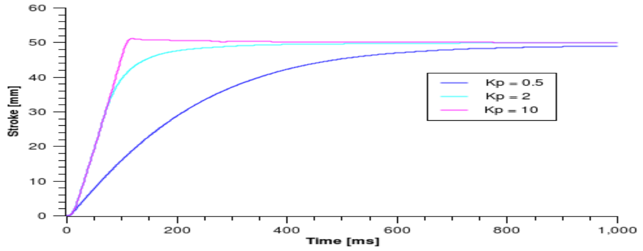


Figure 4: Experimental step responses during 1 sec for P control when the position command is 5 cm and the proportional gains are 0.5, 2 and 10



Next demonstration is PD control (proportional-and-derivative control) of the developed motion control apparatus as shown in Figure 5. The sampling time T_s is 0.1 ms and the PD controller is given by

$$u(t) = K_p * e(t) + K_d * \dot{e}(t) \qquad (2)$$

where $\dot{e}(t)$ is the time derivative of the error signal, and K_d is a derivative gain. Figure 6 shows step responses for 1 sec when the position command is 5 cm and the proportional gains are 0.5, 2 and 10 and the derivative gain is 0.2. The actual code in the program for the PD controller (1) is as follows.

```
double Kp = 2;
double Kd = 0.2;
double Ts = 0.0001;
.
.
.
err = ref – yout;
de = (err – pre_err)/Ts;
pre_err = err;
u = Kp*err +Kd*de;
```

Figure 5: Block diagram of the closed-loop control system with a PD controller

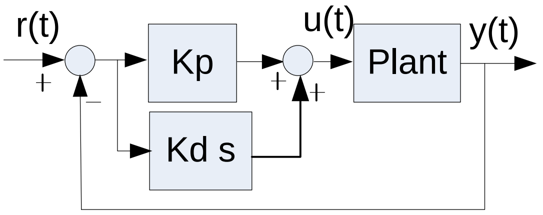
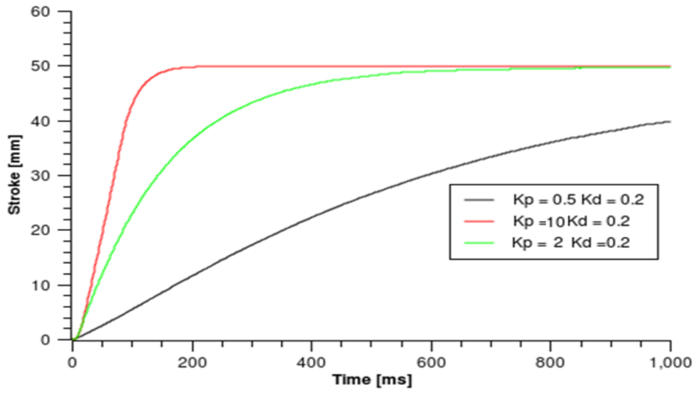


Figure 6: Experimental step responses during 1 sec for PD control when the position command is 5 cm and the proportional gains are 0.5, 2 and 10, and the derivative gain is 0.2.



The final demonstration is residual vibration control using a ZV (Zero Vibration) input shaper. When we attach a flexible beam on top of the moving part of the up-down system as shown in Figure 7, residual vibration will occur when we move the moving part in a step fashion. Figure 8 shows the block diagram of the input shaping control system, and Figure 9 shows the experimental results measured by a laser sensor at the end of the flexible beam. The laser sensor is put on the table and measures the distance between the end of the beam and the table. By adding a ZV input shaper, we see the residual vibration has been removed in Figure 9. The ZV input shaper is given by

$$r_{IS}(t) = A_1\delta(t) + A_2\delta(t-t_2) \qquad (3)$$

$$\begin{bmatrix} t_i \\ A_i \end{bmatrix} = \begin{bmatrix} 0, & \pi/\omega_n \\ 1/(1+K), & K/(1+K) \end{bmatrix}, \quad K = e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}}$$

where $r_{IS}(t)$ is input-shaped reference signal, $\delta(t)$ is Dirac delta function, A_1 and A_2 are impulse magnitudes, t_2 is the time location of the second impulse, ω_n is the natural frequency of the flexible beam, and ζ is the damping ratio of the flexible beam. In the present case, $\omega_n = 5.04$ rad/s and $\zeta = 0.012$. The actual code in the program for the ZV input shaper is as follows.

```
double zeta = 0.012;
double wn = 5.04;
double pi = 3.1416;
double K = exp(-zeta*pi/sqrt(1-zeta^2));
double A1 = 1/(1 + K));
double A2 = K/(1 + K);
double t2 = pi/wn;

double Kp = 2;
double Kd = 0.2;
double Ts = 0.0001;
.
.
.
if( 0<t  &&  t<t2)
{
r_IS = ref*A1;
}
if(t2<t)
{
r_IS = ref; // A1+A2=ref
}
.
.
.
err = r_IS – yout;
de = (err – pre_err)/Ts;
pre_err = err;
u = Kp*err +Kd*de;
```

Figure 7: Experimental apparatus with a flexible beam on top of the moving part of the system

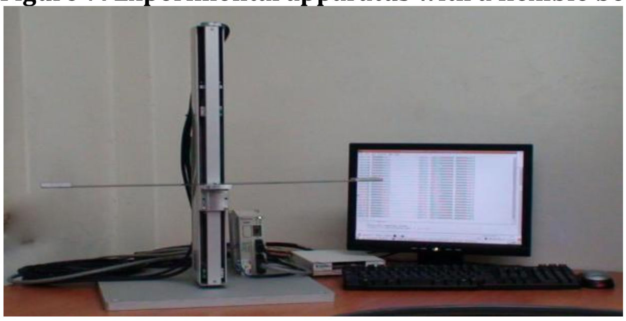


Figure 8: Block diagram of an input shaping control system

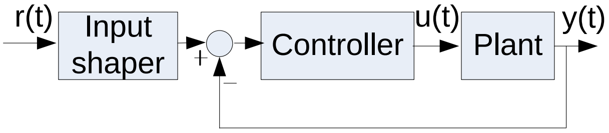
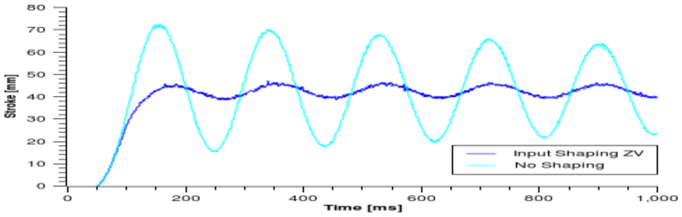


Figure 9: Experimental step responses of the end point of the flexible beam with and without a ZV input shaper



From three experimental demonstrations, we have shown that the developed motion control apparatus is easy-to-use and economic in cost for motion control educations. With this experimental apparatus, users can easily change appropriate values for controller parameters by directly changing a part of the codes and recompiling. Furthermore, users can test various kinds of their control logics by modifying a part of the control logic codes. The developed apparatus has been used for teaching feedback control and input shaping control to undergraduate and graduate students at Konkuk University in Seoul, Korea for last two semesters, and it was very effective to teach motion control concept using the developed apparatus. The students were interested in doing experiments using the apparatus.

4. Conclusion

The open-source operating system LINUX combined with RTAI module and COMEDI library is very useful when we want to design a cheap real-time control system. This paper has shown how to build real-time experimental apparatus using RTAI and LINUX operating system. In this paper, we develop an easy-to-use and economic experimental apparatus for motion control education using an open-source free operation system LINUX and a free real-time kernel RTAI. The developed apparatus is reliable in mechanical structure, and fast enough to realize 100 microsecond sampling time for real-time control purpose. Furthermore we have shown this apparatus can be used successfully for motion control experiments in real-time. The validity of the developed motion control apparatus is demonstrated by testing P, PD control logics and ZV input shapers experimentally.

Acknowledgement: This work was supported by Mid-career Researcher Program through National Research Foundation grant (No. 20120005128) funded by the Ministry of Education, Science and Technology of Korea.

References

Anju, M. A. M. K. (2008). Development of a Laboratory Model of SSSC Using RTAI on Linux Platform. *Sadhana*, 33(5), 643-661.

Joan, M. (2008). RTAI Installation Complete Guide. Available at https://www.rtai.org/RTAICONTRIB/RTAI_Installation_Guide.pdf

Kwak, J. H. & Kang, C. G. (2010). Initialization and Safety of Motion Control System Using a Magnetic Sensor. Proc. 7th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI2010), Nov. 24-27, Busan, Korea. pp. 99-101.

Matej, D. & Martin, K. (2008). Control an Electromechanical System in the Real-Time Linux Environment. *Acta Polytechnica Hungarica*, 4(2).

Michele, B., Robeto, B., Marco, R. & Massimo, V. (2005). Real-Time Control with Linux: A Web Services Approach. Proc. 44th IEEE Conference on Decision and Control, and the European Control Conference, Seville, Spain, December 12-15.