เอกสารอ้างอิง

1. Yodaiken, V., and Barabanov, M. A Real-Time Linux. ftp://rtlinux.cs.nmt.edu/pub/rtlinux/papers/usenix.ps.gz [URL]. New Mexico: New Mexico Institute of Technology, March 3, 1996.
2. Crum, B. (Bill) Real-Time Systems. http://rtlinux.cs.nmt.edu/~tobor/jwz/rtpap.ps [URL]. New Mexico: New Mexico Institute of Technology, April 20, 1996.
3. Yodaiken, V., and Barabanov, M. An Introduction to Real-Time Linux. ftp://rtlinux.cs.nmt.edu/pub/rtlinux/papers/lj.ps.gz [URL]. Linux Journal: 34 (1997): 19.
4. Barabanov, M. A Linux-based Real-Time Operating System. http://rtlinux.cs.nmt.edu/~baraban/thesis/ [URL]. Master's Thesis, New Mexico Institute of Technology, June 1, 1997.
5. Froctor, F. (Fred) Using Shared Menory in RT-Linux. http://isd.cme.nist.gov/proj/emc/shmem.html [URL].
6. Kiesling, R. (Robert) Linux Frequently Asked Questions with Answers. http://www.cl.cam.ac.uk/users/iwj10/linux-faq/index.html [URL].
7. Johnson, M.K. Linux Kernel Hackers' Guide. http://www.redhat.com:8080/HyperNews/get/khg.html [URL].
8. Kuhn, M. (Markus) A vision for Linux 2.2 - POSIX.1b Compatibility and Real-Time Support. ftp://ftp.informatik.uni-erlangen.de/local/cip/mskuhn/misc/linux-posix.1b [URL]. July 9, 1997.
9. Bennett, S. Real-time Computer Control: an introduction. 2nd ed. UK: Prentice Hall, 1994.
10. Levi, S.-T., and Agrawala, A.K. Real-Time System Design. Singapore: McGraw-Hill, 1990.
11. Middleton, R.H., and Goodwin, G.C. Digital Control and Estimation A unified Approach. New York: Prentice-Hall, 1990.
12. Åström, K., and Wittenmark, B. Computer-Controlled Systems Theory and Design. 2nd ed. New York: Prentice-Hall, 1990.
13. The Linux Home Page. http://www.linux.org/ [URL].
14. The Real-Time Linux Home Page. http://rtlinux.cs.nmt.edu/ [URL].

15. Epplin, J. Semaphore Implementation. http://stereotaxis.wustl.edu/~jerry [URL].
16. The Real-Time Linux Mailing List
http://rtlinux.cs.nmt.edu/~rtlinux/mail_arch/maillist.html [URL].
17. Siering, T.C. The Real-Time Registry.
ftp://sokrates.ipk.fhg.de/pub/rtlinux/rt_registry/rt_registry-0.1.tar.gz [URL].
18. Mantegazza, P. mailto: mantegazza@aero.polimi.it [URL].
19. Siering, T.C. The Fixed-Point Arithmatic Library.
http://rtlinux.cs.nmt.edu/~rtlinux/contrib/rt_fixed-0.1b.tar.gz [URL].
20. ZzzThai Project. http://www.fedu.uec.ac.jp/ZzzThai [URL]. University of Electro-Communications, Tokyo.
21. Thai Tex. http://thaigate.nacsis.ac.jp/files/ttex.html [URL].
22. NECTEC. The Thailand Big Picture Home Page. http://www.nectec.or.th/ [URL].

ภาคผนวก ก
การติดตั้ง RT-Linux

1. นำไฟล์ rtlinux-0.V-2.0.XX.tar.gz จาก [14] ซึ่งเป็นโปรแกรมรุ่น 0.V (ล่าสุด) และใช้กับ kernel รุ่น 2.0.XX มาขยายออก ในไฟล์จะประกอบด้วย ไฟล์ kernel_patch ไฟล์คู่มือห การใช้ และตัวอย่างโปรแกรม

   # tar –xzvf rtlinux-0.V-2.0.XX.tar.gz
2. patch ไฟล์ kernel_patch เข้ากับ kernel รุ่นที่ระบุคือ 2.0.XX โดยคำสั่ง

   # cd <directory of kernel source>

   # patch -p1 < /?PATH?/kernel_patch
3. แปลโปรแกรมของ kernel

   # make config หรือ make menuconfig หรือ make xconfig

   # make dep

   # make clean

   # make zImage

   # make modules

   # make modules_install
4. ติดตั้ง kernel ที่ได้ใหม่ (zImage) และทำการบูตเครื่อง
5. บรรจุมอดูลต่อไปนี้เข้าสู่ระบบ

   # insmod rt_prio_sched

   # insmod rt_fifo_new
6. สร้าง device ชื่อ rtf0 rtf1 ... rtf63 ไว้สำหรับใช้งานตามที่ต้องการ

   # for i in 0 1 2 3; do mknod /dev/rtf$i c 63 $i; done
7. ทดสอบการใช้งานจากโปรแกรมตัวอย่างที่มาพร้อมกับไฟล์ rtlinux-0.V-2.0.XX.tar.gz

ภาคผนวก ข
ไลบรารีสำหรับระบบควบคุม
แบบพร็อพเพอร์ (proper)

proper.h

```
/*
 *  Header file for Strickly Proper or Proper Control Law
 */
/* For RT-linux */
#include <linux/rt_sched.h>
#include <linux/rtf.h>
#include <asm/rt_irq.h>
/*#include <linux/cons.h>*/

/* Uncomment below line if your control law is strickly proper */
/* #define STRICKLY */
/* Uncomment below line if you have partial-update of control signal */
/* #define PRECOMPUTE */

/* define FIFO to use for Operation and Status */
#define OperFifo        0
#define StatFifo        1
#define OperFifoSize  128
#define StatFifoSize   256
/* define Timing value */
#define CtrlPeriod     10000          /* usec - 100 Hz */
#define OperPeriod   200000           /* usec -   5 Hz */
#define StatPeriod   250000           /* usec -   4 Hz */
#define Period2Tick(x)          (RT_TICKS_PER_SEC * (x))/1000000
#define CtrlTick        Period2Tick(CtrlPeriod)
#define OperTick        Period2Tick(OperPeriod)
#define StatTick        Period2Tick(StatPeriod)
#define CtrlPrio        110
#define OperPrio        120
#define StatPrio        130
```

```
/* Must edit struct State and Command */
struct Command {
    int code;
    float arg1;
};
struct State {
    float SV,PV,MV,DV;
};

/* Not necessary to edit these lines below. */
#ifdef PROPER

#define MODULE
#include <linux/module.h>
/* #include <linux/version.h> */
#include <linux/kernel.h>
extern void init_param(void);
extern void read_in_data(void);
extern void write_out_data(void);
extern void pre_compute_control_law(void);
extern void compute_control_law(void);
extern void do_command(void);

#else /* PROPER */

#include <linux/kernel.h>
extern void ctrl_start(float period);
extern void stat_start(float period);
extern void ctrl_stop(void);
extern void stat_stop(void);

#endif /* PROPER */
/* End File */
```

proper.c

```
/*
 *   Routine for Strickly Proper or Proper Control Law
 */
#define PROPER
#define MODULE
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/version.h>
/* For RT-linux */
/*
#include <linux/rt_sched.h>
#include <linux/rtf.h>
#include <asm/rt_irq.h>
#include <linux/cons.h>*/

#include "proper.h"

RT_TASK CtrlTask;
RT_TASK OperTask;
RT_TASK StatTask;
/*
 *   Proper Control Law
 */
void ctrl_fun(int t)
{
    rt_use_fp(1);

    init_param();
    while(1){
            read_in_data();
#ifdef STRICTLY
            write_out_data();
            compute_control_law();
#else
            compute_control_law();
```

```
                write_out_data();
#endif
#ifdef PRECOMPUTE
                pre_compute_control_law();
#endif
                rt_task_wait();
        }
}
/*
 *   Operator Communication
 */
void oper_fun(int t)
{
extern struct Command command;
        rt_use_fp(1);
        while(1){
            if(rtf_get(OperFifo,&command,sizeof(command))==sizeof(command)) {
                do_command();
            }
            rt_task_wait();
        }
}
/*
 *   Status of process
 */
void stat_fun(int t)
{
extern struct State state;
        rt_use_fp(1);
        while(1){
                rtf_put(StatFifo,&state,sizeof(state));
                rt_task_wait();
        }
}
/*
 *   Initialize of Module
 */
```

```
int init_module(void)
{
    printk("Initialize PID module ... ");

    rtf_create(OperFifo,OperFifoSize);
    rtf_create(StatFifo,StatFifoSize);

/*  rt_task_init(RT_TASK *,*fn,int t, stack, priority) */
    rt_task_init(&OperTask, oper_fun, 1, 5000, OperPrio);
    rt_task_init(&CtrlTask, ctrl_fun, 1, 5000, CtrlPrio);
    rt_task_init(&StatTask, stat_fun, 1, 3000, StatPrio);

    rt_task_make_periodic(&OperTask,rt_get_time()+1000,OperTick);

    printk("Done.\n");
    return(0);
}
/*
 *  Clean-up
 */
void cleanup_module(void)
{
    printk("Removing PID module ... ");

    rt_task_delete(&OperTask);
    rt_task_delete(&CtrlTask);
    rt_task_delete(&StatTask);

    rtf_destroy(OperFifo);
    rtf_destroy(StatFifo);

    printk("Done\n");
}
/*
 *  Utilities Function
 */
inline void ctrl_start(float period)
```

```
{
    rt_task_make_periodic(&CtrlTask,rt_get_time()+100,Period2Tick(period));
}
inline void ctrl_stop(void)
{
    rt_task_suspend(&CtrlTask);
}
inline void stat_start(float period)
{
    rt_task_make_periodic(&StatTask,rt_get_time()+100,Period2Tick(period));
}
inline void stat_stop(void)
{
    rt_task_suspend(&StatTask);
}
```

Makefile

```
# Makefile for Sample Control Algorithm
# Strickly proper or Proper
CFLAG = -O2 -Wall
EXTRA_CFLAG = -D__KERNEL__ -D__RT__
RTL = /usr/src/linux
# Add new controller here
all: proper.o yourfile.o
# PID Controller (Example)
proper.o: proper.c proper.h
    gcc ${CFLAG} ${EXTRA_CFLAG} -c proper.c
%.o: %.c %.h
    gcc ${CFLAG} ${EXTRA_CFLAG} -c -o $*_tmp.o $<
    ld -r -static -o $@ proper.o $*_tmp.o
    rm -f $*_tmp.o
    @echo "-> Got the controller <$*>"
# For Cleaning
clean:
    rm -f *.o
```

ภาคผนวก ค
ตัวอย่างการใช้งานไลบรารี
สำหรับตัวควบคุมแบบพร็อพเพอร์

ตัวอย่างการใช้งานเป็นตัวควบคุมแบบพีไอดีซึ่งดัดแปลงมาจากกรณีศึกษา

pid.h

```
#define     N       10      /* Parameter for Low Pass Filter */

#define     INCH    2       /* input from ch 0x0 to 0xf */
#define     OUTCH   2       /* output to ch 1 or 2 only */
```

pid.c

```
/*
 *   PID-Controller for Heat Exchanger.
 */

#include "../adda/pclabrt.h"
#include "proper.h"
#include "pid.h"

/* Global Parameter */
static      float   Pb,Ti,Td,Ts;
static      float   SV,PV,MV,DV,PV_1;
static      float   K,P,Iold,D,Dold;

/*
 *   Control-Loop Function
 */
void init_param(void)
{
    Pb=100;Ti=100;Td=0;
    Iold=Dold=PV_1=0;
```

```
    Ts=CtrlPeriod/1000000.0; /* sec */
}
void read_in_data(void){
    PV = itov(adin(INCH));
}
void compute_control_law(void){
    float Inew;

    DV = SV - PV;
    K = 100.0/Pb;
    P = K*DV;
    D = Td/(Td+N*Ts)*Dold - K*Td*N/(Td+N*Ts)*(PV-PV_1);
    if(Ti<100) {
            Inew = K*(Ts/Ti)*DV;
            MV = P + Inew + Iold + D;
            Iold += Inew;
    } else if(Td>0) {
            MV = P + D;
    } else {
            MV = P ;
    }
    if( MV <= 0.0 ){
            MV = 0.0;
    } else if( MV >= 5.0 ){
            MV = 5.0;
    }
    Dold = D;
    PV_1 = PV;
}

struct State state;
void write_out_data(void){
    daout(OUTCH,vtoi(MV));
}
/*
 *  Operation to ...
 *   - update set point
```

```
 *    - PID parameter
 */
struct Command command;
/*
     code
     - toggle stop/start control task  (period)
     - toggle stop/start display task  (period)
     - update set value (SV)
     - set (Pb)
     - set (Ti)
     - set (Td)
*/
void do_command(void)
{
    switch(command.code){
       case 0:
            if(!command.arg1) command.arg1=CtrlTick;
            ctrl_start(command.arg1);
            Iold=Dold=0;
            break;
       case 1:
            ctrl_stop();
            break;
       case 2:
            if(!command.arg1) command.arg1=StatTick;
            stat_start(command.arg1);
            break;
       case 3:
            stat_stop();
            break;
       case 4:
            /* SV=command.arg1; */
            SV=vtoi(command.arg1);
            break;
       case 5:
            Pb=command.arg1;
            break;
```

```
            case 6:
                Ti=command.arg1;
                break;
            case 7:
                Td=command.arg1;
                break;
            default:
                printk("Invalid Command<%d>.\n",command.code);
                break;
        }
    }
```

Makefile

```
# Makefile for Sample Control Algorithm
# Strickly proper or Proper

CFLAG = -O2 -Wall
EXTRA_CFLAG = -D__KERNEL__ -D__RT__
RTL = /usr/src/linux
# Add new controller here
all: proper.o pid.o

# PID Controller (Example)
proper.o: proper.c proper.h
    gcc ${CFLAG} ${EXTRA_CFLAG} -c proper.c

%.o: %.c %.h
    gcc ${CFLAG} ${EXTRA_CFLAG} -c -o $*_tmp.o $<
    ld -r -static -o $@ proper.o $*_tmp.o
    rm -f $*_tmp.o
    @echo "-> Got the controller <$*>"

# For Cleaning
clean:
    rm -f *.o
```

ประวัติผู้เขียน

นายณัฐ ธีรศุทธากร เกิดวันที่ 3 กุมภาพันธ์ พ.ศ. 2518 ที่จังหวัดกรุงเทพมหานคร เป็นบุตรของนายไพฑูรย์ และนางสุดสวาท ธีรศุทธากร สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย ในปี พ.ศ. 2538 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ภาควิชาวิศวกรรมไฟฟ้า สาขาระบบควบคุม ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. 2539