2001 추계학술발표회 논문집
한국원자력학회

표준 PC 하드웨어와 리눅스 환경을 이용한 원자력 시뮬레이터 소프트웨어의 실시간 컴퓨터 시뮬레이션

A Real-Time Computer Simulation of Nuclear Simulator software Using Standard PC Hardware and Linux Environments

차 경호, 권 기춘

한국원자력연구소
대전광역시 유성구 덕진동 150

요 약

본 논문은 표준 PC 하드웨어와 공개 소스 소프트웨어 (Open Source Software: OSS)인 리눅스 (Linux, Real-Time Linux: RTLinux) 환경에서 원자력 시뮬레이터를 위한 소프트웨어의 실시간 컴퓨터 시뮬레이션에 관해 기술한다. 축소형 원자력 시뮬레이터의 소프트웨어를 이용하여 실시간 컴퓨터 시뮬레이션의 Feasibility Prototype 으로 구축하였다. 표준 PC 하드웨어와 실시간 리눅스 환경에서 Feasibility Prototype 의 구현 결과, 적은 노력으로 좋은 실시간 시뮬레이션 성능을 얻을 수 있었으며 컴퓨터-기반 예측 시뮬레이션이 가능할 수 있다.

Abstract

A feasibility study, which standard PC hardware and Real-Time Linux are applied to real-time computer simulation of software for a nuclear simulator, is presented in this paper. The feasibility prototype was established with the existing software in the Compact Nuclear Simulator (CNS). Throughout the real-time implementation in the feasibility prototype, we has identified that the approach can enable the computer-based predictive simulation to be approached, due to both the remarkable improvement in real-time performance and the less efforts for real-time implementation under standard PC hardware and Real-Time Linux environments.

1. 서론

Full-scope Simulator, Compact Simulator, Part-Task (function) Simulator, Engineering Simulator 등 원자력 시뮬레이터 대부분이 컴퓨터 시스템에 의한 시뮬레이션 프로그램을 이용한다. 한편, 실시간 응용에는 실시간 운영체제가 널리 이용되어 왔다.

공개 소스 소프트웨어 (Open Source Software: OSS)가 In-core 모니터링 시스템 (SVRK-M) 개발에 이용되면서[1] 점차 그 응용이 늘어날 것으로 예측된다. 이러한 추세를 반영하여 표준 PC 하드웨어와 실시간 리눅스 (Real-Time Linux: RTLinux)[2]를 통해 실시간 컴퓨터 시뮬레이션에 대한 성능을 고찰하고자 한다.

실시간 컴퓨터 시뮬레이션은 현재 한국원자력연구소에서 계통교육에 사용되는 축

소형 시뮬레이터(Compact Nuclear Simulator: CNS)의 시뮬레이션 소프트웨어와 그래픽 소프트웨어에 대하여 표준 PC 하드웨어와 실시간 리눅스 환경으로 이식하여 Feasibility Prototype 으로 활용한다.

2. 시뮬레이션 환경

2.1 소프트웨어

상용 실시간 운영체제 (Commercial Real-Time Operating System)는 대부분 가격이 비싸기 때문에, 연구실 단위의 소규모 연구에서 사용하는데는 제약이 따른다. 그러므로, 이러한 제약이 없는 실시간 리눅스를 이용하여 원자력 시뮬레이터의 소프트웨어에 대한 실시간 성능을 구현하는 것은 그 노력과 비용 측면에서 만족할만 하다.

표준 PC 하드웨어와 비공개 소스 소프트웨어인 리눅스/실시간 리눅스는 이전에 워크스테이션 컴퓨터와 UNIX 운영체제에서 개발되어온 시뮬레이터 소프트웨어를 재사용하는데 장점을 갖는다. 즉, 리눅스가 UNIX 운영체제와 호환적이기 때문에 UNIX 운영체제 환경에서 개발된 소프트웨어들은 쉽게 리눅스 환경에 이식할 수 있다. 한편, UNIX 환경의 X-Window System 과 네트워킹에서도 호환성도 높다. 그러므로, HP 워크스테이션 하드웨어, UNIX, X-Window System, Ethernet (TCP/IP) 환경의 시뮬레이터 소프트웨어를 PC 하드웨어와 리눅스/실시간 리눅스 환경에서 실시간 구현이 쉽다.

축소형 시뮬레이터(CNS)의 **시뮬레이션 소프트웨어**는 가압경수로형 발전소의 정적 및 동적 특성(static and dynamic behaviors)을 시뮬레이션 한다. 최근, HP 워크스테이션을 시뮬레이션 호스트 컴퓨터, X-터미널, Ethernet (TCP/IP), 프로그래머블 논리제어기 (Programmable Logic Controller: PLC)를 이용하여 축소형 시뮬레이터(CNS)의 성능이 개선되었으며[3], 운용중인 축소형 시뮬레이터의 전경은 [그림-1]과 같다.

[그림-1] 축소형 시뮬레이터 전경

축소형 시뮬레이터는 75 개의 고장함수(malfunctions)가 모델링 되어 시뮬레이터 제어명령어에 의한 고장함수의 주입(injection)이 가능하다. 그리고, X-Window System 을 이용하여 개발된 Picasso 그래픽 도구를 디스플레이 개발에 이용하였다. 이렇게 개발된 그래픽 디스플레이는 정적 및 동적 그래프들로 제공된다. 현재, 이러한 제어 반은 구현범위에 포함되지 않았다.

구 분	자 원 목 록
개발환경 소프트웨어	Linux Kernel 2.2.14/RTL 2.2 GNU gcc 2.7 컴파일러 & f2c Picasso-3 R2.3 (Linux 버전)
시뮬레이션 프로그램	GNU C 로 구현된 CNS 프로그램 (Linux 버전)
그래픽 프로그램	CNS 의 그래픽 프로그램 (Linux 버전)

[표-1] 실시간 시뮬레이션 구현에 사용된 소프트웨어

2.2 하드웨어

실시간 시뮬레이션 구현에 사용된 PC 하드웨어는 [표-2]의 목록과 같다. 성능 측면을 고려한다면 Dual CPU 를 갖는 GHz 급 PC 를 사용하면 된다.

구 분	자 원 목 록
PC 하드웨어	500 MHz Intel processor (CPU) 64 MB memory/2 GB hard-disk 20” color monitor Ethernet LAN card (TCP/IP)

[표-2] 실시간 시뮬레이션 구현에 사용된 PC 하드웨어

3 구현

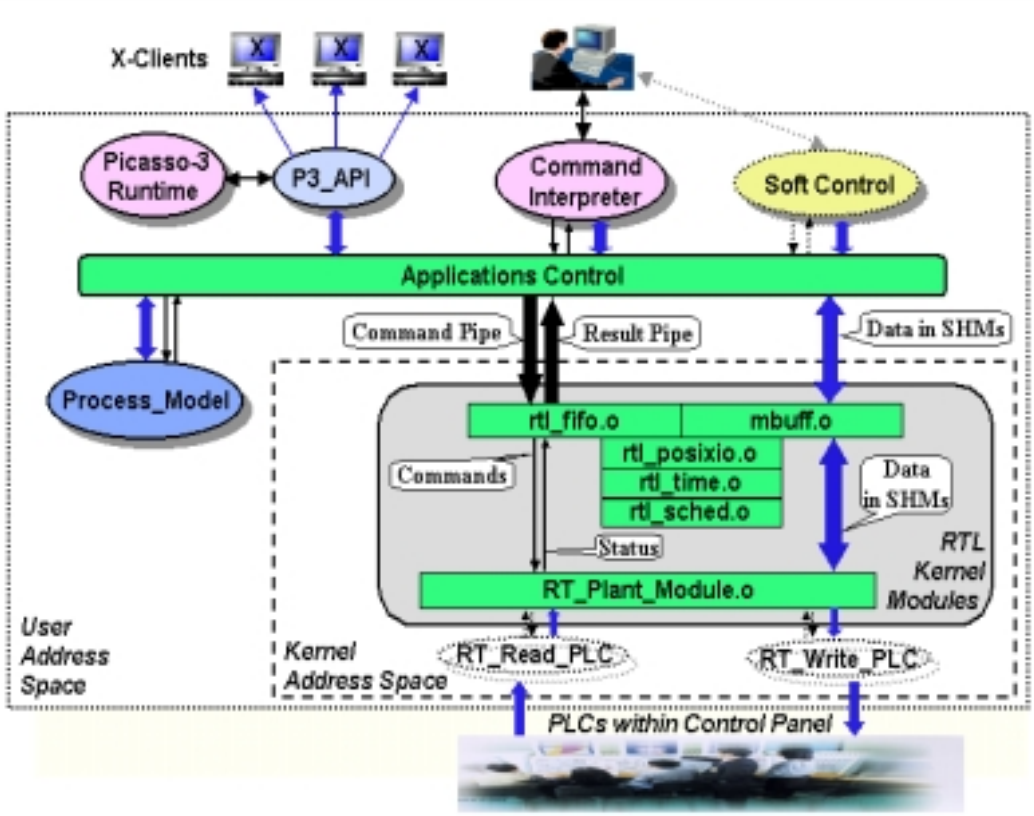
실시간 리눅스 모듈들은 “C” 언어로 구현되었다. 이러한 프로그래밍 언어 환경에 따라 축소형 시뮬레이터에 사용되는 HP-FORTRAN 소스 프로그램을 가능한 “C” 언어로 변환하여 구현한다. 이러한 언어변환은 이전에 구현된 Linux 버전을 실시간 리눅스 환경에 이식한다.

실시간 제어는 실시간 리눅스 모듈을 이용하여 구현된다. 실시간 FIFO 및 scheduler, mbuff 등 미리 제공되는 기능을 이용하여 시뮬레이션 프로그램을 제어하는 모듈로 구현한다 (“RT_Plant_Module.o”). 실시간 시뮬레이션 구현에는 Linux 2.2.14 커널에 patch 한 RTL 2.2 를 사용하였으며, 그래픽 디스플레이를 위해 Picasso-3 Runtime 을 이용하였다.

그래픽 프로그램은 mbuff 에 정의된 시뮬레이션 변수 값을 읽어서(READ ONLY) 해당 디스플레이 변수에 assign 한다. 한편, “Real-Time FIFO”를 이용하여 시뮬레이션 제어 명령어에 대한 인터페이스를 구현한다.

그리고, 소프트 제어 패널의 구현가능성을 위하여 시뮬레이션 제어명령어 일부를

대상으로 그래픽 인터페이스를 구현한다. 시스템 구조는 [그림-2]와 같이 구성되며, 현재 컴퓨터-기반 시뮬레이터를 목적으로 하기 때문에 점선으로 표시된 PLC 인터페이스는 구현범위에 해당하지 않는다. 필요한 경우 추가 구현으로 충분하다.



[그림-2] 시스템 구조

3.1 실시간 프로세스와 리눅스 프로세스 통신

RTLinux는 특정한 주소공간의 메모리를 설정할 수 있다. 이는 실시간 리눅스 모듈의 "mbuff"에 해당하며[5], 시뮬레이션 변수들은 mbuff에 할당한다. 이는 리눅스 프로세스로부터 "mbuff"에 할당된 변수를 직접 접근하는 것을 제한할 수 있다.

아래의 프로그램은 정수형 변수와 실수형 변수에 대하여 각각 1MB 크기의 시뮬레이션 공유 메모리 "cns"와 "smabre"를 할당하고, 각각의 공유 메모리 접근은 pointer(shm1, shm2)를 통해 이루어진다.

```
/* the allocation and deallocation of shared memory */
#include <files>

volatile int* shm1;
volatile float* shm2;

shm1 = (volatile int*) mbuff_alloc("cns",1024*1024);
shm2 = (volatile float*) mbuff_alloc("smabre",1024*1024);
```

```
…….…….. (program segments);

mbuff_free("cns",(int*)shm1);
mbuff_free("smabre",(float*)shm2);
```

실시간 프로세스와 리눅스 프로세스간 통신은 IPC와 mbuff를 통해 가능하다. 이를 위해 "rtl_fifo"가 사용된다. 시뮬레이터 프로그램에서는 이러한 기능을 통해 시뮬레이터 제어명령어를 정의함으로써 프로세스간 통신을 구현한다. 구현은 FREEZE와 RUN 명령어를 대상으로 fifo를 통한 시뮬레이션 제어가 수행된다.

실시간 리눅스는 실시간 디바이스 파일(예: /dev/rtf1)이 사용되고 실시간 프로세스의 activation 및 deactivation을 위해 실시간 시그널 (#32-#63)이 사용된다. RUN, FREEZE 이외의 다른 명령어들도 필요에 따라 확장하면 된다.

3.2 스케줄링 (Task Scheduling)

일정한 시간 (time period, 예: *msg.period = 100000 nanoseconds*)를 갖는 task 는 "*pthread_make_periodic_np()*" 함수를 사용하여, 임의시간의 주기를 정의할 수 있다. 시뮬레이션 cycle time 을 nanosecond(s) 단위로 설정할 수 있다. 다음 프로그램은 FREEZE(task[0]), RUN(task[1])에 대한 모듈 프로그램의 일부를 나타낸다.

```
/* RT_Plant_Module.c */

#include <head_files>
pthread_t tasks[2];
static char *data[] = {"FREEZE", "RUN     "};

void *thread_code(void *t)
{
  …
  while (1) {
  …
  if ((err = rtf_get ()) == sizeof(msg)) {
       switch (msg.command) {
          case START_TASK:
              pthread_setfp_np(tasks[0], 1);
              pthread_setfp_np(tasks[1], 1);
              pthread_make_periodic_np(pthread_self(), gethrtime(), msg.period * 1000);
              pthread_make_periodic_np(pthread_self(), gethrtime(), msg.period * 1000);
              break;
           default:
              rtl_printf("RTL task: bad command\n");
              return 0;
           }
        }
        rtf_put(fifo, data[fifo - 1], 6);
        }
    return 0;
  }
```

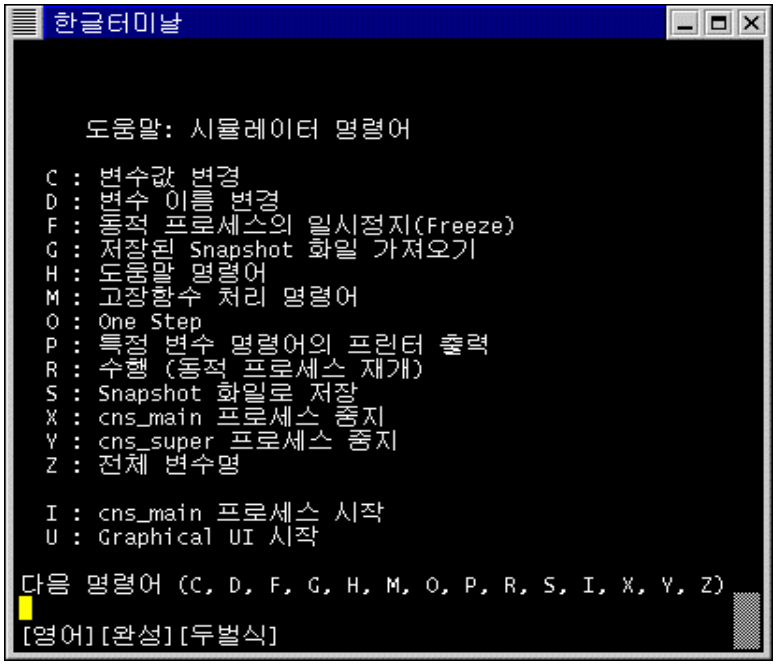
3.3 Float 형 계산

시뮬레이션 프로그램에서는 다음과 같이 각 task 에 대하여 “float 형” 변수 계산이 지원된다.

pthread_setfp_np(tasks[0], 1);
pthread_setfp_np(tasks[1], 1);

3.4 제어명령어

시뮬레이션 제어명령어 인터페이스는 한글로 구현한다. 시뮬레이션 제어 명령어는 가능한 축소형 시뮬레이터에서 사용되는 것과 호환되게 구현되었으며, 시뮬레이션 프로세스 및 그래픽 디스플레이 프로세스를 수행(시작)하는 명령어를 Linux 의 “shell” command 로 구현하였다. [그림-3]은 이러한 명령어 인터페이스를 나타낸다.



[그림-3] 명령어 인터페이스

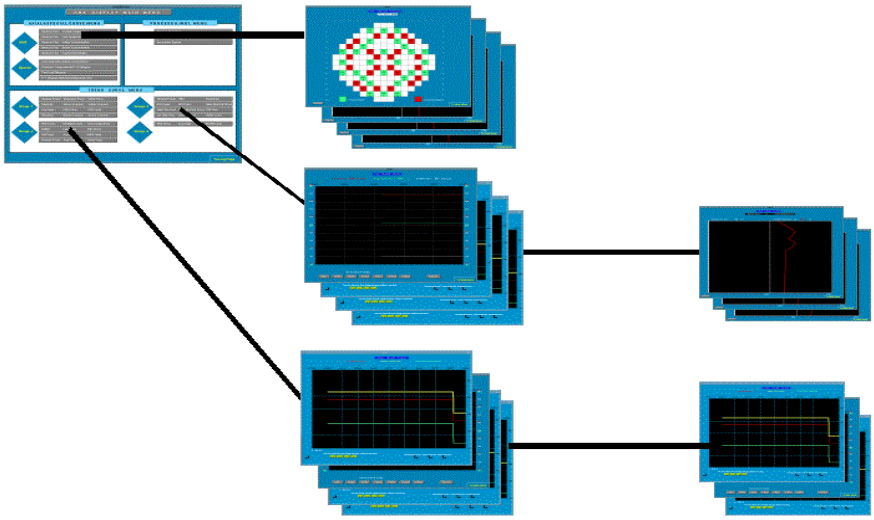
3.5 초기조건

시뮬레이션 프로세스의 초기조건은 현재 한국원자력연구소 원자력연수원에서 운용중인 축소형 시뮬레이터의 초기조건을 그대로 사용한다. 그리고, 축소형 시뮬레이터에서 생성된 snapshot 데이터 (filename.snp)들은 “G(et)” 명령어를 사용하여 시뮬레이션 변수들을 초기화할 수 있으며, 시뮬레이션 수행 도중에 “S(napshot)” 명령어를 사용하여 snapshot 파일 데이터로 저장이 가능하다.

3.6 그래픽 디스플레이 (Simulated Process Monitoring)

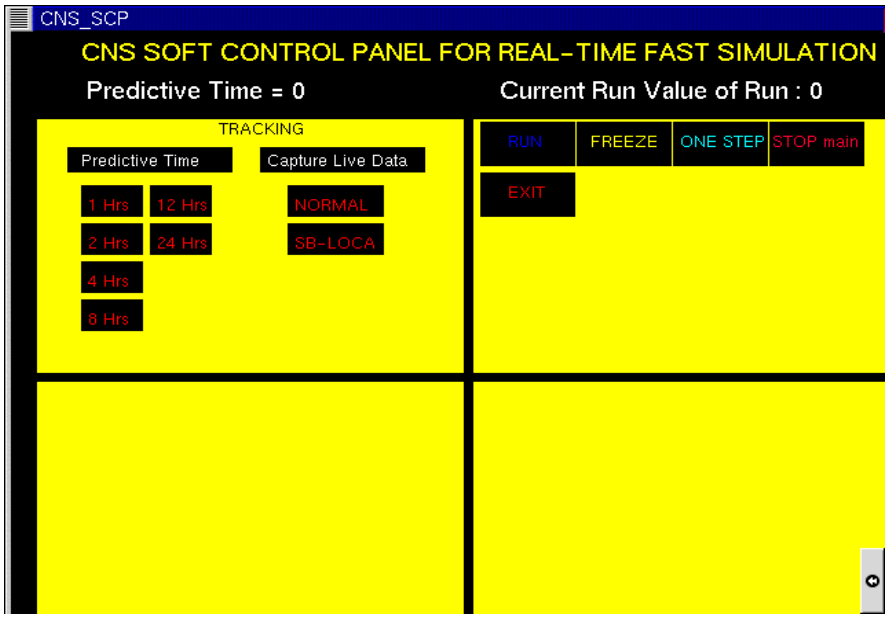
축소형 시뮬레이터를 위해 개발된 정적 및 동적 정보의 그래픽 소스 프로그램을

Linux 버전의 Picasso-3 Runtime 을 이용하여 compilation 함으로써 재사용이 가능하다 (명령어: %gcc <filename.Tdoc>). 이 때, 그래픽 디스플레이를 위한 메모리 접근(access)은 Real-Time FIFO 를 통해 mbuff 접근이 가능하도록 프로그램을 수정한다. [그림-4]는 Linux 버전의 Picasso-3 Runtime 을 이용하여 구축된 그래픽 디스플레이를 나타낸다.



[그림-4] Picasso-3 Runtime 에 의한 그래픽 디스플레이 (정적, 동적정보)

한편, 사용상 편의를 위해 시뮬레이션 제어명령어에 대한 그래픽 구현이 추가되고 있다. 이는 운전제어반에 대한 그래픽 표현으로 소프트 제어반을 구현하는 경우에 도움이 될 수 있다. [그림-5]는 Picasso-3 Runtime 을 이용하여 구현 중에 있는 소프트 제어 인터페이스를 나타낸다.



[그림-5] Picasso-3 Runtime 을 이용한 소프트 제어 명령어

3.7 실시간 성능

Linux-PC 환경에 이식한 CNS 시뮬레이션 프로그램은 500 MHz Processor PC 에서 200 ms 의 simulation cycle 기준에 대하여 약 5 배 정도(40 ms)의 성능이 관찰되었다[4]. OECD Halden Reactor Project(HRP)에서는 HP 735 UNIX 워크스테이션을 이용하여 SCORPIO, CAMS, CYGNUS 의 예측 시뮬레이션(Predictive Simulation)을 수행하였다[6]. 그 결과, CAMS 예측 시뮬레이터는 5 배 빠른 시뮬레이션(최적화되지 않은 상태로)이 가능했으며, CYGNUS 예측 시뮬레이터는 21 time step 을 구성하는 48 시간 transient 를 35 초에 계산하는 것으로 알려진다.

RTLinux 환경에서는 그래픽 프로그램을 제외한 시뮬레이션 프로그램만 수행하는 경우, 최대 1~2ms 의 시뮬레이션 cycle time 으로 실행이 가능하다. 이는 Linux 운영체제 환경에서보다 약 20 ~ 40 배 정도 빠른 예측에 해당한다. 응용 프로그램을 모두 커널 모듈로 구현한다면 성능이 더 향상될 것으로 예상된다.

5. 결론

실시간 컴퓨터 시뮬레이션이나 실시간 응용에는 실시간 운영체제를 사용함으로써 더 좋은 실시간 성능을 얻을 수 있다. 그리고, 연구실 규모의 실시간 응용에 Real-Time Linux 와 같은 공개 소스 소프트웨어를 사용함으로써, 저렴한 비용과 적은 노력으로 목적을 달성할 수가 있다. 특히, UNIX 환경에서 개발된 응용 프로그램을 Linux 환경에 이식(porting)하는 데는 적은 노력이 요구된다. RT-Mach, Windows-NT, QNX, 실시간 객체지향 운영체제 등 PC 하드웨어를 위한 다양한 실시간 운영체제 환경에서 실시간 응용의 성능을 검증하는 접근도 기대된다.

알림

(1) 이 연구는 원자력 기반 분야 원전 계측제어시스템 개발 사업으로 수행되는 디지털 계측제어 인허가 지원기술 개발 (Development of the Licensing Support Technology for Digital I&C) 과제에서 수행되었음.
(2) Linux Kernel 2.2 (ALZZA 6.0, WOW Linux 6.2) 및 RTLinux 2.2 는 공개 소스 소프트웨어임.
(3) "f2c" 소프트웨어는 공개 소스 소프트웨어로써 AT&T, Bellcore 그리고 Lucent 에 의해 1990 년 이래 개발되었슴.
(4) Picasso-3 Runtime 은 OECD Halden Reactor Project 에서 개발된 프로그램임.

참고문헌

[1] Mintin V.I., et al (2000). "Upgraded In-Core Instrumentation Subsystem (SVRK-M), " International Topical Meeting on Nuclear Power Plant Instrumentation Controls, and Human-Machine Interface Technologies (NPIC&HMIT2000), Washington, DC, 13-17 November, 2000. (CD-ROM)

[2] Victor Yodaiken. "The RT-Linux approach to hard real-time (A short position paper)." URL=http://luz.cs.nmt.edu/~rtlinux/documents/papers/whitepaper.html

[3] 박 재창 외 (1999). Compact Nuclear Simulator 의 성능향상 기술개발, 한국원자력학

회 ’99 추계학술발표회 논문집, 1999 년 10 월 30 일.

[4] 차경호 외 (2000). “Linux-PC 에서 Compact Nuclear Simulator 소프트웨어의 재사용,” 한국원자력학회 2000 추계학술발표회 논문집, 2000 년 10 월 26-27 일.

[5] Tomasz "Tomek" Motylevski (1999). "Real Time Programming -- Pitfalls, Problems, and common errors," Real Time Linux Workshop, Institut for Machine and Process Automation, Vienna University of Technology, Vienna, Austria, 13-15 December 1999.

[6] OECD HRP (1996). CAMS Prototype Extension: Integration of Data Acquisition, Signal Validation, Tracking Simulator, Predictive Simulator, State Identification and Probabilistic Safety Assessment, HWR-440, April 1996.