http://dx.doi.org/10.3745/KTCCS.2013.2.7.277

Performance Evaluation of Real-Time Linux Kernel Patch for Exynos4210 Processors

Hyeongseok Kang[†] · Joonwoo Lee[††] · Jinyoung Choi[†††] · Kanghee Kim[††††]

ABSTRACT

Recently, there is a growing need for an open software platform where developers easily write intelligent motion control applications for smart cars, smart robots, smart factories, and so on. To this end, a general-purpose operating system with rich functionalities and various hardware supports can be a candidate for such a platform, but it is known to have limitations in guaranteeing the responsiveness of individual applications. In this paper, to assess the suitability of Linux to be such a platform, we evaluate the real-time performance of Xenomai-patched Linux on an ARM-based processor Exynos4210 with motion control applications. Experimental results show that it is possible to stably provide motion cycle times below 1ms to such applications even with background workloads.

Keywords : Real-Time Linux, Xenomai, Real-Time Performance, Motion Control

Exynos4210 프로세서 상에서 실시간 리눅스 커널 패치의 성능 평가

강 형 석[†] · 이 준 우[††] · 최 진 영[†††] · 김 강 희[††††]

요 약

최근에 지능화된 자동차, 로봇, 공장 등에 대한 관심이 높아지면서 지능화된 모션 제어 응용을 손쉽게 작성할 수 있는 소프트웨어 플랫폼에 대한 필요성이 날로 커지고 있다. 이를 위해서 그 기능과 각종 하드웨어 지원이 풍부한 범용 운영체제를 사용하는 것이 바람직하지만, 일반적으로 개별 응용의 실시간 응답성이 항상 보장되지는 않는다. 본 논문은 범용 운영체제로서 리눅스가 산업용 실시간 시스템에 적합한지 평가하기 위해 ARM 기반 프로세서 Exynos4210 상에서 Xenomai 실시간 리눅스 패치를 적용하고 모션 제어 응용의 실시간 성능을 평가하였다. 평가 결과, 백그라운드 작업 부하의 간섭에도 불구하고 모션 제어 응용에게 1ms 미만의 제어 주기를 안정적으로 제공할 수 있음을 확인하였다.

키워드 : 실시간 리눅스, Xenomai, 실시간 성능, 모션 제어

1. 서 론

최근에 지능화된 자동차, 로봇, 공장 등에 대한 관심이 높아지면서 지능화된 모션 제어 응용을 손쉽게 작성할 수 있는 소프트웨어 플랫폼에 대한 필요성이 날로 커지고 있다. 지능화된 모션 제어 응용을 작성하기 위해서는 그 기능과 각종 하드웨어 지원이 풍부한 범용 운영체제를 사용하는 것이 바람직하지만, 일반적으로 범용 운영체제는 처리량 위주의 설계 철학 때문에 개별 응용의 실시간 응답성이 항상 보장되지는 않는다. 따라서 범용 운영체제는 그 풍부한 기능에도 불구하고 엄격한 실시간 제약조건을 갖는 산업용 실시간 시스템 영역에서는 널리 사용되지는 못했다. 그러나 범용 운영체제의 산업계로의 도입은 여전히 필요하다. 기존의 실시간 운영체제들은 실시간 응답성이 우수하기는 하지만, 폐쇄적인 개발자 생태계로 인하여 기능 확장이 더디며, 다양한 실시간 운영체제들이 난립하여 생태계 통일이 어렵다. 만일 범용 운영체제가 지능화된 모션 기계들이 요구하는 실시간 응답성을 제공할 수만 있다면 범용 운영체제의 개방성은 다양한 모션 제어 응용들을 창출하는데 크게 기여할 수 있을 것이다. 이러한 기대를 반영하듯이, 최근에 실시간 고속 필드버스로 모션 제어 시스템들에서 빠르게 확산되고 있

※ 본 연구는 미래창조과학부 및 정보통신산업진흥원의 IT융합 고급인력 과정지원사업(NIPA-2013-H0401-13-1004)과 2013년도 한국연구재단의 지원을 받아 수행된 연구임(No.2012015266).
† 준 회 원 : 숭실대학교 정보통신공학과 박사과정
†† 비 회 원 : 숭실대학교 정보통신공학과 박사과정
††† 비 회 원 : 숭실대학교 정보통신공학과 석사과정
†††† 비 회 원 : 숭실대학교 정보통신전자공학부 교수
논문접수 : 2013년 3월 26일
수 정 일 : 1차 2013년 5월 20일
심사완료 : 2013년 5월 21일
* Corresponding Author : Kanghee Kim(khkim@ssu.ac.kr)

는 EtherCAT[1]의 마스터 프로토콜의 참조 구현은 리눅스 상에서 개발되어 제공되고 있다.

본 논문은 범용 운영체제로서 리눅스가 산업용 실시간 시스템에 적합한지 평가하기 위해 임베디드 시스템에 널리 사용되는 ARM 기반 프로세서 Exynos4210[2]에 Xenomai[3]라는 실시간 패치를 이식하고, Exynos4210 프로세서가 제공하는 하드웨어 타이머인 PWM 타이머와 멀티코어 타이머(MCT)를 이용하여 고정밀 타이머를 구현하였다. 이를 기반으로 산업용 모션 제어 소프트웨어 환경을 구축하였으며, 구축된 환경을 토대로 하는 모션 제어기 상에서 리눅스 운영체제의 실시간 성능을 평가하였다. [4]에 따르면 다축 모터들의 고속 동기 제어에 필요한 최소 주기, 즉 다축 모션 제어기에서 모든 모터들의 제어 명령들을 생성하고 실시간 필드버스를 통해서 전송하는 주기는 1ms보다 작아야 한다. 본 논문의 실험 결과에 따르면, Xenomai가 적용된 리눅스 상에서 모션 제어 태스크를 간섭하는 작업부하가 존재하는 경우에도 1ms 미만의 주기를 안정적으로 제공할 수 있음을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 실시간 리눅스 기술에 대해 소개하고, 3장에서는 Exynos4210 프로세서를 위한 Xenomai 패치 개발 내용에 대해 설명한다. 4장에서는 성능 평가 결과를 제시하고 5장에서 결론을 맺는다.

2. 실시간 리눅스 기술

리눅스에서 산업용 실시간 응용들을 지원하려는 노력들은 (1) POSIX의 실시간 확장 API의 도입[5], (2) 리눅스 커널의 자체 반응성 개선[6], (3) 듀얼 OS 개념의 실시간 리눅스 패치[3,7,8]의 개발 등 다양한 수준에서 진행되어 왔다. (1)과 (2)의 접근법은 처리량 위주의 리눅스 설계 철학을 뛰어넘을 수는 없는 한계를 가지고 있으나 (3)의 접근법은 리눅스 커널 밑에 가상화 계층을 두어서 실시간 태스크들을 리눅스 운영체제의 간섭 없이 스케줄할 수 있는 가능성을 제공한다. 이러한 형태의 기술들로서는 RTLinux [7], RTAI [8], Xenomai [3] 등이 대표적이다. 이 기술들은 하나의 시스템 안에서 실시간 응용과 리눅스 응용을 동시에 지원하되, 실시간 응용에게는 높은 반응성을 제공하고, 리눅스 응용과의 통신을 허용하여 리눅스 운영체제가 관리하는 다양한 하드웨어 자원을 실시간 응용이 쉽게 접근할 수 있게 한다.

RTLinux는 상기 목적을 위해 다양한 하드웨어 장치들 중에서 오직 인터럽트 제어기만을 가상화하는 마이크로커널 구조를 채용하고 그 위에서 실시간 태스크들과 리눅스 운영체제를 실행시킨다. 리눅스 운영체제는 마이크로커널 위에서 완전 선점이 가능한 하나의 태스크로 취급되며, 리눅스 운영체제 내부의 하드웨어 인터럽트 제어기를 조작하는 모든 코드들은 마이크로커널의 가상 인터럽트 제어기를 에뮬레이션하는 계층을 조작하는 코드들로 대체된다. RTLinux는 이러한 구조를 채용하여, x86 PC에서 15 마이크로초 수준의 인터럽트 처리 시간을 제공할 수 있었다[7].

RTAI (Real-Time Application Interface)는 상용화된 RTLinux를 대체할 목적으로 이탈리아 밀라노 폴리테크니코의 항공우주공학과에서 개발한 오픈 소스 솔류션이다. RTAI 초기 구조는 RTLinux와 유사한 형태를 가졌는데, 하드웨어 인터럽트 제어기를 가상화하는 소프트웨어 모듈을 LKM (Loadable Kernel Module)의 형태로 리눅스 운영체제에 삽입하는 방식이 그 예이다. 그러나 RTAI의 이런 구조는 RTLinux의 특허를 침해한 것으로 이해되면서 특허 침해를 회피하기 위해 LKM 대신에 RTAI 3.0 버전부터 Adeos 나노커널[9]을 채택하는 구조로 변경되었다. Adeos 나노커널은 인터럽트 파이프 개념(3절에서 설명)을 제공하여 하드웨어 인터럽트 제어기를 가상화하며, Adeos 위에서 실시간 응용들과 리눅스 응용들을 동시에 지원한다. RTAI는 인터럽트 처리 시간 최소화에 많은 노력을 기울여 RTLinux에 뒤지지 않는 실시간 성능을 제공한다.

Xenomai는 RTAI와 구조적으로 매우 유사한 또 하나의 오픈 소스 솔류션이다. 즉 인터럽트 파이프 개념을 제공하는 Adeos 나노커널을 기반으로 RTOS와 리눅스를 동시에 실행시킨다. RTAI와 다른 점은 Xenomai가 기존의 Nucleus, VxWorks, pSOS+, VRTX와 같은 RTOS들에서 개발된 실시간 응용들에게 소스 코드 수준의 호환성을 제공하는 것을 목표로 삼고 있다는 것이다. 따라서 RTAI와 달리, 다양한 RTOS 스킨들을 Nucleus 기반 구현 위에서 제공하고 있다. Xenomai의 또 다른 특징으로는, 실시간 태스크들이 실행 중에 리눅스 도메인으로 이주(migration)하고 복귀하는 것을 허용하며, 이 경우 리눅스 운영체제 안에서의 응답성을 개선하기 위해서 HRT (High Resolution Timer)와 PREEMPT_RT 실시간 확장이 적용된 리눅스 커널을 사용한다는 점을 들 수 있다.

기존의 리눅스와 RTAI, 그리고 Xenomai에 대한 정량적인 성능 비교 평가는 몇몇 연구에서 수행된 바 있다. 관심있는 독자는 [10,11]를 참고하기 바란다.

3. Exynos4210용 Xenomai 패치의 개발

본 논문은 Xenomai 실시간 패치를 ARM 기반 프로세서 Exynos4210 상에 적용하여 실시간 성능을 평가한다. 본 절에서는 Exynos4210에 Xenomai 패치를 적용할 때 고려해야 하는 구현 이슈들을 설명한다.

3.1 Xenomai의 구조

Xenomai는 한 시스템 안에서 기존 RTOS 응용과 리눅스 응용을 동시에 실행할 수 있는 환경을 제공한다. 기존 RTOS 응용을 위해서 RTOS API를 제공하여 소스 코드 수준 호환성을 보장하고, 리눅스 API도 호출하는 하이브리드 응용도 허용하여 리눅스가 관리하는 스토리지, 네트워크, 그래픽과 같은 자원을 사용할 수 있게 한다. 기존 리눅스 응용은 Xenomai가 패치된 커널 위에서 바이너리 수준 호환성이 보장된다. 본 논문에서는 앞으로 RTOS 태스크와 하이브

리드 태스크를 포함하여 제노마이 태스크라고 부른다. Xenomai는 Adeos 나노커널 위에서 듀얼 OS 개념으로 RTOS와 리눅스를 함께 실행하는데, RTOS는 예외처리 및 스케줄링 관점에서 항상 우선순위가 높기 때문에 1차 도메인(또는 실시간 도메인)이라고 불리며, Linux는 2차 도메인(또는 리눅스 도메인)으로 불린다.

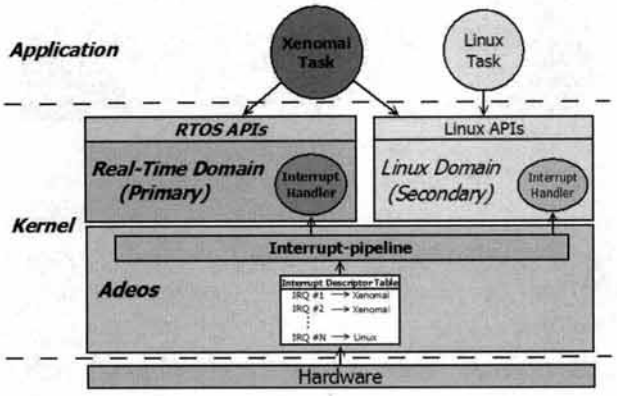

Fig. 1. Overview of Xenomai

Fig. 1은 Xenomai의 구조를 보여준다. [12]에 따르면, Xenomai는 다음과 같은 특징을 가지고 있다. 첫째, 각 도메인에서 실행하는 태스크의 우선순위 값은 해당 도메인뿐만 아니라, 모든 도메인들에서 전역적인 의미를 갖는다. 즉, 서로 다른 우선순위를 가진 제노마이 태스크들이 리눅스 API를 호출하여 리눅스 도메인에서 실행되는 경우에도 제노마이 태스크들 간에 상대적 우선순위는 유지되며, 리눅스 태스크들보다는 우선순위가 항상 높다. 둘째, 하드웨어에서 발생하는 인터럽트들은 제노마이 태스크의 실시간성을 보장하도록 도메인 수준에서 우선순위를 가지고 관리된다. 말하자면, Adeos의 인터럽트 파이프는 실시간 도메인 인터럽트들을 우선적으로 처리하며, 리눅스 도메인 인터럽트들은 제노마이 태스크가 실행하는 동안에는 리눅스 도메인으로의 전달을 지연시킨다. 셋째, 리눅스 커널은 제노마이 태스크의 반응성을 위해서 세밀하게 쪼개어진 임계 구간(critical section)들을 가진 Ingo Molnar의 PREEMPT_RT 커널[6]을 사용한다. 리눅스 태스크가 임계 구간 안에서 실행 중일 때, 제노마이 태스크가 동일한 임계 구간의 진입점에 도착하면, 리눅스 태스크는 가장 가까운 리스케줄링(rescheduling) 지점에서 실행을 자발적으로 멈추고 제노마이 태스크에게 제어를 넘긴다. 마지막으로, Xenomai는 우선순위 역전 현상을 해결하기 위해서 두 도메인에서 모두 우선순위 상속 기법[13]을 사용한다. 리눅스 도메인에서 PREEMPT_RT 버전의 리눅스를 사용하는 다른 이유는 바로 우선순위 상속 기법이 구현되었기 때문이다.

따라서, Xenomai를 Exynos4210 프로세서에 포팅하는 과정은 Adeos를 포팅하는 과정, 정확히는 Adeos의 도메인간 인터럽트 분배 및 관리를 담당하는 인터럽트 파이프를 포팅하는 과정과 실시간 도메인에서 동작해야 하는 인터럽트 핸들러를 작성하는 과정으로 나뉘어진다. 실시간 도메인에서 처리해야 하는 인터럽트 중에서 가장 중요한 것은 제노마이 태스크의 스케줄링에 필수적인 고정밀 타이머 인터럽트이다.

3.2 Adeos 패치의 적용

Adeos의 인터럽트 파이프는 도메인간 인터럽트 분배와 관리가 핵심 기능이다. 이 기능은 실시간 도메인 인터럽트와 리눅스 도메인 인터럽트에 대해서 다음과 같이 도메인 기반 우선순위에 입각하여 처리한다. 도메인 기반 우선순위 처리 방식은 실시간 도메인 인터럽트를 우선적으로 처리하되, 제노마이 태스크가 리눅스 도메인 인터럽트들 때문에 간섭받는 현상을 제거하려는 목적을 가지고 있다.

Table 1. Domain-aware Interrupt Processing

Condition	When a real-time domain interrupt occurs	When a Linux-domain interrupt occurs
If a xenomai task is running	the proper handler is called immediately	the proper handler is delayed until the xenomai task stops
If a Linux task is running	the proper handler is called immediately	the proper handler is called immediately

Adeos는 도메인 간에 공유되는 하드웨어 자원들이 존재할 때, 도메인들의 경쟁 상황을 관리해야 한다. 이렇게 공유되는 하드웨어 자원으로는 인터럽트 제어기가 대표적이다. 인터럽트 제어기는 실시간 도메인 인터럽트 핸들러와 리눅스 도메인 인터럽트 핸들러에서 각기 접근해야 하므로, 어느 한 도메인에서 인터럽트 제어기를 접근할 때에는 반드시 스핀락을 걸고, 설정 작업이 끝나면 스핀락을 풀어야 한다. 이것은 리눅스 도메인 인터럽트 핸들러가 인터럽트 제어기를 설정하는 동안에는 실시간 도메인 인터럽트도 작은 시간이나마 처리가 지연될 수 있다는 뜻이다. 본 논문에서는 도메인간 공유되는 L2 캐시에 대해 L2 캐시 조작 함수들이 두 도메인에서 경쟁적으로 호출되는 경우를 방지하기 위해서 리눅스 도메인의 L2 캐시 조작 함수들이 Adeos가 제공하는 스핀락 함수들을 이용해 실시간 도메인과 상호배제적으로 호출되도록 수정하였다. 반면에 타이머의 경우에는 도메인간 공유를 피하기 위해서 Exynos4210 프로세서에 다양한 하드웨어 타이머들이 존재함에 착안하여 실시간 도메인이 리눅스 도메인과 다른 하드웨어 타이머를 사용하도록 하였다. 이로써 타이머 인터럽트를 처리할 때 스핀락을 사용할 필요를 없앨 수 있었다.

3.3 고정밀 타이머 핸들러의 구현

[14]에 따르면 Xenomai는 실시간 도메인에서 제노마이 태스크가 명세한 정확한 주기에 깨어나 실행할 수 있도록 고정밀 타이머를 제공해야 한다. 고정밀 타이머는 하드웨어 타이머의 해상도 외에도, 다음 두 가지 소프트웨어 구현 이슈들을 고려해야 한다. 첫째, 일정한 주기를 갖는 운영체제

수준의 타이머 틱(tick)으로 임의의 주기값들을 갖는 제노마이 태스크들에 대해서 정확한 주기를 제공하기 어렵다. 타이머 틱을 사용하면, 각 태스크가 요구하는 주기값은 타이머 틱의 배수로 반올림 또는 반내림이 될 수 밖에 없다. 이런 상황을 방지하기 위해서 Xenomai는 주기적인 타이머 틱을 제공하는 것 대신에 타이머 인터럽트 핸들러가 호출될 때마다 다음 태스크를 깨우는 시점까지의 남은 시간을 매번 계산하여 타이머 인터럽트를 예약하는 방식을 사용한다. 본 논문에서는 이를 위해 Exynos4210 프로세서가 제공하는 50MHz의 PWM 타이머를 사용하였으며, PWM 타이머의 one-shot mode를 이용하여 매 인터럽트 핸들러 호출시 다음 타이머 인터럽트를 예약하였다. 둘째, Xenomai의 타이머 인터럽트 핸들러 안에서 다음에 발생할 타이머 인터럽트를 예약할 때에는, 제노마이 태스크가 요구하는 주기 T를 타임아웃값으로 그대로 설정해서는 안 되고, 타이머 인터럽트 핸들러가 소모한 가변적인 시간 D를 차감해야 한다. 타이머 인터럽트 핸들러는 동일한 코드 지점에서 타이머 인터럽트를 예약하겠지만, 실제 물리적인 시간축 관점에서 예약 시점들이 ARM 프로세서의 복잡한 내부 동작 및 캐시 적중률의 차이로 인해서 균일한 시간 간격을 갖는다고 보장할 수 없다. 따라서 제노마이 태스크가 지정한 정확한 시각에

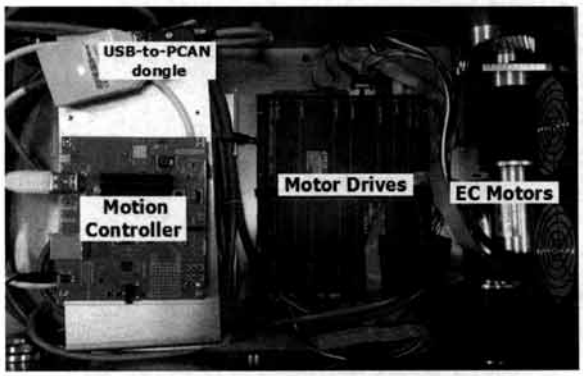

Fig. 2. Target System

타이머 인터럽트를 발생시키기 위해서는, 이전에 호출되는 타이머 인터럽트 핸들러 안에서 핸들러가 소비한 가변적인 시간 D를 측정하여 다음 타이머 인터럽트를 위한 타임아웃값 W에 반영해야 한다(W = T - D). 본 논문에서는 타이머 인터럽트 핸들러의 소비 시간을 측정하기 위해서 Exynos4210 프로세서가 Performance Monitoring Counter로서 제공하는 Multi-Core Timer의 free-running counter를 사용하였다.

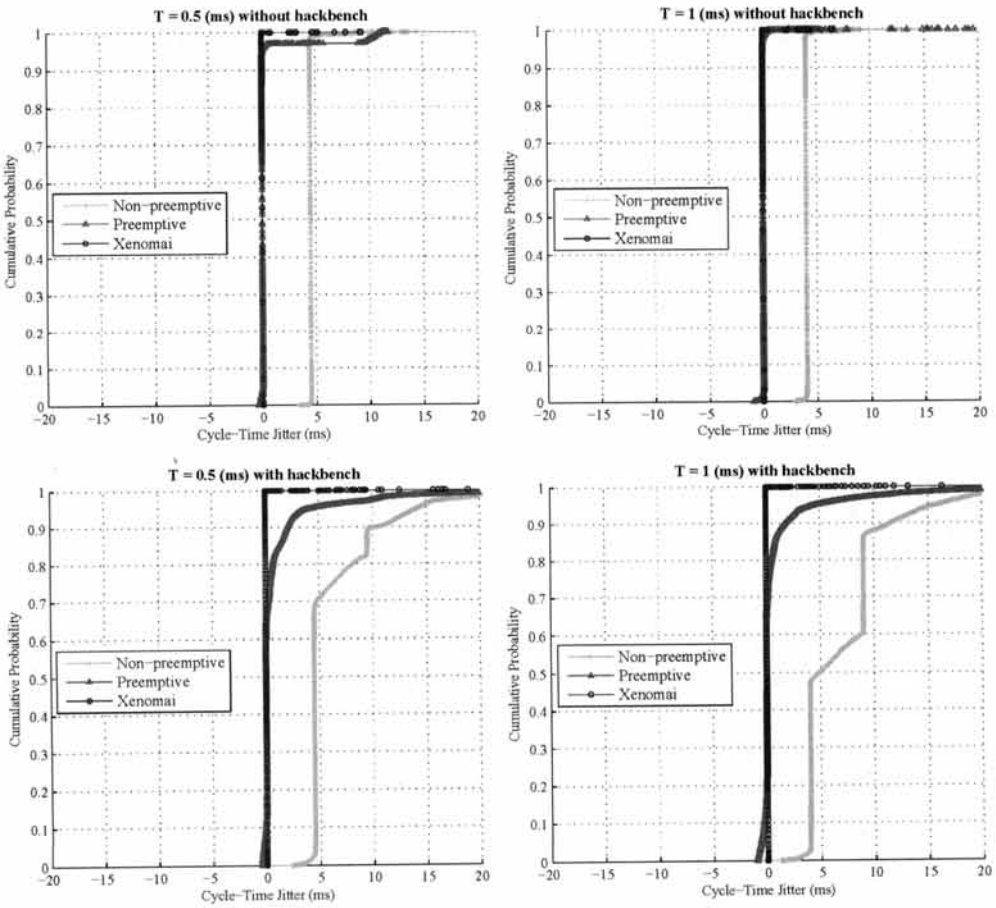

Fig. 3. Jitter Distributions of the Motion Control Task

4. 성능 평가

Xenomai에 대한 성능 평가는 Fig. 2와 같은 평가 시스템 상에서 수행되었다. 시스템은 크게 모션 제어기와 실시간 필드버스, 그리고 모터와 모터 드라이브로 구성된다. 모션 제어기는 Exynos4210 프로세서를 사용하는 odroid-pc 보드[15]로 구성하였고, Xenomai 2.6.0 패치를 적용한 리눅스 커널 2.6.35.7을 사용하였다. 모션 제어기와 2개의 모터 드라이브를 연결하기 위해 CAN 버스를 사용하고, 각 모터 드라이브에는 산업용 정밀 모터 Maxon EC Motor (EC-4pole 30)를 연결하였다.

제어기 내부에서 동작하는 주요 소프트웨어는 Motion Control Engine 및 CAN Festival Library[16]로 구성되는 모션 제어 태스크와 CAN 디바이스 드라이버다. Motion Control Engine은 주기적으로 모터 제어명령을 생성하며, CAN Festival Library를 통해 제어명령을 CAN 메시지로 변환시킨다. CAN 디바이스 드라이버는 CAN 버스로의 메시지 입출력을 관리한다. 타겟 시스템에서 모션 제어 태스크는 실시간 도메인에서 동작하며, CAN 디바이스 드라이버는 리눅스 도메인에서 동작한다.

타겟 시스템의 성능 평가를 위해서 다음 3가지 종류의 커널을 비교실험에 사용하였다. 1) 비선점형 리눅스 커널은 실시간 기술이 적용되지 않은 커널이다. 2) 선점형 리눅스 커널은 PREEMPT_RT 패치와 HRT 타이머를 적용한 커널이다. 3) Xenomai 적용 커널은 PREEMPT_RT와 HRT가 적용된 선점형 커널에 Xenomai 패치를 적용한 커널이다. 본 논문의 성능 평가 지표는 모션 제어 태스크의 주기의 정확도로서 주기적으로 깨어나는 모션 태스크가 매번 깨어나기로 예정된 시각에서 얼마만큼의 지터(혹은 편차)를 가지고 깨어나는지를 측정하여 평가한다. 이를 위해서 0.5ms와 1ms의 주기 T에 대하여 모션 제어 태스크가 10,000개 이상의 주기들을 실행하는 실험을 10회 반복하였다. 또한 CPU의 부하 유무에 따른 성능 차이를 확인하기 위하여 Hackbench [17]를 백그라운드 작업 부하로 사용하였다.

Fig. 3에서 상단 그래프들은 모션 제어 태스크가 Hackbench 없이 동작하는 경우들이다. 비선점형 리눅스 커널(Nonpreemptive)의 경우 지터가 거의 모든 경우에 4ms 넘게 관찰되고 있다. 선점형 리눅스 커널 (Preemptive)의 경우에는 평균 지터가 T=1ms일 때는 7μs, T=0.5ms일 때는 305μs로 나타나 비선점형 리눅스 커널에 비해 안정적인 성능을 보이나 1ms 이상의 지터 샘플들이 상당수 분포되어 있다. 반면 Xenomai가 적용된 선점형 리눅스 커널은 T=0.5ms인 경우에도 평균 지터가 3μs 정도로 매우 안정적인 성능을 보이는 것을 확인할 수 있다. Fig. 3에서 하단 그래프들은 모션 제어 태스크가 Hackbench와 함께 동작하는 경우이다. 비선점형과 선점형 리눅스 커널은 Hackbench의 영향으로 지터가 전체적으로 증가하였으며, 특히 T=0.5ms일 때 지터의 평균값이 각각 6.36ms와 0.98ms로 주기를 초과하는 결과를 보였다. 반면에 Xenomai가 적용된 리눅스 커널은 백그라운드 작업부하의 영향에도 불구하고 지터의 분포가 매우 안정적이며, 특히 T=0.5ms인 경우에도 평균값이 12μs로서 우수한 성능을 보이고 있다. 다만, Xenomai의 경우에도 드물게 지터값이 크게 관찰되는 경우들이 존재하는데, 이는 타겟 시스템에서 CAN 디바이스 드라이버가 실시간 도메인이 아닌 리눅스 도메인에서 동작하기 때문으로 판단되며 향후 실시간 도메인용 드라이버의 개발이 요구되는 대목이다.

5. 결 론

본 논문에서는 Xenomai 실시간 리눅스 커널 패치의 실시간 성능을 Exynos4210 프로세서 기반 모션 제어 시스템 환경에서 평가해 보았다. 성능 평가를 위해 Exynos4210용 Xenomai 패치를 개발하였고, CAN 버스 기반의 모션 제어 시스템을 구축하여 모션 제어 태스크의 제어 주기의 정확도를 평가하였다. 성능 평가 결과, 모션 제어 태스크의 주기가 0.5ms인 경우에도 Xenomai는 매우 안정적인 실시간 성능을 보이는 것을 확인하였다. 따라서 산업용 실시간 시스템에서도 리눅스와 같은 범용 운영체제를 활용할 수 있을 것으로 기대된다.

참 고 문 헌

[1] EtherCAT Technology Group, EtherCAT - Technical Introd-uction and Overview [Internet], http://www.tritek.co.kr/pdf/ethercat/ethercat.pdf.

[2] Samsung Electronics Inc., Samsung Exynos4210, http://www.samsung.com/global/business/semiconductor/minisite/Exynos/products4210.html.

[3] P. Gerum, Xenomai-Implementing a RTOS emulation framework on GNU/Linux, http://www.xenomai.org/documentation/xenomai-2.4/html/xenomai.

[4] I. Kim, S. Park, M. Sung and T. Kim, "Design and Implement-ation of a Real-Time Motion Controller using Open Source Software," KIISE Transactions on Computer Systems and Theory, Vol.39, No.2, pp.84-95, Apr., 2012.

[5] IEEE, "IEEE Standard for Information Technology - Portable Operating System Interface (POSIX) Part 1: System Application Program Interface (API) Amendment 1: Realtime Extension [C Language]," IEEE Std 1003.1b - 1993, Apr., 1994.

[6] S. Rostedt and D. V. Hart, "Internals of the RT Patch," in Proceedings of the Real-Time Systems Symposium, Jun., 2007, Vol.2, pp.161-172.

[7] V. Yodaiken and M. Barabanov, "A Real-Time Linux," in Proceedings of the Linux Applications Development and Deployment Conference (USELINUX), Jan., 1997.

[8] K. Yaghmour, "The Real-Time Application Interface," in Proceedings of the Linux Symposium, 2001.

[9] K. Yaghmour, Adaptive Domain Environment for Operating Systems, http://www.opersys.com/ftp/pub/Adeos/adeos.pdf.

[10] J. Brown and B. Martin, "How fast is fast enough? Choosing between Xenomai and Linux for real-time applications," in Proceedings of the 12thReal-TimeLinuxWorkshop, Oct., 2010.
[11] M. Piatek, "Real-Time Application Interface and Xenomai modified GNU/Linux real-time operating systems dedicated to control," in Proceedings of the Computer Methods and Systems, 2007, pp.179-184.
[12] P. Gerum, Life with Adeos, http://www.xenomai.org/documentation/xenomai-2.3/pdf/Life-with-Adeos-rev-B.pdf.
[13] L. Sha, R. Rajkumar, and J. Lehoczky, "Priority Inheritance Protocols: An Approach to Real-Time Synchronization," IEEE Transactions on Computers, Vol.39, No.9, pp.1175-1185, Sep., 1990.
[14] Xenomai Organization, Adapting the I-pipe core patch to a new ARM SOC, http://www.xenomai.org/index.php/I-pipe-core:ArmPorting.
[15] Hardkernel Inc., ODROID-PC, http://www.hardkernel.com/renewal_2011/products/prdt_info.php?g_code=G132342040298.
[16] CAN Festival: Free software CANopen framework, http://www.canfestival.org.
[17] C. Thomas, Linux 2.6 Performance, http://www.hs-ugsburg.de.

강 형 석

e-mail : hseokkang@ssu.ac.kr
2011년 숭실대학교 정보통신전자공학부 (학사)
2013년 숭실대학교 정보통신공학과(석사)
2013년~현 재 숭실대학교 정보통신공학과 박사과정
관심분야 : 실시간 시스템, 모션 제어 시스템

이 준 우

e-mail : joonwoolee7@gmail.com
2011년 숭실대학교 정보통신전자공학부 (학사)
2013년 숭실대학교 정보통신공학과(석사)
2013년~현 재 숭실대학교 정보통신공학과 박사과정
관심분야 : 실시간 시스템, 모바일 플랫폼

최 진 영

e-mail : kbkbc88@gmail.com
2012년 숭실대학교 정보통신전자공학부 (학사)
2012년~현 재 숭실대학교 정보통신공학과 석사과정
관심분야 : 실시간 시스템, 모션 제어 시스템

김 강 희

e-mail : khkim@ssu.ac.kr
1996년 서울대학교 컴퓨터공학과(학사)
1998년 서울대학교 전기컴퓨터공학부 (석사)
2004년 서울대학교 전기컴퓨터공학부 (Ph.D., 박사)
2004년~2010년 삼성전자㈜ 무선사업부 책임연구원
2010년~현 재 숭실대학교 정보통신전자공학부 교수
관심분야 : 실시간 시스템, 임베디드 시스템, 운영체제, 모바일 플랫폼, 플래시 저장장치, 모션 제어 시스템