http://dx.doi.org/10.3745/KIPSTA.2011.18A.6.271

인공심장의 예측 가능한 제어를 위한 실시간 소프트웨어 설계 구조의 개선

정 세 훈[†] · 김 희 진[††] · 박 상 수[†††] · 차 성 덕[††††]

요 약

실시간 소프트웨어 설계 구조 중의 하나인 시간 구동 구조 (TTA: Time-Triggered Architecture)는 미리 정해진 시간에 따라 특정 태스크를 수행하기 때문에 소프트웨어의 동작을 예측하기 쉽지만 이의 적용을 위해서는 시스템의 시간 제약성을 만족시키는 설계 과정이 필수적이다. 반면 이벤트 구동 구조 (ETA: Event-Triggered Architecture)는 외부의 이벤트가 발생함에 따라 대응되는 태스크를 수행하는 방식으로 소프트웨어의 구조가 직관적이고 이벤트에 대한 반응 시간이 빠르며 확장이 용이하다. 그러나 이 구조는 다양한 이벤트 발생 상황에 대한 시스템 동작의 예측이 어려워서 높은 안전도가 필요한 시스템 구현에 많이 사용되고 있지 않다. 많은 관련 연구에서 높은 안전도가 필요한 안전 지향 시스템에 TTA를 적용 할 것을 권장되지만, 실제 구현에 있어서는 이의 적용에 필요한 실시간성 분석에 많은 노력이 소요되고, 소프트웨어 공학 기술 적용에 대한 인식 부족으로 TTA의 적용이 많지 않은 실정이다. 본 논문은 인간의 생명과 직결되어 이에 따른 실시간성과 안전성이 요구되는 인공심장 제어 시스템에 내장된 소프트웨어 구조를 TTA 기반으로 개선한 연구를 기술한다. 본 연구에서는 인터럽트 측정 소프트웨어를 구현하여 기존에 내장된 실시간 소프트웨어가 가진 태스크의 시간적 속성을 파악하고 RMA (Rate-Monotonic Analysis) 실시간 분석 기법을 통해 시스템의 실시간성을 만족할 수 있도록 설계를 개선하였다. 또한, 이를 바탕으로 인공심장 제어 소프트웨어의 구현을 개선하여 다양한 실험을 통해 개선된 TTA 기반의 소프트웨어를 탑재한 인공심장 시스템은 시스템의 동작 예측도를 획기적으로 높여주면서 기존의 인공심장 시스템과 동일하게 동작함을 확인하였다.

키워드 : 실시간 시스템, 안전 지향 시스템, 소프트웨어 설계, 예측성

Architectural Refactoring of Real-Time Software Design for Predictable Controls of Artificial Heart

Sehun Jeong[†] · Heejin Kim[††] · Sangsoo Park[†††] · Sungdeok Cha[††††]

ABSTRACT

Time-Triggered Architecture (TTA), one of real-time software design paradigms which executes tasks in timely manner, has long been advocated as being better suited in fore-sighting system behavior than event-triggered architecture (ETA). To gain this valuable feature of TTA, however, precise task designing process is mandatory. Alternatively, ETA tries to execute tasks whenever paired events are occurred. It provides intuitive and flexible basement to add/remove tasks and, moreover, better response time performance. However ETA is difficult to analyze because system behavior might be different depending on the order of interrupts detected by the system. Many previous researches recommended TTA when developing safety-critical real-time systems, but cost problem of task designing process and insufficient consensus for applying rigorous software engineering practice are still challenging in practice. This paper describes software refactoring process which applying TTA approach into ETA based embedded software in artificial heart system. We implemented dedicated interrupt monitoring program to capture existing tasks' real-time characteristics. Based on the captured information, proper task designing process is done. Real-time analysis using RMA (Rate-Monotonic Analysis) verified that new design guarantees timeliness of the system. Empirical experiments revealed that revised design is as efficient, when measured in terms of system's external output, as the old design and enhances predictability of the system behavior as well.

Keywords : Real-Time System, Safety-Critical System, Software Design, Predictability

※ 본 논문은 한국연구재단의 기초연구사업 지원(과제번호: 2011-0013422), 정보통신산업진흥원의 SW 공학 요소기술 개발과 전문인력 양성사업의 지원을 받아 수행된 것임.
† 준 회 원 : 고려대학교 컴퓨터학과 석사과정
†† 준 회 원 : 이화여자대학교 컴퓨터공학과 석사과정
††† 정 회 원 : 이화여자대학교 컴퓨터공학과 전임강사(교신저자)
†††† 정 회 원 : 고려대학교 정보통신대학 교수
논문접수 : 2011년 9월 20일
수 정 일 : 1차 2011년 10월 7일
심사완료 : 2011년 10월 7일

1. 서 론

인공심장 시스템은 이전의 심장 이식방법이 가지고 있는 심장 수급과 인체 면역 체계 거부에 의한 문제가 적어 말기 심장 질환 환자들을 치료하기 위한 방법으로 최근 크게 각광받고 있다[1][2][3]. 내장형 실시간 시스템의 발전으로 기존의 인공심장에 필요한 복잡한 펌프 등의 기계 장치는 작은 고성능 모터로 대체되었고, TI사나 Analog Devices사에서 제공되는 저전력 고성능 DSP [4][5] 칩은 인공심장에 필요한 다양한 센서와 모터를 제어할 수 있는 인터페이스를 제공함과 동시에 고수준의 프로그래밍 언어를 사용 가능하게 하여 인공심장 시스템의 소형화와 이동성 확보에 기여하고 있다. 그러나 원자력 발전소에 필요한 제어 시스템과 같은 안전 지향 소프트웨어의 개발에 필요한 안전성 확보 문제는 인공심장 제어 소프트웨어에도 여전히 필수적인 요소이다. 인공심장 제어 소프트웨어는 복수의 센서와 버튼 입력을 통해 현재 모터 상태와 사용자(환자 혹은 의사)의 요구사항을 인식하고 이를 바탕으로 향후 모터의 회전 방향과 속도를 결정하여 외부 환경 변화에 상관 없이 환자에게 필요한 일정한 펌프 속도와 혈류를 유지할 수 있게끔 해야 한다. 인공심장 제어 시스템에서의 요구사항은 1) 인체와 같은 동적인 외부 환경과 상호 작용해야 하는 특성과 2) 센서나 버튼, 모터 제어와 같은 복수의 태스크들이 시스템 자원을 효과적으로 공유해야 한다는 것이며, 이 요구사항을 만족하지 못했을 때에 발생할 수 있는 비정상 동작이 인명 사고로 이어질 수 있다.

인공심장 시스템은 위와 같은 각 기능의 수행이 주어진 시간 이내에 완료되어야 하는 실시간 시스템 (Real-Time System)으로 이러한 기능들은 CPU와 같은 하드웨어 자원을 공유하며 수행된다. 이러한 안전 지향의 실시간 시스템에서는 소프트웨어의 동작 시간 면에서의 예측성 (Predictability)이 필수적이지만, 이에 대한 완벽한 사후 검사는 현실적으로 불가능하다[6][7]. 따라서, 다양한 관련 연구 [9][10][11][12]의 사례를 기반으로 소프트웨어의 수행을 유발하는 주체가 실시간 시스템의 예측성에 큰 영향을 미친다고 알려졌기 때문에, 인공심장과 같은 안전 지향의 실시간 시스템에서는 각 소프트웨어의 기능을 수행하는 태스크를 주기적인 시간 기반으로 수행하는 시간 구동 구조 (TTA: Time-Triggered Architecture)가 적합하다고 제시된 바 있다[8]. 즉, TTA는 모든 태스크를 미리 정의된 주기에 따라 수행하므로 특정 시점의 동작을 분석할 때 유용하게 활용할 수 있다.

그러나 TTA를 실시간 시스템 설계에 적용하는 데에는 크게 두 가지 제약 사항이 있다. 첫째, 미리 정의된 주기에 따라 수행되는 모든 태스크가 주어진 시간 이내에 완료되도록 CPU와 같은 공유되는 자원을 스케줄링하는 설계 과정이 필요하다. 이 과정에서 다양한 태스크의 수행 주기를 고려해야 하고 최대 수행 시간과 발생 주기 등 각 태스크의 시간적 속성을 사전에 파악해야 하는 문제가 있다. 둘째, TTA는 버튼 입력, 모터 센서 등과 같은 외부 이벤트 발생 여부를 즉각 확인하는 것이 아니라 주기적으로 수행되는 태스크에 의해 확인되므로 실제 이벤트 발생 시간과 태스크에 의해 그 이벤트의 발생 여부가 확인된 시점 사이에 시간 차이가 생길 수 있다. 이러한 점은 실시간 시스템의 이벤트에 대한 응답 시간을 저하시킬 수 있는데, 인공심장 시스템과 같은 펌프의 현재 동작 상태의 모니터링을 통해 다음 펌프 동작을 결정하는 시스템의 응답 시간 제약을 만족시키지 못하는 문제가 발생할 수 있다.

본 논문은 H-VAD 인공심장 시스템 [13][14]의 제어 소프트웨어를 TTA 기반으로 개선하여 시간면에서 동작 예측성을 제공하여 시스템의 안전성 확보한다. 개선 대상인 H-VAD 인공심장 시스템은 장기 동물실험에서 180일 이상 정상 작동하여 FDA 인공심장 장기 동물실험 기준을 만족시킨 시스템으로, 이에 내장된 소프트웨어는 비주기적 태스크 6개와 주기적 태스크 1개로 구성되어있다. 이를 위해 제어 소프트웨어에 특화된 이벤트 발생 정보 수집 도구를 구현하여 사용하였고 이 도구를 통해 수집한 각 태스크의 실시간 특성을 기반으로 각 태스크가 수행되어야 할 주기와 순서 등의 스케줄링 모델을 수립하였다. 또한 기존 하드웨어 상에서 수립된 모델이 모든 시간 제약성을 만족하고 수행됨을 보이기 위해 실시간 시스템 분석 기법의 하나인 RMA (Rate-Monotonic Analysis) [15]를 H-VAD 인공심장 제어 소프트웨어의 실시간 특성에 맞게 확장하여 이용하였다.

본 논문의 구성은 다음과 같다. 2장은 구조 개선의 대상인 H-VAD 인공심장 시스템과 TTA, ETA의 장,단점을 소개하고, 3장은 기존 ETA 기반의 H-VAD 인공심장 제어 소프트웨어에의 TTA 적용 과정을 다루며, 4장에서는 소프트웨어 설계 구조가 개선되어 구현된 TT-HVAD의 성능을 실험을 통하여 평가한다. 마지막으로 5장은 본 논문의 결론과 향후 연구를 제시한다.

2. 인공심장 제어 시스템 모델

2.1 시간 구동 구조와 이벤트 구동 구조

실시간 시스템을 설계하는 방법은 크게 이벤트 구동 (ET: Event-Triggered), 시간 구동 (TT: Time-Triggered), 그리고 이 두 방법을 섞어서 같이 사용하는 이벤트 시간 혼합 구동 (MT: Mixed-Triggered) [16] 방식이 있다. ET 방식은 시스템 내에서 인터럽트와 같은 외부 이벤트가 발생했을 때 이를 하드웨어에 의해 즉각 호출된 핸들러에 의해 처리해주는 방법이며, TT 방식은 이벤트를 처리할 태스크를 어떤 시점에 수행시킬 지 미리 정한 후에 이를 주기적으로 처리해주는 방법이다. ET 방식은 시스템 초기 설계 비용과 이벤트 응답 시간은 줄이면서 시스템 유연성은 높일 수 있다는 장점이 있는 반면 이벤트가 발생했을 때 실행되는 이벤트 핸들러로 인해 지속적인 시스템 오버헤드가 발생하고 이로 인한 처리 지연으로 인한 시간 제약 위반을 발생시킬 수 있는 단점이 있다[19]. 또한, 이벤트를 임의의 발생 시점

에 동적으로 처리하기 때문에 시간면에서 시스템의 예측 가능성이 떨어지고 이에 따른 소프트웨어의 결함 검출이 어렵기 때문에 시스템의 안전성의 보장이 어렵다[18].

반면 TT 방식은 시스템은 모든 태스크의 시간 특성을 고려해야하는 초기 설계 비용이 높은 단점이 있는 반면 시간면에서 시스템 동작의 예측이 가능하고 시스템 안전성을 보장할 수 있다는 장점이 있다 [12]. 즉, TT 방식에서는 어떤 태스크가 어떤 시점에서 실행되는 지가 미리 결정되어 있어 수행시간 지연 등 실행에 결함이 발생할 경우 이를 빠른 시간 안에 검출할 수 있다[18].

<표 1>에서 기술된 바와 같이 ET 방식과 TT 방식은 서로 다른 특징들을 갖고 있기 때문에 설계가 비교적 쉽고 시스템의 유연성이 높은 ET 방식은 비안전 필수 실시간 시스템 (Non-Safety Critical Real-Time System)에서 예측 가능성과 안전성이 높은 TT 방식은 안전 필수 실시간 시스템 (Safety-Critical Real-Time System)에서 주로 사용이 된다[20].

마지막 MT 방식은 대부분 주기적이거나 자주 발생하는 태스크는 TT 방식 기반으로 그 외의 태스크는 ET 방식기반으로 설계하는 방법을 말한다[21]. MT 방식은 TT 방식보다 유연성이 높으면서 ET 방식보다 예측 가능성이 좋다는 장점이 있지만, ET 방식의 태스크들로 인해 실시간 시스템에서는 안전성을 보장하기 어렵다는 단점이 존재한다.

〈표 1〉 ET 방식과 TT 방식의 비교

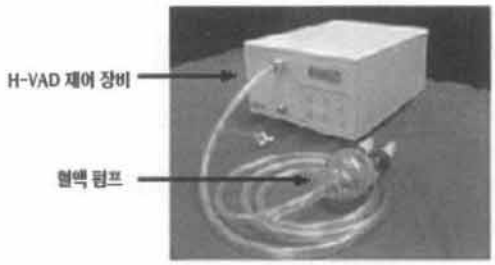
	Event-Triggered (ET)	Time-Triggered (TT)
장점	빠른 응답 시간 높은 유연성	높은 예측 가능성 높은 안전성 결함 검출의 용이함
단점	낮은 예측 가능성 낮은 안전성 결함 검출의 어려움 높은 시스템 오버헤드	높은 초기 설계 비용

개선 대상인 H-VAD 인공심장에 내장된 소프트웨어는 비주기적 태스크 6개와 주기적 태스크 1개로 구성된 MT 방식으로 위에서 기술된 것과 같이 소프트웨어의 대부분을 차지하는 비주기적 태스크에 의해 시간면에서 예측성이 낮고 이로 인해 시스템의 안정성 보장에 어려움이 있다.

2.2 H-VAD 인공심장 시스템 구조

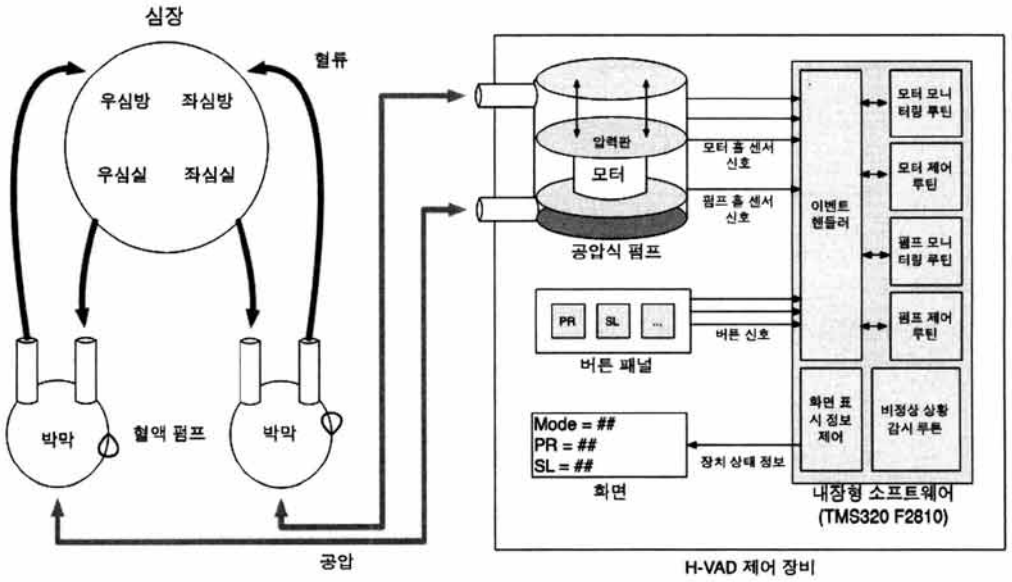
2.2.1 하드웨어

H-VAD (Hybrid Ventricular Assist Device)는 심혈관 질환 환자의 약해진 심장 활동을 보조하기 위한 부분 인공심장 장치로 한국인공장기센터 (KAOC: Korea Artificial Organ Center)에 의해 개발 되었으며 기존의 펌프 방식의 인공심장과 비교했을 때 크기가 작고 배터리로 작동 가능하다.



(그림 1) H-VAD 인공심장 시스템

H-VAD 인공심장 시스템은 주문 제작된 모터로 작동하는 공압식 펌프 (Pneumatic Pump)를 이용하여 최대 2개의 혈액 펌프를 구동할 수 있다. 이를 사용하는 환자나 의사는 (그림 1)에서와 같이 버튼으로 구성된 사용자 인터페이스를 통해 피스톤의 왕복 속도나 이동 거리를 일정한 범위 내에서 조작할 수 있으며, 총 여섯 개의 버튼 (작동, 정지, 증감, 제어 모드 변경) 과 현재 상태를 확인할 수 있는 LCD 화면으로 이루어져 있다.



(그림 2) H-VAD 인공심장 시스템 구조

(그림 2)는 심장과 상호 작용하는 H-VAD 인공심장 시스템을 개괄적으로 보여주며 중요 부분에 대한 설명은 다음과 같다.

① 공압식 펌프: 펌프 피스톤에 포함된 모터가 피스톤 축을 따라 회전하며 압력판을 위/아래로 움직이며 두 개의 혈액 펌프에 필요한 공압을 발생시킨다. 혈액 펌프에 장착된 단방향 밸브가 혈류의 방향이 일정하게 유지되도록 한다.

② 센서: 총 4개의 홀 센서 (Hall-Effect Sensor)가 모터에 3개, 펌프 축에 1개 장착되어 있다. 모터에 설치된 홀 센서는 모터의 회전 방향 및 회전 속도를 모니터링 하는데 사용되며 펌프 축에 설치된 센서는 피스톤의 움직임을 모니터링 한다.

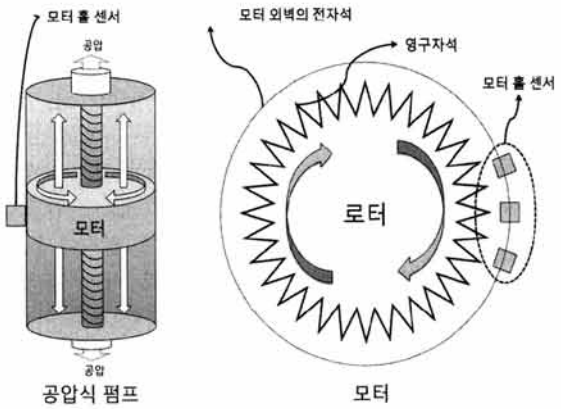
③ DSP (Digital Signal Processor): Texas Instrument사의 TMS320 F2810 기반의 컨트롤러를 사용한다. 4개의 센서 신호와 사용자 버튼 이벤트를 입력으로 받아 모터 출력을 계산한다.

④ 사용자 인터페이스: 6개의 버튼과 LCD로 구성되며 피스톤의 왕복 속도와 이동 거리를 제어할 수 있으며 현재 H-VAD 제어 장비의 상태를 LCD 화면을 통해 사용자에게 보여준다.

2.2.2 제어 소프트웨어

H-VAD 인공심장 시스템에 탑재된 제어 소프트웨어는 총 7개의 태스크를 통해 현재 모터와 펌프 상태를 모니터링하고 사용자의 입력에 따라 일정한 시간 간격(주기적)으로 다음 모터 출력을 계산한다. 각 태스크의 역할은 아래와 같다.

① T3PINT: 1ms 주기로 수행되는 태스크로 다음의 3가지 기능을 수행한다. 첫째, 현재 피스톤 위치에 따라 다음 피스톤 속도의 가속/감속/정속 여부를 결정한다. 둘째, 피스톤이 펌프 양 끝 단에 도달해 정지해 있을 경우 일정시간



(그림 3) H-VAD 인공심장 시스템의 홀 센서 구성

이 지난 뒤 반대 방향으로 진행하게 한다. 셋째, 센서의 작동 상태를 주기적으로 모니터링 한다.

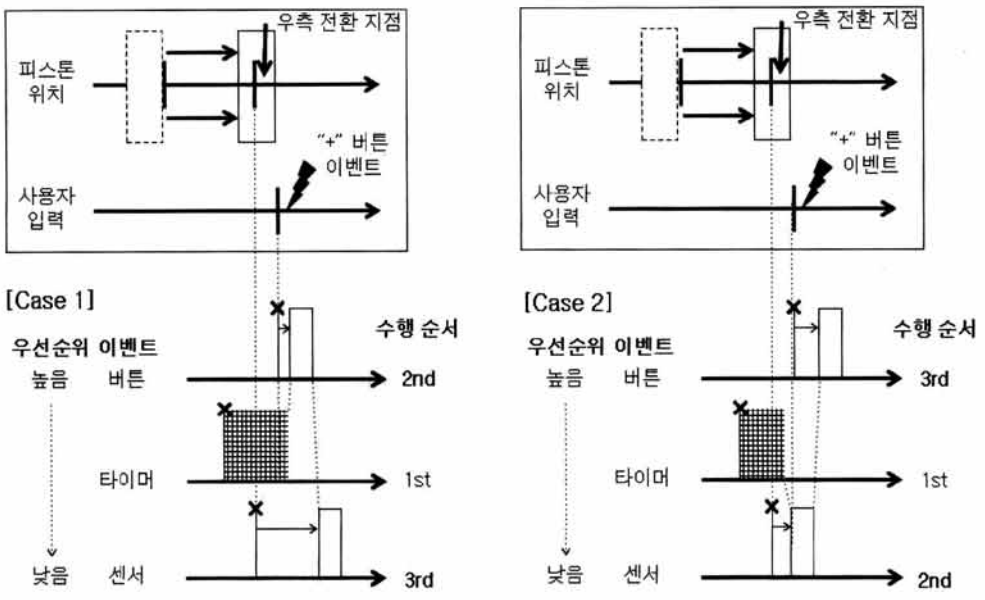
② CAPINT1: (그림 3)의 모터 홀 센서에 의해 발생되는 인터럽트에 의해 비주기적으로 수행된다. 피스톤이 이동 축을 지나는 시간을 이용하여 피스톤의 정상 작동 여부를 모니터링 한다.

③ CAPINT2, 3: 사용자 인터페이스의 버튼 중 펌프 속도 및 이동 거리 조절 버튼을 눌렀을 때 수행된다. H-VAD 제어 소프트웨어의 펌프 제어 모드에 따라 펌프의 속도 혹은 이동 거리를 조정한다.

④ CAPINT4, 5, 6: 모터 내부에 장착된 모터 홀 센서 (그림 3)에 의해 수행되는 태스크이며 모터의 회전 방향, 속도, 피스톤의 위치 정보를 갱신한다. 피스톤이 위치가 피스톤의 이동범위 경계 값일 경우 모터를 정지시킨다.

2.3 태스크 처리의 이벤트 구동 방식으로 인한 예측성 문제

많은 실시간 시스템에서 수행되는 태스크의 처리가 인터



(그림 4) 시간 차이에 따른 시스템의 상이한 동작 예

럽트 등 이벤트 구동 방식으로 설계가 될 때에 시간에 따른 시스템의 동작을 예측하기 어려우며 모터 제어를 위한 대부분의 태스크가 이벤트 구동 방식으로 설계 되어 있는 기존의 H-VAD 인공심장 시스템은 (그림 4)와 같이 동일한 이벤트 발생 환경에 대해서 시간에 따라 상이한 동작을 할 수 있다. (그림 4)는 펌프의 피스톤이 사용자가 정의한 우측 방향 전환 지점에 도달했을 때 사용자가 펌프의 작동 범위를 늘리는 버튼 이벤트를 발생 시킨 것으로, [Case 1]에서는 타이머 태스크의 긴 수행 시간으로 인해 나머지 버튼, 센서 이벤트가 모두 지연되었고 타이머 태스크가 종료되었을 때 두 태스크 중 우선순위가 높게 설정된 버튼 이벤트가 먼저 수행된다. 그 결과 센서 태스크가 피스톤의 전환 지점 도착을 인지하기 전에 버튼 이벤트로 인해 이동 범위가 늘어났으므로 피스톤은 우측으로 더 진행하게 된다. 반면, [Case 2]에서는 비교적 짧은 타이머 태스크 수행 시간으로 인해 버튼 이벤트가 발생하기 전에 대기하고 있던 센서 태스크가 먼저 수행되었고 이로 인해 피스톤의 위치가 우측 방향 전환 지점에 도착했음을 인지하기 때문에 방향을 전환하게 된다. 나중에 수행된 버튼 태스크의 수행 내용은 다음 왕복 사이클에 적용된다. 이와 같이 시스템의 이벤트 구동 방식에 의해 처리되는 경우 시스템의 동작이 예측할 수 없어 이러한 상이한 동작이 시스템 요구사항을 위배할 가능성이 있으며 일시적으로 위배되지 않더라도 전체 시스템의 안정성 검증이 불가능하다.

3. 시간 구동 방식의 실시간 제어 소프트웨어 구조

태스크 처리의 이벤트 구동 방식으로 인한 예측성 문제는 2.3절의 예에서 볼 수 있듯이 센서나 엑츄에이터 혹은 제어기의 시간 차이에 따라 시스템 동작이 상이할 수 있으며 이러한 이벤트 구동 방식에 의해 동작하는 태스크나 이벤트의 수가 많아질수록 예측성 문제는 더욱 심각해 질 수 있다. 특히 안전 필수 실시간 시스템에서는 모든 경우에 대한 올바른 기능과 각 기능의 시간 제약성이 만족 되는지 검증해야하는데, 위와 같이 시스템 동작이 상이할 경우 검증해야 하는 경우의 수가 기하급수적으로 증가하여 실질적으로 완전한 검증이 불가능할 수 있다. 반면 시간 구동 방식의 실시간 제어 소프트웨어의 경우는 각 태스크의 시간 특성이 설계 시간에 정의되고 이러한 방식에 대한 시간 제약성 검증이 가능한 여러 실시간 분석 기법이 존재한다. 이를 위해 본 논문에서는 이벤트 구동 방식으로 설계된 기존의 H-VAD 인공심장 시스템의 소프트웨어를 시간 구동 방식(TTA)으로 재설계하였다. 이때 기존 H-VAD 소프트웨어를 TTA 기반으로 개선한 Time-Triggered H-VAD (TT H-VAD)를 구현하기 위해서는 설계시에 기존 태스크들의 시간적 속성에 대한 분석이 필수적이다. 이 절에서는 기존 태스크들의 시간적 속성을 측정하기 위한 소프트웨어를 구현하고 이를 이용하여 각 태스크의 발생 간격과 수행시간을 측정한다. 이를 통해 획득한 각 태스크의 시간적 속성에 기반하여 각 태스크 수행을 위한 주기를 정의하고 시간 제약성 검증을 위한 실시간 분석 기법의 하나인 RMA (Rate-Monotonic Analysis)를 통해 모든 태스크가 기존 하드웨어 상에서도 실시간성이 보장됨을 보인다.

3.1 태스크의 시간 특성 측정을 위한 소프트웨어의 설계 및 구현

H-VAD의 태스크 타이밍 측정을 위해 구현한 소프트웨어는 크게 태스크와 시간 정보를 수집하는 기능 그리고 H-VAD로부터 수집한 정보를 시리얼 통신으로 호스트 PC로 보내는 기능으로 이루어진다. H-VAD에 탑재된 마이크로 프로세서인 TMS320F2810 DSP에서 수집한 데이터를 저장하는 공간은 <표 2>의 L0 또는 L1으로 그 용량은 4k*16 워드이다. 수집되는 데이터의 단위 크기가 2바이트인 것과 태스크의 수행 시간 단위가 ms임을 감안한 데이터 생성 속도를 생각했을 때 L0, L1메모리의 용량은 자료 분석을 위해 요구되는 충분한 양의 데이터를 수집하는 데 필요한 메모리 용량보다 훨씬 부족하다. 또한 데이터의 빠른 생성 속도에 비해 H-VAD로부터 수집한 정보를 전송하는데 사용되는 시리얼 통신의 전송속도가 매우 느리기 때문에 데이터가 생성될 때마다 바로 호스트 PC로 자료를 송신할 때 통신 병목현상이 발생한다.

〈표 2〉 TMS320F2810 DSP의 메모리 구성

메모리		용량	용도
ROM/ Flash	Flash	128k*16	프로그램 코드 저장
	OTP ROM	1k*16	부트 코드, 중요 시스템 상수 저장
SARAM (Single-Access RAM)	L0, L1	4k*16	데이터 저장
	M0, M1	1k*16	다른 계열 코드 호환
	H0	8k*16	코드 저장

이러한 제약들을 극복하기 위해 태스크의 타이밍 측정 소프트웨어가 데이터만 수집하는 데이터 수집 모드와 수집된 데이터를 호스트 PC로 송신만 하는 모드로 나누어 동작하도록 설계하였다. L0, L1 메모리의 용량을 고려하여 최대한 저장할 수 있는 데이터의 수를 지정해놓고, 수집된 데이터의 수가 저장할 수 있는 최대 데이트의 수와 같아지면 데이터 수집은 잠시 멈추고 수집된 데이터를 호스트 PC로 송신만 한다. 그리고 그 송신이 끝나면 다시 데이터를 수집한다. 이 방식은 데이터 송신 모드일 동안 발생하는 데이터를 분실한다는 단점이 있지만, H-VAD와 같은 경우 시간에 따른 측정값 변화가 크지 않아 반복적인 측정으로 분실되는 복구할 수 있다.

3.2 태스크의 시간 특성 분석

3.1절에서 기술한 태스크의 시간 특성 측정을 위한 소프트웨어를 통해 H-VAD에서 수행되는 태스크들의 시간 특성

〈표 3〉 측정된 H-VAD의 태스크 시간 특성

태스크 명	종류	주기 (단위:ms)	수행 시간 (단위:ms)	이벤트 소스
CAPINT1	산발적 태스크	400	<0.01	센터 홀 센서
CAPINT2	비주기적 태스크	N/A	<0.01	버튼
CAPINT3	비주기적 태스크	N/A	<0.01	버튼
T3PINT	주기적 태스크	1	0.23	타이머 인터럽트
CAPINT4	산발적 태스크	1.3	<0.01	모터 홀 센서
CAPINT5	산발적 태스크	1.3	<0.01	모터 홀 센서
CAPINT6	산발적 태스크	1.3	<0.01	모터 홀 센서

인 태스크 수행 주기와 각 태스크 별 수행 시간을 측정하였다. 시스템의 예측 가능성과 안전성 확보를 위해 시스템이 최대 부하일 때를 기준으로 주기와 수행 시간을 측정하였으며, 주기는 각 태스크의 측정된 주기 중 가장 짧은 시간을 기준으로, 수행 시간은 측정된 수행 시간 중 가장 긴 시간을 기준으로 측정 하였다. 또한 본 연구에서는 시간 특성 중의 하나인 최대 수행 시간의 분석 (WCETA: Worst Case Execution Time Analysis)을 위해서 기존의 정적 분석 도구를 사용하지 않고 H-VAD 소프트웨어 테스팅 관련 연구 [22]에서 사용한 최대 커버리지 달성 테스트 케이스를 사용하여 측정하였다. <표 3>은 측정 결과를 나타내며 측정 결과에서는 H-VAD에 사용된 DSP의 한계로 인해 0.01ms이하의 시간 간격에 대해서는 측정이 불가능했고, 이로 인해 수행 시간이 0.01ms로 나온 태스크들은 그 수행 시간이 0.01ms 미만인 것으로 표기하였다.

3.3 시간 구동 방식의 소프트웨어 (TT H-VAD) 설계를 위한 실시간성 분석

본 논문에서 목표로 하는 시간 구동 방식의 H-VAD 제어 소프트웨어인 TT H-VAD의 설계를 위해서는 먼저 각 태스크를 수행하기 위한 시간 주기를 정의하여야 한다. 3.2절의 각 태스크 별 시간 특성 분석을 통해 <표 3>에서 센서 이벤트를 포함한 모든 태스크의 발생 주기의 최소값이 1ms보다 크기 때문에 설계 기본 수행 주기를 1ms로 정의하였다. 즉, 정의된 주기 1ms마다 이벤트 발생 여부를 확인하고 이벤트가 발생한 경우에 해당 태스크를 수행하면 모든 주기/비 주기적 이벤트에 대해 대응되는 태스크를 수행할 수 있다. 이때 전체 시스템의 실시간성 (시간 제약성) 만족 여부를 확인하기 위해 주어진 태스크들과 스케줄링 정책에 대해서 모든 경우에 대해 태스크 집합이 완료시간 전에 종료하는지를 보인다. 본 논문에서는 선점형 스케줄링 정책과 주기적 태스크 집합만을 가정하한 RMA (Rate-Monotonic Analysis)를 TT H-VAD의 조건에 맞는 비선점 형 스케줄링 [23]과 산발적 태스크 집합 조건 [24]을 포함하도록 확장하여 사용하였다. 즉, RMA를 확장하여 수행 주기가 가변적이되 최소 수행 주기가 알려진 산발적 태스크를 포함할 수 있게 하였다.

이를 위해 Mok [23]의 접근 방법을 적용하면, 산발적 태스크 집합 $M \ni T_i(c_i, p_i, d_i)$을 (수식 1) 따라 동치인 주기적 태스크 집합 $M' \ni T'_i(c'_i, p'_i, d'_i)$으로 변환하였을 때 태스크 집합 M'이 RMA를 통해 스케줄링 가능하다면 원래 태스크 집합 M도 스케줄링 가능하다.

$$c'_i = c_i$$
$$p'_i = \min(p_i, l_i + 1CPU\ cycle\ time)$$
$$d'_i = c_i \qquad \text{(수식 1)}$$

여기서 c, p, d는 각각 수행시간, 최소 주기, 임계 시간을 의미하여, l은 지연시간이며 각 태스크가 임계시간을 위배하지 않고 지연될 수 있는 최대 시간을 의미한다. 여기에서 기본값으로 지연 시간은 임계 시간과 수행 시간 사이의 차를 갖도록 한다. Mok의 방법에 따라 지연시간을 주기로 간주하였을 때 산발적 태스크의 수행 시점에 상관없이 임계시간을 준수하는 주기적 태스크를 구할 수 있다. <표 4>는 앞서 설명한 Mok의 방법에 따라 변환된 H-VAD의 태스크 집합 M'의 시간적 특성을 기술하며 이를 바탕으로 RMA를 수행했을 때 총 CPU 사용 정도를 (수식 2)에 따라 구하였다.

〈표 4〉 변환된 태스크 집합 M'

태스크 명	주기(=임계 시간) (단위 ms)	수행 시간 (단위 ms)	지연 시간 (단위 ms)
CAPINT1 ~ 6	0.99 + 1CT	0.01	0.99
T3PINT	0.77 + 1CT	0.23	0.77

CT: Cycle time, 6.67ns

$$U_{M'} = \sum_{i=1}^{7} \frac{c'_i}{p'_i} \approx 0.35 \qquad \text{(수식 2)}$$

이때 계산된 CPU 사용 정도가 스케줄링 가능성을 판단하는 임계 값보다 작을 경우 태스크 집합 M'은 항상 스케

줄링 가능한데, 임계치를 Park [24]의 n개의 주기적 태스크가 비 선점 형 스케줄러에 의해 수행될 때의 임계치인 (수식 3)에 적용할 결과 계산된 임계 값은 0.77로 <표 4>에서와 같이 변환된 TT H-VAD 태스크 집합 M'의 0.35보다 크므로 TT H-VAD의 태스크 집합은 항상 스케줄링 가능함을 확인하였다.

$$U_{M'} = \sum_{i=1}^{7} \frac{c'_i}{p'_i} \leq \frac{1}{r}$$

$$r = \frac{The\ logest\ period}{The\ shortest\ period} \qquad \text{(수식 3)}$$

4. TT-HVAD 구현 및 실험 결과

4.1 이벤트 구동 방식 태스크의 시간 구동 방식으로의 구현

3절의 시간 구동 방식의 소프트웨어 설계와 측정을 통해 획득된 시간 특성을 고려하여 TT H-VAD를 구현하기 위해서 정의된 주기인 1ms마다 이벤트 발생 여부를 확인하고 이벤트가 발생한 경우 수행하는 모니터링 코드를 구현하였다. 이를 위해 같은 수행 주기를 갖는 기존 H-VAD의 태스크 중 T3PINT 태스크를 (그림 5)와 같이 수정하였다. 이벤트 발생 여부를 확인하는 데에 있어서 기존 H-VAD의 이벤트 구동 방식 구조를 위해 사용되던 인터럽트 핸들러를 비활성화 하였고 대신 마이크로 프로세서에 내장된 이벤트 상태 레지스터를 이용하여 각 이벤트 발생 여부를 확인하였다.

기존의 H-VAD 소프트웨어 구조에서 모터가 회전함에 따라서 발생하는 이벤트 (CAPINT)는 그 물리적인 특성상 회전 방향으로 순차적으로 발생하기 때문에 제안된 TT H-VAD에서 새로운 태스크 정의에서 조건문으로 과거 이벤트의 발생 여부를 검사한 후에 이를 모터 제어의 입력으로 사용을 해도 이벤트 처리의 순서가 바뀌는 현상이 발생하지 않으며 이에 따른 우선순위 역전 현상도 발생하지 않는다.

기존의 인터럽트 기반의 방식 (Orig. H-VAD)에서 2.4절의 (그림 4)와 같이 발생이 가능하였던 시간 차이에 따른 동작의 상이한 결과는 개선된 시간 구동 방식의 소프트웨어 설계(TT H-VAD)에서 발생하지 않음을 확인하기 위해서 H-VAD 인공심장의 PR (Pumping Rate), SL (Stroke Length)의 두 설정 값을 설정 기본값을 포함하여 최악의 경우에 대해 이벤트 간 시간차이를 측정하였다. PR은 펌프가 한쪽 끝에서 다른 쪽 끝으로 이동하는 빈도를 의미하고 값이 클수록 펌프가 빨리 동작하며, SL은 펌프가 이동하는 양 끝 단의 거리를 의미하여 거리가 짧을수록 펌프가 빨리 동작한다. 단, 이때 펌프의 물리적인 동작 한계상 그 값들은 제한되어 실험에는 물리적으로 가능한 최악의 값을 각각 적용하여 기존의 인터럽트 기반의 방식에서 발생하는 인터럽트의 발생 간격을 측정하였고 개선된 시간 구동 방식의 소프트웨어에서 주기적으로 동작하는 태스크의 수행 간격을 측정하였다. 실험 결과는 (그림 6)과 같으며 점선 안의 결과를 보면 TT H-VAD에서는 Orig. H-VAD의 어떠한 최악의 경우의 동작 환경일지라도 주기적으로 태스크의 수행 간격이 더 짧기 때문에 2.4절의 (그림 4)에서 발생하였던 이벤트를 잃어버리는 경우가 발생하지 않는다는 것을 확인하였다.

또한, (그림 7)과 같이 위의 실험에서 최악의 설정에 대해 이벤트 처리의 지연시간을 측정한 결과 주기적 태스크의 데드라인 (태스크의 주기와 동일)인 1ms보다 작은 0.3m 이내로 나타나 개선된 소프트웨어 설계 구조에 따른 TT H-VAD는 모든 시간 제약성을 만족하는 것을 확인하였다.

구현된 TT H-VAD는 주지적인 태스크에 의해 1ms마다 이벤트 발생 여부를 확인하고 이벤트가 발생한 경우 수행하는 모니터링 코드를 수행하였다. 이를 통해 모터의 회전에 따라 발생하는 이벤트에 의한 인터럽트를 제거할 수 있지만 모터 제어만을 수행하던 기존의 주기적인 태스크에 모니터링을 위한 추가적인 오버헤드가 발생하게 된다. 이러한 오버헤드를 확인하기 위해 인공심장의 기본 설정 및 PR, SL의 최악의 설정에 대해 펌프의 한 주기(한쪽 끝에서 다른 쪽 끝으로 이동)의 CPU 수행 시간을 각각 10회 측정하여 평균을 구하였다.

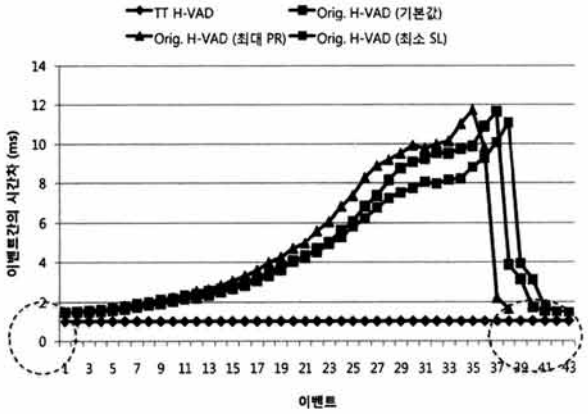
(그림 8)의 실험 결과를 보면 제안된 TT H-VAD가 오히려 CPU 수행 시간이 기존의 소프트웨어 구조에 비해 근소하게 작은 것을 알 수 있다. 그 원인을 분석해보면 모터의 회전에 따른 다수의 하드웨어 인터럽트 발생에 따른 인터럽트 지연시간, 문맥 교환 시간 등의 인터럽트 오버헤드가 제

```
<기존 태스크 정의>
1:  interrupt CAPINT1()
2:      <service routines>
3:      Reset event flag CAP1FLAG to receive more interrupt
4:  endinterrupt
5:  interrupt CAPINT2()
6:      <service routines>
7:      Reset event flag CAP2FLAG to receive more interrupt
8:  endinterrupt
9:  ...
10: interrupt T3PINT()
11:     <service routines>
12:     Reset event flag T3FLAG to receive more interrupt
13: endinterrupt
```

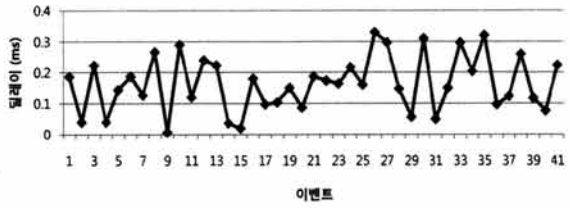
```
<새로운 태스크 정의>
1:  interrupt T3PINT()
2:      if CAP1FLAG is TRUE
3:          <CAPINT1 service routines>
4:          Reset event flag CAP1FLAG to receive more interrupt
5:      endif
6:      if CAP2FLAG is TRUE
7:          <CAPINT2 service routines>
8:          Reset event flag CAP2FLAG to receive more interrupt
9:      endif
10:     ...
11:     <T3PINT service routines>
12:     Reset event flag T3FLAG to receive more interrupt
13: endinterrupt
```

(그림 5) 구현된 이벤트 모니터링 코드

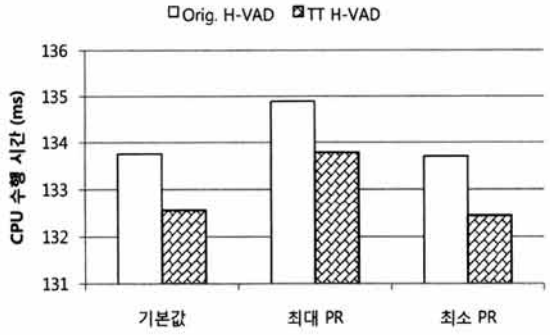
안된 소프트웨어 구조에 사용된 모니터링 코드에 비해 오버헤드가 큰 것을 알 수 있다. 단, 본 논문에서 제안된 소프트웨어 설계 구조 개선에 따라 이벤트 처리를 주기적인 태스크에 의해 순차적으로 처리할 경우 태스크의 주기에 따라 이러한 오버헤드는 증가할 수 있다. 즉, 주기적인 태스크에 의해 증가되는 오버헤드가 인터럽트에 의한 오버헤드보다 작을 경우 특히 본 논문에서 제안된 소프트웨어 설계 구조 개선의 방법이 효과적이며, 이는 인터럽트에 의한 오버헤드와 주기적인 태스크에 의한 오버헤드가 서로 상반관계(trade-off) 에 있다는 점을 나타낸다.



(그림 6) 펌프가 한쪽 끝에서 다른 끝으로 이동할 때 발생한 이벤트간의 시간차



(그림 7) 최악의 설정 값에 대한 이벤트 처리시간

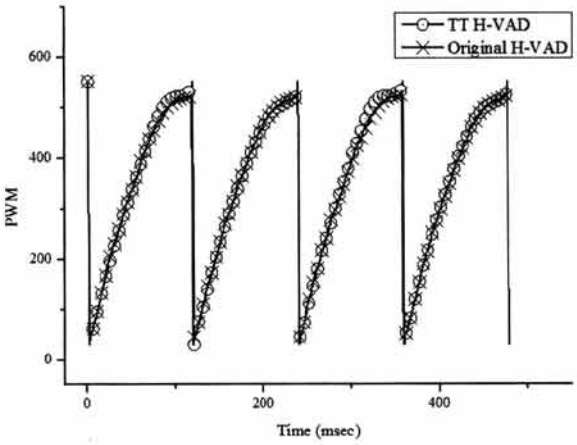


(그림 8) 펌프가 한쪽 끝에서 다른 끝으로 이동할 때 CPU가 수행된 시간

4.2 개선된 소프트웨어의 동작 동일성 확인을 위한 실험

본 논문에서는 이벤트 구동 방식의 소프트웨어 설계 구조를 시간 구동 방식으로 개선함으로써 시스템의 예측성을 향상 시킬 수 있음을 기술하였다. 단, 개선된 소프트웨어 설계 구조에 의해 제어된 모터의 동작이 기존의 소프트웨어 구조에 의해 제어된 모터의 정상적인 상황에서의 동작과 동일함을 보여야 한다. H-VAD 인공심장 시스템은 모터의 센서를 통해 펌프의 상태를 확인하고 모터의 제어를 위해 마이크로프로세서에 탑재된 제어 소프트웨어에서 계산된 PWM (Pulse-width modulation) 값을 모터의 제어 신호로 내보내는데 본 절에서는 동작 동일성을 확인하기 위해 각 소프트웨어 설계 구조에 의해 계산된 제어 값인 PWM의 값과 계산되는 시간을 비교해보았다.

(그림 9)은 PWM 변화를 비교한 것으로 본 논문에서 제안된 TT H-VAD가 기존 H-VAD와 비교했을 때 거의 동일한 모터 출력 특성을 보임을 확인할 수 있다.



(그림 9) 소프트웨어 설계 별 PWM 비교

5. 결 론

인공심장 시스템은 오동작시에 환자의 생명 유지에 큰 위협이 될 수 있는 안전 필수 실시간 시스템으로 이러한 시스템의 검증에는 시스템 동작 면에서 높은 예측성을 요구한다. 기존의 많은 실시간 시스템은 설계의 편의성과 효율인 구현으로 인하여 인터럽트를 기반으로하고 있는 이벤트 구동 방식의 소프트웨어 설계 구조를 채택하여 왔다. 그러나 이러한 이벤트 구동 방식의 경우 본 논문에서 제시한 것과 같이 제어되는 물리 환경의 시간 차이에 따라 그 동작이 상이할 수 있으며 이는 곧 시스템의 예측성에 영향을 미치게 된다. 반면 주기적인 태스크에 의해 제어되는 시간 구동 방식의 소프트웨어 구조는 주어진 주기의 시간에 따라 모든 동작이 이루어지기 때문에 예측이 쉽다. 이를 위해 본 논문에서는 인공심장 시스템이라는 실질적인 시스템에 대해서 이벤트 구동 방식 기반의 안전 필수 실시간 시스템을 시간 구동 방식 기반으로 재설계하는 과정을 보이고 두 시스템의 동작 비교 결과를 제시했다. 이를 위해 시간 구동 방식 소프트웨어 구조로 재설계 하는 과정에서 기존 시스템 태스크들의 시간적 속성을 측정하는 소프트웨어를 구현하였고 여기서 측정된 결과를 RMA 실시간 분석기법에 적용하여 시

스템의 실시간성을 만족시킬 수 있는 적정 태스크 수행 주기를 결정하였다. 이를 바탕으로 인공심장 제어 소프트웨어의 구현을 개선하여 다양한 실험을 통해 정량적으로 분석하여 개선된 소프트웨어를 탑재한 인공심장 시스템은 시스템의 동작 예측도를 획기적으로 높여주면서 기존의 인공심장 시스템과 동일하게 동작함을 확인하였다.

참 조 문 헌

[1] American Heart Association, Heart disease and stroke statistics 2010 update at-a-glance.

[2] US Department of Health and Human Services, The 2008 annual report of the OPTN and SRTR: heart transplantation.

[3] US Department of Health and Human Services, Organ procurement and transplantation network national data.

[4] Texas Instruments, Inc., "Digital Signal Processors (DSP)", http://www.ti.com/home_tsw_dsp.

[5] Analog Devices, Inc., "Motor and Power Control", http://motorcontrol.analog.com/.

[6] D. Frund, "Toward a formal definition of timing predectability," Talk at Workshop on Reconciling Performance with Predictability, October, 2009.

[7] G. Candea, "Predictable software-a shortcut to dependable computing?," CoRR, 2004.

[8] M. Short, "Development guidelines for dependable real-time embedded systems," Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on, pp.1032 - 1039, 31 2008-april 4 2008.

[9] M. Pont, "Patterns for time-triggered embedded systems: Building reliable applications with the 8051 family of microcontrollers," ACM Press/Addison-Wesley Publishing Co., 2001.

[10] A. Albert, "Comparison of event-triggered and time-triggered concepts with regard to distributed control systems," Embedded World, vol.2004, pp.235 - 252, 2004.

[11] I. Bate, "Scheduling and timing analysis for safety critical real-time systems," UNIVERSITY OF YORK DEPARTMENT OF COMPUTER SCIENCE-PUBLICATIONS-YCST SCIENCEPUBLICATIONS-YCST, 1999.

[12] H. Kopetz, "Event-triggered versus time-triggered real-time systems," Operating Systems of the 90s and Beyond, pp.86 - 101, 1991.

[13] G. Jeong, C. Hwang, K. Nam, C. Ahn, J. Lee, J. Choi, and H. Son, "Development of a closed air loop electropenumatic cctuator for driving a penumatic blood pump," Artificial Organs, Vol.33, pp.657 - 662, 2009.

[14] J. Lee, B. Kim, J. Choi, and C. Ahn, "Optimal pressure regulation of the penumatic ventricular assist device with bellows-type driver," Artificial Organs, Vol.33, pp.627 - 633, 2009.

[15] C.L. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," Journal of the ACM, Vol.20, No.1, pp.46 - 61, January, 1973.

[16] T. Pop, P. Eles, Z. Peng, "Holistic scheduling and analysis of mixed time/event-triggered distributed embedded systems," CODES '02 Proceedings of the tenth international symposium on Hardware/software codesign, 2002.

[17] A. Albert, "Comparison of Event-Triggered and Time-Triggered Concepts with Regard to Distributed Control Systems," Embedded World, pp.235-252, 2004.

[18] H. Kopetz, A. Damm, C. Koza, M. Mulazzani,W. Schwabl, C. Senft, and R. Zainlinger, "Distributed fault-tolerant real-time systems: the mars approach," Micro, IEEE, Vol.9, No.1, pp.25 - 40, Feb., 1989.

[19] K. Sandstrom, C. Eriksson, G. Fohler, "Handling interrupts with static scheduling in an automotive vehicle control system," Fifth International Conference on Real-Time Computing Systems and Applications, pp.158-165, 1998.

[20] G. LeLann, "Critical issues in real-time computing," Proceedings of Workshop on Communication Networks and Distributed Operating Systems within the Space Environment, October, 1989.

[21] Y. Itami, T. Ishigooka, T. Yokoyama, "A Distributed Computing Environment for Embedded Control Systems with Time-Triggered and Event-Triggered Processing," Embedded and Real-Time Computing Systems and Applications, pp.45-54, 2008.

[22] S. Cha, S. Jeong, J. Yoo, and Y. Kim, "Testing of safety-critical software embedded in an artificial heart," Advances in Systems Safety, pp.143 - 153, 2011.

[23] A. Mok, "The design of real-time programming systems based on process models," Proc. of IEEE Real-Time Systems Symposium, pp.5 - 17, 1984.

[24] M. Park, "Non-preemptive fixed priority scheduling of hard realtime periodic tasks," Computational Science - ICCS 2007, pp.881 - 888, 2007.

정 세 훈

e-mail : gifaranga@korea.ac.kr

2010년 고려대학교 컴퓨터학과(학사)

2010년~현 재 고려대학교 컴퓨터학과 석사과정

관심분야 : 소프트웨어 테스팅, 정형 검증 등

김 희 진

e-mail : heejin.kim@ewhain.net
2010년 이화여자대학교 컴퓨터공학과(학사)
2010년~현 재 이화여자대학교 컴퓨터공학과 석사과정
관심분야 : 제어시스템, RTOS, 모바일, 시뮬레이션 등

박 상 수

e-mail : sangsoo.park@ewha.ac.kr
1998년 한국과학기술원 전산학과(학사)
2000년 서울대학교 컴퓨터공학과(공학석사)
2006년 서울대학교 컴퓨터공학과(공학박사)
2006년~2008년 Univ. of Michigan Research Fellow
2008년~2008년 삼성전자 종합기술원 전문연구원
2009년~현 재 이화여자대학교 컴퓨터공학과 전임강사
관심분야 : 임베디드 소프트웨어, 실시간 시스템, 시스템 시뮬레이션 등

차 성 덕

e-mail : scha@korea.ac.kr
1983년 UC Irvine 전산학과(학사)
1986년 UC Irvine 전산학과(전산학 석사)
1991년 UC Irvine 전산학과(전산학 박사)
1991년~1994년 The Aerospace Corporation 기술 위원
1994년~2008년 한국과학기술원 교수
2008년~현 재 고려대학교 교수
2009년~현 재 고려대학교 고신뢰 융합 소프트웨어공학 연구 센터 센터장
관심분야 : 소프트웨어 신뢰성 보장 및 테스팅, 요구공학, 웹 보안 등