석사학위논문

바이오/의료 응용에 특화된 임베디드 커널

Embedded Kernel for
Bio/Medical Applications

지도교수 : 조진성

경희대학교 대학원
컴퓨터공학과

백용규

2009년 2월

바이오/의료 응용에 특화된 임베디드 커널

Embedded Kernel for
Bio/Medical Applications

지도교수 : 조진성

이 논문을 석사 학위논문으로 제출함

경희대학교 대학원
컴퓨터공학과

백용규

2009년 2월

백용규의 공학 석사학위 논문을 인준함

주심교수 : 홍 충 선 ㊞

부심교수 : 허 의 남 ㊞

부심교수 : 조 진 성 ㊞

경희대학교 대학원

2009년 2월

국 문 요 약

바이오/의료 응용에 특화된 임베디드 커널

경희대학교 대학원
컴 퓨 터 공 학 과
백 용 규

인구 고령화와 함께 생활양식 및 환경의 변화로 사람들의 건강에 관한 관심이 높아지고 있다. 언제, 어디서나 자신의 건상 상태를 알고 싶어하는 사람들의 욕구로 바이오/의료 분야가 더욱 더 발전하고 있다. 또한 정보 통신의 발전으로 Ubiquitous 기술을 이용한 u-Lifecare에 대한 관심이 높아지고 있다. u-Lifecare는 아픈 사람뿐만 아니라 건강한 사람들을 포함하는 서비스로, 시간과 공간의 제약 없이 원격 의료 기술을 활용한 건강 관리 서비스를 말한다. u-Lifecare가 구현된 환경에서는 환자가 의식하지 않은 상태에서 환자의 건강 상태를 실시간으로 모니터링하고 환자의 상태가 악화되면 바로 응급센터 및 병원 등에 환자 상태에 대한 정보가 전달되어 신속한 의료 서비스가 이루어진다.

이러한 바이오/의료분야는 임베디드 시스템의 한 분야에 속한다. 임베디드 시스템은 특정 목적을 수행하기 위해 설계된 시스템으로 우리가 쉽게 접할 수 있는 PDA, 휴대폰, TV, MP3, 셋탑 박스, 디지털 카메라, 바이오/의료 응용 등 일상 생활과 매우 밀접한 관계를 가지고 있다.

바이오/의료 분야에 사용되는 프로그램은 이전에는 단순한 기능만을 수행하기 위해 펌웨어 수준의 프로그램이 사용되었지만 근래에는 다양한 기능과 복잡한 작업을 수행을 하게 되어 임베디드 시스템에 사용되는 실시간 운영체제를 사용하고 있다.

그러나 기존의 실시간 운영체제는 바이오/의료의 요구사항을 반영하지 못하고 있다. 따라서 본 논문에서는 바이오/의료의 요구사항을 분석 한 뒤 이를 반영하는 Bio-K 임베디드 커널을 설계하고 구현하였다. Bio-K는 실시간 운영체제로 바이오/의료분야의 요구사항인 시간성, 안정성, 저 전력, 개발 편의성을 제공한다.

키워드 : 바이오/의료, 임베디드 시스템, 실시간 시스템, 실시간 운영체제, 저전력 시스템, Bio-K, 바이오 커널, DP-DVS

<목 차>

<그 림 목 차>

<표 목 차>

1. 서론

인구 고령화와 함께 생활양식 및 환경의 변화로 사람들의 건강에 관한 관심이 높아지고 있다. 언제, 어디서나 자신의 건상 상태를 알고 싶어 하는 사람들의 욕구로 바이오/의료 분야가 더욱 더 발전하고 있다. 또한 정보 통신의 발전으로 Ubiquitous 기술을 이용한 u-Lifecare에 대한 관심이 높아지고 있다. u-Lifecare는 아픈 사람뿐만 아니라 건강한 사람들을 포함하는 서비스로, 시간과 공간의 제약 없이 원격 의료 기술을 활용한 건강관리 서비스를 말한다. u-Lifecare가 구현된 환경에서는 환자가 의식하지 않은 상태에서 환자의 건강 상태를 실시간으로 모니터링하고 환자의 상태가 악화되면 바로 응급센터 및 병원 등에 환자 상태에 대한 정보가 전달되어 신속한 의료 서비스가 이루어진다.

임베디드 시스템은 특정 목적을 수행하기 위해 설계된 시스템으로 우리가 쉽게 접할 수 있는 PDA, 휴대폰, TV, MP3, 셋탑 박스, 디지털 카메라, 바이오/의료 응용 등 일상생활과 매우 밀접한 관계를 가지고 있다.

임베디드 시스템의 큰 특징은 실시간 시스템이다. 실시간 시스템은 정해진 시간 내에 결과를 처리해야 하는 시스템으로 주어진 작업을 빨리 처리한다는 의미가 아니라 정해진 시간을 넘겨서는 안 된다는 의미이다. 실시간 시스템은 크게 경성 실시간, 연성 실시간으로 구분된다. 경성 실시간은 정해진 시간 내에 작업의 결과가 절대적으로 처리되어야 하는 시스템으로 비행기의 비행제어 시스템, 핵발전소의 제어 시스템을 예를 들 수 있다. 연성 실시간 정해진 시간 내에 작업의 결과가 처리되지 않더라도 치명적인 결과를 초래하지 않고 시스템 오류가 되지 않는 시스템으로 라우터와 핸드폰을 예를 들 수 있다. 이 중 바이오/의료 응용은 사람의 생명을 다루는 분야로 경성 실시간 시스템에 속한다. 바이오/의료 응용은 수술기구, 해부장비, 진단기기 등 매우 광범위하게 존재하며, 단순한 기계식 제어 프로그램에서 임베디드 소프트웨어로 대체하고 있다. 이러한 임베디드 소프

트웨어도 점차 다양한 기능을 요구하여 최근에는 실시간 운영체제를 사용하고 있다.

실시간 운영체제는 마이크로프로세서가 내장된 제한된 하드웨어를 가지고 주변 상황을 고려하여 요구되는 기능을 효율적으로 수행하는 임베디드 시스템을 위해 작은 크기로 최적화된 운영체제이다. 이러한 실시간 운영체제는 바이오/의료 응용의 요구사항을 제대로 반영하지 못하고 있다. 따라서 본 논문에서는 바이오/의료 응용의 요구사항을 분석하여, 바이오/의료 응용에 특화된 임베디드 커널을 설계 및 구현한다.

본 논문은 2장에서 바이오/의료 분야에 대해 알아본 후, 바이오/의료 분야의 요구사항을 분석 하고 3장에서 바이오/의료 분야의 요구사항을 반영한 Bio-K 임베디드 커널의 설계에 대한 내용을 다루며, 4장에서는 구현 5장에서는 평가 6장에서는 관련 연구에 대한 내용을 다루며, 마지막 7장에서는 결론 및 향후 연구로 구성이 된다.

2. 바이오/의료 응용

2.1 바이오/의료

인구 고령화와 함께 생활양식 및 환경의 변화로 사람들의 건강에 관한 관심이 높아지고 있다. 언제, 어디서나 자신의 건상 상태를 알고 싶어 하는 사람들의 욕구로 바이오/의료 분야가 더욱 더 발전하고 있다. 또한 정보 통신의 발전으로 Ubiquitous 기술을 이용한 u-Lifecare에 대한 관심이 높아지고 있다. u-Lifecare는 아픈 사람뿐만 아니라 건강한 사람들을 포함하는 서비스로, 시간과 공간의 제약 없이 원격 의료 기술을 활용한 건강 관리 서비스를 말한다. u-Lifecare가 구현된 환경에서는 환자가 의식하지 않은 상태에서 환자의 건강 상태를 실시간으로 모니터링하고 환자의 상태가 악화되면 바로 응급센터 및 병원 등에 환자 상태에 대한 정보가 전달되어 신속한 의료 서비스가 이루어진다.

2.2 바이오/의료 요구사항

2.2.1 멀티 태스킹

바이오/의료 응용의 프로그램 다양한 센서를 부착하여 동시에 여러 개의 채널(ADC)을 통해 센서 정보를 수집한다. 수집한 정보를 가공하고, 네트워크를 통해 전달하고, 출력 장치를 통해 사용자에게 정보를 제공 등 다양한 기능을 수행하기 위해서는 바이오/의료용 운영체제는 멀티 태스킹을 지원해야 한다.

2.2.2 안전성

바이오/의료 응용은 사람의 생명을 다루는 분야로 수행되는 프로그램이 정확한 동작을 보장해야 하며, 프로그램이 멈추면 안 된다. 혹 프로그램이 멈추는 경우가 발생하면 바로 판단하여 재실행 시켜야 된다.

2.2.3 실시간성

바이오/의료 응용은 경성 실시간 시스템으로 사람의 생명과 관련된 센서의 정보 수집과 같이 주기적으로 처리하며, 해당 작업은 주어진 시간 내에 처리해야 되는 특징을 가지고 있다. 따라서 바이오/응용 운영체제는 이러한 태스크들을 쉽게 생성하는 인터페이스를 제공해야 하며, 시간성을 만족해야 한다.

2.2.4 스케줄링

바이오/의료 응용은 경성 실시간 시스템이므로 데드라인 및 주기를 만족하는 스케줄링을 제공해야 한다. 데드라인 기반 스케줄링으로 EDF, LLF 스케줄링 및 우선순위 기반의 스케줄링을 지원해야 한다.

2.2.5 네트워크 지원

바이오/의료 응용은 u-Healthcare 의료 시장의 대중화와 무선 네트워크의 비중이 커지면서 수요가 증가하고 있는 추세이다. 무선 네트워크의 사용이 요구됨에 따라 바이오/응용 운영체제에서는 TCP/IP, Zigbee 등의 네트워크를 지원해야 한다.

2.2.6 저 전력

바이오/의료 응용의 경량화 및 휴대용으로 인해 주로 배터리로 동작한다. 이로 인해 전력소비에 대한 문제가 큰 이슈가 되고 있다. 바이오/응용 운영체제는 사용하지 않는 구간에서는 슬립모드(sleep mode)로 전환하는 저 전력 기능을 지원해야 한다.

2.2.7 빠른 부팅

바이오/의료 응용은 원하는 시간에 바로 수행이 가능해야 된다. PC와 같은 환경처럼 부팅시간이 10초 이상이 걸리게 되면 긴급을 요구하는 상황에서는 큰 문제가 된다. 따라서 빠른 부팅을 통해 원하는 기능을 바로 제

공받아야 한다.

2.2.8 자가 진단

바이오/의료 응용 중 일부는 스스로 판단하여 처방하는 시스템을 요구한다. 예를 들어 당뇨병 환자 몸속에 인슐린을 투여하는 칩을 내장하였을 때 칩을 직접 컨트롤 하지 못하는 상황이 발생한다. 따라서 칩 자체에서 혈당치를 체크해 분석한 후 인슐린을 투여할지 여부를 정해야 한다.

2.2.9 업그레이드

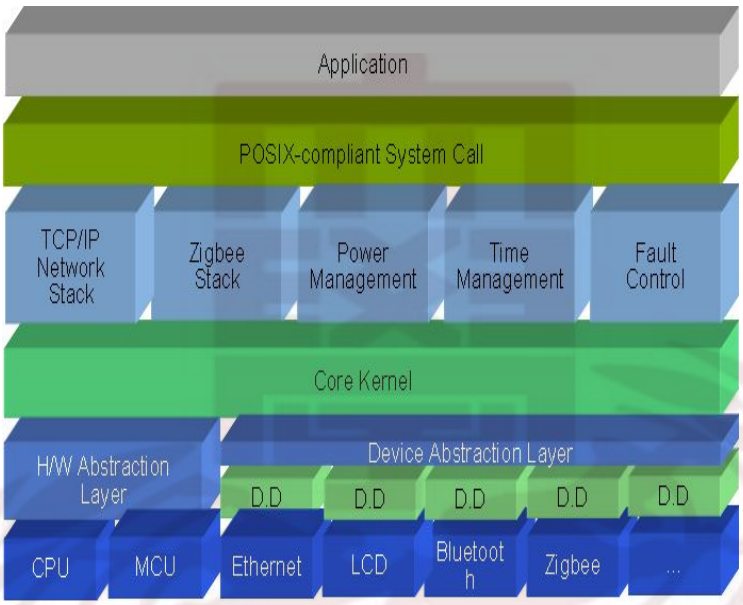
바이오/의료 응용 프로그램은 사람의 몸속에 있는 경우 모듈의 업그레이드에 문제가 된다. 사람의 몸속에 있는 칩을 다시 꺼내 업그레이드를 할 수 없기 때문에 원격을 통한 업그레이드, 운영체제에서 동적으로 모듈 구성하는 방법 등이 지원이 되어야 한다.

2.2.10 개발 시간

바이오/의료 응용 프로그램은 소량으로 생산되지만 다양한 종류의 응용 프로그램이 존재한다. 따라서 개발에 들어가는 시간을 줄이기 위해 응용 프로그램 작성에 편한 인터페이스를 제공해야 한다. POSIX 기반의 API 제공이 대표적인 예가 될 수 있다.

3. Bio-K

3.1 Bio-K 구성



[그림 1] Bio-Kernel 구성도

3.2 커널의 기능

3.2.1 HAL(Hardware Abstract Layer)

임베디드 시스템 하드웨어의 발전으로 임베디드 시스템에 사용되는 MCU가 다양해 졌다. 이로 인해서 새로운 MCU에 이식에 쉽게 되어야 되는 구조로 필요로 하였고 이를 위해 HAL을 정의하게 되었다. HAL은 모든 하드웨어 의존적인 부분을 모아서 이식성을 향상 시켰다. HAL의 내용은 startup, context switching, UART, Timer, GPIO 등이다. 이로 인해 다른 프로세서에 이식할 경우 다른 기능들의 수정 없이 HAL의 수정만을 통해 다른 프로세서에 쉽게 이식이 가능하다. 뿐만 아니라 전통적인 시스

템 개발 과정에서 하드웨어가 먼저 개발되고 이 후에 소프트웨어가 개발되는 구조에서 HAL을 통해 하드웨어와 소프트웨어를 병렬적으로 개발하는 것이 가능해졌다.

3.2.2 다중 쓰레딩(Multi-Threading)

Bio-Kernel은 다중 쓰레딩 방식으로 동작한다. 각 쓰레드는 실시간성이 지원되는 쓰레드이며 POSIX 기반의 pthread API를 지원하게 된다.

3.2.3 스케줄링

Bio-Kernel는 기본적으로 다중 쓰레딩 뿐만 아니라 실시간성이 요구되는 임베디드 시스템을 위하여 고정 우선순위 선점형 스케줄링을 지원한다. 또한 범용성을 위해 동일한 우선순위의 경우 FIFO와 Round-Robin 스케줄링을 지원하게 되며, Round-Robin내의 시간 할당량을 조절 가능하다.

Bio-Kernel은 이러한 실시간성 보장 뿐만 아니라 기능의 수정 및 확장이 쉽게 가능한 구조로 되어 있어 다양한 임베디드 시스템의 요구를 반영할 수 있다.

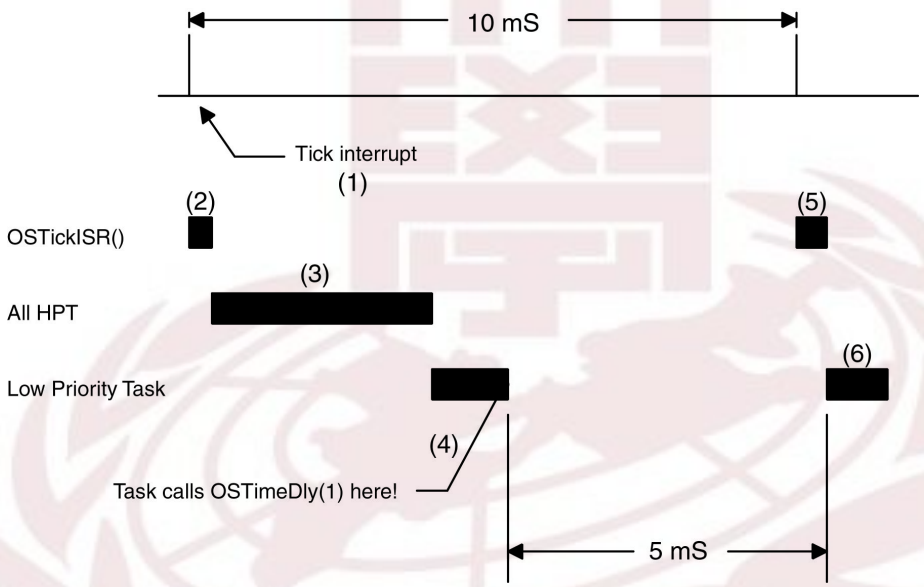
3.2.4 네트워크(Network)

u-Healthcare의 발전으로 무선 네트워크의 사용이 늘어남에 따라 무선 네트워크를 지원해야 한다. 인터넷과의 통신을 위한 TCP/IP 뿐만 아니라 센서에서의 통신을 위한 Zigbee까지 구현이 되어야 하며, 해당 모듈들은 경량화 시켜 적용 시켜야 된다.

3.3 주기적 쓰레드(Periodic Thread)

바이오/의료 응용의 실시간성을 위해 해당 태스크가 주기적으로 생성될 수 있도록 하며, 생성된 태스크의 데드라인을 보장하도록 해야 한다. 주기적 thread는 POSIX API의 pThread_create() 함수를 수정하여 제공한다.

일반적인 운영체제에서의 주기적 쓰레드의 구현은 타이머를 이용하거나 S/W 함수인 sleep을 이용한다. 그러나 타이머의 개수가 제한적인 상황과 sleep을 이용한 주기적 쓰레드는 정확성에 대한 문제가 있다.



[그림 2] 일반적인 운영체제에서의 주기적 쓰레드 구현 시 문제점

3.4 안정성(stability)

바이오/의료 응용의 안정성을 위해 Watchdog Timer를 이용한 Watchdog thread를 만들어 전체 시스템에 이상이 생길 경우 리셋을 시켜주게 된다. 또한 각 thread의 주기를 이용해 thread가 이상 유무를 체크하여 해당 thread의 이상이 있을 경우 해당 thread만 리셋 시키는 Watchcat thread를 제공한다.

3.4.1 WatchDog 쓰레드

Watchdog Timer를 이용해 전체 시스템의 이상 유무를 검사한다. Watchdog 쓰레드는 Watchdog Timer를 활성화 시킨 후 주기적으로 Watchdog를 리셋 해준다. 하지만 전체 시스템에 문제가 생길 경우 리셋을 하지 못해 Watchdog Timer가 0이 되면 전체 시스템에 이상이 있는 것으로 판단하고 해당 시스템을 리셋 시킨다.

[그림 3] Watchdog Timer

3.4.2 WatchCat 쓰레드

WatchCat Thread는 S/W 기능으로 쓰레드 별로 이상 유무를 검사한다. 스케줄 동안 쓰레드의 데드라인을 검사하여 데드라인을 만족하지 못하면 WatchCat 리스트에 해당 쓰레드를 등록한다. WatchCat 쓰레드가 활성화 시 WatchCat 리스트에 등록된 쓰레드들을 POSIX API 중 pthread를 이용해 리셋을 시킨다.

3.5 전력관리(Power Management)

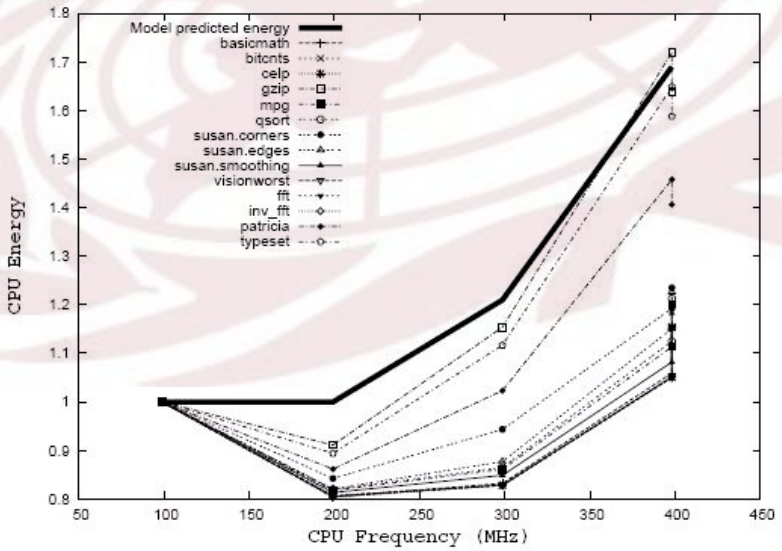
임베디드 시스템에서 사용되는 기기들은 센서와 같이 배터리 중심으로 운영되는 경우가 많다. 따라서 전력 소비를 줄이기 위한 방안이 많이 연구 되고 있다.

따라서 KHIX 임베디드 시스템 운영체제는 이러한 전력 소비를 줄이기 위한 기능을 위해 전력 관리 기능을 추가하였다. 전력 관리 기능은 크게 DVS(Dynamic Voltage Scaling)과 DPM(Dynamic Power Management) 2가지 기법이 존재 한다. DVS 기법을 위해 MCU의 클럭이 조절 가능하

도록 하고, DPM 기법을 위해 MCU의 모드를 변경 가능하도록 한다. 이러한 기법을 가상 디바이스 드라이버 형태로 구현하여 응용 프로그램 작성 시 쉽게 사용 가능하도록 한다. 또한 이러한 DVS, DPM 기법을 스케줄링 레벨에서 특정 알고리즘을 적용하는 방안도 사용 가능하다.

3.5.1 문제점 및 제안 사항

기존의 DVS 알고리즘은 CPU의 전력 소비는 전압의 제곱에 비례한다고 가정하고 제안되어져 있다. 그렇지만 실제 전력 소비를 측정해 보면 그림4와 같이 PXA255에서는 이론적이라면 가장 낮은 CPU 클럭에서 가장 낮은 전력 소비를 보여야 되지만 가장 낮은 100Mhz가 200Mhz보다 전력 소비가 큰 것을 볼 수 있다. 또한 대부분의 DVS 알고리즘이 각 태스크의 수행시간을 WCET로 가정하고 있다. 하지만 실제 시스템에서는 WCET를 사전에 체크 할 수 없다는 문제점이 발생한다. 따라서 본 논문의 DP-DVS 알고리즘은 CPU의 클럭 별 실제 전력 소비와 WCET 대신 실제 수행 시간 측정에 기반하는 알고리즘을 제안한다.



[그림 4] PXA255에서 각 클럭별 전력 소비

3.5.2 태스크 모델

Task Model	
Symbol	Description
T	Set of n period tasks
T_i	i th periodic task in the T = [T_1, T_2,T_3, ⋯ , T_n]
P_i	period of task T_i
D_i	deadline of task T_i ($P_i = D_i$)
AET_i	Actual execution time of task T_i
$ACET_i$	average-case execution time of task T_i

3.5.3 시스템 모델

System Model	
Symbol	Description
Q_i	Queue for DP-DVS (T_i operated) Q_i = <($C_{i1}E_{i1}$), ($C_{z1}E_{z1}$), ⋯ , ($C_{ii}E_{ii}$)>
C_{ii}	Current time (T_i operated at F_i)
E_{ii}	Energyy consumption (T_i operated at F_i)
F_i	jth Frequency [Lowset <= F_i <=Highest]
A_i	Adaptive time of T_i
P_i	Policy of T_i [Performance, Energy, Optimal]

3.5.4 에너지 모델

Energy Model	
Symbol	Description
$E_{frequency}(F_i)$	Energy consumption at F_i
$E_{task}(T_i)$	Energy consumption of T_i $E_{task}(T_i) = E_{frequency}(F_i) \times AET_i + E_{frequency}(F_{idle}) \times \max(0, D_i - AET_i)$
$E_{total}(T)$	Energy consumption of pried task set (=T) $E_{total}(T) = E_{task}(T_1) + E_{task}(T_2) + \cdots + E_{task}(T_n)$ $E_{total}(T) = \sum E_{task}(T_i)$

3.5.5 DP-DVS

Algorithms Energy minimization for periodic task using dynamic programming

$Q_0 = <(0,0)>$
for i = 1 to n **do**
  if P_i == Performance or $ACET_i$ == NULL
    $Q'_{ij} = Q_{i-1} + (C_{ihighest}, E_{iHighest})$
  else if P_i == Energy
    $Q'_{ij} = Q_{i-1} + (C_{iLowest}, E_{ijLowest})$
  else if Pi == Optimal
    For all speeds j ∋ F
      $C_{ij} = C_{i-1} + ACET_i + A_i$
      $Q'_{ij} = Q_{i-1} + (C_i, E_{ij})$
    end for
  merge Q'_{ij} into a list Q'_i
  delete all state in Q'_i with $C_i > D_i$
end for
return the smallest state in Q_n

3.6 POSIX API

임베디드 시스템에 사용되는 하드웨어의 발전으로 임베디드 시스템의 적용분야가 확산되면서, 새로운 시스템의 개발은 새로운 응용 프로그램의 개발을 필요로 하고 있다. 이전에는 임베디드 시스템에 사용되는 하드웨어의 성능이 시스템의 중요한 요소가 되었지만 최근에는 응용 프로그램의 개발이 임베디드 시스템의 성공 여부를 결정짓는 중요한 요소가 되고 있다. 따라서 응용 계층에 임베디드 운영체제의 호환성 지원을 위한 API 표준화 문제가 대두하게 되었다.

따라서 Bio-K에서도 이러한 API 표준화를 위해서 POSIX 기반의 API를 지원하게 되었으며 다음과 같은 POSX 기반의 API를 지원하게 되었다.

3.6.1 쓰레드(pthread)

Bio-K은 다중 쓰레딩 방식으로 동작한다. 쓰레드 생성을 위한 pthread_create()와 쓰레드 종료를 위한 pthread_exit() 뿐만 아니라 POSIX에서 지정한 pthread API를 제공한다.

3.6.2 디바이스 드라이버(Device Driver)

Bio-K은 리눅스 및 유닉스 계열 운영체제와 유사한 디바이스 드라이버 등록 구조 및 인터페이스를 제공한다. 이로 인해 응용 프로그램의 작성 시 기존의 리눅스에서 사용했던 코드를 재사용 할 경우에 크게 수정 할 필요 없이 바로 사용 가능하다.

디바이스 드라이버 동작	설명
open	장치 초기화
close	장치 해제
read	장치로부터 데이터 입력
write	장치로부터 데이터 출력
ioctl	입출력 제어

[표 1] Bio-K 드바이스 드라이버 인터페이스

위의 디바이스 드라이버 인터페이스를 통해서 LED, FND, LCD, Ethernet 등 장치에 대한 제어가 가능하다.

3.6.3 ITC(Inter-Thread-Communiation) 및 기타

Bio-K 임베디드 커널의 ITC는 쓰레드와 쓰레드의 공동 작업을 위한 동기화 및 통신을 위하여 semaphore, Mutex, Message Queue를 지원하고 있다. 그 밖에 sleep, alarm, signal을 POSIX 기반 API를 지원하고 있다.

4. 구현

4.1 구현 내용

4.1.1 이식성

Bio Kernel은 다양한 MCU에 이식이 용의하도록 HAL(Hardwayre Abstract Layer)를 제공하며 응용 프로그램 작성에 용의하도록 POSIX 기반의 API를 제공한다. 본 장에서는 실제 KHIX 임베디드 시스템 운영체제의 확장 및 재구성을 확인하기 위해서 고성능의 PXA255, 저성능의 ATmega128에 이식을 하였다. 따라서 HAL부분에 해당하는 내용만 수정하면 쉽게 다른 시스템에 이식을 할 수 있다. HAL에 들어가는 내용으로는 Context Switching, Task Stack Init, Main clock & Timer 설정(Clock Tick 설정)등이 들어간다.

4.1.2 주기적 쓰레드(period thread)

바이오/의료 분야는 실시간 시스템의 한 종류로 주로 주기적인 쓰레드로 구성이 된다. 따라서 해당 시스템에서는 쉽게 주기적인 쓰레드를 생성 할 수 있도록 API를 제공해야 한다. Bio Kernel에서는 pthread의 attribute에 period를 변수를 추가하였다. pthread_create()함수를 이용해 생성 할 수 있으며, 별도의 period thread를 관리하는 리스트를 만들어 관리한다.

```
typedef struct __pthread_attr_s
{
  int __detachstate;
  int __schedpolicy;
  struct __sched_param __schedparam;
  int __inheritsched;
  int __scope;
  size_t __guardsize;
  int __stackaddr_set;
  void *__stackaddr;
  size_t __stacksize;
  int      __period;
} pthread_attr_t;
```

<표2> pthread attribute

```
pthread_t           tid1;
pthread_attr_t      attr1;
pthread_attr_init( &attr1 );
pthread_attr_setstack( &attr1, &Stack[0][STACK_SIZE-1], STACK_SIZE );
attr1.__schedparam.__sched_priority = PRIORITY_NORMAL;
attr1.__period      =       500;      // 500 ms periodic task
result = pthread_create( &tid1, &attr1, thread1, "Task1" );
```

<표3> period thread의 생성

4.1.3 안정성(stability)

바이오/의료 응용은 사람의 생명을 다루는 분야이기 때문에 안정성을 제공해야 한다. Bio Kernel은 두 가지 방법으로 안정성을 보장하고 있다. 첫 째는 Watchdog Timer를 이용한 안정성 보장이다. 이는 Watch thread를 제공해 전체적인 시스템의 안정성을 판단한다. Watch thread의 구현 방법은 ATmega128에서 제공하는 WDT 함수를 이용해 구현하였다. 두 번

째는 Watchcat thread이다. 이는 각각의 thread의 주기를 이용해 해당 thread의 이상 여부를 판단한 뒤 thread가 문제가 있을 경우에는 리스트에 등록 한 뒤 Watchcat thread가 수행 되면 해당 thread를 리셋 시키는 기능을 한다.

```
#if USE_WATCHDOGTHREAD
        pthread_attr_setstack( &attr1, &WatchStack[STACK_SIZE-1], STACK_SIZE );
        attr.__schedparam.__sched_priority = PRIORITY_HIGH;
        attr1.__period      = 1000 ;
        pthread_create( &tid1, &attr1, WatchThread, "WatchThread" );
        WDT_Enable() ;              // WDT init
#endif // USE_WATCHDOGTHREAD
```

<표4> Watchdog thread의 생성

```
#if USE_WATCHCATTHREAD
        pthread_attr_setstack( &attr2, &CatchStack[STACK_SIZE-1], STACK_SIZE );
        attr.__schedparam.__sched_priority = PRIORITY_HIGH;
        attr1.__period      = 1000 ;
        pthread_create( &tid2, &attr2, CatchThread, "CatchThread" );
#endif // USE_WATCHCATTHREAD
```

<표5> Watchcat thread의 생성

```
#if USE_CATCHTHREAD
        if ( g_pNextThread->Deadline < 0 )
            InsertToCatchList( g_pNextThread );
#endif // USE_CATCHTHREAD

void* Catchthread( void* pArg )
{
        for( int I = 0 ; I < CatchCount ; I++ )
        {
            ResetThread( CatchList[i] ) ;
        }
}
```

<표6> Watchcat List에 등록 및 Watchcat thread

4.1.4 POSIX API

<표6>의 소스코드는 LED를 깜박거리는 테스트용 응용프로그램의 소스코드 중 일부이며. 같은 소스코드를 PXA255, ATmega128에 이식하였다. 응용 프로그램의 소스코드에서 볼 수 있듯이 POSIX 기반의 API를 이용해 쓰레드를 생성하며, Semaphore, Mutex, Message Queue, sleet, usleet 및 기타 POSIX API 함수들의 사용이 가능하다.

또한 디바이스 드라이브의 인터페이스를 통해서 open(), close(), read(), write(), ioctl()의 함수들을 사용할 수 있다. 위의 응용 프로그램은 동일한 소스코드를 사용하여 고성능의 PXA255, 저성능의 ATmega128에서 정상 작동 하는 것을 확인하였다.

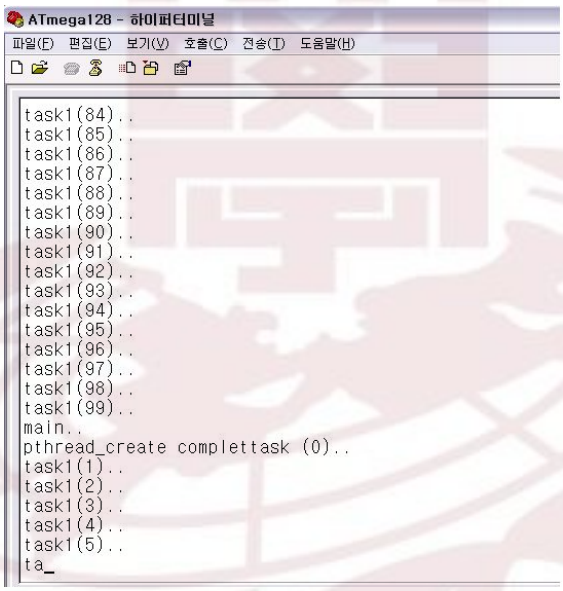
```
pthread_t          tid1, tid2, tid3 ;
pthread_attr_t     attr1, attr2, attr3 ;
sem_t sem ;
int result ;

sem_init( &sem, 1 );

result = pthread_create( &tid1, &attr1, thread1, "Task1" );

if( result < 0 )
{
    printf("(%d)pthread_create failed.₩n", result);
}
…
…
void* thread1( void* pArg )
{
    int num = 1;
    Int fd;
    int i;

    fd = open( "LED", 1 );
    If( fd < 0 )
    {
        printf("GreenLED open failed!₩n");
        printf("Thread will be terminated..");
        pthread_exit(1);
        return NULL;
    }
    while( 1 )
    {
        sem_wait( &sem );
        for( i = 0; i < 10; ++i )
        {
            num ^= 0x1;
            printf("(T1)Write = %d₩n", num);
            write( fd, (void*)&num, sizeof(num) );
            usleep(300);
        }
        sem_post( &sem );
        sleep(3);
    }
    close( fd );
}
```

<표7> POSIX API를 지원하는 Bio Kernel 응용 프로그램

5. 평가

5.1 주기적 쓰레드

주기적 쓰레드의 동작을 확인하기위해 Thread1은 주기를 측정하는 태스크 Thread2는 100ms 주기의 태스크이다. 수행 결과는 아래와 같이 100ms가 되는 시점에 thread2가 생성되는 모습을 볼 수 있었다.



[그림 5] Preodic thread의 주기 측정

5.2 안정성

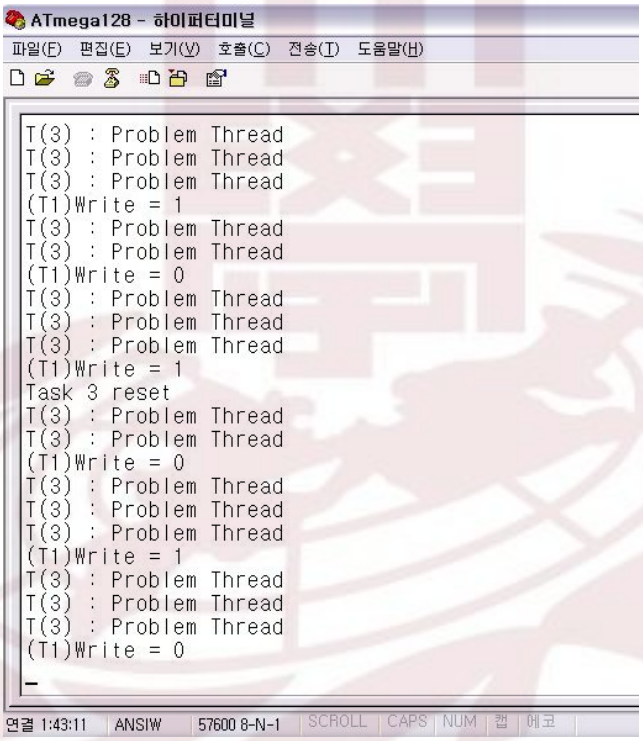
5.2.1 Watchdog Thread

안정성을 위한 기능 중 Watchdog Thread의 검증을 위해 전체 시스템 다운을 발생 시키도록 스택 오버 플로우를 포함하는 Thread 0 과 기타 LED를 컨트롤하는 Thread 1,2로 구성된 프로그램을 작성하였다. 수행 결과는 Thread 0의 스택 오버플로우가 발생 한뒤 Watchdog Thread의 리셋이 되지 않아 전체 시스템을 리셋 시키는 모습을 확인 할 수 있었다.

[그림 6] Watchdog Thread 수행 모습

5.2.2 Watchcat thread

안정성을 위한 기능 중 Watchcat Thread의 검증을 위해 특정 상황에 무한 루프에 빠지는 thread를 추가 하였다. Watchcat Thread는 이 thread가 문제가 발생하여 무한 루프에 빠졌을 때 해당 thread를 리셋 시켜주는 역할을 하는 것을 확인 할 수 있었다.



[그림 7] Watchcat thread 수행 모습

5.3 저 전력

ACET	Period	Policy
10ms	50ms	Optimal
15ms	100ms	Optimal
20ms	100ms	Optimal

<표8> POSIX API를 지원하는 Bio Kernel 응용 프로그램

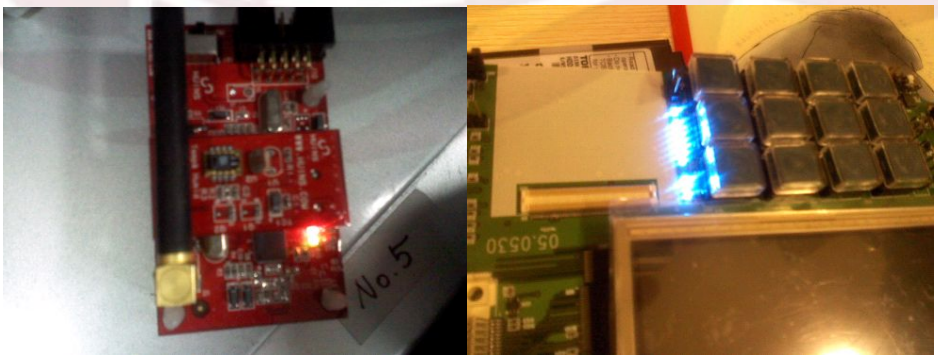
Bio-K의 저 전력 알고리즘인 DP-DVS 확인을 위해 표8과 같은 Thread의 형태로 다양한 태스크 셋을 만들어 시뮬레이션을 수행 하였다. 시뮬레이션 결과 태스크 셋에 따라 0 ~ 35%의 에너지 소비 절약 효과를 볼 수 있었다.

5.4 POSIX API

```
void* thread1( void* pArg )
{
   int num = 1;
   Int fd;
   int i;


   fd = open( "LED", 1 );
   sem_wait( &sem );
   for( i = 0; i < 10; ++i )
   {
      num ^= 0x1;
      printf("(T1)Write = %d\n", num);
      write( fd, &num, sizeof(num) );
   }
   sem_post( &sem );
   close( fd );
}
```

<표9> POSIX API를 지원하는 Bio Kernel 응용 프로그램

표9와 같은 thread를 만들어 PXA255와 ATmega128에서 수행을 시킨 결과 그림 8과 같이 LED가 동작하는 것을 확인 할 수 있었다.



[그림 8] PXA255와 ATmega128에서의LED 프로그램 수행 모습

6. 관련 연구

임베디드 시스템은 실시간성 특성 때문에 다중처리가 가능한 상용 실시간 운영체제가 주로 사용되었다. VxWorks[1], pSOS[2], Nucleus[3] 등 이러한 상용 운영체제는 실시간 처리 기능을 제공하는데 목적을 두고 신뢰성에 중점을 두며, 범용성이 아닌 주로 한 가지 특수한 목적에 최적화된 운영체제이다. 이러한 상용 운영체제는 단순히 바이너리 코드만을 제공한다. 따라서 기능의 수정이 어렵다는 단점을 가지고 있다.

이로 인해서 국내, 해외 대학 및 연구기관에서는 핵심 기술 확보를 위해 일반적인 실시간 운영체제와 센서와 같은 저성능의 시스템에서 동작하는 센서용 운영체제가 개발 되어왔다.

일반적인 실시간 운영체제인 µC/OS-II[4]는 교육용 실시간 운영체제로 간단한 소스코드로 실시간 운영체제의 기본적인 기능들을 제공하고 있다. 또한 국내 대학에서 개발한 Velos[5]는 실시간 스케줄링 지원 뿐만 아니라 TCP/IP, 동적 프로그램 로딩지원, Microwindows 시스템 및 KVM을 지원하고 있다. 센서용 운영체제로는 전통적으로 TinyOS[6]와 최근 국내에서 개발된 나노Qplus[7]가 대표적인 센서용 운영체제이다. TinyOS는 UC 버클리에서 제작된 이벤트 발생 중심의 상태 변화 방식을 채택한 센서 네트워크용 운영체제로 C와 유사한 NesC를 통해 센터 네트워크용 응용 프로그램을 작성 할 수 있다. 나노 Qplus 운영체제는 한국전자통신연구원(ETRI)에서 개발된 센서 네트워크용 운영체제이다. 특징은 제한된 메모리 사용을 최소화하기 위한 멀티 스레드 간의 스택 공유, 저전력 파워 소비 지원 등이 있다.

이러한 임베디드 운영체제는 독자적인 API를 가지고 있어서 사용자가 응용 프로그램을 작성할 경우 해당 API를 새로 익혀야 되는 단점이 있다. 따라서 최근의 운영체제 개발시에는 POSIX 기반의 API를 지원하는 추세로 가고 있다. [10][11]

또한 최근의 임베디드 시스템에 사용되는 하드웨어의 발전으로 인해 사

용할 수 있는 하드웨어 자원량이 늘어나 인터넷 연결, 멀티미디어 처리 등의 기능이 요구됨에 따라 운영체제도 점차 범용 운영체제가 많이 사용되고 있다.

이러한 범용 운영체제로는 임베디드 리눅스[8]와 WinCE[9]가 있다. 임베디드 리눅스는 오픈소스모델로 인해 수많은 개발자들을 통해 개발이 되어졌다. 임베디드 시스템에서 실시간성 지원을 위해 기존의 커널의 수정을 통한 실시간 지원, sub커널을 통한 실시간 지원하고 있다. 또한 POSIX 기반의 API를 지원하고 GNU도구 사용으로 인해 응용 프로그램 작성에 용의하며, 다양양 종류의 네트워킹, 파일 시스템, 프로토콜 지원하며 다양한 종류의 하드웨어를 지원하고 있다. 또한 여러 개발자들에 의해 개발이 되었기 때문에 기술성, 견고성, 보안 기술이 뛰어나다. [12]

MS에서는 WinCE 3.0를 발표하면서 기존의 WinCE 2.0에서 실시간성 지원이 떨어진다는 점을 보안하였다. Win CE는 기존의 PC환경에서의 사용자들에게 이미 익숙해진 윈도우 인터페이스를 임베디드 환경에서도 동일하게 사용할 수 있도록 제공하고 있다.

7. 결론 및 향후 연구

인구 고령화와 함께 생활양식 및 환경의 변화로 사람들의 건강에 관한 관심이 높아지고 있다. 언제, 어디서나 자신의 건상 상태를 알고 싶어하는 사람들의 욕구로 바이오/의료 분야가 더욱 더 발전하고 있다. 또한 정보 통신의 발전으로 Ubiquitous 기술을 이용한 u-Lifecare에 대한 관심이 높아지고 있다. u-Lifecare는 아픈 사람뿐만 아니라 건강한 사람들을 포함하는 서비스로, 시간과 공간의 제약 없이 원격 의료 기술을 활용한 건강 관리 서비스를 말한다. u-Lifecare가 구현된 환경에서는 환자가 의식하지 않은 상태에서 환자의 건강 상태를 실시간으로 모니터링하고 환자의 상태가 악화되면 바로 응급센터 및 병원 등에 환자 상태에 대한 정보가 전달되어 신속한 의료 서비스가 이루어진다.

이러한 바이오/의료분야는 임베디드 시스템의 한분야에 속한다. 임베디드 시스템은 특정 목적을 수행하기 위해 설계된 시스템으로 우리가 쉽게 접할 수 있는 PDA, 휴대폰, TV, MP3, 셋탑 박스, 디지털 카메라, 바이오/의료 응용 등 일상 생활과 매우 밀접한 관계를 가지고 있다.

바이오/의료 분야에 사용되는 프로그램은 이전에는 단순한 기능만을 수행하기 위해 펌웨어 수준의 프로그램이 사용되었지만 근래에는 다양한 기능과 복잡한 작업을 수행을 하게 되어 임베디드 시스템에 사용되는 실시간 운영체제를 사용하고 있다.

그러나 기존의 실시간 운영체제는 바이오/의료의 요구사항을 반영하지 못하고 있다. 따라서 본 논문에서는 바이오/의료의 요구사항을 분석 한 뒤 이를 반영하는 Bio-K 임베디드 커널을 설계하고 구현

하였다. Bio-K는 실시간 운영체제로 바이오/의료분야의 요구사항인 시간성, 안정성, 저 전력, 개발 편의성을 제공한다.

향후 연구로는 Bio-K의 안정성을 더 높이기 위해 Memory Protection 부분 및 네트워크를 위한 Zigbee Network Stack의 구현이 요구 된다.

8. 참고문헌

[1] VxWorks, http://www.windriver.com

[2] pSOS, "pSOS System Concepts", 1996.

[3] Nucleus, http://www.mentor.com

[4] µC/OS-II. http://www.ucos-ii.com

[5] Velos, http://www.mdstec.com

[6] TinyOS, http://www.tinyos.net

[7] Nano-Qplus, http://www.qplus.or.kr

[8] Embedded Linux, http://embedded-linux.org

[9] Win CE, http://www.microsoft.com

[10] 김선자, 김흥남, 김채규, "임베디드 운영체제 표준화 동향", 정보처리학회지 제9권 1호, 2002년 1월

[11] Institute for Elecrical and Electronic Engineers. In IEEE Std 1003.13-1998, IEEE Standard for Information Technology - Standardized Application Environment Profile-POSIX Realtime Application Support (AEP).

[12] 이형석, 정영준, “임베디드 운영체제 커널 기술 동향”, 전자통신동향분석 제21권 제1호 2006년 2월.

[13] 김정환, “전 세계 임베디드 S/W 시장 동향”, 정보통신진흥연구원, 주간 기술동향 통권 1107호, 2003년 8월.

[14] MicroC/OS-II 실시간 커널 제2판, 성원호 역, Jean J.Labrosse 저, 에이콘 출판.

[15] Intel PXA255 Processor web page, http://www.intel.com/design/pca/products/pxa255/techdocs.htm

[16] ATmega128 Datasheet, http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

[17] 박승민, “센서 네트워크 노드 플랫폼 및 운영체제 기술 동향”, 전자통신동향분석 제21권 제 1호

[18] 백용규, 조진성, “KHIX: 확장 및 재구성 가능한 임베디드 시스템 운영체제”, 한국정보과학회 하계종합학술대회, 2007.6.

[19] 박경환, "동영상 재생기를 위한 윈도우 기반 동적 전압조절 알고리즘," 석사학위논문, 경희대학교, 2007.2.

Abstract

Embedded Kernel for Bio/Medical Applications

Yong Gyu, Baek
Dept. of Computer Engineering
Graduate School of Kyung Hee University

With population becoming older in age and change of life style, environment, about health of the people the interest is coming to be high. the condition of oneself knows the Bio/Medical field more developed. also the interest is coming to be high about u-Lifecare because Ubiquitous techniques developed.

u-Lifecare is healthcare service which applies a remote medical technique without restriction of time and space for people that include sick person and health person.

The Bio/Medical field belongs in one field of embedded system.

Embedded system being in order to accomplish a specific goal such as PDA, cellular phone, TV, MP3, digital camera that can contact easily in daily life.

The program which is used in bio/medical field is made simple program such as firmware, but recent use real-time operating system for various the function and complex work.

But the real-time operating system of existing dose not reflect the requirement of bio/medical field. from the present paper after analyzing the requirement of bio/medical field, design the Bio-K that embedded kernel reflects bio/medical requirement. Bio-k, as real-time operating system, provides a periodic thread, stability, low-power, usability which is a requirement of bio/medical field.

Key words: Bio/Medical, Embedded System, Real-Time System, Real-Time Operating System, Low-Power System, Bio-K, Embedded Kernel, DP-DVS

감사의 글

연구실에서 학부 3학년 때부터 공부를 시작하여 어느덧 4년이라는 시간이 흘러 석사 졸업을 하게 되었습니다. 부족한 저를 지금까지 지도해 주신 조진성 교수님께 진심으로 감사를 드립니다. 학업뿐만 아니라 외적인 부분까지 항상 신경 써주신 교수님의 은혜를 평생 잊지 못할 것 같습니다. 또한 논문의 부족한 부분을 정성껏 심사해주신 한치근 교수님, 유인태 교수님, 홍충선 교수님, 허의남 교수님께도 감사드립니다.

4년간의 연구실 생활로 만은 추억을 남기고 가는 것 같습니다. 지금은 졸업을 하고 없지만 대학원 생활의 시작을 도와준 건백이형, 두경이형, 준하형, 그리고 같은 팀으로써 저를 이끌어 주던 경환이형, 상하형. 다른 팀이였지만 연구실 생활에 많은 도움을 주었던 재호형, 권택이형, 같이 연구실 생활은 못했지만 학부 시절 농구를 하면서 친해졌던 천환이형, 형관이형, 아쉽게 짧게 생활했던 창현이형, 학부시절 학부 연구생으로 같이 생활했던 정현이형, 헌준이형, 요한이형, 석사 시절 같이 학부 연구생으로 같이 생활한 권식이형, 승민이형, 윤성이형, 귀로형 준성이. 일룡이형, 연구실을 이끄느라 고생하는 대영이형, 충용이형과 연구실을 더욱 빛나게 할 경원이형, 의연이형, 학수형, 영선이형과 범석이에게 감사와 함께 아쉬움을 전합니다. 대학생활의 시작부터 함께 했던 경희대학교 컴퓨터공학과 동아리 HACKER 및 03학번 동기들에게도 감사의 인사를 드립니다.

마지막으로 어떤 말로도 표현할 수 없는 사랑으로 지금까지 키워주신 아버지, 어머니, 그리고 내 동생 주향, 우리 가족에게 한없는 감사를 드리며 부족하지만 이 논문을 바칩니다.

2009년 겨울 백용규 드림