μC/OS-II를 적용한 차량용 제어시스템의 설계에 관한 연구

A Study on Design of Vehicle Control System Based on μC/OS-II

송영호*, 이태양*, 박원용*, 문찬우*, 안현식*, 정구민**
Young-Ho Song*, Tae-Yang Lee*, Won-Yong Park*, Chan-Woo Moon*,
Hyun-Sik Ahn*, Gu-Min Jeong**

요 약

본 논문에서는 신뢰도와 내구성이 보장되어 차량용 바디 제어기에 많이 사용되는 16bit 마이크로 컨트롤러인 XC2287에 μC/OS-II를 이식하고 차량의 전자제어 액츄에이터로 많이 사용되는 모터로 시스템을 구성하여 실시간 운영체제 기반의 모터구동시스템을 구현한다. 구현한 실시간 모터구동시스템은 XC2287 마이크로 컨트롤러의 범용입출력 포트로부터 펄스폭 변조 신호를 출력하고 드라이버 회로를 통해 증폭된 신호가 DC모터에 인가된다. 사용자는 XC2287에 장착된 전위차계를 조작하여, 포트로부터 출력되는 펄스폭을 조절하고, 이를 통해 DC모터의 속도를 제어하고 출력된 신호를 모니터링 한다. 전위차계 조작에 의한 입력과정과 펄스폭 변조 신호 출력과정을 세마포어를 이용하여 동기화하는 실험을 통하여 μC/OS-II 이 올바르게 이식되었는지를 검증한다.

Abstract

In this paper, we study on design of vehicle control system which is based on μC/OS-II. We component a electric motor drive system for simulator because the most of vehicle part use electric motor for actuator. We use the XC2287 microcontroller which is often used vehicle body controller because XC2287 guarantee high confidence and durability in vehicle industry. The electric motor control system derive PWM from general I/O port in XC2287 microcontroller. The signal is supplied at electric motor after amplifying that using driver circuit. The user control duty of PWM signal through controlling potentiometer which is connected to XC2287. through that, the user control speed of electric motor. we synchronize both input process via controlling potentiometer and PWM output process using semaphore. we verify porting of μC/OS-II via experimentation.

Keywords : μC/OS-II, RTOS(Real Time Operating System), Microcontroller

I. 서 론

마이크로 컨트롤러를 사용하는 임베디드 시스템의 전 세계 시장수요는 2002년 기준 약 1000억 달러의 규모에서 매년 평균 20%씩 성장하여 2007년에는 약 1조9천5백억 원 규모에 이르렀다[1]. 임베디드 시스템의 시장성은 꾸준한 성장을 하고 있고, 임베디드 시스템은 프로세서, 운영체제, 프로토콜 등 전 분야와 관련이 있으며, 미국은 군사/과학용 임베디드 소프트웨어를 21세기 핵심 분야로 선정하여 매년 4천억 달러 이상을 연구개발에 투자를 하고 있다. 최근 자동차에서도 연비, 동특성, 안정성 및 안락성을 향상시키기 위하여 마이크로 컨트롤러(MCU: Micro Controller Unit)를 사용한 다양한 전자제어장치(ECU : Electronic Control Unit)들의 적용이 증가하고 있다. 그림 1 에서와 같이 2003년 메르세데스 벤츠의 S클래스의 경우 50개 이상의 전자제어장치들이 3개의 네트워크로 연결되어 150여 종류의 메시지를 통해 600여개의 신호를 송수신하고 있다[2].

이러한 자동차 전자제어장치의 비중이 증가하면서 소프트웨어의 양과 복잡성 또한 증가하고 있으며 그림 2 에서와 같이 2010년 까지 전장 분야 혁신의 80%가 소프트웨어 의 영역이 될 것으로 예측되고 있다[3], [4]. 또한 점차 늘어가는 시장의 요구와 짧은 개발주기로 인해 소프트웨어의 표준화가 중요한 사항으로 인식되기 시작 하였다.

안정적이고 효율적으로 전자제어장치를 관리하고 소프트웨어 개발자의 효율을 향상시키기 위한 방법 중의 하나가 표준화된 실시간 운영체제 시스템을 도입하는 것이다.

*국민대학교 전자공학과
**국민대학교 전자공학부 부교수/교신저자
투고 일자 : 2009. 4. 28 수정완료일자 : 2009. 7. 28
게재확정일자 : 2009. 7. 29
* 이 논문은 2008년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2008-331-D00456)

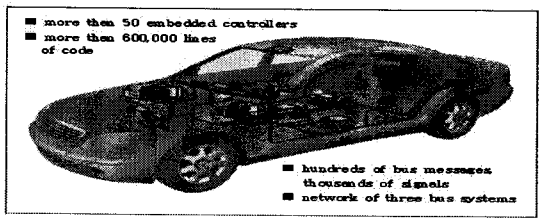


그림 1. 메르세데스 벤츠 S클래스의 전자화

Fig. 1. The complexity of the current Mercedes-Benz S-Class

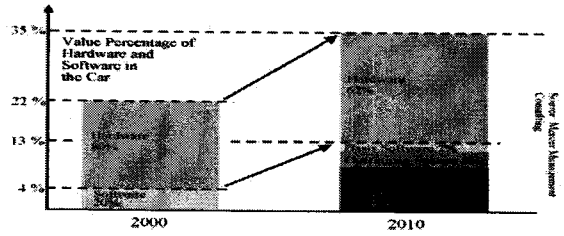


그림 2. 차량 내 소프트웨어의 비중

Fig. 2. Rise of importance of software in the car

실시간 운영체제를 도입함으로써 간단하고 체계적인 응용프로그램 인터페이스(API : Application Program Interface)를 운영체제로 부터 개발자에게 제공하여 소프트웨어 디자인이 용이해질 뿐만 아니라, 개발자가 어플리케이션 개발에 집중하여 소프트웨어 개발의 효율을 높일 수 있다. 또한 상이한 마이크로 컨트롤러 환경에서도 적용 가능한 표준화된 응용프로그램 인터페이스를 운영체제로 부터 제공받음으로써 향후 하드웨어 변경 시 이식이 용이하여 유지보수 및 개발시간 또한 단축된다.

본 논문에서는 현재 임베디드 소프트웨어 분야에서 많이 사용되는 μC/OS-II를 신뢰도와 내구성이 보장되어 차량용 바디 제어기에 많이 사용되는 XC2287 마이크로 컨트롤러에 이식하고 실시간 운영체제 기반의 모터구동시스템을 구현하고 올바르게 이식되었는지 검증한다.

II. 관련 연구

본 절에서는 실시간 운영체제인 μC/OS-II 와 실시간 운영체제에 대해서 설명한다.

2.1. RTOS

RTOS(Real Time Operation system)는 태스크(Task)를 시간에 따라 정확히 관리하고, 시스템 자원을 관리하며, 응용프로그램 개발을 위한 일관된 기반을 제공하는 일종의 프로그램이다[5]. RTOS의 주요 특성은 다음과 같다.

- 신뢰성(Reliability) : 사람이 관리하지 않고도 아주 오랜 기간 동안 혼자 동작해야 한다. 자동차에 탑재되는 대부분의 전자제어장치는 사람의 별도의 조작 없이 외부환경의 변화에 따라 동작한다. 따라서 사람의 관리 없이 오랜 기간 동작하므로 높은 신뢰성이 요구된다.

- 예측성(Predictability) : 시간 제약을 만족시키는 것은 시스템의 적합한 동작을 보장하기 위해 필수적인 요소다. 흔히 확정적(deterministic)이란 용어를 사용하며 이것은 커널이 제공하는 시스템 서비스가 정해진 시간 안에 완료되는 것을 뜻한다. 자동차에 탑재되는 다수의 전자제어장치가 브레이크, 엔진을 제어하는데 사용되므로, 시스템은 정확한 시간에 요구되는 동작을 수행하여야 한다.

- 간결성(Compactness) : 일반적으로 실시간 운영체제는 임베디드 시스템에 사용되며, 임베디드 시스템은 하드웨어의 크기와 비용에 크게 제약을 받는다. 이러한 임베디드 시스템에서 RTOS 또한 크기가 작으면서 효율적으로 동작해야 한다.

- 스케일러빌리티(Scalability) : RTOS는 응용 프로그램의 요구사항이 변하더라도 변함없이 잘 동작할 수 있어야 한다.

RTOS는 자원관리 알고리즘, 스케줄링 등을 관리하는 핵심적인 소프트웨어인 커널을 중심으로 응용프로그램에 따라 파일 시스템, 네트워크 프로토콜 스택 등을 조합하여 구성한다. 대부분의 RTOS는 공통적으로 다음 요소를 포함한다.

- 스케줄러 : 커널에 포함돼 있으며 언제 어떤 태스크를 실행할 지를 결정하는 알고리즘을 제공한다.

- 오브젝트 : 개발자가 실시간 임베디드 시스템용 응용프로그램을 만들 때 사용하는 특별한 구조체다.

- 서비스 : 커널이 오브젝트를 대상으로 수행하는 동작을 의미한다. 자원관리, 인터럽트 처리 등이 이에 속한다.

2.2. μC/OS-II

μC/OS-II는 1992년 Jean J. Labrosse 가 개발한 RTOS이다. 이미 안정성을 인정받은 μC/OS의 핵심부분을 그대로 사용하고, 미국 연방항공청(FAA : Federal Aviation Administration)이 항공전자공학 장치에 μC/OS-II를 적용할 수 있다는 승인을 함으로 μC/OS-II의 안정성을 인정받았다.[6] 이후 수백여 상용제품에 적용되어 안정성이 더욱 확보되었으며 안정성이 크게 요구되는 자동차용 임베디드 시스템의 Operating System으로 적절하다[7]. 또한 다양한 마이크로 컨트롤러에 이식이 가능하도록 대부분의 코드가 ANSI C언어로 작성되어 있으며, 일부 프로세서에 의존적인 최소한의 부분만 어셈블리 언어로 작성됐다.

μC/OS-II는 완전한 선점형 리얼타임 커널이다. 최대 64개의 태스크를 지원하지만, 실제 8개의 태스크를 시스템 용도로 예약해두었기 때문에 사용자는 최대 56개의 태스크를 사용할 수 있다. μC/OS-II의 태스크는 그림3과 같은 상태를 지니며, 스케줄링 과정을 통해 상태를 천이한다. μC/OS-II는 사용하는 코드공간과 데이터 공간의 낭비를 줄이기 위해 응용프로그램에서 필요한 기능만 이미지에 포함할 수 있으며, 커널 크기를 조절할 수 있다. 따라서 . 응용프로그램에 필요한 최소한의 서비스만 사용할 수 있으며, 다음과 같은 커널 서비스를 제공한다.

-세마포어, 이벤트 플래그, 상호배제 세마포어
-메시지 메일박스, 메시지 큐
-태스크 관리
-고정크기 메모리 블록 관리, 시간 관리

그림 3. μC/OS-II 태스크 상태
Fig. 3. μC/OS-II Task states

Ⅲ. μC/OS-II 이식

본 절에서는 μC/OS-II를 XC2287에 이식하는 과정에 대하여 설명한다. 본 논문에 사용된 XC2287은 차량의 바디 제어기, 게이트웨이, 도어, 냉난방 공조설비와 같은 차량 바리 응용에 주로 사용되는 16 bit 마이크로컨트롤러이다. 주요 특징은 다음과 같다.

표 1 . XC2287 주요 특징
Table. 1. Main features of XC2287

Flash Size	768 KBytes
PSRAM Size	64 KBytes
CCU6 Modules	4
Serial Channels	6
CAN Nodes	5
CAN Msg. Objects	128
ADC Channels	24

μC/OS-II는 그림 5와 같이 응용프로그램에 따라 커널의 크기를 조절하기 위한 코드와, 서비스를 제공하기 위한 코드, 그리고 프로세서의 레지스터를 제어하기 위한 프로세서 의존적인 코드로 구분되어 있다.

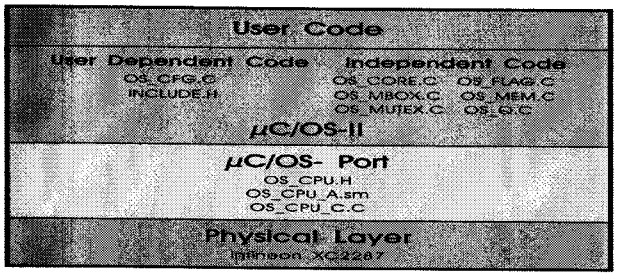


그림 5. μC/OS-II 소프트웨어 구조
Fig. 5. μC/OS-II Software structure

XC2287에서 μC/OS-II를 사용하기 위해서는 프로세서 의존적인 코드를 XC2287에 맞게 작성하고, 응용프로그램에 따라 커널을 변경하여야 한다. μC/OS-II에서는 프로세서 의존적인 코드는 OS_CPU라는 전치어를 붙여 구분하였다. 프로세서 의존적인 코드를 작성함에 있어서 가장 먼저 해야 할 것은 구현방법에 의존적인 코드를 정의 하는 것이다. 그 중 가장 먼저 해야 할 것은 컴파일러 의존적인 데이터 타입의 정의이다. 프로세서에 따라 워드 길이가 다르기 때문에 μC/OS-II에서는 이식성을 위한 데이터 타입을 선언해놓았다. 표2에 μC/OS-II에서 사용하는 데이터 타입을 표현하였다.

표 2 . μC/OS-II 데이터 타입
Table. 2. μC/OS-II data type

데이터 타입	구분
BOOLEAN	불 타입
INT8U	무부호 8비트 데이터
INT8S	부호 있는 8비트 데이터
INT16U	무부호 16비트 데이터
INT16S	부호 있는 16비트 데이터
INT32U	무부호 32비트 데이터
INT32S	부호 있는 32비트 데이터
FP32	단정도 부동소수 데이터
FP64	배정도 부동소수 데이터
OS_STK	스택요소의 크기(16bit)

다음으로 고려해야 할 사항은 임계영역에의 진입방법과 탈출방법의 선언이다. μC/OS-II에서는 3가지 방법을 정의하며, 프로세서의 동작속도와 개발자가 중요시하는 사항에 따라 적합한 방법이 다르다. 세 번째의 고려사항은 스택포인터의 방향이다. XC2287는 대부분의 마이크로 프로세서와 마찬가지로 메모리의 높은 번지에서 낮은 번지로 스택을 사용한다.

프로세서 의존적인 코드를 작성함에 있어서 다음 고려사항은 커널의 대표적인 기능인 스케줄링과 주기적 타이밍 소스를 작성하는 것이다. μC/OS-II는 우선순위 기반의 스케줄링을 사용하며, 수행중인 Task보다 더 높은 우선순위의 Task가 준비상태가 되면 OS_Task_SW()라는 매크로를 사용하여 Task를 전환한다.

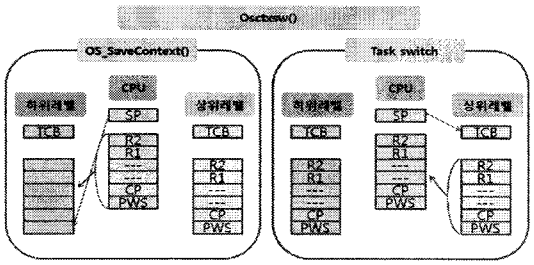


그림 6. 문맥전환
Fig. 6. Context switching

이 매크로는 OSCtxSw() 함수를 호출하는 기능을 가지고 있으며 CPU의 레지스터들을 직접 다뤄야하므로 어셈블리 언어로 구현한다. 그림 6처럼 μC/OS-II에서 준비상태의 Task 스택은 마치 인터럽트가 발생하여 레지스터들을 저장해놓은 모습을 취한다. 그러므로 OSCtxSw() 함수는 먼저 프로세서의 레지스터들을 수행 중이던 Task의 스택에 저장하고, 현재 Task의 스택 포인터를 현재 Task의 Task 컨트롤 블록에 저장하여 현재의 상태를 기억해야한다. 그리고 현재의 Task 블록을 실행할 Task의 컨트롤 블록으로 교체한 후 실행할 Task의 컨트롤 블록에서 스택포인터를 복구하고, 실행할 Task의 스택으로부터 프로세서의 레지스터를 복구한다. Tasking compiler는 인라인 어셈블리 기능을 지원하지만, 일반적인 방법을 따라 어셈블리 함수들은 별도의 어셈블리 파일로 관리하였다.

다음으로 주기적인 타이밍 소스를 제공하기 위해 OSTickISR()함수를 작성한다. XC2287는 T2~T6의 5개의 GPT(General Purpose Timer)를 제공하며 이 중의 하나를 타이밍 소스로 사용할 수 있다. 본 논문에서는 T3를 사용하였으며, 타임 틱은 초당 1회로 설정하였다. 마지막으로 Task에 관련된 함수를 작성하여야 한다. Task에 관련된 함수들로는 개발자가 확장할 수 있는 다양한 Hook 함수들과 Task의 스택을 초기화하는 함수가 존재한다. 이 중 Task의 스택을 초기화하는 함수는 반드시 필요하다. 왜냐하면 Task를 수행하는 과정과 전환하는 과정은 항상 동일하며, 이 과정은 앞에 자세히 설명하였듯이 수행 중이던 Task의 내용을 수행 중이던 Task의 스택에 저장하고 실행될 Task의 스택으로부터 이전의 상황을 복구하는 과정이기 때문이다. Task를 생성 할 때는 Task의 시작번지, Task가 사용하게 될 스택 공간의 포인터, Task의 우선순위만을 전달한다. 따라서 Task가 처음 수행될 때에는 마치 이전의 작업 과정이 있듯이 초기화해주는 과정이 필요하다.

Ⅳ. 실험 및 고찰

μC/OS-Ⅱ는 어플리케이션과 커널이 서로 구분 없이 결합되어 리소스를 공유하며 실행되기 때문에 XC2287에 μC/OS-Ⅱ 가 올바르게 이식이 되었는지 확인하기 위해서는 적절한 어플리케이션을 작성하여야한다. 본 논문에서는 실시간 모터구동시스템을 구현하였다.

본 논문에서 구현한 실시간 모터구동시스템은 XC2287 컨트롤러의 범용입출력 포트로부터 펄스폭 변조 신호를 출력한다. 사용자는 XC2287에 장착된 전위차계를 조작하여, 포트로부터 출력되는 펄스폭 변조 신호의 펄스폭을 조절한다. 컨트롤러에서 출력된 펄스폭 변조 신호는 드라이버를 통해 증폭되어 DC모터에 인가된다. DC모터는 인가되는 전류에 비례하여 속도가 증가되므로, 펄스폭을 제어하여 DC모터의 속도를 변화시키며 구동한다. 또한 DC모터의 속도에 따라서 LED의 점멸속도를 변경하여 가시적인 확인이 가능하도록 구현하였다. 이를 아래의 그림7에 나타내었다.

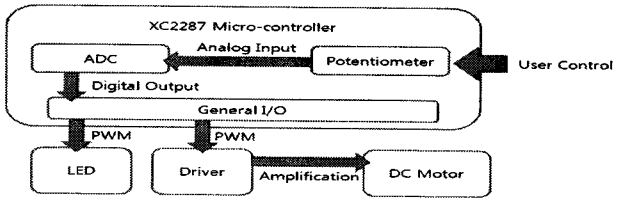


그림 7. 실시간 모터구동시스템의 블록다이어그램

Fig. 7. Block diagram of realtime motor drive system

실시간 모터구동시스템을 위해 3개의 태스크를 생성하였다. 먼저 전위차계에 의한 아날로그 값을 ADC(Analog / Digital Converter)를 통해 디지털 값으로 변조하는 ADC_Input 태스크를 생성하였다. 그리고 디지털로 변조된 값에 따른 펄스를 발생시키는 PWM_Generate 태스크와 DC모터의 동작속도에 따라 LED의 점멸속도를 변환시켜주는 LED_Toggle 태스크를 생성하였다. 태스크 간의 동기화는 세마포어를 이용하였다.

표 3 . 태스크 구성

Table. 3. Task structure

태스크 번호	태스크에 할당된 명령	우선순위
TASK1	PWM_Generate	7
TASK2	ADC_Input	8
TASK3	LED_Toggle	9

주기적으로 Task2가 전위차계로부터의 입력을 아날로그-디지털 변환회로를 이용하여 변환한다. 변환된 값은 ADC_Result 라는 변수를 할당하여 저장하였다. 각 태스크의 우선순위는 표3과 같다. Task1과 Task2가 ADC_Result 라는 자원을 공유하기 때문에 이를 고려하여 우선순위를 할당하였다. Task2보다 Task1의 우선순위가 높을 경우, 펄스폭 변조 신호를 구현하기 전에 Task1이 실행되어 펄스폭 변조 신호가 올바르게 구현되지 않는 경우가 발생하기 때문이다. Task1은 DC모터의 구동속도에 직접적인 영향을 미치기 때문에 우선순위를 고려한 정확한 제어가 필요하지만 Task3는 DC모터의 속도를 가시적으로 판단하는데 목적이 있으며, 정밀한 제어가 요구되지 않아 가장 낮은 우선순위를 부여하였다.

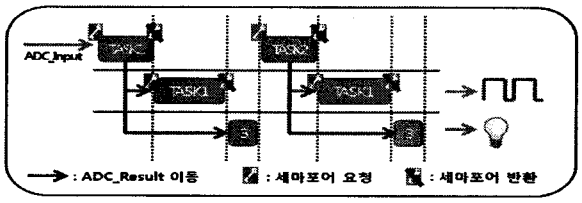


그림 8. 세마포어를 이용한 동기화

Fig. 8. Synchronization using semaphore

그림8은 공유자원의 흐름을 나타낸다. 최상위 우선순위를 가지는 Task2가 실행되며 아나로그 량을 디지털 량으로 변환한다. Task2는 세마포어를 소유하고 있으므로, Task1은 아나로그 량이 디지털 량으로 변환되기를 기다린다. 아나로그 량이 디지털 량으로 변환되면 Task1이 세마포어를 획득하게 된다. Task2는 Task1이 펄스폭 변조 신호를 발생시키기 전에는 공유자원인 ADC_Result에 접근할 수 없으므로 펄스폭 변조 신호 발생을 기다리게 되어 동기화를 이룬다. 그림9는 전위차계를 조작하여 펄스폭을 조정하는 것을 나타내며, μC/OS-Ⅱ를 올바르게 이식하였음을 확인할 수 있다. 그림10에 전체 시스템의 모습을 나타내었다.

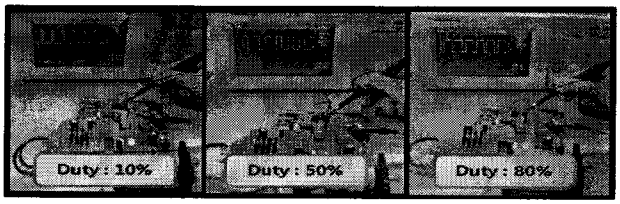


그림 9. 펄스폭 변조 신호 펄스 폭 제어

Fig. 9. PWM duty control

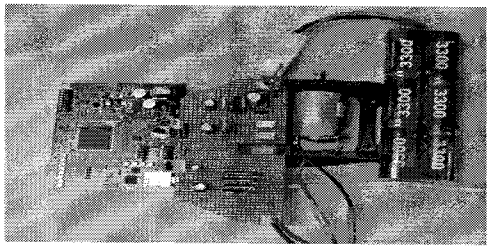


그림 10. Cμ/OS-II 기반의 실시간 모터구동시스템
Fig. 10. Motor drive system based on μC/OS-II

V. 결 론

본 논문에서는 IT융합이라는 IT업계의 패러다임에 발맞추어 범용 RTOS로 마이크로컨트롤러에 많이 사용되는 μC/OS-II를 차량용 16bit XC2287 마이크로컨트롤러에 이식하고 μC/OS-II 기반의 실시간 모터구동시스템을 구현하였다. 모터구동시스템에 μC/OS-II를 접목함으로서 하드웨어의 레지스터에 대한 간접제어가 가능하여 개발의 편의성을 증진시킬 수 있다. 추후 연구과제로는 μC/OS-II 기반의 실시간 모터제어시스템을 설계하고 제어 시스템의 성능을 분석할 예정이다.

참 고 문 헌

[1] 정보통신연구진흥원, "임베디드 S/W 교안", 정보통신연구진흥원 학술기사, 2003. 8L. R.

[2] Klaus Grimm, "Software Technology in an Automotive Company - Major Challenges", Proceedings of the 25th International Conference on Software Engineering, pp.498-503, IEEE Computer Society, 2003

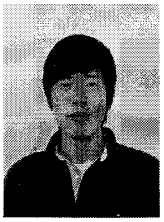
[3] Hardung Bernd, Kolzow Thorsten and Kruger Adnreas, "Reuse of Software in Distributed Embedded Automotive Systems", Proceedings of the 4th ACM International Conference on Embedded Software, ACM Press, pp.203~210, 2004

[4] Mercer Management Consulting and Hypovereinsbank. studies, Automobil technologie 2010. München, 2001.

[5] Qing Li, Caroline Yao, Real-Time Concept for Embedded Systems, CMP Books, 2004

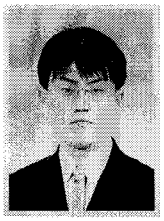
[6] Jean J. Labrosse, MicroC/OS-II The Real-Time Kernel, CMP Books, 2002

[7] 박지강, 서한석, 김정국 "실시간 객체 TMO를 위한 Micro C / OS - II 실시간 스케줄러의 설계 및 구현" 한국컴퓨터종합학술대회 2005 논문집(A), pp.835-837



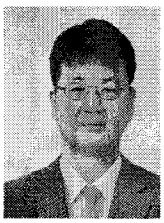
이 태 양(Tae-Yang Lee)
2009년 2월 선문대 전자공학과(공학사)
2009년 3월~ 현재 국민대학교 대학원 전자공학과 석사과정

※주관심분야 : 임베디드 시스템, 차량 전자제어



박 원 용(Won-Yong Park)
2008년 2월 국민대학교 전자공학과(공학사)
2008년 3월~ 현재 국민대학교 대학원 전자공학과 석사과정

※주관심분야 : 차량 전자제어, 실시간 운영체제



문 찬 우(Chan-Woo Moon)
1989년 2월 서울대 전기공학과(공학사)
1991년 2월 서울대 전기공학과(공학석사)
2001년 2월 서울대 전자공학과(공학박사)
2006년 8월 ~ 국민대학교 전자공학과 조교수

※주관심분야 : 모바일 로봇 네비게이션, 정밀제어, 모터제어

안 현 식(Hyun-Sik Ahn)
1982년 2월 서울대 전기공학과(공학사)
1984년 2월 서울대 전자공학과(공학석사)
1992년 2월 서울대 전자공학과(공학박사)
1993년 3월 ~ 현재 국민대학교 전자공학과 교수

※주관심분야 : 임베디드 시스템, 차량 전자제어

정 구 민(Gu-Min Jeong)
1995년 2월 서울대 전기공학과(공학사)
1997년 2월 서울대 전자공학과(공학석사)
2001년 2월 서울대 전자공학과(공학박사)
2005년 3월 ~ 2009년 2월 국민대학교 전자공학과 조교수
2009년 3월 ~ 현재 국민대학교 전자공학과 부교수

※주관심분야 : 임베디드 시스템, 휴대폰 어플리케이션, 차량 전자제어