논문 2012-07-19

유비쿼터스 컴퓨팅을 위한 센서 디바이스 Plug & Play

(Sensor Device Plug & Play for Ubiquitous Computing)

박 정 선, 은 성 배*, 윤 현 주
(Jung-Sun Park, Seongbae Eun, Hyeon-Ju Yoon)

Abstract : When mounting the sensor device in the way of Plug&Play, sensor device drivers need to be loaded and linked dynamically. Since a sensor node platform is based on small 8 bit MCU, dynamic loading and linking technique used in Windows and Linux can not be applied. In this paper, we present how to link and load dynamically sensor device drivers for sensor device Plug&Play. We implement a prototype and evaluate it to make sure that there is no performance degradation like sensor device driver connection speed and memory usage. Connection speed overhead increases to 0.2ms. Memory usage overhead increases to hundreds byte. It shows that there is no heavy influence in running the actual program.

Keywords : Sensor Device Plug&Play, Sensor Device Driver, Dynamic Loading and Linking, Sensor Node Platform

Ⅰ. 서 론

유비쿼터스 컴퓨팅은 센서와 RFID를 현장에 설치, 상황인지를 통하여 서비스를 제공한다. 특히 유비쿼터스 센서 네트워크(USN)와 유비쿼터스 지능형 로봇(URC)에서는 다양한 주변 상황을 올바로 인식하기 위해 센서의 역할이 중요하다. 센서를 통해 물리 공간의 빛, 소리, 온도, 움직임 등 같은 물리적 데이터를 감지하고 측정하여 무선 네트워크를 통해 사용자에게 전달한다 [1].

센서는 감지대상, 감지수단, 재료/구조, 응용 분야 등에 따라 종류가 다양하다. 동일한 응용 분야라도 응용 시스템의 용도, 변화될 값의 범위, 동작 환경에 따라 다른 센서가 사용된다. 또한 측정한 값을 전송하기 위한 인터페이스도 ADC, 주파수 입력, 인터럽트, SPI, I2C, Serial 등으로 다양하다. 예를 들어 온도 센서는 측정값의 범위에 따라 상온에서 동작하는 반도체 온도 센서, 수온 측정용 막대형 온도 센서, 1천도 이상의 고온 측정용 온도 센서 등으로 다양하다. 센서의 출력 형식도 전약, 전류, 주파수 등으로 다양하며, 증폭이 필요한 센서, ADC (Analog Digital Converter)에 바로 붙일 수 있는 센서 등으로 다양하다.

이처럼, 센서의 종류와 인터페이스가 다양하기 때문에 유비쿼터스 컴퓨팅에서 센서를 사용한 응용 개발시 응용개발 기간 및 비용이 증가한다. 또한 센서 교체가 빈번하므로 Hardware Abstract Layer (HAL)의 변경, 센서 디바이스 드라이버의 교체 등의 작업이 요구되고 이에 따른 유지보수 비용이 증가한다.

그 동안 센서 사용에 따른 센서노드 운영체제가 개발되고 상용화 되었다. 하지만, NanoQ+ [2], Tiny-OS [3], SOS [4] 등 기존 센서노드 운영체제에서는 센서 인터페이스의 다양성을 고려한 표준화된 API를 제공하지 못하며, 센서의 자동인식도 지원하지 않는다.

유비쿼터스 컴퓨팅 발전 및 산업 적용의 주요 걸림돌을 해결하기 위한 방안으로, 센서디바이스 Plug&Play를 지원하는 센서노드 플랫폼이 연구되었다 [5-7]. 센서노드 운영체제에서 응용개발자에게 표준화된 API [5] 및 센서 디바이스 매니저 기능 [6]을 제공하였고, Plug&Play가 지원 되는 센서 인터페이스 플랫폼 [7]에 관한 연구가 진행되었다.

센서 디바이스 Plug&Play가 지원되기 위해서는

* 교신저자(Corresponding Author)
논문접수 : 2012. 02. 16., 수정일 : 2012. 02. 29.,
채택확정 : 2012. 03. 10.
박정선, 은성배 : 한남대학교 정보통신공학과
윤현주 : 금오공과대학교 컴퓨터공학부
ⓒ J. IEMEK 2012 Jun.: 7(3) 151-156
ISSN : 1975-5066

센서 디바이스 드라이버를 센서노드 플랫폼에서 장착하고 있어야 한다. 센서노드 플랫폼이 센서 디바이스 드라이버를 장착하고 있지 않을 경우 센서모듈이나 원격 센서 디바이스 드라이버 관리 서버로부터 다운로드 한다. 다운로드된 센서 디바이스 드라이버를 센서노드 플랫폼에 동적으로 적재 및 연결이 이루어져야 하나 기존 연구 [5-7]에서는 포함되지 않았다. 또한 소형의 8bit MCU기반의 센서노드 플랫폼에서는 Windows나 Linux에서 사용하는 동적 적재 및 연결 방법을 적용할 수 없다.

본 논문에서는 센서 디바이스 Plug&Play를 위한 센서 디바이스 드라이버 동적 적재 및 연결하는 방법을 제시한다. 또한, 센서 디바이스 드라이버 연결속도 및 메모리 사용량에 대한 성능평가를 통해 제시한 동적 적재 및 연결방법에 문제가 없음을 보인다.

II. 관련 연구

센서노드 운영체제에서 응용개발자에게 표준화된 API을 제공하고, 센서의 자동인식을 고려한 연구가 진행되었다.

센서투명성을 지원하는 센서노드 운영체제 구조 [5]는 센서장착을 용이하게 하는 표준화된 센서 HW 인터페이스를 정의하였다. 추상화된 센서접근 API 제공 및 센서 디바이스 드라이버를 작성할 때 활용될 HAL라이브러리를 정의 하였다. 그 결과 센서제작자는 HAL을 이용하여 센서 디바이스 드라이버를 작성하고 응용프로그래머는 API 이용하여 응용개발이 용이하도록 하였다.

센서 투명성을 지원하는 센서 디바이스 매니저 [6]는 ETRI에서 개발한 NanoQ+ 에서 센서 투명성을 지원하는 센서 디바이스 매니저 기능을 구현하였다. 이러한 센서 투명성을 지원하는 OS 구조가 다양한 센서에 대한 통일된 접근 방식으로 응용프로그래머의 개발 부담을 줄였다.

센서 Plug&Play를 지원하는 센서노드 플랫폼 [7]은 센서 Plug&Play 기능을 제공하기 위해 센서모듈과 인터페이스 규격, 센서 식별 아이디 규칙 및 전송방법을 설계하였다. 그 결과 센서 네트워크를 구성할 때 빠른 시스템의 설치가 가능하게 하고, 센서의 수리 및 교체로 인한 시스템의 다운타임을 감소시킨다.

위와 같은 연구에서는 센서 디바이스 드라이버를 센서노드 플랫폼에서 장착하고 있어야만 하는 단점이 있다. 센서 디바이스 드라이버를 센서모듈이나 원격에 있는 센서 디바이스 드라이버 관리서버로부터 다운로드하여 처리하는 과정이 빠져있다.

III. 설 계

1. 전체 구조도

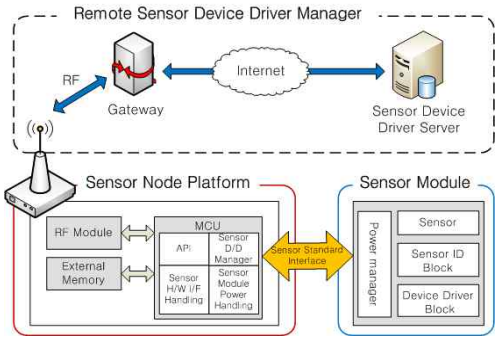

그림 1. 전체 시스템 구조

Fig. 1. Overall system architecture

그림 1과 같이 센서 디바이스 Plug & Play를 지원하는 전체 시스템은 크게 원격 센서 디바이스 드라이버 관리와 센서 노드에서의 처리로 나누어진다.

원격 센서 디바이스 드라이버 관리는 센서 디바이스 드라이버 검색 프로토콜과 디렉토리 서비스 프로토콜, 새로운 디바이스 드라이버 등록 프로토콜 등으로 이루어진다. 이 기능들은 센서 드라이버를 검색하거나 센서 드라이버를 보유하는 서버의 접근 그리고 새로운 디바이스 드라이버를 등록하는 절차로 이루어진다.

센서 노드에서의 처리는 센서 식별 정보 수신, 센서 디바이스 드라이버 비교 프로토콜, 센서 디바이스 드라이버 요청 절차 등으로 이루어진다. 이 기능들은 센서노드가 초기화 되었을 때 센서 식별 정보를 수신하고, 장착하고 있는 센서 디바이스 드라이버와 비교하여 센서 디바이스 드라이버가 없을 때 센서 모듈이나 원격 센서 디바이스 드라이버 관리 서버로부터 새로운 센서 디바이스 드라이버를 다운로드 한다. 다운로드된 센서 디바이스 드라이버를 동적 적재 및 연결과정을 거쳐 센서를 초기화하고, 센서로부터 센서데이터를 획득한다.

2. 센서 노드 플랫폼 구조

그림 2는 센서노드 플랫폼에 센서를 장착하였을

때 센서모듈을 인식하고 센서 디바이스 드라이버를 다운로드하는 과정을 보여준다.

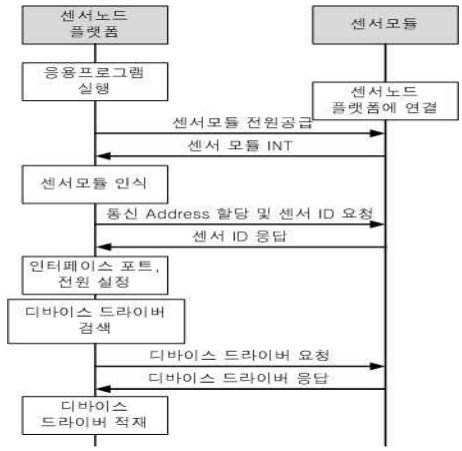

그림 2. 센서 모듈 인식에 따른 절차
Fig. 2. Procedure according to sensor module recognition

센서노드 플랫폼의 응용프로그램 수행 중 센서 모듈이 연결되면 센서의 동작을 위해 센서 식별 정보 수신, 센서 디바이스 드라이버 검색, 센서 디바이스 드라이버 수신 및 동적 적재가 진행된다. 센서 식별 정보를 수신하고, 센서 디바이스 드라이버를 센서 모듈로부터 얻어와 동작한다.

그림 3과 같이 응용개발자는 표준화된 API를 기반으로 응용을 개발한다. 이때 센서 데이터 처리는 센서 디바이스 드라이버 공급자가 작성한 디바이스 드라이버가 담당한다. 센서 디바이스 드라이버 공급자는 표준화된 센서 HW인터페이스와 HAL 라이브러리를 기반으로 센서 디바이스 드라이버를 작성한다. HW를 직접 접근하지 않고 HAL라이브러리를 통하여 접근하는 센서 투명성 [5]을 기반으로 한다.

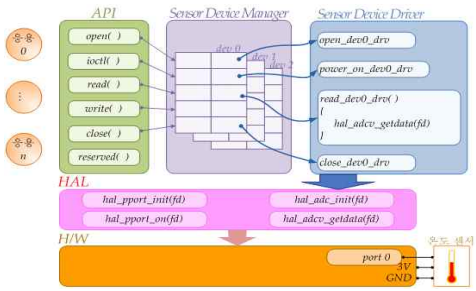

그림 3. 센서노드 플랫폼 기반 구조도
Fig. 3. Sensor node platform based structure

3. 센서 디바이스 드라이버 매니저

그림 4는 센서 디바이스 드라이버 매니저 구조를 보여준다. 센서노드 운영체제 내에서 응용 API 호출로부터 하드웨어 인터페이스와 센서 디바이스 드라이버를 연결한다.

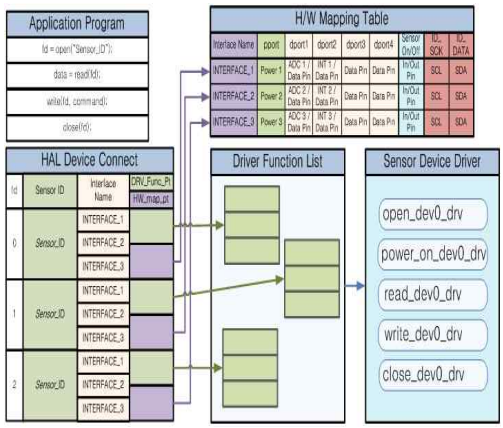

그림 4. 센서 디바이스 드라이버 매니저 구조
Fig. 4. Sensor device driver manager structure

센서노드 플랫폼에 센서모듈이 장착될 때 운영체제가 가지고 있는 센서 디바이스 드라이버가 Driver Function List에 등록된다. 운영체제 내에 해당 센서 디바이스 드라이버를 보유하지 않을 경우 센서모듈이나 원격 센서 디바이스 드라이버 관리 서버로부터 받아온 센서 디바이스 드라이버가 등록된다. 센서노드 플랫폼에 따라 인터페이스가 달라지므로 H/W Mapping Table로 실제 센서 디바이스가 하드웨어 어떤 포트에 연결되었는지 지정한다.

4. 센서 H/W 인터페이스

센서마다 커넥터의 구조나 연결방식이 달라지므로 센서의 인터페이스는 센서 마다 다르다. 그림 5와 같이 통합 인터페이스를 제공하여 응용 개발자의 개발 부담을 줄여준다.

그림 5와 같이 통합 인터페이스는 9핀을 기본으로 사용한다. 센서모듈에 인가되는 전원 VCC와 GND를 가지고 있고, 센서 인터페이스 핀은 인터페이스 종류 중 가장 많은 핀을 사용하는 SPI통신을 지원하기 위해 4핀을 할당했다. 센서 정보를 전송받기 위해 I2C통신방식을 사용하여 SCK, DATA핀을 할당했다. Connection핀은 센서노드 플랫폼에 센서모듈이 장착 및 탈착 여부를 알려주기 위해 사용한다.

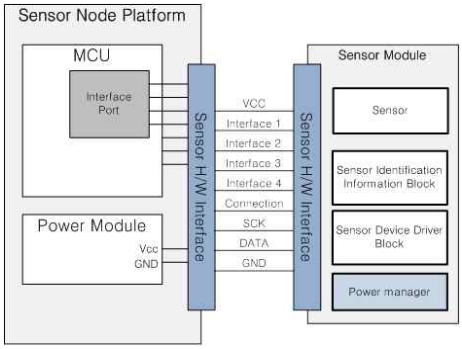

그림 5. 센서 H/W 인터페이스 구조
Fig. 5. Sensor H/W interface structure

5. 센서 디바이스 드라이버 동적 적재

센서모듈이나 원격 센서 디바이스 드라이버 관리서버로부터 센서 디바이스 드라이버를 수신하여 센서노드 플랫폼 플래시 메모리에 적재해야 한다.

그림 6은 플래시 메모리 구조를 나타내었다. 센서모듈에 보유하고 있는 센서 디바이스 드라이버는 센서모듈 응용프로그램과 별도로 컴파일된 이미지이며, 센서노드 플랫폼에 전송된다. 운영체제에서 센서 디바이스 드라이버를 플래시 메모리 적재하기 위해서는 두 가지 문제가 발생된다.

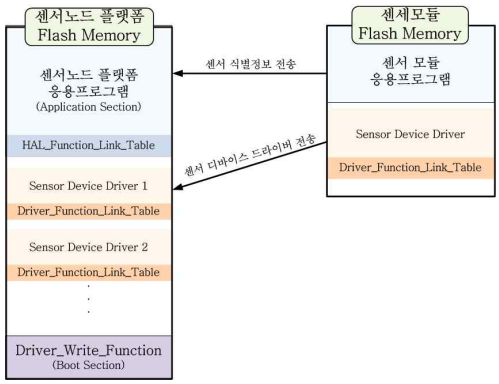

그림 6. 플래시 메모리 구조
Fig. 6. Flash memory structure

첫째, 응용프로그램 동작 중에는 메모리에 쓰기 권한이 없다. 메모리에 쓰기 위해서는 부트로더를 올려야 한다. 부트로더를 올리고 운영체제를 올리는 것 자체가 응용개발자에게 부담된다. 그리고 센서모듈을 인식하고 메모리에 디바이스 드라이버를 적재하기 위해서는 리셋과정을 거쳐야 한다. 이러한 문제를 없애기 위해 센서노드 운영체제 내에 부트로더 기능을 갖는 함수를 Boot Section에 할당하여 운영체제에서 센서 디바이스 드라이버를 동적으로 적재한다.

둘째, 센서 디바이스 드라이버는 컴파일된 물리주소로 적재 되어야 동작을 할 수 있다. 컴파일된 물리 주소가 아닌 다른 물리 주소에 적재될 경우 센서 디바이스 드라이버는 동작하지 않는다. 운영체제는 메모리에 적재하기 하기 전에 센서 디바이스 드라이버 내에 함수끼리의 호출 주소를 실제 적재되는 메모리 물리 주소에 맞게 변경한다. 수신되는 기계어코드 중 call명령을 찾아 함수의 물리 주소를 수정한다. 그 결과 센서 디바이스 드라이버의 컴파일된 물리 주소와 관계없이 운영체제의 판단에 따라 메모리에 동적 적재하여 동작하도록 하였다.

6. 센서 디바이스 드라이버 동적 연결

운영체제와 센서디바이스 드라이버는 별도로 컴파일된다. 메모리에 적재된 센서 디바이스 드라이버와 운영체제간의 동적 연결하는 과정이 필요하다.

메모리에 적재한 센서 디바이스 드라이버 함수를 접근하기 위해서는 그림 7와 같은 구조를 가져야 한다.

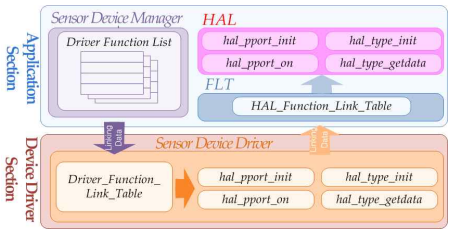

그림 7. 디바이스 드라이버 동적 연결 구조
Fig. 7. Device driver dynamic linking structure

기존 응용프로그램과 적재된 디바이스 드라이버는 하나의 프로그램으로 컴파일이 되지 않았기 때문에 서로간의 주소를 모르고 있다. 그래서 다음과 같은 함수의 주소를 절대주소로 지정하여 운영체제의 HAL라이브러리 함수와 센서 디바이스 드라이버를 접근한다.

1. Driver_Function_Link_Table( )
   : Sensor Device Manager → Sensor Device Driver
2. HAL_Function_Link_Table( )
   : Sensor Device Driver → HAL

운영체제내의 HAL라이브러리 함수나 센서 디바

이스 드라이버 함수를 호출하기 위해 Linking_Data를 이용하여 함수의 번호를 알려준다. 또한, 함수의 매개변수 및 반환 값도 Linking_Data를 이용한다.

Linking_Data는 운영체제와 센서 디바이스 드라이버간의 공용으로 쓰는 전역변수이다. 그림 8에서처럼 SRAM 가장 앞부분에 할당하여 사용한다.

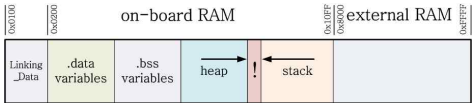

그림 8. SRAM 구조
Fig. 8. SRAM structure

Ⅳ. 성능평가

본 장에서는 센서 디바이스 드라이버를 센서노드 플랫폼에서 장착하고 있을 경우와 센서모듈로부터 다운로드된 센서 디바이스 드라이버를 사용할 경우 처리 속도 및 플래시 메모리 사용량에 대해 비교하였다.

1. 처리 속도 비교

처리 속도 비교는 TSL2560센서(조도센서)를 초기화하고 센서로부터 값을 수신하는 과정까지 Instruction 수를 이용하여 수행시간을 비교하였다.

표 1. 처리 속도 비교
Table 1. Processing speed comparison

	Number of Instruction	Run Time
내장형 센서 디바이스 드라이버	7,884	0.98ms
동적 센서 디바이스 드라이버	9,426	1.18ms

표 1과 같이 Instruction 수가 증가하여 Overhead가 발생하였지만, 수행시간이 0.2ms 차이므로 처리속도에 거의 영향을 주지 않는다.

2. 플래시 메모리 사용량 비교

센서노드 플랫폼에서 메모리 사용량을 확인하였다. 기존에 센서노드 플랫폼에 저장된 센서 디바이스 드라이버와 동적으로 다운로드된 센서 디바이스 드라이버의 플래시 메모리 사이즈를 비교하였다.

표 2. 센서 디바이스 드라이버의 메모리 사용량
Table 2. Memory usage of sensor device driver

	TSL2560 (조도센서)	SHT11 (온습도센서)
내장형 센서 디바이스 드라이버	1.39KB	1.96KB
동적 센서 디바이스 드라이버	1.41KB	2.10KB
증가율	+1.43%	+7.14%

표 2와 같이 기존 내장형 센서 디바이스 드라이버에 비하여 센서마다 차이가 있지만 10%내의 Overhead가 발생하였다. 그러나 센서 디바이스 드라이버 크기 자체가 수Kbyte내 이기 때문에 메모리 사용량에는 문제가 되지 않는다.

Ⅴ. 결 론

본 논문에서는 유비쿼터스 컴퓨팅을 위한 센서 디바이스 Plug&Play 방법을 제시하였다. 센서 디바이스 드라이버를 센서모듈이나 센서 디바이스 드라이버 관리서버로부터 다운로드할 경우 디바이스 드라이버를 메모리에 동적 적재하는 방법을 제시하였고, 기존 센서노드 운영체제와 동적 연결하는 방법을 제시하였다. 제시한 동적 적재 및 연결 방법에 Overhead가 발생하였지만, 처리속도는 0.2ms 증가하였고 플래시 메모리 사용량은 수백byte내로 증가하여 실제 프로그램 구동에 미치는 영향이 매주 작음을 확인하였다. 유비쿼터스 컴퓨팅에서 센서를 이용한 응용 개발시 빠른 시스템의 설치가 가능하게 하고, 센서의 수리 및 교체를 할 때에 시스템의 다운타임을 감소시킨다. 개발기간 및 시스템 구축에 필요한 비용, 유지보수 비용 등을 절감하여 경쟁력 있는 가격에 보다 빠르게 출시하는 결과를 가지고 오는 효과를 기대할 수 있다.

향후 연구방향은 본 논문에서 사용한 ATmega가 아닌 MSP나 8051같은 다른 8bit MCU에서 본 구조를 구현해야 한다. 또한 하나의 센서 디바이스 드라이버가 다른 8bit MCU와 호환되도록 연구가 진행되어야 한다.

참고문헌

[1] 은성배, 소선섭, 채의근, “유비쿼터스 센서네트워크 서비스 분류 기법 및 상용화 이슈,” 대한임베디드공학회 논문지, Vol. 2, No. 3, pp.202-208, 2007.

[2] S. Park, J. Kim, K. Lee, K. Shin, D. Kim, "Embedded Sensor Network Operating System," Proceedings on 9th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, 2006.

[3] P. Levis, S. Madden, D. Gay, J. Poastre, R. Szewczyk, A. Woo, E. Brewer, D. Culler, "Theremergence of networking abstrzctions and techniques in tinyos,“ Proceedings on the First USENIX/ACM Symposium on Networked System Design and Implementation, 2004.

[4] C.C. Han, R. Kumar, R. Shea, E. Kohler, M. Sricastava, "A dynamic operating system for sensor nodes," Proceedings on MobiSys, pp.163-176, 2005.

[5] 은성배, 소선섭, 김병호, “센서 투명성을 지원하는 센서노드 운영체제 구조,” 한국정보과학회 종합학술대회 논문집, Vol. 35, No. 1(A), pp.311-312, 2008.

[6] 방상호, 은성배, “센서 투명성을 지원하는 센서 디바이스 매니저,” 한국정보처리학회 추계학술발표대회 논문집, Vol. 15, No. 2, pp.998-1000, 2008.

[7] 박영범, 은성배, “센서 Plug&Play를 지원하는 센서 노드 플랫폼,” 신호처리 합동학술대회 논문집, Vol. 22, No. 1, pp.127-130, 2009.

저 자 소 개

박 정 선

2011년 한남대학교 정보통신공학과 학사.
현재, 한남대학교 정보통신공학과 석사과정.
관심분야: 센서노드 운영체제, 임베디드 시스템

Email: jslovecom@gmail.com

은 성 배

1985년 서울대 전산학과 학사.
1987년 KAIST 전산학과 석사.
1995 KAIST 전산학과 박사.
현재, 한남대학교 정보통신공학과 교수.
관심분야: 실시간시스템, 유비쿼터스, 센서네트워크
Email: sbeun@hnu.kr

윤 현 주

1988년 서울대 컴퓨터공학과 학사.
1990년 KAIST 전산학과 석사.
1997년 KAIST 전산학과 박사.
현재, 금오공과대학교 컴퓨터공학과 조교수.
관심분야: 운영체제, 임베디드 시스템, 센서네트워크
Email: juyoon@kumoh.ac.kr