컴포넌트 기반 멀티 스레드 지원 초경량 운영체제 설계 및 구현

김태훈, 서대홍, 이승룡
경희대학교 컴퓨터공학과
e-mail:kimth@oslab.khu.ac.kr

Design and Implementation of Component Based Multi-Thread Lightweight Sensor OS

Tae-Hoon Kim, Dae-Hong Seo, Seung-Young Lee
Dept of Computer Science, Kyung-Hee University

요 약

센서 네트워크 환경에서 센서 노드에게 가장 큰 이슈는 저전력이다. 이러한 센서 노드에서 저전력화를 제공하는 것은 하드웨어 뿐만 아니라 소프트웨어에서도 중요하다. 유휴 시간에 마이크로 컨트롤러가 활성 상태로 대기 하는 대신 마이크로 컨트롤러가 제공하는 파워 슬립 모드를 이용하여 모든 주변 장치의 전원을 차단함으로서 저전력을 실현할 수 있다. 그러나 빈번한 슬립 모드 진입은 오히려 더 많은 전력을 소비하기 때문에 슬립 모드로 들어가는 시기와 나오는 시기를 적절하게 제어 하는 것은 쉬운 일이 아니다. 그러므로 본 논문에서는 타이머 인터럽트를 활용하여 슬립 모드 제어 정책을 포함하는 컴포넌트 기반의 멀티 스레드 지원 센서 OS를 구현하였다. 코드 크기의 최적화로 성능 향상을 꾀하였으며, 이로서 전력 소비도 줄일 수 있다. 또한, 컴포넌트 기반의 구조는 다양한 하드웨어를 쉽게 지원할 수 있으며, 응용 분야에 따라 다양한 어플리케이션을 쉽게 제작할 수 있도록 설계하였다.

1. 서론

컴퓨팅 환경이 유비쿼터스 환경으로 진화됨에 따라 마이크로 컨트롤러의 활용이 극대화 되어가고 있다. 센서 네트워크 환경의 센서 노드에 장착된 마이크로 컨트롤러들은 주위 환경으로부터 센서들이 데이터 들을 수집하고 수집된 결과를 전송하는데 대한 제어를 담당하고 있다. 예전의 센서 노드들에 장착된 마이크로 컨트롤러의 경우 데이터를 수집하고 전송하는 간단한 기능을 수행한 데 비해, 요구사항이 증대됨에 따라 하나의 노드에 다양한 기능의 센서가 부착될 수 있고, 그 노드들은 다양한 센서로부터 데이터를 수집함은 문론, 네트워크로 이웃 노드들에게 얼마나 효율적인 방법으로 전달할 것인가가 중요하게 되었다. 불필요한 패킷을 줄이거나, 적절한 라우팅 방법을 채택하여 전력을 적게 소비해야 한다. 따라서 마이크로 컨트롤러는 기본적으로 커널 스레드, 라우팅 또는 어드레싱 등 네트워킹을 담당하는 스레드등 여러개의 스레드를 가질 수 있으며, 그 스레드들을 적절하게 스케쥴링 할 수 있어야 하며, 스레드를 선점형과 비선점형 중에 선택해야 한다.

센서 네트워크 환경의 센서 노드는 배터리와 같은 전원을 공급받아 전력 공급이 원활하지 않을 수도 있으며, 무한정 사용할 수도 없다. 따라서 적은 전력을 소비하여 오랜 시간 동작 가능하도록 하는 것이 센서 노드에서 가장 중요하며, 하드웨어부터 소프트웨어까지 저전력을 고려하지 않으면 안된다.

예전의 임베디드 시스템 프로그래밍 기법은 운영체제 없이 프로그램을 구현하는 것이다. 이런 프로그래밍 기법은 마이크로 프로세서가 간단한 디바이스를 제어하는 용도와 같은 간단한 기능을 수행하는 데는 문제가 없지만, 센서 노드에 장착된 마이크로 컨트롤러의 경우 여러 센서들의 동작을 지원해야 함은 문론 통신 모듈의 통신까지 주기적으로 제어해야 하는 상황이므로 직접 모든 프로그램을 운영체제 없이 작성하는 것은 재사용성이 떨어지며, 유지 보수가 어렵다. 지금까지 운영체제 없이 프로그램을 작

성했던 것은 하드웨어 적인 제약이 컸기 때문이다. 즉, 메모리의 한계와 프로세서의 오버헤드를 고려해서 였다. 그러나 현재 마이크로 프로세서의 경우 충분한 처리 능력과 내부 RAM을 가지고 있으므로 운영체제가 올라갈 수 있다. 그렇다고 범용 운영체제만큼 큰 운영체제를 올릴 수는 없다. 범용 운영체제는 전력 소비 보다는 사용자의 편이성과 다양한 기능을 제공하는 것이 중요하지만 센서 노드에 올라가는 운영체제는 필요한 최소한의 기능만을 포함해야 하고, 저전력을 고려하여야만 하며, 경우에 따라 실시간 처리를 요구하기도 한다.

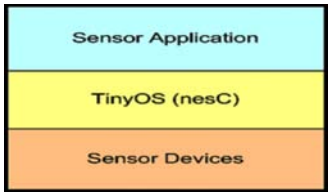
따라서 본 논문에서는 다양한 센서 디바이스 드라이버를 지원하며, 센서 노드들간의 통신이 가능한 센서 네트워크 프로토콜을 지원하는 컴포넌트 기반의 초경량 운영체제 (CAMEL : Component Based Multi-Thread Lightweight Sensor OS) 를 개발하였으며 저전력화를 실현하기 위해 코드 크기를 최적화 하였으며 전력 관리 모듈을 적용하였다.

2. 관련 연구

대표적인 임베디드 센서 운영체제로는 버클리 대학의 TinyOS[1], Mantis Project[2], 그리고 국내 ETRI에서 만든 QPlus-N[3] 을 들 수 있다

● Berkeley TinyOS

TinyOS는 버클리 대학에서 자체 개발한 Mote-kit이라는 개발 보드에서 프로그래밍 할 수 있도록 제공된 Tool-Kit이라 생각할 수 있다. Mote-kit은 빛 센서, 온도 센서, 음성 센서등을 장착할 수 있으며, NesC라는 컴포넌트 기반의 자체 언어를 사용하여 개발할 수 있다. 문론 Mote-kit 하드웨어를 사용하여 C언어나 어셈블러로 직접 프로그래밍할 수도 있다. NesC를 통해 작성된 프로그램을 컴파일 하면 C언어로 변환된 후 다시 avr-gcc컴파일러를 통해 최종 바이너리 형태로 변환되는 형태를 가지고 있다.



[그림 1]. TinyOS 구조도

TinyOS는 인터럽트 기반의 비선점형 스케쥴링 방식을 체택하고 있으며, 공유된 Single-Stack을 사용하고 있다. 즉, 컨텍스트 스위칭이 이루어 질 때 하나의 스택을 사용하여 현 스레드의 컨텍스트를 저장한다. TinyOS는 Mote-kit 드라이버만 지원하여 다른 하드웨어에서 실행될 수 없다.

● Mantis Project

Colorado 대학에서 만든 Mantis Project Team은 C언어의 Cross-Platform API를 제공하고 있다. Interrupt기반 비선점형 스케쥴링을 지원하며, Multi-Stack 모델을 채택하였다. 즉 각각의 스레드 당 자신의 레지스터 값을 저장할 수 있는 메모리 영역을 가진다.

● QPlus-N

ETRI에서 만든 QPlus-N은 현재 0.8.0 버전까지 나와 있으며, 역시 C언어 API를 제공하고 있다. 역시 비선점형 Interrupt 기반의 이벤트 드리븐 방식을 사용하며, Single-Stack의 MultiThread OS이다. 스레드 스케쥴링 알고리즘으로는 Simple-FIFO, Timed-FIFO, Preemption-RR (선점형 Round Robin)을 지원한다.

3. CAMEL 센서 OS 설계

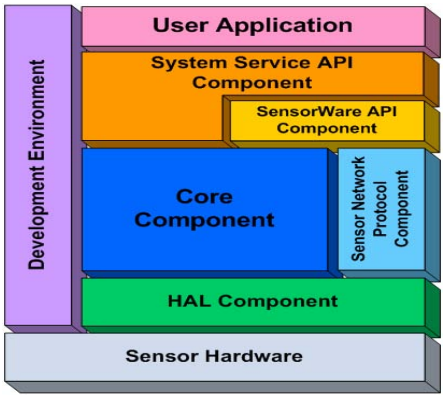
CAMEL 센서 OS는 크게 RealTime-커널, non-Realtime-커널로 나뉜다. 컴포넌트 구조는 이와 같은 요소를 컴포넌트화 하여 사용자로 하여금 응용 대상에 따라 재구성할 수 있게 만든다.

HAL은 Hardware Abstraction Layer의 약자로서 현존하는 다양한 하드웨어 드라이버를 지원하며, 또한 컴포넌트 기반으로 추가되는 하드웨어 사양에도 신속하게 적응할 수 있도록 한다. 현재 Mote-kit과 Nano-24의 개발 보드의 드라이버가 개발 되었다. 이들은 모두 ATmega128을 사용하지만 Port의 사용과, 센서 사양이 다르므로 서로 다른 드라이버의 개발이 필요하다. 또한 CAMEL의 커널은 저전력, 소스 코드 크기의 최소화, 센서 네트워크 프로토콜의 구현, 표준 API의 제공을 목표로 하고 있다.

3.1 CAMEL 센서 OS 구조

CAMEL 센서 OS는 하드웨어 계층, 하드웨어 추상화 계층(HAL:Hardware Abstraction Layer), 커널이 포함된 코어 컴포넌트 계층, 네트워크 프로토콜을 포함하는 네트워크 계층, 그 위에 응용 프로그램 개발자를 위한 API와 응용 프로그램이 존재한다. SensorWare는 상황 인지 미들웨어(Context

Awareness Middle-Ware)를 지원하기 위한 API 계층으로서 특정 목적에 알맞도록 데이터 값을 추출해 전달하거나 파싱하는 역할을 담당한다. 기존의 개발 환경은 텍스트 기반으로서 응용 프로그램 개발자에게 어려운 점이 많다. 그러나 컴포넌트 구조의 장점을 이용하여 그래픽 환경 개발 툴킷을 제공할 수 있으며 응용 프로그램 개발자는 좀 더 쉽게 응용 프로그램을 개발할 수 있다. [그림 2] 에서는 제안하는 센서 OS의 구조를 나타내고 있다.



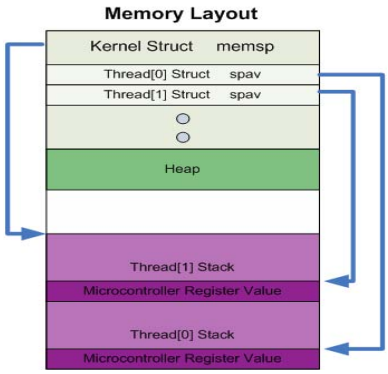
[그림 2] CAMEL 센서 OS 구조

Sensor Hardware 계층은 ATmega128을 기반으로 하고 있으며 향후 ARM CPU도 지원할 계획이다. HAL Component에는 USART Communication Component, Battery Check Module, 센서 드라이버 Module, MCU Module, Timer Module, RF 네트워크 Module이 포함된다. USART Communication Module은 Serial Port를 통해 PC와 통신하는 방법을 제공한다. 센서 드라이버에는 빛 센서 드라이버, 온도 센서 드라이버가 구현되었다.

Core Component에는 Power Manager, 센서 Manager, 스레드 Manager, Event Manager가 존재한다. 이들은 각각 실시간성을 지원하는 것과 그렇지 않은 것 두가지로 분류된다. Power Manager는 Sleep Mode Control을 통해 저전력을 실현한다. 센서 Manager는 요청에 따라 센서로부터 값을 받아오거나 Actuator 센서로 값을 전송하여 해당 작업을 수행하도록 한다. 스레드 Manager에서는 RoundRobin 스케쥴링, Priority Based 스케쥴링을 지원한다. Event Manager는 Timer1,2,3,4 인터럽트를 관리하고 사용자가 사용할 수 있도록 하며 그 외 외부 인터럽트 등 다른 이벤트들도 처리할 수 있게 해준다.

3.2 프로세스 스케쥴링

CAMEL 센서 OS는 스레드가 생성될 때마다 자신의 컨텍스트 데이터를 저장할 메모리 영역을 할당받는다. 그러므로 마이크로 프로세서의 내부 RAM 용량에 따라 생성될 수 있는 최대 스레드 수가 달라질 수 있다. [그림 3]에서는 커널 메모리 레이아웃을 나타낸다



[그림 3]. 커널 메모리 레이아웃

커널 구조체는 현재 스택 포인터 값, 스레드 수, 현재 수행되는 스레드의 인덱스를 가진다. 스레드를 생성하면 자신의 스레드 컨트롤 블록이 메모리에 할당된다. 이 스레드 컨트롤 블록에는 우선순위, 현재 상태, 남은 Sleep Time, 스택 포인터 위치를 포함한다. 스레드가 생성되면 스레드 컨트롤 블록에 생성된 콜백 주소 값과 컨텍스트 데이터를 스택에 저장하고 저장한 후의 스택 포인터를 스레드 컨트롤 블록의 스택 포인터 변수에 저장한다. 컨텍스트 데이터는 그 스레드가 수행되는 동안의 MCU의 레지스터 값들을 의미한다. memsp는 현재 스택 포인터 값을 가르키는 변수이며, spav는 자신의 컨텍스트 데이터가 저장되어 있는 메모리 위치를 가르킨다. 스케쥴러에 의해 컨텍스트 Switch가 이루어져야 되는 상황에 이르면, 스케쥴러는 먼저 현재 실행되고 있는 스레드의 StackPointer와 컨텍스트 데이터를 스레드를 생성할 때 할당되었던 자신의 메모리 영역에 저장한다. 그리고 바뀌어야 할 스레드의 Stack Pointer를 스택에 푸시하고, 컨텍스트 데이터를 레지스터로 옮긴다. 이로서 컨텍스트 Switch가 끝나고 인터럽트 복귀 명령어를 만나면 스택 포인터에 저장된 새로운 스레드의 복귀주소로 점프한다.

현재 CAMEL 센서 OS에서 채택하고 있는 스케쥴링 알고리즘은 RoundRobin 스케쥴링, Priority Based 스케쥴링이다. 스레드를 생성할 때 Priority값을 할당받도록 되어있으며, Priority가 같을 경우 RoundRobin 스케쥴링 알고리즘을 사용한다.

3.3 센서로부터 값 추출

빛 센서, 조도 센서의 값을 추출하여 USART를 통하여 Serial Port로 값을 수신받을 수 있다. 또한 Serial Port로 특정 값을 전송하면 센서 노드는 그 값에 따라 각각 다른 행동을 할 수 있도록 프로그래밍 할 수도 있다. 센서의 값은 Analog to Digital Converter를 사용하여 읽어들일 수 있다. 즉 마이크로 프로세서의 포트에 연결된 센서는 전원을 공급받으면 전압이나 전류 레벨로 출력한다. ADC는 전압 또는 전류 레벨을 디지털 값으로 바꿔서 AVCL 레지스터에 전송한다.

3.4 전원 관리 모듈

전원 관리 모듈(Power Management Module)은 ATmega128 MCU가 Sleep Mode로 전환한 후 외부 인터럽트나 타이머 0 인터럽트를 사용하여 깨어날 수 있는 것을 활용하였다. MCUCR 레지스터에 값을 할당함으로서 Sleep Mode를 설정할 수 있는데 ATmega128이 지원하는 Sleep Mode는 다음과 같다. Idle Mode, ADC Noise Reduction Mode, Power Down Mode, Power Save Mode, Standby Mode, Extended Standby Mode를 지원한다. CAMEL SensorOS에서는 Idle 스레드 수행 중에 전원 관리 모듈에 의해 전력 관리가 이루어지며, Timer0의 최대 분주 카운트가 7초 까지 이고, 2초 이내로 Sleep Mode로 진입과 복귀를 반복하면 전력 소비가 더 커지는 것을 감안하여 최소 2초 이상 Sleep할 경우 Sleep Mode로 전환된다.

4. 구현 및 평가

구현한 센서 OS를 테스트 하기 위해 버클리의 Mote-kit 보드를 사용 하였으며 Mote-kit의 사양은 다음과 같다.

[표 1] Mote-kit 테스트 보드 사양

요 소	사 양
Micro Controller	ATMEL ATMega128
Internal ROM	4KB
Internal RAM	4KB
Internal FLASH	128KB

그리고 CAMEL 센서 OS를 개발한 개발 환경은 다음과 같다.

[표 2] 센서 OS 개발 환경

요 소	사 양
운영체제	Redhat Linux 9 (2.4.18)
컴파일러	avr-gcc 3.3
유틸리티	binutils-avr 2.15-1
표준 라이브러리	avr-libc 1.0.4-1
ISP 전송 유틸리티	uisp20030820tinyOS
개발언어	C, 어셈블러

개발한 커널은 타이머 모듈, 전원 관리 모듈, 인터럽트 핸들링 모듈, 비선점형의 스레드 생성 및 스케쥴러 모듈, 아날로그-디지털 컨버터 모듈, USART 통신 모듈을 개발하였다. 드라이버는 버클리 Mote-kit의 빛-센서, 온도 센서, 배터리 체크 모듈, led controller를 개발하였다. 개발 환경은 Cross-Compile 환경으로 Redhat Linux에서 프로그램을 작성한 후 ISP 유틸리티를 사용하여 ATmega128의 FLASH memory에 저장하게 된다.

5. 결론

센서 네트워킹 환경에서 가장 중요한 이슈로 부각되고 있는 저전력화를 실현하기 위해 다양한 전원 관리 모드를 지원하는 컴포넌트 기반 센서 OS를 개발하였다. 컴포넌트 기반 구조는 새롭게 추가되는 다양한 하드웨어의 드라이버를 쉽게 추가할 수 있으며, 사용자도 응용 프로그램의 소스 코드를 수정하지 않고 다른 하드웨어에서도 동작 가능하도록 할 수 있다. 차후에 연구될 사항으로 Zigbee 표준을 지원하는 센서 네트워크 프로토콜을 지원하는 것과 커널의 안정화 이다. 커널을 소수의 개발자가 개발한다는 것은 쉬운 일이 아니다. 특히 스케쥴러는 수많은 테스트와 디버깅을 필요로 한다. 어떠한 경우에서도 시스템이 다운되는 일이 없는 완벽한 스케쥴러는 구현되어야만 할것이다.

참고문헌

[1] http://www.tinyos.net
[2] http://mantis.cs.colorado.edu
[3] http://octacomm.net
[4] 윤덕용, “AVR ATmega128 마스터”, Ohm사, 2004
[5] Edgar H. Callaway, Jr. “Wireless Sensor Networks", 2004
[6] AVR-libc 10.4 manual, http://www.avrfaks.net/
[7] 홍성수, 김태형, “동적으로 재구성 가능한 센서 네트워크를 위한 상태 기반 운영체제의 설계”, 홍성수, 2004.11