TinyOS를 위한 향상된 전력관리 기법

우장복*, 서효중*
*가톨릭대학교 컴퓨터공학과
e-mail:sofe4u@catholic.ac.kr

An Improved Power Management for TinyOS

Jangbok Woo*, Hyo-joong Suh*
*Dept of Computer Science and Engineering, The Catholic University of Korea

요 약

센서 네트워크는 관찰 지역 내의 정보를 수집하는 센서 노드들로 구성된다. 센서 노드는 제한된 용량의 배터리를 갖고 동작하므로 센서 노드의 배터리 파워를 효과적으로 사용하여 최대한 센서 노드의 수명을 길게 하는 것이 센서 네트워크의 중요한 고려사항 중의 하나이다. 센서 네트워크에서 사용되는 운영체제들은 이를 위해 대부분 저전력 모드를 고려하여 설계된다. 무선 임베디드 센서 네트워크를 위해 설계된 운영체제인 TinyOS도 간단하며 강력한 전력관리 기법을 제공한다. 그러나 TinyOS에서 제공하는 전력관리 기법은 마이크로컨트롤러 자체의 저전력 모드를 고려하지 않아서 마이크로컨트롤러가 제공하는 저전력 모드를 실제로 충분히 사용할 수 없다. 본 논문에서는 TinyOS에서 마이크로컨트롤러의 저전력 모드를 충분히 활용할 수 있도록 개선하여 보다 향상된 전력관리 기법을 제안한다.

1. 서론

최근 무선 통신 기술과 전자 장비의 발달, 초소형 네트워크 디바이스 기술의 발달로 무선 센서 네트워크가 전 세계적으로 매우 빠르게 개발되고 있다. 센서 네트워크란 각종 센서에서 수집한 정보를 데이터화하여 다른 컴퓨팅 객체에 무선으로 전달할 수 있도록 구성한 네트워크를 말한다. 센서 네트워크의 대부분은 정확한 정보를 빠르게 전달하는 신뢰성과, 효율적인 라우팅을 위한 집합 알고리즘, 센서 노드의 소형화, 저전력 소모에 초점을 맞추고 있다. 이러한 센서 네트워크를 구성하는 센서 노드는 다양한 센싱 작업을 위한 센싱 장치와, 해당 값을 처리하기 위한 처리 장치, 센서 노드들 간의 통신을 위한 무선 송수신 장치, 센서 노드에 전력을 공급하는 전력 장치 등으로 구성된다[1].

센서 네트워크를 위한 운영체제로는 TinyOS, YATOS, Contiki, MANTIS, SOS 등이 있는데, 이 중에서 주목받고 있는 운영체제가 TinyOS이다[2]. TinyOS는 무선 임베디드 센서 네트워크를 위해 설계된 오픈 소스 운영체제로써, 이벤트 발생에 의한 상태 머신 기반의 프로그래밍 개념을 사용하며, 제한된 메모리 공간의 효율적인 이용과 프로세싱의 동시성 등을 지원한다. 또한, TinyOS는 재사용 가능한 컴포넌트 기반의 운영체제이며, 센서 노드의 중요한 요구사항의 하나인 저전력 소모를 구현하기 위해서 마이크로컨트롤러가 사용되지 않는 동안 수면 상태로 들어가 전력소모를 줄인다[3]. 그러나 TinyOS에서 제공하는 전력관리 기법은 마이크로컨트롤러 자체의 저전력 모드를 고려하지 않아서 마이크로컨트롤러가 실제로 제공하는 저전력 모드를 충분히 활용할 수 없다[4]. 본 논문에서는 MSP430 마이크로컨트롤러를 사용하는 센서 노드에서 수면 모드의 세분화된 정의를 통해서 TinyOS의 전력관리 기법을 개선하여 마이크로컨트롤러의 저전력 모드를 충분히 활용할 수 있는 방법을 제안한다.

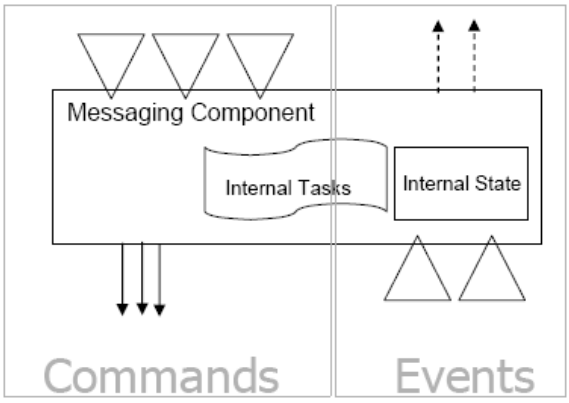
본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해서 기술하고, 3장에서는 제안하는 기법에 대해서 설명하고, 4장에서는 제안하는 기법의 성

능을 평가하며, 5장에서는 결론 및 향후 연구과제에 대해서 기술한다.

2. 관련연구

2.1 TinyOS의 특징

TinyOS는 미국의 UC 버클리 대학에서 진행해 온 스마트 더스트(Smart Dust) 프로젝트에 사용하기 위하여 개발된 컴포넌트 기반의 내장형 운영 체제이다. 센서 네트워크의 요구사항인 제한된 자원, 반응 동시성, 유연성, 저전력 소모를 고려하여 설계되었으며, 핵심 코드는 4000바이트 이하이고, 데이터 메모리는 256바이트 이하이며, 이벤트 기반의 멀티 태스킹을 지원한다[3]. TinyOS는 기본적으로 아래 (그림 1)과 같은 컴포넌트 모델로 구성되며, 컴포넌트는 커맨드 핸들러, 이벤트 핸들러, 고정 크기의 프레임, 태스크로 구성된다.



(그림 1) TinyOS의 컴포넌트 모델

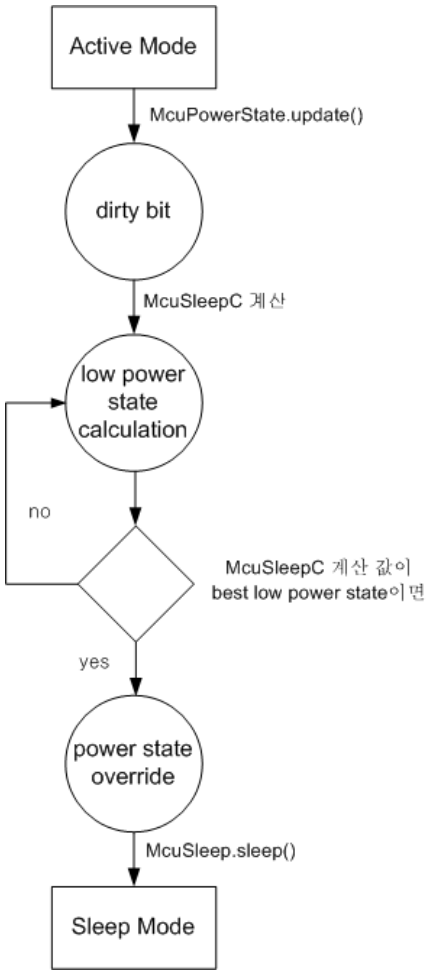
2.2 TinyOS의 전력관리 기법

TinyOS의 전력관리 기법은 단순하고 매우 강력하다. 기본적으로 태스크 큐의 작업이 모두 처리되어 태스크 큐가 비어있는 상태가 되면, 프로세서는 자동적으로 수면 모드로 전환되게 된다. 이러한 수면모드, 활동 모드의 상태 변화는 컴포넌트 수준의 'StdControl' 인터페이스를 통해서 제어되며 'adjustPower' 코드가 다른 센서 노드의 구성요소와의 의존관계를 판단하여 수면 모드로의 진입여부가 결정되게 된다. (그림 2)는 'adjustPower'의 코드의 예이다. 태스크 큐가 비게 되면 스케줄러는 'adjustPower'를 고려하여 'McuSleep.sleep()' 함수를 호출하는데, 'McuSleep.sleep()' 함수는 마이크로컨트롤러를 수면 모드로 진입하게 만드는 역할을 한다. 그리고 수면 모드의 해제는 인터럽트를 통해 이루어진다.

```
command HPLPwrMgr.adjustPower {
   // chips/CPU determines exact sleep mode
   if (external timers running) {
     mode = IDLE;
   } else if (radio running) {
     mode = IDLE;
   } else if (adc running) {
     mode = ADC_NR;
   } else {
     mode = POWER_SAVE;
   }
   sleep_state = mode;
}
```

(그림 2) adjustPower 코드

TinyOS는 마이크로컨트롤러, 플랫폼, 어플리케이션의 3가지를 위한 각각의 전력관리 기법을 제공한다. 본 논문에서는 이 중에서 마이크로컨트롤러에 대한 전력관리 기법에 대해서 구체적으로 살펴보도록 한다. TinyOs는 마이크로컨트롤러의 전력관리를 위해서 기본적으로 'dirty-bit', 'low power state calculation', 'power state override'의 세 가지 정책을 사용한다.



(그림 3) 전력관리 상태 변화

첫 번째로 마이크로컨트롤러가 저전력 모드로 진입할 수 있는 상태로 변환되어 HPL(Hardware

Presentation Layer) 컴포넌트가 바뀔 때 'McuPowerState.update()' 함수가 호출되게 되는데 이것을 'dirty-bit'라고 한다. 해당 함수가 호출되게 되면 다음번에 'McuSleep.sleep()'가 호출되어 수면 모드로 상태가 전이될 때, McuSleepC를 다시 계산하게 된다. McuSleepC는 'McuSleep', 'PowerState', 'PowerOverride' 인터페이스를 제공하는 컴포넌트이다. 두 번째 정책인 'low power sate calculation'은 센서 노드의 다른 구성요소와의 의존관계를 고려하여 마이크로컨트롤러의 전력소모를 최소화하기 위해 McuSleepC에 대한 계산을 수행하는 것이다. 이 때 잦은 계산은 시스템의 부하와 지터(jitter)를 증가시키므로 McuSleepC에 대한 계산을 가능한 한 최소화할 필요가 있다. 세 번째 정책인 'power state override'는 'low power state calculation' 단계에서 McuSleepC를 계산하여 마이크로컨트롤러의 최적의 저전력 상태를 도출했을 때 'PowerOverride.lowestState()' 함수를 호출하는 것이다. McuSleepC에는 이를 위한 명령어가 기본적으로 포함되어 있다.

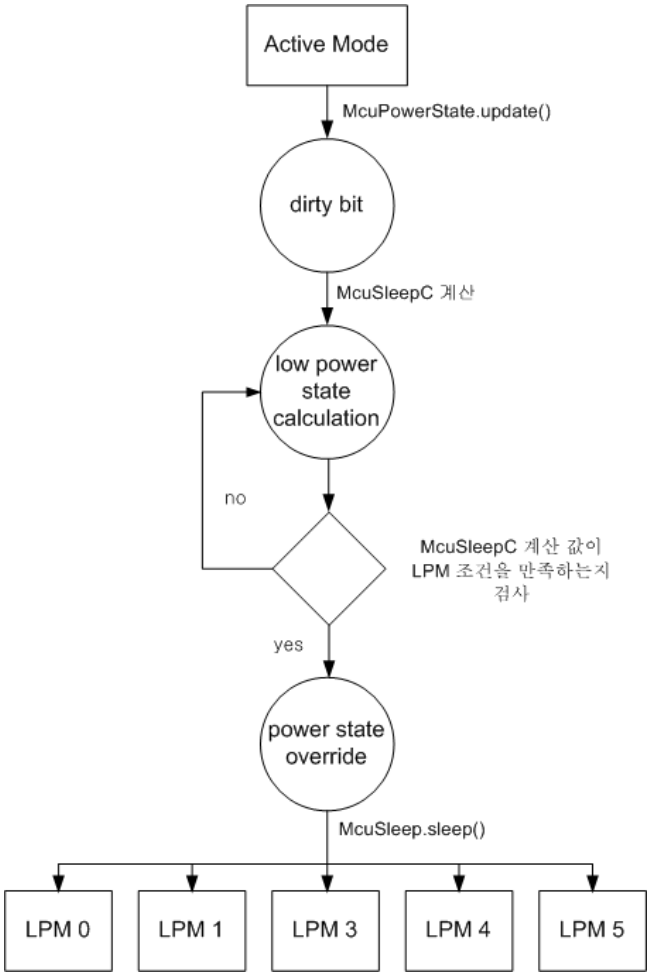
3. 제안하는 기법

TinyOS의 전력관리 기법은 간단하고 강력하지만 단순하게 태스크 큐에 작업이 들어있는지의 유무에 따라서 스케줄러가 수면 모드로의 진입을 결정하게 되므로 마이크로컨트롤러가 실제로 제공하는 저전력 모드를 충분히 활용할 수 없는 단점이 있다.

<표 1> MSP430의 전력 모드에 따른 상태 변화

전력 모드	소모 전류	CPU와 클럭 상태
Active Mode	340uA	CPU is active, all enabled clocks are active
LPM 0	75uA	CPU, MCLK are disabled SMCLK , ACLK are active
LPM 1	-	CPU, MCLK, DCO osc. are disabled DC generator is disabled if the DCO is not used for MCLK or SMCLK in active mode SMCLK , ACLK are active
LPM 2	17uA	CPU, MCLK, SMCLK, DCO osc. are disabled DC generator remains enabled ACLK is active
LPM 3	2uA	CPU, MCLK, SMCLK, DCO osc. are disabled DC generator disabled ACLK is active
LPM 4	0.2uA	CPU and all clocks disabled

<표 1>은 전압이 3V일 때, MSP430 마이크로컨트롤러의 저전력 모드에 따른 소모 전류 및 CPU와 클럭의 상태 변화를 나타낸다[5]. MSP430 마이크로컨트롤러를 사용하는 센서 노드에 TinyOS의 전력 기법을 적용하면, 태스크 큐에 작업이 있을 때는 Active Mode를, 태스크 큐에 작업이 없는 경우에는 LPM(Low Power Mode)3만을 사용하게 되므로, LPM0, LPM1, LPM2, LPM4는 사용하지 않게 된다. 따라서 해당 모드로 동작할 수 있는 경우에도 Active 모드로 동작하게 됨으로써 불필요한 전력 소모가 발생하게 된다.



(그림 4) 변형된 전력관리 상태

위의 (그림 4)는 본 논문에서 제안하는 기법에 따른 변형된 전력관리 상태 변화를 나타낸 그림이다. McuSleepC 를 계산할 때 <표 1>의 전력 모드에 따른 CPU와 클럭 상태를 반영하기 위해서 해당 인자를 검사하는 값을 추가하였다. 그리고 'power state override' 정책에서 수면 모드로 전환하는 대신 조건 값에 따른 저전력 모드로 상태를 전환하도록 수면모드를 세분화하여 MSP430 마이크로컨트롤러의 저전력 모드를 모두 반영할 수 있도록 전력관리 기법을

수정하였다.

4. 성능평가

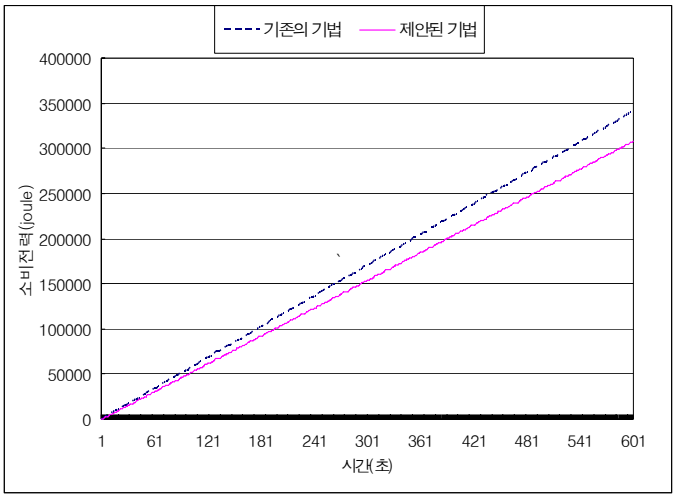
본 절에서는 3절에서 제시하는 기법이 효율적임을 보이기 위해서 기존의 TinyOS의 전력관리 기법과 제안하는 기법을 비교하였다.

4.1 실험 환경

3V에서 동작하는 MSP430 마이크로컨트롤러를 사용하는 센서 노드들과 베이스 스테이션으로 구성된 센서 네트워크에서, 각각의 센서 노드는 자신의 센싱 범위 내의 조도 및 온도를 파악하여 조도는 3초마다, 온도는 4초 마다 해당 값을 베이스 스테이션으로 전송하며, 베이스 스테이션은 10초마다 자신이 관리하는 센서 노드들에게 신호를 보낸다고 가정하였다.

그리고 상기의 센서 네트워크에서 어떤 한 센서 노드가 10분 동안 동작한다고 가정하고, 기존의 TinyOS의 전력관리 기법을 적용했을 때의 MSP430 마이크로컨트롤러의 소모 전력과 제안하는 기법을 적용했을 때의 MSP430 마이크로컨트롤러의 소모 전력을 비교하였다.

4.2 실험 결과



(그림 5) 단위시간당 전력소모 비교 그래프

위의 (그림 5)에서 점선은 기존의 TinyOS의 전력기법을 적용했을 때의 수행결과이고, 실선은 제안한 기법을 적용했을 때의 수행결과이다. MSP430의 마이크로컨트롤러의 저전력 모드에 맞게 TinyOS의 저전력 기법을 수정한 결과 평균적으로 전력소모가 10% 정도 감소하였다.

5. 결론 및 향후과제

본 논문에서는 TinyOS의 전력관리 기법 중에서 마이크로컨트롤러에 대한 전력관리 기법을 MSP430 마이크로컨트롤러의 저전력 모드의 특성에 맞게 개선하여 보다 효율적인 전력관리 기법을 제안하였다. 향후 연구과제는 본 논문에서 제안한 기법을 확장하여 MSP430 마이크로컨트롤러와 함께 센서 노드에서 주로 사용되는 ATmega 마이크로컨트롤러의 저전력 모드의 특성도 만족하는 전력관리 기법을 연구하는 것이다.

참고문헌

[1] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci, "A Survey on Sensor Networks," IEEE Communications Magazine, 2002.

[2] Lucas Francisco Wanner, Arliones Stevert Hoeller Junior, Fauze Valério Polpeta and Antônio Augusto Fröhlich, Operating System Support for Handling Heterogeneity in Wireless Sensor Networks, In: Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation, Catania, Italy, 2005

[3] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An operating system for wireless sensor networks," in Ambient Intelligence. New York, NY: Springer-Verlag, To Appear

[4] Vlado Handziski, Joseph Polastre, Jan-Hinrich Hauer, Cory Sharp, Adam Wolisz, David Culler, “Flexible Hardware Abstraction for Wireless Sensor Networks” in proceedings of the 2nd European Workshop on Wireless Sensor Networks, 2005

[5] http://www.ti.com/msp430