양방향 통신을 지원하는 시분할 기반 선형 무선 센서 네트워크를 위한 원격 펌웨어 업데이트 방법

A Remote Firmware Update Mechanism for a TDMA-based Bidirectional Linear Wireless Sensor Network

문 정 호*, 김 대 일, 박 래 정, 이 형 봉, 정 태 윤
(Jung-Ho Moon, Dae Il Kim, Lae Jeong Park, Hyung Bong Lee, and Tae Yoon Chung)

Abstract: A wireless sensor network inherently comprises a plurality of sensor nodes widely deployed for sensing environmental information. To add new functions or to correct some faulty functions of an existing wireless sensor network, the firmware for each sensor node needs to be updated. Firmware update would be quite troublesome if it requires the gathering, reprogramming, and redeploy of all of already deployed sensor nodes. Over-the-air programming (OTA) facilitates the firmware update process, thereby allowing convenient maintenance of an already-deployed sensor network. This paper proposes and implements a remote firmware update mechanism for a TDMA-based wireless sensor network, in which the firmware for sensor nodes constituting the TDMA-based sensor network can be easily updated and the update process can be conveniently monitored from a remote site. We verify the validity of the proposed firmware update method via experiments and introduce three wireless sensor networks installed in outdoor sites in which the proposed firmware update mechanism has been exploited.

Keywords: WSN, firmware update, OTA, BiWSLP, TDMA

I. 서론

무선 센서 네트워크(Wireless Sensor Network, WSN)는 그림 1과 같이 주변의 환경 정보를 감지할 수 있는 센서와 근거리 무선 통신 기능을 탑재한 다수의 센서 노드(sensor node)들로 구성되는 무선 네트워크이다. 센서 노드는 주변 환경 데이터를 수집하고 이를 무선 네트워크를 통하여 비교적 가까운 거리에 위치한 주변의 다른 센서 노드 또는 싱크 노드(sink node)에게 전송하는 정도의 비교적 제한적인 기능만을 가지고 있다.

센서 노드는 유선을 이용하기 어려운 지역에 설치되어 소용량 배터리로 동작하는 경우가 많기 때문에 저전력 구동 능력이 필수적이며 따라서 소비 전력은 무선 센서 네트워크에 있어서 매우 중요하게 고려되는 요소이다. 이런 이유로 무선 센서 네트워크에 적합하도록 개발된 네트워크 기술은 저전력 구동을 주요 기능으로 가지고 있는데 S-MAC [1], T-MAC [2], B-MAC [3] 등이 그 예라고 할 수 있다.

센서 네트워크를 설치한 후 새로운 기능을 추가하거나 동작의 결점이 발견되어 그 기능을 수정하려면 센서 노드의 동작을 결정하는 펌웨어를 교체하는 작업이 필요하다. 이를 위하여 센서 노드가 설치되어 있는 곳으로 이동하여 펌웨어 다운로드를 수행하거나 설치되어 있는 모든 노드들을 수거하여 펌웨어 다운로드 후 재설치 한다면 많은 시간과 비용이 든다. 더욱이 센서 노드가 지역적으로 광범위하게 위치하거나 교량이나 산악 지역 같이 접근이 용이하지 않은 곳에 위치하는 경우에는 펌웨어 교체가 더욱 어렵다. 이런 문제를 해결하기 위하여 원격지에서 무선으로 센서 노드들에게 펌웨어를 전송하여 업데이트 하는 방법이 고안되었다.

이 논문은 [4]와 [5]에서 발표한 양방향 통신을 지원하는 시분할 기반 선형 네트워크용 프로토콜(BiWSLP)을 적용한 무선 센서 네트워크에서 원격에서 펌웨어 업데이트를 수행할 수 있는 무선 펌웨어 업데이트 방법을 제시하고 이를 실제로 구현하여 그 효용성과 가능성을 보인다. 제 II 장에서 양방향 통신을 지원하는 시분할 기반 선형 무선 센서 네트워크용 프로토콜 BiWSLP에 대해 간략하게 소개하고 제 III 장에서 기존에 알려져 있는 무선 펌웨어 업데이트 방법에 대한 설명과 아울러 BiWSLP에 적합한 무선 펌웨어 업데이트 방법을 제안한다. 제 IV 장에서는 이 논문에서 제안한 방법을 적용한 실험 결과를 보이고 현재 외부 현장에서 운영되고 있는 센서 네트워크에 이 논문의 업데이트 방법을 적용한 사례에 대해서 소개한다. 그리고 V 장에서는 이 논문에서 제안한 무선 펌웨어 업데이트 방법의 구현에 필요한 소비 전력을 분석하고 이를 다른 방법과 비교한다.

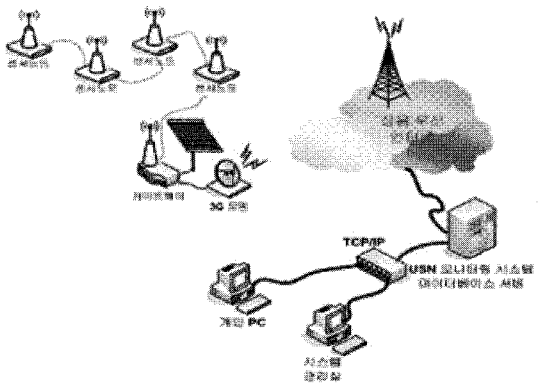


그림 1. 무선 센서 네트워크 개념도.
Fig. 1. Wireless sensor network concept diagram.

* 책임저자(Corresponding Author)
논문접수: 2008. 10. 29., 수정: 2009. 2. 26., 채택확정: 2009. 5. 20.
문정호, 박래정, 정태윤: 강릉대학교 전자공학과
(itsmoon@kangnung.ac.kr/ljpark@kangnung.ac.kr/tychung@kangnung.ac.kr)
김대일: 강원임베디드SW연구센터(daeilkim@kangnung.ac.kr)
이형봉: 강릉대학교 컴퓨터공학과(hblee@kangnung.ac.kr)
※ 이 논문은 강원임베디드 SW연구센터의 지원에 의한 연구 결과임.

II. BiWSLP: 양방향 통신을 지원하는 시분할 기반 선형 무선 센서 네트워크 프로토콜

이 논문에서 제안하는 무선 펌웨어 업데이트 방법은 양방향 통신을 지원하는 시분할 기반 선형 무선 센서 프로토콜인 BiWSLP (Bidirectional Wireless Sensor Line Protocol)을 가정하고 있다. BiWSLP는 주로 야외에서의 데이터 수집 및 감시를 목적으로 제안된 선형 네트워크용 프로토콜이다[4,5]. 센서 노드를 40 m ~ 50 m 간격으로 선형으로 배치할 경우 BiWSLP 는 약 1 km 거리까지 효율적인 실시간 양방향 통신 수단을 제공한다. 이와 같이 센서 노드들의 배치 영역이 넓은 경우에는 무선 펌웨어 업데이트의 필요성은 더욱 커진다.

BiWSLP에서 각각의 노드는 그림 2와 같이 수신(R), 송신(T), 수신확인(A)이라는 고유의 시간 조각(time slot)을 할당받는다. 한 노드의 송신 구간은 자신의 상위 노드의 수신 구간과 하위 노드의 수신확인 구간과 시간적으로 일치한다. 그러므로 각 노드는 수신 구간 동안 하위 노드로부터 데이터를 전달받은 후 송신 구간 동안 상위 노드로 데이터를 전송하는데 이 때 상위 노드로 보내는 데이터는 자신의 하위 노드에게 하위 노드가 바로 직전에 송신한 데이터에 대한 수신 확인 신호로 인식된다. 이와 같은 과정의 데이터 전송 링크를 이용하여 단말 노드로부터 싱크 노드 방향으로의 상향 링크와 싱크 노드로부터 단말 노드 방향으로의 하향 링크를 형성하여 그림 2의 화살표 방향으로 양방향 통신을 진행한다.

BiWSLP에서 한 상향 링크와 그 다음 상향 링크 사이의 시간을 수퍼프레임(superframe)이라고 정의하는데 수퍼프레임은 BiWSLP에서 센서 노드의 통신 주기가 된다. 수퍼프레임은 실제로 노드가 동작하면서 데이터를 주고 받는 활성 구간(active period)과 노드들의 전력 소비를 최소화하기 위하여 휴면(sleep) 상태를 유지하는 비활성 구간(inactive period)으로 나뉘어진다.

BiWSLP는 비경쟁 기반 무선 MAC인 TDMA를 기반으로 하기 때문에 데이터 전송 지연이 상대적으로 짧고, 전체 수퍼프레임에서 한 노드의 활동 시간이 차지하는 비율, 즉 듀티비(duty cycle)가 매우 낮아 전력 소비를 극소화시킬 수 있다는 장점을 가지고 있다. 뒤에서 설명할 실험에서처럼 각 노드의 활동 시간 (R+T+A)*2=64 ms로 설정하고 수퍼프레임을 10초로 둔다면 듀티비는 약 0.6%에 지나지 않아서 전력 효율이 매우 뛰어나다. BiWSLP에 대한 보다 구체적인 설명은 참고 문헌 [4]와 [5]에서 찾을 수 있다.

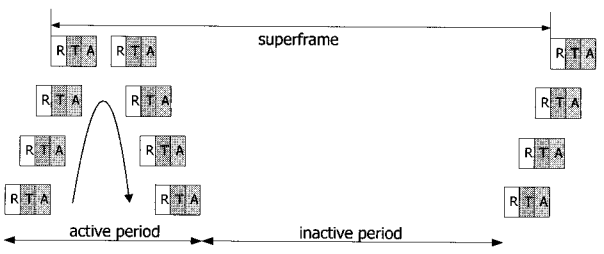


그림 2. BiWSLP의 통신 구조.
Fig. 2. The communication structure of BiWSLP.

III. BiWSLP에 적용 가능한 무선 펌웨어 업데이트 메커니즘

1. 기존의 무선 펌웨어 업데이트 방법들

센서 네트워크에서 무선 펌웨어 업그레이드를 위한 기존의 방법으로 TinyOS의 Deluge와 XNP 및 TDMA를 기반으로 하는 Infuse 등이 소개되어 있다.

XNP는 PC에서 베이스 스테이션(base station)을 통해 하나 또는 그룹을 이루고 있는 노드들에게 펌웨어를 전송하는 방식이다[6,7]. 베이스 스테이션은 펌웨어를 TinyOS 패킷으로 분리하여 노드들에게 전송한다. 노드들은 수신한 펌웨어를 외부 플래시 메모리에 S-Record 형태로 저장하고 누락된 패킷에 대한 재전송을 요구한다. 이와 같은 방법을 반복하여 모든 펌웨어 이미지의 저장이 완료되면 XNP 부트로더를 이용하여 펌웨어 이미지를 응용 프로그램용 플래시 메모리로 옮기고 재부팅 과정을 통해 펌웨어 업데이트를 완료한다. XNP는 오직 싱글홉(single-hop) 통신만을 지원하며 모든 노드들은 베이스 스테이션의 통신 범위 내에 위치해 있어야 하는 단점이 있다.

Deluge는 XNP의 이러한 단점을 보완한 것으로 Mesh형 구성에서 센서 노드들 간의 멀티홉(multi-hop) 통신을 통해서 펌웨어를 업데이트하는 방법이다[8]. 소스 노드가 자신이 가지고 있는 펌웨어의 버전 정보를 다른 노드들에게 알리면 이를 수신한 노드들은 수신한 버전 정보와 자신의 펌웨어 버전을 비교한 후 필요한 경우 새로운 버전의 펌웨어의 전송을 요구한다. 요청을 받은 소스 노드는 펌웨어를 전송하고 펌웨어를 수신하는 노드들은 수신하지 못한 펌웨어 패킷에 대해 재전송을 요청하며 소스 노드는 요청 펌웨어 패킷을 재전송한다. 더 이상 수신 노드들로부터 펌웨어 패킷에 대한 재요청이 없으면 새로운 펌웨어를 수신 받은 센서 노드는 이를 프로그램 메모리로 옮기고 재부팅 과정을 수행한다. 새 펌웨어 전체를 전송 받은 노드는 새로운 소스 노드가 되어 주변의 노드들에게 펌웨어를 전송시키는 방식으로 멀티홉 전송을 한다. 전송되는 펌웨어는 버퍼 크기에 맞게 여러 페이지로 나뉘어지며, 각 페이지는 TinyOS의 네트워크 스택에 저장할 수 있는 크기의 패킷으로 나뉘어진다.

Infuse는 TDMA를 기반으로 하는 무선 펌웨어 업데이트 알고리즘이다[9]. Infuse는 외부 통신망과 연결되어 있는 베이스 스테이션에서 펌웨어 이미지를 수신하면 이미지를 캡슐(capsule)이라 하는 작은 크기의 패킷으로 분리한다. 패킷에는 캡슐의 번호와 이미지의 버전 정보가 포함되어 있다. 이 패킷은 시간적으로 분리되어 있는 송신 슬롯(slot)을 통해 이웃 노드에게 전달되고 이를 수신한 이웃 노드는 패킷에 있는 데이터를 저장한 후 자신의 이웃 노드에게 패킷을 넘겨준다. 누락된 패킷은 Go-back_N 알고리즘을 통해 재전송된다.

2. 활성 구간에서 펌웨어를 전송하는 방법

BiWSLP은 TDMA를 기반으로 동작하기 때문에 위에서 언급한 무선 펌웨어 업데이트 알고리즘인 Infuse를 적용할 수 있다. 이 방법은 펌웨어 코드를 작은 크기의 캡슐로 분리한 후 데이터 패킷에 실어 BiWSLP의 하향 링크를 이용하여 전송하는 것이다. 이 방법은 구현하기 쉽지만 BiWSLP 특성상 모든 펌웨어 조각을 전송하기까지 긴 시간이 소요된다는 단점이 있다. 약 64 KB의 펌웨어를 최대 크기가 100 바이트인

BiWSLP 패킷에 실을 경우 660회의 하향 가상 링크 통신이 요구되고, 이를 논문에서 적용한 10초의 수퍼프레임으로 환산하면 펌웨어 전송에 110분 의 시간이 필요로 한데 이는 현실적이지 않다.

3. 비활성 구간에서 펌웨어를 전송하는 방법

BiWSLP는 비활성 구간이 길기 때문에 이 구간 동안 대용량 데이터를 전송하기 위한 별도의 방법을 고려할 수 있다. 이 방법을 사용하면 1~2 수퍼프레임 이내에 하위 노드에게 펌웨어를 모두 전송할 수 있으므로 펌웨어의 전송 시간을 매우 단축할 수 있다는 장점이 있다[10]. 그리고 실제로 펌웨어를 주고 받는 두 노드만 비활성 구간에서 동작하기 때문에 펌웨어 전송으로 인한 추가 전력 소비가 크지 않다. 이 방법을 구현하려면 펌웨어를 주고 받는 두 노드가 비활성 구간에서 동시에 깨어나 동기를 맞추어 데이터를 주고 받을 수 있도록 하기 위한 새로운 프로토콜이 필요하고 데이터 링크에 다양한 명령 패킷을 통하여 펌웨어 전송 중에 노드들의 상태를 변화시켜 효율적인 업데이트가 이루어지도록 관리해야 한다.

3.1 게이트웨이와 싱크 노드간의 펌웨어 전송

무선 센서 네트워크에서 게이트웨이는 센서 네트워크와 모니터링 서버를 연결해주는 연결 고리 역할을 담당한다. 즉 게이트웨이는 센서 네트워크의 최상위 노드인 싱크 노드로부터 수집된 데이터를 전송 받아 이를 CDMA, 위성망, LAN 등의 기간 상용망을 통해 모니터링 서버에게 전달하는 기능을 수행한다. 게이트웨이와 싱크 노드는 보통 유선으로 연결되어 있는데 BiWSLP에서는 싱크 노드의 활성구간에 맞춰 서로 데이터 통신을 진행한다.

게이트웨이가 싱크 노드로 펌웨어를 전송하는 과정을 요약하면 그림 3과 같다. 게이트웨이가 BiWSLP 활성 구간 동안 펌웨어 전송 시작을 알리는 패킷을 전송한 후 대부분 비활성 구간 동안 펌웨어 전송을 진행한다.

게이트웨이와 싱크 노드간의 펌웨어 전송을 위한 절차는 아래와 같다.

A. 게이트웨이는 모니터링 서버로부터 펌웨어를 전송받은 후 ①Update req을 통해 싱크 노드에게 전달할 펌웨어가 있음을 알려준다.

B. 싱크 노드는 비활성 구간 동안 ②Update ack을 전송하여 ①Update req을 수신하였음을 확인시킨다.

C. 게이트웨이는 ③Firmware pack에 펌웨어 데이터를 실어서 싱크 노드에게 전송한다.

D. 싱크 노드는 ③Firmware pack에 실려있는 펌웨어 데이터를 외부 메모리에 저장하고, ④Packet ack를 전송하여 데이터를 수신하였음을 알린다.

E. 과정 C와 D 를 반복하면서 모든 펌웨어 데이터의 송신을 완료하면 게이트웨이는 ⑤Packet tx end를 보내어 펌웨어 전송이 끝났음을 알린다.

F. 싱크 노드는 ⑤Packet tx end을 수신하면 수신한 펌웨어 데이터 크기와 ①Update req에 실려있던 펌웨어의 크기 정보를 비교한 후 ⑥Complete or Fail을 통하여 펌웨어 수신 성공 여부를 게이트웨이에 알려준다.

G. 게이트웨이는 싱크 노드로부터 수신한 ⑥Complete or Fail 정보를 모니터링 서버에 전송하고 관리자에게 싱크 노드까지의 펌웨어 전송 성공여부를 알려준다.

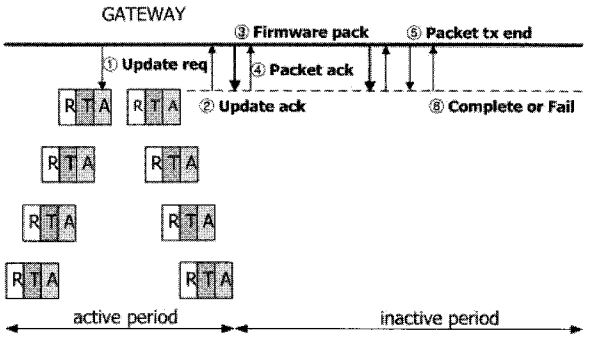


그림 3. 게이트웨이와 싱크 노드간의 펌웨어 전송 프로토콜.

Fig. 3. The protocol for transmitting firmware from the gateway to the sink node.

3.2 노드와 노드간의 무선 펌웨어 전송

노드와 노드 사이의 펌웨어 전송은 비활성 구간 동안 무선으로 진행된다. BiWSLP에서 단말 노드는 주기적으로 자신의 깊이(depth) 정보를 싱크 노드에게 전송한다. 노드들이 선형으로 연결되어 있기 때문에 단말 노드의 깊이는 현재 네트워크를 구성하는 노드들의 수와 같다. 또한 이 정보는 노드들이 펌웨어를 업데이트한 후 새롭게 네트워크를 구성할 때 업데이트 전에 네트워크에 참여했던 노드들의 수와 비교하여 업데이트 성공 여부를 최종 판단하는 근거로 이용된다. 관리자는 깊이 정보를 통해 현 네트워크의 참여하고 있는 노드들의 수와 이탈 노드들의 수를 알 수 있기 때문에 펌웨어 업데이트 시기를 결정할 수 있다. 펌웨어 전송에 관여하는 두 노드는 주기적으로 전송되어 오는 깊이 정보를 통해 모든 노드가 비활성 구간에 진입하는 시간을 계산하고, 펌웨어 전송을 위하여 자신들이 깨어날 시간을 결정한다. 이렇게 하는 이유는 BiWSLP 특성상 각각 노드들의 비활성 구간 진입 시간이 다르므로 펌웨어 전송과 노드들의 통신 구간이 겹치지 않도록 하기 위해서이다. 노드와 노드간 무선 펌웨어 전송은 그림 4와 같이 진행된다.

A. 펌웨어를 가지고 있는 노드는 하향 링크를 통하여 자신의 하위 노드에게 펌웨어 전송을 알리는 ①Firmware tx req를 전송한다. 이 패킷에는 하위 노드의 주소와 전송할 펌웨어의 크기 정보가 포함되어 있다.

B. 하위 노드는 상위 노드가 전송한 ①Firmware tx req를 수신한 후, 상위 노드와 하위 노드는 서로 가지고 있던 깊이

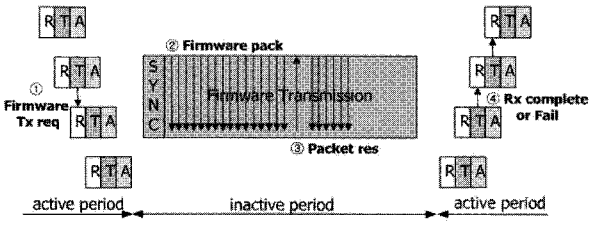


그림 4 노드와 노드간의 펌웨어 전송 프로토콜.

Fig. 4. The protocol for transmitting firmware between sensor nodes.

정보를 이용하여 비활성 구간에서 같은 시간에 깨어난다.

C. 비활성 구간에 같이 깨어난 두 노드는 동기화 구간 동안 펌웨어 전송을 위한 통신 동기를 맞춘다.

D. 송신 노드는 ②Firmware pack 16개를 차례로 전송한다. 수신 노드는 16 개의 패킷을 모두 받았거나, 일정 시간이 경과하면 송신 노드에게 ②의 응답으로 ③Packet res을 보내다. ③Packet res에는 수신되지 않았거나 오류인 패킷 번호를 기록한다. 연속으로 보내는 펌웨어 패킷 수는 사용한 RF 칩의 버퍼 크기에 따라 다르게 설정할 수 있다.

E. 송신 노드는 ③Packet res에 있는 패킷 번호에 해당 되는 패킷을 다음 16 개의 패킷에 포함하여 전송한다. 두 노드는 과정 D와 E 를 반복한다.

F. 펌웨어의 전달이 끝나면 하위 노드는 상향 링크를 통해 펌웨어 ④Rx complete을 싱크 노드에게 전송하고 실패하면 A-E 과정을 반복한다. 실패 횟수가 3회 발생하면 펌웨어 전송 실패로 판단하고 ④Rx fail을 전송한다. 펌웨어의 수신을 완료한 상위 노드들이 ④Rx fail을 수신하면 펌웨어에 관련된 모든 상태 정보와 변수를 초기화 한다.

G. 싱크 노드는 ④Rx complete or fail을 게이트웨이에 전송하고 게이트웨이는 이를 모니터링 서버로 전송한다. 관리자는 이 정보를 통해 현재 펌웨어 업데이트 진행 상황을 알 수 있다.

4. 노드의 재부팅

펌웨어 전송은 네트워크 프로토콜에 의해 이루어지므로 하드웨어와는 거의 무관하지만, 새로운 펌웨어로 업데이트하고 재부팅하는 과정은 하드웨어에 따라 달라진다. 이 논문의 센서 노드는 ATMEL사의 ATMega2560 프로세서를 사용하므로 재부팅 과정은 이 프로세서에 의존적이다. ATMega2560은 내부 256 kb의 플래쉬 메모리와 4 kb의 EEPROM, 8 kb의 SRAM을 가지고 있다[11]. 이와 별도로 각 센서 노드는 128 kb의 SRAM을 추가로 가지고 있는데 이 외부 SRAM은 펌웨어 업데이트를 위하여 수신한 펌웨어를 플래시 메모리의 애플리케이션 프로그램 영역에 기록하기 전에 임시로 저장하기 위한 메모리로 사용된다. 노드의 재부팅은 새로운 펌웨어를 수신한 후 부트로더를 실행하여 펌웨어를 플래쉬 메모리에 기록하고 새로운 펌웨어를 실행하는 것을 의미한다. 재부팅 과정은 그림 5의 방법으로 진행된다.

단말 노드까지 펌웨어 전송이 완료되면 단말 노드는 상향 링크를 통해 싱크 노드에게 ①Reboot req을 전송하여 네트워크의 모든 노드들이 펌웨어 수신을 완료했음을 알린다 또한 이는 게이트웨이를 통해 모니터링 서버로 전송되어 관리자로 하여금 단말 노드까지 펌웨어 전송이 완료되었음을 알리는 정보로 이용된다. ①Reboot req를 수신한 각 노드는 재부팅 준비 상태로 전환한다. ①Reboot req를 수신한 싱크 노드는 하위 노드들에게 재부팅을 지시하는 ②Reboot ack를 전송한 후 재부팅 과정을 수행한다. ②Reboot ack를 수신한 노드들은 자신의 하위 노드들에게 이를 전송하고 재부팅 과정을 수행한다. 만일 노드가 ②Reboot ack를 수신하지 못했다면, 이미 상위 노드는 재부팅 과정의 실행을 위하여 네트워크에서 이탈된 상태이고 이로 인해 노드는 링크 복구 또는 주변 탐색 상태로 전환되기 때문에 재부팅 준비 상태로 전환된 노드들은 스스로 재부팅을 수행한다. 이러한 과정이 하위 노드들에게 연쇄적으로 일어나면서 전체 노드가 재부팅 과정을 수행하게 된다.

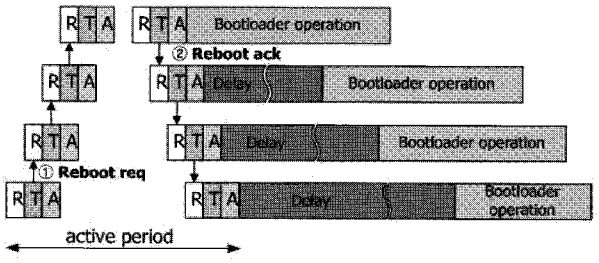


그림 5. 노드의 재부팅 과정.

Fig. 5. The procedure for rebooting sensor nodes.

IV. 실험 및 적용 사례

1. 실험

III 장에서 설명한 무선 펌웨어 업데이트 알고리즘의 구현과 실험을 위해서는 BiWSLP에 관련된 각종 시간 정보를 설정해야 한다. 실험에서 설정한 시간 정보는 아래 표 1과 같다. 수퍼프레임이 10초이므로 각 노드는 10초마다 한번씩 휴면 상태에서 깨어나 약 40 ms 정도 동작을 한 후 다시 휴면 상태로 진입한다.

다음 그림 6은 실험에 사용한 센서 노드의 사진이며 표 2는 이 노드의 사양이다.

실험에서는 다음 그림 7과 같이 BiWSLP가 탑재된 20 개의 센서 노드들을 약 30 cm 간격으로 배치하여 네트워크를 구성하고, 센서 노드의 수를 5 개에서 20 개까지 5 개 단위로 변경하면서 총 50 번에 걸쳐 펌웨어 업데이트를 수행하여 단말 노드까지 새로운 펌웨어가 전송되는데 필요한 시간을 측정하였다.

표 1. BiWSLP 시간 정보 설정.

Table 1. Time information relevant to BiWSLP.

항목	시간 (ms)	항목	시간 (ms)
Rx	10.255	ACK	10.255
Tx	10.255	Guard	2.703
Processing	6.5	Superframe	10,000

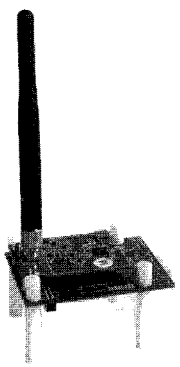


그림 6. 센서 노드 플랫폼.

Fig. 6. The sensor node platform.

표 2. 노드 사양.
Table 2. The specification for a sensor node.

MCU	
모델	ATMega2560
플래쉬 메모리	256 kb
EEPROM	4 kb
외부 메모리	
SRAM	128 kb
Radio	
RF processor	CC2420 (2.4 GHz)
전송속도	250 kbps
기타	
안테나	1/2 λ dipole
전원	리튬 AA 배터리 2개 (3.6V)

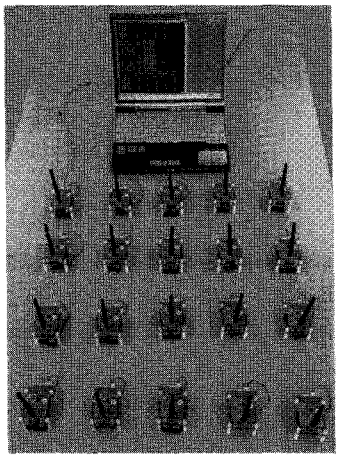


그림 7. 실험 환경.
Fig. 7. Experimental setup.

표 3. 실험 결과.
Table 3. Experimental results.

노드 수	성공률 (%)	소요 시간 (s)
5	100	69.4
10	100	110.3
15	100	162.3
20	100	203.9

표 3은 노드 수에 따른 펌웨어 업데이트 실험의 성공률과 소요시간을 나타낸다. 이러한 실험실 환경의 실험에 있어서는 통신 방해 요인이 많지 않으므로 실패 사례가 없었지만 실제 현장에서는 무선 통신 환경에 따라 전송 중 실패가 일어날 수 있는데, 이런 문제점에 대비하여 통신 오류가 발생하면 무선 펌웨어 업데이트 절차를 여러 번 시도하도록 하였다. 표 3에서 볼 수 있듯이 펌웨어 업데이트 소요 시간은 노드의 수에 비례하여 증가하는데 이는 매우 당연한 결과이다.

그림 7은 실험대 위에 모든 노드들을 위치시킨 매우 단순한 환경에서 펌웨어 업데이트 실험을 실시하였기 때문에 이 결과만으로는 이 논문에서 제안한 방법을 야외에서 동작하는 센서 네트워크에 적용 가능한지 여부를 판단하기에는 무리가 있다. 그렇기 때문에 이 논문에서 제안한 업데이트 방법을 실제로 야외에 설치되어 운영중인 센서 네트워크에 성공적으로 적용하여 그 기능을 확인하고 검증하였는데, 그 중 세가지 사례에 대해서 소개한다.

2. 적용 사례 1

이 예는 센서 네트워크의의 신뢰성 검증과 프로그램 디버깅을 목적으로 강릉대학교 캠퍼스 내에 설치하여 운영 중인 센서 네트워크에 이 논문의 무선 펌웨어 업데이트 방법을 적용한 것이다. 이 선형 네트워크는 50 개의 노드로 구성되어 있으며 약 2 km 거리에 걸쳐 배치되어 있다. 각 노드는 이미지 센서, 조도 센서, 온/습도 센서 등을 가지고 있어서 주기적으로 이미지, 조도, 온/습도, 배터리 전압 등의 데이터를 서버로 송신한다. 그림 9는 노드들의 위치와 번호를 보여준다.

이 예에서는 약 97.2 kb 크기의 펌웨어 업데이트 실험을 모두 5회에 걸쳐 수행하였으며 그 실험 결과를 표 4에 나타내었다.

이 사례는 이 논문의 무선 펌웨어 업데이트 방법이 다수의 노드들로 구성된 선형 센서 네트워크의 펌웨어 업데이트에 적용가능 함을 보여준다. 특이한 점은 실험 도중에 노드 하나를 분실하여 노드의 수가 50개에서 49개로 줄었음에도 불구하고 업데이트 소요 시간이 더 길어진 것을 볼 수 있었다. 그 이유는 펌웨어가 단말 노드까지 도착하는 동안 몇 개의 노드에서 펌웨어의 재전송이 이루어졌기 때문이다. 실험실 내의 테스트에서 나타나지 않았던 통신 환경에 따른 재전송이 야외 네트워크에서 발생할 수 있음을 확인할 수 있었다.

3. 적용 사례 2

이 사례는 새 터널을 건설하기 위한 발파 작업이 바로 옆에 위치한 기존 터널에 미치는 영향을 관측할 목적으로 전북 전주에 위치한 고덕 터널에 설치하여 운영하였던 센서 네트워크에 이 논문의 무선 펌웨어 업데이트 방법을 적용한 것이다. 이 네트워크는 6개의 노드들로 구성되어 있으며, 각 노드는 거동량 센서와 기울기 센서를 사용하여 터널 외벽의 변화

그림 8. 강릉대학교내에 설치되어 있는 노드 위치와 번호.
Fig. 8. Locations and number s of sensor nodes installed at the campus of Kangnung National University.

표 4. 펌웨어 업데이트 실험 결과.
Table 4. The result of the firmware update experiments.

노드 수	횟수	평균소요 시간 (s)
50	3	559
49	2	649

그림 9. 고덕 터널에 센서 노드를 설치하는 모습.
Fig. 9. The scene in which sensor nodes are being installed in the Goduk Tunnel.

표 5. 펌웨어 업데이트 실험 결과.
Table 5. The result of the firmware update experiments.

노드 수	횟수	평균소요 시간 (s)
6	4	218.3

를 모니터링한다. 그림 9는 전주 고덕터널에 센서 노드를 설치하는 모습이다.

이 예는 사례 1에 비해 훨씬 적은 수의 노드들로 구성된 네트워크를 다루고 있다. 그렇지만 관리자와는 매우 떨어진 곳에 위치하고 있으며 또한 열차가 수시로 왕래하는 터널 내부에 있기 때문에 관리자가 직접 접근할 수 없는 환경이므로 펌웨어 업데이트의 신뢰성이 매우 중요하다. 센서 네트워크 운영 중 발견한 펌웨어의 오류를 수정하거나 사용자의 새로운 요구 사항을 반영할 필요가 있을 때에만 펌웨어 업데이트 실험을 하였다. 서버를 통하여 약 90.3 kb 크기의 펌웨어 업데이트를 4회에 걸쳐 실시하였으며 그 결과는 표 5와 같다.

4. 적용 사례 3

이 사례는 도로 상황과 수위를 상시 측정할 목적으로 충남 당진에 위치한 삽교대교에 시범 설치하여 운영중인 센서 네트워크에 무선 펌웨어 업데이트 방법을 적용한 것이다. 이 센서 네트워크는 이미지 센서를 가진 노드와 야간에 수위를 파악하기 위한 LED를 제어하기 위한 노드들로 구성되어 있다. 그림 10은 삽교대교에 설치되어 있는 노드의 위치와 노드의 모습을 보여준다.

이 센서 네트워크의 주된 기능은 삽교대교 아래의 수위 변화를 관측하는 것이다. 어두운 야간에도 수위를 파악할 수 있도록 수위표에 LED를 부착하고 LED의 이미지를 이미지 센서로 포착하여 이를 서버로 전송한다. 상당히 제한적인 전원으로 동작하는 센서 네트워크에서 LED는 전력 소비가 많은 부품이므로 수위를 측정하는 순간에만 LED를 켜고 이를 떨어져 있는 상대 노드에서 이미지 센서로 포착해야 하는데 이 때 이미지 센서와 LED의 동기가 중요하다. 정확한 동기를 위하여 몇 차례 펌웨어 업데이트를 실시하였다. 앞에서 제시한 사례 2와 마찬가지로 이 경우도 네트워크는 관리자와는 먼 거리에 설치되어 있으며 한 번 설치한 후에는 접근

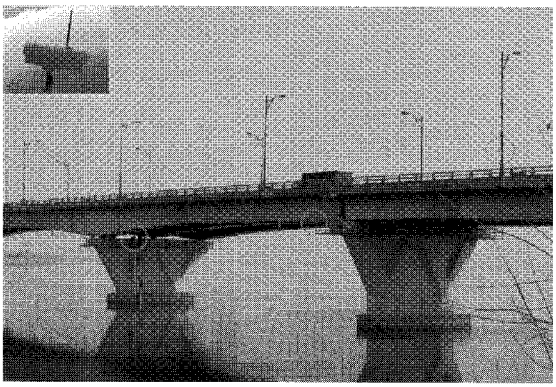


그림 10. 삽교대교에 설치되어 있는 무선 센서 네트워크.
Fig. 10. The wireless sensor network installed at Sapkyo Bridge.

표 6. 펌웨어 업데이트 실험 결과.
Table 6. The firmware update experimental result.

노드 수	횟수	평균소요 시간 (s)
4	40	79.8

하기 어려운 환경이므로 신뢰성 있는 무선 펌웨어 업데이트 방법이 절실히 요구되는 예라고 할 수 있다. 펌웨어의 크기는 약 89.2 kb이며 업데이트 실험 결과는 아래 표 6과 같다.

사례 1에서와 같이 많은 수의 노드가 네트워크를 구성하는 환경과 사례 2, 3과 같이 원거리의 접근이 어려운 곳에 설치되어 관리자가 직접 할 수 없는 환경에서의 시험을 통하여 이 논문에서 제안한 무선 펌웨어 업데이트 방법의 타당성과 신뢰성 및 유용성을 확인할 수 있었다.

V. 소비 전력 분석

TDMA 기반의 프로토콜인 BiWSLP 하에서 펌웨어를 업데이트하는 방법에는 활성 구간에서 Infuse 방식을 적용하는 방식과 이 논문에서 제안한 비활성 구간을 활용하는 방식 두 가지를 생각할 수 있다. 센서 네트워크에서 펌웨어 업데이트 방법의 선택에 영향을 미치는 요소는 펌웨어 업데이트에 소요되는 시간과 이 때의 소비 전력이다. 앞에서 이 논문에서 제안한 펌웨어 전송 방식이 Infuse 방법보다 시간 측면에서 훨씬 효율적이라는 것은 설명하였다. 여기에서는 Infuse 방식을 적용할 때와 이 논문에서 제안한 펌웨어 전송 방식을 적용할 때의 소비 에너지를 비교하고자 한다.

BiWSLP에서 각 노드는 활성 구간 내의 수신 슬롯(Rx slot) 이내라 할지라도 데이터의 수신을 완료하면 즉시 R/F 부를 idle 상태로 변경하면서 처리(processing) 상태로 전환한다. 마찬가지로 송신 슬롯(Tx slot)에도 송신을 종료한 후 R/F 부를 idle 상태로 변경하면서 처리 상태로 전환한다. 이와 같이 BiWSLP는 미리 정해진 시간 조각 내에서도 데이터 양에 따라 적응적으로 상태를 전환함으로써 R/F 부에 의해 필요 없는 전력 소비를 줄인다. 노드의 각 동작 상태에 따른 소비 전류는 표 7과 같다.

센서 노드가 소비하는 에너지 분석의 편의를 위하여 노드가 소비하는 에너지를 소비 전류(mA)와 시간(s)의 곱인 전기량(mAs) 단위로 표시하기로 하자. BiWSLP의 활성 구간에서 전송되는 패킷은 IEEE 802.15.4의 패킷 구조를 따른다. BiWSLP

표 7. 센서 노드의 소비 전류.

Table 7. The current consumption of a sensor node in each operation mode.

상태	소비전류 (mA)
Rx	30
Tx	40
Processing	10
Sleep	0.09

하에서 1 byte를 송신할 때 0.032 ms의 시간이 필요하다. 그리고 BiWSLP에서 수퍼프레임 10초 중 휴면(sleep) 시간은 9.91583초 (10,000-42.085 x 2) 이므로 휴면 시간 동안 노드의 구동에 필요한 전기량은 0.89243 mAs (9.91583 x 0.09) 이다.

수신 데이터의 양에 따른 수신 구간에서의 필요 전기량은

$$P_{Rx} = (T_{R_x} - D_R \times T_{byte}) \cdot A_{proc} + D_R \cdot T_{byte} \cdot A_{Rx} \quad (1)$$

과 같이 계산할 수 있다. 여기에서 T_{R_x} 는 미리 할당된 수신 슬롯의 총 시간이며, D_R 은 수신 데이터 양, T_{byte} 는 1 byte를 수신하는데 필요한 시간, A_{proc} 는 처리 상태에서의 소비 전류, A_{Rx} 는 수신 상태에서의 소비 전류이다. 수신 확인 slot (Ack slot)도 이와 동일하므로 식 (1)을 사용하여 각 시간 조각에서 필요 전기량을 계산할 수 있다.

송신 데이터 양에 따른 송신 슬롯에서의 필요 전기량 또한

$$P_{Tx} = (Tx_t - D_T \times T_{byte}) \cdot A_{proc} + D_T \cdot T_{byte} \cdot A_{Tx} \quad (2)$$

와 같이 계산할 수 있다. 여기서 Tx_t 는 송신 슬롯의 총 시간이며, D_T 는 송신 데이터 양, T_{byte} 는 1 byte를 송신하는데 필요한 시간, A_{Tx} 는 송신 상태에서의 소비 전류이다.

1. Infuse 방식

BiWSLP에서 Infuse 방식을 이용하여 펌웨어를 전송하기 위해서는 활성 구간에서 전송되는 BiWSLP의 패킷에 펌웨어 캡슐을 포함하여 전송해야 한다. 이때 사용되는 펌웨어 패킷 구조는 그림 11과 같다.

또한 펌웨어 캡슐이 단말 노드까지 전송 되었음을 싱크 노드에게 알리기 위해 단말 노드의 깊이 정보와 수신된 캡슐 정보를 포함한 그림 12와 같은 응답 패킷을 싱크 노드에게 전송한다.

BiWSLP에서 노드가 한번의 펌웨어 캡슐을 수신하기 위해서는 활성 구간에서 하향 링크의 수신 구간 동안 상위 노드로부터 펌웨어 캡슐을 수신하고, 송신 구간 동안 하위 노드에게 수신한 펌웨어 캡슐을 송신하고 수신 확인 구간에서 그 하위 노드가 그 다음 하위 노드에게 전송하는 펌웨어 캡슐을

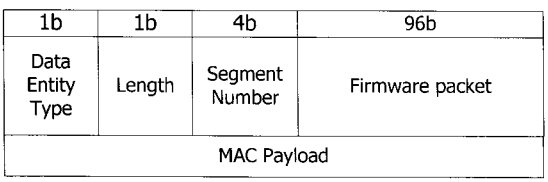


그림 11. 펌웨어 패킷 구조.

Fig. 11. The structure of a firmware packet.

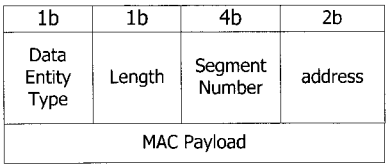


그림 12. 응답 패킷 구조.

Fig. 12. The structure of a response packet.

수신 확인 신호로 수신한다. 또한 상향 링크 동안 하위 노드로부터 펌웨어 캡슐에 대한 응답 패킷을 수신하고, 이를 상위 노드에게 전송하며, 수신 확인 구간에 그 상위 노드가 그 다음 상위 노드에게 송신하는 신호를 수신 확인 신호로 수신한다. 펌웨어를 96 byte의 캡슐로 나누었을 때 BiWSLP에서 하나의 펌웨어 캡슐을 전송하려면 하향 링크로 114 byte의 데이터를 전송한 후 그 데이터를 수신한 하위 노드에서 받은 20 byte 응답을 상향 링크로 전송해야 한다.

하나의 펌웨어 캡슐을 수신하는데 필요한 전기량은 식 (1)과 (2)를 사용하여 구할 수 있다. 식 (1)을 이용하여 하향 링크의 수신 구간 동안 소비하는 전기량을 계산하면 0.21991 mAs이며, 식 (2)를 이용하여 송신 구간 동안 소비하는 전기량을 계산하면 0.23639 mAs이다. 수신 확인 구간은 기본적으로 R/F가 수신 상태이기 때문에 수신 구간의 전기량과 동일하므로 하향 가상 링크에서 소비하는 총 전기량은 0.67621 mAs (0.21991×2+0.23639) 이다. 상향 링크에서 20 byte를 송수신 하는데 소비하는 전기량을 앞서 하향 가상 링크 때와 동일하게 계산하면 0.46565 mAs이다. BiWSLP에서 한 노드가 수퍼프레임 동안 필요로 하는 전기량은 휴면 구간, 하향 링크, 상향 링크에서의 전기량을 모두 합한 2.03429 mAs이다. 펌웨어의 크기가 96,000 byte라고 가정하면 단말 노드까지 펌웨어의 전송을 완료하기 위해서는 이 과정을 모두 1000 번 반복해야 하므로 이 때 소비하는 총 전기량은 2034.26 mAs (2.03429×1000)가 된다.

2. 논문에서 제안한 방식

이 논문에서 제시한 펌웨어 전송 방법은 BiWSLP의 비활성 구간을 이용하기 때문에 한번의 수퍼프레임 동안 펌웨어 수신을 완료할 수 있으며, 그 다음 수퍼프레임 동안 수신한 펌웨어의 송신을 완료할 수 있다. 또한 비활성 구간 동안 통신에 참여하는 두 노드는 동기화 과정을 거친 후 바로 송수신 동작을 진행하기 때문에 송수신 시간 조각의 동기 오차 가능성이 매우 낮다. 따라서 송수신 구간의 동기 보호 시간을 활성 구간에 비해서 느슨하게 적용해도 문제가 발생할 가능성이 낮다는 장점이 있다.

BiWSLP는 비활성 구간 내 펌웨어 전송을 위한 송수신 구간을 6 ms로 설정하였다. 그리고 네트워크를 구성하는 많은 노드 중 특정한 두 노드만 통신에 참여하므로 활성화 구간에서 사용하는 패킷보다 더 간단한 구조의 패킷을 이용한다. 비활성화 구간에서 사용하는 펌웨어 패킷 구조는 그림 13과 같다. 펌웨어를 수신 받는 노드는 16개의 펌웨어 패킷을 수신한 후 응답 패킷을 전송하는데 응답 패킷의 구조는 그림 14와 같다.

비활성 구간의 동기화 구간에서 통신에 참여하는 두 노드는 펌웨어 전송을 위한 동기화를 수행하는데 이를 위해서 동

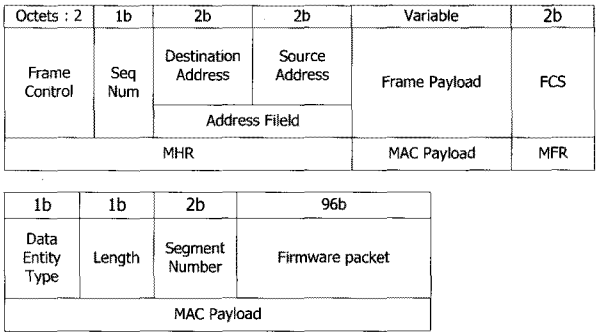


그림 13. 펌웨어 패킷 구조.
Fig. 13. The structure of a firmware packet.

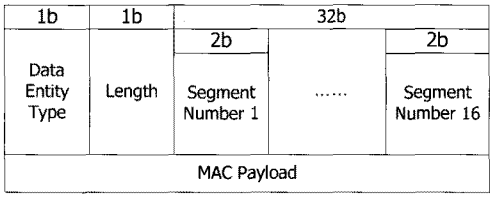


그림 14. 펌웨어 패킷에 대한 응답 패킷 구조.
Fig. 14. The structure of a response packet.

기화를 위한 패킷을 교환해야 한다. 그리고 비활성 구간에서 펌웨어 전달을 하기 전에 펌웨어를 전송할 노드는 활성 구간의 하향 링크를 통해 하위 노드에게 펌웨어의 송신을 알리는 패킷을 전송한다. 또한 하위 노드는 펌웨어 수신을 완료하면 상향 링크를 통하여 수신 성공 여부를 알리는 패킷을 싱크 노드에게 전송한다. 이와 같이 펌웨어 패킷의 전송에 필요한 부가적인 패킷의 송수신에 필요한 에너지도 고려해야 한다.

소비 에너지를 계산하기 위해서는 상위 노드로부터 펌웨어를 수신하는 상황과 수신한 펌웨어를 하위 노드로 전송하는 상황을 모두 고려해야 한다. 비활성 구간에서 96,000 byte의 펌웨어를 전송하기 위해서는 우선 동기를 맞추기 위한 동기 패킷을 활성 구간의 하향 링크를 통해 두 번 전송하고 한 번 수신한다. 또한 1000번의 펌웨어 패킷을 송신해야 하며, 펌웨어 패킷에 대한 응답 패킷을 63번 수신해야 한다. 따라서 전송하는 데이터 총량은 109,026 (109×1000+13×2) byte이며, 수신하는 데이터 양은 2,722 (13+43×63) byte이다.

각 동작 상태에서 필요로 하는 전기량은

$$\begin{aligned} P_{sleep} &= (T_{sleep} - T_{Tx} - T_{Rx}) \cdot A_{sleep} \\ P_{Tx} &= (T_{Tx} - (D_T \cdot T_{byte})) \cdot A_{proc} + D_T \cdot T_{byte} \cdot A_{Tx} \\ P_{Rx} &= (T_{Rx} - (D_R \cdot T_{byte})) \cdot A_{proc} + D_R \cdot T_{byte} \cdot A_{Rx} \\ P_{fimrware} &= P_{sleep} + P_{Tx} + P_{Rx} \end{aligned} \quad (3)$$

와 같이 계산할 수 있다. 여기에서 P_{sleep} 는 비활성 구간의 휴면 상태에서 필요로 하는 전기량, T_{sleep} 은 프로세서의 휴면 시간이다. 그리고 T_{Tx} 는 송신 상태 시간, T_{Rx} 는 수신 상태 시간이다. 식 (3)을 사용하여 비활성 구간 동안 펌웨어 전송에 필요한 전기량을 계산하면 170.68 mAs이다. 마찬가지 방법으로 펌웨어의 수신에 필요한 전기량을 계산하면 136.67 mAs이다. 따라서 펌웨어 전달에 참여하는 한 노드가 비활성 구간 동안 펌웨어를 수신하고 다시 이를 송신하는데 필요한 전기량은 307.35 mAs (170.68+136.67)이다.

펌웨어 전송에 필요한 정보를 수퍼프레임의 활성 구간 동안 하향 링크를 통해 하위 노드에게 전송하는 데이터의 양은 25 byte이며, 상향 링크를 통해 펌웨어 수신 성공 여부를 알리기 위해 전송하는 데이터 양은 19 byte이다. 식 (1)과 식 (2)를 이용하여 전기량을 계산하면 0.94 mAs인데 이 값은 한 노드가 펌웨어 전송을 위해 활성 구간 동안 소비하는 전기량이다. 펌웨어를 수신할 때도 노드는 이와 동일한 일을 하므로 결과적으로 활성화 구간에서 사용하는 총 전기량은 1.88 mAs이다. 따라서 두 번의 수퍼프레임 동안 노드가 펌웨어를 수신하고, 이를 하위 노드에게 송신하는데 필요로 하는 총 전기량은 309.23 mAs (307.35+1.88)가 된다.

이 논문에서 제시한 업데이트 방식과 Infuse 방식을 정확하게 비교하기 위해서는 펌웨어가 단말 노드까지 전송되어 노드의 재부팅이 이루어지기까지 노드가 네트워크를 유지하기 위해 소비하는 전력도 고려해야 한다. BiWSLP는 네트워크를 유지하기 위해서 12 byte의 데이터를 전송한다. 따라서 노드가 한 수퍼프레임 동안 소모하는 전기량은 1.772mAs이다. 네트워크를 유지하기 위하여 한 수퍼프레임동안 노드가 소비하는 전기량은

$$p_{node} = 309.23 + (1.772 \times N_{node}) \quad (4)$$

와 같이 계산할 수 있다. 여기에서 N_{node} 는 BiWSLP 네트워크를 구성하는 총 노드 수이다.

Infuse 방법과 이 논문에서 제안한 방법에 대한 전기량 계산 결과를 보면 이 논문에서 제안한 방법이 전력 관리면에서 훨씬 더 효율적인 것을 알 수 있다. 다만 Infuse 방식은 펌웨어를 업데이트하는데 필요한 시간과 전력이 네트워크를 구성하는 노드의 수와 무관하지만 이 논문의 방법은 업데이트 시간과 소비 전력이 노드의 수에 비례한다. 계산을 해보면 센서 네트워크를 구성하는 노드들의 수가 970개 이상이라면 Infuse 방식을 적용하는 것이 보다 효율적이나 실제로 이렇게 많은 노드가 하나의 네트워크를 구성하는 예는 드물다.

VI. 결론

이 논문에서는 야외 환경 모니터링에 적합하도록 설계된 양방향 통신을 지원하는 시분할 기반 선형 무선 센서 네트워크를 위한 무선 펌웨어 업데이트 방법을 제시하고 이를 실제 구현하여 그 효용성을 검증하였다. 무선 펌웨어 업데이트 기능은 관리자가 센서 네트워크가 설치된 위치와 관계없이 간단하게 네트워크 구성 노드들의 펌웨어를 최적으로 관리할 수 있다는 점에서 센서 네트워크의 유지에 필수적인 기능이라고 할 수 있다. 현재는 센서 노드 간에 1:1 store-and-forward 방식의 통신을 사용하고 있으나 1:N store-and-forward 방식의 통신을 사용한다면 펌웨어 업데이트 시간을 1/3 수준으로 줄일 수 있을 것으로 생각한다. 또한 업데이트 시 누락되었거나 신규 노드가 네트워크에 합류할 때 자신의 펌웨어 버전 정보와 네트워크의 펌웨어 버전 정보를 비교하여 네트워크 노드로부터 펌웨어를 전송 받고 업데이트 후에 합류하는 방법에 대한 연구도 필요하리라 판단된다.

참고문헌

[1] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," *in Proc. INFOCOM 2002*, vol. 3, pp. 1567-1576, Jun. 2002.

[2] T. V. Dam and Koen Langendoen. "An adaptive energy-efficient MAC protocol for wireless sensor networks," *in Proc. SenSys'03*, pp. 171-180, Los Angeles, Nov. 2003.

[3] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," *in Proc. SenSys'04*, pp. 95-107, Baltnore, Nov. 2004.

[4] 이형봉, 박래정, 문정호, 정태윤, "양방향 통신을 지원하는 시분할 기반 무선 센서 네트워크의 구현," 한국정보과학회논문지, vol. 14, no. 4, pp. 341-351, 2008.

[5] 정한수, 문정호, "양방향 데이터통신이 가능한 선형구조를 갖는 무선센서네트워크 및 그 방법," 대한민국 특허청, 국내특허출원, 10-2007-0008935, 2007.

[6] J. Jeng, S. Kim, and A. Broad, Network Reprogramming, University of California at Berkeley, Berkeley, CA, USA, Aug 12, 2003.

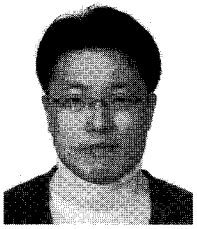
[7] S. Brown and C. J. Sreenan, "Updating software in wireless sensor networks : a survey," Department of Computer Science, University College Cork, Ireland, Technical Report UCC-CS-2006.

[8] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proc. SenSys '04*, Baltimore, USA, pp. 81-94, Nov. 2004.

[9] S. S. Kulkarni and M. Arumugam, "Infuse: A TDMA based data dissemination protocol for sensor networks," *Technical Report MSU-CSE-04-46.* Dept. of Computer Science and Engineering, Michigan State University, Nov. 2004.

[10] 김대일, 정태윤, 이형봉, "무선 센서 네트워크의 노드들에 대한 펌웨어 업데이트 방법," 대한민국 특허청, 국내특허출원, 10-2007-0084768, 2007.

[11] ATmega2560 Datasheet, Atmel Corporation.



문 정 호

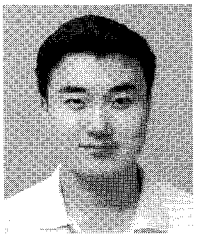
1969년 3월 8일생. 1991년 서울대학교 제어계측공학과 졸업. 한국과학기술원 전기및전자공학과 석사(1993) 및 박사(1998). 1998년~2002년 삼성전자 및 휴맥스 연구원. 2003년부터 강릉대학교 전자공학과에서 근무하고 있으며 현재는 부교수. 관심분야는 디지털 제어, 임베디드시스템, 센서 네트워크.

박 래 정

1991년 서울대학교 전기공학과 졸업. 한국과학기술원 전기및전자공학과 석사(1993) 및 박사(1997). 1997년~2000년 LG 종합기술원 선임연구원. 2000년~현재 강릉대학교 전자공학과 부교수. 관심분야는 기계학습, 최적화, 임베디드 시스템, 센서 네트워크.

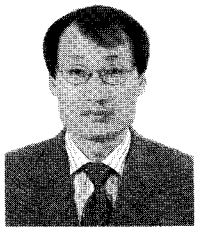
정 태 윤

1987년 연세대학교 전기공학과 학사. 1989년 연세대학교 전기공학과 석사. 2000년 연세대학교 전기컴퓨터공학과 박사. 1989년~1996년 삼성종합기술원. 1996년~2001년 삼성전자 중앙연구소 책임연구원. 2001년~현재 강릉대학교 전자공학과 부교수, 강원 임베디드 소프트웨어 연구센터 센터장. 관심분야는 임베디드 시스템, 센서 네트워크, 영상 부호화.



김 대 일

2001년 강릉대학교 전자공학과 졸업. 강릉대학교 전자공학 석사(2003). 2003년~2006년 이너텍 전임연구원. 2006년부터 강원임베디드SW센터에서 선임 연구원. 관심분야는 임베디드시스템, 하드웨어설계, 센서 네트워크.



이 형 봉

1984년 서울대학교 계산통계학과 학사. 1986년 서울대학교 계산통계학(전산과학)과 석사. 2002년 강원대학교 컴퓨터과학과 박사. 1986년~1993년 LG전자 컴퓨터연구소 선임연구원. 1994년~1999년 한국디지탈(주) 임컨설턴트. 1999년~2003년 호남대학교 조교수. 2004년~현재 강릉대학교 컴퓨터공학과 부교수. 관심분야는 임베디드시스템, 센서 네트워크, 데이터 마이닝 알고리즘.