플래시 메모리를 이용한 TinyOS 기반 센서 노드를 위한 데이터 저장 시스템의 설계 및 구현

한형진, 이기혁, 송준영, 최원철, 손기락
한국외국어대학교 컴퓨터공학과
e-mail:nom96ny@hufs.ac.kr

Design and Implementation of a Data Storage System using Flash Memory for a TinyOS-based Sensor Node

HyungJin Han, KiHyuk Lee, JunYoung Song, WonCul Choi, Kirack Sohn
Dept of Computer and Information Communication Engineering, HanKuk University of Foreign Studies

요 약

본 논문은 무선 센서노드에서 측정되는 데이터들에 대한 저장 및 검색을 효율적으로 하기 위한 플래시 메모리 공간 관리 기법을 제안한다. 플래시 메모리는 외부 충격에 강하고, 비휘발성이며 접근이 빠른 장점이 있지만, 덮어쓰기 및 쓰기 횟수가 제한되는 단점이 있다. 이러한 특성으로 플래시 메모리는 기존의 저장매체와는 다른 관리 방법이 요구되었고 지금까지의 센서노드에서는 플래시 메모리를 사용하지 않았다. 본 논문에서는 센서노드안의 플래시 메모리에서 순차적으로 측정되는 데이터를 관리하기 위해 LFS(Log-Structured File System)방식을 제안한다. 그리고 순차적으로 정렬된 데이터에 효율적인 검색방법을 제시하고, 이를 ZigbeX Mote의 TinyOS안에서 NesC로 구현하였다.

1. 서론

최근 매우 작은 크기의 독립된 무선 센서들이 주위의 온도, 습도 및 조도 등의 데이터를 실시간으로 감지, 관리할 수 있는 센서 네트워크가 주목받고 있다. 센서 네트워크에 쓰이는 무선 센서노드에서 가장 중요한 점은 에너지의 효율적인 관리라고 할 수 있다. 센서노드에서 측정한 데이터를 전송할 때 필요한 에너지는 측정한 데이터를 저장할 때 필요한 에너지보다 100배에 이르기 때문에 보다 많은 데이터의 저장이 요구되고 있다[1]. 지금까지의 센서노드는 저장매체로 자체 ROM을 이용했으나, 용량의 한계가 있다. 그래서 현재 새로운 저장매체로 플래시 메모리(flash memory)가 각광받고 있는 추세이다.

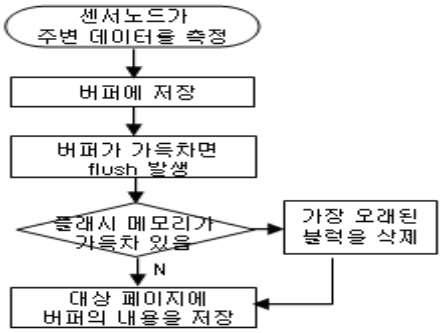
본 연구는 21 세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스 컴퓨팅 및 네트워크 원천기반 기술 개발사업의 지원을 받았음

플래시 메모리는 비휘발성(non-volatile), 빠른 접근속도, 크기가 작고 전력의 소모가 적은 장점이 있다. 이와 더불어 플래시 메모리의 집적도 및 입출력 성능은 계속 개선되고 있어, 플래시 메모리에 데이터를 저장하고, 관리할 수 있는 방법의 중요성이 부각되고 있다. 그러나 플래시 메모리는 덮어쓰기(in-place update)가 불가능하고, erase 횟수가 제한되는 단점이 있다. 이를 고려하여, 본 논문에서는 무선 센서노드에서 플래시 메모리를 이용, 측정되는 데이터들을 저장하고 검색하는 방법을 제시한다. 데이터들의 관리를 위해 LFS(Log-Sturctured File System)방식을 제안하고, 샘플링 시간간격에 의해서 탐색위치를 찾아갈 수 있는 검색방안을 제시한다[2]. 그리고 이를 ZigbeX Mote의 TinyOS안에서 NesC로 구현하였다. TinyOS는 제한된 자원을 사용하는

센서 네트워크 Mote를 위해 설계된 event -driven operation system으로써, TinyOS의 목표는 Task와 Event의 빠른 처리로 에너지 효율을 최대한 높이는 데에 있다.

2. 저장 알고리즘

LFS는 리눅스 기반으로 만들어진 로그구조의 파일시스템으로, 처리량을 최대로 하기 위해 버퍼를 사용하여 누적된 데이터를 한번에 플래시에 쓰게 된다. 본 논문에서는 무선 센서가 일정한 시간간격으로 주변 환경의 데이터를 고정된 크기로 순서대로 측정하므로 LFS방식이 효율적이라고 판단해 플래시 메모리에서 LFS방식으로 구현하였다[3]. 또한 플래시 메모리가 가득 찼을 경우, 가장 먼저 쓰여진 블록을 삭제하는 방식을 취함으로써 일정 수준으로 erase 횟수를 평준화하였다. 본 논문을 통해 구현한 플래시 메모리 데이터 저장 시스템의 동작을 (그림 1)로 나타내었다.



(그림 1) 플래시 메모리에 쓰기 과정

(그림 1)에서 보면, 무선 센서가 데이터를 감지하고, 플래시 메모리에 쓰는 과정은 다음과 같다.

① 무선센서는 일정한 시간간격으로 현재 시간, 온도, 습도의 데이터를 감지한다.

② 측정한 데이터를, 센서의 버퍼에 순차적으로 더해간다.

③ 버퍼가 가득차면, 플래시 메모리의 페이지로 flush연산을 발생한다.

④ 플래시 메모리의 페이지가 가득찬 상태이면, 가장 오래된 블록을 erase한다.

⑤ 대상 블록의 페이지로 버퍼의 내용을 쓴다.

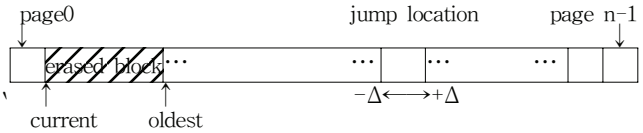
3. 검색 알고리즘

본 논문에서 데이터는 일정한 시간 간격대로 플래시 메모리에 물리적으로 연속되어 저장되어진다. 플래시 메모리는 모두 2047페이지로 구성되며, 각 페이지 당 10개의 데이터들이 들어있다. 데이터의 크기는 23Byte이고 그 중 앞 부분의 20Byte가 TimeStamp이며 나머지는 각각 1Byte씩의 온도, 습도 그리고 조도 값으로 이루어진다. (그림 2)

측정시간(20Byte)	온도(1Byte)	습도(1Byte)	조도(1Byte)

(그림 2) 패킷

이러한 조건을 가진 플래시 메모리 저장 시스템에서 샘플링 시간간격에 의해서 탐색위치를 찾아가는 검색방법을 택하였다. 본 논문에서 구현된 프로그램은 4K의 SRAM안에 가장 오래전에 쓰여진 페이지의 주소와 마지막으로 쓰여진 페이지의 주소를 가지고 있다. 사용자가 콘솔을 통해 원하는 검색시간을 입력하면, 프로그램은 가장 오래된 페이지의 측정시간과 입력된 시간과의 차이를 계산한다. 데이터는 일정한 시간간격으로 저장되므로, 측정시간의 차이로 가장 오래된 페이지에서 찾고자 하는 데이터까지의 거리를 알 수 있다. 그러나 반환된 주소 값의 측정시간이 데이터의 손실 등으로 인하여 찾고자 하는 시간과 다를 수 있다. 그러면 반환된 값의 이전 데이터와 다음 데이터의 값을 살펴, 점차적으로 찾고자하는 데이터로 접근하도록 하였다[4]. 위에 서술한 검색알고리즘을 (그림 3)으로 나타내었다.



(그림 3) 샘플링 시간간격에 의한 검색 알고리즘

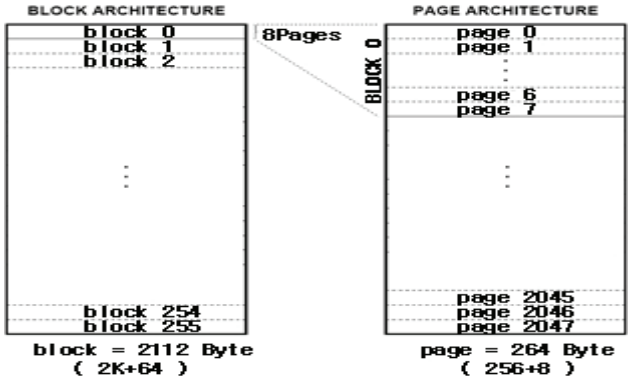
4. 설계 및 구현

본 논문에서는 플래시 메모리를 ZigbeX Mote에 내장된 TinyOS에서 NesC를 이용하여 LFS방식으로 구현하였다. ZigbeX Mote의 하드웨어 스펙은 <표 1>과 같다.

<표 1> ZigbeX Mote의 하드웨어 스펙

구분	세부사양
Computing	- Atmel 8-bit microcontroller(CPU 7.3728Mhz) - 128KB Flash program memory - 4KB EEPROM
Radio Tranceiver	- Choipcon CC2420(IEEE 802.15.4제공) - Radio range: (130m) - Data rate: 240kbits/sec - Frequency range: 2.4GHz (ISM)

ZigbeX Mote는 512KB의 플래시 메모리를 가지고 있고, 두 개의 264Byte의 SRAM 버퍼와 4KB의 EEPROM을 지원한다. 플래시 메모리의 구조는 (그림 4)와 같다.



(그림 4) 메모리 구조

한 block은 8개의 페이지로 이루어지고, 한 페이지는 256Byte의 데이터영역과 8Byte의 예비영역으로 구성된다. 한 페이지에는 한번의 쓰기연산만 가능하며, 새로운 데이터를 쓰기 위해서는 해당 블록을 erase한 후에, 쓸 수 있다. 본 논문에서 LFS구현을 위해 ZigbeX모듈에서 측정하는 데이터는 현재시간(ts), 온도(t), 습도(h)의 데이터이다. (그림 2)

센서가 측정한 23Byte의 데이터를 버퍼에 유지하면서, 새로 측정되는 데이터를 버퍼에 추가적으로 저장한다. 그리고 버퍼가 가득차면 플래시 메모리에 페이지로 flush하게 된다.

센서가 플래시 메모리에 쓰는 과정의 구현은 다음과 같다. 우선 ZigbeX Mote가 측정한 값들을 아래의 Buffer_Read_Byte함수를 이용하여 해당 버퍼에 순서대로 넣게 된다.

```
unsigned char Buffer_Read_Byte (unsigned char BufferNo, unsigned int IntPageAdr)
// BufferNo : Decides usage of either Buffer 1 or 2
// IntPageAdr : Internal page address
```

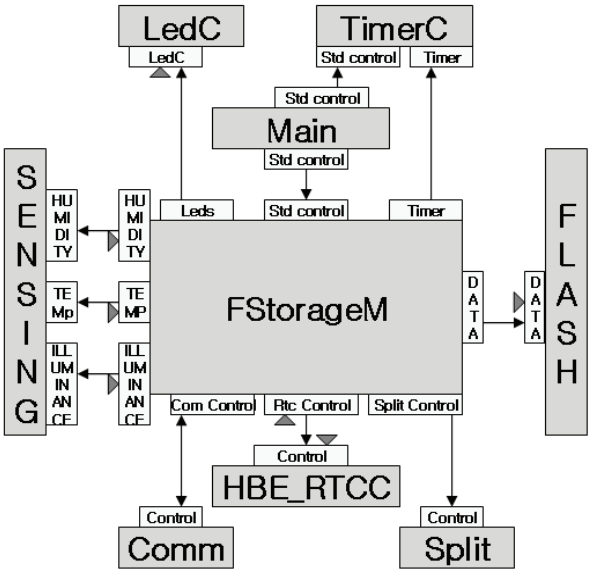
버퍼가 가득차면 Buffer_To_Page함수를 이용하여 버퍼의 자료들을 플래시 메모리의 지정 페이지에 저장한다.

```
void Buffer_To_Page (unsigned char BufferNo, unsigned int PageAdr)
// BufferNo : Decides usage of either Buffer 1 or 2
// PageAdr : Address of flash page to be programmed
```

그리고 사용자가 데이터를 찾을 경우 아래의 Page_To_Buffer함수를 이용하여 플래시메모리의 데이터를 버퍼에 올린 후 검색알고리즘을 통하여 원하는 자료를 찾은 후 콘솔창을 통하여 자료를 보여주게 된다.

```
void Page_To_Buffer (unsigned int PageAdr, unsigned char BufferNo)
// BufferNo : Decides usage of either Buffer1 or 2
// PageAdr : Address of page to be transferred to Buffer
```

아래 (그림 5)는 구현된 FStorageFM의 컴포넌트 구성도와 구현된 nesC코드이다.



(그림 5) 컴포넌트구성도

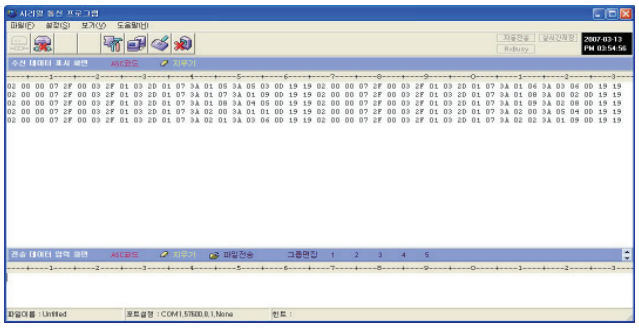
```
module FStorageM
{    provides interface FStorage;
     uses { interface HBE_RTC;
            interface Sensing;
            interface Flash;
            interface Leds;
            interface Timer;
            interface SplitControl;
            interface StdControl as RtcControl;
            interface StdControl as CommControl;
            interface StdControl as SensorControl;
            interface StdControl as FlashControl; }
}

interface FStorage {
   command result_t init();
   command result_t append(unsigned int temp, unsigned int
                           humi, unsigned int Illumi);
   event void appendDone(result_t res);
```

```
    command result_t retrieveRecentStart(unsigned int n);
    command result_t retrieveRecentNext();
    event void retrieveRecentNextDone();
    command result_t retrieveHistoryStart(unsigned int n,
                                          unsigned int timestamp);
    command result_t retrieveHistoryNext();
    event void retrieveHistoryNextDone();
}
```

위의 FStorage 인터페이스는 Split-Phase형태로 구현하였다. retrieveRecentStart는 요청된 retrieveRecentNext Command를 호출하고, 실행이 끝나면 retrieveRecentNextDone 이벤트가 발생한다. 이를 요청된 n번만큼 반복한다. 아래는 Flash메모리에 대한 인터페이스이다.

```
interface Flash {
 command result_t FWrite(unsigned char BufferNo, unsigned
                         int PageAdr);
 event void FWriteDone(result_t success);
 command result_t FRead(unsigned int PageAdr, unsigned char
                        BufferNo);
 event void FReadDone(result_t success);
 command result_t BWrite(unsigned char BufferNo, unsigned
                         int IntPageAdr, unsigned char Data);
 event void BWriteDone(result_t success);
 command result_t BRead(unsigned char BufferNo, unsigned
                        int IntPageAdr, unsigned int No_of_bytes,
                        unsigned char *BufferPtr);
 event void BReadDone(result_t success);
 command result_t Flush(unsigned char BufferNo, unsigned int
                        PageAdr);
 event void FlushDone(result_t success);
}
```

5. 동작 확인



(그림 6) ZigbeX Mote의 데이터를 사용자가 확인

위의 (그림 6)은 ZigbeX Mote가 측정한 값을 플래시 메모리에 저장하고, 저장된 데이터를 페이지단위로 버퍼에 읽어온 후 시리얼 포트로 데이터를 보여주는 화면이다. (그림 6)은 2007년 3월 13일 14:17:02에 측정된 값으로 40초 간격으로 온도, 습도, 조도를 보여주고 있다.

6. 결론 및 향후 계획

본 논문에서는 실시간으로 주변의 온도, 습도의 데이터를 측정하고, 이 데이터들을 ZigbeX Mote를 기반으로 Log-Structured File System으로 구현하였다. 그리고 샘플링 시간간격에 의해서 탐색위치를 찾아갈 수 있는 검색방안을 제시하였다. 본 논문의 방식은 물리적 순서대로 측정되는 데이터에 대한 저장 및 검색에 효율적이다. 또한 무선 센서노드에서 플래시 메모리를 이용, 데이터를 저장함으로써 기존보다 더 많은 양의 데이터를 저장할 수 있게 하였다. 이로써 사용자와 무선 센서노드 사이의 전송횟수를 줄일 수 있게 함으로써, 센서노드의 에너지 효율을 높일 수 있었다. 본 논문의 향후 과제로는 플래시 메모리내의 손상된 블록 체크, 보다 빠른 검색과 일정 범위의 데이터 검색을 위한 검색 알고리즘의 개선이 요구된다.

참고문헌

[1] Gaurav Mathur, Peter Desnoyers, Deepak Ganesan and Prashant Shenoy, “Capsule: An Energy-Optimized Object Storage System for Memory-Constrained Sensor Devices”, Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys), Boulder CO, November 1-3, 2006.

[2] Eran Gal and Sivan Toledo "Algorithms and Data Structures for Flash Memories", ACM Computing Surveys Vol 37, Issue 2, pp138-163, 2005.

[3] M. Rosenblum and J. K. Ousterhout. “The Design and Implementation of a Log-Structured File System", ACM TOCS, 10(1):26 - 52, 1992.

[4] D. ZeFinalipour-Yazti, S. Lin, V. Kalogeraki, D. Gunopulos and W. Najjar “MicroHash: An Efficient Index Structure for Flash-Based Sensor Devices", 4th USENIX Conference on File and Storage Technologies (FAST'2005), San Francisco, CA, December 14-16, pp. 31 44, 2005.