센서 네트워크용 운영체제 SenOS에서 동적 재구성 기능 구현

김도혁°, 김민규, 김태형
한양대학교 컴퓨터공학과
{dohyuk°, mgkim, tkim}@cse.hanyang.ac.kr

Implementing Dynamic Reconfiguration in Sensor Network Operating System SenOS

Do-Hyuk Kim°, Min-Kyu Kim, Tae-Hyung Kim
Dept. of Computer Science & Engineering, Hanyang University

요 약

센서 노드는 정보 수집, 데이터 통신, 협력을 통한 모니터링과 같은 작업들을 수행하기 위해 군사 작전 지역, 산업 시설, 생태 환경 등에 배치된다. 응용 프로그램과 운영체제가 설치된 센서 노드를 센서 필드에 배치하고 나면 센서 노드는 쉽게 수거되기 어렵고 재프로그래밍을 위한 물리적인 연결이 힘들게 되어 응용의 변화에 따른 새로운 응용 프로그램의 설치, 수정과 같은 업데이트가 쉽지 않다. 또한 제한적인 시스템 자원을 가진 센서 노드의 특성상 이러한 재구성 기능은 업데이트에 사용되는 비용이 고려 되어야한다. 본 논문에서는 유한 상태 머신 (finite state machine) 기반의 운영체제인 SenOS에서 응용의 변화에 대처할 수 있도록 동적 재구성 기능이 구현된 형태와 특징을 기술한다.

1. 서 론

센서 네트워크에서 센서 노드는 정보 수집, 데이터 통신, 협력을 통한 모니터링과 같은 작업들을 수행하며 군사 작전 지역, 산업 시설, 생태 환경 등에 배치된다. 이때 사용되는 센서 노드들은 적은 용량의 배터리, 5~50M의 통신 범위를 갖는 무선 모듈, 8~128KB 정도의 프로그램 가능한 메모리, 512B~ 4KB 정도의 RAM과 같이 제한된 시스템 자원을 가지고 동작한다 [1][2]. 이렇게 자원이 제한적인 센서 노드를 이용하여 산불 감시, 오염도 측정, 구조물의 안전도 검사와 같은 작업들을 효과적으로 처리하고 시스템 자원을 관리하기 위해서 센서 노드에는 작은 크기의 운영체제가 탑재되며 센서 노드용 운영체로는 TinyOS, MantisOS, SOS, SenOS 등이 있다[2][3][4][5].

응용 프로그램과 운영체제가 설치된 센서 노드를 센서 필드에 배치하고 나면 센서 노드는 쉽게 수거되기 어렵고 재프로그래밍을 위한 물리적인 연결이 힘들게 되어 새로운 응용 프로그램의 설치, 수정과 같은 업데이트가 쉽지 않다. 이를 해결하기 위해 무선 통신을 이용하여 환경 변화에 따른 센서 노드의 업데이트, 기능 추가, 사용되지 않는 기능의 제거와 같은 동적 재구성 기능을 운영체제에서 지원하는 것이 고려되어야 한다 [4][6].

현재 여러 센서 노드용 운영체제 중 가상 머신을 TinyOS에 적용한 Mate, 바이너리 모듈 업데이트를 이용한 SOS, 교체 가능한 상태 전이 테이블을 이용한 SenOS가 이러한 재구성 기능을 지원하는 시스템이다.

본 논문에서는 유한 상태 머신 기반 운영체제인 SenOS의 구조에 대해 기술하고 기존 시스템과 비교하여 동적 재구성 기능의 요구 조건들을 어떻게 해결하는지 기술하였다.

2. 관련 연구

운영체제와 응용 프로그램으로 구성된 시스템 이미지는 소스 코드 작성, 컴파일, 링크 과정을 거쳐 생성되어 유선 통신을 통해 센서 노드에 프로그램된다. 이렇게 프로그램된 센서 노드가 필드에 배치되고 나면 특별한 기능이 제공되지 않는 이상 환경 변화와 목적에 따라 변할 수 있는 응용 프로그램의 업데이트가 쉽지 않게 된다.

에너지가 제한적이고 개별적인 자원을 가진 수백이나 수천의 노드들로 이루어진 무선 센서 네트워크 환경에서 동적 재구성은 센서 노드가 필드에 배치되고 초기화된 이후 각각의 노드들에 설치된 소프트웨어를 시스템 운영 중에도 수정할 수 있는 기능을 말한다. 이를 이용하면 노드들이 필드에 배치된 이후에도 무선 통신을 이용하여 새로운 응용 프로그램을 추가하거나 시스템에 설치된 응용 프로그램의 변경, 오랫동안 사용되지 않는 소프트웨어 모듈을 제거할 수 있다. 하지만, 제한적인 시스템 자원을 가진 센서 노드의 특성상 이러한 재구성 기능은 업데이트에 따른 CPU 사용 비용, 무선 통신의 데이터 전송 비용, ROM과 RAM을 사용한 비용 등을 고려하여야 하며 이는 전력

공급이 제한적인 센서 노드의 생명 주기와도 밀접한 관련이 있다 [1].

센서 노드의 자원과 태스크를 효과적으로 관리하기 위해 운영체제를 사용하게 되는데 대표적인 센서 노드용 운영체제로는 TinyOS가 있다 [2]. TinyOS는 센서 노드에 최적화시킨 아주 작은 운영체제이지만 동적 재구성 기능을 지원하지는 않는다. 노드에 프로그램된 TinyOS 시스템은 컴파일 시점에 정적으로 링크된 바이너리 이미지이다. 그래서 변경된 응용 프로그램은 재컴파일 과정을 통해 새로운 이미지를 생성하고 센서 노드에 유선 통신을 이용하여 프로그램하여야 하므로 실행 중인 시스템을 동적으로 재구성하는 것이 쉽지 않다.

이러한 TinyOS에 동적 재구성 기능을 지원하기 위해서 무선 통신을 통해 시스템을 업데이트 할 수 있는 XNP를 사용하기도 한다 [7]. XNP는 새롭게 생성된 시스템 이미지를 센서 노드로 전송하고 외부 플래시 메모리에 저장한다. 외부 플래시 메모리에 저장된 시스템 이미지는 마이크로프로세서의 In-System Programming 기능을 사용하여 센서 노드의 메인 메모리로 프로그램되고 재부팅을 거치면 업데이트가 완료된다. 하지만 업데이트에는 컴파일 시간에 정적으로 링크되어 생성된 새로운 TinyOS의 시스템 이미지가 사용되고 전체 시스템 이미지를 전송해야 하므로 코드 업데이트 비용과 통신 비용이 높아진다 [5].

코드 업데이트 비용과 통신 비용을 줄이기 위해 가상 머신을 시스템에 적용하여 동적 재구성 기능을 제공하는 Mate가 있다 [6]. Mate는 TinyOS 상에서 동작하며 응용 프로그램은 가상 머신에서 정의한 명령어들을 이용하여 작성한 스크립트 형태이며 스크립트는 바이트 코드 인터프리터에 의해 해석된다. 프로그램 업데이트 시 캡슐화된 프로그램을 전송함으로써 코드 업데이트 비용과 통신 비용을 낮추었지만 프로그램 작성이 어렵고 캡슐화된 프로그램 계산에 오버헤드를 가지게 된다. 그리고 노드마다 가상 머신이 설치되어야 하며 가상 머신을 위한 시스템 자원이 추가로 요구된다 [6].

동적 재구성 기능을 지원하는 또 다른 센서 노드용 운영체제에는 SOS가 있다 [4]. SOS에서 응용 프로그램은 비동기적인 메시지나 함수 호출을 통해 서로 교신하는 하나나 그 이상의 모듈들의 조합이된다. 동적으로 로드 할 수 있는 바이너리 모듈을 지원하여 실행중인 커널로 바이너리 모듈들을 삽입, 제거할 수 있다. 모듈 업데이트는 네트워크에 새로운 모듈의 메시지를 듣고 있는 콜백 모듈에 의해 시작된다. 콜백 모듈은 메시지를 통해 모듈의 버전, 필요한 모듈인지, 모듈을 위한 프로그램 메모리가 충분하지를 검사한다. 만약 위의 조건 중 두 가지 조건을 만족하게 되면 무선 통신을 통해 모듈 다운로드를 시작하고 패킷 헤더에 있는 메타데이터를 확인한다. 헤더에는 모듈에 대한 유일한 식별자, 필요한 메모리 크기, 새로운 모듈 버전을 구별하기 위해 사용되는 버전 정보가 등록되어 있다. SOS 커널은 모듈 추가시 모듈에 대한 메모리 할당이 가능하지 않다는 것을 발견하면 즉시 취소되고 모듈에 대한 init 메시지를 스케쥴링함으로써 모듈의 핸들러를 호출하여 설치를 완료한다. 이러한 모듈 업데이트를 통한 SOS의 동적 재구성 기능은 Mate의 가상머신 보다 CPU 오버헤드, 점유하는 메모리의 용량이 작지만 코드 업데이트 비용은 높다 [4].

유한 상태 머신 (finite state machine) 기반의 운영체제인 SenOS는 동적 재구성 기능을 지원하기 위해 교체 가능한 상태 전이 테이블을 이용한다. 상태 전이 테이블은 센서 노드가 수행하는 태스크에 관한 내용이 기술되어 있다. 상태 전이 테이블은 시스템 실행 중에 업데이트가 가능하며 상태 전이 테이블의 간결한 구조와 관리 메커니즘을 이용하여 코드 업데이트 비용과 CPU 오버헤드, 시스템 자원의 사용 비용을 낮추고자 하였다 [5].

3. 동적 재구성 기능을 지원하는 상태 기반 운영체제

센서 네트워크 운영체제로 제안된 SenOS는 이벤트 구동 방식으로 동작하고 유한 상태 머신 기반의 실행 모델을 이용하여 설계되었다 [5]. 센서 네트워크 환경에 맞는 운영체제를 구현하기 위해서는 제한된 시스템 자원에 적합한 작은 크기의 운영체제가 되도록 해야한다 [1]. 이를 위해 센서 응용에 따라 시스템을 재구성 가능한 형태가 되도록 구현 하였다. SenOS의 재구성 기능은 System-level, OS-level reconfiguration과 같은 정적인 재구성 기능과 응용 프로그램에 해당하는 상태 전이 테이블과 콜백 함수를 응용의 변화에 대처하여 변경할 수 있는 동적 재구성 기능으로 나눌 수 있다.

3.1 SenOS의 정적 재구성

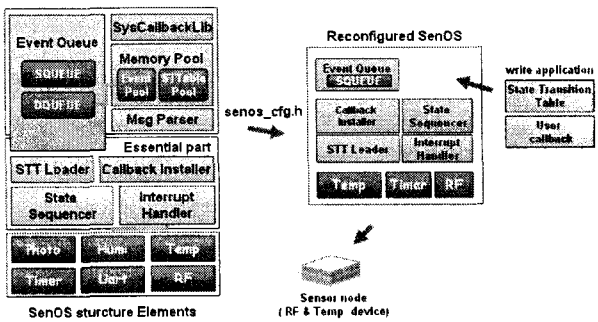

그림 1. SenOS의 시스템 구조와 정적 재구성

그림 1에서와 같이 센서 노드마다 가지고 있는 하드웨어 장치가 틀리고, 응용에 따라 필요한 모듈이 다를 수 있기 때문에 SenOS는 다음과 같은 재구성 기능을 제공하고 이를 통해 시스템 자원을 절약할 수 있다.

System-level reconfiguration을 지원하여 응용 프로그램 개발 시 필요한 하드웨어 모듈만을 선택할 수 있으며 OS-level reconfiguration을 통해 SenOS에서 제공하는 커널 기능 중 응용 프로그램의 특성에 따라 필요한 부분을 선택하여 시스템을 구성할 수 있다. 위의 System-level, OS-level reconfiguration 외에도 SenOS의 응용 프로그램에 해당하는 상태 전이 테이블을 교체하거나 추가하는 기능을 제공함으로써 동적으로 응용 프로그램을 재구성하는 것이 가능하다.

3.2 SenOS의 동적 재구성

SenOS에서 응용 프로그램 개발은 센서 노드의 태스크를 상태 모델링하여 상태 전이 테이블을 생성하고 필요한 콜백 함수를 선택하는 과정으로 진행된다. 이때 생성되는 상태 전이 테이블과 콜백 함수는 센서 노드의 기능을 나타내며 응용의 변화에 따라 시스템에 이를 반영할 수 있는 동적 재구성 기능과도 밀접한 관련이 있다.

SenOS는 이러한 상태 전이 테이블의 변경과 콜백 함수의 교체를 이용한 2단계의 동적 재구성이 가능하며 이를 통해 업데이트 비용을 줄일 수 있다.

3.2.1 상태 전이 테이블 교체를 이용한 동적 재구성

상태 전이 테이블은 동적 재구성 기능을 제공하기 위해 교체 가능하게 설계되었다. 시스템 구성 설정을 통해 동적 재구성 기능을 활성화 시키면 새로운 상태 전이 테이블의 등록과 관리에 필요한 메모리 공간과 테이블 관리 함수들이 활성화되어 동적 재구성에 이용되며, 그림2는 상태 전이 테이블이 시스템에 등록되는 과정을 나타낸다.

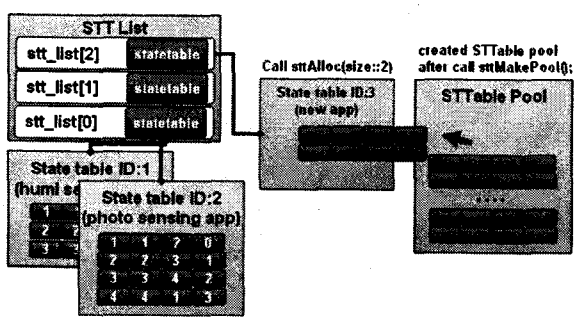

그림 2. 상태 전이 테이블의 관리

응용 프로그램의 업데이트나 새로운 응용 프로그램의 설치는 콜백 함수가 시스템에 등록되어 있는 경우 간단하게 새로운 상태 전이 테이블을 등록하는 것으로 동적 재구성 기능을 수행할 수 있다. 타이머 이벤트를 이용하여 데이터를 수집하고 전송하는 응용 프로그램이 설치되어 필드에 배치되어 있던 센서 노드들이 싱크 노드에서 전달된 명령을 통해 수집한 데이터를 저장하고 저장된 데이터를 전송하는 새로운 응용 프로그램으로 변경하고자 할때, 상태 전이 테이블 교체를 통해 동적 재구성되는 과정은 다음과 같다.

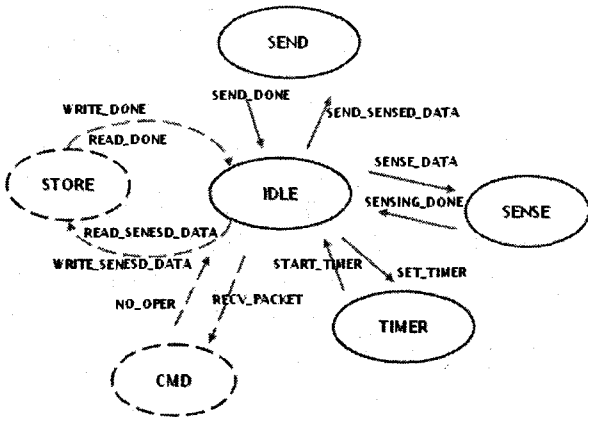

그림 3. 변경된 센서 노드 응용 프로그램의 상태 모델링

Current State	Input event	Destination State	Action
Idle	sense data	Sense	select sensor, get sensed data
Sense	sensing complete	Idle	save sensed data to buffer
Idle	send sensed data	Send	send data packet
Send	send complete	Idle	do nothing
Idle	set timer	Timer	set timer period
Timer	start timer	Idle	start timer
Idle	**read sensed data**	**Store**	**read sensed data to external flash**
Store	**reading complete**	**Idle**	**make packet for send**
Idle	**write sensed data**	**Store**	**write sensed data to external flash**
Store	**write complete**	**Idle**	**do nothing**
Idle	**recv packet**	**Cmd**	**check recv packet**
Cmd	**no oper**	**Idle**	**do nothing**

그림 4. 변경된 상태 전이 테이블

우선 상태 모델링을 통해 변경후 수행할 시스템의 동작을 나타내고 상태 전이 테이블을 생성한다. 그림 3과 4는 앞서 설명한 응용 프로그램의 상태 모델링을 나타내며 새로운 상태가 추가되고 상태 전이 테이블에 반영된 것을 볼 수 있다.

위의 응용 프로그램의 경우 변경전 일정한 주기의 타이머 이벤트를 이용하여 센싱과 데이터 전송을 수행했던것과 달리 새롭게 변경된 응용 프로그램의 경우 일정한 주기로 정해진 횟수만큼의 센싱작업을 통해 평균값을 전송하므로 이러한 동작을 제어할 수 있는 데이터가 추가로 요구된다. 그래서 새로운 상태 전이 테이블을 전송할 때 시스템 초기화 정보를 포함하여 보낼 수 있게 하였다. 새로운 상태 전이 테이블의 크기와 테이블의 각 항목 값을 저장할 공간외에도 상태 전이 테이블 교체시 초기화할 시스템 자

원과 이때 적용할 주기와 샘플링 값들을 updateSysRes 구조체로 나타낼 수 있고 필요에 따라 확장 시킬 수 있다. 변경된 상태 전이 테이블과 시스템 초기화 정보를 패킷으로 변환하고 싱크 노드에서 상태 전이 테이블 교체 신호 명령과 함께 이웃 노드로 전송한다. 패킷을 수신한 노드는 테이블 교체 신호 명령이 있는지 확인하고 명령이 있으면 상태 전이 테이블 교체 이벤트를 발생하게 된다.

call eventCreate(STT_CHANGE_EVNT);

상태 정렬기에서 처리할 이벤트가 STT_CHANGE_EVNT이면 하드웨어 인터럽트를 비활성화 시켜 인터럽트에 의한 잘못된 연산을 방지한다. 그 후 상태 전이 테이블 교체 실행을 위한 함수가 호출된다.

call INT8 senosChangeSTT(void);

senosChangeSTT함수에서는 수신한 패킷의 데이터 필드에서 새로운 상태 전이 테이블의 크기와 테이블 정보를 추출하고 상태 전이 테이블 풀에서 메모리를 할당 받아 새로운 상태 전이 테이블 값을 저장하고

*INT16 senosSTTLoader(INT16 size, STT *stt);*

를 호출하여 새로운 상태 전이 테이블을 시스템에 등록한 후 초기화 시킨다. 새로운 상태 전이 테이블이 등록된 시스템은 이전과 달리 다른 노드로부터 명령이 도착해야 센싱 작업을 수행하게 되며 외부 플래시 메모리에 데이터를 기록하고 기록된 정보를 읽어 전송하는 기능이 추가되었다. 위와 같이 응용 프로그램 운영에 필요한 콜백 함수가 센서 노드에 있는 경우 상태 전이 테이블 교체를 통해서 새로운 응용 프로그램으로 재구성할 수 있으며 적은 데이터 통신 비용과 코드 업데이트 비용으로도 가능하다.

3.2.2 콜백 함수 교체를 이용한 동적 재구성

센서 노드는 정보 수집, 데이터 전송, 데이터 저장과 같은 여러 가지 태스크들을 처리하지만 센서 노드의 특성상 수행되는 기능은 한정된다. SenOS에서는 이러한 센서 노드의 특징을 반영하여 수행해야 할 여러 태스크를 분할하여 독립된 하나의 태스크로 나타내고 콜백 함수를 통해 처리하게 하였으며 여러 시스템 콜백 함수들을 제공하고 있다. 응용 프로그램 개발은 센서 노드의 태스크를 상태 모델링하여 상태 전이 테이블을 생성하고 필요한 콜백 함수를 선택하여 테이블에 콜백 인덱스를 등록하는 과정으로 진행된다. 이때 선택되는 콜백 함수가 독립된 태스크로 구성되어 있으면 콜백 함수의 호출만으로 해당 태스크를 처리 할수 있게 되어 상태 전이 테이블을 통한 응용 개발시 사용하기 편리하다는 장점이 있다.

콜백 함수는 조도나 온도 측정, 무선 데이터 전송, 배터리 체크를 수행하는 드라이버 함수와 라우팅 경로 계산, 측정된 센서 값의 변환, C 표준 라이브러리와 같은 함수들을 이용하여 구성하며 형태는 다음과 같다.

*void callbackLibraryProtoType(void *param);*

응용에 사용되는 여러 콜백 함수들의 일괄적인 관리를 위해 리턴 타입과 함수 인자형은 같다. 타이머나 센서, 무선 데이터 전송을 담당하는 콜백 함수는 주기의 변화, 동작하는 센서의 선택, 전송할 데이터와 같은 가변적인 데이터 처리를 위해 함수 인자를 사용할 때가 있는데 이벤트 발생시 콜백 함수에서 사용할 데이터를 등록하여 함수 인자로 넘겨 주거나 전역 변수를 이용해서 처리할 수 있다.

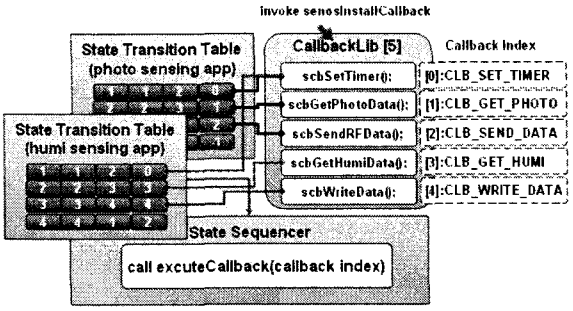

그림 5. 상태 전이 테이블과 콜백 함수의 관계

그림 5는 상태 전이 테이블과 콜백 함수의 관계를 보여준다. 조도를 측정하고 데이터를 전송하는 응용 프로그램은 SenOS에서 제공하는 시스템 콜백 함수 scbGetPhotoData, scbSendRFData, scbSetTimer를 사용하고 상태 전이 테이블에 등록된 콜백 인덱스로 접근하고 있는 것을 볼 수 있다.

응용 프로그램의 업데이트나 새로운 응용 프로그램의 설치는 콜백 함수가 시스템에 등록 되어 있는 경우 간단하게 새로운 상태 전이 테이블을 등록하고 응용에서 처리할 태스크를 담당하는 콜백 함수를 선택함으로써 처리될 수 있지만 시스템에 없는 경우 새로운 콜백 함수를 전송하여 추가해야 한다.

4. 결론

응용 프로그램과 운영체제가 설치된 센서 노드를 센서 필드에 배치하고 나면 센서 노드는 쉽게 수거되기 어렵고 재프로그래밍을 위한 물리적인 연결이 힘들게 되어 응용의 변화에 따른 새로운 응용 프로그램의 설치, 수정과 같은 업데이트가 쉽지 않다. 제한적인 시스템 자원을 가진 센서 노드의 특성상 이러한 재구성 기능은 업데이트에 사용되는 비용이 고려되어야 한다.

동적 재구성을 위해 XNP와 같이 전체 시스템 이미지를 전송하여 업데이트하는 방법이 있지만 코드 업데이트와 무선 통신 비용이 너무 크다는 단점이 있다. TinyOS에 가

상머신을 추가한 Mate는 코드 업데이트 비용과 데이터 전송 비용은 작지만 동작속도상 현저한 성능저하를 보이게 되며 노드마다 가상 머신 운영을 위한 추가적인 비용이 들게된다. SOS는 바이너리 모듈을 이용하여 동적 재구성을 지원하며 Mate의 가상 머신을 사용하여 발생되는 CPU 오버헤드, 메모리 용량 등의 시스템 자원 소모는 적지만, 바이너리 모듈이 업데이트에 사용되므로 Mate 보다 코드 업데이트 비용과 데이터 전송에 드는 비용이 상승하게 된다.

SenOS의 응용 프로그램은 상태 전이 테이블과 콜백 함수에 해당하며 상태 전이 테이블의 교체와 콜백 함수의 변경을 이용하여 동적 재구성 기능을 실행한다. 센서 노드의 응용 프로그램 변경시 필요한 콜백 함수가 없는 경우 새로운 콜백 함수를 노드로 내려보내야 하기 때문에 업데이트 비용 절감에 대한 기대치는 낮아 지겠지만 콜백 함수가 시스템에 있는 경우 상태 전이 테이블 교체만으로도 업데이트가 가능하게 되어 업데이트 비용이 SOS의 모듈 업데이트 비용보다 작아 질 수 있다.

향후 콜백 함수 업데이트 기능을 보완하여 SenOS의 동적 재구성 기능의 효율성과 업데이트 비용을 Mate, SOS와 비교, 평가할 예정이다.

참고 문헌

[1] I.F.Akyildiz, W. Su et al., " A Survey on Sensor Networks" , IEEE Communications magazine, August 2002.

[2] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, " System Architecture Directions for Networked Sensors" International Conference on Architectural Support for Programming Languages and Operating Systems (2000).

[3] H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker, J. Deng, R. " Han MANTIS: System Support for Multimodal Networks of In-situ Sensors" Appeared in 2nd ACM International Workshop on Wireless Sensor Networks and Applications (WSNA) 2003.

[4] Chih-Chieh Han, Ram Kumar Rengaswamy, Roy Shea, Eddie Kohler, and Mani Srivastava. Sos: A dynamic operating system for sensor networks. In MobiSYS ' 05: Proceedings of the 3rd international conference on Mobile systems, applications, and services. ACM Press, 2005.

[5] T.-H. Kim : Seongsoo Hong, "State machine based operating system architecture for wireless sensor networks", *Lecture Notes in Computer Science*, vol. 3320 no. pp.803-803, Dec. 2004

[6] P.Levis and D.Culler. Mate: A tiny virtual machine for sensor networks. In *International Conference on Architectural Support for Programming-Languages and Pperating Systems, San Jose, CA, USA*, Oct. 2002.

[7] Crossbow Technology, Inc. Mote In-Network Programming User Reference, 2003