센서 네트워크를 위한 자바 가상 기계

A Java Virtual Machine for Sensor Networks

김 성 우*, 이 종 민, 이 중 화, 신 진 호
(Seong-Woo Kim, Jong-Min Lee, Jung-Hwa Lee, and Jin-Ho Shin)

Abstract : Sensor network consists of a large number of sensor node distributed in the environment being sensed and controlled. The resource-constrained sensor nodes tend to have various and heterogeneous architecture. Thus, it is important to make its software environment platform-independent and reprogrammable. In this paper, we present BeeVM, a Java operating system designed for sensor networks. BeeVM offers a platform-independent Java programming environment with its efficiently executable file format and a set of class APIs for basic operating functions, sensing and wireless networking. BeeVM's high-level native interface and layered network subsystem allow complex program for sensor network to be short and readable. Our platform has been ported on two currently popular hardware platforms and we show its effectiveness through the evaluation of a simple application.

Keywords : embedded operating system, Java virtual machine, sensor network

I. 서론

근래에 유비쿼터스 사회를 실현하기 위한 기반 기술로서 무선 센서 네트워크에 대한 많은 연구가 이루어지고 있다. 무선 센서 네트워크는 센서, 무선 통신, 저전력 하드웨어, 컴퓨터 기술이 집적된 값싼 무선 센서 노드들로 구성된다. 이러한 센서 네트워크는 기후, 동식물, 차량, 인공 구조물을 비롯한 환경 전반에 대하여 감시하고, 제품 및 물류 정보를 수집하여 효율적으로 관리할 수 있도록 해 준다.

무선 센서 네트워크에 사용되는 센서 노드는 8 또는 16 비트 마이크로컨트롤러, 100K바이트의 프로그램 메모리, 20K바이트 이하의 램을 탑재한 소형 컴퓨터를 사용하며 경량의 운영체제를 필요로 한다. 센서 네트워크 운영체제는 초기에는 이러한 제한된 자원을 효율적으로 관리하면서 저전력 무선 통신과 감지에 관한 프로그래밍이 가능하도록 설계하는 데 중점을 두었다. 하지만, TinyOS와 같은 대표적인 센서 네트워크는 응용 프로그램이 조금만 수정되어도 전체 시스템 이미지를 다시 만들어야 하며, 플랫폼에 의존적인 실행 코드로 구현되어 있어 이기종의 하드웨어 플랫폼 간에 전송이나 갱신이 불가능하다는 치명적인 단점을 가진다.

반면에, 가상 기계 방식으로 운영체제를 구현하면 일반적인 운영체제에 비해 몇 가지 장점을 가진다. 먼저, 응용 프로그램이 센서 네트워크 하드웨어 플랫폼의 종류나 구조에 관계없이 투명하게 실행될 수 있다[1] 따라서, 특정한 하드웨어를 위해 응용 프로그램을 재작성하거나 재컴파일할 필요가 없으며, 이기종의 센서 네트워크 노드들이 배치된 환경에서 특정한 프로그램 코드를 전송하여 분산 실행할 수 있게 된다[2]. 더구나, 응용 프로그램의 전체 코드가 일반적으로 간결하며, 소프트웨어 추상화와 가상화의 수준을 쉽게 확장할 수 있다. 이러한 가상 기계에서 범용의 소프트웨어 개발을 돕도록 만든 대표적인 고급 언어로 자바(java)가 있다. 하지만, 동적인 클래스 적재기, 쓰레기 수집기, 클래스 검증기 등을 포함하는 자바와 같은 고급 실행 환경을 제한적인 메모리를 사용하는 센서 네트워크에 적용하려면, 자바 가상 기계에서 불필요한 명령어나 기능을 제거하고, 반대로 타이머, 센서와 무선 통신 장치 등과 같은 각종 장치와 기능을 모듈화하여 센서 네트워크 환경에 최적화할 필요가 있다. 현재까지 많은 가상 기계 방식의 센서 네트워크 운영체제가 개발되고 있지만, 특정한 목적을 위해 구성된 명령어 집합을 제공하거나 표준적인 센서 노드보다 큰 하드웨어에 적합한 자바 가상 기계가 개발되고 있는 실정이다.

본 논문에서는 센서 네트워크를 위한 임베디드 자바 운영 체제로서 BeeVM(emBEddEd Virtual Machine)을 설계하고 구현하였다. BeeVM은 자바 가상 기계와 자바 응용 프로그램을 타겟 플랫폼에 따로 적재하여 실행하므로 필요할 때마다 응용 프로그램을 교체할 수 있으며, 메모리 효율적으로 설계된 자바 실행 파일 형식을 사용한다. BeeVM의 가상 기계 해석기 부분은 레고 마인드스톰 등에서 이미 검증된 leJOS 자바 가상 기계 [3]를 바탕으로 제작되었으며, 적재되는 메모리 종류에 관계없이 응용 프로그램을 구동시킬 수 있다. BeeVM은 다중 쓰레드가 가능한 기본적인 자바 실행 환경을 제공하며 센서 네트워크 플랫폼의 하드웨어 기능을 활용하고 네트워크 계층 구조에 최적화된 자바 클래스 API를 제공한다. 또한, BeeVM은 현재 ATmega128과 MSP430 마이크로컨트롤러를 사용하는 대표적인 센서 네트워크 플랫폼들 상에 구현되었고, 간단한 자바 응용 프로그램을 적재하여 성능을 평가할 수 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 센서 네트워크 운영체제에 관한 연구 동향을 알아본다. 3장에서는

* 책임저자(Corresponding Author)
논문접수 : 2007. 9. 29., 채택확정 : 2007. 10. 26.
김성우, 이종민, 이중화 : 동의대학교 컴퓨터소프트웨어공학과
(libero@deu.ac.kr/jongmin@deu.ac.kr/junghwa@deu.ac.kr)
신진호 : 동의대학교 메카트로닉스공학과(jhshin7@deu.ac.kr)

BeeVM 자바 가상 기계의 전체적인 구조와 실행 파일 형식에 대하여 설명하고, 이어지는 4장에서 자바 API와 네트워크 환경을 포함한 BeeVM 자바 실행 환경에 대하여 상세히 서술한다. 5장에서 BeeVM과 간단한 응용 프로그램을 실제로 센서 네트워크 플랫폼에 구현하고 그 결과를 평가하였다. 마지막으로, 6장에서는 본 논문의 결론과 향후 연구 과제를 제시한다.

II. 관련 연구

무선 센서 네트워크(WSN) 을 위한 시스템 소프트웨어는 크게 두 가지 종류로 분류할 수 있다. 그 중 하나는 운영체제 기반 소프트웨어로서 TinyOS[4], Contiki[5], Mantis [6] 등이 여기에 해당되고, 다른 하나는 가상 기계 기반 소프트웨어로서 ASVM[8], VM*[9], Squawk[10], SwissQM[11] 등을 들 수 있다.

TinyOS는 이벤트 구동(event-driven) 프로그래밍 모델 기반의 운영체제로서 가장 널리 쓰인다[4]. nesC라는 컴포넌트 기반 프로그래밍 언어로 작성된 응용 프로그램과 시스템 소프트웨어들이 정적으로 결합되어 실행된다. 그래서, 응용 프로그램을 수정하여야 할 때, 전체 소프트웨어를 바꿔야 하는 부담을 가진다.

Contiki는 이벤트 구동 프로그래밍 모델을 기반으로 한 경량의 운영체제이지만, 선점 쓰레드도 추가로 지원한다[5]. Contiki 는 C 언어로 작성된 응용 프로그램을 실행 시간에 동적으로 적재하거나 제거할 수 있으며, uIP 라는 작은 TCP/IP 스택과 rime 이라는 저전력 통신 스택을 지원한다. 하지만, 이벤트 구동 시스템을 구현한 protothread 구조에서는 스택을 사용하지 않으므로, 지역 변수를 사용할 수 없다는 단점을 가진다.

MANTIS는 멀티쓰레드를 지원하는 최초의 센서 네트워크용 경량 운영체제이다[6]. MANTIS 운영체제는 다양한 플랫폼에 적용 가능하도록 유연성을 지니고 있으며, 메모리 및 에너지 효율적인 구조를 포함하고 있다. 하지만, 현재는 정적인 운영체제 이미지를 센서 노드에 적재하여 실행하고, 제한적으로 원격 접속과 변수 수정을 지원하고 있다.

ASVM은 TinyOS 위에 구현된 스택 기반의 가상 기계인 Mate[7]를 개선한 가상 기계이다[8]. 이들은 TinyOS의 비동기적인 세부 구현 위에 동기적인 프로그래밍 인터페이스를 제공하고, 특정한 명령어 집합을 사용하여 작성된 코드 캡슐을 FIFO 라운드로빈 방식의 스케줄러에 통해 쓰레드 방식으로 실행시킨다. 특히, ASVM은 다양한 응용 범위에 적합하도록 명령어 집합을 지정할 수 있는 유연한 구조를 가진다.

VM*는 자바 프로그래밍 언어로 작성된 프로그램을 센서 네트워크 플랫폼에서 실행할 수 있는 가상 기계의 일종이며, 프로그램의 동적인 갱신이 가능하다는 특징을 가진다[9]. 응용 프로그램의 동적인 갱신은 이전 프로그램과의 차이(diff)를 계산하여 반영하는 증가 결합(incremental linking) 기법을 사용하는데, 코드의 작은 변화도 큰 이동(shift)을 불러올 수 있고, 차이 계산에 대한 부담을 고려하면 그 효과는 크지 않다.

Squawk는 무선 센서 플랫폼에서 운영 체제 없이 자바 프로그램을 실행하도록 만들어진 자바 가상 기계이다[10]. 무선 센서 네트워크 응용 프로그램을 작성하기 쉽도록 GCF(generic connection framework)와 같은 표준 무선 API를 지원하고 장치들 간 파일 인증과 배포를 수행하는 기능도 가진다. 하지만, Squawk 은 ARM과 같은 비교적 고성능의 CPU를 가지고 메모리 사용량이 80~150KB 이상인 하드웨어를 대상으로 만들어져 보다 작은 센서 노드에는 적합하지 않다는 단점이 있다.

SwissQM은 저수준 운영체제와 고수준 프로그래밍 환경 사이의 소프트웨어 계층으로 동작하도록 설계된 스택 기반 가상 기계이다[11]. 작은 메모리 용량에서도 잘 동작하며, 센서 데이터베이스 용으로 특히 적합하다. 하지만, 현재 어셈블리 언어 수준의 프로그래밍 환경만 지원하고 있다.

III. BeeVM의 구조

내장형 자바 운영 체제는 내장형 기기에서 특정한 자바 소프트웨어를 수행하기 위하여 자바 가상 기계를 포함한 운영 체제를 말한다. 내장형 자바 운영 체제는 메모리를 포함한 자원의 제약성, 여러 기기에 적용할 수 있는 유연성 및 확장성, 신뢰성 등을 설계 목표로 정할 수 있다. 본 연구에서 개발한 BeeVM은 이러한 설계 목표를 고려하여 센서 네트워크 용으로 구현하였으며 표준 자바와 유사한 프로그래밍 환경을 제공한다.

1. BeeVM의 실행 환경

일반적인 자바 실행 환경에서는 클래스 적재기(class loader)가 동적으로 실행에 필요한 클래스 파일들을 메모리에 별도로 적재한 후 자바 가상 기계가 이를 실행한다. 하지만, 센서 네트워크와 같은 내장형 시스템에서는 일반적으로 모든 클래스 파일을 적재할 만한 충분한 메모리 자원이 없으므로 기존의 자바 실행 환경을 그대로 적용할 수 없다. 이것을 해결하기 위하여 호스트 컴퓨터에서 클래스 파일을 미리 실행 가능한 형태로 변환하고, 이것을 내장형 시스템의 롬이나 메모리에 탑재한 후에 내장형 시스템에서 구동된 가상 기계 해석기를 통해 직접 실행되도록 한다. 이러한 방식은 분리된(split) VM 방식이라고 할 수 있다[10].

그림 1은 BeeVM에서 사용하는 프로그램 실행 환경을

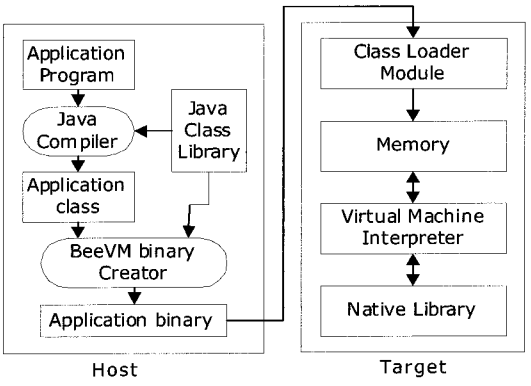


그림 1. 프로그램 실행을 위한 BeeVM의 구조.

Fig. 1. The BeeVM architecture for program execution.

보여준다. 호스트 컴퓨터에서는 다음과 같은 일이 이루어진다. 자바 컴파일러는 프로그래머가 자바 언어로 작성한 응용 프로그램과 자바 클래스 라이브러리를 참조하여 클래스 파일을 생성한다. 생성된 클래스 파일과 자바 클래스 라이브러리는 내장형 센서 네트워크 플랫폼에서 실행하기에 적합한 형태의 실행 파일로 변환된다. 이것과 관련된 내용은 다음 절에 설명한다. 한편, 내장형 플랫폼에서는 가상 기계 해석기와 클래스 적재기가 플래시 메모리 등에 탑재되어 미리 실행하고 있다가 호스트로부터 실행 파일을 다운로드하여 메모리에 적재한 다음 실행한다. 이 때, 가상 기계 해석기는 필요한 경우-특히, 센서 네트워크 하드웨어 장치의 구동과 관련된-네이티브 라이브러리를 참조하기도 한다.

2. BeeVM 의 실행 파일 형식

자바 언어로 작성된 소스 파일을 컴파일하여 생성된 클래스 파일은 흔히 실행 가능한 바이트코드와 함께 객체 하나에 대한 필드, 메소드, 컨스턴트 풀(constant pool) 등에 대한 심볼릭 정보를 포함하고 있다[12]. 자바 가상 기계는 바이트코드 실행 도중 객체 내의 클래스, 필드, 메소드 등을 참조하는 과정에서 인덱스 참조 또는 심볼릭 참조를 직접 참조로 변환한다. 또한, 자바 인터프리터가 여러 객체를 참조하는 응용 프로그램을 실행하기 위하여 필요한 클래스 파일들을 모두 메모리에 적재하여야 한다. 하지만, 센서 네트워크의 메모리 및 속도 제한을 고려한다면, BeeVM 내에 자바 클래스 파일들을 각기 따로 메모리에 적재하기보다는 새로운 형식으로 최적화된 하나의 실행 파일을 대신 적재하도록 하는 것이 더욱 효율적이다.

BeeVM의 실행 파일 생성기는 BCEL(Byte Code Engineering Library)을 사용하여 자바 클래스 파일로부터 필요한 정보를 추출한 다음, 바이트코드의 심볼릭 참조를 직접 참조로 미리 전환(pre-resolve)하여 일반 자바 클래스 파일의 과다한 심볼릭 정보를 제거하고, 참조하는 여러 클래스 파일들을 통합하여 최적화된 자바 실행 파일을 만들어 내장형 플랫폼의 메모리에 적재한다. BeeVM의 자바 실행 파일 형식은 그림 2와 같다.

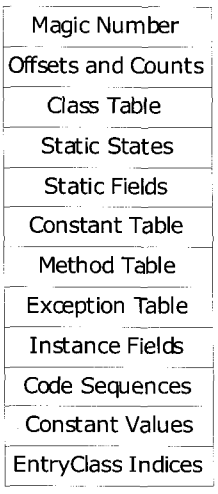


그림 2. BeeVM 의 실행 파일 형식.

Fig. 2. The executable file format for BeeVM.

3. BeeVM 가상 기계 해석기

자바 가상 기계는 클래스 적재기, 쓰레기 수집기, 가상 기계 엔진, 네이티브 메소드 인터페이스, 각종 레지스터와 힙, 스택을 포함하는 메모리 영역 등의 다양한 요소로 구성되며, 구현된 시스템에 따라 특정 요소가 간소화되거나 수정되기도 한다[1,12]. 본 논문에서는 이미 구현된 수많은 자바 가상 기계 중에서 연구의 특성에 부합하는 것을 적극 활용하는 방안을 채택하였다.

BeeVM은 레고 마인드스톰을 위해 개발된 자바 운영 체제 leJOS에서 사용하는 자바 가상 기계 TinyVM를 참고하여 상당 부분을 그대로 적용하였다. 레고 사에서 만든 로봇 키트인 레고 마인드스톰은 Hitachi H8/3292 마이크로컨트롤러와 16KB의 롬 및 32KB의 외부 램 메모리를 포함한 RCX 마이크로컴퓨터를 기반으로 동작하며, TinyVM은 이러한 제한된 기억 용량에서도 다중 선점 쓰레드, 배열, 동기화, 실수 연산은 물론 예외처리 기능을 지원하고 있다[12]. BeeVM은 TinyVM의 장점을 그대로 수용하면서 몇 가지 유용한 기능을 추가하고 수정하였다. BeeVM의 구조적인 특징은 다음과 같다.

바이트코드 해석기는 직접적인 주소 접근이 불가능한 플래시 메모리나 EEPROM으로부터 바이트코드를 읽어들이는 것도 가능하도록 구현하였다. 예를 들면, ATmega128 마이크로컨트롤러를 가진 플랫폼의 경우, 바이트코드를 내부 EEPROM에 적재하고 이를 읽어 실행할 수 있다.

BeeVM은 여러 자바 쓰레드들이 선점 체계 하에서 거의 동시에 수행하는 다중 쓰레드 환경을 제공한다. 쓰레드들은 아주 간단한 쓰레드 스케줄러에 의해 처리되어 한 쓰레드에서 다음 쓰레드로 자동적으로 전환되며, 최대 255 개의 쓰레드를 생성할 수 있다. 또한, 쓰레드간의 동기화와 인터럽트 처리 기능도 사용할 수 있도록 구현되었다.

BeeVM은 다중차원 배열을 지원하여 여러 개의 숫자들이나 객체들을 효과적으로 저장하도록 한다. 더구나, TinyVM에서는 가능하지 않았던 배열의 형 변환(casting)을 지원함으로써 가변 크기의 버퍼 생성 등을 적용할 수 있게 되었다.

4. 네이티브 인터페이스와 장치 구동기

자바 응용 프로그램은 네이티브 인터페이스(native interface)를 통해 특정한 시스템 서비스를 수행하는 네이티브 코드를 호출할 수 있다. 네이티브 인터페이스는 주어진 센서 네트워크 플랫폼의 하드웨어 자원 구동 및 관리를 위해 활용되거나 바이트코드보다 더욱 효율적이고 빠른 실행을 요구하는 부분에도 유용하다.

네이티브 메소드는 가상 기계를 만들 때 정적으로 결합한다. 이 때, 각각의 네이티브 메소드 시그내처(method signature)에 대응하는 메소드 번호(index)를 미리 지정하여 나중에 가상 기계가 인식하도록 한다. 자바 응용 프로그램 내에서 특정한 네이티브 메소드를 호출하는 부분은 BeeVM 실행 파일 생성기를 통하여 메소드 번호를 인자로 하는 바이트코드(invokespecial)로 변환된다. 자바 가상 기계가 이 코드를 해석하여 네이티브 인터페이스를 통해 메소드 번호와 자바 스택으로부터의 매개변수를 전달하고 네이티브 코

```
package beevm.comm;
  /**
   * WPAN Low-level comms (LLC)...
   **/
  public class WPANLLC {
   ...
   public static native int read(int src, byte [] buf, int
len);
```

```
   ...
  void dispatch_native (TWOBYTES signature,
STACKWORD *paramBase)
  {
    STACKWORD *paramBase1 = paramBase+1;
    STACKWORD *paramBase2 = paramBase+2;
    switch (signature) {
     ...
#ifdef ZIGBEE_MODULE
      ...
      case read_4I_1BI_5I: /* beevm.comm.WPANLLC */
       {
        int addr = paramBase[0];
        byte *byteArray = ((byte *) word2ptr (paramBase
[1])) + HEADER_SIZE;
        int len = paramBase[2];
        push_word(llc_receive_bytes(addr, len, byteArray));
       }
      return;
   ...
```

그림 3. WPAN 무선통신 네이티브 인터페이스.

Fig. 3. The native interface of WPAN wireless communication.

드가 이를 적절한 자료형으로 처리하고 반환값을 다시 자바 스택으로 되돌려줌으로써 실행 과정이 종료된다.

BeeVM은 메모리, UART 직렬 통신, LED, 센서, 무선 통신 모듈에 대한 네이티브 인터페이스를 구현하였다. 그림 3은 네이티브 인터페이스의 예로서 WPAN 무선통신 모듈에 대한 네이티브 메소드 정의와 해당하는 네이티브 코드 부분을 보여준다.

IV. BeeVM 의 프로그래밍 환경

BeeVM은 제한된 자원을 가진 센서 네트워크 하드웨어 플랫폼 상에서 자바 프로그래밍을 쉽게 할 수 있도록 하는 프로그래밍 환경을 제공한다.

1. 프로그래밍 모델과 지원 API

BeeVM은 여러 개의 응용 프로그램을 타겟 플랫폼에 적재할 수 있도록 설계하였지만, 현재는 하나의 응용 프로그램만 적재하는 것을 검증하였다. 하지만, 프로그램의 병행 수행이 필요한 경우에는 여러 개의 쓰레드를 생성하여 실행할 수 있다.

표 1. BeeVM 전체 클래스 패키지.

Table 1. Total class packages for BeeVM.

패키지 이름	설명
java.lang	자바 기본 패키지
java.util	유용한 클래스 패키지
beevm.platform.nano24	nano24 보드 관련 패키지
beevm.platform.hmote	hmote 보드 관련 패키지
beevm.comm	통신 관련 패키지
beevm.util	기타 유용한 클래스 패키지
Beevm.hostcomm	호스트 플랫폼 용 통신 관련 패키지

BeeVM은 센서나 통신 등의 하드웨어 기능을 이용할 수 있도록 구축된 자바 클래스 API 를 제공한다. 노드는 일정한 시간 간격마다 동기적인 호출을 통하여 센서 값을 추출하거나 데이터를 전송할 수 있다. 하지만, 더욱 복잡한 이벤트 처리가 필요한 응용 프로그램을 위해 자바 이벤트 모델을 사용하여 이벤트 리스너를 통해 발생한 이벤트를 처리할 수 있도록 하는 방안을 구현하고 있는 중이다.

현재 BeeVM이 제공하는 클래스 API 패키지는 다음 그림과 같이 요약할 수 있다. java.lang 패키지는 object, thread, class, string 등의 기본적인 자바 클래스들을 모아놓은 패키지이며, java.util 과 beevm.util 패키지는 타이머와 같은 유용한 기능을 가진 클래스들을 모아놓은 패키지이다. beevm.platform 패키지는 센서 네트워크 플랫폼 하드웨어 상의 메모리, LED, 센서, 전원, UART 등의 구성요소들에 접근하거나 제어하는 기능을 가지며, 직렬/무선 통신 및 네트웍 계층 구조를 위하여 beevm.comm 패키지를 제공하고 있다. 호스트 플랫폼에서는 타겟 플랫폼과의 통신을 위해 beevm.hostcomm 패키지가 따로 필요하다.

2. 통신 체계

BeeVM은 센서 네트워크 환경에서 최소한의 자원을 활용하면서 유연성을 동시에 가질 수 있도록 하는 효율적인 다중 계층의 통신 체계를 갖추고 있다.

통신 계층은 크게 두 부분으로 나누어진다. 하위 계층은 주로 물리 계층과 MAC 계층을 담당하며, 제한된 하드웨어 자원과 속도를 고려하여 네이티브 코드로 구현되어 있다. 상위 계층은 하위 계층에서 미리 등록된 자바 네이티브 함수와 연결하여 내부에 정의된 버퍼를 통해 패킷 메시지를 처리하는 메시지 처리 계층과 서비스 및 라우팅 프로토콜 등의 계층으로 구분하여 사용자가 자유롭게 통신 체계를 구성할 수 있다. 다음 그림은 이러한 BeeVM의 계층적인 통신 체계를 보여준다.

현재 BeeVM내에는 직렬 통신 장치와 IEEE 802.15.4 근거리 무선 장치를 위한 구동 모듈이 구현되어 있다. 직렬 통신 장치는 간단한 싱크(sync) 기능을 가진 프로토콜, SLIP, PPP 프로토콜 등을 지원하므로, SLIP 프로토콜을 사용하는 Contiki 운영체제와 PPP 프로토콜을 사용하는 TinyOS의 개발 환경과도 어느 정도 호환할 수 있다. 근거리 무선 장치를 위해 IEEE 802.15.4 표준 프로토콜을 지원하는 모듈이 구현되어 있다. 뿐만 아니라, 약간의 수정을 거치면 메시지

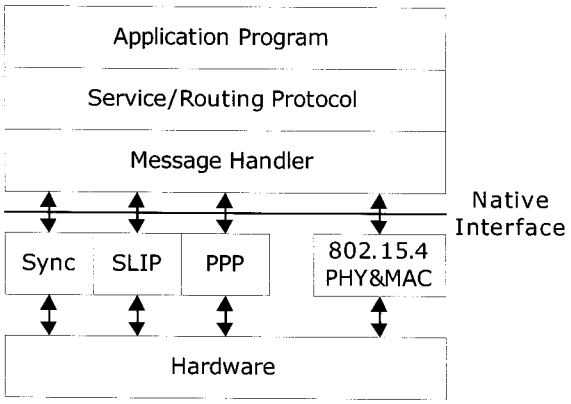


그림 4. 통신 부시스템.
Fig. 4. Communication subsystem.

```
import beevm.comm.*;
import beevm.platform.nano24.*;

public class SensorReader {
 public static void main(String args[]) throws Exception {
   byte[] sensorID = new byte[1];
   int sensorValue;
   byte[] pPacket = new byte[1];
   WPANMessagePort port = null;

   try {
    port = new WPANMessagePort(0, 1);
    if(port.readPacket(sensorID) == 0) return;
    while (true) {
      sensorValue =
Sensor.readSensorValue(sensorID[0], 0);
      pPacket[0] = (byte)(sensorValue & 0xFF);
      port.writePacket(pPacket);
    }
   } catch (Exception e) {
    LED.Red.on();
    Thread.sleep(500);
    LED.Red.off();
   } finally {
     port.close();
   }
 }
}
```

그림 5. 센서 노드 응용 프로그램(SensorReader.java).
Fig. 5. SensorReader.java, an application for the sensor node.

처리기를 통하여 Contiki의 uIP 메시지 또는 TinyOS의 활성 메시지(active message)를 송수신하는 것이 가능하다.

3. 프로그래밍 예제

BeeVM 프로그래밍 환경에서 자바 응용 프로그램은 호스트와 타겟 플랫폼 용으로 구분하여 작성하여야 한다. 호

```
import beevm.comm.*;
import beevm.platform.nano24.*;

public class Coordinator {
 public static void main(String args[]) throws Exception {
   byte[] sensorID = new byte[1];
   byte[] pPacket = new byte[1];
   WPANMessagePort wpan = null;
   SerialMessagePort serial = null;

   try {
    wpan = new WPANMessagePort(1, 0);
    serial = new SerialMessagePort();
    if(serial.readPacket(sensorID)==0) return;
    wpan.writePacket(sensorID);
    while (true) {
      wpan.readPacket(pPacket);
      serial.writePacket(pPacket);
    }
   } catch (Exception e) {
    LED.Red.on();
    Thread.sleep(500);
    LED.Red.off();
   } finally {
     port.close();
   }
 }
}
```

그림 6. 조정자 노드 응용 프로그램(Coordinator.java).
Fig. 6. Coordinator.java, an application for the coordinator.

스트용 자바 응용 프로그램은 주로 직접 연결된 수신 모드로부터 센서 정보를 수집하여 사용자에게 보여주거나 인터넷을 통해 재전송하는 기능을 하며, 일반적인 자바 클래스 라이브러리와 호스트 용 통신 패키지를 결합하여 구현한다.

센서 네트워크 플랫폼 용 자바 응용 프로그램은 센서를 직접 읽거나 다른 노드의 센서 정보를 전달해 주는 센서 노드를 위한 프로그램이나 센서 노드들에게 센서 정보를 보내도록 명령하거나 센서 정보를 수집하는 조정자(coordinator) 노드를 위한 프로그램의 역할을 담당한다. 이 프로그램은 타겟 플랫폼을 위한 자바 클래스 라이브러리를 활용하여 구현한다.

다음은 센서 수집에 대한 간단한 예를 생각해 보자. 호스트 모니터링 프로그램은 직렬 통신을 통해 조정자 노드에게 수집할 센서 ID 또는 번호를 전달하고 조정자 노드는 무선 통신으로 이것을 센서 노드에 전달한다. 센서 노드는 전달받은 센서 번호에 해당하는 센서 값을 읽어 조정자에 무선 통신으로 전송하고, 조정자 노드는 이것을 수신하여 직렬 포트를 통해 호스트에 보내준다. 그리고, 이런 과정이 계속 반복된다. 그림 5와 그림 6은 센서를 읽어 조정자에게 보내주는 센서 노드용 프로그램과 센서 노드로부터 센서

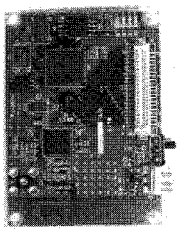
정보를 수집하여 호스트로 보내주는 조정자 노드용 프로그램의 예를 보여준다. 각 노드의 프로그램 코드를 보면, 근거리 무선 통신(WPAN)을 위해 beevm.comm 패키지의 WPANMessagePort 형의 객체를 생성하여 연결 설정을 수행하며, 이 때 생성자 메소드의 인자로 자신의 주소와 통신할 대상의 주소를 지정한다. 연결이 설정되면 읽기 메소드 readPacket() 또는 쓰기 메소드 writePacket() 를 통하여 패킷을 전송할 수 있다.

V. 구현 및 평가

BeeVM은 다양한 센서 네트워크 플랫폼에 적용 가능하도록 설계되었다. 현재, BeeVM 은 Atmel 사의 AVR ATmega128L 마이크로컨트롤러를 사용하는 옥타컴 사의 nano24 보드 [14]와 Texas Instrument 사의 MSP430F1611 마이크로컨트롤러를 사용하는 하이버스 사의 Hmote2420 [15] 라는 대표적인 두 가지 센서 네트워크 플랫폼에 구현되었다. MSP430F1611 마이크로컨트롤러는 프로그램을 위한 48KB의 플래시 메모리와 10KB 의 RAM 메모리를 가지고 있다. ATmega128L 마이크로컨트롤러는 프로그램을 위한 128KB의 플래시 메모리, 4KB의 EEPROM 메모리, 4KB의 RAM 메모리를 가지고 있다. 다음 그림은 BeeVM을 탑재한 센서 네트워크 플랫폼 하드웨어의 외양을 보여준다.

BeeVM을 여러 플랫폼에 탑재 가능하도록 하기 위해서는 해당하는 플랫폼의 내부 구조에 무관하게 프로그램 코드와 데이터를 재배치할 수 있어야 한다. 본 논문의 자바 실행 파일은 플랫폼 독립적으로 만들어져 있어서 BeeVM 가상 기계는 파일이 적재되는 메모리의 시작 위치만 파악하면 실행할 수 있다.

두 가지 적용 플랫폼에서 BeeVM이 탑재된 상태의 메모리 맵은 다음 그림 8과 같다. BeeVM 가상 기계 이미지는 내부 플래시 메모리에 적재되어 실행된다. 자바 실행 파일의 경우, ATmega128L에서는 4KB의 내부 EEPROM 메모리에 적재하고 MSP430F1611에서는 RAM 메모리 중 일부(현재 4KB)를 할당하여 사용한다. 여기서, ATmega128L의 EEPROM은 RAM 메모리처럼 직접 접근할 수 없고 특정한 읽기 동작을 수행하여야 한다. BeeVM 가상 기계가 실행하며 사용하는 C 데이터 및 C 스택 영역과 자바 실행 파일을 위한 자바 스택 및 자바 힙 영역은 각각 RAM 메모리에 적당량을 할당하여 사용한다. 다음 표는 BeeVM이 탑재된 적용 플랫폼 별로 각 영역의 메모리 할당량을 보여준다.



(a) Nano24

(b) Hmote2420

그림 7. 개발 플랫폼의 외형.

Fig. 7. Sensor network hardware platforms.

보통의 자바 응용 프로그램은 표준 자바 클래스 파일(.class)이나 압축 클래스 파일(.jar)을 만들어 실행하지만, BeeVM은 최적화된 자바 실행 파일(.bin)을 실행한다. 다음 표 3은 앞서 언급한 조정자 및 센서 노드를 위한 자바 응용 프로그램에 대하여 클래스 파일(.class), 압축된 클래스 파일(.jar), 그리고 BeeVM 자바 실행 파일에 대한 크기 비교를 보여준다. 표에서 알 수 있는 것처럼 BeeVM 자바 실행 파일의 크기는 표준 자바 클래스 파일에 비해 15% 수준이며, jar 압축 클래스 파일에 비해서도 17% 수준이므로, 대략 1/6~1/7 정도의 크기를 가진다는 것을 알 수 있다. 따라서, 제한적인 메모리를 가지는 임베디드 시스템의 자바 실행 환경에서 표준 클래스 파일 또는 압축 클래스 파일을 사용하는 것보다 BeeVM 자바 실행 파일과 같은 최적화된 형식을 사용하는 것이 훨씬 효율적이다.

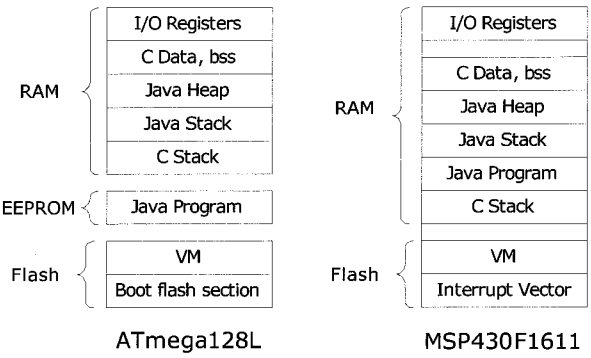


그림 8. BeeVM이 탑재된 적용 플랫폼의 메모리 맵.

Fig. 8. The memory map on the H/W platforms with BeeVM.

표 2. 적용 플랫폼 별 각 영역의 메모리 용량.

Table 2. The memory footprint of each area on the H/W platforms.

(단위: 바이트)

	nano24	Hmote2420
가상 기계 이미지	28,418	22,262
C 데이터	1,245	1,017
C 스택	최대 512	최대 512
자바 스택	최대 480	최대 480
자바 힙	1,859	4,135
자바 프로그램	최대 4,096	최대 4,096

표 3. 응용에 대한 클래스 파일, jar 파일, BeeVM 실행 파일 크기 비교.

Table 3. The byte size comparison of class file, jar file, and BeeVM executable file for the application.

(단위: 바이트)

프로그램	클래스 파일 (.class)	jar 파일 (.jar)	실행파일 (.bin)	.bin /.class	.bin /.jar
Coordinator	25,290	21,456	3,802	0.150	0.177
SensorReader	23,498	19,886	3,251	0.138	0.163

다음으로, 두 가지 센서 네트워크 플랫폼으로 구성된 조정자와 센서 노드에 대하여 각각 해당하는 자바 응용 프로그램을 수행하고 호스트 컴퓨터에서는 조정자와 직렬 통신으로 연결하여 수행 결과를 텍스트 화면을 통해 출력하면서 모니터링하였다. 호스트와 조정자 및 센서 노드 간에 직렬 또는 무선 통신으로 전송된 모든 자료가 정상적으로 송수신되는 것을 확인하였다.

V. 결론 및 추후 연구 과제

BeeVM은 제한된 자원을 가진 센서 네트워크 플랫폼을 위해 구현된 자바 가상 기계이며, 기존의 센서 네트워크 용 소프트웨어 개발 환경의 대안으로서 설계되고 구현되었다. BeeVM은 기존의 TinyOS와 같은 이벤트 구동 방식이나 네이티브 운영 체제 대신 가상 기계 방식을 채택하여 보편적인 쓰레드 방식으로 프로그래밍이 가능하며 임의의 플랫폼에서 실행 가능하다. 더구나, 가상 기계와 응용 프로그램이 분리된 방식을 적용하여 소프트웨어의 갱신이 간편하며, 독자적인 자바 실행 파일을 적용하여 최소한의 메모리가 필요할 만큼 효율적이다. 우리는 본 논문을 통하여 구현한 자바 가상 기계가 대표적인 센서 네트워크 플랫폼들에 적용되어 구동 가능하다는 것을 보여주고, 간단한 자바 응용 프로그램을 통해 성능을 평가하고 검증하였다.

현재 다양한 센서 모듈과 라우팅이나 시간 동기화와 같은 많은 서비스들을 위한 자바 클래스와 성능 평가를 위한 다양한 응용 프로그램을 구현하고 있으며, BeeVM 가상 기계 해석기의 메모리 및 쓰레드 관리를 최적화하는 방안을 연구하고 있다. 아직 연구의 초기인 만큼 향후 구현될 시스템 상에서 동적인 소프트웨어 전송 및 갱신, 센서 데이터베이스, 그래픽 사용자 인터페이스 기반의 센서 네트워크 모니터링 도구, 쓰레기 수집기 등의 다양한 성과가 기대된다.

참고문헌

[1] J. E. Smith and R. Nair, *Virtual Machines*, Morgan Kaufmann Publishers, 2005.

[2] 오세만, 이양선, 고광만, "임베디드 시스템을 위한 가상기계의 설계 및 구현," 한국멀티미디어학회논문지, 제 8권, 제 9호, pp. 1282-1291, 2005. 9.

[3] leJOS Homepage, http://lejos.sourceforge.net

[4] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *Proc. of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Cambridge, MA, November 2000.

[5] A. Dunkels, B. Groenvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," *Proc. of the First IEEE Workshop on Embedded Networked Sensors (EmNets)*, Tampa, Florida, November 2004.

[6] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, R. Han, "MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms," *ACM/Kluwer Mobile Networks & Applications, Special Issue on Wireless Sensor Networks* , vol. 10, no. 4, pp. 563-579, August 2005.

[7] P. Levis and D. Culler, "Mate: A tiny virtual machine for sensor network," *Proc. of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2002.

[8] P. Levis, D. Gay and D. Culler, "Active sensor networks," *Proc. of the 2nd USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI)*, May 2005.

[9] J. Koshy and R. Pandey, "VM*: Synthesizing scalable runtime environments for sensor networks," *SenSys'05*, November 2005.

[10] D. Simon, C. Cifuentes, D. Cleal, J. Daniels, and D. White, "Java on the bare metal of wireless sensor devices: The squawk java virtual machine," *2nd International Conference on Virtual Execution Environments(VEE)*, June 2006.

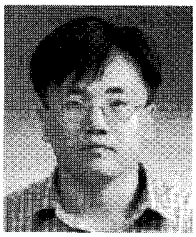
[11] R. Muller, G. Alonso, and D. Kossmann "A virtual machine for sensor networks," *Proc. of EuroSys 2007*, Lisbon, Portugal, March 2007.

[12] Tim Lindholm and F. Yellin, *The JavaTM Virtual Machine Specification*, Addison-Wesley, 2nd Ed., 1997.

[13] B. Bagnall, *Core LEGO Mindstorms Programming*, Prentice Hall PTR, 2002.

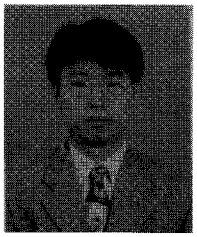
[14] Nano24 platform, http://www.octacomm.net

[15] Hmote2420 platform, http://www.hybus.net



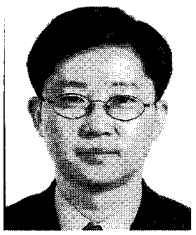
김 성 우

1991년 한국과학기술원 전기및전자공학과 졸업(공학사). 1993년 동 대학원 석사. 1999년 동 대학원 박사. 1999년~2001년 한국전자통신연구원 선임연구원. 2002년~현재 동의대학교 컴퓨터소프트웨어공학과 조교수. 관심분야는 임베디드 소프트웨어, 센서 네트워크, 지능제어.



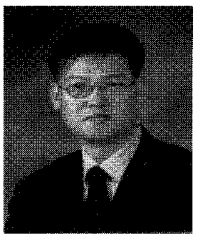
이 중 화

1992년 부산대학교 전자계산학과 졸업(이학사). 1995년 동 대학원 석사. 2001년 동 대학원 박사. 2002년~현재 동의대학교 컴퓨터소프트웨어공학과 조교수. 관심분야는 데이터베이스, XML, 시맨틱 웹 등.



이 종 민

1992년 경북대학교 컴퓨터공학과 졸업(공학사). 1994년 한국과학기술원 전산학과 졸업(공학석사). 2000년 동 대학원 전자전산학과 졸업(공학박사). 1997년~2002년 삼성전자 무선사업부 책임연구원. 2005년 University of California at Santa Cruz, Research Associate. 2002년~현재 동의대학교 컴퓨터소프트웨어공학과 조교수. 관심분야는 모바일 컴퓨팅, 라우팅, 센서 네트워크.



신 진 호

1991년 한양대학교 공과대학 전자공학과 졸업(공학사). 1993년 한국과학기술원 전기및전자공학과 졸업(공학석사). 1999년 동 대학원 박사. 2000년~2002년 도쿄대학교 대학원 기계정보공학과 박사후연구원(JSPS Postdoctoral Fellow). 2002년~현재 동의대학교 메카트로닉스공학과 조교수. 관심분야는 제어 응용, 지능 시스템, 마이크로프로세서 응용, 로보틱스 등.