TinyOS 기반 임베디드 센서 네트워크 소프트웨어 개발의 프로덕트라인 적용 연구

이민태*, 박승범*, 이상준**, 김병기*
*전남대학교 전자컴퓨터공학과,
**전남대학교 경영학과
e-mail:lightwire@naver.com

A Study of Apply Product Line to Embedded Wireless Sensor Network Software development based on TinyOS

Min-Tae Lee*, Seung-Beom Park*, Sang-Jun Lee**, Byung-Gi Kim*
*Dept of Electronics, & Computer Engineering, Chonnam University,
**College of Business Administration, Chonnam University

요 약

낮은 처리속도와 기억용량을 가진 임베디드 센서 네트워크의 소프트웨어를 개발할 때 Tiny OS는 컴포넌트방식의 nesC 언어를 이용하여 효율적인 설계를 돕는다. 컴포넌트 방식을 이용하기 때문에 여러 가지 소프트웨어의 개발에 자주 사용되는 자산을 분리하여 관리한다면 개발시간과 비용을 절감할 수 있다. 본 논문에서는 Tiny OS 기반에서 임베디드 센서 네트워크 소프트웨어 개발 시 프로덕트라인 방법을 적용하여 공통적인 자산과 가변적인 자산을 구별하고 재사용성을 높이며 개발 효율을 증대하는 방안을 제시한다. 이러한 연구는 이종의 임베디드 소프트웨어 개발에도 적용되어 늘어나는 임베디드 소프트웨어의 양적, 질적 수요를 충족시켜주는데 도움이 될 것이다.

1. 서론

최근 무선 통신 기술과 하드웨어 기술의 빠른 성장, 그리고 네트워크 장비의 초소형화 기술의 발달로 임베디드 센서 네트워크가 매우 빠르게 확산되고 있다. 센서 네트워크란 각종 센서에서 수집된 정보를 데이터화하고 다른 컴퓨터에서의 작업을 위하여 무선으로 정보들을 제공할 수 있도록 구성한 네트워크를 뜻한다. 이러한 센서 네트워크 기술은 외부 환경의 감지와 제어기능을 수행하도록 사물에 컴퓨팅, 센싱, 통신 기능을 내장하는 유비쿼터스 컴퓨팅을 실현시키는 핵심기술 중의 하나로서 각광받고 있다[1]. 센서 네트워크를 구성하는 센서노드는 다양한 센싱 작업을 위한 장치와, 수신된 데이터를 처리하는 처리 장치, 센서 노드들 간의 통신을 위한 무선 송수신 장치, 센서 노드에 전력을 공급하는 전력 장치 등이 있다[2]. 각각의 장치들을 효율적으로 관리하는 임베디드 소프트웨어를 개발하느냐가 임베디드 센서 네트워크를 설계하는데 있어서 핵심이며 늘어나는 임베디드 센서 네트워크의 수요를 충족시키는 해결책이 된다.

센서 네트워크 운영체제는 센서네트워크의 응용 개발을 지원하며, 제한된 센서 노드 자원을 관리하는 기반 소프트웨어이다. 자원 제약 사항과 센서 네트워크의 응용을 고려하면, 센서 네트워크 운영체제를 위한 설계 요구사항은 다음과 같다.

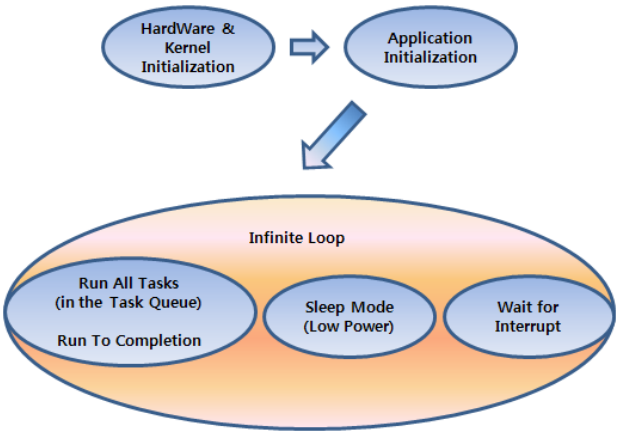
센서 네트워크의 운영체제는 작은 크기여야 하며, 특히 메모리의 사용을 많이 줄여야 한다. 때문에 센서 운영체제의 테스크들은 스택을 공유하거나 선점을 허용하지 않는 경우가 있다. 또한 전력 소모를 관리하기 위해서 전력 관리자를 두며 응용분야에 따라 하드웨어와 소프트웨어가 크게 달라질 수 있으므로 어떠한 응용에서도 사용할 수 있도록 유연성과 모듈성을 갖추고 있어야 한다[1]. 센서 네트워크를 위한 운영체제로는 Contiki, MANTIS, SOS, TinyOS, YATOS 등이 있는데 이중에서 가장 주목받고 있는 운영체제가 TinyOS이다[3]. TinyOS는 운영체제를 포함하고 있을 뿐만 아니라 다양한 미들웨어와 개발 환경을 가지고 있다. TinyOS는 이벤트 발생에 의한 상태 머신 기반의 프로그래밍 개념을 사용하며, 제한된 메모리 공간의 효율적인 이용과 프로세싱의 동시성 등을 지원한다[1].

현재 TinyOS는 재사용성을 강조한 소프트웨어 컴포넌트 기반의 운영체제라는 이점에도 불구하고 소프트웨어 공학적인 측면에서 TinyOS기반 임베디드 소프트웨어 개발 방법론적인 연구가 없는 실정이다. 따라서 기존의 데스크탑 운영체제 환경 하의 소프트웨어개발 방법으로 소프트웨어 재사용을 강조한 소프트웨어 프로덕트라인(PL) 방법론을 적용하여 TinyOS의 특성을 살리고 개발의 효율성을 높여주는 방법을 연구해 본다.

2. 관련 연구

2. 1 TinyOS

TinyOS는 오픈 소스로 임베디드 무선 센서 네트워크용 OS이며 일반적인 PC용이나 임베디드 시스템용 OS들과 다르게 한 번에 하나의 어플리케이션만을 실행한다.[4] 미국 UC 버클리 대학에서 진행해 온 Smart Dust 프로젝트에 사용하기 위해 개발된 컴포넌트 기반의 운영체제로서 센서 네크워크의 요구사항인 제한된 자원과 유연성 등을 고려하여 설계되었다. 핵심코드는 4000바이트 이하이고, 데이터 메모리는 256바이트 이하이며, 이벤트 기반 멀티 테스킹을 지원한다[3].



(그림 1) 부팅 후 플랫폼 동작 순서

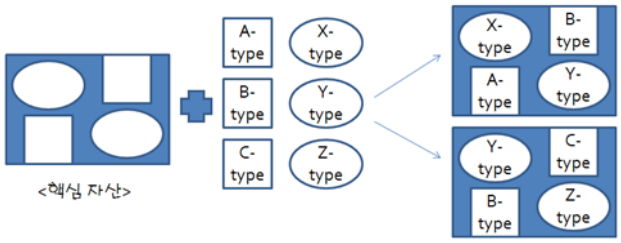
TinyOS의 모든 라이브러리, 컴포넌트는 nesC라는 C언어에 근거를 둔 컴포넌트 언어를 사용하여 만들어졌으며 TinyOS에서 소프트웨어를 설계할때도 nesC를 이용하게 된다[5]. nesC로 작성된 코드는 컴파일되어 C언어로 변환되며 GCC컴파일에 의해 바이너리 파일로 컴파일되게 된다. 컴파일된 바이너리 파일을 플랫폼에 다운로드한 후, 플랫폼을 동작시키면 (그림 1)에서 볼 수 있듯이 하드웨어와 커널을 초기화하며 이후 TinyOS의 커널이 동작하게 된다[4].

2. 2 nesC

nesC는 앞서 언급하였듯이 컴포넌트 기반의 프로그래밍 언어이며 TinyOS의 구조적인 개념들과 실행 모델들을 구현하기 위해 사용되었다. 어플리케이션 개발자들에게 센서 네트워크와 같은 임베디드 네트워크 시스템들의 구현을 지원하기 위해 개발된 프로그래밍언어로 C언어의 확장으로 볼 수 있다. C언어는 마이크로 컨트롤러들을 위한 효율적인 코드 생성이 가능하고, 하드웨어 엑세스에 필요한 기본적인 특성들의 지원이 가능하며 기존 C코드와의 상호작용이 단순하다는 장점이 있는 반면, 안전한 코드를 만들지 못하며 어플리케이션들을 구조화해야 한다는 단점이 있다. nesC는 C언어의 장점을 수용하고 컴포넌트라는 개념에 기반하여 TinyOS의 이벤트 기반 동시성을 지원하고, 공유된 데이터의 동시 엑세스를 가능하게 하였다. nesC는 CPU에 연결되는 각 디바이스들을 동작시킬 수 있도록 컴포넌트로 만들어 놓고 이용하기 때문에 컴파일할 때에 적합한 장치를 적합한 방식으로 사용하게 되고 실제 동작 시에 발생할 수 있는 오류를 컴파일 과정 중에 체크할 수도 있다[4].

2. 3 프로덕트라인

먼저 개발된 프로덕트에서 공통적인 부분을 식별하고 수정하여 재사용하는 것은 프로덕트를 새로 설계하는 것보다 효율적이고 편리하다. 재사용을 위해서는 기존 프로덕트에 대한 확실한 식별과 정확한 분석이 매우 중요하다. 프로덕트라인은 이러한 재사용을 위한 소프트웨어 개발 방법이다. 프로덕트라인은 체계적이고, 계획된 보다 큰 범위의 재사용을 가능하게 하고 새로운 프로덕트를 개발할 때 모든 것을 새롭게 개발하는 것이 아니라 (그림 2)에서와 같이 재사용 가능한 핵심 자산을 이용해 프로덕트들을 개발한다.



(그림 2) 프로덕트라인 개발 모식도

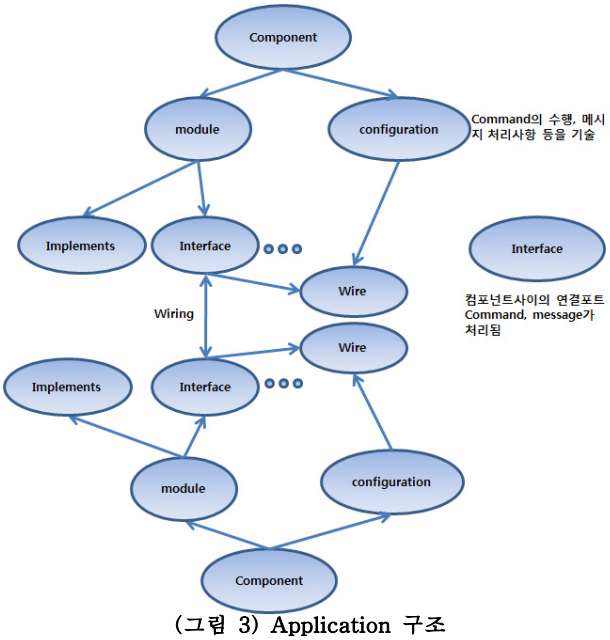
프로덕트라인은 프로덕트군을 개발할 때 최대한 범용적으로 사용할 수 있는 핵심 자산을 추출한 후, 나중에는 각 프로덕트의 개발 요구 사항에 맞춰 가변적인 부분을 프로덕트 구성 요소로 개발하고 핵심 자산과 결합시키는 방법이다[7].

3. TinyOS 기반 개발 방식의 PL 적용 분석

TinyOS에서 사용되는 컴포넌트 기반 프로그래밍 언어인 nesC는 각 장치에 대한 컴포넌트화된 소프트웨어를 부품으로 생각하여 조립하는 설계방식을 가지고 있다. 이러한 설계방식은 다른 임베디드 센서 네트워크 소프트웨어를 개발 할 때 재사용을 용이하게 해주며 재사용성이 많은 프로그래밍 언어라는 점은 데스크탑 환경에서 재사용성을 말하는 PL과 동일하다. nesC의 함수는 크게 Command와 Event라는 두 가지 함수를 사용하며 Command는 일반적인 C언어의 함수와 비슷하고 명령의 방향이 상위 소프트웨어에서 하위 하드웨어로 호출된다. Event는 Command에 따른 기능 수행 후에 발생하는 이벤트를 호출하는 함수이며 하드웨어 동작에 대한 이해가 필요하다[4].

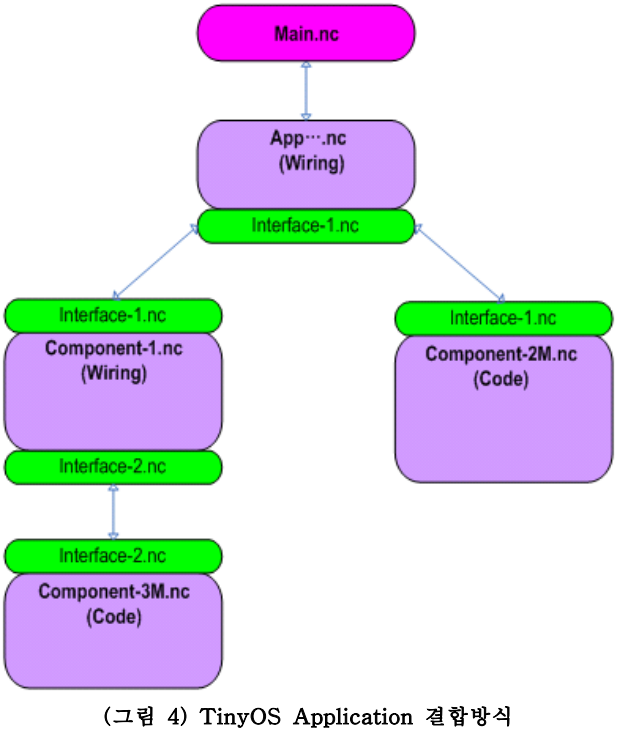
nesC에 추가된 개념이 Component와 Interface인데 일반적으로 알고 사용하는 동명의 개념과 다르다. Component는 nesC의 기본 블록이며 기능을 모아 놓은 일종의 부품이다. module과 configuration 두 가지 타입으

로 구분되는데 RF통신의 기능이 필요하다면 해당 Component를 사용하게 되며 여기서 Component안의 RF 통신기능은 Module안의 Implements에 더 직접적으로 해당이 된다. (그림 3)의 Application의 구조는 TinyOS로 개발되는 응용프로그램들의 정의로서 하나이상의 컴포넌트로 구성되어 컴포넌트사이에 와이어로 연결된 실행가능한 프로그램의 가장 간단한 단위를 설명하고 있다.



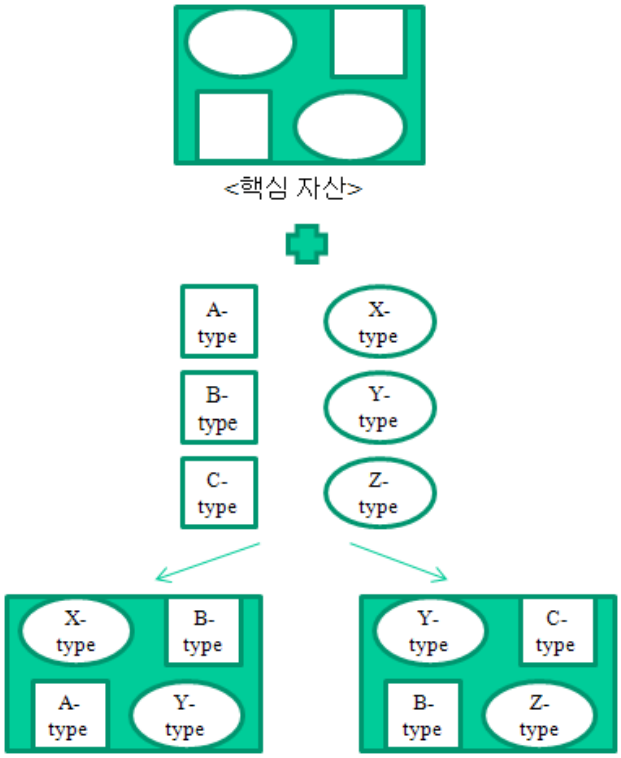
(그림 3) Application 구조

Component를 PL의 관점에서 보면 이미 작동하고자 하는 디바이스를 처리하는 모든 기능이 구현되어 있고 이것을 핵심자산(core asset)으로 재사용성이 가능한 완성된 형태로 판단할 수도 있지만 PL의 공통성과 가변성을 나누는 관점에서 보면 Component는 그 안에 장치의 Specification을 가지고 있다는 점에서 차이가 있다. 실제적인 nesC구현 측면에서 다시 그려진(그림 4)과 PL(그림 5)를 비교해 보면 PL이 핵심자산에 가변적인 기능들만을 조립하여 서로 같은 핵심자산을 가지지만 그 기능은 일부 같을 수도 완전히 다를 수도 있는 Application을 만들어 내는 반면, Component는 서로 완벽한 상호 통신구조인 Interface를 가지고 있는 비슷한 틀을 가진 각각의 컴포넌트들의 조합으로 Application을 만들어 내고 있다.



(그림 4) TinyOS Application 결합방식

nesC는 이미 장치들에 대한 완성된 Component를 정의해 놓았기 때문에 (그림 4)에서 볼 수 있는 각각의 다른 Component들이 공통적으로 가지고 있는 눈에 띄는 특정 요소인 Interface마저 그 안에 포함되어있다.



(그림 5) PL Application결합방식

4. 결 론

임베디드 센서네트워크에 대한 완성된 디바이스 컴포넌트를 제공하는 TinyOS는 재사용을 가능하게 해주고 컴파일 단계에서 에러 검증을 가능하게 해주며 효율적인 소프트웨어 개발이라는 목적을 실현하게 해주는 개발자에게 편리한 환경을 제공해 주는 운영체제이다. 본 논문에서는 현재 가장 부가가치가 높은 임베디드 핵심기술임에도 불구하고 아직 소프트웨어개발 방법론 적인 측면에서 연구가 없는 임베디드 센서 네트워크 소프트웨어를 TinyOS에서 개발할 때 동일한 목적과 개발방법을 제공하는 PL개발방법을 적용하여 보는 것을 연구해 보았다. TinyOS를 운용하는 nesC의 Component라는 구조적인 측면에서 볼 때 PL개발방법은 적용에 있어서 해결해야 할 몇 가지 문제를 가지고 있다. 첫째로 이미 개발자에게 제공되고 있는 라이

브러리와 Component는 그 자체를 디바이스를 동작시키는 가변적인 부분(Variability)이라고 PL관점에서 판단할 수 있지만 동시에 공통적인 부분(Commonality)이라고 불릴 수 있는 성질의 것까지 포함되어 있다. 따라서 PL방법론을 적용하기 위해서는 컴포넌트를 구성하는 모듈 부분을 추출하여 컴포넌트에서 모듈을 선택(Choice)할 수 있도록 하는 것도 방법이 될 수 있다. 둘째로 컴포넌트와 컴포넌트들을 연결시켜놓은 Application을 큰 맥락으로 보고 센서 네트워크를 작동시키기 위한 여러 Application이 가져야 할 필요한 컴포넌트를 다 가지고 있는 하나의 툴킷을 생각할 수 있다. 그러나 임베디드 센서 네트워크의 하드웨어 발전 속도를 생각했을 때 늘어나는 모든 장치의 컴포넌트를 탑재시키고자 하는 관점, 그리고 TinyOS가 가진 특성 중 작은 크기와 적은 메모리 사용이라는 본래의 목적에 부합하지 않게 된다는 반어적인 결과를 고려해야 한다. 향후엔 PL 적용에 있어서 앞서 언급한 부분에 대한 연구가 추가로 필요하겠다.

참고문헌

[1] 김대영, 양진영, 이인선, 유성은, 성종우, Tomas Sanchez Lopez, "무선 센서 네트워크를 위한 임베디드 소프트웨어 기술," 전자공학회지 제31권 제11호, pp.84~98, 11월 2004년.

[2] Ian F. Akydildiz, Weilian Su, Yogesh Sankarasubra-maniam, and Erdal Cayirci, "A Survey on Sensor Networks," IEEE Communications Magazine, 2002.

[3] 우장복, 서효중, "TinyOS를 위한 향상된 전력관리 기법," 제25회 한국정보처리학회 춘계학술발표대회 눈문집, 제13권 제1호, pp.1371~1376, 5월 2006년.

[4] 강정훈, 유준재, 윤명현, 이민구, 임호정, "TinyOS 프로그래밍," TinyOS Korea Forum, URL:www.tinyos.or.kr.

[5] Getting Started with ZigbeX, (주)한백전자, pp.21.

[6] 이민태, 김철현, 김귀연, 김병기, "프로덕트라인 모바일 콘텐츠 시스템의 목표, 시나리오, 휘처 기반의 도메인 분석 연구," 제27회 한국정보처리학회 춘계학술발표대회 논문집, 제14권, 제1호, pp.275~278, 5월 2007년.