제22회 한국정보처리학회 추계학술발표대회 논문집 제11권 제2호 (2004. 11)

임베디드 환경에서의 효율적인 디버깅을 위한 모니터링 시스템 설계

신원, 김태완, 장천현
건국대학교 컴퓨터 정보통신공학과

e-mail:wonjjang@cse.konkuk.ac.kr
{twkim, chchang}@konkuk.ac.kr

Design of Monitoring System for efficient debugging on Embedded Environment

Won Shin, Tae Wan Kim, Chun Hyon Chang
Dept of Computer Science and Engineering, Konkuk University

요 약

최근 가정 혹은 사무실과 같은 장소에서 DVD 플레이어, 셋탑박스, MP3 플레이어 등 많은 임베디드 시스템들을 볼 수 있다. 임베디드 시스템(embedded system)이 점점 많은 분야에서 사용됨에 따라 시스템 운영을 위한 임베디드 소프트웨어들도 각 분야에 맞는 다양한 구조와 기능들이 필요하다. 하지만, 한정된 시간에 다양한 구조와 기능들을 구현해야 하는 소프트웨어 개발은 큰 어려움이 따른다. 이러한 소프트웨어 개발을 좀 더 빠르고 쉽게 하기 위해 프로파일링, 디버거 등의 도구들이 등장했다. 그 중 디버거는 개발 기간 단축을 위한 필수적인 도구이다. 기존의 디버거는 모든 변수에 대한 모니터링으로 생기는 오버헤드와 디버거 모듈을 삽입함으로써 많은 자원을 소비하는 문제가 발생한다. 한정된 자원을 사용하는 임베디드 시스템에서의 불필요한 자원소비와 복잡한 처리 등은 프로그램 강제 종료, 시스템 오작동 등의 큰 문제를 발생시키는 요인이 된다. 본 논문에서는 이와 같은 문제 해결을 위해 사용자가 원하는 변수만을 모니터링 하여 자원소모를 최소화할 수 있는 모니터링 센서 기법과 실행시간 중에 모니터링 대상을 변경하여 빠른 디버깅을 지원 하는 디버깅 레벨 기법을 제안한다.

1. 서론

최근 가정 혹은 사무실과 같은 장소에서 DVD 플레이어, 셋탑박스, MP3 플레이어 등의 많은 가전기기를 볼 수 있다. 이러한 가전기기들을 총칭하여 임베디드 시스템이라 한다. 임베디드 시스템이란 어떤 시스템에 내장되어 특정한 목적만을 수행하는 컴퓨터이다. 이러한 임베디드 시스템에는 시스템을 운영하기 위한 임베디드 소프트웨어가 탑재된다.

임베디드 시스템의 응용 분야가 다양해지면서, 예전에는 단순했던 임베디드 소프트웨어가 점점 더 복잡하고 다양한 기능을 처리하게 되었다. Time-to-market이 생명인 임베디드 시스템 제조업자들은 임베디드 소프트웨어를 빨리 개발하여 프로젝트 기간을 단축하는데 반드시 필요한 임베디드 소프트웨어 개발 환경을 강력히 원하고 있다[1][2].

임베디드 소프트웨어 개발환경에는 프로파일링 도구, 디바이스 드라이버 개발 툴킷, 디버거 등이 있다. 이 중 디버거는 개발기간이 짧은 임베디드 소프트웨어의 개발기간 단축을 위해 필수적인 도구이다.

기존의 디버거는 디버깅 모듈을 사용하여 모든 변수에 대한 값을 도출하고 그 값 중에서 사용자가 요청한 변수를 모니터링 하는 방법을 사용하였다. 이러한 디버깅 방식은 모든 변수를 모니터링 함으로써 많은 부하가 발생된다는 것과 모니터링 모듈의 탑재로 인한 많은 자원을 사용한다는 문제점이 있다.

이러한 문제점을 해결하기 위해 본 논문에서는 사용자가 선택한 변수만을 모니터링하여 최소한의 처리가 가능하게 하는 모니터링 센서 기능과 컴파일 과정을 다시 거치지 않고 모니터링 대상을 변경하여

본 논문은 2004년도 중소기업청 산학연 공동 개발 컨소시엄에 의한 연구 결과임

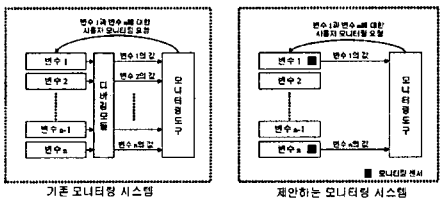
빠른 디버깅 처리를 지원하는 디버깅 레벨 기능을 제공하는 모니터링 시스템을 제안한다.

본 논문은 2장에서 기존의 모니터링 시스템에 대한 관련 연구를 다루고, 3장에서는 임베디드 소프트웨어 디버깅을 위한 모니터링 시스템의 설계 내용을 기술한다. 마지막 4장에서는 본 논문의 성과 및 향후 연구 방향에 대해서 기술한다.

2. 기존 모니터링 시스템

기존 모니터링 시스템에서는 타겟시스템에 디버깅 모듈을 포함한 프로그램을 실행하여 모든 변수에 대한 모니터링 값을 처리하고 모니터링 값 중에서 사용자가 원하는 변수의 값을 시각화한다. 이러한 방식의 사용은 모든 변수에 대한 모니터링 값 처리와 프로그램상에 디버깅 모듈을 포함함으로써 모든 자원의 사용량이 많아진다는 문제점이 발생한다.

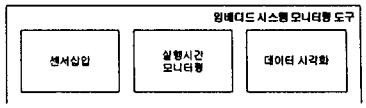
이러한 문제점을 해결하기 위해 모든 변수가 아닌 사용자가 원하는 변수에 모니터링 센서를 삽입하여 자원 사용량을 줄이고 프로그램 실행 중에 모니터링 대상을 변경할 수 있는 디버깅 레벨 개념을 적용하여 보다 빠른 디버깅 환경을 지원한다. <그림 1>은 기존 모니터링 시스템과 제안하는 모니터링 시스템을 나타낸 그림이다.



<그림 1> 기존 모니터링시스템과 제안하는 모니터링시스템 비교

3. 임베디드 시스템 모니터링 도구

제한된 자원을 사용하는 임베디드 시스템에서 복잡한 연산 또는 많은 자원의 사용은 시스템과 소프트웨어를 불안정하게 만드는 큰 요인이 된다. 그러므로 타겟시스템의 소프트웨어는 복잡한 연산과 자원 사용을 최소화 하는데 많은 노력을 기울여야 한다. 임베디드 시스템 모니터링 도구는 한정된 자원을 사용하는 타겟시스템에서는 간단한 요구와 응답, 데이터 전송만을 담당하고 데이터 분석 및 구성, 시각화 등은 호스트시스템에서 수행된다. 임베디드 시스템 모니터링 도구는 크게 센서 삽입, 실행시간 모니터링, 데이터 시각화로 구성된다. <그림2>는 임베디드 시스템 모니터링 도구의 전체 구조이다.



<그림 2> 전체 구조

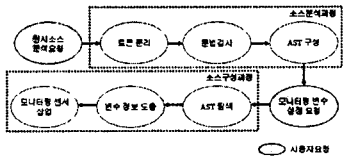
임베디드 시스템 모니터링 도구는 입력된 소스를 분석하여 AST(Abstract Syntax Tree)를 구성하고 사용자가 감시하고 싶은 변수들에 모니터링 센서를 삽입하는 모니터링 센서 삽입과정과 프로그램의 실행시간에 디버깅 레벨 정의 요청을 통하여 감시하고 싶은 변수들을 모니터링 하는 실행시간 변수 모니터링 과정과 모니터링 된 데이터들을 분석하고 정보로 구성하는 데이터 시각화 과정을 거친다.

3.1 모니터링 센서 삽입

임베디드 모니터링 시스템은 사용자가 원하는 변수에 대한 모니터링을 하기 위해 모니터링 센서를 삽입한다. <그림 3>은 모니터링 센서 삽입 구조를 나타낸다.

<그림 3> 모니터링 센서 삽입 구조

모니터링을 하기 위한 원시소스는 어휘분석기를 통하여 토큰으로 분리된다. 여기서 토큰이란 의미있는 문법단위이다. 구문분석기는 각각의 토큰을 사용하여 문법이 적절한지를 검사한 후 AST 구성 모듈을 통하여 AST를 작성한다. 사용자가 원하는 변수에 대해 모니터링 센서 삽입요청을 하면 AST 검색을 통하여 추출한 위치에 모니터링 센서를 삽입한다. <그림 4>는 모니터링 센서 삽입 과정을 나타낸다.



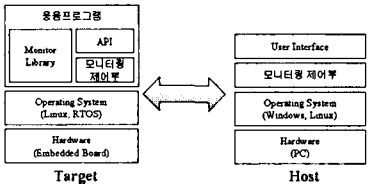
<그림 4> 모니터링 센서 삽입 과정

모니터링 센서가 삽입된 원시소스는 컴파일 과정

을 거쳐 타겟시스템에 전송된다.

3.2 실행시간 모니터링

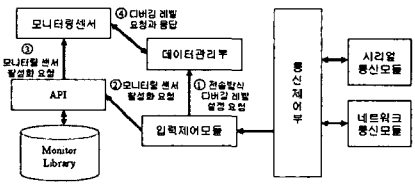
실행시간에 특정 변수에 대한 모니터링은 타겟시스템과 호스트시스템과의 통신으로 이루어진다. <그림 5>는 타겟시스템과 호스트시스템의 구성을 나타낸다.



<그림 5> 실행시간 모니터링 구조

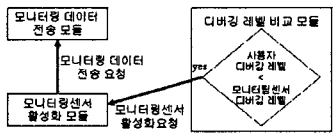
타겟시스템은 사용자의 요청에 대한 응답과 모니터링 데이터 전송을 위한 모니터링 제어부, 모니터링과 관련된 함수를 모아 놓은 모니터 라이브러리, 모니터 라이브러리에 접근하기 위한 API로 구성된다. 또한 호스트시스템은 사용자 요청을 타겟시스템으로 전송하고 타겟시스템으로부터 전송된 데이터들을 분석하고 구성하는 모니터링 제어부, 사용자의 편의를 위한 인터페이스로 구성된다.

사용자가 특정 변수에 대한 모니터링을 하기 위해 디버깅 레벨 설정 요청을 한다. 사용자의 디버깅 레벨 설정 요청이 타겟시스템의 통신제어부를 통해서 입력되면 <그림 6>과 같이 디버깅 레벨 설정 과정과 모니터링 센서 활성화 과정이 수행된다.



<그림 6> 디버깅 레벨 설정과정과 모니터링 센서 제어 과정

사용자가 전송한 전송방식과 디버깅 레벨은 통신제어부를 통해 입력제어모듈에 입력된다. 입력제어모듈은 입력된 데이터를 데이터관리부를 통해 저장한 후 API를 통해 모니터링 센서 활성화 요청을 한다. 활성화 요청을 받은 모니터링 센서는 데이터관리부로부터 얻은 사용자 디버깅 레벨과 모니터링 센서 디버깅 레벨을 비교하고 만약 모니터링 센서 디버깅 레벨이 크다면 모니터링 센서 활성화 모듈을 통해 모니터링 데이터 전송모듈을 활성화 시킨다. <그림 7>은 모니터링 센서 동작 과정을 나타낸다.



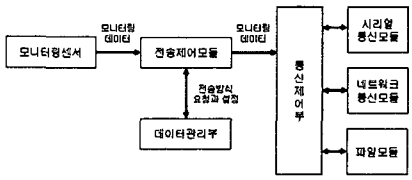
<그림 7> 모니터링 센서 동작 과정

호스트와 타겟시스템 간의 요청과 응답은 규약된 프로토콜을 통하여 진행된다. <표 1>은 프로토콜에 삽입되는 내용 중 데이터의 전송상태, 출력 형태, 디버깅 레벨에 대한 내용이다. 표시 필드는 프로토콜에 삽입될 때의 실제 값이다.

구 분	종 류	표 시
전송상태	transfer	00
	stop	01
	wait	10
출력형태	serial	00
	network	01
	file	10
디버깅레벨	0 ~ 9	000 ~ 101

<표 1> 프로토콜 구성 데이터의 구성

모니터링 센서가 활성화가 되면 모니터링 데이터를 전송제어 모듈로 전송한다. 전송제어모듈은 데이터 관리부로부터 전송방식을 얻고 통신제어부에 설정한 후 모니터링 데이터를 통신제어부로 전송한다. 통신 제어부는 시리얼 통신 모듈, 네트워크 통신 모듈 그리고 파일 모듈 중 설정된 모듈을 통하여 호스트시스템으로 전송한다. 파일 모듈의 경우는 타겟시스템 내의 메모리에 모니터링 데이터를 저장하고 주기적으로 메모리 덤프를 하여 파일로 저장한다. <그림 8>은 이러한 모니터링 데이터 출력과정을 나타낸다.



<그림 8> 모니터링 데이터 출력과정

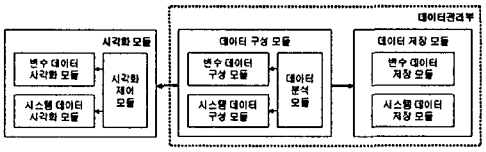
3.3 데이터 시각화

데이터는 크게 변수 데이터와 시스템 데이터로 구성된다. 변수 데이터는 실제 사용자가 모니터링을

위해 지정한 변수 값이다. 시스템 데이터는 CPU 사용량, I/O 사용량, 메모리 사용량을 가리킨다. 변수 데이터만을 통한 모니터링은 프로그램 코드에 의존하는 값이다. 이러한 값을 통한 모니터링은 실제 시스템의 상태에 의해 변화되는 값에 대한 추측이 불가능하다. 이에 따라 CPU, I/O, 메모리 등의 사용량을 통하여 현재 시스템의 상태를 파악함으로써 시스템에 따른 변수 데이터의 변화를 볼 수 있다. 하지만 시스템 데이터를 모니터링 하는 것은 많은 오버헤드가 따르기 때문에 사용자의 선택에 따라 동작유무를 결정한다.

각각의 모니터링 데이터들은 데이터 구성 모듈을 통해 XML 형태 또는 규약된 파일 형태로 저장되거나 도표 또는 그래프 등으로 사용자에게 보여 진다.

<그림 9>는 데이터 관리부와 시각화 모듈을 나타낸다.



<그림 9> 데이터 관리부와 시각화 모듈

데이터 관리부는 데이터 구성 모듈을 통해 모니터링 데이터들을 변수 데이터와 시스템 데이터로 분류하고 각각의 데이터들을 사용자의 요청에 따라 데이터 저장 모듈을 통하여 저장하거나 시각화 모듈로 전송한다. 시각화 모듈은 데이터 관리부로부터 전송받은 모니터링 데이터를 데이터에 따라 각각의 시각화 모듈을 통하여 그래프 또는 테이블 등으로 시각화된다.

3.4 임베디드 모니터링 시스템의 특징

임베디드 소프트웨어의 빠른 개발을 위한 개발환경 중 디버거에 초점을 맞춘 임베디드 모니터링 시스템은 기존 디버거의 문제점인 디버깅 모듈을 프로그램 내에 삽입하여 모든 변수에 대한 모니터링 처리를 함으로써 생기는 많은 자원 소모를 해결하기 위해 모니터링 센서 삽입 기법과 디버깅 레벨 정의 기법을 사용한다.

사용자가 원하는 변수의 모니터링을 위한 모니터링 센서는 최소한의 처리만으로 모니터링 과정이 진행된다. 이때 타겟시스템과 호스트시스템은 데이터 전송 속도 향상을 목적으로 한 규약된 프로토콜을 통하여 모니터링 요청 및 응답을 한다. 프로토콜은 통신상태, 통신 방법 등 모니터링을 하기 위한 가장 기본적인 데이터인 모니터링 데이터 전송 요청과 응답 등으로 구성된다.

제공하는 모든 기능들은 최소한의 자원소모를 위해 최적화된 후 라이브러리 형태로 제공된다. 개발자는 라이브러리 추가만으로 쉽게 변수를 모니터링 한다. 또한 추가적으로 현재 시스템 상태를 모니터링 하는 기능을 제공함으로써 시스템에 의한 오류를 확인할 수 있다. 변수 값과 시스템 상태를 동시에 모니터링 하는 것으로 보다 정확한 프로그램 분석이 가능해져 프로그램의 신뢰성을 높일 수 있다.

4. 결론

임베디드 소프트웨어가 점점 더 복잡하고 다양한 기능을 처리하게 됨에 따라, 개발자들에게는 한정된 시간 내에 복잡하고 다양한 기능이 있는 소프트웨어를 개발을 해야 하는 어려움이 생겼다. 이에 따라 좀 더 빠르게 개발을 하기 위한 도구가 필요하게 되었다.

본 논문에서는 소프트웨어의 빠른 개발을 위한 개발환경 중 디버거에 초점을 맞추고 기존 디버거의 문제점인 모든 변수에 대한 모니터링과 디버깅 모듈 삽입으로 생기는 오버헤드를 해결하기 위해 모니터링 센서 삽입 기법과 디버깅 레벨 정의 기법을 설계하였다. 또한 시스템 상태를 모니터링 함으로써 보다 정확한 프로그램 분석을 통해 프로그램의 신뢰성을 높일 수 있다.

이러한 임베디드 모니터링 시스템은 각종 임베디드 환경의 모든 기술에 탑재, 핵심기술로 적용될 수 있다.

향후에는 보다 편리하고 다양한 기능을 라이브러리에 추가하고 보다 실시간에 가까운 모니터링을 위해 모니터링 센서의 속도 개선이 요구된다.

5. 참고문헌

[1] "The Embedded Software Strategin Market Intelligence Program 2002/2003", Feb. 2003.

[2] 임베디드 소프트웨어 개발 도구 동향, 임형택

[3] Jonathan B. Rosenberg, "How Debuggers Work," John Wiley&Sons, 1996

[4] Nathan Field, Debugging Embedded Linux Application, Nov. 2001.