멀티 스레드 방식을 이용한 임베디드 시스템 데이터의 효율적인 처리 연구

구자진° 최정헌 원유집
한양대학교 전자컴퓨터통신공학과
{space0215°, rhythmical, yjwon}@ece.hanyang.ac.kr

An Efficient Handling Study for Embedded System Data using Multi-Thread Method

Jajin Koo° Jeongheon Choi Youjip Won
Dept. of Electronics and Computer Engineering Hanyang University

요 약

최근 임베디드 시스템 분야의 기기는 하드웨어적, 소프트웨어적 범위가 다양해지고 있다. 하지만 이로 인하여 생성되는 데이터의 종류와 양 또한 증가되어 이를 사용하는 사용자의 시스템에 좋지 않은 영향을 미칠 수 있게 되었다. 하지만 이런 문제에 대해 기존의 관련 연구는 장치의 모델링 기법 및 하드웨어적 구성에 초점이 맞추어져 있었다. 즉 임베디드 기기에서 생성된 다수의 데이터를 사용자에게 적절하게 보여줄 수 있는 방법이 필요로 하게 된다. 본 논문에서는 임베디드 기기 중 차량용 블랙박스 시스템으로부터 추출되는 데이터를 사용자에게 보여주는 시스템을 개발하고, 제공되는 데이터를 기반으로 사용자의 시스템에서 효율적으로 처리 될 수 있는 방법을 제시한다.

1. 서론

최근 임베디드 시스템(Embedded System) 분야에서는 기존의 산업 제어 시스템은 물론 개인의 편리성을 충족 시킬 수 있는 다양한 기기가 지속적으로 개발되고 있다. 임베디드 시스템이란 미리 정해진 특정한 기능을 수행하기 위하여 소프트웨어와 하드웨어를 조합한 컴퓨터 시스템을 말하며 응용 프로그램에 따라 여러 기능을 수행 할 수 있는 기존의 컴퓨터 시스템과는 구별되는 내장된 시스템의 성격을 지닌다.

이러한 임베디드 기기는 자동화 기기의 제어장치에서부터 가전 기기와 개인용 휴대 단말기에 이르기 까지 그 적용 하드웨어 범위가 다양해지고 있으며, 더불어 각각의 하드웨어와 연동되어 소프트웨어 또한 복잡하고 다양하게 개발되고 있는 추세이다.

반면 임베디드 기기에서 생산하고 사용자가 활용하는 데이터의 종류와 양 또한 증가하게 되었는데 이점은 데이터를 사용하는 사용자의 측면에서 볼 때 사용자의 시스템에 무리를 줄 수 있고, 또한 시스템의 자원활용에 문제점을 줄 수 있다. 기존의 연구들은[1][2] 증가되는 데이터를 저장하는 임베디드 기기의 모델링 기법과 하드웨어적 구성에만 초점이 맞추어져 있고, 사용자 환경에서의 데이터 처리에 대한 기법은 연구가 진행 되지 않았다. 따라서 임베디드 기기에서 생성된 다양한 데이터를 사용자 시스템에서 이용할 때 데이터 처리 방안과 데이터를 재생함에 있어 효율적인 시스템 사용에 대한 방안이 필요하다.

본 논문에서는 임베디드 기기 중 최근 이슈가 되고 있고, 또한 기기에 의해 생성되는 데이터의 종류와 양이 타 기기보다 상대적으로 많은 차량용 블랙박스에 대하여 저장되는 데이터를 사용자에게 제공하는 시스템을 개발하고, 멀티 스레드 기법을 사용하여 데이터가 사용자의 시스템에서 재생될 때 사용자에 의해 발생되는 컨트롤 정보에 대한 각 데이터 처리기의 반응 속도를 증가시키고, 시스템이 데이터를 처리하기 위해 사용하는 시간을 줄이는 방법을 제안한다.

2. 관련기술

2.1 멀티 스레드 모델링

스레드란 어떠한 프로그램이 실행되는 흐름의 단위를 말한다. 일반적으로 한 프로그램은 하나의 스레드

를 가지고 있지만, 프로그램 환경에 따라 둘 이상의 스레드를 동시에 실행할 수 있고, 이러한 실행 방식을 멀티 스레드라고 한다. 멀티 스레드를 사용할 경우의 프로그램은 사용자에 대한 응답 성을 향상 시킬 수 있고, 또한 프로세스가 생성시 필요한 자원을 재할당 받는 것에 반해 스레드는 자신이 속한 프로세스의 자원을 공유하고, 또한 문맥 교환이 보다 경제적이다.

본 논문에서는 사용자에게 제공되는 다양한 종류의 데이터를 각 데이터 별로 서로 다른 스레드로 관리하여 프로그램의 응답성과 프로그램에 사용되는 시스템 자원의 양을 줄이고자 한다.

2.3 MVC (MODEL-VIEW-CONTROLLER) MODELING

MVC는[3] 모델(Model), 뷰(View), 컨트롤(Controller)로 나누어지는 구조로 소프트웨어를 설계하는 패턴을 말한다. 모델의 경우 어플리케이션의 유효성 검증이나 로직과 데이터에 대한 모든 설계 사항을 포함하고 있다. 뷰는 사용자 인터페이스를 위한 외부 입력 요소를 말한다. 컨트롤은 뷰와 모델 사이에서 데이터나 로직의 흐름을 제어 하는 역할을 한다.

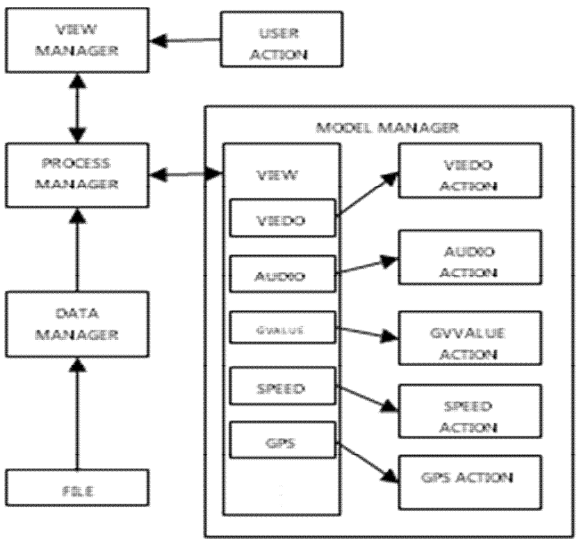
MVC 패턴에서는 각 구성요소들이 서로 연관돼 작업을 수행하지만 각각의 역할구분은 명확히 나눠져 있으며 서로의 역할에 대해서 독립성을 보장한다. MVC 패턴의 가장 큰 장점은 외부의 변화에 대해서 비교적 유연하게 반응할 수 있다는 것이다. 뿐만 아니라 소프트웨어가 복잡해짐에 따라 발생하는 품질 관리에 대한 테스트 또한 MVC 패턴의 구조적 단순함으로 인하여 편리성을 가져다 준다.

본 논문에서 실험용으로 구현한 어플리케이션은 MVC 패턴 기반으로 구성되어 있으며, 실험의 주요 대상이 되는 데이터 처리기는 MVC의 모델 부분과 매핑된다.

3. 시스템 구성

3.1 프로그램 구성

본 논문을 위해 구현한 프로그램의의 구성도는 그림 1 과 같으며 그 내용은 다음과 같다. 프로그램이 실행되면 DATA MANAGER는 파일 에서 로드 한 데이터의 헤더를 분석하여 파일에 속해 있는 데이터의 속성 마다 필요한 메모리 공간을 할당하고 매핑 시킨다. 이후 PROCESS MANAGER는 데이터 포인터와 사용자로부터 받은 컨트롤 메시지를 수집하는 VIEW MANAGER부터 변경사항을 전달받아 MODEL MANAGER에 전달하는 역할을 한다. 마지막으로 MODEL MANAGER는 데이터를 이용하여 각 뷰의 생성, 외부 반응에 대한 처리를 담당하게 된다. 이때 프로그램이 하나의 단일 스레드로 동작 할 경우 각 뷰를 처리하는 프로세싱은 순차적으로 진행되게 된다. 즉 각의 뷰들이 외부 반응으로부터 동시에 동작할 수 없으며, 또한 동일 시점에 해당하는 데이터를 출력할 수 없는 동기화 문제가 발생하게 된다.



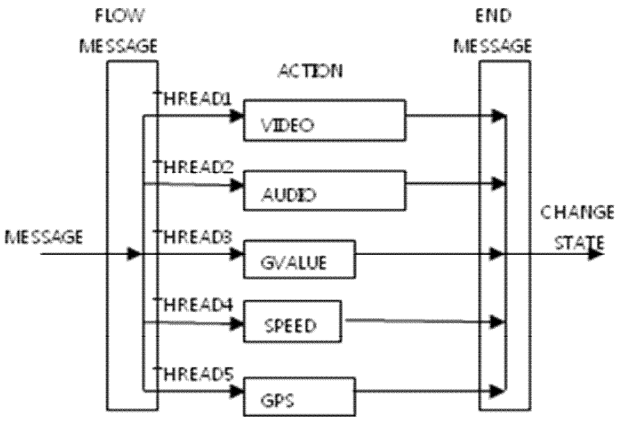
<그림1> 프로그램 모델

3.2 멀티 스레드와 동기화 방법

본 논문에서는 각 뷰를 관리하는 MODEL MANAGER 부분이 단일 스레드의 사용으로 인해 발생되는 문제점을 해결하고자 해당 부분을 멀티 스레드로 구성하고, 외부로부터 받은 메시지를 각 뷰에 처리 하는 방식은 그림2와 같이 구성하였다.

외부의 환경에 의해 변경된 데이터가 뷰를 관리하는 매니저에 알리게 되면 매니저는 각 뷰의 스레드에 동일한 형태의 메시지를 각 뷰에 전달하게 되고, 각 뷰는 독립적으로 해당 뷰를 동작하게 된다. 동작이 끝난 뷰는 다른 뷰의 동작이 끝남을 기다리게 되고, 전체 뷰의 동작이 끝났을 때 외부로부터 받은 메시지에 대

한 뷰의 처리가 끝나게 되어 각 뷰는 메시지에 의해 변경된 상태를 유지하게 된다.



<그림2> 멀티 스레드 기법을 이용한 뷰 처리 과정

또한 멀티 스레드를 사용하게 될 경우 각 뷰의 작업이 동시에 진행되고, 끝나는 동기화 기법이 필요하다. 본 논문에서는 뷰를 관리하는 매니저에 의해 일정 시간 마다 FLOW MESSAGE를 발생하게 되고, 각 뷰의 실행 시간은 해당 메시지를 기반으로 동작을 수행하게 된다. 그리고 각 뷰의 작업이 끝날 경우 END MESSAGE를 발생하게 되고, 모든 뷰의 실행 종료를 감지한 함으로 인해 전체 뷰의 동기화를 유지한다.

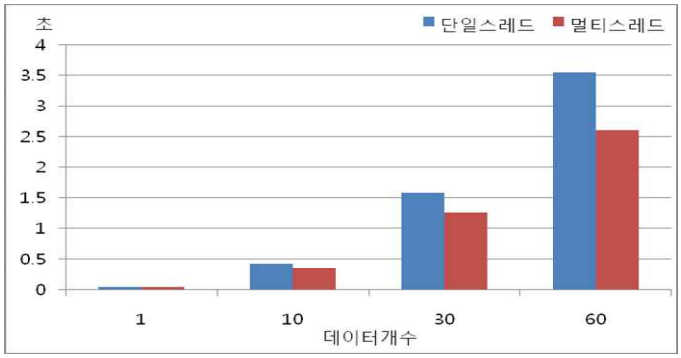
4. 실험

본 논문에서 제안된 기법에 시뮬레이션을 구현하여 평가하였다. 이를 위하여 마이크로 소프트 윈도우 XP를 기반으로 한 데스크 탑 컴퓨터에서 차량용 블랙박스에서 생성한 데이터를 재생할 수 있는 MFC 기반의 어플리케이션을 개발하였다. 또한 뷰를 제어하는 매니저는 단일 스레드 방식과 멀티 스레드 방식으로 구현하여 실험 하였다.

첫 번째 실험은 프로그램에서 처리되는 데이터의 개수에 따른 처리시간을 나타낸 것이다. 실험에 사용되는 데이터는 영상과 음성 값, Gvalue값, Speed값, GPS 값의 5개 데이터를 하나의 작업셋으로 묶어 놓은 것을 말한다. 이때 영상 값은 초당 24frame 값이 저장되어 있는 Mpeg 데이터 1frame 을 말하며 음성의 경우 영상 1frame이 재생되는 동안 사용되는 데이터 양을 기준으로 사용하였다.

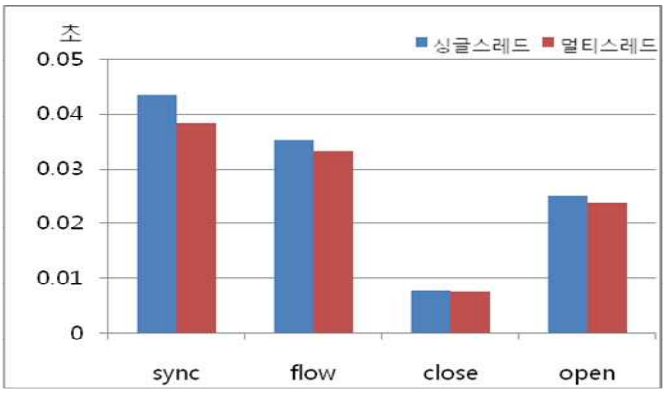
그림1 에서 보는 것과 같이 데이터에 상관없이 멀티 스레드를 사용하는 경우 단일 스레드를 사용할 경우보다 데이터를 처리하는 시간이 더 적게 걸리는 것을 알 수 있다. 이유는 단일 스레드로 뷰를 처리 할 경우 각 뷰는 순차적으로 실행되게 되어 다른 뷰의 데이터 처리 시간, I/O 작업 시간, 메모리 액세스 시간에 대한 페널티를 가지게 된다. 하지만 멀티 스레드로 동작 할 경우 각 뷰는 서로 다른 스레드에서 동작하게 되므로 대기 시간이 줄어들게 된다.

또한 데이터의 개수가 증가 할수록 단일 스레드 대비 멀티 스레드의 성능향상이 증가하는 것을 알 수 있다.



<그림3> 데이터 개수에 따른 처리 시간

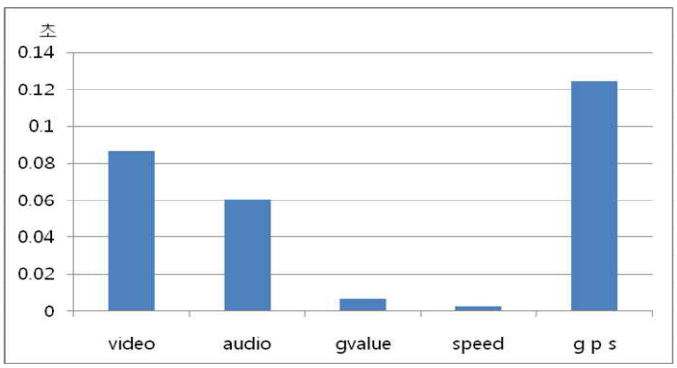
두번째 실험은 프로그램 실행 도중 외부에 의해 컨트롤 메시지가 발생 하였을 때 단일 스레드, 멀티 스레드 상황에서 모든 뷰가 반응 하기 까지의 시간을 측정 하였다. 그림 4에서 보는 것과 같이 멀티 스레드일 경우 모든 메시지에 대한 반응 시간이 다소 빠른 것을 알 수 있다. 이유는 각 뷰의 메시지 전달이 순차적으로 발생하는 단일 스레드의 경우 메시지 처리에 따른 메모리 엑세스 타임이 뷰들의 대기 시간을 발생 시키는 반면 멀티 스레드의 경우 각 뷰는 서로 다른 객체에서 실행 되어 다른 뷰의 처리 시간동안 대기 하는 시간이 발생 하지 않기 때문이다. 또한 메모리 엑세스가 상대적으로 더 많이 발생하는 데이터의 재생 위치 변경 메시지인 sync 메시지와 일정 시간 마다 모든 뷰에 동기화 정보를 알려주는 flow 메시지의 경우에 뷰를 열고 닫는 처리를하는 메시지보다 멀티 스레드를 사용할 경우 처리 시간 감소가 더 많음을 알 수 있다.



<그림4> 각 상황에 따른 뷰의 메시지 처리 반응 속도

마지막 실험은 단일 스레드, 멀티 스레드 상황에 따라 프로그램을 실행 할 때 각 뷰를 초기화 하는 시간을 측정하고, 두 상황에 대한 소요시간을 측정하였다. 그림 5는 각 뷰의 초기화 시간을 나타낸 것이다.

각 뷰는 초기화 작업시 데이터 처리에 사용할 메모리 공간을 확보하고, 파싱 하는 시간이 소요되게 된다. 대체 적으로 처리하는 데이터의 양이 많은 영상과 음성에서 상대적으로 긴 초기화 시간이 발생하게 됨을 알 수 있다. GPS의 경우 가장 긴 초기화 시간을 나타내고 있다. 그 이유는 처리하는 데이터의 양이 영상과 음성에 비해 적지만 GPS를 처리하는 웹 브라우저의 로딩 시간이 발생하여 그림과 같이 뷰에서 가진 긴 초기화 시간이 발생하게 된다.



<그림5 > 각 뷰의 초기화 시간

다음의 표1은 단일 스레드와 멀티 스레드 사용시 모든 뷰들을 초기화 하는데 소요되는 시간을 나타낸 것이다. 멀티 스레드를 사용할 경우 단일 스레드를 사용할 때보다 약 2배 정도 초기화 시간이 빨리지는 것을 알 수 있다

상황	소요 시간 (초)
단일 스레드	0.28
멀티 스레드	0.12

<표1> 상황에 따른 초기화 시간

5. 결론

이상에서 설명한 바와 같이 본 논문에서는 임베디드 시스템에서 생성된 다양한 데이터를 사용자 시스템에서 재생할 때 각 데이터를 출력하는 뷰를 멀티 스레드로 사용할 경우에 대한 효율성을 제시하였다. 추가적으로 개선된 사항을 확인하기 위하여 차량용 블랙박스에서 생성된 데이터를 기반으로 재생되는 뷰 어플리케이션을 개발하여 단일 스레드 모드일 경우와 멀티 스레드 모드일 경우의 각 환경에서 데이터 처리 시간, 외부 메시지가 발생 하였을 경우 모든 뷰에 반응시간, 뷰의 초기화 시간과 관련한 실험하여 멀티 스레드를 사용할 경우 약 20%의 실행 속도 향상과, 약 10%의 반응 속도 향상을 보였으며 뷰의 초기화 속도는 약 2배 증가함을 보였다.

Acknowledgement

본 연구는 지식경제부 및 한국산업기술평가 관리원의 IT R&D 프로그램(Large Scale Hyper-MLC SSD Technology Development, No. 10035202)의 일환으로 수행하였음.

본 연구는 ㈜만도에서 시행하는 차량용 블랙박스 시스템 개발 사업(No. 200900000001788)의 일환으로 수행하였음.

참고문헌

[1]]VEDI Technical Committee, "SAE J1698: Vehicle Event Data Interface-Vehicular Output Data Definition" ,SAE, Feb. 2005.

[2] Yungyu Kim, Bum Han Kim, "Real-time Integrity for Vehicle Black Box System", Journal of the Korea Institute of Information Security and Cryptology / v.19, no.6, pp.49-61, 2009