Web SQL Database 기술을 이용한 유비쿼터스 화재 방재 모니터링 시스템 연구

강승구*, 김영혁*, 임일권*, LiQiGui*, 이준우*, 김명진**, 이재광*
*한남대학교 컴퓨터공학과
**랜스(주)
e-mail:(sgkang, yhkim, iklim, qgli, jwlee, jklee)@netwk.hannam.ac.kr* , mjkim@lans.co.kr**

A Study on Web SQL Database technology using Ubiquitous Fire Prevention Monitoring System

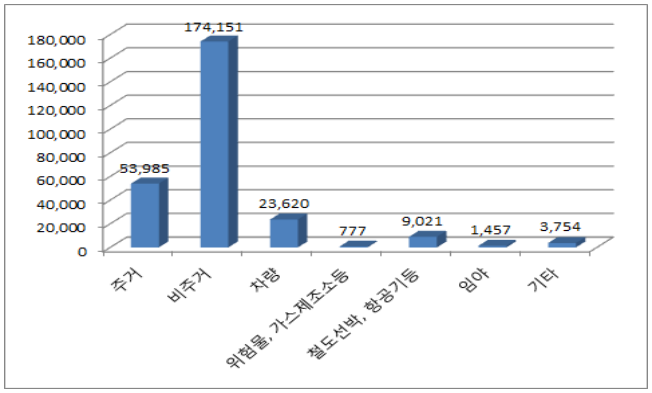
Seung-Gu Kang*, Young-Hyuk Kim*, Il-Kwon Lim*, LiQiGui*, Jun-Woo Lee*, Myung-Jin Kim**, Jae-Kwang Lee*
*Dept. of Computer Engineering, Hannam University
**LANS Inc.

요 약

유비쿼터스 화재 방재 시스템은 온도, 습도, CO, CO2 등의 다양한 센서로부터 얻는 값을 이용해 화재를 판별하여 관리자에게 전달하고 시스템 설정 값에 따라 소화설비를 동작시키는 지능형 화재 탐지 시스템이다. 기존의 시스템은 센서로부터 얻는 값을 서버의 데이터베이스에 저장하였으며, 모니터링 시스템은 모든 정보를 MS-SQL, MySql, Oracle 등의 서버측의 데이터베이스에 의존하였다. 앞선 방법들을 분석하고, 서버의 부하를 줄이기 위하여 HTML5부터 지원하는 Web SQL Database를 이용하는 방법을 제안한다.

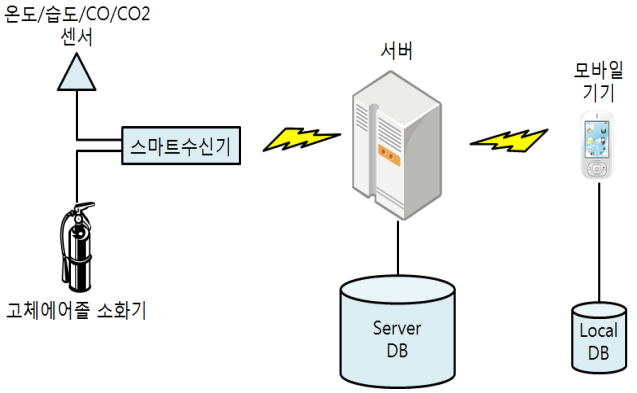
1. 서론

소방방재청 2010년 화재발생현황 분석 자료에 따르면, 2010년 화재건수는 41,862건으로 전년대비 11.5%(5,456건) 감소하였고, 인명피해는 1,891명으로 22.5%(550명) 감소하였지만, 재산피해는 266,765백만원으로 5.9%(14,912백만원) 증가하여 일평균 화재는 114.7건, 인명피해 5.18명, 재산피해 730만원의 손실이 발생하였다[1]. 무엇보다 화재장소별 항목에서 비주거(교육시설, 판매·업무시설, 집합시설, 의료복지시설, 산업시설, 운수·자동차시설, 문화재시설, 생활서비스시설, 기타 건축물·시설물)가 65.3%(174,151건)로 1위를 차지함으로써 사람이 거주하지 않아 상시감시가 필요한 장소의 화재발생건수가 가장 높았다는 점은 기존 시설물에 인가되고 설치된 화재탐지 및 방재 시스템이 미흡하다는 것을 나타내는 결과이다.



(그림 1) 2010년 화재장소별 화재 발생 건수

유비쿼터스 화재 방재 시스템이란 일반적인 P형·R형의 소화설비 대신 설치되어 있는 센서로부터 온도, 습도, CO, CO2 데이터를 수신 받는 수신기가 존재하여 데이터를 서버로 전송하고, 내부적으로 화재를 판단하는 알고리즘을 내포하여 화재로 판단되면 소화기를 작동시켜 극초기에 화재를 진압해 피해가 커지는 것을 최소화하는 시스템이다[2]. 서버는 수신되는 데이터를 데이터베이스에 저장해 소화기의 오작동 및 오탐으로 인한 시설물 피해가 발생했을 시의 근거자료를 보존하게 되며, 설치된 센서들의 정보와 시설물의 환경정보를 저장한 데이터베이스에 일반 PC는 물론 모바일 환경에서도 인터넷을 통한 접근으로 감독하고, 누적된 통계치를 실시간 그래프로 확인할 수 있게 되어 쾌적한 환경을 유지할 수 있는 관리감독 시스템으로써의 역할을 병행한다.



(그림 2) 유비쿼터스 화재 방재 시스템 구성도

2. 관련 연구

1) 모니터링 시스템 연구

2011년 김포대학 정보통신과 이재수 교수 발표 논문의 WSN 기반의 화재감지 모니터링 시스템은 각종 센서들이 탑재된 센서 노드들이 화재를 24시간 감시하며 무선 네트워크를 통하여 원격에서의 실시간 모니터링을 제공한다. 해당 시스템은 ZigBee 기술을 이용한 열, 가스, 습도감지 센서를 사용하여 센서 노드에 센서 데이터를 전송하고, 모니터링 클라이언트는 센서 노드들의 정보를 수집하고 제어하도록 설계 되어 있다. 모니터링 클라이언트로는 수집된 정보를 바탕으로 화재나 가스 누출 등의 재난 발생을 알 수 있으며, 화재 발생시 스프링클러의 올바른 동작 여부 역시 센서 노드를 통해서 전달 받는다[3].

2) 데이터베이스 연구

I. 2007년 경원대학교 소방방재공학과 손봉세 연구원 발표 논문의 Microsoft Access를 활용하여 구성한 감지기별 화재사례 데이터베이스의 구성을 보면, 감지기별 데이터베이스는 Access 파일 내부의 테이블 별로 사례 데이터베이스를 구성하고, 데이터베이스 연결 파일을 생성하여 데이터 연결파일의 속성 중 연결대상을 지정하면 다른 프로그램을 활용하여 접근이 가능하다. 데이터베이스 호출 과정은 통합시스템 내부에 상수 값으로 지정되어 있는 데이터 연결파일을 이용하여 데이터베이스를 호출하게 되고, 생성된 인덱스를 이용하여 데이터베이스 내부 테이블에 접근하게 된다[4].

II. W3C(World Wide Web Consortium) HTML (Hyper Text Markup Language) Working Group에서 발표한 HTML5의 Web SQL Database[5]는 기존의 어떠한 웹언어에서도 정식으로 지원하지 않았던 로컬 측 데이터베이스를 웹브라우저에 생성하는 기술로 SQLite라는 경량의 무료 데이터베이스 엔진을 사용한다. SQLite는 일반적인 관계형 데이터베이스에 비해 대규모 작업에는 적합하지 않지만 중소 규모에서 엔진을 웹브라우저에 내장시키면, 구조적이고 체계화된 관계형 데이터를 대량으로 로컬에 저장할 수 있음은 물론 표준 SQL 쿼리를 사용해 데이터를 유연하게 다룰 수 있다[6].

Web SQL Database - unoff

Global user stats*: Support: 22.29%

Method of storing data client-side, allows Sqlite database queries for access and manipulation

Resources: HTML5 Doctor article

	IE	Firefox	Safari	Chrome	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
Two versions back	7.0	3.5	3.2	8.0	10.5	3.2			2.1
Previous version	8.0	3.6	4.0	9.0	10.6	4.0-4.1			2.2
Current	9.0	4.0	5.0	10.0	11.0	4.2	5.0	10.0	2.3
Near future		5.0		11.0	11.1				
Farther future	10.0	6.0	6.0	12.0	11.5				

Note: The Web SQL Database specification is no longer being maintained and support may be dropped in future versions. Feedback

(그림 3) 웹브라우저별 Web SQL Database 지원 현황

(그림 3)은 현재 웹브라우저별 Web SQL Database 지원 현황이다. 일부 웹브라우저에서는 SQLite를 사용하는 Web SQL Database를 지원하고 있지만, 파이어폭스 및 인터넷 익스플로러에서는 지원되지 않고 있다.

III. HTML5 스펙을 작성하고 있는 W3C에서는 저장되는 데이터 스키마의 유연성이 떨어질 수 있고, SQL이라는 별도의 독립된 언어를 기반으로 하기 때문에 브라우저 간의 표준화 및 호환성이 문제가 될 수 있어 유망한 기술 중 하나로 지목된 Web SQL Database 사양을 유지하지 않고 앞으로도 그럴 의향이 없다고 밝혔지만 새로운 대안으로 보다 표준화가 쉬운 Indexed Database API라는 새로운 스펙으로 로컬 데이터베이스를 계속해 나갈 것이라고 밝혔다[7].

3) Chart API(Application Programming Interface)

구글은 그래프의 이미지를 별도의 코드나 스크립트 없이 단지 URL(Uniform Resource Locator)파라미터만으로 생성해주는 Chart API를 발표하였다[8]. Chart API를 이용하여 그린 그래프는 HTML의 img 태그안에 src 속성으로 넣어주면 웹페이지에 쉽게 그래프를 그림으로 보여줄 수 있다. 따라서 다른 스크립트와 연동하여 동적으로 그래프를 화면상에 보여줄 수 있다.

3. 유비쿼터스 화재 방재 모니터링 시스템 설계

본 연구에서 제안하는 유비쿼터스 화재 방재 시스템은 SHT-75센서를 이용하여 온도/습도/CO/CO2 값을 센싱하여 ARM9 S3C2440으로 제작된 수신기에 센싱 값을 송신한다. 수신기는 서버에 수신기ID/온도/습도/CO/CO2/소화설비 동작 여부의 총 6가지 값의 형태로 구성하여 전송하게 되며, 구성된 패킷 형태는 (그림 4)와 같다.

(그림 4) 수신기-서버간 통신 패킷 구조

사용된 서버는 Microsoft사의 Windows Server 2003 운영체제를 사용하고, 웹페이지를 보여주기 위한 웹서버로는 Apache2, 데이터베이스는 MySql을 이용하며, MySql에 접근하기 위하여 서버 사이드 스크립트인 PHP언어를 이용한다. (그림 4)의 패킷은 「ID_온도_습도_CO_CO2_IO」의 형태로 서버에 송신되는데, 서버의 C#으로 만들어진 서버 프로그램은 이 패킷을 수신 받아 밑줄(「_」)을 기준으로 각각의 값을 서버 측의 MySql 데이터베이스에 실시간으로 기록하게 된다. 서버의 MySql의 테이블은 No, ID, Temp, Hum, CO, CO2, IO, Time의 필드로 구성되어 있는데 No필드는 입력된 값의 순서를 매기는 필드로 자동 증가하게 되며, Time필드는 값이 입력되는 순간의 타

임스탬프를 찍는다. 이렇게 저장된 센서 데이터는 웹을 통하여 모니터링 시스템에 전송되며, 모니터링 시스템은 모바일 기반에서 웹에 접속하여 서버 측의 MySql 데이터베이스에 저장된 센서 데이터를 모두 읽어 로컬 데이터베이스에 기록하고, 기록된 값을 기준으로 AJAX와 Javascript로 실시간으로 갱신되는 표와 그래프로써 화면에 보여준다. 본 모니터링 시스템은 Web SQL Database의 새로운 대안인 Indexed Database를 제대로 지원하는 브라우저와 실효성이 없고, 모바일 환경에서의 모니터링 서비스를 제공하는 시스템을 구현하기 위하여 Web SQL Database인 SQLite를 이용하여 설계하였다.

4. 구현 및 평가

본 화재 방재 모니터링 시스템은 수신기로부터 받은 센서 데이터를 서버가 데이터를 받은 시간을 기준으로 서버측의 MySql 데이터베이스에 저장하고, 웹을 통하여 실시간으로 모니터링 할 수 있는 서비스를 제공한다.

```
strReceiveMsg = Encoding.Default.GetString(buff, 0, buff.Length);

strLog = string.Format("[서버-데이타수신] : {0}", strReceiveMsg);

string[] strReceiveMsg_arr = strReceiveMsg.Split('_');

strID = strReceiveMsg_arr[0];
strTemp = strReceiveMsg_arr[1];
strHum = strReceiveMsg_arr[2];
strCO = strReceiveMsg_arr[3];
strCO2 = strReceiveMsg_arr[4];
strIO = strReceiveMsg_arr[5];
```

(그림 5) 서버측 패킷 수신 코드

(그림 5)는 수신 받은 센서 데이터를 밑줄(「_」)을 기준으로 ID, 온도, 습도, CO, CO2, IO 값을 나누어 각각의 변수에 저장하는 C#으로 작성된 서버 프로그램의 패킷 수신 코드이다.

```
OpenMysql();
string sql = "insert into ufps(ID, Temp, Hum, CO, CO2, IO) values('"
    + strID + "', '" + strTemp + "', '" + strHum + "', '" + strCO
    + "', '" + strCO2 + "', '" + strIO + "')";
var Comm = new MySqlCommand(sql, Mcon);
int i = Comm.ExecuteNonQuery();
```

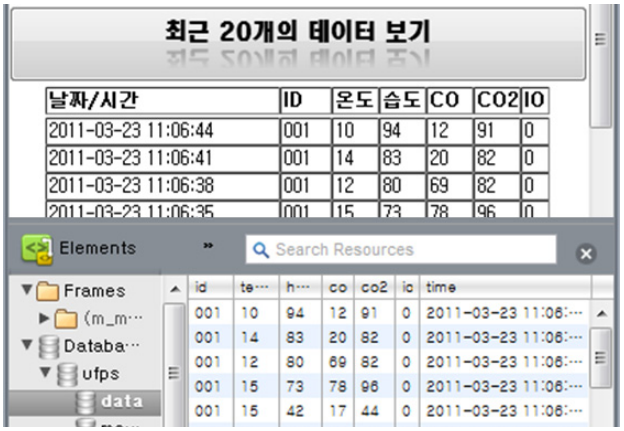
(그림 6) 수신 받은 데이터를 데이터베이스에 저장하는 코드

(그림 6)는 (그림 5)에서 변수에 저장된 값을 서버 측의 MySql 데이터베이스의 ufps라는 테이블의 필드에 각각 저장하기 위한 서버 프로그램의 코드이며 역시 C#으로 작성되었다.

	No	ID	Temp	Hum	CO	CO2	IO	Time
	179651	001	10	94	12	91	0	2011-03-23 11:06:44
	179650	001	14	83	20	82	0	2011-03-23 11:06:41
	179649	001	12	80	69	82	0	2011-03-23 11:06:38
	179648	001	15	73	78	96	0	2011-03-23 11:06:35
	179647	001	15	42	17	44	0	2011-03-23 11:06:32
	179646	001	19	37	67	47	0	2011-03-23 11:06:29

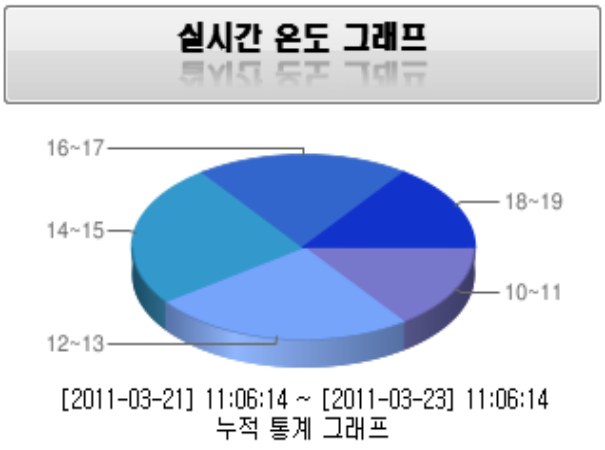
(그림 7) 서버의 MySql 데이터베이스에 저장된 센서 데이터

서버 측의 데이터베이스에 저장된 센서 데이터는 웹을 통한 모니터링 서비스를 제공하는 웹 페이지에 보이게 된다. 이 때 웹 페이지에 보이는 데이터 값은 Web SQL Database에 저장된 값이고, 매 1초마다 서버에서 값을 받아와서 Web SQL Database에 기록되며 실시간으로 모니터링 화면에 반영이 된다. (그림 7)은 서버의 데이터베이스에 있는 데이터를 받아와서 Webkit 기반 웹브라우저인 크롬 브라우저의 Web SQL Database에 저장된 것을 크롬 브라우저의 개발자 도구로 본 그림이다.



(그림 8) 크롬 브라우저의 개발자 도구로 본 로컬 측의 Web SQL Database에 기록된 센서 데이터

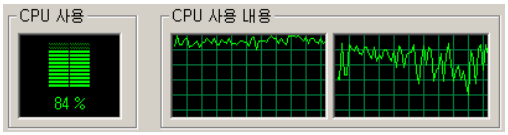
모니터링 시스템에서 제공할 최근 1분간의 값을 보여주거나, 누적된 값으로 통계를 내고 실시간으로 갱신되는 그래프를 보여주기 위하여 기존의 시스템은 AJAX를 사용하여 매 초당 서버 측의 데이터베이스에 접근하여 필요한 센서 데이터를 모두 가져와서 그래프를 그려준다.



(그림 9) Web SQL Database에 저장된 값으로 구한 누적 통계치(Chart API사용)

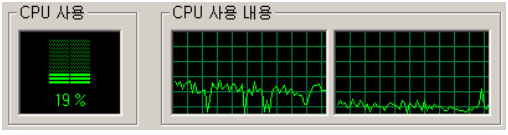
본 연구에서 제안하는 방법은 처음 세션이 열리는 순간에 누적 데이터를 모두 가져와서 Web SQL Database에 저장을 하고, 이후로는 매 초당 서버에 접근하여 새롭게 저장되는 센서 데이터만을 가져와서 Web SQL Database

에 기록하여 서버의 데이터베이스와 로컬 데이터베이스는 세션이 열려있는 동안은 같은 값을 갖게 된다. 이렇게 Web SQL Database에 저장된 센서 데이터를 이용하여 1분간 누적 데이터를 보여주거나, 실시간으로 갱신되는 그래프를 보여준다면 서버에 매 초당 접근하여 통계에 필요한 모든 데이터를 가져오는 방식보다 서버의 부하를 줄일 수 있고, 어떠한 웹 언어보다 모바일 환경에 최적화 된 HTML5를 이용하여 모바일 환경으로의 서비스 제공에 더욱 적합할 것이다.



(그림 10) 서버 측의 데이터베이스 사용 모니터링시 서버의 CPU사용률

(그림 10)에서 보는 바와 같이 실시간 데이터, 최근 20개의 데이터, 그리고 실시간 누적 통계치 그래프를 보여주기 위해 서버 사이드 스크립트에서 서버 측의 MySql 데이터베이스에 쿼리를 보내는 양이 매우 많아 서버의 CPU 사용률이 매우 높다.



(그림 11) Web SQL Database 사용 모니터링시 서버의 CPU사용률

반면 (그림 11)에서 보이는 바와 같이 Web SQL Database를 사용하여 같은 작업을 하는 모니터링 시스템을 사용할 때의 CPU사용률을 보면 현저히 서버의 CPU부하가 적은 것을 확인하였다.

5. 결론

2009년과 2010년의 화재 발생 현황을 보면 최근 화재 발생 빈도가 줄어들고 있지만, 사람이 상시 거주하지 않는 곳에서의 화재 발생은 여전히 많은 비중을 차지하고 있다. 그리하여 비주거 환경에서 화재 방재 시스템은 필수 사항이 되었다고 해도 과언이 아니다. 기존의 화재 방재 시스템에 관한 연구들은 모니터링 서비스를 위한 시스템에 대하여 관심 있게 다루고 있지 않다. 서버가 언제나 쾌적하게 서비스를 제공하기 위해서 가용성이 중요한 문제가 되는데, 서버의 자원이 부족하여 실시간 모니터링이 지속되기 힘들 경우 본 연구에서 제안하는 로컬 데이터베이스를 사용하는 방법을 사용한다면 보다 서버의 부하를 줄이고, 어떠한 웹 언어보다 모바일 환경에 최적화된 HTML5를 이용하여 원활한 서비스를 제공할 수 있는 모니터링 시스템을 구축 할 수 있다. 그리하여 본 연구에서는 Web SQL Database를 사용하여 서버의 부하를 줄이는 모니터링 시스템을 제안하였으며, 실험을 통해 서버의 부하가 낮음을 확인하였다. 하지만, 서버의 데이터를 SQLite 데이터베이스에 저장하는 시간이 조금 오래 걸린다는 문제가 있으며, 보안적인 측면에서 볼 때 기존의 모니터링 시스템은 서버의 보안에만 신경을 쓰면 되는 반면, 제안하는 방식은 서버의 보안에도 신경을 써야 하고 모니터링을 하게 되는 모바일 디바이스 또한 공격에 취약할 수 있다. 만약 전산실과 같은 환경에 설치된 본 시스템의 세션이 열려 있는 동안 공격자에 의해 로컬 데이터베이스 값의 변조가 일어나 화재 상황이 아님에도 관리자가 오인으로 소화기를 터트리는 일이 발생한다면, 값비싼 장비들의 손상으로 이어져 피해가 커질 수도 있다. 따라서 보안 적인 측면을 고려하지 않아도 괜찮을 경우 제한적으로 사용이 가능할 것으로 보인다. 또한 속도의 문제는 추후 Indexed Database가 널리 사용되고 모바일 브라우저에서 이를 지원하게 된다면 간단하게 구조만 바꾸면 더욱 빠르고 간결하게 작동할 것으로 기대되며, Indexed Database의 표준화 책정이 진행되어 모바일 기기의 웹브라우저가 이를 지원하게 된다면, 속도 문제를 일부 개선할 수 있을 것으로 보인다. 이와 관련해서는 향후 연구로 남긴다.

본 연구는 중소기업청에서 지원하는 2009년도 산학연협력 기업부설연구소 지원사업(00037442-3)의 연구 수행으로 인한 결과물임을 밝힙니다.

참고문헌

[1] 소방방재청, “2010년 화재발생현황 분석”

[2] 김영혁 외, “가속도·패턴인식 기술을 이용한 유비쿼터스 화재 방재 시스템 연구”, Vol. 17, No. 1, 한국정보처리학회 춘계학술발표대회 논문집, 2010. 4

[3] 이재수 외, “WSN 환경에서 전송률 향상을 고려한 화재감지 모니터링 시스템 구축에 관현 연구”, Vol. 25, No. 2, 한국조명·전기설비학회 논문지, 2011. 2

[4] 손봉세 외, “유비쿼터스와 통합방재시스템의 유기적 적용방안”, 한국화재소방학회 학술대회 논문집, 2007. 11

[5] W3C, “Web Database”, W3C Working Group Draft 29, Oct 2001

[6] Remy Sharp, “Introducing Web SQL Databases”, html5doctor, 2010

[7] W3C, “Web SQL Database”, W3C Working Group Note 18, Nov 2010

[8] Google, “Google Chart Tools / Image Charts (aka Chart API)”