확장성과 개방성을 지원하는 SCADA 시스템 설계 및 구현

Design and Implementation of SCADA System to Support Scalability and Openness

김 형 일, 이 승 룡, 전 태 웅, 박 영 택
(Hyung-Ill Kim, Sungyoung Lee, Taewoong Jeon, and Young-Tack Park)

Abstracts : The existing SCADA (Supervisory Control and Data Acquisition) system software is usually developed to suitable for the specific hardware platforms. However, as per rapid improvement of computer performance and development of network technology, it is required to support scalability and inter-operability in existing different SCADA systems. In order to meet such requirements, in this paper, we propose a new type of SCADA testbed using Java for electric distribution applications. The system consists of three modules; development support tools, client and server modules. The basic architecture of the proposed SCADA system is similar to existing one, however. we improve the function of MTU and MMI interface to facilitate LAN and WAN environment. Also, the proposed system can deals with alarm and history data by using heterogeneous DBMS. Since the system is built in Java environment, the development cost is cheap and it can support sacalability and portability Our experience can be utilized to develop next generation of small and medium size of SCADA system.

Keywords : SCADA, real-time, scalability, inter-operability, Java, Internet

I. 서론

SCADA란 컴퓨터와 통신기술에 의해 가능하게된 고속, 정확한 정보수집, 처리, 분석, 제어 및 송수신 기능들을 산업 및 원격제어 시스템에 적용하여 경제적이고 안전하게 종합 운용을 하는 시스템을 말한다. SCADA를 기반으로 한 실시간 산업 설비 및 공정 제어 시스템 응용 분야는 고속, 병렬, 분산 처리 및 고장 허용을 가능하게 하는 하드웨어 구조뿐만 아니라 그러한 시스템 구조 하에서 병행 태스크들의 반응 및 처리, 스케줄, 동기화. 통신등을 실시간으로 처리할 수 있는 소프트웨어의 개발을 요구한다. SCADA 시스템이 가져야할 기본적인 소프트웨어 기능으로는 사용자의 편리를 도모하기 위한 GUI(Graphic User Interface), 데이터 수집, 처리, 전달(통신), 고장허용 및 고장복구, 트랜딩. 동적 그래픽 디스플레이, 알람 등이다[2][8][9].

SCADA 시스템은 크게 RTU(Remote Terminal Unit), MTU(Master Terminal Unit), 그리고 이들을 연결해 주는 통신설비로 이루어져 있다. 그림 1에서 RTU는 필드에 분산되어 있는 각종 디바이스로부터 이산 또는 아날로그 신호, 알람, 상태값 등을 입력받은 후 그것들을 저장한다. 그 후 MTU의 요청이 있으면 이러한 값들을 전송해 준다. MTU는 스캔 형식으로 RTU를 호출하고 그들로부터 얻은 정보를 가지고 상황을 판단하여 필요한 명령을 RTU에게 내린다. MTU와 RTU사이의 통신은 유무선 방법을 모두 사용할 수 있다.

SCADA 시스템 소프트웨어는 지금까지 특정한 하드웨어에 적합하도록 개발되어 왔으나 PC 하드웨어 성능 향상과 인터넷과 같은 네트워크 기술, 그리고 CORBA나 Java 플랫폼과 같은 시스템 통합을 위한 미들웨어기술의 발전으로 인하여 이기종 SCADA 시스템간의 통합과 호환성, 그리고 확장성을 잘 지원할 수 있는 새로운 시스템 개발요구가 증대되고 있다. 예를들어 웹브라우저는 높은 이동성을 보장할 수 있고 인터넷 환경에서 Java 플랫폼을 이용하면 확장성과 호환성을 동시에 지원할 수 있다. 이러한 변화는 업무통합 환경 구축이 일반화함에 따라 산업 현장에서도 요구되고 있다.

따라서, 본 논문에서는 향후 확장성과 개방성을 지원할 수 있는 Java기반의 전력계통 응용을 위한 SCADA 소프트웨어 시스템 설계와 테스트베드(testbed) 개발 경험을 기술한다. 개발된 SCADA 시스템은 기존의 SCADA 시스템과 기본 구조는 동일하지만, MTU와 MMI 인터페이스를 확장하여 LAN뿐 아니라 인터넷을 통한 WAN 환경으로 시스템을 확장할 수 있다. 시스템 개발 시 중점을 둔 부분은 첫째, 객체 지향형 설계 방식을 도입하고, 둘째, 통합이 용이하고 개방형 시스템에 적합한 Java 기술을 사용하였으며, 셋째, 인터넷/인트라넷 환경에 적합한 클라이언트-서버 모델을 구축하고, 넷째, 확장성을 고려하여 모듈화된 시스템으로 구성하였으며, 마지막으로, 다른 업무 환경과 쉽게 연동될 수 있도록 표준화 기술을 지원하도록 하였다.

현재 제품화되어 있는 SCADA 시스템 소프트웨어는 크게 PC 기반의 소프트웨어와 Unix 계열의 소프트웨어가 있다. 이중 국내 산업 현장에서 현재 많이 사용되고 있는 시스템은 가격대비 성능이 우수한 PC 기반의 소프

접수일자 : 1998. 12. 11, 수정완료 : 1999. 6 24.
김형일, 이승룡 : 경희대학교 전자정보학부
전태웅 고려대학교 자연과학대학 전산학과
박영택 : 숭실대학교 정보과학대학 컴퓨터학부
※ 본 논문은 한국과학재단 특정개발연구과제(과제번호 : KOSEF 95-0100-07-01-3)의 지원에 의해서 작성되었습니다.

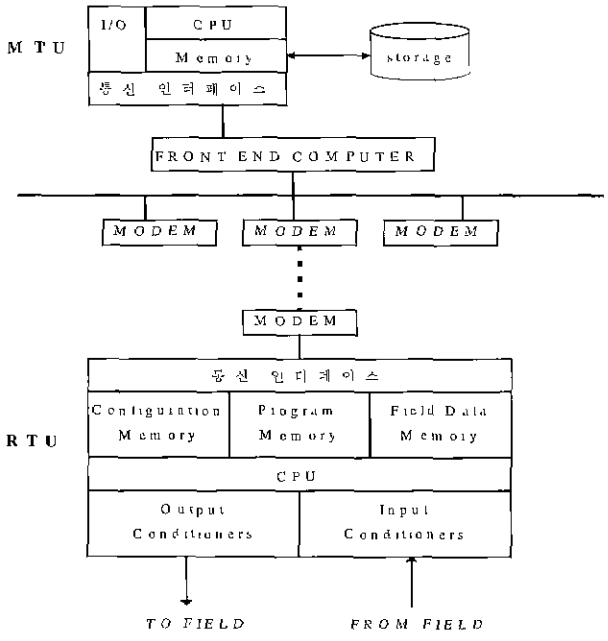

그림 1. SCADA 시스템 구조.
Fig. 1. SCADA system architecture.

트웨어들이다. Citect[3]은 PLC(Programable Logic Control)를 이용한 PC 기반의 소규모 계장 시스템에 적합한 제품이다. Citect의 특징은 편리한 기능과 거의 모든 PLC 제품을 연동 할 수 있는 드라이버 제공에 있으며, 무엇보다도 가격 대 성능비가 뛰어난 제품이다. 하지만, DBMS 접속 기능이 매우 미약한 단점을 지니고 있다. Wizcon은 이스라엘의 PC Soft International 만든 PC 기반의 SCADA 시스템이다[12][13]. PC 운영체제로 DOS, OS/2, Windows95/NT를 사용할 수 있으며 최근에 Wizcon for Internet도 개발되었다. 이는 기존의 PC 기반의 Wizcon 소프트웨어를 Java를 이용하여 호환성 있게 확장한 것이다. Web@aGlance[14]는 Sun 사의 100% Java로 개발한 프로그램으로 기존의 DCS(Distributed Control System), HMI(Human-Machine Interface), SCADA 시스템의 인터페이스를 웹으로 확장할 수 있도록 되어 있다. Auspex[1] 사는 현재 자사의 RCS-7 제품군으로 Java 기반의 SCADA 소프트웨어를 개발하고 있는 중이다. 그러나 이들 모두가 기존의 시스템에 부가적인 확장 모듈로 Java를 사용하는 것에 비하여 본 논문에서 개발한 소프트웨어는 프레임워크부터 100% 자바로 구성되어 있는 특징을 지닌다.

본 논문의 구성은 다음과 같다. 2 장에서는 제안한 SCADA 시스템 모델과 설계방법에 대하여 기술하며, 3 장에서는 시스템의 구현 방법을 개발지원도구, 서버모듈, 클라이언트 모듈의 관점에서 살펴본다. 그리고, 마지막으로 4 장에서 결론을 맺는다.

II. 시스템 모델 및 설계

1. 시스템 모델

제안된 SCADA 시스템은 100% Java 프레임워크를 이용하여 설계되었다. 시스템은 그림 2와 같이 개발 도구, 클라이언트와 서버로 구성되어 있다.

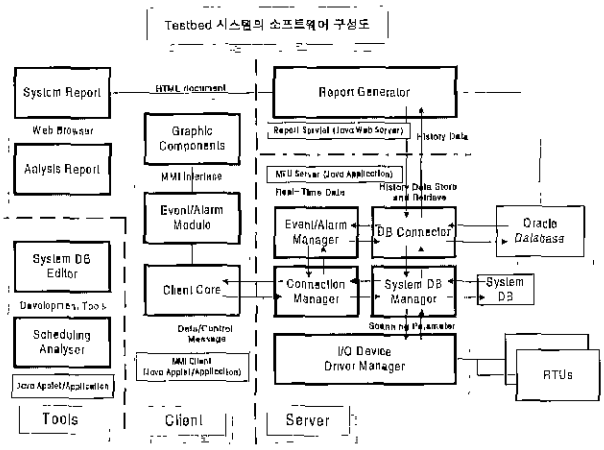

그림 2. 제안한 SCADA 시스템 모듈 구성도.
Fig. 2. Proposed SCADA system module architecture.

개발 도구는 스케줄링 분석기와 SCADA 시스템 DB 에디터로 구성되어 있는데 스케줄 분석기는 실시간 스케줄러의 동작을 시뮬레이션하여 실시간 태스크가 주어진 스케줄 환경에서 제대로 동작하는지를 확인하는 기능을 갖는다. 시스템 DB 에디터는 SCADA 시스템의 각종 파라메타와 구성을 정의하는 것으로 서버가 기동할 때 이용하게 된다. 서버는 시스템 운영에 핵심적인 역할을 하는 것으로 Java I/O 드라이버를 이용하여 RTU로부터 데이터를 획득하고, 이를 데이터베이스에 저장하는 기능을 담당한다. MTU 서버는 시스템 DB 관리자(System DB Manager), I/O 디바이스 드라이버 관리자(I/O Device Driver Manager), DB 연결자(DB Connector), 이벤트/경보 처리자(Event/Alarm Manager)로 구성되어 있다. RTU는 I/O 디바이스 드라이버 관리자를 통하여 직렬포트로 통신 하게 되며 MMI 클라이언트의 요청에 따라 RTU에 제어 명령을 보내거나, RTU로 부터 스캐닝되는 정보를 데이터베이스에 저장하도록 구성되어 있다. 데이터베이스에 저장된 정보들은 보고서 생성기를 통하여 웹 브라우저에서 확인할 수 있도록 되어 있다. MMI 클라이언트는 서버에 접속되어 GUI를 통하여 모니터링과 관리 기능을 제공하는 것으로 그래픽 컴포넌트 모듈, 이벤트/경보 처리 모듈, 서버 접속 모듈로 나뉜다. MMI 클라이언트는 웹 브라우저 또는 Java 애플리케이션으로 동작하도록 되어 있다. 이처럼, 제안한 SCADA 시스템은 클라이언트와 서버 기능을 명확하게 나누고, 네트워크를 통하여 접속할 수 있도록 설계하였기 때문에 인터넷 또는 인트라넷 환경에 적합한 기능을 제공할 수 있다.

2. 개발 지원 도구 설계

2.1 스케줄링 분석기

스케줄링 분석기는 수행하여야 할 태스크에 대하여 실제 실시간 스케줄링이 가능한지를 사전에 판단하므로써 시스템 개발 과정 시 스케줄링 오류에 의한 문제들을 제거하는 기능을 수행한다. 입력받은 태스크 집합과 실시간 스케줄링 알고리즘을 선택하면 스케줄링이 가능한

지 아닌지를 판단하여 결과를 화면에 보여주며 이를 통하여 가장 적당한 태스크 집합을 만들어 내고 이들을 시스템 설계에 사용한다. 스케줄링 분석기는 다양한 스케줄링 정책을 선택할 수 있고, 새로운 태스크를 추가하거나 삭제할 때 스케줄링 정책에 어떤 영향을 주는지 알 수 있으며, 사용해야 할 리소스를 정의함으로써 프로세스 간 통신에 대한 분석이 가능하다. 또한, 스케줄링 분석기는 Java와 IFC[4]를 기반으로 제작되었기 때문에 Java를 지원하는 모든 플랫폼에서 사용이 가능하다.

스케줄링 분석기는 Java를 기반으로 한 IFC 인터페이스를 장착하였으며, 분석기 클래스가 메인 클래스로, Task 클래스, Scheduler 클래스, MonitorView 클래스로 구성되어 있다. 메인 클래스인 Analyzer 클래스는 IFC View 클래스를 계승한 것으로 비가시 클래스인 Task 클래스와, Scheduler 클래스, 가시 클래스인 MonitorView 클래스를 포함하고 있으며(그림 3), 이들 객체간의 관계는 그림 4와 같다. 메인 클래스는 파일로부터 태스크 집합의 정보를 읽어오기 위하여 FileViewer 클래스를

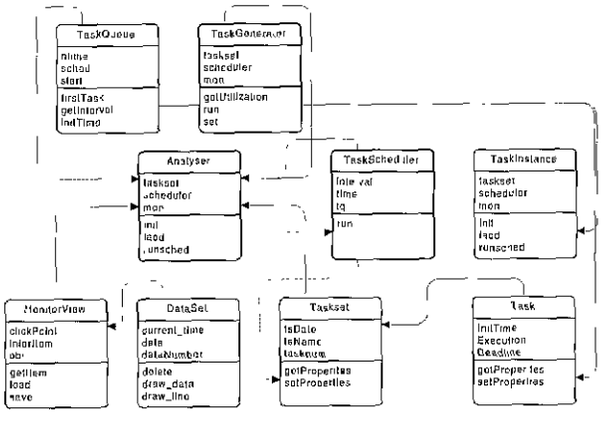

그림 3. 스케줄링 분석기 클래스 구조도.
Fig. 3. Class structure diagram of the scheduling analyzer.

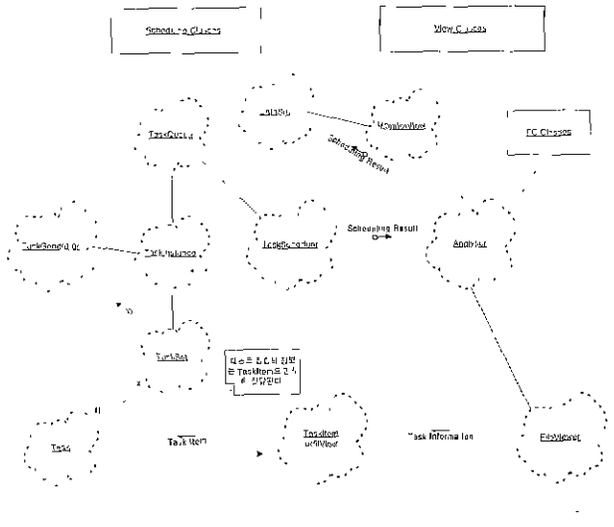

그림 4. 스케줄링 분석기 객체 관계도.
Fig. 4. Object diagram of the scheduling analyzer.

이용하며 화면상의 편집을 위해서 TaskView 클래스를 이용한다. 태스크의 정보가 파일로부터 읽어오거나 태스크 정보 화면에서 편집이 되면 TaskSet 클래스에 의하여 태스크 정보가 변경된다. 변경된 태스크 집합의 정보는 태스크 이벤트 클래스에 의하여 인스탄스화되어 태스크 큐 클래스에 넘겨지고 스케줄러 클래스는 태스크 큐 클래스로부터 태스크를 입력받아 이를 분석기 클래스로 전송하게 된다. 분석기 클래스는 이를 모니터 뷰 클래스로 전달하고 모니터 뷰 클래스는 정보를 데이터 세트 클래스에 저장하고 화면에 스케줄링 정보를 나타낸다.

2.2 시스템 DB 에디터

시스템 DB 에디터는 시스템 DB를 설정하고 변경하는 모듈이다. 시스템 데이터베이스는 모든 상태, 아나로그, 계산 포인트와 각종 시스템 관련 데이터를 포함하고 있으며, 이를 통하여 다양한 환경의 SCADA 시스템을 구성할 수 있게 되어 있다. 시스템 DB에디터의 구성은 그림 5와 같이 각각의 수집 포인트를 정의하거나, 시스템의 파라메타를 정의할 수 있는 다이얼로그를 가지고 있고, 이에 대한 실제 구성 데이터는 시스템 DB 제어 모듈을 통하여 시스템 DB에 저장된다

3. 서버 모듈 설계

서버 모듈은 SCADA 시스템의 핵심 모듈로 MTU 서버와 보고서 작성을 위한 웹 서버 모듈로 구성되어 있다. MTU 서버는 실시간 데이터를 취득하고, 이를 저장하거나, MMI 클라이언트에게 전송하는 역할을 하며, 기본 기능은 실시간 데이터의 취득, 취득한 데이터의 정보 분석 및 처리, 모니터링 데이터 제공(MMI 연결 기능) 등이다. 이를 구현하기 위하여, MTU 서버는 그림 6과 같은 프로그램 모듈로 구성되어 있다.

시스템 DB 관리자는 시스템 DB로 부터 필요한 메모리를 할당하고, MTU 서버의 초기화를 담당하는 부분이다. 사용자는 시스템 DB 에디터를 통하여 시스템 DB를 조작할 수 있다. 시스템 DB는 시스템 DB 관리자를 통하여 MTU 서버의 초기화에 이용된다. I/O 디바이스 관리자는 System DB를 통하여 사전에 정의되어 있는 RTU 연결 디바이스를 로딩하고 이를 동작시키는 역할

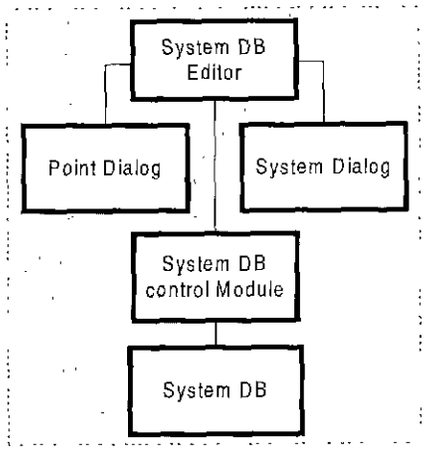

그림 5. 시스템 DB 에디터의 구성.
Fig. 5. Construction of the system DB editor.

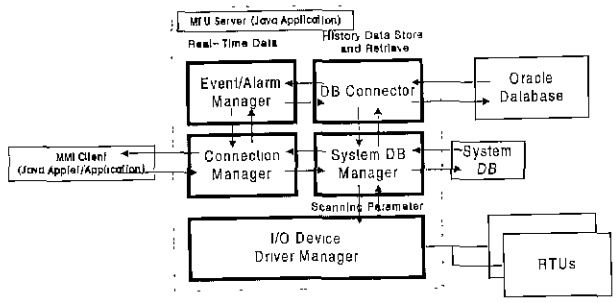

그림 6. MTU Server 구성도.
Fig. 6. Construction diagram of the MTU server.

을 한다. I/O 디바이스 드라이버를 통하여 얻어진 스캐닝 정보는 DB 연결자(DB connector) 또는 이벤트/경보 관리자(event/alarm manager)를 통하여 클라이언트에 전달되거나 DB에 저장된다. DB 연결자는 오라클 DBMS와 연결을 지원해준다. 여기에는 각종 시스템 DB를 포함하여, 이벤트, 경보, 스캐닝 정보등의 이력 데이터를 저장 및 검색할 수 있도록 도와준다. 표준 JDBC를 이용하기 때문에 오라클 DBMS 뿐 아니라 다른 DBMS와의 연결이 용이하다.

이벤트/경보 관리자는 이벤트가 MMI 클라이언트로부터 발생하였을 경우에 이를 처리하거나, 경보가 발생하였을 경우에 연결 관리자를 통하여 TCP 소켓으로 접속되어 있는 MMI 클라이언트에게 전달해주고, DB 연결자를 통하여 이를 저장하는 역할을 한다.

연결 관리자는 MMI 클라이언트의 접속을 허가하고 필요한 데이터를 전송해주는 역할을 한다. 이는 MMI 클라이언트의 접속 허가를 판단하고, 이벤트/경보를 처리한다. 연결이 되었을 경우에 이에 대한 결과를 로그 파일로 만들어준다. MTU 서버에 포함되지 않는 서버 모듈인 리포트 생성기(report generator)는 보고서 작성을 위한 웹 모듈로 Java 서블릿[10]을 통하여 표준 HTML로 이력 데이터 및 전문가 시스템의 결과를 리포팅 해주는 시스템 보고서를 말한다(그림 2 참조). 리포팅 도구는 예를 들어, 시보, 일보, 월보 년보 등을 HTML 문서로 작성하여 웹 브라이져 상에서 확인할 수 있는 기능을 포함한다.

3.1 시스템 DB 관리자(system DB manager)

시스템 DB 관리자는 시스템 DB 에디터를 통하여 변경된 시스템 파라메타를 읽어서 이를 통하여 MTU 서버의 초기화와 시스템 구성을 담당하는 모듈이다(그림 7).

이때, 시스템 DB는 Java의 serialization 기능을 이용하여 저장되고 로딩된다. 또한, 시스템 DB 관리자는 I/O 장치 구동 관리자로부터 얻어진 스캐닝 정보를 메모리에 저장하는 역할을 담당한다. 따라서, 시스템 DB 관리자는 MTU 서버의 핵심 엔진에 해당된다. 시스템 DB 관리자의 동작 흐름은 그림 7과 같이 초기화 과정과 장치 관리자 구동 그리고, 연결 관리자 통신 등으로 나눌 수 있다.

3.2 I/O 디바이스 드라이버 관리자(I/O device driver manager)

I/O 디바이스 드라이버 관리자는 I/O 디바이스 드라

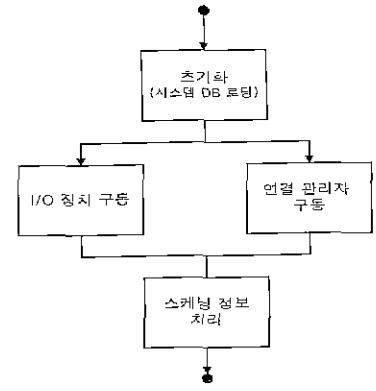

그림 7. 시스템 DB 관리자의 동작 흐름도.
Fig. 7. Operational flow diagram of the system DB mamager.

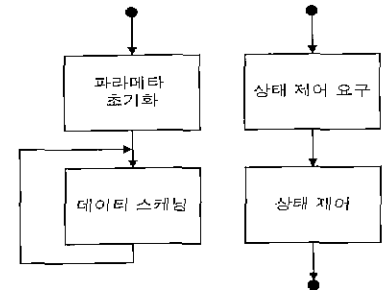

그림 8 I/O 디바이스 드라이버의 동작 흐름도.
Fig. 8. Operational flow diagram of I/O device driver.

이버와 이벤트/경보 관리자를 연결해주는 역할을 한다. I/O 디바이스 드라이버는 실시간 데이터 취득을 위한 하드웨어와 접속하여 데이터를 얻어오는 기능을 한다. I/O 디바이스 드라이버는 취득 데이터 정보 분석 및 처리 모듈과 RTU의 데이터를 연결시켜주는 역할을 한다. 그림 8은 I/O 디바이스 드라이버의 동작을 나타낸다. I/O 디바이스 드라이버의 동작은 속도, 패리티, 비트들, 스캐닝 주기 및 범위를 설정하는 초기화 단계와 실제 데이터를 읽어들이는 스캐닝 단계, 특정 위치의 상태를 제어하는 단계로 나누어진다.

테스트베드 시스템에서 사용된 RTU는 직렬 포트와 연결되어 Harris Protocol[6]을 기반으로 데이터 통신이 이루어지는데, 크게 제어 포트와 수집 포트로 나뉘어진

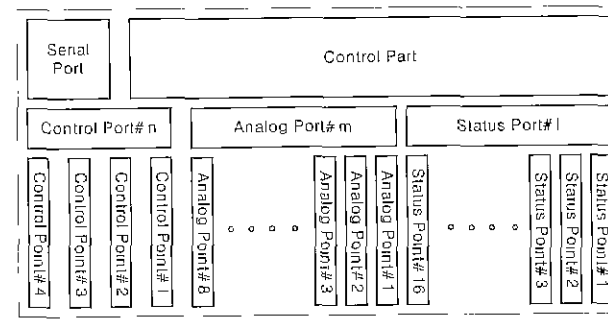

그림 9. RTU의 하드웨어 구조.
Fig. 9. Hardware structure of RTU.

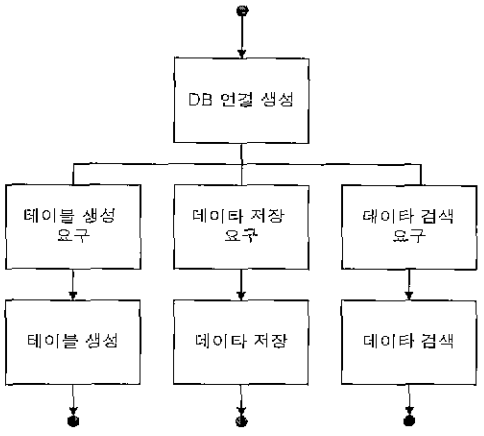

그림 10. DB 연결자의 동작 흐름도.
Fig. 10. Operational flow diagram of DB connector.

다. 수집 포트는 다시 아나로그 수집 포트와 상태 수집 포트로 나눌 수 있다. 테스트베드 시스템에서 사용된 RTU의 구조는 그림 9와 같으며, 제어 포인트는 총 4개, 아나로그 포인트는 총 8개, 상태 포인트는 총 16개로 구성되어 있다.

3.3 DB 연결자(DB connector)

DB 연결자는 시스템 DB에 저장되어 있는 RTU 스캔 정보와 정해진 개별 포인트에 따라 획득된 값들을 DBMS를 통하여 저장하거나, MMI 또는 보고서 생성자로 부터의 이력 데이터 요구를 처리하는 모듈이다. DB 연결자는 MTU 서버 안에서 동작하고, 보고서 생성자의 서블릿 모듈 안에서 동작한다. DB 연결자의 동작 흐름은 MTU 서버의 동작과 보고서 생성자의 작동과 비슷하며, 시스템 DB 관리자가 요구하는 DB 연결을 생성하고, I/O 장치 관리자나 이벤트/경보 관리자로 부터 요구되는 데이터의 저장이나 리포트 생성자로 부터 요구되는 데이터의 검색을 처리하게 된다. DB 연결자의 흐름도는 그림 10과 같다.

3.4 이벤트/경보 관리자(event/alarm manager)

이벤트/경보 관리자는 시스템 DB에서 정의된 포인트에 따라 경보가 발생되었을 경우에, 이를 DB에 전달하거나 접속 관리자로부터 얻어진 이벤트를 처리하는 역할을 한다. 이벤트/경보 관리자의 동작은 시스템 DB 관리자로 인하여 접속 관리자와 함께 초기화되고, 계속적으로 시스템 DB를 통하여 알람을 처리하고, 연결 관리자를 통하여 이벤트를 처리하게 된다. 이벤트/경보 관리자의 동작은 그림 11과 같다. 이벤트/경보의 처리 과정은 I/O 디바이스 관리자로 부터 얻어진 이벤트/경보 정보를 이벤트/경보 관리자가 수집하여 이를 메모리와 DB에 저장한 뒤 현재 접속되어 있는 클라이언트에게 필요한 이벤트/경보를 연결관리자에 의해서 전달해주게 된다. 이를 전달받은 클라이언트들은 자신의 화면에 출력하게 되는 것이다.

3.5 연결 관리자(connection manager)

연결관리자는 접속된 MMI 클라이언트에서 발생된 이

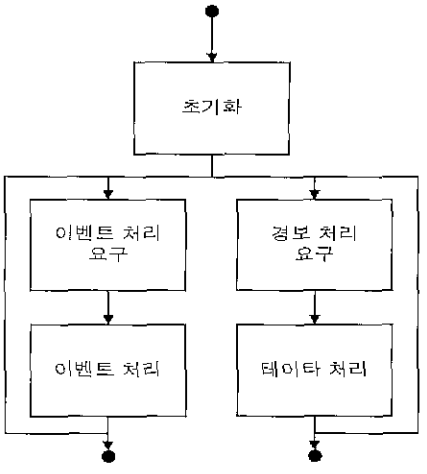

그림 11. 이벤트/경보 관리자의 동작 흐름도.
Fig. 11. Operational flow diagram of event/alert manager.

벤트를 처리하는 역할을 한다. MMI 클라이언트가 하나 이상 접속되었을 경우에는 경보 처리에 대한 동기화가 필요하며 이를 위하여 MMI 클라이언트 접속 모듈을 포함하고 있다.

연결 관리자의 동작 흐름은 시스템 DB 관리자에 의해서 초기화되고, MMI 클라이언트의 접속 요구가 발생하면, 이에 대한 인증 절차를 거쳐서 MMI 클라이언트에게 필요한 데이터를 전달하게 된다. 연결관리자는 MMI 클라이언트로 부터 발생한 이벤트를 전달받아 이벤트/알람 관리자에게 전달하여 처리하게 된다. MMI 클라이언트에게 전달할 데이터는 연결 관리자에 의해서 필터링 작업이 수행된 뒤에 전달된다. 연결 관리자의 흐름도는 그림 12와 같다.

3.6 보고서 생성자(report generator)

보고서 생성기는 특정 아나로그 포인트에 대한 시간 변화에 따른 추이를 관찰할 수 있는 기능을 제공하며, 웹 브라우저를 통한 화면 출력을 담당한다. 따라서, Java

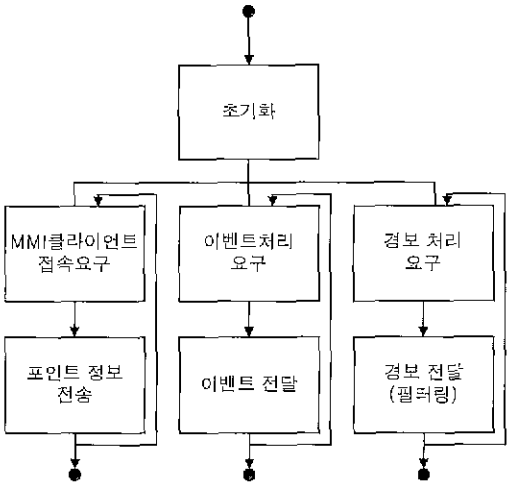

그림 12. 연결 관리자의 동작 흐름도.
Fig. 12. Operational flow diagram of connection manager.

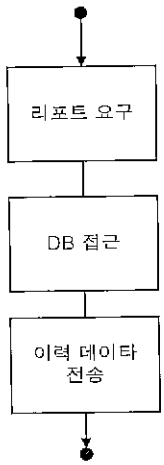

그림 13. 보고서 생성기의 동작 흐름도.
Fig. 13. Operational flow diagram of report generator.

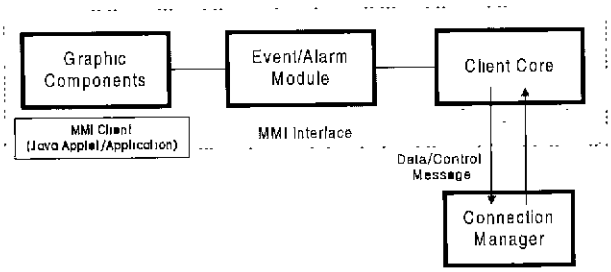

그림 14. MMI 클라이언트 모듈 구성.
Fig. 14. Construction of MMI client modules.

서블릿으로 프로그래밍되어 있다. 보고서 생성기의 동작 흐름은 그림 13과 같이 웹 브라우저의 요구가 있을 경우에 DB 연결자를 통하여 이력 데이터를 DB로 부터 얻어와 화면에 출력하게 된다. 기본적으로 HTML 표준을 따르고, 그래프의 경우에는 그래프 자바 애플릿의 도움을 받아 이를 통하여 화면에 출력하게된다.

4. 클라이언트 모듈 설계

MMI 클라이언트는 원격지에서 SCADA 시스템에 접근하여 모니터링과 제어를 하는 사용자 인터페이스를 말한다 MMI 클라이언트는 메뉴 접근 방식과 아이콘 방식을 사용할 수 있다.

MMI 클라이언트의 기능은 포인트의 선택, 포인트의 제어, 경보 요약 화면, 비정상 요약 화면, 알람 요약 화면, 트랜드 화면 등을 제공해준다. MMI 클라이언트 모듈의 구성은 그림 14와 같이 클라이언트 코어, 이벤트/경보 모듈, 그래픽 컴포넌트로 나뉜다.

4.1 그래픽 컴포넌트(graphic component)

그래픽 컴포넌트는 화면에 포인트를 나타내는 아이콘으로 상태나 값을 나타내는 기능을 하는 컴포넌트 집합이다. 이들은 다양한 크기와 형태를 가질 수 있도록 설계되었다. 이는 기본 그래픽 컴포넌트 클래스를 확장하여 추상 메소드를 정의함으로써 가능하다. 기본적으로 제공되는 그래픽 컴포넌트에는 개폐기, 변압기, 유압계 등 수십종의 그래픽 컴포넌트를 지원할 수 있으며 추가적으로 확장할 수 있다.

4.2 이벤트/경보 모듈(event/alarm module)

이벤트/경보 모듈은 이벤트 또는 경보 발생 시 이들을 처리하는 기능을 한다. 이것은 경보 리스트를 보여주는 컴포넌트를 포함한다. 이벤트와 경보는 그 레벨에 따라서 색상이 정의되어 있으며, 이에 따라 화면에 표시되는 배경 색상이 달라진다.

4.3 클라이언트 코어(client core)

MMI 클라이언트 코어는 서버 연결과 메뉴 구성등 기본적인 기능을 담당하는 부분이다. MMI 클라이언트가 제공하는 메뉴체계는 간단명료하게 구성되었으며, 쉽게

```
String name;
 // 데이터베이스 이름
Vector stps,   amps,   alps;
 // 수집상태 포인트,
    수집아나로그 포인트, 수집누산 포인트
Vector cstps, camps, calps;
  // 계산상태 포인트, 계산아나로그 포인트,
     계산누산 포인트
Vector rtus;
  // RTU 배열 선언
float deadband;
 // 데드밴드 배열 선언
float smooth;
 // 스무싱 배열 선언
Scale scaletable;
 // 스케일 배열 선언
Trigger triggertable;
 // 트리거 배열 선언
Control controltable;
 // 제어 배열 선언
AlarmLevel alarmtable;
 // 알람 레벨 배열 선언
StateCalculator stscaltable;
 // 상태 계산기 배열 선언
String unittable;
 // 단위 이름 배열 선언
String statenametable;
 // 상태 이름 배열 선언
String ioDrivers;

// 시스템 데이터베이스 크기 지정

int             r__rtu__start__no;
int             r__rtu__total__no;
int             p__rtu__start__no;
int             p__rtu__total__no;
int             sioc__count;
int             scan__block__count;
int             tel__status__point__total__count;

int             cal__status__point__total__count;
int             tel__analog__point__total__count;
int             cal__analog__point__total__count;
int             tel__accmul__point__total__count;
int             cal__accmul__point__total__count;
int             dev__cntrl__point__total__count;
int             trigger__max;
int             console__count;
int             logger__count;
int             trend__scr__count;
int             cal__module__count;
int             report__var__count;
int             report__buffer__size;
int             report__format__count;
int             map__module__count;
int             pen__count;
```

그림 15. 시스템 DB 데이터 구조.
Fig. 15. Structure of system DB data.

변경이 가능하도록 하였다. 클라이언트 코어의 기능은 ① MTU 서버와의 통신, ② 동작 화면의 표현, ③ 경보 리스트 제공, ④ 이벤트 리스트 제공, ⑤ 도움말 및 웹 사이트 접속 기능 제공(단, 웹 브라우저에서 작동할 경우)이다. 클라이언트 코어는 서버와의 통신 처리에 있어서 객체를 직접 직렬화하여 전송하기 때문에 다양한 데이터를 전송할 수 있으며 이를 이용하여 다양한 기능들을 추가할 수 있다.

III. 확장성과 개방성 지원을 위한 SCADA 시스템 구현

본 논문에서 제안된 SCADA 시스템을 구현하기 위하여 Java 프로그래밍 언어를 사용하였으며, 객체지향 소프트웨어 모델링 기법에 따라서 각 모듈을 세분화하고 상세화 하였다. 본 논문에서 제안하는 SCADA 시스템은 Java를 지원하는 모든 플랫폼에서 동작이 가능하다. 그리고, MMI 클라이언트와 MTU 서버를 기능적으로 나누었으며, 웹 환경에 적합하도록 설계하여 웹 브라우저 환경에서 MMI 클라이언트 기능을 제공할 수 있도록 하였다. 또한, 호환성이 뛰어난 동적인 GUI 인터페이스를 제공하는 Java 플랫폼을 이용하였기 때문에 이기종 SCADA 환경에서 동일한 화면을 얻을 수 있다.

1. 개발 지원 도구 구현

시스템 DB 편집기는 SCADA 시스템의 구성 데이터에 대한 정보를 편집하는 역할을 한다. 시스템 DB는 모든 상태, 아나로그, 계산 포인트와 각종 시스템 관련 데이터를 포함하고 있다. 이를 통하여, 다양한 환경의 SCADA 시스템을 구성할 수 있게 되어 있다. 시스템 DB는 Java의 직렬화를 통하여 디스크에 저장되고 로딩된다. 시스템 DB의 데이터 구조는 그림 15와 같다.

시스템 DB 관리자의 초기화면은 그림 16과 같다. 상위 메뉴바를 통하여 시스템, 포인터, 공동 정의, 자료 취득, 장치 정의, 애플리케이션에 관련된 데이터 편집 화면을 호출할 수 있다.

그림 17은 상태 포인터 정의 화면을 나타내는데, 포인터 정의 메뉴 수집 상태를 호출하면 화면이 나타난다. 포인터의 정의를 편집하는 방법은 일반적인 데이터베이스 제어 화면과 유사하다. 아래 버튼을 통하여 데이터 레코드를 움직일 수 있으며 새로운 포인터를 입력할 수 있다.

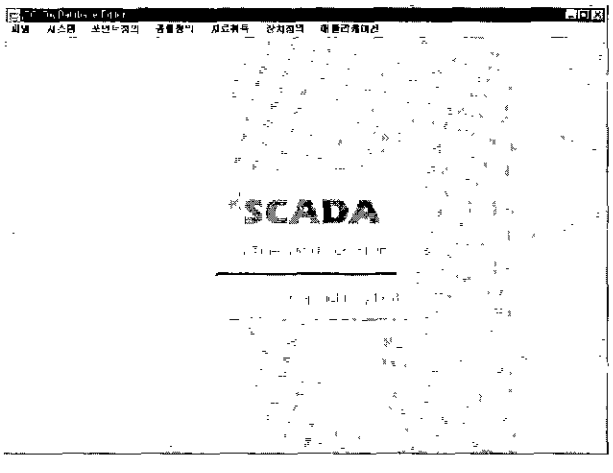

그림 16. 상태 포인터 정의 화면.
Fig. 16. Screen-shot of defining state pointer.

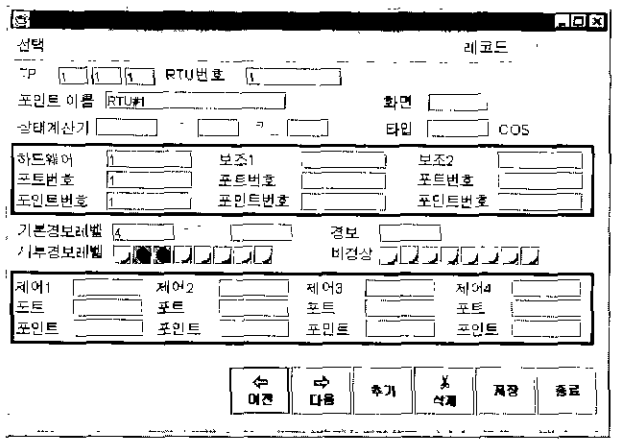

그림 17. 시스템 DB 관리자의 초기 화면.
Fig. 17. Screen-shot of system DB manager.

스케줄 분석기는 애플릿으로 제작되었으며 appletviewer 또는 웹 브라우저를 이용하여 실행할 있다(그림 18).

스케줄 분석기는 실시간 태스크들의 집합을 정의하면 그 태스크 집합의 스케줄 가능성을 분석하고 이를 시뮬레이션 하여 화면상에 보여주는 것으로 지원되는 스케줄 알고리즘은 주기가 짧은 태스크에게 우선권을 주는 rate monotonic algorithm[15]과 가장 빠른 마감시간을 갖는 태스크에게 우선권을 주는 earliest deadline first algorithm[15]을 지원한다.

그림 18의 화면 윗 부분은 태스크가 실행되는 모습을 볼 수 있으며, 아랫 부분은 태스크 집합에 대한 정보를 나타낸다. 태스크 집합의 정보는 왼편의 아이콘을 통하여 수정, 추가, 삭제, 저장 될 수 있다. 화면 오른쪽에는 스케줄 메시지가 나타나는 결과 출력 부분이 된다.

2. MTU 서버 구현

MTU 서버는 Java 애플리케이션으로 동작한다. MTU 서버가 실행되면 시스템 DB 편집기를 통하여 정의된 시

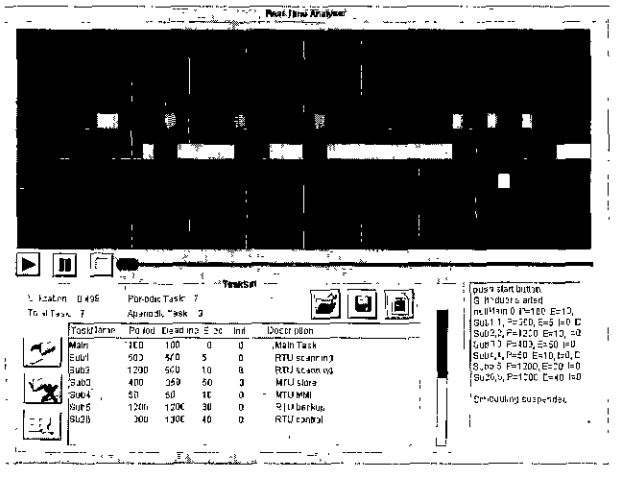

그림 18. 스케줄링 분석기 실행 모습.
Fig. 18. Screen-shot of scheduling analyzer.

그림 19. MTU 서버 실행화면.
Fig. 19. Screen-shot of MTU server.

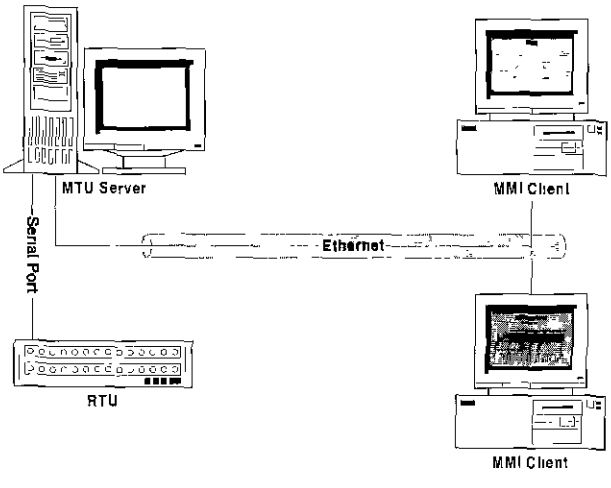

그림 20. MTU 서버와 MMI 클라이언트의 물리적 구성.
Fig. 20. Physical construction between MTU server and MMI clients

스템 DB를 로딩하고 이를 이용하여 시스템을 초기화 한다. I/O 디바이스 드라이버 관리자를 통하여 RTU와 통신을 시작하고, 각종 포인트에 대한 스캐닝을 시작하게 된다. 그림 19는 MTU 서버를 실행시킨 모습이다.

MTU 서버는 I/O 디바이스 드라이버 관리자를 통하여 RTU와 통신을 하게 된다. MTU 서버에서 RTU는 하나의 *실제 RTU 드라이버와 2개의 가상 RTU*로 구성되어 있다. RTU의 상태 데이터는 0 또는 1의 값으로 나타나고 아나로그 데이터는 0부터 1024 사이의 값을 나타낸다. 아나로그 데이터는 데이터 보정을 통하여 부동소수점형으로 변환된다. 그림 20은 MTU 서버와 MMI 클라이언트의 물리적 구성을 나타낸다

MTU 서버와 MMI 클라이언트는 인터넷/인트라넷 망을 통하여 서로 접속하게 되며, MTU 서버는 RTU와 직렬 포트를 통하여 접속해서 데이터를 교환한다. 이러한 역할은 I/O 디바이스 드라이버 관리자에 의해서 수행되는데 기본적인 I/O 디바이스에 대한 정의는 그림 21과 같다.

```
public interface IODevice extends Runnable {
   public void init();
   public IOConfig getIOConfig();
   public
   void addChangeListener(ChangeListener chg);
   public int getValues(int rtu__no, int port);
   public int getValue(int rtu__no, int port,
                                    int point);
}
```

그림 21. I/O 디바이스를 정의한 인터페이스.
Fig. 21. *Interface of defined I/O devices.*

MTU 서버와 MMI 클라이언트 사이에는 네트워크를 통하여 자체적인 프로토콜을 이용한 데이터를 교환하는데 기본적으로는 자바 직렬화를 이용하여 객체들을 주고받도록 되어 있다. 따라서, MTU 서버와 MMI 클라이언트 사이에 전달되는 이벤트와 경보는 모두 객체로 되어 있다. 이벤트객체는 MMI 클라이언트와 MTU 서버 간의 통신을 통하여 메시지로 전달되고, 이를 받아서 이벤트/경보를 처리를 하게 된다. 보고서 생성기는 DBMS에 저장된 이력 데이터를 HTML 코드로 출력해주는 역할을 한다. 이력 데이터의 저장은 DB 접속자를 통하여 MTU 서버에서 이루어진다. JDBC[5][7]를 통해 DBMS에 접속하는 DB 접속자는 실시간으로 발생하는 데이터를 클라이언트 요청과는 상관없이 DBMS에 접속하여 계속 데이터를 추가하게 되어 있다. 이렇게 함으로써 항상 주기적으로 이력 데이터를 축적할 수 있다 이력데이터의 테이블 구조는 시간과 데이터로 이루어진 단순한 구조를 갖는다. 이력 데이터는 입출력이 빈번하므로 테이블의 구조가 단순한 것이 질의의 성능 향상에 도움이 된다. 이력 데이터는 일반적인 SQL 문을 이용하여 질의 요청을 할 수 있는데, 새벽 1시부터 새벽 2시까지의 1시간 동안의 각 부분의 평균값은 다음과 같은 SQL 문에 의해서 얻어질 수 있다[11]. 여기서 시간 단위는 초이다.

```
SELECT AVG(), AVG(RTUA1), AVG(RTUA2),
       AVG(RTUA3)
FROM histdata WHERE puttime
     BETWEEN 3601 AND 7200
```

이것을 JDBC를 이용하여 Java로 기술하면 다음과 같다.

```
// 각 시스템의 평균 부하의 변수를 정의한다.
   String rtua1, rtua2, netutl, pgrate;
// SQL문을 실행하여 그 결과를 얻는다.
   Statement stmt = con.statement();
   Result rs = stmt.executeQuery
        ("SELECT AVG(RTUA1), " +
         "AVG(RTUA2), AVG(RTUA3) " +
         "FROM histdata WHERE puttime
              BETWEEN 3601 AND 7200");
// 얻어진 질의 결과를 변수에 입력한다.
         rtua1 = rs.getString(1);
         rtua2 = rs.getString(2);
         rtua3 = rs.getString(3);
         rtua4 = rs.getStirng(4);
```

표 1. HISDATA 테이블의 구조.
Table 1. Structure of HISDATA table.

FIELDNAME	TYPE	DESCRIPTION
PUTTIME	INTEGER	데이터가 생성된 시간
RTUD1	INTEGER	RTU#1-1-1의 상태
RTUD2	INTEGER	RTU#1-1-2의 상태
RTUD3	INTEGER	RTU#1-1-3의 상태
RTUD4	INTEGER	RTU#1-1-4의 상태
RTUD5	INTEGER	RTU#1-1-5의 상태
RTUD6	INTEGER	RTU#1-1-6의 상태
RTUA1	INTEGER	RTU#1-2-1의 상태
RTUA2	INTEGER	RTU#1-2-2의 상태
RTUA3	INTEGER	RTU#1-2-3의 상태

표 1은 이력 데이터를 위한 데이터베이스 테이블 구조이다.

JDBC를 이용하는 방법은 크게 2-tier와 3-tier로 나누어지는데, 본 연구에서 사용한 DBMS 연동은 애플릿이 직접 DBMS에 접속하는 것이 아니라 자바 서블릿과 DBMS가 연동되고 이것의 결과값을 웹 브라우저 또는 애플릿(애폴리케이션)을 통하여 이용하는 것으로 3-tier 방법으로 분류될 수 있다.

3. MMI 클라이언트 모듈 구현

전력 계통 응용을 지원하는 MMI 클라이언트는 원격지에서 MTU 서버에 연결하여 MTU에서 수집된 원격지 데이터를 그래픽 화면을 통하여 모니터링하거나 특정 포인트의 제어를 가능하게 하는 기능을 한다. MMI 클라이언트의 초기화면은 그림 22와 같다.

전체 메뉴는 2개의 변전소 화면과 이벤트/경보 화면, 도움말 및 웹 사이트로 구성되어 있다. 제1변전소를 선택하면 그림 23과 같은 화면이 나오면서 원격지 정보들을 살펴볼 수 있다.

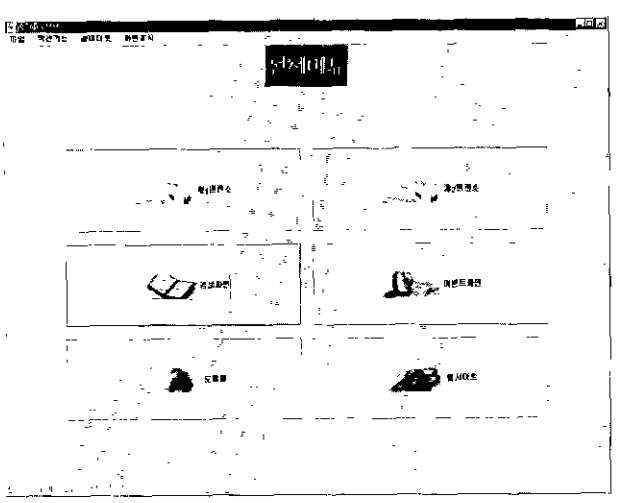

그림 22. MMI 클라이언트 메뉴 화면.
Fig. 22. Screen-shot of MMI client menu.

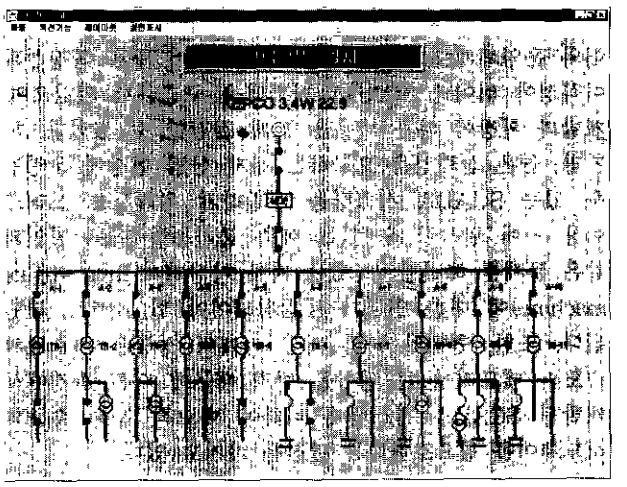

그림 23. 제1변전소에 대한 상황을 감시하는 예.
Fig. 23. Monitoring of the first power plant.

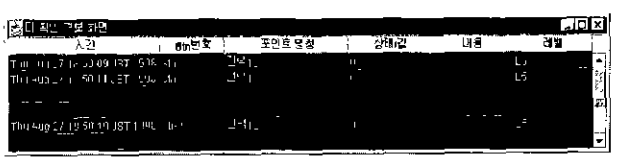

그림 24. 미확인 경보 화면.
Fig. 24. Screen-shot of unidentified alert.

화면은 16개의 접점으로 구성되어 있으며, 이들 정보는 실제 RTU를 통하여 전달되는 정보와 가상 RTU를 통하여 전달되는 정보로 실시간으로 화면에 반영되어 open 또는 close 상태를 표현해준다. 여기서, 개별 접점 또는 값들은 사전에 개별 포인트로 정의되어 있으며, 각각 SComp 클래스를 기본 클래스로 확장하여 구현된다. 경보/이벤트 화면은 실시간으로 발생되는 경보에 대한 정보를 화면에 표시하는 기능을 한다. 그림 24는 미확인 경보 화면의 예이다.

경보/이벤트 화면은 경보 등급에 따라 색상이 달라지며, 경보 등급 0(L0)과 경보 등급 1(L1)의 경우는 화면에서 깜빡임으로 표시되어 긴급한 경보임을 사용자에게 알려준다. 미 확인 경보는 사용자의 선택에 따라 다시 확인 경보 화면에 전달되고, 이러한 사용자 선택은 MTU 서버에게 전달되고 다른 MMI 클라이언트에게 이벤트로 전달되어 처리된다. 경보/이벤트 화면에 대한 처리를 위한 데이터 모델은 AlarmModel 클래스에 정의되어 있다. AlarmModel 클래스에 정의된 데이터에 따라 경보/이벤트 리스트 화면에 표현이 되며, 경보 등급에 따라 색상과 깜빡임에 대한 처리는 JFrame을 확장한 AlarmList 클래스에서 처리한다.

이와 같이 MTU 서버와 MMI 클라이언트는 실시간으로 데이터를 주고받으면서 원격지의 상황을 감시하고 제어할 수 있다. 본 논문에서 제안된 SCADA 시스템은 정상적으로 동작하는 RTU를 연결하여 실제 상황과 유사한 환경을 구현 하였으며, 인터넷 환경이라면 어디서든 MMI 클라이언트를 통하여 감시할 수 있다. 이러한 구현을 통하여 하드웨어 또는 운영체제에 종속적이었던 개발

환경을 벗어나서 Java 플랫폼을 이용하여 다양한 하드웨어와 운영체제를 동시에 지원할 수 있어 향후 확장성과 이식성이 뛰어난 SCADA 시스템을 구축하게 되었으며, 개발과 확장에 따른 비용을 크게 절감할 수 있었다. 하지만, 높은 신뢰성을 요구하는 실제 산업 현장에 운영될 수 있는 SCADA 시스템을 개발하기 위해서는 쓰레드 이용 시에 발생할 수 있는 예측 불가능한 속도 지연과 같은 문제를 해결할 수 있는 자바 가상기계(Java virtual machine)에 대한 연구가 뒷받침되어야 할 것으로 판단된다. 또한, 현재의 자바 가상 기계의 성능은 시스템의 속도와 메모리에 큰 영향을 받기 때문에 다른 환경에서 개발된 SCADA 시스템에 비하여 비교적 높은 수준의 하드웨어를 필요로한다는 단점을 가지고 있다.

IV. 결론

본 논문은 향후 확장성과 개방성을 고려한 Java기반의 전력계통 응용 SCADA 시스템의 설계와 테스트베드 개발 경험을 기술하였다. 현재 소프트웨어의 발전 방향은 인터넷의 등장과 더불어 이동성과 호환성을 만족하는 쪽으로 나아가고 있다. 이러한 새로운 환경은 웹 브라우저를 통한 인터페이스의 통합과 Java와 같은 동적 환경에 적응이 용이한 소프트웨어 프레임을 요구하고 있다. 본 논문에서 제안한 SCADA 시스템 소프트웨어는 이러한 요구 사항에 적절한 해결책을 제시하였으며, 100% Java 프로그래밍 언어를 사용하여 높은 호환성을 제공하였다.

개발된 SCADA 시스템은 기존의 SCADA 시스템과 기본 구조는 동일하지만, MTU와 MMI 인터페이스를 확장하여 기존의 LAN 환경 뿐 아니라 인터넷을 통한 WAN 환경으로 확장할 수 있다. 또한, 이기종간의 DBMS를 이용하여 경보 및 이력 데이터 처리가 가능하여 이식성이 있으며, 기존의 SCADA 시스템의 저차원 수준의 I/O를 유지하면서 MMI 대체가 가능하고, 웹 환경의 인트라넷에 쉽게 적용이 가능하다.

현재 SCADA 시스템을 운영하고 있는 산업 현장에서는 웹 브라우저를 통한 감시 기능 제공, 현장 감시를 위한 동영상 지원, GIS 시스템 지원 및, 체계적인 도면 관리 시스템 지원 등을 요구하고 있으며, 본 연구 결과를 SCADA 시스템의 웹 확장 모듈에 적용하여 웹 브라우저를 통한 원격감시 및 제어 기능을 제공할 수 있는 응용 제품을 개발할 수 있을 것으로 판단된다.

참고문헌

[1] Auspex 홈페이지, http://www.auspex-inc.com
[2] 최병철, 강창수, "SCADA/RTU의 기술동향," pp. 25-31, *제어계측, 1996년 12월호.*
[3] Citect User's Guide For Windows NT, Windows 95, Citect (C).
[4] IFC 프로그램 가이드, *http://developer.netscape.com/library/ifc/documentation/cadmium/developerguide/index.html*
[5] Java DataBase Connection Relase 1.2, JavaSoft, Jun., 1997.
[6] Harris Protocol Manual, Harris (C).
[7] B. Jepson. *Java Database Programming*, Wiley 1996.
[8] 김영경, "최고의 SCADA개발 도구인 WIZCON 소개" pp. 73-77, 제어계측 1996, 10.
[9] 손장균, "SCADA/RTU 시스템의 기술동향", pp. 7-13, 제어계측 1996, 12.
[10] Java Servlet API, http://www.javasoft.com/products/servlet/index.html
[11] Introduction to Oracle SQL/SQL*Plus, vol. 1-3, Oracle, Feb., 1994.
[12] Wizbroz 홈페이지, http://www.pcsoftintl.com/wiznet.htm
[13] Wizcon 홈페이지, http://www.pcsoftintl.com/
[14] http://wwwwseast2.usec.sun.com/smi/Press/sunflash/9802/sunflash.980217.1.html
[15] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environments," *Journal of the Association for Computing Machinery*, vol. 20, no. 1, pp. 46-61, 1973.

김 형 일

1972년 2월 27일생, 1994 경희대학교 물리학과학사, 1996 경희대학교 전자계산공학과 석사, 1996-현재 경희대학교 전자계산공학과 박사과정. 관심분야는 실시간 시스템, 멀티미디어 시스템, 자바.

전 태 웅

1958년 11월 20일생. 1981년 서울대학교 계산통계학과 학사, 1983년 2월 서울대학교 계산통계학과 석사. 1992년 5월 Illinois Institute of Technology. 전산과학 박사, 1983년 2월-1987년 6월 금성통신(현 LG전자) 연구소 주임연구원, 1992년 9월-1995년 2월 LG산전 연구소 책임연구원, 1995년 3월-현재 고려대학교 전산학과 부교수. 관심분야는 소프트웨어 공학(소프트웨어 테스팅, 소프트웨어 아키텍처. 설계 패턴, 객체지향 프레임웍크, 실시간 소프트웨어 공학 등)

이 승 룡

1955년 3월 1일생, 1978 고려대학교 재료공학과 학사, 1986년 12월 Illinois Institute of Technology 전산학과 석사, 1991년 12월 Illinois Institute of Technology 전산학 박사, 1992-1993 Governors State University 조교수. 1993-현재 경희대학교 전자계산공학과 부교수. 관심 분야는 실시간 시스템, 실시간 고장허용 시스템, 멀티미디어 시스템.

박 영 택

1955년 10월 5일생, 1978 서울대학교 전자공학과 학사, 1980년 한국과학기술원 전산학과 석사, 1992년 University of Illinois at Urbana Champaign 전산학 박사. 1992-현재 숭실대학교 컴퓨터학부 교수. 관심분야는 인공지능, 에이전트.