HTTP 기반의 자바를 이용한 원격 감시 및 제어 시스템

HTTP based remote monitoring and control system using JAVA

이 경 웅, 최 한 수*
(Kyoung-Woong Yi and Han-Soo Choi)

Abstract : In this paper, It is studied to control and to monitor the remote system state using HTTP(HyperText Transfer Protocol) object communication. The remote control system is controlled by using a web browser or a application program. This system is organized by three different part depending on functionality-server part, client part, controller part. The java technology is used to composite the server part and the client part and C language is used for a controller. The server part is waiting for the request of client part and then the request is reached, the server part saves client data to the database and send a command set to the client part. The administrator can control the remote system just using a web browser. Remote part is worked by timer that is activated per 1 second. It gets the measurement data of the controller part, and then send the request to the server part and get a command set in the command repository of server part using the client ID. After interpreting the command set, the client part transfers the command set to the controller part. Controller part can be activated by the client part. If send command is transmitted by the client part, it sends sensor monitoring data to the client part and command set is transmitted then setting up the value of the controlled system.

Keywords : HTTP, remote monitoring, remote control, client, server, java technology, web browser

I. 서론

각종 산업현장에서는 중앙 통제실에서 원거리에 있는 각종 전자 기기들의 현재 운영 상황을 모니터링하고 제어하기 위하여 각종 원격 감시 제어 시스템들을 운용하고 있으며, 또한 최근에는 외부에서 가정에 있는 전자 제품들의 상태를 파악하고 제어하고자 하는 요구의 증대에 따라 각종 원격 관리 시스템들에 관한 연구가 진행되어지고 있다[1-4].

기존의 원격제어 시스템들은 클라이언트/서버 환경하에서 서버를 직접 구성함으로써 시스템의 최적화를 위해 많은 시간과 노력이 필요하다. 또한 웹 환경에서의 개발 필요성이 대두 되어 기존의 개발 방식을 유지 하면서 웹 기반의 전환이 상대적으로 용이한 CGI 기반의 개발이 이루어졌다. 그러나 CGI의 경우 단일 쓰레드 사용으로 인해 다중 접속이 요구되어지는 시스템에서는 성능 저하가 발생 하며 플랫폼에 종속적이어서 플랫폼 변화에 능동적으로 대처 할 수 없다[6-11].

본 논문에서는 기존의 원격 관리 시스템에 비해 개발이 용이하고 거리의 증가에 따른 영향이 적으며 시스템 구성 시 시간과 비용을 절약할 수 있고 플랫폼 독립적이며 다중 쓰레드 사용으로 성능 향상을 기대할 수 있는 JAVA와 Servlet을 이용한 HTTP기반 원격 관리 시스템의 모델을 제안하고자 한다. 제안된 원격관리 시스템은 HTTP(HyperText Transfer Protocol)를 이용함으로써 기존의 웹 어플리케이션 서버에 서버 프로그램의 등록만으로 적용할 수 있게 되어 기존 시스템의 일부로써 사용 가능하게 되고, 또한 기존의 다른 시스템과의 연동도 용이하게 이루어 질 수 있다[10], [11]. 그리고 웹 어플리케이션 서버의 관리하에 서버프로그램이 등록되어지므로 클라이언트가 증가하여도 연결 관리나 부하 분산적 측면에서 능동적으로 대처할 수 있게 된다. 또한 클라이언트에서 서버로 전송된 데이터는 데이터베이스에 저장되어지므로 관리자 및 시스템 사용자는 현재 온도 및 온도 변화의 트랜드(trend)를 관찰할 수 있으며 웹 브라우저를 통하여 시간과 장소에 관계없이 원격지 시스템의 현재 상태를 확인할 수 있게 된다[5], [6], [12].

제안된 원격 감시 제어 시스템은 HTTP를 이용하여 통신을 하며, 클라이언트 부분과 서버 부분 그리고 컨트롤러 부분으로 구성된다. 클라이언트 부분이 타이머에 의해서 매초에 한번씩 컨트롤러에 데이터 전송을 요구하여 전송받은 컨트롤러의 메시지를 서버에 전송하고 다시 서버로부터 메시지를 전송 받는다.

본 논문에서는 HTTP 기반의 원격 감시 제어 시스템 구성을 위하여 서버와 클라이언트 사이의 데이터 전송을 위한 전송 규약을 정의하고 이를 Java technology를 이용하여 구현하였다. 그리고 제안된 시스템의 가용성을 확인하기 위해 서버와 근거리(1km 이내) 및 원거리(400km 이내)에 클라이언트를 위치시켜 거리에 따른 서버와 클라이언트의 응답을 측정하였다.

II. 시스템 구성

본 연구에서의 원격 감시 및 제어 시스템은 서버 부분과 클라이언트 부분 그리고 제어 대상체와 인터페이스를 위한 컨트롤러 부분으로 나눌 수가 있다. 서버 부분과 클라이언트 부분은 플랫폼 독립적이고 네트워크 관련 풍부한 API를 제공하는 Java로 작성되었으며 컨트롤러 부분은 C언어로 작

* 책임저자(Corresponding Author)
논문접수 : 2003. 12. 26., 채택확정 : 2004. 6. 25.
이경웅 : 조선대학교 대학원 제어계측공학과(yikw@yncc.co.kr)
최한수 : 조선대학교 정보제어계측공학과(hschoi@chosun.ac.kr)
※ 이 논문은 2001년도 조선대학교 학술연구비의 지원을 받아 연구되었음.

성되었다. 그리고 감시 및 제어 대상으로는 온도 측정을 위한 서미스터와 발열기를 사용하였다. 서버 부분은 Java Servlet의 사용을 위해 Java Servlet 컨테이너인 Tomcat을 사용하였다. 그리고 클라이언트는 Java 어플리케이션 단독으로 실행되도록 하였다. 데이터베이스는 마이크로소프트사의 MS SQL Server 7.0을 사용하였다. 서버 사이드와 클라이언트 사이드는 HTTP Object 통신을 이용하여 데이터를 교환하며 클라이언트 부분과 컨트롤러 부분은 RS-232를 통하여 데이터를 교환한다. 그리고 데이터베이스와의 통신은 JDBC를 사용하였다.

1. 서버 부분

서블릿은 Java 표준 API의 확장 인터페이스 집합이다. 웹 서비스를 위하여 서블릿 응용 프로그램은 Java Servlet API를 지원하는 다양한 웹 서버 및 서블릿 컨테이너 환경에서 실행되어야 한다. Tomcat은 Java Servlet과 JSP를 지원하는 서블릿 컨테이너로서 Apache Jakarta 프로젝트에서 제공하고 있다. Tomcat은 Java 기반으로 작성되어진 HTTP 웹서버 기능을 지원하며 웹서버 Plugin과 웹서버 내부의 JVM에서 수행되는 자바 컨테이너 구현이 결합된 형태이다. 웹서버 Plugin은 JVM을 웹서버의 어드레스 공간 내에 오픈하고, 자바 컨테이너가 해당 JVM 안에서 수행될 수 있도록 해준다. 그리고 다중 쓰레드의 사용으로 다중 사용자 접속시 향상된 성능을 보여 주며 플랫폼 독립적으로 한번 개발되어진 시스템에 대해서는 플랫폼에 관계없이 적용할 수 있다[11]. JDBC는 산업계의 다양한 관계형 Database에 접근하는 표준 API set을 제공한다. 이 JDBC API는 Java2 플랫폼 Standard Edition에 포함되어 있다. JDBC API는 java.sql 패키지에 제공되고 있어서 어떤 Java 코드에서도 Database에 접근할 수 있다. 서버 부분은 클라이언트와 통신을 위한 Servlet 부분과 웹상에서 데이터 표현을 위한 JSP 부분으로 이루어져 있다[10].

2. 클라이언트 부분

클라이언트 부분은 서버와 HTTP Object 통신에 의해 데이터를 교환하고 컨트롤러 부분과는 RS-232를 통해 데이터를 교환한다. 클라이언트 부분은 타이머에 의해 일정 시간마다 컨트롤러에 센서의 값을 요구하여 전송되어진 센서의 값을 다시 서버에 전송하며 서버의 Command Repository에서 자신의 클라이언트 ID를 키 값으로 자신에게 할당된 명령어를 가지고 와서 이를 다시 컨트롤러에 전송한다. Object 통신을 이용함으로써 자바의 객체 변수를 그대로 전송할 수 있게 된다[11].

3. 컨트롤러 부분

컨트롤러 부분은 89C196 마이크로 컨트롤러를 사용하였다. 그리고 서미스터의 온도를 측정하고 발열기를 제어하기 위하여 ADC와 DAC를 사용하였다.

III. 시스템 간 데이터 전송

본 연구 시스템은 HTTP를 사용하기 때문에 반 이중 방식의 통신이 이루어져야 한다. 서버 부분에서 클라이언트 부분으로 보내온 데이터를 처리하기 위해서 데이터 프레임을 정의하여야 한다. 반 이중 방식은 클라이언트에서 서버

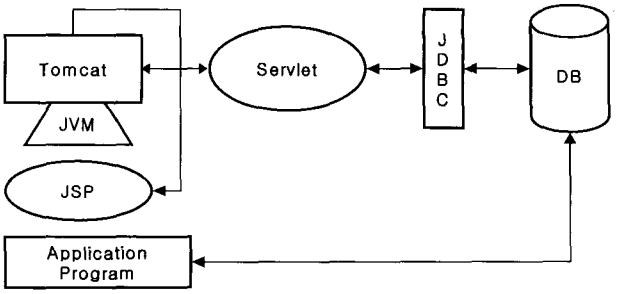


그림 1. 서버부분 구성도.

Fig. 1. Structure of server part.

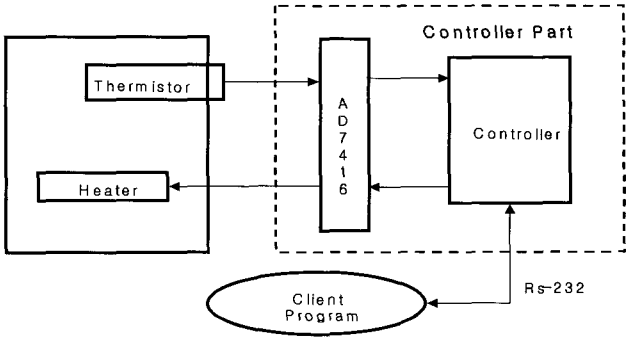


그림 2. 컨트롤러 부분 구성도.

Fig. 2. Structure of controller part.

에 데이터를 전송할 때와 서버에서 내려진 명령을 가져가는 부분이 서로 독립적으로 존재한다. 클라이언트는 정의되어진 시간마다 서버에 접속하여 자신의 현재 데이터를 서버에 전송하고 또한 서버에서 처리되어진 데이터 중 자신에게 할당되어진 값을 가져오게 된다.

1. 클라이언트에서 서버로 측정데이터 전송

클라이언트에서 서버로 측정데이터 전송은 JDK에서 정의되어진 Hashtable Class를 사용하였다. Hashtable Class는 Key, Value 쌍으로 정의되어지는 Class로서 직관적인 사용을 가능하게 한다. 표 1은 본 연구에서 사용되어진 데이터에 대한 정의로써 클라이언트가 서버에 측정된 데이터 값을 전송하기 위해 사용하는 Hashtable의 구조를 나타내고 있다. ID는 각 클라이언트에 클라이언트 모듈을 설치할 때 부여되며 Time은 클라이언트가 컨트롤러에 센서의 측정 데이터를 요구하여 센서의 측정값을 받은 시간을 나타낸다. 그리고 Sensor는 컨트롤러에서 측정되어진 데이터 값을 가지고 있는 Vector 클래스로서 순차적으로 측정 값을 변수에 저장함으로써 컨트롤러에서 사용한 센서의 수에 능동적으로 대처할 수 있다.

2. 서버에서 클라이언트로 명령 전송

서버에서 클라이언트로의 명령 전송은 HTTP의 특성상 서버가 클라이언트에 통신을 요구할 수가 없으므로 클라이언트가 서버에 접속하여 명령을 가져가야 한다. 따라서 본 연구에서는 서버에 명령 저장소를 두어 클라이언트에서 서버로 측정 데이터를 전송할 때 명령 저장소에서 자신에 대한 명령을 가져가게 하였다. 명령 저장소는 Hashtable Class로

표 1. 서버 클라이언트 통신 hashtable의 내용.
Table 1. Contents of server and client transmitted hash-table.

Key	Value	Type	comment
ID	Client ID	String	각각의 클라이언트에 부여된 ID
Time	데이터 측정시간	String	클라이언트에서 컨트롤러에게서 측정값을 받은 시간
Sensor	Sensor 측정값	Array	클라이언트에서 측정한 값

표 2. 명령 hashtable 내용.
Table 2. Contents of command hashtable.

Key	Value	Type	Comment
UP	온도변화값	Int	온도를 올리고자 할 때
DOWN	온도변화값	Int	온도를 내리고자 할 때
STOP	yes or no	boolean	컨트롤러를 정지시키고자 할 때
RESET	yes or no	boolean	컨트롤러를 재시작하고자 할 때

이루어져있으며 Client ID가 Key값으로 그리고 Value는 명령 Hashtable로 되어있다. 표 2는 본 연구에서 사용되어진 명령 Hashtable의 구성을 보여 주고 있다. UP은 클라이언트에 온도를 올리도록 할 때 사용한다. Default는 0으로 올리지 않으면 0으로 한다. DOWN은 클라이언트에 온도를 내리도록 하고자 할 때 사용한다. Default는 역시 0으로 올리지 않으면 0으로 한다. STOP은 컨트롤러를 정지시키고자 할 경우 사용한다. RESET은 클라이언트에게 컨트롤러를 재 시작 하도록 하는 명령이다.

3. 클라이언트와 컨트롤러간 데이터 전송

클라이언트와 컨트롤러간 데이터 전송은 텍스트 기반의 전송규약을 정하였다. 클라이언트에서 컨트롤러로 특정 문자가 전송되면 컨트롤러 측에서는 특정 문자를 해석하여 명령을 결정하고 작업 후 RS-232를 통해 클라이언트로 결과값을 전송한다. 컨트롤러는 전적으로 클라이언트의 명령에 의존한다. 그림 3은 클라이언트에서 컨트롤러의 Command Set을 보여주고 있다. Command Set의 시작은 ASCII 35이고 Command와 Value의 구분자는 ASCII 36이며 Command Set 종료문자는 ASCII 94이다.

표 3은 각각의 Command와 그에 따른 값을 설명하는 것으로써 UP은 온도의 상승을 위하여 DO는 온도를 내리고자 할 경우 ST는 컨트롤러를 정지하고자 할 때 RS는 컨트롤러를 재 시작 하고자 할 때를 나타낸다.

그림 4는 컨트롤러에서 클라이언트로의 Command Set을 보여주고 있다. Command Set의 시작은 ASCII 35이고 key와 Value의 구분자는 ASCII 36이며 Command Set 종료문자는

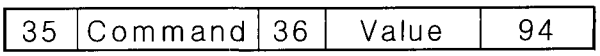


그림 3. 클라이언트에서 컨트롤러로의 command set.
Fig. 3. Command set from client to controller.

표 3. 컨트롤러의 command.
Table 3. Controller command.

Command	Value	Comment
UP	온도증가치	온도를 상승시킨다.
DO	온도감소치	온도를 감소시킨다.
ST	NO Value	컨트롤러를 정지시킨다.
RS	NO Value	컨트롤러를 재시작한다.

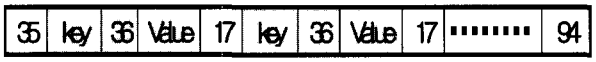


그림 4. 컨트롤러에서 클라이언트로의 command set.
Fig. 4. Command set from controller to client.

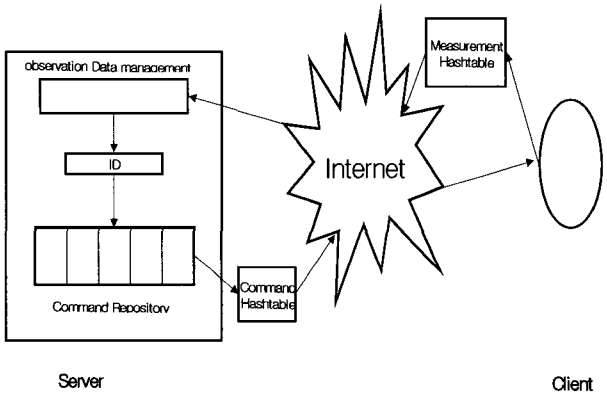


그림 5. 시스템 간 데이터 전송 개념도.
Fig. 5. Structure of data transmission between systems.

ASCII 94이다. key 값은 각각의 센서를 나타내며 Value는 센서에서 나타내는 값을 나타낸다. 그리고 ASCII 17의 데이터 쌍 간의 구분자이다. 이렇게 함으로써 센서의 갯수에 관계없이 값을 읽어 올 수가 있다.

컨트롤러에서 전송된 Command Set은 클라이언트에서 데이터 쌍으로 분리되어 Vector에 저장되며 이 값은 다시 서버 클라이언트 통신 Hashtable에 담기어 서버로 전송되어진다.

그림 5는 서버와 클라이언트사이의 데이터 전송 개념도를 나타내고 있다. 클라이언트가 매초마다 컨트롤러의 측정 데이터를 Hashtable의 형태로 서버에 전송하면 서버는 클라이언트의 ID 값을 근거로 클라이언트에 컨트롤데이터를 전송한다.

그림 6은 컨트롤러, 클라이언트, 서버 부분의 데이터 전송에 따른 흐름도를 나타내고 있다. 시스템의 시작은 클라이언트 부분에 의해 전적으로 좌우된다. 클라이언트 부분에

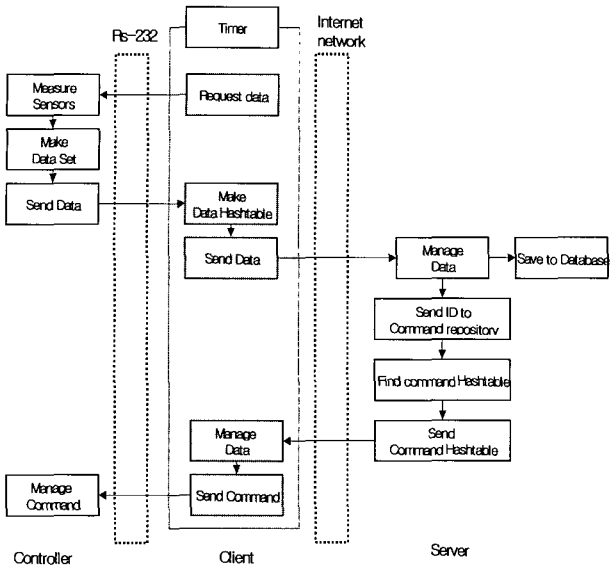


그림 6. 시스템 간 데이터 전송 흐름도.
Fig. 6. Flowchart of data transmission between systems.

서 매초마다 컨트롤러에 측정데이터를 요구하고 컨트롤러는 클라이언트의 요구에 따라 데이터를 전송한다. 그리고 클라이언트는 이 데이터를 전송규약에 맞게 Hashtable을 만들어 서버에 전송한다. 서버는 클라이언트로부터 데이터 전송이 있으면 그 데이터를 풀이하여 데이터베이스에 저장하고 클라이언트의 ID를 이용하여 현재 Command repository에 요청해온 클라이언트에 해당하는 데이터의 유무를 검색한 후 데이터가 있을 경우 시스템 통신 규약에 맞게 Hashtable을 구성하여 클라이언트에 전송한다. 클라이언트는 서버로부터 데이터의 전송이 있을 경우 데이터를 풀이하여 컨트롤러에 명령을 보낸다. 이 모든 일련의 작업은 전적으로 클라이언트의 타이머에 의해 작동된다. 따라서 서버와 컨트롤러는 클라이언트의 요청이 있을 경우에만 데이터의 전송을 받고 데이터를 전송한다.

IV. 시스템 설계 및 구현

본 연구에서는 밀폐된 공간에서의 온도를 실험 대상으로 하였다. 밀폐된 공간에 특정 온도를 유지하게 하기 위하여 컨트롤러에서 알고리즘에 의한 제어를 실행하기보다는 관리자가 프로그램을 통해 모니터링하고 또한 온도가 올라갔을 경우 명령을 통해 온도를 내리도록 하였다. 시스템은 서버 프로그램, 클라이언트 프로그램, 마이크로 프로세서 시스템으로 이루어져있다.

1. 서버 프로그램

서버 프로그램은 Java로 작성된 Servlet을 사용하였으며 내부적으로 Command repository로 관리자가 온도를 설정한 값을 저장한다. 클라이언트의 요구가 있을 경우에만 작동을 하는 수동적 서버의 형태로써 클라이언트의 요청이 있을 경우 클라이언트의 측정값을 데이터베이스에 저장하고 Command repository에서 현재 접속 클라이언트에 대한 Command를 찾아서 전송한다. 클라이언트의 Data set을 데이

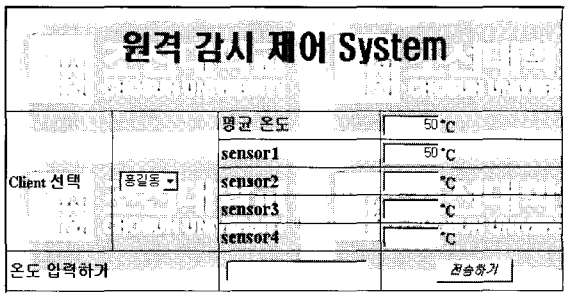


그림 7. 서버 프로그램 작동 화면.
Fig. 7. Operation screen of server program.

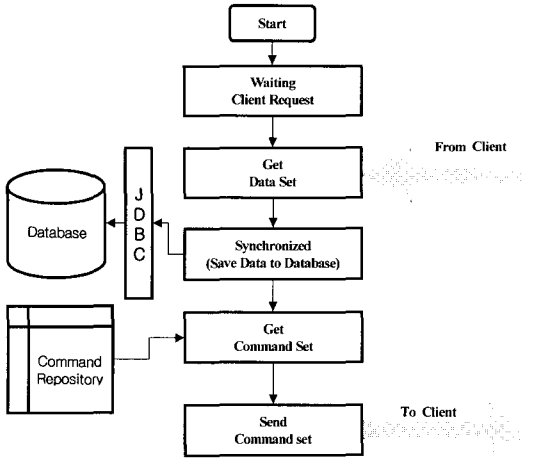


그림 8. 서버 프로그램 플로우차트.
Fig. 8. Flowchart of server program.

터베이스에 저장하기 위하여 Java Technology에서 제공하는 JDBC(Java DataBase Connectivity)를 사용하였다. JDBC는 Java에서 데이터베이스를 사용할 경우 데이터베이스 벤더들에 상관없이 Java를 사용할 수 있도록 하는 인터페이스이다. 그리고 Servlet은 Thread로 실행되기 때문에 데이터를 데이터베이스에 저장시킬 때 직렬화의 문제가 발생한다. 이를 해결하기 위해서 Java에서는 synchronized라는 명령어를 제공하며 synchronized는 한번에 하나의 thread만 접근할 수 있도록 한다. 본 실험에서는 데이터베이스에 클라이언트의 데이터를 저장하는데 이를 사용하였다. 또한 관리자의 요구에 의해 클라이언트에 보낼 명령을 저장하기 위해 Hashtable class를 이용하여 Command repository를 생성하였다. 이것을 통하여 클라이언트의 ID를 이용하여 각각의 클라이언트에 할당된 Command set을 검색 및 전송할 수 있게 된다. 그림 7은 컨트롤러에 전송되어질 Command set을 만들기 위한 제어 화면이다. 관리자는 클라이언트를 선택하여 클라이언트에서 보내온 데이터를 볼 수 있으며 또한 희망 온도를 전송함으로써 원격지의 온도를 조정할 수 있게 된다.

그림 8은 서버 부분의 순서도를 나타내고 있다. 서버는 항상 클라이언트의 요청을 위해 대기하고 있다. 클라이언트의 요청이 있게 되면 먼저 전송 Hashtable을 풀이하고 데이터베이스에 저장하게 된다. 그러나 동시에 여러 클라이언트가

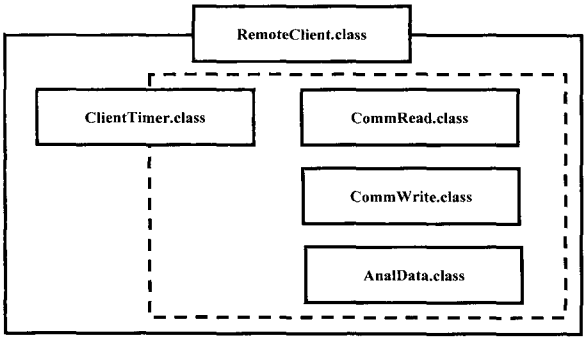


그림 9. 클라이언트 프로그램 클래스 구성도.
Fig. 9. Structure of client program class.

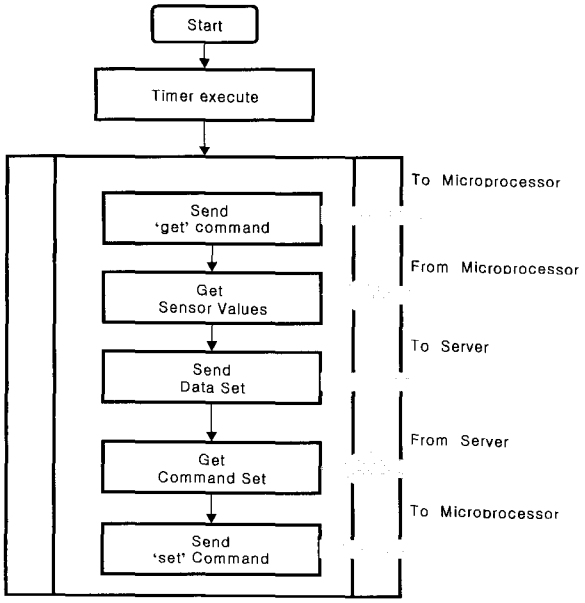


그림 10. 클라이언트 플로우차트.
Fig. 10. Client flowchart.

동일한 테이블에 대해 조작을 하게 될 경우 데이터 조작의 누락이 발생하게 된다. 따라서 데이터의 조작이 일어나는 부분에 대해서는 synchronized 명령을 사용하여 동기화 시켰다. 그런 다음 서버는 클라이언트의 ID를 이용하여 Command repository에서 해당 클라이언트에 전해질 Command를 가져온다. 그리고 다시 전송규약에 맞게 Hashtable형태로 구성하여 클라이언트에 전송하게 된다.

2. 클라이언트 프로그램

클라이언트 프로그램은 타이머에 의해 일정 시간마다 컨트롤러에 측정 값을 요구하고 그 값을 서버에 전송한다. 그리고 서버에서 할당된 명령을 해석하여 그것을 마이크로 컨트롤러에 전송한다. 서버가 클라이언트에 의해 작동하게 되므로 클라이언트의 타이머 주기에 의해 전체 시스템의 관측주기와 제어 주기가 결정된다고 할 수 있겠다. 클라이언트 프로그램은 크게 네 부분으로 나눌 수가 있는데 서버와 통신을 담당하는 부분과 마이크로프로세서와 RS-232통신을 하기 위한 부분, 통신 데이터들을 처리하기 위한 부분 그리고 이들을 통제하는 부분으로 구분할 수 있다. 본 실험에서

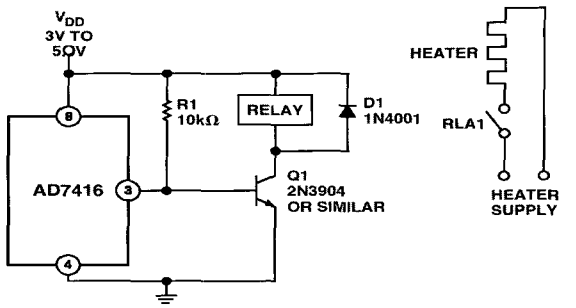


그림 11. AD7416에 Heater 작동 구성도.
Fig. 11. Structure of heater operating to AD7416.

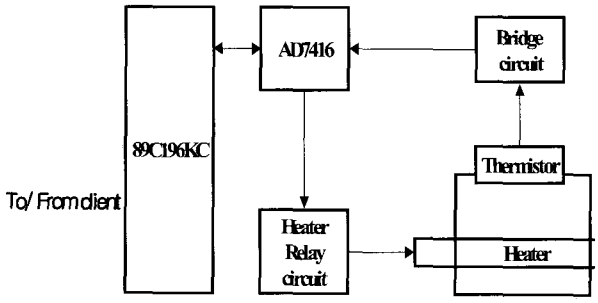


그림 12. 하드웨어 전체 구성도.
Fig. 12. Structure of hardware

는 각각을 위하여 별개의 class를 만들어 사용하였다.

그림 9는 클라이언트 프로그램의 클래스 구성도를 나타내고 있다. 클라이언트의 main 함수는 RemoteClient class에 있으며 ClientTimer에 의해 일정 시간마다 한번씩 작동하게 된다. CommRead.class, Comm Writeclass는 컨트롤러와 RS-232 통신을 위한 클래스이며 AnalData.class는 서버에서 전송되어진 데이터를 해석하여 컨트롤러에 전송하거나 컨트롤러의 데이터를 서버에 전송하기 위한 Hashtable을 만들 때 사용한다.

그림 10은 클라이언트의 플로우차트를 나타내고 있다. 클라이언트는 타이머에 의해 매초마다 한번씩 작동하며 먼저 컨트롤러의 데이터를 읽어 와서 다시 그것을 서버에 전송하고 서버에서 클라이언트의 ID를 통해 전송하는 데이터를 받는다.

3. 마이크로 프로세서 시스템

3.1 하드웨어 설계

마이크로프로세서는 89C196KC를 사용하였으며, AD Converter로는 Analog Device사의 AD7416을 사용하였다. AD7416은 온도 측정을 위한 것으로 설정 온도 보다 높은 온도일 경우 이를 감지하는 기능이 있다. 서미스터 값의 측정을 위해 서미스터 전단에 브리지 회로를 두었으며 전열체의 on/off 작동을 위하여 릴레이 작동 회로를 두었다. 그림 11은 AD7416을 이용한 시스템을 나타내고 있다. Q1에 의해 측정되어진 온도가 레지스터에 설정된 온도보다 클 경우 릴레이가 작동되어 heater의 전원이 차단된다. 따라서 본 시스템에서는 온도를 낮추기 위해 냉각을 시키는 방법이 아닌 자연 감소 방법을 적용한다.

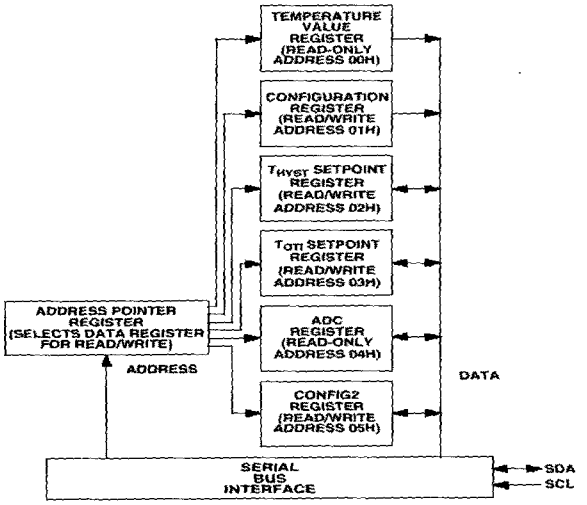


그림 13. AD7416 레지스터 구조.
Fig. 13. Structure of AD7416 registers.

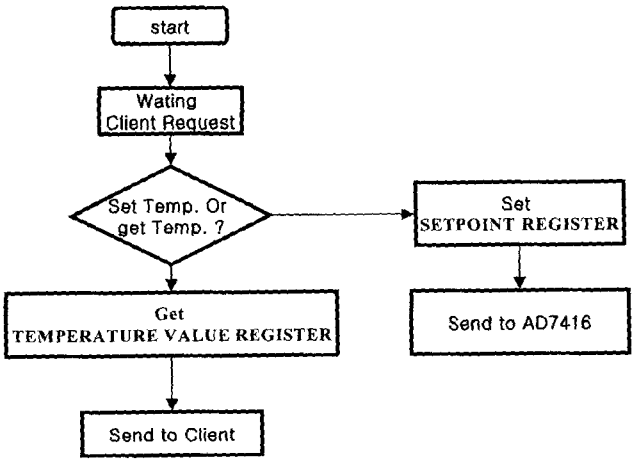


그림 14. 마이크로프로세서 플로우차트.
Fig. 14. Microprocessor flowchart.

그림 12는 하드웨어의 전체 구성을 나타내고 있다. 89C196 KC는 컨트롤러와 RS232를 통해 통신을 하고 AD7416의 레지스터의 값을 읽거나 설정하여 클라이언트의 요구에 응답한다.

3.2 소프트웨어 설계

AD7416을 사용하여 클라이언트 프로그램은 서버로부터 받은 Command set을 해석하여 마이크로프로세서로 보내고 마이크로프로세서는 이 값을 이용하여 AD7416의 THYST SETPOINT REGISTER 값을 설정하게 된다. 그림 13은 AD7416의 레지스터 구조를 나타내고 있다. AD7416은 THYST 값 보다 큰 값이 들어오면 OTI(OverTemperature Indicator) 출력이 활성화되고 THYST 값 보다 작은 값이 되면 활성화가 되지 않는다. 온도를 읽을 경우 AD7416의 TEMPERATURE VALUE REGISTER 값을 읽어 온다. 본 실험에서는 클라이언트에서 전송되어진 Command를 먼저 해석하여 온도를 설정하고자 하는 것인지 아니면 온도를 읽기 위한 것인지를 구분하여 각각의 로직을 운영한다.

그림 14는 컨트롤러의 프로그램 순서도를 나타내고 있다. 컨트롤러는 클라이언트의 요구가 있을 경우 그것이 현재 온도를 읽기 위한 것인지 아니면 온도를 설정하기 위한 것인지를 판단하여 온도 설정을 위한 것일 경우 AD7416의 SETPOINT RESISTOR의 값을 설정한다. 만약 온도를 읽기 위한 것일 경우에는 TEMPERATURE VALUE REGISTER 값을 읽어와 다시 클라이언트에 전송하게 된다. 컨트롤러는 전적으로 클라이언트의 요구에 의해서 작동하게 된다. 따라서 클라이언트의 요구 주기가 데이터 측정 주기가 된다.

V. 클라이언트와 서버간 응답시간 실험

서버 부분과 클라이언트 부분의 데이터 통신 응답시간을 측정하기 위하여 일정한 데이터를 클라이언트에서 전송하여 서버에서 데이터를 수신하는데 필요한 응답시간을 측정하였다. 전체 시스템에서 클라이언트 부분과 컨트롤러 부분사이의 응답시간은 수 마이크로 초[㎲]로 수 밀리 초[㎳]인 서버와 클라이언트의 응답시간에 비해 미비함으로 서버와 클라이언트간의 응답시간만을 측정하여 시스템의 응답 특성 및 응용 여부를 검사하였다. 서버는 서울 소공동에 두었으며 클라이언트의 실험은 서울 명일동과 광주광역시 서석동에서 수행하였다. 클라이언트와 서버의 응답시간은 각 기관 및 학교, 회사 등의 네트워크 환경과 네트워크 사용자 수에 따라 크게 변화한다. 실험은 사용자가 가장 많은 시간인 오전 11:00에 수행하였다.

1. 실험환경

가. 서버 : 서울 소공동
나. 클라이언트 : 서울 명일동, 광주광역시 서석동
다. 시간 : 오전 11시 00분
라. 주기 : 1초
마. 기간 : 10분
바. WebServer: Tomcat ver. 3.2
사. Program Language: Java 1.3, J2EE 1.2.1

2. 실험 모델

실험은 지역적으로 근거리와 원거리에서 실시하였으며 클라이언트의 네트워크는 일반 사업자에 의한 네트워크와 LAN 기반의 네트워크에서 각각 실시하였다. 그림 15는 실험을 위한 순서도로서 클라이언트에서 매초마다 서버에 특정 메시지를 전송하면 서버에서는 클라이언트의 ID를 이용하여 DataHashtable에서 데이터를 검색하여 클라이언트에 다시 전송하게 된다. 클라이언트에서는 서버에 데이터를 전송하기 전의 시간을 저장한 후 서버의 응답을 받은 시간에서 뺀 값을 파일에 저장한다.

그림 16은 실험 시스템의 개념도를 나타내고 있다. 근거리와 원거리를 위하여 서버는 서울의 소공동에 위치하고 근거리 클라이언트는 같은 LAN을 사용하는 서울 명일동에 위치하며, 원거리 클라이언트는 400km 정도의 거리에 있는 광주광역시 서석동에 있다.

3. 실험 결과

표 4에서 볼 수 있듯이 동일 LAN 상에 있는 클라이언트의 응답시간은 평균 56.5 ms이었으며 클라이언트(서울)에서 서버로의 Ping test에 의한 평균 RTT 값은 10ms 이하여서

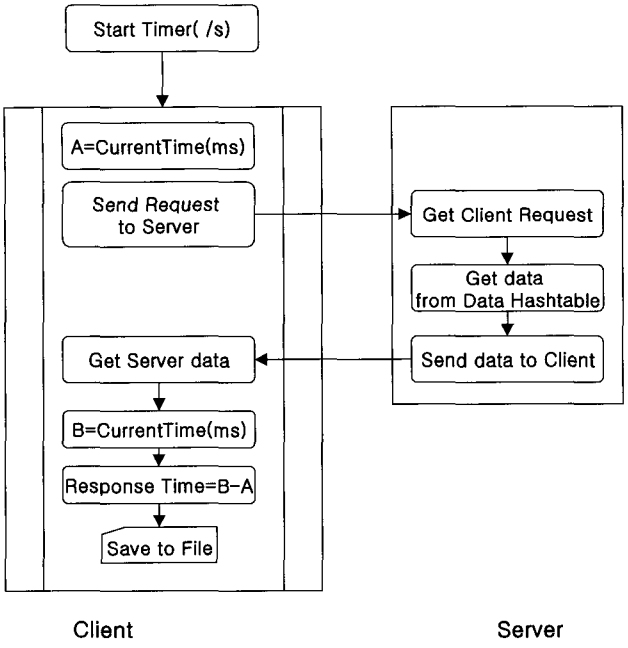


그림 15. 실험 시스템 순서도.

Fig. 15. Experimental system flowchart.

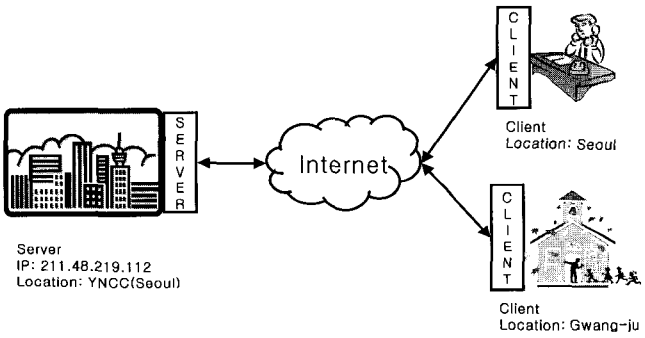


그림 16. 실험 시스템 개념도.

Fig. 16. Structure of experimental system.

표 4. 응답시간의 평균값, 최대값, 최소값.

Table 4. Mean, maximum, minimum of response time.

구분	서울	광주
평균응답시간(ms)	56.5	289.7
최대값(ms)	101	790
최소값(ms)	40	110

Ping test에는 나타나지 않았다. 지리적으로 400km 가량 떨어져 있는 광주광역시 서석동까지의 응답시간은 평균 289.7ms이었고 그림 19와 같이 광주에서 서버로 Ping test를 실시한 결과 평균 RTT 값은 119.33ms를 나타내었다. Ping test와 서버 클라이언트 응답시간 간의 차이는 서버 어플리케이션에서 클라이언트의 응답을 받아서 처리하는 시간에 의한 차이이다. 또한 표 4에서 볼 수 있듯이 각각의 응답 최대시간은 101ms, 790ms이었다. 이 응답시간은 클라이언트

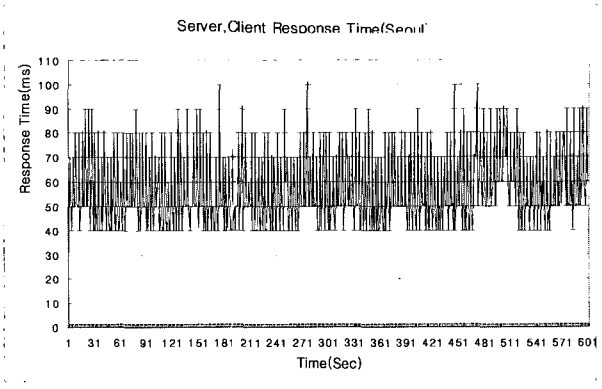


그림 17. 서버와 클라이언트(서울) 응답시간.

Fig. 17. Response time between server and client(seoul).

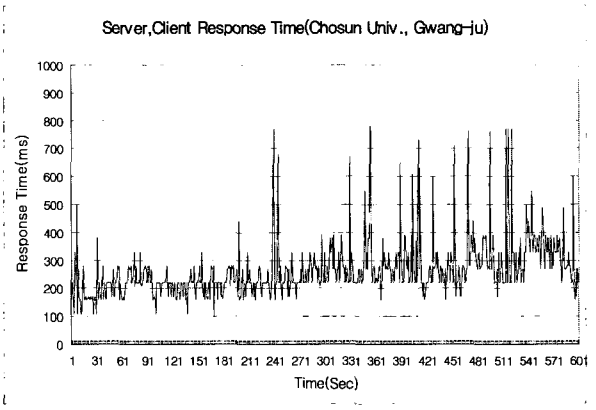


그림 18. 서버와 클라이언트 응답시간(광주).

Fig. 18. Response time between server and client(gwangju).

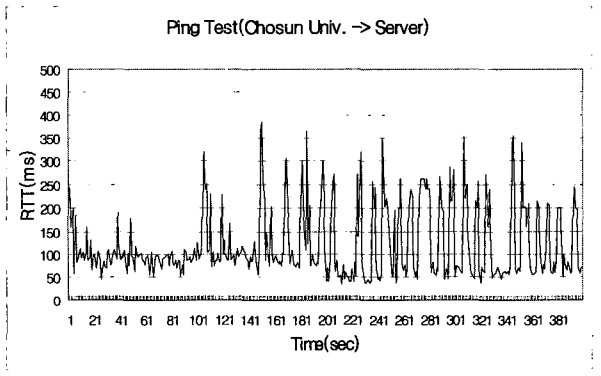


그림 19. Ping test(광주 → 서버).

Fig. 19. Ping test (gwangju → server).

에서 서버에 데이터를 전송하기 전 클라이언트 시간을 서버에서 클라이언트의 응답을 받고 Hashtable에서 해당 데이터를 가져오고 다시 클라이언트에 데이터를 전송한 후 이를 클라이언트에서 받은 시간에 뺀 값이다. 따라서 평균 응답시간을 보면 충분히 초당 응답이 가능하다고 할 수 있으나 최대 응답시간의 측면에서는 서버에서 데이터 조작 시간이 길어 질 경우 데이터가 무시 될 수도 있음을 보여 주고 있다. 그러나 서버 내에서의 작업시간이 μs 단위로 이루

어진다고 보았을 때 충분히 초당 응답 시스템에 사용할 수 있음을 보여주고 있다.

VI. 결론

본 논문에서는 기존의 원격 감시 및 제어 시스템에 비해 서버와 클라이언트간의 거리에 영향이 적으며, 시스템 구축 및 관리가 용이하고 또한 효율적인 시스템 운영이 가능하며, 웹 브라우저를 통하여 언제 어디에서나 시스템을 관리할 수 있는 HTTP 기반의 자바를 이용한 원격 감시 및 제어 시스템을 제안하였다.

제안된 시스템의 타당성을 알아보기 위하여 원격관리에서 가장 중요시되는 응답시간에 대한 실험을 통해 가능한 시스템 응답 시간을 알아보았다. 이를 통하여 근거리 뿐 아니라 원거리에서도 추가적인 설비 없이도 충분히 사용 가능한 시스템 모델임을 보였다. 이는 저속의 응답을 요하는 공장의 각 기계에 대한 모니터링 및 설정 값 제어 그리고 가정의 각 기기에 대한 감시 제어 등 저속의 제어와 모니터링이 요구되는 시스템에 대해서 효과적으로 이용될 수 있음을 보여주고 있다. 또한 기존의 웹 서비스에 추가적으로 적용할 수 있어 기존의 웹 서비스의 질적 향상과 기능 강화를 이룰 수 있었고 적은 비용으로 시스템 구축이 가능함을 알 수 있었다. 클라이언트가 서버와 항상 접속을 유지하지 않고 통신을 요구할 때만 접속을 하므로 서버 부분의 자원 관리 측면에서도 효율적임을 알 수 있었다. 이를 통하여 관리자 및 서비스 이용자는 시간과 공간적 제약 없이 단지 웹 브라우저만을 통하여 시스템을 관리 및 모니터링 할 수 있다는 점에서 그 가치가 있으며 응용은 더욱 증대될 것이다.

네트워크상의 데이터전송 시간은 일반적으로 Ping test 및 서버 클라이언트 응답시간 실험을 통해 볼 수 있듯이 직접적인 연결이 아니면서 많은 네트워크 경로를 거쳐야 하는 네트워크의 특성상 빠른 실시간 제어가 요구되는 시스템의 경우 아직은 직접적인 적용이 힘들지만 서버의 환경 개선과 고속 네트워크 사용을 통하여 실제 응답시간을 향상시킬 수 있을 것이다. 향후 SOAP(Simple Object Access Protocol)와 같은 새로운 개념의 원격 객체 프로토콜의 사용으로 단순 HTTP를 사용하는 것보다 효율적이고 안정적인 서비스를 제공할 수 있는 시스템에 대한 연구가 수행되어져야 할 것이다.

참고문헌

[1] W. K. Edwards, "Coordination infrastructure in collaborative systems", *A Dissertation Degree of Doctor of Computer Science,* Georgia Institute of Technology, Nov. 22, 1995.

[2] C. M. W. enbrick, E. C. Rosxn, D. D. E. Long, "Real-time system for managing environmental data", *Proceeding of Conference on Software Engineering and Knowledge Engineering,* June 1996.

[3] T. R. Haining, D. D. E. Long, P. E. Mantey, C. M. Witt enbrink, "The realtime environmental information network and analysis system(REINAS)", *Proceeding of COMPCON,* Mar. 1995.

[4] A., Banerjea D., Ferrari B., Mh. M., Moran D. Verma and H. Zhang "The tenet realtime protocol smite design, implement action and experiences." IEEE ACM Trans .Net working 4(1)1- 10, 1996.

[5] R., Braden D. Clark and S. Shenker "Integrservices in the internet architecture : An overview. "Technical report RFC- 1633, Network Working Group 1994.

[6] R. Anderson and M. Spong "Bilateral control of teleoperators with time delay." *IEEE Trans. on automatic control 34(5)* : 494- 501, 1989.

[7] J. Bolot "End-to-end packet delay and loss behavior in the internet." *In SIGCOMM* ' 93. New York : ACM, pp.289- 298, 1993(Ithaca,NY).

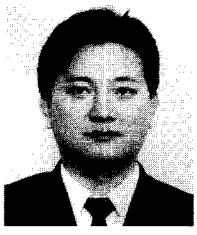
[8] P. Mohapatra, H. Chen, WebGraph : A framework for managing and improving performance of dynamic web content, *IEEE Journal On Selected Areas in Communications,* 20(7), Sep. 2002.

[9] J. Steinberg and J. Pasquale, A web middleware architecture for dynamic customization of content for wireless clients, *Proc. 11th Int'l WWW Conf.,* Honolulu, HI, May 2002.

[10] http://java.sun.com

[11] http://www.apache.org

[12] Lorenz P., H. Dorwarth K.-C. Ritter T. J. Schriber, Towards a web based simulation environment, Winter Simulation Conference(WSC'97), Atlanta GA, 1997.



이 경 웅

1973년 10월 5일생. 1998년 조선대학교 제어계측공학과 졸업. 2003년 동대학원 석사졸업. 현재 동대학원 박사과정. 관심분야는 자동제어 및 네트워크 시스템.

최 한 수

제어 · 자동화 · 시스템공학 논문지 제 8 권 제 12 호 참조.