HTTP에 의한 원격 모니터링

이 경 웅, *최 한 수
조선대학교 제어계측공학과, *조선대학교 정보제어계측공학부

Remote monitoring using HTTP

Yi, Kyoung-Woong, *Choi, Han-Soo
Dept. of Control and Instrumentation Eng. Chosun Univ.
*School of Information, Control and Instrumentation Eng. Chosun Univ.

ABSTRACT- In this paper, It is studied to monitor the remote system status using HTTP(HyperText Transfer Protocol) object communication. 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, the client part and C language is used for a controller.

1.서 론

본 연구에서는 HTTP를 통한 원격 감시 시스템 구성을 위하여 서버와 클라이언트 사이의 데이터 전송을 위한 전송 규약을 정의하고 이를 JAVA technology를 이용하여 구현하였다. 그리고 제안된 시스템의 가용성에 대해 알아보기 위해 서버와 근거리(1Km 내) 및 원거리(400Km 내)에 클라이언트를 위치 시켜 거리에 따른 서버와 클라이언트의 응답을 측정하였다.

2. 시스템 구성

본 연구에서의 원격감시 제어 시스템은 크게 서버 부분과 클라이언트 부분 그리고 제어 대상체와 인터페이스를 위한 컨트롤러 부분으로 나눌 수가 있다. 서버 부분과 클라이언트 부분은 플랫폼 독립적이며 네트워크 관련 풍부한 API를 제공하는 JAVA로 작성되었으며 컨트롤러 부분은 C언어로 작성되었다. 그리고 제어 감시 대상체로는 온도 측정을 위한 서미스터와 발열기를 사용하였다.

2.1. 서버 부분

서버 부분은 클라이언트와 통신을 위한 Servlet 부분과 웹상에서 데이터 표현을 위한 JSP 부분으로 이루어져 있으며 데이터베이스와 통신을 통한 데이터베이스 관리를 위하여 JAVA technology에서 제공하는 데이터베이스 관련 API인 JDBC(Java DataBase Connectivity)를 사용하였다.

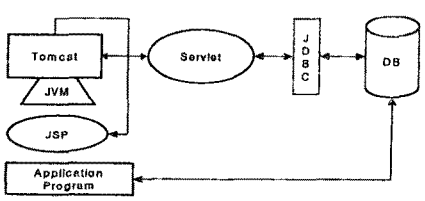

그림 2-1 서버부분 구성도

2.2 클라이언트 부분

클라이언트 부분은 서버와 HTTP object 통신을 통해서 데이터를 주고받고 컨트롤러 부분과는 RS-232를 통해서 데이터를 주고받는다. 클라이언트 부분은 타이머에 의해 매시간 마다 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. Object를 통신을 이용함으로써 자바의 객체 변수를 그대로 전송할 수가 있게 된다.

2.3. 컨트롤러 부분

컨트롤러 부분은 89c196 마이크로 컨트롤러를 사용하였다. 그리고 서미스터의 온도를 측정하고 발열기의 모니터링 및 제어를 위하여 ADC와 DAC를 사용하였다.

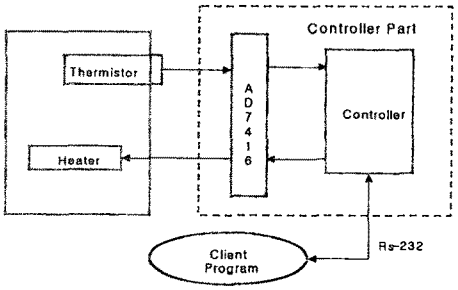

그림 2-2 컨트롤러 부분 구성도

3. 시스템 설계 및 구현

3.1. 시스템 구성

본 연구에서는 밀폐된 공간에서의 온도를 실험 대상으로 하였다. 밀폐된 공간에 특정 온도를 유지하게 하기 위하여 컨트롤러에서 알고리즘에 의한 자동제어를 실행하기보다는 관리자가 프로그램을 통해 모니터링하고 또한 온도가 올라갔을 경우 명령을 통해 온도를 내리도록 하였다. 시스템은 크게 마이크로프로세서 시스템, 클라이언트 프로그램, 서버 프로그램으로 이루어져있다.

3.2. 서버 프로그램

서버 프로그램은 JAVA로 작성된 servlet을 사용하였

으며 내부적으로 command repository를 이용하여 관리자가 온도를 설정한 값을 저장한다. 클라이언트의 요구가 있을 경우에만 작동을 하는 수동적 서버의 형태로서 클라이언트의 요청이 있을 경우 클라이언트의 측정값을 데이터베이스에 저장하고 command repository에서 현재 접속 클라이언트에 대한 command를 찾아서 전송한다. 그리고 servlet은 thread로 실행되기 때문에 데이터를 데이터베이스에 저장시킬 때 직렬화의 문제가 발생한다. 이를 해결하기 위해서 JAVA에서는 synchronized라는 명령어를 제공하는 synchronized는 한번에 하나의 thread만 접근할 수 있도록 한다. 그림 3-1은 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 동일한 테이블에 대해 조작하게될 경우 데이터의 조작의 누락이 발생하게된다. 따라서 데이터의 조작이 일어나는 부분에 대해서는 synchronized명령을 사용하여 동기화 시켰다. 그런 다음 서버는 클라이언트의 ID를 이용하여 command repository에서 해당 클라이언트에게 전해질 command를 가져온다. 그리고 다시 전송규약에 맞게 hashtable형태로 구성하여 클라이언트에 전송하게 된다.

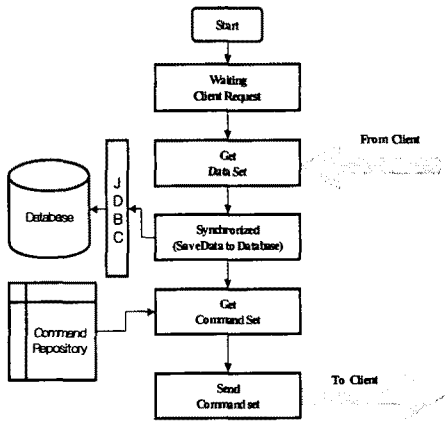

그림 3-1 서버 프로그램 플로우차트

3.3. 클라이언트 프로그램

클라이언트 프로그램은 타이머에 의해 일정 시간마다 컨트롤러에 측정값을 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 관측 주기와 제어 주기가 결정된다고 할 수 있겠다. 클라이언트 프로그램은 크게 4가지 부분으로 나눌 수가 있는데 서버와 통신을 담당하는 부분과 마이크로프로세서와 RS-232 통신을 하기 위한 부분, 통신 데이터들을 처리하기 위한 부분 그리고 이들을 통제하는 부분 등으로 구분 할 수가 있다. 그림 3-2는 클라이언트의 플로우차트를 나타내고 있다. 클라이언트는 타이머에 의해 매1초마다 한번씩 작동하며 먼저 컨트롤러의 데이터를 읽어 와서 다시 그것을 서버에 전송하고 서버에서 클라이언트의 ID를 통해 전송하는 데이터를 받는다.

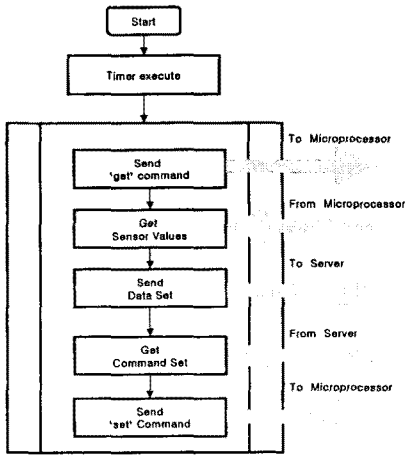

그림 3-2 클라이언트 플로우차트

3.4. 마이크로프로세서 시스템

3.4.1 하드웨어 설계

마이크로 프로세서는 89C196KC를 사용하였으며, AD converter로는 Analog Device사의 AD7416을 사용하였다. AD7416은 온도 측정을 위한 것으로 설정 온도 보다 높은 온도일 경우 이를 감지하는 기능이 있다. 써미스터의 값을 측정하기 위하여 서미스터 전단에 브리지 회로를 두었으며 전열체의 on/off 작동을 위하여 릴레이 작동 회로를 두었다. 그림 3-3은 컨트롤러의 전체 구성을 나타내고 있다. 98C196KC는 컨트롤러와 RS232를 통해 통신을 하며 AD7416의 레지스터의 값을 읽거나 설정하여 클라이언트의 요구에 응답한다.

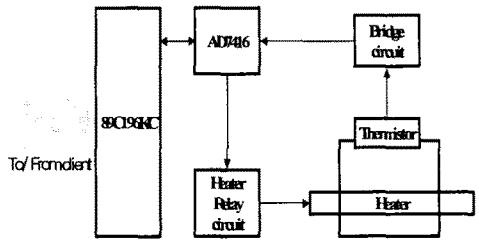

그림 3-3 하드웨어 전체 구성도

3.4.2. 소프트웨어 설계

AD7416을 사용하여 클라이언트 프로그램은 서버에서 받은 command set을 해석하여 마이크로 프로세서에 보내고 마이크로 프로세서는 이 값을 이용하여 AD7416의 T_{HYST} setpoint register 값을 설정하게 되다. 그림 3-4는 컨트롤러의 프로그램 순서도를 나타낸다. 컨트롤러는 클라이언트의 요구가 있을 경우 그것이 현재 온도를 읽기 위한 것인지 아니면 온도를 설정하기 위한 것인지를 판단하여 온도 설정을 위한 것일 경우 AD7416의 setpoint register의 값을 설정한다. 만약 온도를 읽기 위한 것일 경우에는 temperature value register값을 읽어와 다시 클라이언트에 전송하게 된다. 컨트롤러는 전적으로 클라이언트의 요구에 의해서 작동하게 된다. 따라서 클라이언트의 요구 주기가 데이터 측정 주기가 된다.

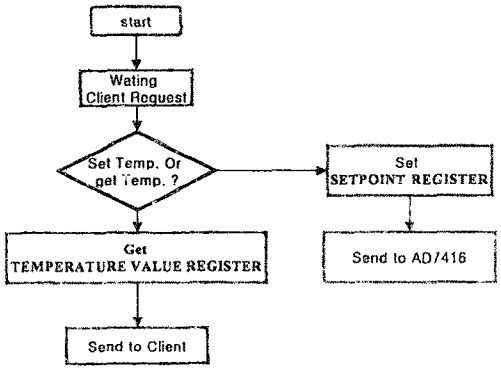

그림 3-4 마이크로프로세서 플로우차트

4. 클라이언트와 서버간 응답시간 실험

서버부분과 클라이언트 부분의 데이터 통신의 응답시간을 측정하기 위하여 일정한 데이터를 클라이언트에서 전송하여 서버에서 데이터를 다시 받는데 필요한 응답시간을 측정하였다. 전체 시스템에서 클라이언트 부분과 컨트롤러 부분사이의 응답시간은 수 μs로, 수 ms인 서버와 클라이언트의 응답시간에 비해 미비함으로 서버와 클라이언트간의 응답시간만을 측정하여 시스템의 응답 특성 및 응용 여부를 검사하였다. 서버는 서울 소공동에 두었으며 클라이언트의 실행은 서울 명일동과 광주에서 하였다. 클라이언트와 서버의 응답시간은 각 기관 및 학교, 회사 등의 네트워크 환경, 현재 네트워크 사용자에 따라 크게 변화한다. 따라서 사용자가 가장 많은 시간인 오전 11:00에 실험을 하였다.

4.1. 실험환경

가. 서버: 서울 소공동 YNCC본사
나. 클라이언트: YNCC 본사, 광주광역시 조선대학교
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, C(8051)

4.2. 실험 결과

표 4-1에서 볼 수 있듯이 동일 LAN상에 있는 클라이언트의 응답시간은 평균 56.6 ms 였으며 클라이언트에서 서버로의 ping test에 의한 평균 RTT값은 10ms 이하이므로 ping test에서 나타나지 않았다. 지리적으로 400Km가량 떨어져 있는 광주까지의 응답시간은 평균 289.7ms이었고 광주에서 서버로 ping test를 실시한 결과 평균 RTT값은 119.33ms를 나타내었다. ping test와 서버 클라이언트 응답시간 간의 차이는 서버 어플리케이션에서 클라이언트의 응답을 받아서 처리하는 시간에 의한 차이 이다. 또한 표 4-1에서 볼 수 있듯이 각각의 응답 최대 시간은 101 ms, 790 ms이었다. 이 응답 시간은 클라이언트에서 서버에 데이터를 전송하기 전 클라이언트 시간을 서버에서 클라이언트의 응답을 받고 hashtable에서 해당데이터를 가져오고 다시 클라이언트에 데이터를 전송한 후 이를 클라이언트에서 받은 시간에서 뺀 값이다. 따라서 평균 응답시간을 볼 경우 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 이루어진다고 보았을 때 충분히 초당 응답 시스템에 사용할 수 있음을 보여주고 있다.

표 4-1 응답시간의 평균값, 최대값, 최소값

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

5. 결 론

본 연구에서는 기존의 원격 감시 시스템에 비하여 서버와 클라이언트간의 거리에 영향이 적으며, 시스템 구축 및 관리가 용이하고 웹 브라우저를 통하여 언제 어디에서나 시스템을 감시할 수 있는 HTTP를 이용한 원격 감시 시스템을 제안하였다.

제안된 시스템의 타당성을 알아보기 위하여 원격관리에서 가장 중요시되는 응답시간에 대하여 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 모니터링과 가정의 각 기계에 대한 감시 등 저속의 모니터링이 요구되는 시스템에 대해서 효과적으로 이용될 수 있음을 보여 주고 있다. 또한 기존의 웹 서비스에 추가적으로 적용할 수 있어 기존의 웹 서비스의 질적 향상 및 기능적 강화를 이룰 수 있었고 적은 비용으로 시스템 구축이 가능함을 알 수 있었다. 클라이언트가 서버와 항상 접속을 유지하지 않고 통신을 요구할 때만 접속을 하고 접속을 끊으므로 서버 부분의 자원 관리 측면에서도 효율적임을 알 수 있었다.

참 고 문 헌

[1] Banerjea, a., ferrari, d., Mh, B., moran, M., verma, d., and Zhang, H. 1996. " the tenet real_time protocol smite design, implement action, and experiences." IEEE ACM Trans . Net working 4(1)1- 10

[2] Braden, R., Clark, D., and Shenker , S. 1994. " Integrated services in the Internet architecture: An overview. " Technical report RFC- 1633, Network Working Group.

[3] httpd://vive.cs.berkeley.edu/capek/

[4] W. R. Ferrell, " Remote manipulation with transmission delay", IEEE Trans Human Factors in Electronics HFE- 6, no.1 1965

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

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

[7] http://stat.nic.or.kr/public_html/iuser.html