최 종
연구보고서

저온저장고의 원격 감시 및 제어 시스템 개발

The development of remote monitoring and control system for low temperature storage house

연구기관
조 선 대 학 교

농 림 부

제 출 문

농림부 장관 귀하

본 보고서를 “저온저장고의 원격 감시 및 제어 시스템 개발” 과제의 최종보고서로 제출합니다.

2004 년 10 월 14 일

주관연구기관명 : 조선대학교
총괄연구책임자 : 최 한 수
위탁연구기관명 : 초당대학교
위탁연구책임자 : 정 헌
연 구 기 관 명 : 원예연구소
연 구 원 : 김 명 수

요 약 문

Ⅰ. 제 목

저온저장고의 원격 감시 및 제어 시스템 개발

Ⅱ. 연구개발의 목적 및 필요성

저온저장고는 농산물의 신선도를 유지하는 수단이다. 또한 수확 후 수 개월에 걸쳐 출하시기를 조절할 수 있어 농가 수입에 직결되는 농업 시설이다. 저온저장고에서 관리하는 환경은 주로 온도와 습도이며 품목에 따라 이들의 환경을 제어함으로써 신선도를 유지할 수 있도록 한다.

농가에서 저온저장고의 필요성을 인식하고 농가 단위로 저온저장고를 보유해 가는 추세이다. 그런데 저장고를 관리하는 데에 어려움이 있다. 저장고의 환경 상태를 확인하는 표시기가 저장고에 부착되어 있어서 직접 저장고까지 가야만 환경상태를 확인할 수 있다는 점과, 저장고 관리자가 저장고로부터 멀리 떨어진 곳에 있는 동안이나 취침 등과 같은 감시가 불가능한 경우에 발생할 수 있는 돌발 상황에 대처할 수 없다는 점이다. 저온저장고의 환경 상태를 감시하고 최적의 환경 상태를 유지할 수 있도록 온·습도를 원격 제어할 수 있는 시스템을 개발하여 농가에 설치하면, 저장고까지 가지 않고도 농가의 거실이나 안방에서 저장고의 환경을 감시하고 제어할 수 있다. 또 원거리에서 감시하고 제어할 수 있는 시스템을 구축하면 환경 상태의 감시 및 제어를 전문가에게 위탁할 수 있다.

저장고의 환경상태가 비정상 상태이거나 저장고의 장치들이 고장 시에 저장 농산물에 치명적 손상을 가져 올 수 있기 때문에 저온저장고에 대한 전문지식이 없는 농가에서 저장고를 관리하는 것 보다는 전문가가 인터넷을 이용하여 각 농가에 설치된 저온저장고의 환경상태를 감시하고 최적의 환경으로 제어할 수 있는 시스템을 구축하여 저장고 관리의 효율성을 높일 수 있는 기술을 개발함으로써 저온저장 분야의 기술을 향상시킬 수 있다.

본 연구에서는 인터넷을 이용하여 원거리에서 저온저장고 내의 환경상태를 감시하고 또 정상 상태가 아닐 경우 원거리에서 제어할 수 있는 저온저장고의 원격 감시 및 제어 시스템을 개발하였다.

연구개발 범위는 농가의 거실이나 안방에서 저온저장고의 온·습도를 감시하고 제어하는 시스템과, 전문가가 저장고를 보유한 다수의 농가들로부터 저장고의 관리를 위탁받아 원거리에서 온도와 습도를 중앙 집중 관리하는 온·습도 중앙 집중 감시 및 제어 시스템을 개발하는 것이다.

본 연구결과를 활용하면 농가에서 관리하는 경우는 방안이나 거실에서 저장고를 관리하기 때문에 저장고까지 직접 접근하지 않아도 되고, 중앙 집중관리의 경우는 전문가가 저온저장고를 관리하기 때문에 저장고내의 환경상태를 최적으로 관리할 수 있으며, 저온저장고를 위탁 관리해 주기 때문에 농가에서는 안심하고 생업에 종사할 수 있다.

Ⅲ. 연구개발 내용 및 범위

본 연구의 대상은 나주 지역에 있는 배 저장 시설을 중심으로 한다. 농가의 거실이나 안방에서 저장고의 환경을 감시하고 제어할 수 있는 시스템과, 여러 농가들에 대한 저온저장고의 환경상태를 전문업체가 집단으로 중앙 관리할 수 있는 시스템을 구현하기 위하여 인터넷상에서 저온저장고의 환경변화를 감시하고 최적의 상태를 유지하도록 원격 제어하는 시스템을 개발하는 것이 본 연구개발의 목적이며 목적을 달성하기 위한 세부 연구개발의 내용과 범위는 다음과 같다.

1. 자료조사

저온저장고의 온도와 습도의 측정을 위하여 온도·습도 측정기에 대한 정보들을 조사하고 원격제어의 방법에 대하여 조사하며 인터넷 보급률에 관하여 조사한다.

2. 제어시스템 구성

저온저장고의 온·습도를 제어하기 위한 저온저장고용 컨트롤러를 개발하고

제어 알고리즘 구현한다.

3. 측정시스템 구성

저온저장고의 실시간 온·습도 측정을 위한 측정시스템을 구성하고 저온저장고의 컨트롤러와 측정시스템간 인터페이스를 구축한다. 또한 정밀 측정을 위하여 추가적인 측정 센서를 설치하는 정밀 온·습도 측정을 위한 시스템을 설계하고 센서의 정밀도 향상을 위한 처리 알고리즘 구현한다.

4. 제어·측정시스템과 PC간 통신 프로토콜

제어·측정시스템과 PC간의 통신을 위하여 통신 프로토콜을 정의한다. 통신을 위한 하드웨어를 설계하고 통신프로그램을 작성한다.

5. PC기반 제어 프로그램 개발

농가의 PC에서도 직접 저온저장고의 온·습도를 측정 및 제어할 수 있는 PC 기반의 프로그램을 설계 및 구현하고 저온저장고의 관리 프로그램을 작성한다.

6. 저온저장고 온·습도 제어실험

온도와 습도의 상관관계를 조사하고, 시스템 오작동에 대한 안정성 확보 방안을 수립한다.

7. 데이터베이스 구축

클라이언트에서 서버로 전송된 데이터를 저장하기 위하여 데이터베이스를 구축한다.

8. Client와 Server간 프로토콜

Server와 Client간의 통신을 위하여 Server와 Client간 통신 프로토콜 정의하고 이를 구현한다.

9. Client용 통신 프로그램 개발

제어·측정시스템에서 측정된 온·습도 데이터의 수집을 위한 Client 정보 수

집 프로그램을 개발하고 이를 Server에 전송하기 위한 네트워크 접속 프로그램을 개발한다. 이를 통하여 제어·측정시스템의 정보를 Server로 전송하는 제어 및 감시 시스템 상태정보 전송 프로그램을 개발한다.

10. Server용 프로그램 개발

농가 PC에서 접속이 있을 경우 먼저 타당하고 인증된 PC인가를 결정하기 위한 Client의 인증 처리 프로그램을 개발하고 Client와의 통신 접속을 위한 네트워크 접속 프로그램을 개발한다. 또 클라이언트에서 전송된 정보를 관리하기 위한 클라이언트 전송 정보 관리 프로그램을 개발하며 클라이언트에서 전송한 데이터를 데이터베이스에 저장하기 위한 데이터베이스 연동 프로그램을 개발한다.

11. 조회 및 관리 프로그램 개발

관리자가 원격지의 저온저장고의 상태를 조회하고 온·습도 조정 등의 관리를 위한 관리 프로그램을 개발하며 사용자가 저온저장고의 상태를 확인할 수 있는 사용자 조회프로그램을 개발한다.

12. 시작품제작

연구 목표에 따른 시작품을 제작한다.

Ⅳ. 연구개발 결과 및 활용에 대한 건의

1. 연구개발한 결과는 인터넷을 이용한 원격 감시 및 제어 기술이다. 인터넷을 이용한 원격 감시 및 제어 기술은 인터넷 통신망을 통해 원거리에서 저온저장고의 환경상태를 감시하고 최적의 환경상태를 유지할 수 있도록 온·습도 장치를 원격 제어하는 기술이다.
2. 활용분야는 저온저장고를 관리하는 유통 분야에 적용할 수 있다.
3. 활용유형은 주관기관 명의로 특허를 출원하고, 관련업체에 기술을 이전하여 상품화한 후, 저온저장고를 보유한 농가 또는 관리전문 업체에 상품을 보급하여 농가의 애로사항을 해결할 수 있도록 한다.

4. 활용방안은 저온저장고를 보유한 농가에서 관리할 수 있는 방안과, 저온저장고의 관리를 관리 전문업체에 위탁하여 위탁받은 전문업체가 원격 중앙 집중관리를 위해 활용하는 방안이 있다.
5. 현장보급 방안 및 산업화계획 방안은 기술을 이전 받은 업체가 산업화하며 개발된 상품을 농가와 관리 대행업체에 보급한다.
6. 추가기술 방안 및 기술이전 방안은 정보 통신 관련업체 또는 저온저장고를 시공하는 업체에 기술을 이전한 후 현장감각에 맞도록 추가되는 기술은 기술을 이전받은 업체가 현장감을 살린 기술을 추가로 개발한다.

SUMMARY
(영문요약문)

Farmers want to get the low temperature storage house for agricultural products. So storage house have increased in the country. But farmer is not accustomed to manage the storage house. The farmer want to be managed the storage house safely. The farmer want to entrust management to the expert. We need to make system that monitoring and control environment from the long distance.

The purpose of this study is the development of remote monitoring and remote control system using internet. The system is applied to low temperature storage house. The targets are temperature and humidity. This system is applied monitoring and control of temperature and humidity in the low temperature storage house. We can monitor temperature and humidity at near and far away. We can control temperature and humidity at near and far away also.

The system has server and client. One server can monitor one client or many clients at the same time. When many clients are used the system, many farmers can entrust management of his low temperature storage houses with a expert. This is called central control system. The system can realize to optimal control for temperature and humidity. The system can be sure farmer from non-steady state. So the farmer is devote to an occupation.

The controller has multi-channels for sensors. We designed to connect four temperature sensors and four humidity sensors. The temperature and humidity in the storage house is different at the each position. When we apply one temperature sensor. The agricultural products can suffer damage due to cold weather. Because of lowest temperature in the storage house is less than set-point temperature. We select a average value or a lowest value of several temperature and humidity value.

CONTENTS
(영 문 목 차)

목 차

제 1 장 연구개발과제의 개요

제 1 절 연구개발의 목적 및 필요성

WTO 체제하에 세계 무역은 완전 개방되었고 미국과 중국 등 농업 대국의 농산물이 밀려오는 현실 앞에 우리의 농업은 어려움에 직면해 있으며 앞으로 국내 농업은 끊임없는 노력으로 이를 타파해 나가야할 것이다. 이러한 환경변화로 인하여 우리 농업은 외국 농산물과 차별화해야 하며, 특히 이웃 중국의 값싼 농산물에 대해서는 우수한 품질로 대응할 수밖에 없다. 우수한 품질은 종자, 재배, 수확, 유통 등 모든 생산과정에서 최선의 관리가 있어야 하겠지만 특히 수확 후 농산물의 품질은 유통 과정에 달려있다 할 수 있다. 품질 유지를 위해 수확 후 산지에서 소비자에게 까지 유통과정 중 신선도를 유지하는 것이 무엇보다도 중요하다 할 것이다. 신선도 유지 기술은 수입 농산물에 대한 경쟁력을 확보하는 입장과 국내 농산물을 외국에 수출하는데 있어서 경쟁력을 확보하는 방법이기도 하다. 저온저장고는 농산물의 신선도를 유지하는 수단이다. 또한 수확 후 수 개월에 걸쳐 출하시기를 조절할 수 있어 농가 수입에 직결되는 농업 시설이다. 저온저장고에서 관리하는 환경은 주로 온도와 습도이며 품목에 따라 이들의 환경을 제어함으로써 신선도를 유지할 수 있도록 한다.

농가에서 저온저장고의 필요성을 인식하고 농가 단위로 저온저장고를 보유해 가는 추세이다. 그런데 저장고를 관리하는 데에 어려움이 있다. 저장고의 환경상태를 확인하는 표시기가 저장고에 부착되어 있어서 직접 저장고까지 가야만 환경상태를 확인할 수 있다는 점과, 저장고 관리자가 저장고로부터 멀리 떨어진 곳에 있는 동안이나 취침 등과 같은 감시가 불가능한 경우에 발생할 수 있는 돌발 상황에 대처할 수 없다는 점이다. 저온저장고의 환경 상태를 감시하고 최적의 환경 상태를 유지할 수 있도록 온·습도를 원격 제어할 수 있는 시스템을 개발하여 농가에 설치하면, 저장고까지 가지 않고도 농가의 거실이나 안방에서 저장고의 환경을 감시하고 제어할 수 있다. 또 원거리에서 감시하고 제어할 수 있는 시스템을 구축하면 환경 상태의 감시 및 제어를 전문가에게 위탁할 수 있다.

저장고의 환경상태가 비정상 상태이거나 저장고의 장치들이 고장 시에 저장

농산물에 치명적 손상을 가져올 수 있기 때문에 저온저장고에 대한 전문지식이 없는 농가에서 저장고를 관리하는 것 보다는 전문가가 인터넷을 이용하여 각 농가에 설치된 저온저장고의 환경상태를 감시하고 최적의 환경으로 제어할 수 있는 시스템을 구축하여 저장고 관리의 효율성을 높일 수 있는 기술을 개발함으로써 저온저장 분야의 기술을 향상시킬 수 있다.

본 연구에서는 인터넷을 이용하여 원거리에서 저온저장고 내의 환경상태를 감시하고 또 정상 상태가 아닐 경우 원거리에서 제어할 수 있는 저온저장고의 원격 감시 및 제어 시스템을 개발하였다.

연구개발 범위는 농가의 거실이나 안방에서 저온저장고의 온·습도를 감시하고 제어하는 시스템과, 전문가가 저장고를 보유한 다수의 농가들로부터 저장고의 관리를 위탁받아 원거리에서 온도와 습도를 중앙 집중 관리하는 온·습도 중앙 집중 감시 및 제어 시스템을 개발하는 것이다.

본 연구결과를 활용하면 농가에서 관리하는 경우는 방안이나 거실에서 저장고를 관리하기 때문에 저장고까지 직접 접근하지 않아도 되고, 중앙 집중관리의 경우는 전문가가 저온저장고를 관리하기 때문에 저장고내의 환경상태를 최적으로 관리할 수 있으며, 저온저장고를 위탁 관리해 주기 때문에 농가에서는 안심하고 생업에 종사할 수 있다.

제 2 절 연구개발 내용 및 범위

본 연구의 대상은 나주 지역에 있는 배 저장 시설을 중심으로 한다. 농가의 거실이나 안방에서 저장고의 환경을 감시하고 제어할 수 있는 시스템과, 여러 농가들에 대한 저온저장고의 환경상태를 전문업체가 집단으로 중앙 관리할 수 있는 시스템을 구현하기 위하여 인터넷상에서 저온저장고의 환경변화를 감시하고 최적의 상태를 유지하도록 원격 제어하는 시스템을 개발하는 것이 본 연구개발의 목적이며 목적을 달성하기 위한 세부 연구개발의 내용과 범위는 다음과 같다.

1) 자료조사

저온저장고의 온도와 습도의 측정을 위하여 온도·습도 측정기에 대한 정보들을 조사하고 원격제어의 방법에 대하여 조사하며 인터넷 보급률에 관하여 조사한다.

2) 제어시스템 구성

저온저장고의 온·습도를 제어하기 위한 저온저장고용 컨트롤러를 개발하고 제어 알고리즘 구현한다.

3) 측정시스템 구성

저온저장고의 실시간 온·습도 측정을 위한 측정시스템을 구성하고 저온저장고의 컨트롤러와 측정시스템간 인터페이스를 구축한다. 또한 정밀 측정을 위하여 추가적인 측정 센서를 설치하는 정밀 온·습도 측정을 위한 시스템을 설계하고 센서의 정밀도 향상을 위한 처리 알고리즘 구현한다.

4) 제어·측정시스템과 PC간 통신 프로토콜

제어·측정시스템과 PC간의 통신을 위하여 통신 프로토콜을 정의한다. 통신을 위한 하드웨어를 설계하고 통신프로그램을 작성한다.

5) PC기반 제어 프로그램 개발

농가의 PC에서도 직접 저온저장고의 온·습도를 측정 및 제어할 수 있는 PC 기반의 프로그램을 설계 및 구현하고 저온저장고의 관리 프로그램을 작성한다.

6) 저온저장고 온·습도 제어실험

온도와 습도의 상관관계를 조사하고, 시스템 오작동에 대한 안정성 확보 방안을 수립한다.

7) 데이터베이스 구축

클라이언트에서 서버로 전송된 데이터를 저장하기 위하여 데이터베이스를 구축한다.

8) Client와 Server간 프로토콜

Server와 Client간의 통신을 위하여 Server와 Client간 통신 프로토콜 정의하고 이를 구현한다.

9) Client용 통신 프로그램 개발

제어·측정시스템에서 측정된 온·습도 데이터의 수집을 위한 Client 정보 수집 프로그램을 개발하고 이를 Server에 전송하기 위한 네트워크 접속 프로그램을 개발한다. 이를 통하여 제어·측정시스템의 정보를 Server로 전송하는 제어 및 감시 시스템 상태정보 전송 프로그램을 개발한다.

10) Server용 프로그램 개발

농가 PC에서 접속이 있을 경우 먼저 타당하고 인증된 PC인가를 결정하기 위한 Client의 인증 처리 프로그램을 개발하고 Client와의 통신 접속을 위한 네트워크 접속 프로그램을 개발한다. 또 클라이언트에서 전송된 정보를 관리하기 위한 클라이언트 전송 정보 관리 프로그램을 개발하며 클라이언트에서 전송한 데이터를 데이터베이스에 저장하기 위한 데이터베이스 연동 프로그램을 개발한다.

11) 조회 및 관리 프로그램 개발

관리자가 원격지의 저온저장고의 상태를 조회하고 온·습도 조정 등의 관리를 위한 관리 프로그램을 개발하며 사용자가 저온저장고의 상태를 확인할 수 있는 사용자 조회프로그램을 개발한다.

12) 시작품제작

연구 목표에 따른 시작품을 제작한다.

제 2 장　국내외 기술개발 현황

제 1 절　국내·외 관련기술의 현황과 문제점

1. 국내기술현황

국내 원예산물의 경우 수확 후 관리기술 수준이 선진국에 크게 낙후되어 이로 인한 손실은 20%~50%에 달하고, 콜드체인시스템의 보급비율은 현재 5%이하에 불과한 실정이다.

국내에 보급된 저온저장고는 2000년 말 기준으로 8,766개소에 341천 평에 달하고 그 이후 보유 농가가 계속 증가하고 있으며 저온저장고에 대한 기술도 연구되어지고 있는 추세이다. 그러나 현재 과수 농가에 설치된 저장고의 원격 감시 및 제어 시스템은 아직 초기 단계라 할 수 있다. 감시 시스템은 인터넷이나 이동통신을 이용하여 연구되어지고 있으나, 원격 제어시스템은 거의 전무한 상태이다. 특히 인터넷을 이용하여 원거리에서 중앙 집중 관리하는 중앙 집중 감시 및 제어 시스템은 아직 개발되지 않은 상태이다.

2. 국외기술현황

최근 선진국들의 저온유통 기계기술 개발 동향은 원예 산물의 신선도를 기존의 저장설비 보다 훨씬 더 연장시키면서도 저비용, 생 에너지, 환경 친화적인 기술개발에 중점을 두고 있다. 생산된 농산물의 유통개념을 연중 출하를 목적으로 하기 위해, 거의 모든 농산물의 저온저장에 CA저장고를 사용하고 있다. 에너지 절감 및 효율성을 높이기 위한 축열방식 도입, 전처리를 위한 예냉시스템과 콜드체인시스템이 일반화되어 있으며, 환경 제어방법도 정밀화 되어 있다. 인터넷을 이용한 감시 및 제어기술은 저온저장고를 비롯한 시설하우스 등에 적용되고 있다.

제 3 장 연구개발 수행내용 및 결과

1. 자료 조사

가. 온도·습도 측정기의 정보조사

1) 온도측정기 정보조사

온도 측정기는 크게 접촉식 방법과 비접촉식 방법으로 나뉘며, 측정 방법에 따른 분류로는 기계식 온도측정기, 저항온도계, 서미스터, 열전온도계, 방사온도계, 적외선 열상계, 열류계 등이 있다.

다음에 개략적인 온도 측정기의 원리 및 특징을 기술한다.

표 1 접촉식과 비접촉식 온도 센서의 비교

	접촉방식	비접촉 방식
필요조건	·측정대상과 센서를 잘 접촉시킬 것 ·측정대상에 센서를 접촉시켰을 때 측정대상의 온도가 변화하지 않을 것	·측정대상에서의 방사가 충분히 센서에 도달할 것 ·측정대상물의 실효 방사율이 명확하게 알려져 있거나 혹은 재현 가능할 것
특징	·열용량이 적은 측정대상에서는 센서접촉에 의한 측정대상 온도 변화가 쉬움 ·움직이는 물체는 측정하기 어려움	·측정대상 온도가 변화하지 않음 ·움직이는 물체 측정 가능 ·물체의 표면온도 측정
정밀도	·높음	·접촉방식에 비해 낮음
응답속도	·늦음	·접촉방식에 비해 빠름
종류	·기계식, 저항식, 서미스터, 열전대	·방사온도계, 적외선열상계, 열류계

가) 기계식 온도측정기

금속, 액체, 기체 등의 온도에 의한 팽창과 수축하는 원리를 이용하는 측정기이다. 기계식에는 바이메탈온도계와 같은 금속팽창식 온도계, 측온부에 봉입한 열팽창에 따른 압력변화로 브로돈관을 변위시켜 온도를 측정하는 액체압력식 온도계, 감온부에 봉입한 액체의 포화증기압이 온도에 따라 변화하는 것을 이용하는 증기 압력식 온도계 등이 있다.

나) 저항온도계

접촉식 온도계로 열전온도계와 함께 자주 사용되고 있는 온도계로 온도에 의해서 변화하는 물질의 전기저항을 이용하는 원리를 이용하는 것으로 가장 안정적이고 측온범위가 없기 때문에 고정밀도를 필요로 하는 온도계측에 주로 사용되고, 저온저장고에서도 대부분 사용되고 있다.

저항온도계는 안정도가 높고, 감도가 크고, 기준접점 보상회로가 필요 없으며, 비교적 간단한 부가회로로 직선출력을 얻을 수 있는 장점이 있으나, 저항소자의 구조가 복잡하기 때문에 외형이 크고, 고온은 측정할 수 없으며, 기계적인 진동이나 충격에 약하다.

다) 서미스터

여러 종류의 금속산화물을 혼합 소결하여 제작되었으며, 그 저항온도계수는 금속에 비해 약 10배 정도 크며 감온부를 작게 제작할 수 있으므로 고감도로 응답이 빠르며 협부의 측온을 행하는 곳 등에 사용된다.

라) 열전온도계

상호 다른 2종의 금속 또는 합금으로 된 선이 한 끝에서 접속되어 있을 때 이 접속점과 기준 접점간에 온도차가 있으면 열기전력이 발생한다. 이 2개의 선을 조합한 것을 열전대라고 한다.

기준접점의 온도를 알면 측정된 열기전력으로부터 측정개소의 온도가 구해진다.

마) 기타 온도계

방사 온도계, 적외선 열상계, 열류계 등

2) 습도측정기의 정보조사

농산물의 저장에 있어서 수분을 유지시키기 위해 해당 농산물에 적합한 습도의 유지는 중요한 요소이다. 습도를 측정할 수 있는 습도센서는 건구와 습구의 온도차로부터 습도를 측정하는 건습구습도계, 모발 길이의 변화로써 측정하는 모발습도계 그리고 전기적 양의 변화를 이용하는 정전용량식 습도센서, 전기저항식 습도센서 등이 있다.

가) 건습구습도계 (psychrometer)

건구습도계(dry-bulb)와 습구습도계(wet-bulb)로 구성되어 있다. 두 습도계의 온도 차이가 커지면 상대습도는 더 낮아지고, 온도 차이가 작아지면 상대습도는 더 높아진다. 건습구 대조표를 통해 상대습도를 알 수 있다.

나) 모발습도계 (hair hygrometer)

상대습도가 높아지면 머리카락이 늘어나고, 낮아지면 머리카락이 오므라든다. 많이 사용되고는 있지만 건습구 습도계보다 정확성이 떨어지고 자주 계산해야 하며, 습도변화에 반응하는데 많은 시간이 걸린다. 특히 낮은 온도에서 그렇다.

다) 정전용량식 습도센서

정전용량식 습도센서는 습도의 변화와 함께 센서 단자간의 정전용량이 변화하는 센서로서 전기저항식 습도센서와 함께 현재 많이 사용되는 센서 중의 하나이다. 본 연구에서는 정전용량식 습도센서를 사용하였다.

라) 전기저항식 습도센서

금속산화물의 고온소결에 의한 다공질 세라믹을 베이스로 하여 소결체의 작은 구멍에 물분자가 흡착되어 전기저항이 변화하는 현상을 이용한 것으로 측정범위가 넓고 연속측정이 가능하지만 시간의 경과에 따라 특성이 변화하므로 정확성이 떨어지는 문제점이 있다.

3) 저온저장고용 온도 습도 센서

저온저장고와 각종 계측장비의 센서로는 pt100Ω이 많이 사용된다. pt100Ω은 정밀한 측정을 할 수 있고 내구성이 강한 특징을 가지고 있다. 따라서 본 연구에서는 저항온도 센서인 pt100Ω을 사용하여 시스템 개발을 수행하였다.

또한 습도센서로는 HIH-3610 Series를 사용하였다. HIH-3610 Series 습도센서는 가공된 열경화성수지의 커버로 덮여있고, 저 전력에서의 동작, 높은 정밀도, 빠른 반응, 안정되고 신뢰할만한 동작 등의 특징이 있다. 본 연구에서는 정전용량식 습도센서인 HIH-3610 Series 습도 센서를 사용하였다.

4) 습도 제어시스템(가습기)의 종류

습도를 제어하는 가습방법에는 초음파 가습기(이온발생기), 가열식 가습기, 복합식 가습기, 자연식 가습법 등이 있다.

가) 초음파 발생기(이온발생기)

물을 넣는 용기의 밑 부분에서 초음파를 발생시켜 물을 작은 입자로 내뿜는 방식으로서 적은 전력으로 많은 분무량을 낼 수 있는 장점이 있으나, 가습기에 들어 있는 물에 미생물이 번식할 우려가 있고, 차가운 수증기가 나오는 관계로 실내온도가 낮아지는 단점이 있다.

나) 가열식 가습기

가습기 내부에서 물을 끓여 수증기로 뿜어 주는 방식으로서 완벽한 살균이 가능하다는 장점이 있지만, 가습량이 적고 전력 소모가 많다.

다) 복합식 가습기

초음파식과 가열식의 장점을 취하는 방식으로, 먼저 가열관에서 물의 온도를 60~65℃로 올려 살균 시킨 후, 초음파를 이용해서 뿜어 주는 방식이다.

라) 자연식 가습기

수분과 바람을 이용하여 가습을 가하는 방법이다. 가장 자연에 가까운 수분을 얻을 수 있는 방법이며, 입자가 매우 작아 저온에서 결빙 가능성이 적다.

나. 원격제어 방법에 대한 조사

1) 원격제어

원래 원격제어는 RS-232C와 같은 네트워크 케이블을 이용해 공장의 생산기계를 중앙에서 콘트롤 할 때 사용하던 기술이다. 이러한 기술은 점차 발전하여 이제는 각종 네트워크 장비나 프로그램을 개발할 때 관리자가 편리한 장소에서 조작할 수 있는 원격접속 기능을 포함시킬 정도로 보편화되었다. 기존의 컴퓨터를 이용한 분산 제어시스템의 하위 레벨에서는 RS232C/ 422A/ 485 등 자체적인 네트워크를 형성, 제어 감시 데이터를 컴퓨터로 수집하고 상위 레벨에서는 Ethernet 등의 망을 이용하여 데이터 통신하는 시스템을 구성하고 있다. 그러나 최근에는 자동화시스템이 고속, 복잡, 초정밀화 되어감에 따라 호스트 중심의 수직구조를 가진 하위 레벨 네트워크에서는 배선이 복잡해져 자동화시스템을 유지보수 하는데 많은 비용 부담이 발생하고 있다 . 또한 회사마다 전송방식이 다르므로 이기종의 자동화 장비들 간의 통신에 어려움이 있으며 RS232C/ 422A/ 485와 같은 비동기 통신 방식은 데이터 전송속도와 시스템을 구성하는 스테이션의 수가 제한되어 있다. 그러므로 원거리 통신망에는 적합하지 않으며 원격제어가 불가능하다. 따라서 하위 레벨에서도 기존의 Ethernet 망과 표준 프로토콜을 이용하면 위의 문제점들을 보완할 수 있게 된다.

생산 자동화 설비를 구축함에 있어서 기기들 간의 정보 교환을 위한 실시간 네트워크는 중요한 구성요소 중의 하나이다. 1980년대에 미국의 보잉사와 G/E사를 중심으로 상호 호환성이 없는 이기종의 자동화 장비들 간의 통신을 위한 표준화된 네트워크 시스템으로 MAP(Manufacturing Automation Protocol)이 개발되었다. 그러나 자동화 네트웍시스템의 연결 대상이 점차 공정 하부 기기로 내려감에 따라 OSI(Open System Interconnection) 표준모델에서 제시하는 7계층 구조를 모두 가지고 있는 MAP을 채용하기에는 실시간 통신에 여러 가지 문제점이 대두되었고 이에 물리층, 데이터링크층, 응용층의 3계층으로만 이루어진 mini-MAP이 개발되었다. 한편 유럽을 중심으로 제어기기들 간의 통신을 위한 비교적 저가이고 통신에 필요한 최소한도의 기능을 수행하는 단순한 구조를 가지면서도 빠른 응답시간으로 실시간 처리에 적용될 수 있는 네트워크 시스템으

로서 필드버스 네트워크가 개발되었다. 필드버스 네트워크는 CIM(Computer Integrated Manufacturing)의 계층구조상 가장 하위 계층인 생산 필드의 분산 공정 제어에 있어서 센서, 공정제어기, 액츄에이터, 계측기기, 스위치, 분석기 등과 같은 제어기기들 간의 통신 네트워크시스템으로서 프로토콜로는 SERCOS, CAN, WorldFT P, Profibus 등이 있으며 이들은 OSI의 7계층에서 물리계층, 데이터 링크 계층, 응용 계층의 3계층과 새롭게 설정된 사용자 계층(user layer)으로 구성되어 있다 . 최근에는 LAN이 근거리 통신망에서 가장 널리 사용되고 있으며 인터넷의 급속한 보급으로 인하여 TCP/ IP에 대한 중요성이 크게 부각되고 있다. 이러한 TCP/IP 프로토콜은 꾸준이 연구되고 개량되어 매우 안정적이며 수많은 응용 프로그램을 가지고 있어 자동화시스템과 인터넷을 통한 감시 제어 시스템에서 그 이용도가 급증하고 있다.

2) 인터넷을 이용한 원격 감시 제어 시스템

인터넷이나 정보 통신망을 이용하여 멀리 떨어져 있는 기기를 제어(Control)하거나 감시(Monitering)하는 방법이 원격 제어시스템이다. 기존의 방식으로는, 전화나 모뎀을 통해 접속하여 제어 신호를 보내는 방식과 직렬 통신을 이용하여 서로 데이터를 주고받는 방식이 있다. 그러나 기존의 방식은 직렬 통신(RS-232, 422, 485)을 이용한 양방향 제어방식의 경우 거리의 제한이 있고, 전화를 이용한 단방향 제어방식은 제어 대상이 정확하게 제어되었는지에 대한 확인이 불가능하다는 문제점을 지니고 있다. 또한 이러한 방식은 하나의 모니터링 시스템이 여러 기기를 동시에 제어할 수 없다. 이런 문제점을 극복하기 위해서는 인터넷이나 정보통신망을 이용하는 것이 좋다. 이 방법을 이용하면 원거리에서도 기기들을 실시간으로 관리할 수 있으며 직접 제어도 가능하게 된다.

임의의 클라이언트에서 원격의 대상체를 제어하기 위해서는 대상체의 작업환경과 동작을 파악해야한다. 원격조작 시스템은 제어 기법에 따라 단방향 제어와 양방향 제어, 반이중 제어로 분류할 수 있다. 단방향 제어의 경우 작업자는 시각정보만을 이용하여 비교적 간단한 작업의 수행만이 가능하다. 그러나 양방향 제어시스템에서는 제어결과가 작업자에게 feedback 되므로 결과의 확인이 가능하다. 그러나 양방향 시스템의 경우 거리가 멀 경우 시간지연에 의해 전체적으로 시스템이 불안정하게 동작할 수 있으며 서버와 지속적인 접속이 이루어져 있어

야하므로 대상이 많아지면 네트워크 커넥션 자원을 많이 점유하게 되어 정상적인 제어 및 감시에 어려움을 겪게 된다. 반 이중 방식은 서버와 지속적인 커넥션을 유지하는 것이 아니라 클라이언트에서 작업 요청이 있을 경우에만 서버에 접속하여 클라이언트의 상태 정보를 서버에 전송하고 서버에서 내리는 명령 중 자신에게 할당된 명령만을 가지고 오게 된다. 이렇게 함으로써 다수의 대상이 존재한다고 할지라도 양방향 방법과 똑같은 결과를 얻으면서 효과적으로 네트워크 커넥션자원을 이용할 수가 있게 된다. 특히 HTTP를 사용하면 기존에 서비스 중인 웹 서비스에 일부로서 사용하게 되고 기존에 서비스 중인 시스템과 연동이 가능하므로 관리자에게 향상된 정보를 제공할 수 있게 된다.

3) 무선 제어

무선 LAN은 주로 구내통신용으로 이미 오래 전부터 개발되어온 기술로서 특히 주파수 대역별로 다양한 표준기술이 존재하고 있다. 표 2는 무선 LAN 표준기술 현황을 주파수대 별로 정리한 것이다.

표 2 무선 LAN 표준기술 개발 현황

주파수 대	방식구분	표준화 및 기술특성
90MHz	독자 방식	1Mbps 이하
2.4GHz	독자 방식	FHSS 1Mbps
	IEEE802.11	'97.6, 표준화, 2Mbps
	IEEE802.11TGb	'99.5, 표준화 Ⅰ, 11Mbps
5GHz	독자 방식	10Mps(5.7GHz)
	IEEE802.11TGa	'99.5, 표준화, CFDM
	ATM	25Mbps HiperLAN2
60GHz	ATM	155Mbps 완전 ATM

무선 LAN은 특정 구역 내에서 활용할 수 있다. 전체 네트워크를 수용할 수 있는 구조로 설치되기에는 주파수 간섭이나 전달 영역 한계의 모호성, 보안 및 데이터 노출에 의한 불안정성 등의 문제점이 지적되고 있어 일부 영역을 소화하

는 범위에서 응용되는 것이 안정적이라 할 수 있다.

무선 LAN 구축 방안으로 ,RF, Bluetooth, 원격 무선계측기(RT) 등의 방법이 연구되고 있다.

4) 저온저장고 제어기기 제어방법

일반적으로 제어시스템은 센서, 제어기, 구동장치 등으로 구성되며, 제어 신호의 형태에 따라 아날로그제어, 디지털제어, 순차제어(ON/OFF제어)로 구분된다. 대부분은 가장 간단한 제어방법인 ON/OFF 제어방법이 많이 사용되지만, 특수한 경우 아날로그 제어(속도제어)가 사용되기도 한다. 이러한 아날로그 제어, 디지털 제어는 ON/OFF 제어에 비하여 가격이 많이 들고, 유지보수가 불편한 단점을 가지고 있다. ON/OFF 제어방법을 구성할 수 있는 제어소자로서는 매케니컬 릴레이, 솔리드스테이트 릴레이, 포토 MOS릴레이 등이 있다. 다음은 본 연구에서 사용될 수 있는 ON/OFF 제어용 소자들이다.

■ 매케니컬 릴레이

그림 1은 프린트 기판 실장 타입 릴레이의 예 이다. 접점의 허용 전류에 따라 다양한 크기와 종류가 있으며 동시에 움직이는 점점의 수에 따른 종류도 다양하다.

그림 1 메케니컬 릴레이

■ 솔리드스테이트 릴레이SSR)

반도체로 구성된 릴레이로, 그 원리는 포토커플러와 동일하며 발광다이오드와 빛 트리거 타입의 트라이액을 마주보게 하여 몰드 한 것이다. 트라이액이기 때문에 제로크로스 스위치로 동작하여 ＡＣ 전류를 제로크로스로 On/Off

할 수 있다. 소형이며 스파크가 발생하지 않고 절연되는 메리트가 있기 때문에 릴레이 대신에 많이 사용되고 있다.

그림 2 솔리드스테이트 릴레이

■ 포토 MOS 릴레이

포토커플러와 완전히 동일한 구성에서 포토셀과 발광다이오드를 마주보게 몰드한 것으로 포토셀에는 MOS형 FET가 내부에서 접속 구성되어 있다. MOS형 FET에는 내압이 400[V] 이상인 것도 있어서 고압 고전류 제어도 가능하다.

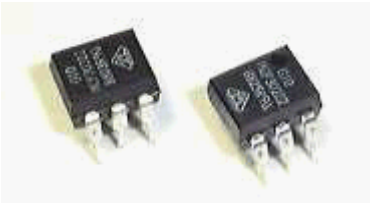

그림 3 포토MOS 릴레이

다. 인터넷 보급률 조사

2001년 12월말 통계에 의하면 우리나라의 PC 총 보유대수는 22,495천 대로 추정되는 가운데, 부문별로는 가구부문이 전체의 57.0%인 12,812천 대를 보유하고 있으며, 종사자수 5명 이상 사업체가 전체의 43.0%인 9,683천대를 보유하고 있는 것으로 추정된다. 전체 보유 PC들을 종류별로 살펴보면, 데스크탑 컴퓨터가 전체의 88.5%(19,903천 대)를 차지하고 있으며, 노트북은 아직 10.0%(2,261천 대)

에 머물러 있는 것으로 나타났다. 또한, 전체 19,903천 대의 데스크탑 컴퓨터 중 팬티엄3 이상 기종은 58.4%(11,624천 대), 팬티엄2 이하 팬티엄급은 35.9%(7,137천 대)를 차지하고 있는 반면, 486 이하 기종은 전체의 5.7%(1,141천 대)에 불과한 것으로 나타나, 현재 전체 가구와 종사자수 5명 이상 사업체들에서 사용되고 있는 데스크탑 컴퓨터의 주종은 팬티엄3 이상임을 알 수 있다.

1) 사업체부문 네트워크 구축

2002년 6월말 현재, 금융을 중심으로 한 서비스업의 네트워크 구축 비율이 다른 업종에 비하여 두 배 이상을 나타내고 있으며 농림수산업, 경공업, 석유화학 산업 등에서는 30% 이하의 구축률을 보이고 있다.

표 3 네트워크 구축 현황 (단위 : 사업체수, %)

구분	전체 사업체	구축		구축중		구축예정		현재미구축이나 향후에도 구축계획없음		해당없음	
		사례수	비율	사례수	비율	사례수	비율	사례수	비율	사례수	비율
전체	442,655	180,149	40.7	5,708	1.3	10,423	2.4	161,521	36.5	84,853	19.2
업종별											
농림 수산업	2,781	739	26.6	-	-	-	-	1,199	43.1	843	30.3
경공업	39,301	11,062	28.1	255	0.6	2,058	5.2	13,844	35.2	12,082	30.7
중공업	43,374	17,630	40.6	320	0.7	1,033	2.4	19,522	45	4,869	11.2
석유화학	20,249	5,620	27.8	256	1.3	386	1.9	7,495	37	6,492	32.1
건설업	31,161	10,887	34.9	171	0.5	707	2.3	16,718	53.6	2,678	8.6
유통업	149,931	45,539	30.4	3,659	2.4	2,861	1.9	51,328	34.2	46,544	31
금융보험업	25,261	21,796	86.3	310	1.2	685	2.7	1,728	6.8	742	2.9
기타 서비스업	130,597	66,874	51.2	737	0.6	2,694	2.1	49,687	38	10,605	8.1

* 기준시점 : 2002년 6월 30일

* 네트워크 : 모뎀을 제외한 전용통신망

2) 인터넷 주 이용장소

2002년 6월 현재, 인터넷 이용자들이 인터넷을 주로 이용하는 장소는 가정

(87.9%), 회사(23.1%), PC방(15.7%), 학교(11.0%), 공공장소(1.1%) 등의 순서인 것으로 나타났다. 즉 최근 몇 년간 가정내 초고속인터넷 구축률이 대폭 증가하면서 이제 인터넷 이용자들의 일반적인 인터넷 접속 장소는 PC방과 같은 외부 장소가 아닌 가정이 되었음을 알 수 있다.

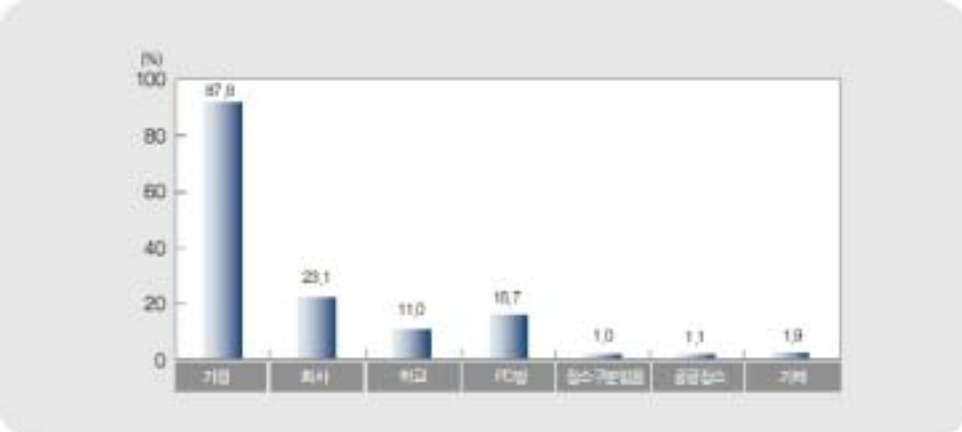

그림 4 인터넷 이용자수 및 이용형태

* 출처 : 한국인터넷정보센터(2002 인터넷 이용자수 및 이용형태 조사)
* 기준시점 : 2002년 6월
* 조사대상 : 만 6세 이상 전체 인구
* 주 : 응답자 기준 복수 응답 수치임

3) 연도별 인터넷 이용자수

인터넷 이용자 증가율은 1999년(71%)을 기점으로 점점 감소하고 있는 것으로 보이나 인터넷 이용자의 수는 계속 증가 추세에 있다.

표 4 연도별 인터넷 이용자수 (단위 : 천명)

구분	1994년	1995년	1996년	1997년	1998년	1999년	2000년	2001년	2002년 6월
이용자수	138	366	731	1,634	3,103	10,860	19,040	24,380	25,650
증감		228	365	903	1,469	7,757	8,180	5,340	
증감율		62%	50%	55%	47%	71%	43%	22%	

출처 : 한국인터넷정보센터(KRNIC)
증감율: 증감/이용자수 *100

4) 인터넷 이용자

인터넷 이용에 있어서는 전 연령층에서 인터넷 사용자의 50% 이상이 매일 사용 하고 있으며 10대와 20대에서는 매일 사용자가 60% 이상을 보이고 있다. 또한 인터넷 사용자중 50~60대의 90%이상이 주 1~2회 이상 인터넷을 사용하는 것으로 나타났다.

표 5 2002년도 인터넷 이용자수 및 이용형태

구분	인터넷 이용자	매일		주 3~4회		주 1~2회		월 3~4회		월 1~2회	
		사례수	비율	사례수	비율	사례수	비율	사례수	비율	사례수	비율
전체	25,652,883	15,956,093	62.2	4,232,726	16.5	4,489,255	17.5	256,529	1	718,281	2.8
성별											
남자	14,045,204	9,382,196	66.8	2,092,735	14.9	2,134,871	15.2	112,362	0.8	323,040	2.3
여자	11,585,995	6,546,087	56.5	2,131,823	18.4	2,363,543	20.4	150,618	1.3	405,510	3.5
연령											
6-19세	8,706,413	5,319,618	61.1	1,567,154	18	1,645,512	18.9	60,945	0.7	113,183	1.3
20-29세	7,087,118	5,209,032	73.5	956,761	13.5	751,235	10.6	56,697	0.8	106,307	1.5
30-39세	5,915,506	3,425,078	57.9	1,070,707	18.1	1,106,200	18.7	82,817	1.4	236,620	4
40-49세	2,973,724	1,486,862	50	481,743	16.2	734,510	24.7	53,527	1.8	217,082	7.3
50-59세	779,407	390,483	50.1	118,470	15.2	208,102	26.7	10,132	1.3	52,220	6.7
60세 이상	173,166	101,129	58.4	27,880	16.1	39,482	22.8	0	0	4,849	2.8

* 출처 : 한국인터넷정보센터(2002 인터넷이용자 수 및 이용행태 조사)
* 기준시점 : 2002년 6월
* 가구원 : 만 6세 이상(통계청, 2001년 전국주민등록인구통계 참조)
* 인터넷이용자 정의 : 월 평균 1회 이상 인터넷을 이용하는 만 6세 이상의 인구

5) 인터넷 보급지역

정보통신부가 2002년 11월 29일에 발표한 “2002년도 종합자체심사평가보고서”의 정보화 부문 성과에서 “2002년 9월 현재 전국의 읍(100%), 면(98%) 지역에 초고속정보통신 서비스 제공 환경구축” 이라는 결과에서와 같이 전국 농가에 인

터넷 사용을 위한 인프라가 구축되어 있다는 것을 알 수 있다. 따라서 전국 어떤 농가의 저온저장고를 대상으로 하더라도 인터넷을 이용한 원격 감시 및 제어가 가능하다는 결론을 얻을 수 있다.

2. 제어 및 측정시스템

본 연구는 인터넷상에서 저온저장고의 환경변화를 감시하고 최적의 상태를 유지하도록 원격 제어하기 위한 '저온저장고 제어 및 측정시스템'을 개발하였다. 개발한 제어 및 측정시스템은 기존의 현장에서 직접 제어하는 시스템과는 달리 원거리 송 수신을 위한 통신기능이 추가되어 있으며, 온·습도 센서값 처리, 온·습도 목표값 제어, 목표값과 현재값을 비교 처리하는 알고리즘 그리고 외부 기기와의 통신을 위한 인터페이스 기능 등을 구현하도록 설계하였다. 제어 및 측정시스템은 마이크로프로세서, A/D컨버터, 온도·습도센서 처리부, 통신부, 입력부, 출력부, 표시부로 구성되어 있다.

본 연구에서 개발한 저온저장고용 제어 및 측정시스템의 블럭도는 그림 5와 같다.

■ 온도센서

저온저장고 내부의 온도값을 감지하여 A/D컨버터로 보낸다. 본 연구에서는 온도센서로 PT100Ω을 사용하였다.

■ 습도센서

저온저장고 내부의 습도값을 감지하여 A/D컨버터로 보낸다. 사용된 습도센서는 HIH-3610이다.

■ Micro Processor

제어 및 측정시스템의 제어를 수행하는 Micro Processor는 AVR칩 시리즈인 ATmega128을 사용하여 구현하였다.

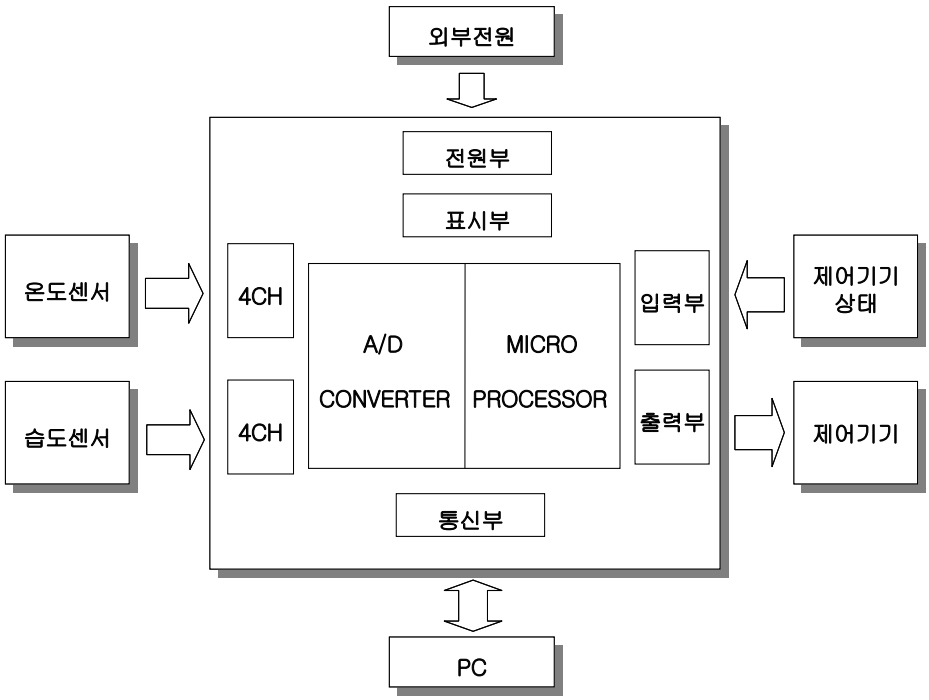

그림 5 제어 및 측정시스템의 블럭도

본 연구에서 사용된 AVR칩은 Atmel사에서 만든 RISC 타입의 고속 8비트 마이크로 컨트롤러로 PIC보다 처리 속도가 빠르고 8051보다 자원이 풍부하여 중소 규모의 산업용 제어기 제작에 적합하다. 또한, 다양한 AVR 제품군이 있어 저가의 소자 선정이 가능하고, 풍부한 개발 환경 및 응용 기술 자료가 많아 본 저온저장고용 제어 및 측정시스템의 마이크로프로세서로 선정하였다.

■ Atmega128의 특성

- 133개의 파워풀한 명령어, 단일 Clock당 한 명령어를 처리할 수 있는 RISC 구조
- 32×8 개의 일반목적 래지스터
- 최대 16MHz에 16MIPS를 수행
- 128K Byte 프로그램 Flash ROM 내장
- 4K Byte 내부 SRAM 내장
- 4K Byte EEPROM 메모리 내장

・In-system progrming(간단한 프로그램 다운로드)

・Analog 비교기, Watchdog Timer, UART, SPI, RTC, PWM내장

・8-Channel, 10bit A/D컨버터 내장

■ A/D 컨버터

온도센서(PT100Ω)와 습도센서(HIH-3610)에서 얻어진 값은 아날로그 값이므로 Micro Processor에 적용 되어질 수 있도록 디지털 값으로 변환하는 장치이다. 본 연구에 사용된 마이크로프로세서의 내부에 내장되어있는 A/D 컨버터를 이용하였다.

■ 표시부

제어시스템의 설정값과 저온저장고의 내부환경 정보들을 나타내는 LCD 이다. 설정값은 온도, 습도, 제상주기, 제상시간, 지연시간 등이 있으며, 이 값들을 입력할 때 LCD에 설정값이 나타난다. 저온저장고의 내부환경 정보인 현재의 고내 온도, 습도를 LCD에 나타낸다.

■ 제어기기 상태 입력부

본 저온저장고의 각 제어기기들에서 발생할 수 있는 과전류나 에러 등을 체크하는 부분이다. 알고리즘에 의해 과전류, 과전압 등이 발생시 다시 초기값으로 설정되도록 처리하였다.

■ 제어기기 출력부

설정한 환경상태로 저온저장고가 작동하도록 각 장치들을 구동시키기 위한 부분이다. 온도제어부에 해당되는 장치는 Compressor, Condensor Fan Motor, 증발기 Fan Motor, Solenoid Valve, 제상 Heater, Drain Heater 등이 있으며, 습도제어부에 해당되는 장치로는 Water Pump, 증발용 Fan Motor 등이 있다.

■ 통신부

저온저장고의 원격 감시 및 제어를 위해 설계한 부분이다. 본 연구에서는 RS485 통신규격을 이용하였다.

3. 제어 시스템 구성

가. 온도·습도 제어 시스템 구성

1) 온도 제어시스템

가) 온도 제어시스템의 개요

그림 6 온도제어시스템

사용자가 마이크로프로세서에 설정 온도를 입력하면, 현재 고내 온도가 설정 온도보다 높은 경우에는 V/V 변환을 통해서 제어기기들을 동작시켜 저온저장고의 내부 온도를 설정 온도와 일치하도록 제어한다. 온도 센서(pt100Ω)를 통하여 저온저장고의 온도를 측정하고, 마이크로프로세서에서 처리할 수 있도록 측정값을 A/D 변환을 통해 마이크로프로세서로 되돌려 준다. 설정 온도와 고내의 현재 온도를 비교하여 고내온도가 목표치에 수렴하도록 제어한다.

■ Micro Processor (ATMega 128)

설정 온도를 현재온도와 비교하여 현재 온도가 설정된 온도 보다 높을 경우에 제어기기를 동작하도록 제어하는 부분으로 16 MIPS로 처리 가능하다.

■ V/V 변환

마이크로프로세서 전원전압은 DC 5[V]이다. 이러한 전압으로는 제어기기들을

구동할 수 없다. 따라서, 제어신호를 구동전압에 맞게 변환시켜야 한다. 본 연구에서 사용되는 제어기기의 구동전압은 AC 220[V]이므로, DC 5[V]를 AC 220[V]로 변환시키기 위해 V/V변환장치를 사용하였다. 본 연구에서는 V/V 변환기로 RY 5W-K를 사용하였다.

■ 제어기기

온도제어를 위한 냉장시스템에 포함된 제어기기로는 Compressor, Condensor Fan Motor, 증발기 Fan Motor, Solenoid Valve, 제상 Heater, Drain Heater 등이 있다.

■ Sensor

저온저장고의 내부 온도를 측정하기 위해 본 연구에서는 온도센서 Pt100Ω을 사용하였다.

■ A/D 변환

센서를 통해 받아들여진 아날로그 값을 디지털 값으로 처리하기 위해 A/D 컨버팅을 시킨다.

나) 공냉식 저온냉장 장치

냉매를 응축시키는 방법에 따라 수냉식과 공냉식으로 구분된다. 수냉식 방법은 물로써 콘덴서의 온도를 낮추는 방법으로 냉각효과가 뛰어난 반면 유체를 사용하기 때문에 배관을 해야 하고 금속물질의 경우 녹이 발생할 수 있으므로 수명이 짧아지는 단점이 있다. 공냉식 방법은 Fan을 이용한 송풍으로 흡열반응을 이용하여 냉각을 시키는 방법으로 수냉식에 비하여 구조가 간단하다.

본 연구에서는 경제성과 응용성 등을 고려하여 공냉식 방법을 사용하였다.

그림 7은 본 연구에서 사용된 저온냉장 장치의 구성도이다. 저온냉장 장치는 다음 과정에 의해 동작된다.

■ Compressor에 의해 냉매를 순환시키며, 순환 냉매는 먼저 Oil Separator를 거친다.

■ Oil Separator에서는 냉매 중에 섞여있는 실린더 오일을 분리해 내고 Condenser를 거치면서 액화된다.

■ 액화된 냉매 중에는 기체상태의 냉매가 혼합되어 있으므로, 다시 Receiver에서 액화 냉매만을 추출한다.

■ Expansion Valve와 Evaporator를 거치면서 팽창 및 기화됨에 따른 흡열반응에 의해 냉각된 주위 공기를 팬을 이용하여 고내로 분산시킨다.

■ 기체상태의 냉매 중에는 액체 냉매가 혼합되어 있으므로 Accumulator에서 액체 냉매는 분리되고 기체 냉매만이 Compressor에 흡입되어 진다.

■ Solenoid Valve는 액체 냉매의 통전을 개폐하는 역할을 한다. Solenoid Valve가 닫히면 관내 압력이 낮아지므로 Compressor의 저압 스위치가 열리면서 Compressor가 멈춘다(Pump Down). 따라서 고내 온도제어를 위해 Solenoid Valve를 제어 수단으로 사용한다.

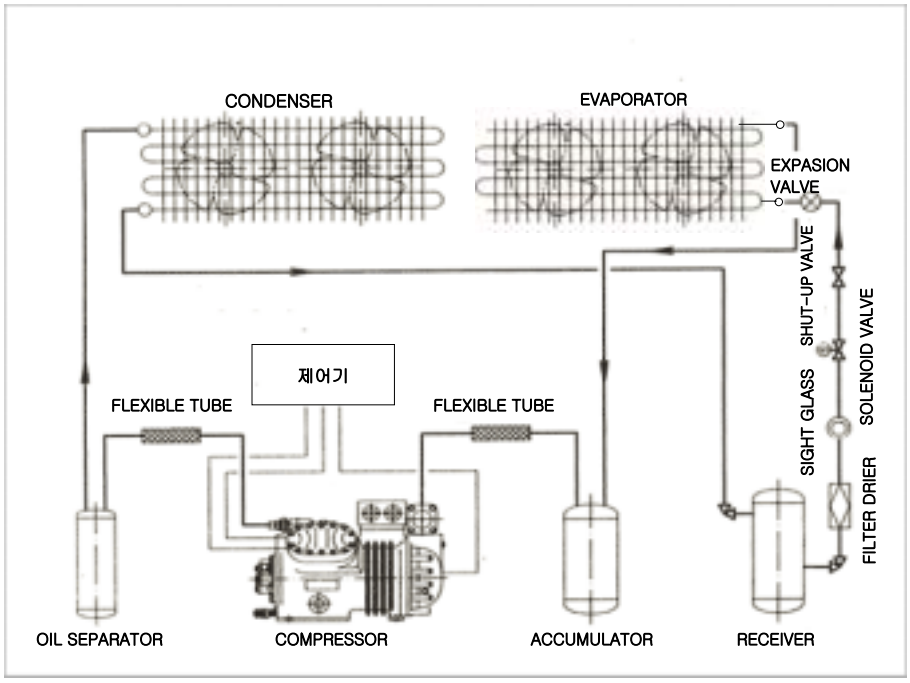

그림 7 저온 냉장장치

다) 저온 냉장장치의 구성요소

■ COMPRESSOR(압축기)

증발기의 낮은 온도에서 증발한 기체를 흡입하여 응축기에서 응축 온도로 액화되는 정도의 포화압력까지 압축해 주는 기기, 실린더를 포함하며 실린더 내부에는 윤활유인 Oil이 들어있다.

■ FLEXIBLE TUBE

냉매가스관의 진동 및 힘 전달을 차단하여 배관의 파손을 방지하기 위해 배관 중간에 완충용 유연관을 설치한다.

■ OIL SEPARATOR(유분리기)

Compressor에서 포화압력까지 압축하는 과정에서 냉매(R22-후레온)와 Compressor 실린더의 오일이 혼합되어 관에 유입된다. Condenser에 혼합액이 유입되기 전에 Oil을 제거하기 위해 유 분리기를 거친다.

■ CONDENSER(응축기)

압축기로 압축된 고온고압인 기체의 열을 방출하여 액화시키는 기기를 말한다. 물이나 공기로 냉각시켜 준다. 실외에 설치하는 실외기이다.

■ RECEIVER(수액기)

응축기에 의해 액화된 냉매에는 기체상태인 가스가 포함되어 있는데 이중 액체냉매만을 추출하는 통. 비중 차에 의해 액체는 하부에 기체는 상부에 위치하므로 하부에서 추출하면 액체 냉매만을 추출할 수 있다.

■ FILTER DRIER

실리카겔을 통과시켜 수분 및 이물질을 제거한다.

■ SIGHT GLASS

액체 냉매와 수분함유 상태를 육안으로 확인할 수 있도록 투명 유리를 관에 부착한다.

■ SOLENOID VALVE(전자변)

액체 냉매의 통전 및 차단장치로서 고내온도가 T/C에서 설정한 온도까지 내려가면 전기신호가 S/V를 기동한다. 이때 관은 차단되고 관의 압력이 떨어지면, DPS의 LP나 LPS에 의해 압축기를 비롯한 모든 장치가 멈춘다.

(T/C: Temperature Controller, DPS: Dual Pressure Switch, LP: Low Pressure, LPS: Low Pressure Switch)

■ SHUT-OFF VALVE

수동밸브로서 장치의 수리 시 냉매 유출을 방지하기 위해 사용한다.

■ EXPANSION VALVE(팽창변)

응축기에서 액화한 고압의 액체를 죄임작용(교축작용)에 의하여 증발을 일으킬 수 있는 압력까지 감압해 주는 밸브이다.

■ EVAPORATOR(증발기)

팽창밸브에 의해 감압된 저 온도상태의 액체냉매가 증발작용을 일으켜 주위의 열을 흡수하는 기기이다. 실내에 설치하는 실내기이다.

■ 감온구

온도를 감지하는 구이며 구내에는 가스로 채워져 있고, 팽창변과 모세관으로 연결되어 있다. 증발이 끝나는 부분의 동관 상부에 45도로 부착한다. 냉매가스 온도를 감지하여 팽창변의 오림피스 구경을 자동 조절한다. 냉매의 온도에 따라 감온구 내부 가스의 팽창정도가 달라짐을 이용한다.

■ ACCUMULATOR(액분리기)

기체 냉매에 포함되어 있는 액체를 분리시키는 기기이다. 증발기에서 액체 냉매가 100[%] 증발하지 못하기 때문에(80%~90% 증발) 증발 후에도 10%~20% 정도의 액체냉매가 기체 냉매에 혼합되어 있다. 따라서 냉매를 고압으로 압축하기 전에 액체냉매를 분리해야 한다. 액체냉매가 내포된 상태에서 압축을 하면 액체가 고압과 작용하여 액 햄머 현상에 의하여 Compressor 부품을 파손하게 된다.

2) 습도 제어시스템

가) 습도 제어시스템의 개요

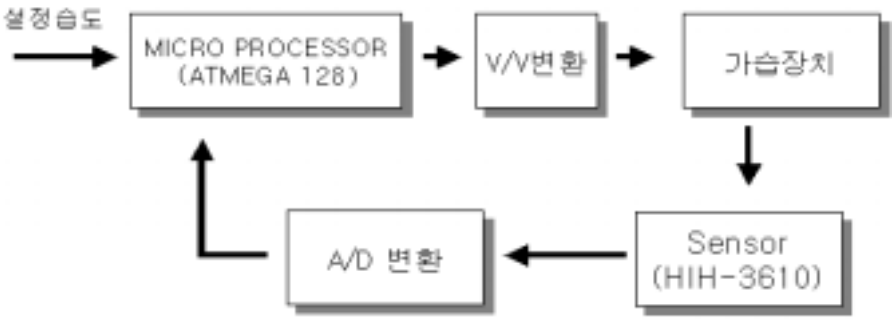

그림 8 습도제어시스템

저온저장고의 내부 습도가 설정한 습도보다 낮을 경우에는 V/V변환을 통해서 가습장치 즉, 워터 펌프와 습기 발생용 Fan Motor를 가동시켜 저온저장고의 내부 습도를 증가시켜 준다.

습도 센서(HIH-3610)를 통하여 저온저장고의 습도를 측정하고, 마이크로프로세서에서 처리할 수 있도록 측정값을 A/D변환 하여 마이크로프로세서로 되돌려 준다. 설정 습도와 현재 습도를 비교하여 목표치에 일치 할 때까지 반복된다.

■ Micro Processor ATmega128

온도 제어 시스템 부분과 동일.

■ V/V 변환

온도 제어 시스템 부분과 동일.

■ 가습장치

습도를 증가시키기 위하여 저온저장고 내부에 수분을 공급하는 장치이다. 가습장치에는 Water Pump와 수분 발생용 Fan Motor 등과 같은 가습기기가 있다.

■ Sensor(HIH-3610)

저온저장고의 내부 습도를 측정하기 위해 본 연구에서는 분해능과 응답속도가 빠른 습도센서 HIH-3610을 사용하였다.

■ A/D 변환

온도 제어시스템 부분과 동일.

나) 가습 장치

그림 9는 가습장치에 대한 계통도 이다. 이것은 수분과 바람을 이용하여 습기를 발생시키는 자연 가습방법이다. 가습장치는 크게 가습용수를 순환시켜주는 순환부와 가습용수를 외부로 불어내주는 통풍부로 나뉜다. 자연 가습의 원리는 Water Pump에서 관을 통하여 유입된 물은 Fan 앞에 설치된 포습재를 통과하게 된다. 이 과정에서 포습재에 포습된 가습 용수는 팬의 회전에 의해 미세한 수분으로 증발되고 미세한 수분들이 저장고 내의 습도를 높이게 된다. 가습의 정도를 결정하는 요소는 팬의 회전속도와 가습용수의 양을 들 수 있다. Fan의 동작을 제어하는 단말장치로서 전자접촉기를 사용하였으며, 가습용수 공급 양을 조절하기 위한 제어장치로서 솔레노이드 밸브를 사용하였다.

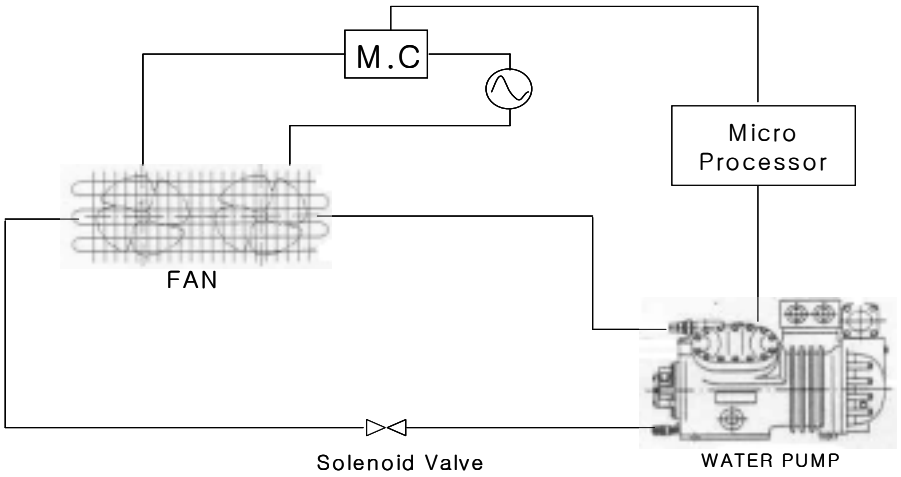

그림 9 가습 장치 계통도

나. 제어시스템과 컨트롤러간 인터페이스 구축

1) V/V 변환

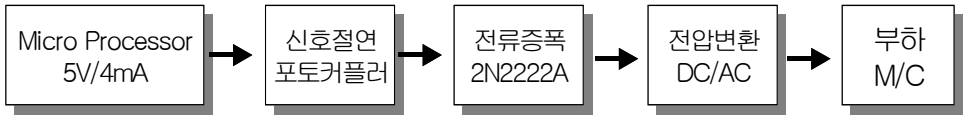

그림 10 V/V 변환

마이크로프로세서에서 각 I/O 포트당 출력되는 전압과 전류는 각각 5[V]와 4[mA] 이다. 5[V]의 입력전압을 받아 제어기기를 동작시킬 수 있는 220[V]의 출력 전압을 얻기 위해 그림 10과 같은 V/V변환을 한다.

마이크로프로세서와 냉장시스템의 절연을 위해 포토 커플러를 사용하였고, 트랜지스터 2N2222을 사용하여 전류를 증폭하고, DC/AC 변환 릴레이 RY 5WK를 사용하여 입력신호가 DC 0[V] 일때 릴레이 내부회로는 열린 상태가 되고 입력신호가 DC 5[V]일때 릴레이 내부회로는 닫힌 상태가 되어 AC 220[V]로 스위칭된다. 그러나 릴레이의 출력은 220[V]/1[A]이므로 제어기기를 작동하기 위한 전류가 충분치 않다. 따라서 전자접촉기(MC)를 이용하여 냉장장치를 구동하도록 설계하였다.

■ Micro Processor ATmega128

온도 제어 시스템 부분과 동일하며, 각 I/O 포트당 해당 제어기기를 각각 제어하도록 설계하였다.

■ 포토 커플러

마이크로프로세서부와 제어기기를 광을 통해 절연시킴으로써 시스템의 안정성과 전기 절연성을 향상시킬 수 있다.

■ 2N2222

이 소자는 선형증폭기와 스위칭 회로로 사용되는 트랜지스터이다. 트랜지스터 2N2222는 마이크로프로세서와 I/O포트 전류가 적어 릴레이(RY 5WK)를 직접 구동할 수 없기 때문에 마이크로프로세서의 I/O 포트 전류를 증폭시키는 역할을 한다. 또한 마이크로프로세서의 제어신호(0V 또는 5V)로 2N2222 입력단을 제어함으로써 전류를 개방 또는 차단할 수 있고, 이를 통해서 2N2222의 컬랙터 단에 접속된 전압변환부의 릴레이를 제어한다.

■ 전압변환

DC 5[V] 전압을 AC 220[V]로 출력을 변환하기 위해 RY 5WK 릴레이를 이용하여 상용전원으로 변환시키는 부분이다.

■ 전자접촉기(Magnetic Contact)

본 연구에서는 전자접촉기를 사용하여 제어기기인 Compressor와 Water Pump 등을 구동하였다. 사용된 전자접촉기는 220[V]/30[A]의 용량을 가진 접촉기를 사용하여 냉장장치에 사용된 대용량 모터를 제어하도록 설계하였다.

2) 제어회로 구성

약전의 입력을 받아 강전을 제어하는 방법으로는 유접점 제어와 무접점 제어가 있다. 유접점 제어는 릴레이 스위치 제어에 이용되고, 무접점 제어는 다이오드, 트랜지스터, 디지털 IC 제어에 사용된다. 이와 같은 제어 방법의 특징은 기계적인 가동부가 없기 때문에 수명이 길고 신뢰도가 높으며 진동에도 강하다.

본 연구에서는 마이크로프로세서의 I/O 출력을 이용하여 220[V]/30[A] 출력의 정격을 지닌 기기들을 직접 구동시키는 무접점/유접점 혼합 방식을 채택하였다.

3) 회로구성

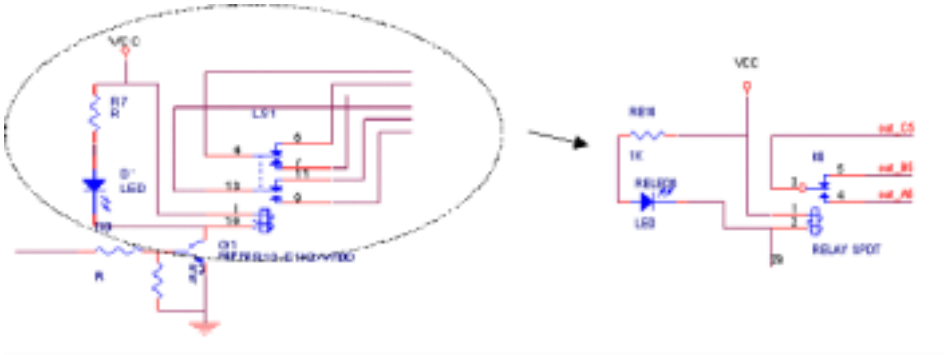

그림 11 릴레이 부분 회로도

NPN Switching Transistor인 2N2222A를 사용하여 Relay를 제어하도록 전류를 증폭하였다. 2N2222A의 컬렉터단의 최대증폭전류는 800[mA]이며 컬렉터 에미터 개방전압은 40[V]이다. 선형증폭기나 스위칭회로에 쓰인다. 트랜지스터에 부하선상의 포화 또는 차단영역에서 동작하는 특성을 이용하여 스위치로 응용하였다. 트랜지스터가 포화되었을 때는 트랜지스터의 콜렉터와 에미터 사이에 닫혀진 스위치와 같고, 트랜지스터가 차단되었을 때는 개방된 스위치와 같다.

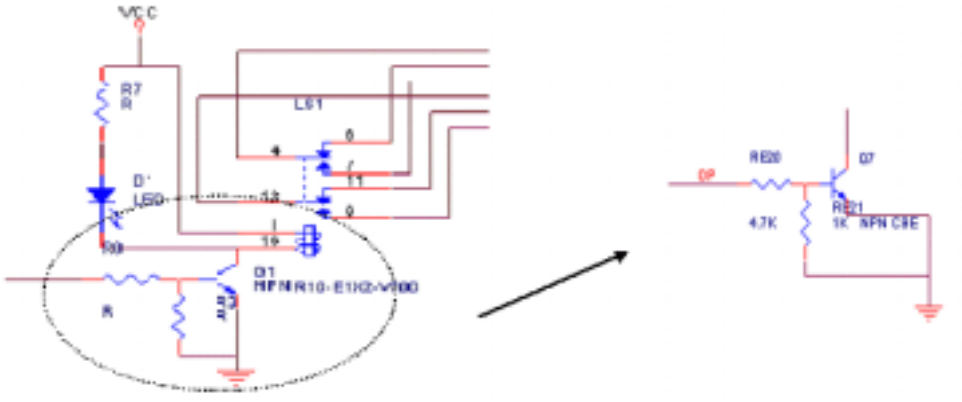

그림 12 트랜지스터의 스위칭 회로도

다. 제어 알고리즘 구현

1) 제어시스템의 구성

본 연구를 위해 제어 알고리즘을 구현하는 측면에서 마이크로프로세서와 관련된 요소들을 구체적으로 표현한 온·습도 제어시스템의 구성은 다음과 같다.

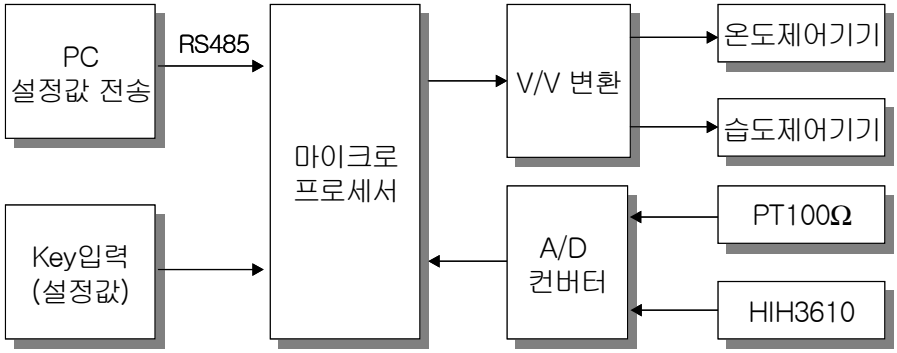

그림 13 온·습도 제어시스템의 구성

그림 13에 나타낸 바와 같이 제어시스템은 제어값 설정부, 온도와 습도센서로부터 저온저장고의 상태를 입력받는 입력부, 온·습도 제어기기를 제어하는 제어부 그리고 이들을 통합 관리하는 마이크로프로세서로 구성되어 있다. 설정 값은 원격의 PC로부터 RS485 통신에 의해 설정 및 변경될 수 있으며, 자체의 keypad에 의해서도 설정이 가능하다. 온도 센서인 Pt100Ω과 습도센서인 HIH-3610으로부터 입력되는 전압 값들은 A/D 변환을 통해 마이크로프로세서로 전달된다. 컨트롤러는 사용자가 설정한 값들과 현재 값들을 비교하여 그에 대응하는 동작을 하도록 제어한다.

■ 마이크로프로세서

온도 제어시스템 부분과 동일

■ V/V변환

온도 제어시스템 부분과 동일

■ A/D 컨버터

온도 제어시스템 부분과 동일

■ 온도제어 기기

온도 제어시스템 부분과 동일

■ 습도제어 기기

습도 제어시스템 부분과 동일

■ 온도센서(Pt100Ω)

온도 제어시스템 부분과 동일

■ 습도센서(HIH-3610)

습도 제어시스템 부분과 동일

■ RS485

RS485 통신규격은 원거리(4000피트) 통신이 가능하고, 1: n의 통신 방법을 사용할 수 있기 때문에 컴퓨터 한 대로 다수의 저장고를 동시에 제어하여야 하는 본 연구에 가장 적합한 통신방법이다.

2) 온도제어 알고리즘

그림 14는 본 연구에서 사용된 제어시스템의 온도제어 알고리즘 이다. 알고리즘에서 사용된 용어의 표현은 다음과 같다.

INT Power : Compressor 내부전원

T/C : Temperature Sensor Controller

PL : Power Lamp

DH : Drain Heater

AL : Alarm Lamp

SV : Solenoid Valve

HM : Heater Magnetic Contact for Defrost

CCH : Crank Case Heater

CHL : Heater Lamp

CFM : Condenser Fan Magnetic Contact

EV FAN : Evaporator Fan

OPS : Oil Pressure Switch

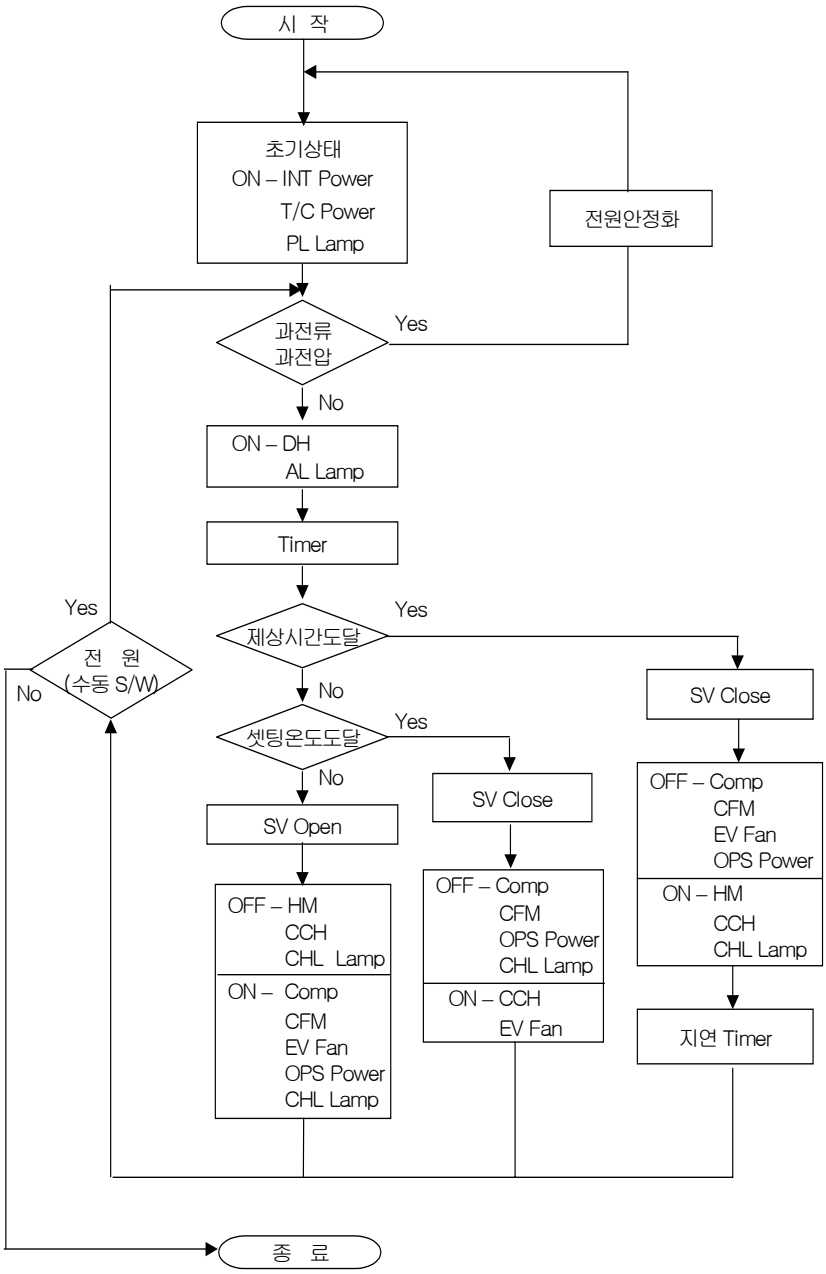

그림 14 온도 제어 알고리즘

■ 온도제어 방법은 초기상태에서 압축기와 온도센서를 구동하기 위하여 압축기의 내부전원(INT Power)과 온도센서 제어기의 전원(T/C power) 스위치를 ON 시키고 사용자로 하여금 전원이 공급되는 것을 알 수 있도록 전원램

프를 켠다.

■ 과전류/과전압 제어기는 외부 전원이나 온도 제어기기로부터 공급되는 신호나 전원에 과전류/과전압이 있는가를 판단한다.

■ 과전류/과전압이 검출되면 그 과전류/과전압을 제거하여 외부 전원이나 온도 제어기기로부터 공급되는 신호나 전원을 안정화시키고 초기단계를 재 수행 한다.

■ 과전류/과전압이 없으면 드레인히터 'DH'와 경보램프(Alarm Lamp)를 구동시킨다.

■ 마이크로프로세서는 타이머를 동작시키고, 타이머의 카운트값을 감시하여 온도 제어기기의 동작시간을 감시하여 그 동작시간이 제상시간에 도달하는가를 판단한다. 그리고 그 동작시간이 아직 제상시간에 도달되지 않은 것으로 판단되면 마이크로프로세서는 온도센서에 의해 감지되는 저장고 내의 현재 온도가 미리 설정된 최적 온도에 도달되는가를 판정한다. 현재 온도가 최적 온도에 도달되지 않으면 솔레노이드밸브 'SV'가 열린다. 그리고 제상을 위한 전자접촉기 'HM'과 크랭크 케이스 히터(Crank Case Heater : CCH)가 꺼지게 된다. 반면에, 압축기 'Comp', 응축기 'CFM', 증발기 'EV Fan'이 구동되고 압축기의 오일 압력 스위치(Oil Pressure Switch : OPS)가 ON 되면서 압축기의 실린더에 오일이 공급되어 저장고 내의 온도가 낮아지게 된다.

■ 제상시간에 도달하지 않은 상황에서 현재 온도가 최적 온도에 도달되면 솔레노이드밸브 'SV'가 닫힌다. 그리고 압축기 'Comp', 응축기 'CFM', 오일 압력 스위치 'OPS'가 구동을 멈추는 반면에 크랭크 케이스 히터 'CCH'와 증발기 'EV Fan'이 구동된다.

■ 제상시간에 도달하면 솔레노이드밸브가 닫히고, 압축기 'Comp', 응축기 'CFM', 증발기 'EV Fan', 오일 압력 스위치 'OPS'의 구동이 멈추는 반면에, 제상을 위한 전자접촉기 'HM'과 크랭크 케이스히터 'CCH'가 구동되어 제상을 하고 제상이 진행 중임을 나타내기 위한 히터램프가 켜지게 된다. 그리고 온도 제어기기는 마이크로프로세서의 제어 하에 미리 설정된 시간 동안 제상된 후에 소정의 지연시간 동안 대기하고 그 지연시간이 경과한 후에 재가동된다.

■ 사용자가 수동 스위치로 전원을 끄지 않으면 과전압과 과전류를 체크하는

단계로 되돌아간다.

3) 습도제어 알고리즘

그림 15는 저온저장고의 습도제어 방법의 제어 수순을 단계적으로 나타내는 습도제어 알고리즘이다.

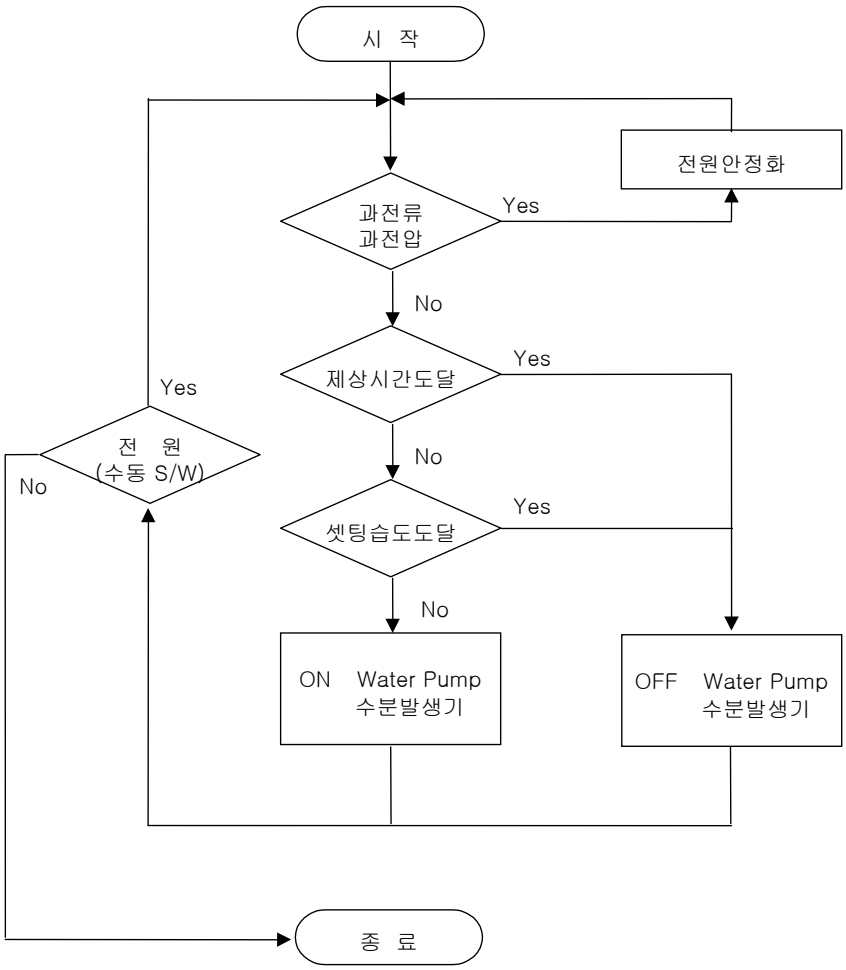

그림 15 습도제어 알고리즘

■ 습도제어 방법에 있어서 과전류/과전압 제어기는 외부 전원이나 가습장치로부터 공급되는 신호나 전원에 과전류/과전압이 있는가를 판단한다.

■ 과전류/과전압이 검출되면 그 과전류/과전압을 제거하여 외부 전원이나 가

습장치로부터 공급되는 신호나 전원을 안정화시키고 시작 단계를 재 수행한다.

■ 과전류/과전압이 없으면 마이크로프로세서는 동작 경과시간이 제상시간에 도달하는가를 판단한다.

■ 동작시간이 아직 제상시간에 도달되지 않은 것으로 판단되면 마이크로프로세서는 습도센서에 의해 감지되는 저장고 내의 현재 습도가 미리 설정된 최적 습도에 도달되는가를 판정한다. 현재 습도가 최적 습도에 도달되지 않으면 마이크로프로세서는 물펌프와 수분발생기를 구동하여 저장고 내에 미세한 수분을 공급함으로써 저장고의 습도를 높인다.

■ 제상시간에 도달하거나 저장고의 현재 습도가 미리 설정된 최적 습도에 도달한 것으로 판단되면 마이크로프로세서는 물펌프와 수분발생기를 정지시킨다.

■ 사용자가 수동 스위치로 전원을 끄지 않으면 시작 단계를 재 수행한다.

4) 제어 프로그램 개발

```
if((set_temp+3 < act_temp) && (set_humi+1 > act_humi))
                        ppi1b_data = 0xc3;
        else if((set_temp+3 < act_temp) && (set_humi-1 < act_humi))

                        ppi1b_data = 0xc1;
    else if((set_temp-3 > act_temp) && (set_humi+1 > act_humi))
                        ppi1b_data = 0x6b;
    else if((set_temp-3 > act_temp) && (set_humi-1 < act_humi))
                        ppi1b_data = 0x69;
```

위의 프로그램은 온도와 습도 값에 따른 제어 부분이다.

■ 첫 번째의 경우 셋팅 온도가 작동 온도 보다 낮으면 온도를 낮추기 위해 온도 제어기기들을 작동시키고 셋팅 습도가 작동 습도 보다 높으면 습도 제어기기들이 작동하도록 제어한 부분이다.

■ 두 번째의 경우는 온도는 높고 습도는 적당한 경우이고

■ 세 번째의 경우는 온도는 충분히 낮고 셋팅 습도에 현 습도가 도달하지

못했을 경우이고

■ 네 번째 경우는 온도도 충분히 낮고 습도도 충분히 높은 경우이다.

셋팅 온도의 뒤에 있는 +, -는 데드존을 설정하기 위함이다.

5) 동작 여부 결정 방법

동작여부를 결정할 수 있는 경계치 값을 1개 사용하였을 경우 잦은 온도 및 습도의 변화에 의해 엑츄에이터의 동작이 채터링(chattering)될 수 있다. 이를 방지하기 위해서 본 연구에서는 Dead Zone을 부여 하였으며, 경계치 값을 상한 경계치와 하한 경계치 두 부분으로 나누어 사용하였다. 상한 경계치를 통과하였을 경우 엑츄에이터는 동작을 시작하고, 하한 경계치를 통과했을 경우 엑츄에이터가 멈추도록 하였다. 이를 통해 작업환경 내에서의 노이즈나 외란의 영향에서도 일정한 값 간격 내에서는 정상적으로 작동한다. 본 시스템에서는 ±1℃ 온도 입력값을 Dead zone으로 설정하였다.

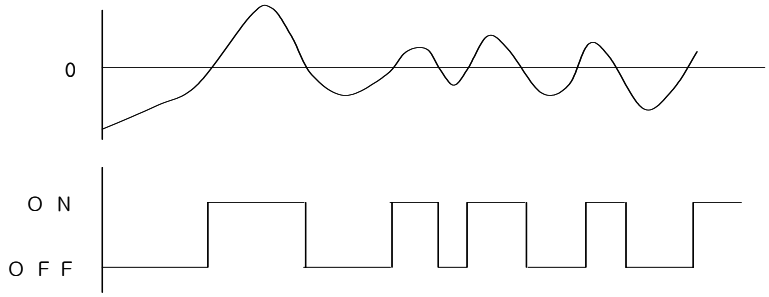

그림 16 일반적인 ON/OFF

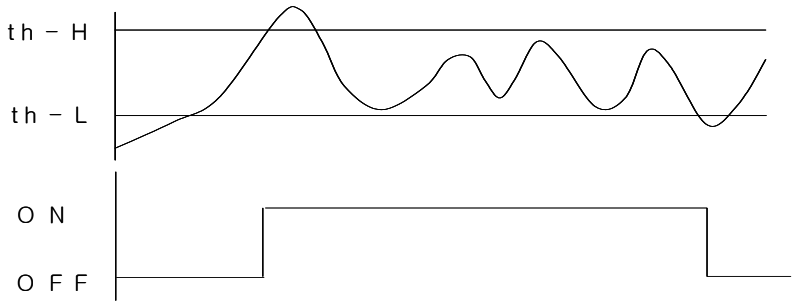

그림 17 Dead Zone

4. 측정 시스템 구성

가. 온도·습도 측정시스템 구성

1) 온도측정시스템 선정

가) 백금저항온도센서(pt100Ω)의 특성

일반적으로 공업용 백금 온도 센서는 일반 계측 분야에 널리 사용되는 규격화된 센서로서 0℃, 100Ω을 기준으로 하고 있다. 규정전류는 2[mA]를 사용한다. 측온 저항체는 저항 소자 양단에 각 1개의 도선을 연결한 2선식, 도선 저항 분을 제거하기 위하여 한쪽에 2개의 도선을 연결한 3선식과 양단에 각 2개씩의 도선을 연결한 4선식으로 분류되며, 도선 수가 많을수록 고 정확도용으로 이용된다. 보통 현장에서는 3선식이 주류를 이루고 있으나 최근 제어장비의 발달에 따른 정확도 향상으로 4선식의 사용이 늘어나고 있다. 만약 현장형 온도 전송기를 선택할 경우는 향후를 대비하여 4선식까지 사용이 가능한 제품을 선정할 필요가 있다. 백금의 100[Ω]에서 공칭 오차한계는 +/- 0.06[Ω] 정도 이다.

본 연구에서 저온저장고의 온도 계측을 위한 센서로서 백금저항 온도센서인 pt100Ω을 사용하였다.

표 6 금속의 저항률과 저항 온도계수

금속	저항율 uΩcm,20℃	저항 온도 계수 $\alpha_0/10^{-3}$
Au	2.4	4.0
Ag	1.62	4.1
Cu	1.72	4.3
Pt	**10.6**	**3.9**
Ni	7.24	6.7
Fe	9.8	6.6
Ta	15	3.5
W	5.5	5.5

금속 저항값의 온도 의존성을 이용해 니켈, 구리, 백금 등의 금속과 합금을

재료로 하는 많은 센서들이 개발되어 왔다. 그러나 재료의 안정성이 우수하고 온도-저항값 특성이 좋은 백금을 주로 사용한다. 온도에 따라 백금의 저항치가 변하는 원리를 이용한 것으로 현존하는 온도센서 중 가장 정확도가 높아 -260℃~630℃ 영역에서는 표준온도센서로 사용된다. 국제 실용눈금에서는 백금의 온도영역을 - 259.34℃(수소3중점)~630.74℃(안티몬 응고점)로 설정 사용하고 있다.

나) 백금저항 온도센서(pt100Ω)의 규격

표 7 pt100Ω의 규격표

TEMP.[℃]	Ohm[Ω]	TEMP.[℃]	Ohm[Ω]
0	100.00	52	120.51
1	100.40	53	120.91
2	100.80	54	121.30
3	101.19	55	121.69
4	101.58	56	122.08
5	101.99	57	122.47
6	102.38	58	122.86
7	102.78	59	123.25
8	103.18	60	123.64
9	103.57	61	124.03
10	103.97	62	124.42
11	104.37	63	124.81
12	104.76	64	125.20
13	105.16	65	125.59
14	105.56	66	125.98
15	105.95	67	128.37
16	106.35	68	126.76
17	106.74	69	127.15
18	107.14	70	127.54
19	107.53	71	127.93
20	107.93	72	128.32
21	108.32	73	128.75
22	108.72	74	129.09
23	109.11	75	129.48
24	109.51	76	128.37
25	109.90	77	130.26
26	110.30	78	130.65
27	110.69	79	131.04
28	111.09	80	131.42

TEMP.[℃]	Ohm[Ω]	TEMP.[℃]	Ohm[Ω]
29	111.48	81	131.81
30	111.88	82	132.20
31	112.27	83	132.59
32	112.68	84	132.98
33	113.06	85	133.36
34	113.45	86	133.75
35	113.84	87	134.14
36	114.24	88	134.52
37	114.63	89	134.91
38	115.02	90	135.30
39	115.42	91	135.68
40	115.61	92	136.07
41	116.20	93	136.48
42	116.59	94	136.84
43	116.99	95	137.23
44	117.38	96	137.62
45	117.77	97	138.00
46	118.16	98	138.39
47	118.56	99	138.77
48	118.96	100	139.16
49	119.34	101	139.55
50	119.73	102	139.93

다) 온도측정시스템의 구성

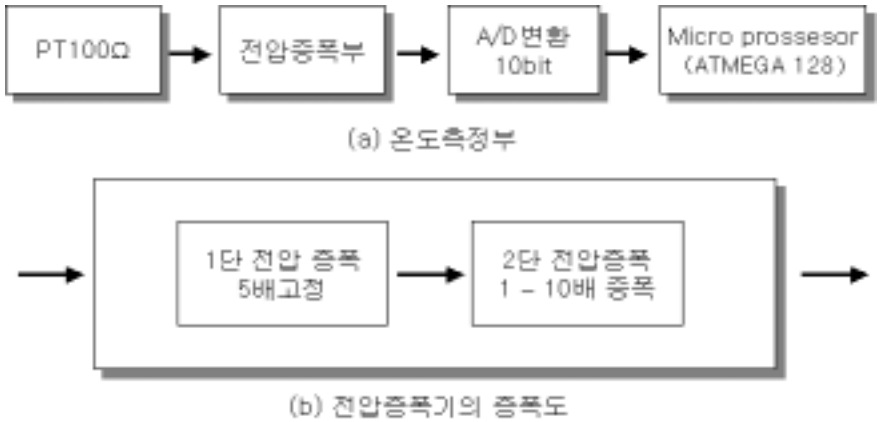

그림 18 온도측정시스템

온도센서 Pt100Ω으로부터 얻어진 온도 값은 아주 미세하기 때문에 마이크로프로세서에서 처리할 수 없다. OP AMP를 사용한 비반전 증폭기를 사용하여 최대 50배의 전압을 증폭하고 그 값을 A/D 변환하여 마이크로프로세서가 값을 받아들일 수 있도록 처리하였다.

■ Pt100Ω

온도 센서로서 저항의 변화로 주변 온도를 감지한다.

■ 전압증폭부

입력받은 저항은 미세한 값을 보이므로, OP AMP를 통하여 증폭 후 측정한다. 본 연구에서는 영점은 잡고 증폭률은 고정하여 사용하였다.

■ A/D 변환부

온도 제어시스템 부분과 동일

2) 습도측정 시스템

가) 습도측정 센서의 선정

■ HIH-3610 센서의 개요

습도를 측정하기 위하여 신뢰성과 응답성이 우수한 HIH-3610 Series 습도센서를 사용하였다.

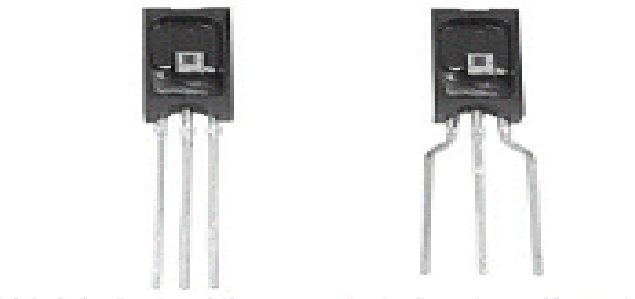

그림 19 HIH-3610 Series Humidity Sensor의 외형

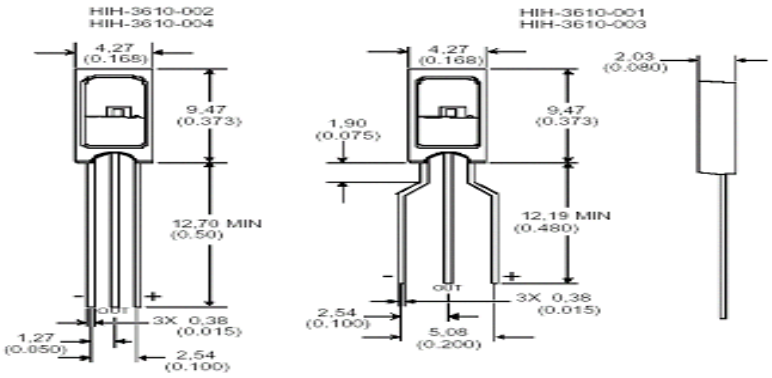

그림 20 HIH-3610 Series Humidity Sensor의 내부

■ HIH-3610 센서의 특징

HIH-3610 Series Humidity Sensor는 가공된 열경화성 수지의 커버로 덮여있고, 출력전압과 상대습도는 선형을 이루는 특징을 가진다. HIH-3610 센서의 장점은 저 전력에서도 동작하고, 높은 정밀도를 가지며, 반응 속도가 빠르고, 안정되고 신뢰할 만한 동작을 이룬다.

■ HIH-3610 Series Humidity Sensor가 대표적으로 사용된 기기

냉장고, 드라이기, 도량기, Battery Power System

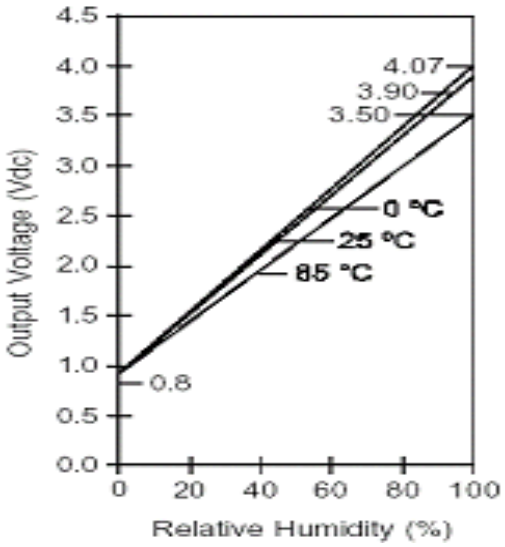

그림 21 상대습도와 출력전압 특성

그림 21은 습도센서의 상대습도와 출력전압간의 특성곡선이다. 그림에서 볼 수 있는 바와 같이 습도센서는 온도의 변화에 비교적 강인함을 알 수 있으며, 또한 출력전압과 습도 간에 선형적 변화가 이루어짐을 알 수 있다.

나) 습도측정시스템의 구성

그림 22 습도측정 시스템

HIH-3610은 Pt100Ω과 같은 미세한 저항 차에 의에 값들이 리턴되는 구조를 갖는 것이 아니고, HIH-3610에서 5[V]의 전압을 인가하여 주면 상대습도(RH)에 맞는 전압 값이 출력되는 센서이다. 따라서 증폭부 없이 센서의 출력단자를 A/D 변환기의 입력단자에 직접 연결하여 처리하도록 구현하였다.

■ HIH-3610

습도센서로서 높은 정밀도와 신뢰성을 보인다.

■ A/D 변환

습도센서(HIH-3610)에서 감지한 아날로그 신호값을 마이크로프로세서에서 처리할 수 있도록 디지털 값으로 변환하는 장치이다. 마이크로프로세서에 내장된 A/D 컨버터를 이용하여 10bit 분해능의 디지털 값으로 변환하여 사용하였다.

■ Micro Processor ATmega128

온도제어 시스템 부분과 동일.

나. 측정시스템과 컨트롤러간 I/F 구축

1) 온도센서 인터페이스

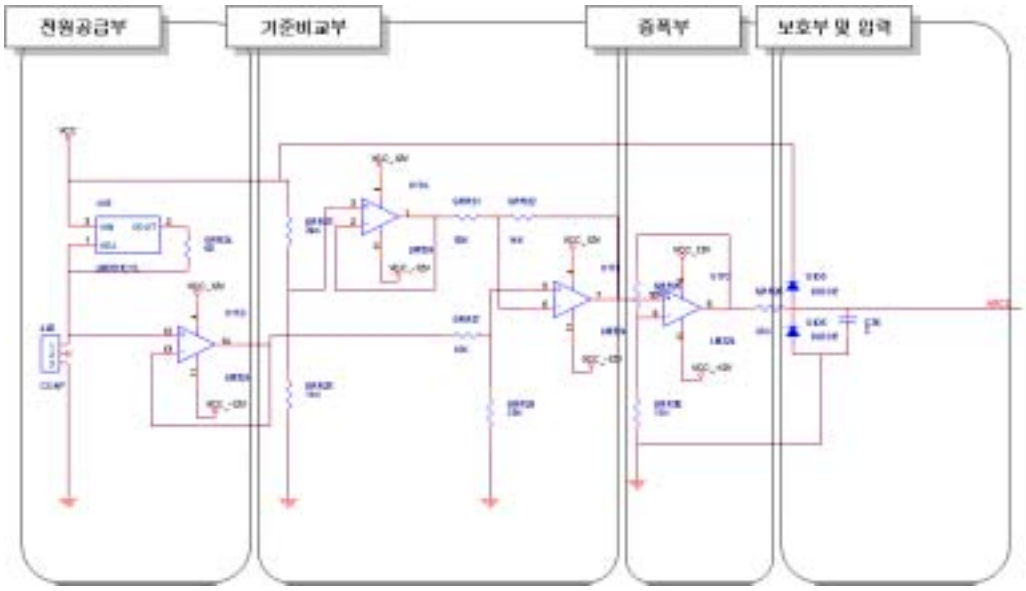

그림 23 온도 입력처리 회로

그림 23은 온도센서 Pt100을 이용하여 온도 정보를 입력받는 회로이다. 회로의 구성은 전원공급부, 기준비교부, 증폭부, 보호부로 구성되어 있다.

전원공급부는 온도센서(Pt100)에 정전류를 공급한다. 이 회로를 통하여 안정된 전원을 공급할 수 있다. 정전류 공급을 위해 본 연구에서는 LM317을 사용하였으며, 정전류 회로용 저항으로 680[Ω]을 사용하여 850[mA]의 정전류가 Pt100에 공급되도록 설계하였다.

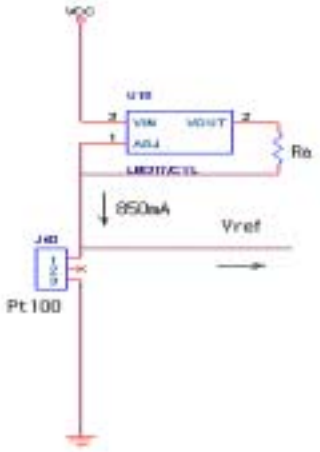

그림 24 정전류 회로

그림 25는 기준 비교부 회로도이다. 기준 비교부를 통하여 측정하고자 하는 온도 영역을 결정한다. 저항 R3과 R8에 의해 측정하고자 하는 전압범위가 결정된다. 본 연구에서는 -10[℃]~20[℃] 영역을 측정할 수 있도록 저항 R3과 R8을 각각 36[kΩ], 0.82[kΩ]로 채택하였다.

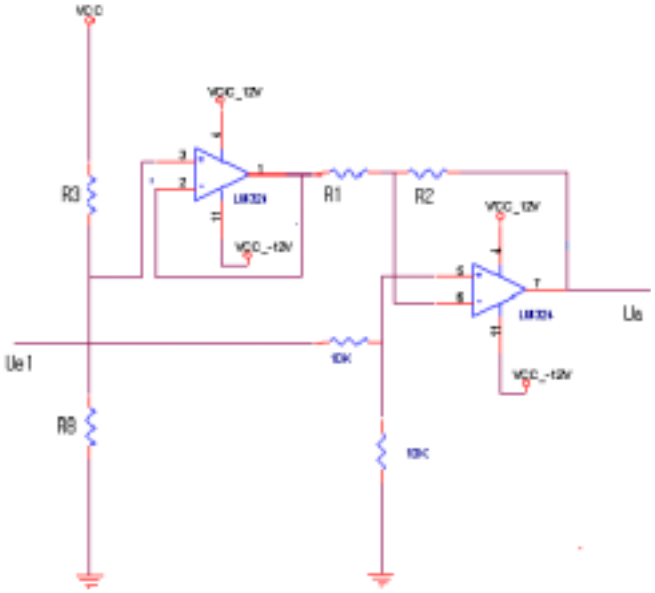

그림 25 기준 비교부 회로도

다음은 기준전압을 산출하는 식이다.

$$\frac{Ue2}{Ub} = \frac{R8}{R3 + R8}$$

$$Ue2 = Ub * \frac{R8}{R3 + R8}$$

$$Ue2 = 5[V] * \frac{0.82[K\Omega]}{36[K\Omega] + 0.82[K\Omega]}$$

$$Ue2 = 0.11[V]$$

산출 식에 의해 얻어진 기준전압(Ue2)은 0.11[v]이다. 이를 적용하면 전압 변동값은 약 164mV에서 191mV까지 변화한다.

175[mV] - (11[mV]) = 164[mV]
202[mV] - (11[mV]) = 191[mV]

전압 변동값을 저항 R1과 R2를 통하여 3배 증폭시키면 492[mV]~573[mV]이 출력전압 Ua가 된다.

다음 과정인 그림 26에서는 얻어진 전압신호(Ua)가 1V에도 못 미치는 낮은 전압이므로 이 전압을 높은 분해능을 가질 수 있도록 저항 R5과 R10을 이용하여 비반전 증폭한다. 또 그림 26은 다이오드를 이용하여 과전류로 인한 MCU의 고장을 방지하고, 낮은 전압으로 인하여 역전류가 흐르는 것을 방지하는 보호회로를 포함하고 있다. 높은 전압일 경우는 다이오드 D1을 통하여 Vcc로 전류가 흐르고 낮은 전압일 경우에는 다이오드 D2를 통하여 GND에서 전류가 흐르도록 함으로써 회로를 보호할 수 있다.

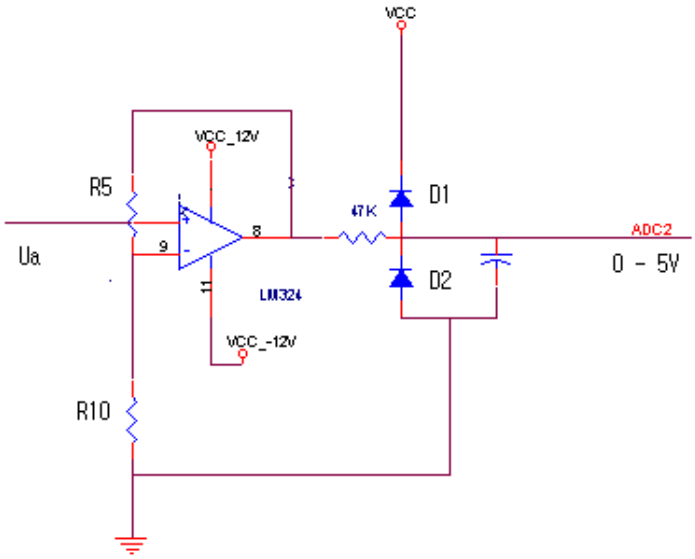

그림 26 비반전 증폭 및 보호회로

다음은 저항 R5를 산출하는 식이다.

$$A = 1 + \frac{R5}{R10}, \quad A = \frac{Ua}{Ue}$$

$$\frac{R5}{R10} = \frac{Ua}{Ue} - 1$$

$$R5 = 820[\Omega] * \left(\frac{5[V]}{573[mV]} - 1 \right)$$

$$R5 \fallingdotseq 6.3k\Omega$$

AD 변환부는 본 연구에서 사용한 마이크로프로세서에 내장된 AD변환기를 사용하였다. 입력전압 범위는 0[V]~5[V]이며, 온도센서 Pt100의 온도특성은 저항에 비례한다. 본 연구에서는 측정온도의 범위를 -10[℃]~20[℃]로 설정하였으며, 1[%] 정밀저항을 사용하여 노이즈에 대한 강인성을 높였다. AD값에 대한 온도특성은 그림 27과 같다.

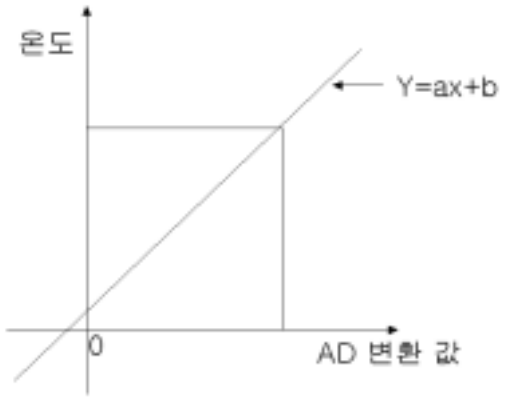

그림 27 AD 변환값에 대한 온도특성

이와 같은 사양에 부합하도록 계산된 수식은 다음과 같다.

y = (0.275 * x) - 84.71

2) 습도센서 인터페이스

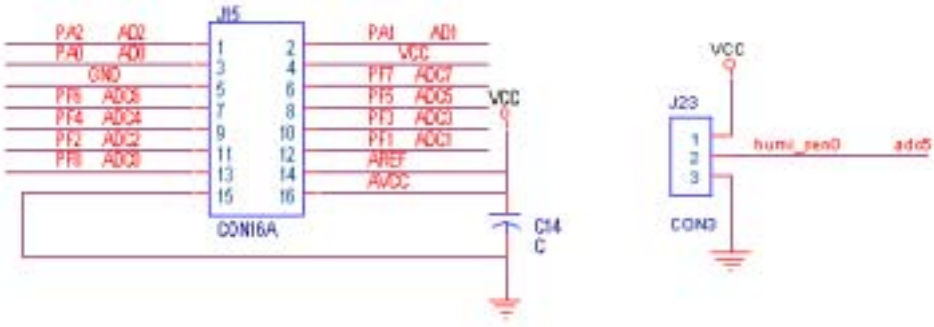

그림 28 습도측정 시스템 회로도

습도센서 부분에서는 온도센서와 같이 앞에서 기술한 내용과 같은 방식으로 5[V] 전원을 인가하여 주면 상대 습도(RH 0[%]~100[%])에 상응하는 출력 데이터(Output)가 나온다. 출력 부분의 데이터는 상대 습도 0[%]~100[%]가 0.8[V]~3.8[V]사이의 비로 출력 범위가 결정되어진다. 그러한 출력 부분을 처리하기 위해서 마이크로프로세서 즉, ATMega 128에 내장되어 있는 A/D 컨버터를 통해서

측정시스템과 제어시스템 간의 인터페이스를 구축하여 준다. 가장 이상적인 경우의 A/D 컨버팅의 범위는 0[V]~5[V]이지만 여러 가지 노이즈와 정확한 데이터의 측정 그리고 회로의 간결성과 효율성을 고려하여 회로를 구성함으로써 0.8[V]~3.8[V](ATmega128은 10비트 AD컨버터가 내장되어 있으므로 범위는 164~778)까지의 A/D 컨버팅 범위를 갖는 인터페이스를 구축하였다. 도출한 공식은 다음과 같다.

y=(100/614)x-16400/614

센서에 정확한 데이터의 출력을 위하여 다이오드를 사용하여 역전류를 방지하도록 하였고, 전원의 각 끝단에 컨덴서를 사용함으로써 여러 가지 상황에 따른 전원이 불안정한 경우와 노이즈에 대비하였다.

3) A/D 변환기

센서들로부터 나오는 전기적 신호는 대부분 아날로그 신호인데, 이를 컴퓨터에서 처리하기 위해서는 디지털 신호로 바꾸어 주어야 한다. 이러한 과정을 AD변환이라 하고, AD변환 전자소자를 ADC(Analog to Digital Convertor)라 한다. AD변환 방법에는 Successive-Approximation AD변환, Dual-Slope Integration AD변환, Parallel AD변환 등이 있다.

본 연구에서는 마이크로프로세서에 내장된 AD변환기를 사용하였다. ATmega128은 8개의 AD변환기를 내장하고 있으며, 10bit의 디지털 정보를 도출시킨다. 그림 29는 ATmega128에 내장된 Successive-Approximation AD변환 방법의 변환기 구조를 나타낸 것이다.

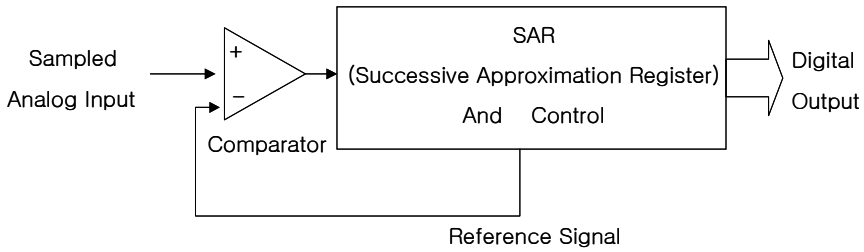

그림 29 Successive-Approximation AD변환

먼저 입력신호를 디지털 신호로 변환하는 도중, 입력신호가 변하는 경우를 막

기 위해서 AD 컨버터는 샘플링과 홀드 과정을 수행한다. 이렇게 함으로써 한번의 AD변환이 완전히 수행될 때까지 일정한 입력 신호레벨을 유지할 수 있다. 이는 곧 안정적이고 정확한 AD변환을 의미한다. 다음 단계에서, AD컨버터가 받아들일 수 있는 입력범위의 중간값(이 신호는 SAR로부터 나온다)과 샘플링된 아날로그 입력신호를 비교한다. 아날로그 입력신호가 비교신호 보다 크면 MSB를 1로 고정하고, 아날로그 입력신호가 비교신호 보다 작으면 MSB를 0으로 만든다. 이를 또다시 아날로그 입력신호와 비교하면 아직까지 입력신호가 비교신호보다 큼으로 다음 bit를 1로 만든다. 이렇게 해서 순차적으로 SAR에서 나온 비교신호와 아날로그 입력신호를 계속 비교하여 가장 근사 값이 얻어질 때까지 하위 bit들을 설정해 나간다. 모든 bit들의 설정이 끝나면 AD변환이 끝난다. Successive-Approximation AD변환 방법은 AD변환 속도가 비교적 빨라서 중·고속 ADC에 일반적으로 쓰이는 기술이다.

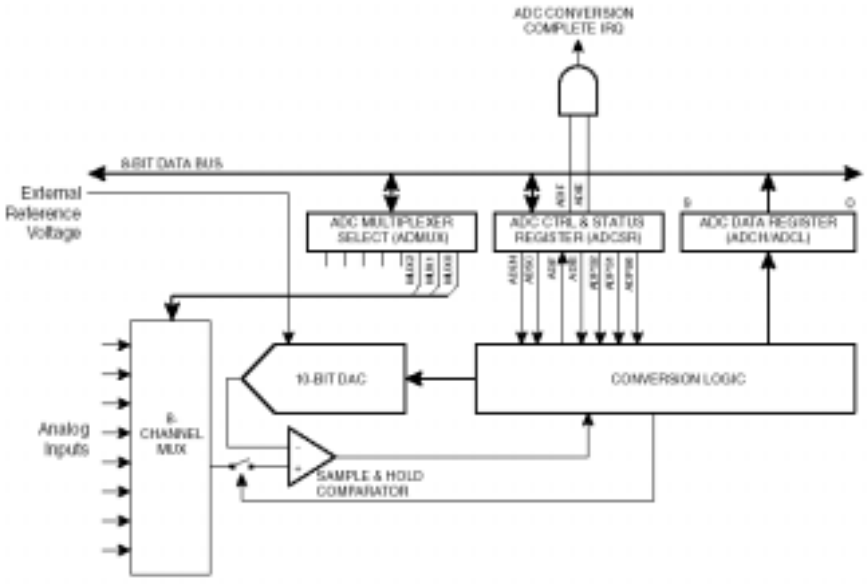

그림 30 ATmega 128의 AD변환부 블록도

그림 30은 ATmega 128의 AD 변환부이다. ATmega 128 내부의 AD변환기는 ADMUX와 ADCSR 레지스터를 사용하여 설정된다. ADMUX 레지스터는 8개의 채널 중 AD변환을 수행하고자하는 채널을 선택할 수 있는 레지스터이며, ADCSR은 AD변환기의 동작 및 상태를 제어하는데 사용하는 레지스터이다. 이들

레지스터를 이용하여 아날로그로 형태의 온도 및 습도 정보를 디지털 정보로 변환한다.

Bit	7	6	5	4	3	2	1	0	
$07 ($27)	–	–	–	–	–	MUX2	MUX1	MUX0	ADMUX
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

그림 31 ADMUX 레지스터

Bit	7	6	5	4	3	2	1	0	
$06 ($26)	ADEN	ADSC	–	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

그림 32 ADCSR 레지스터

다. 정밀 온도·습도 측정을 위한 시스템 설계

본 연구에서 사용된 제어시스템은 보다 더 정밀한 온도와 습도의 측정을 위하여 온도잡음 처리 회로설계, 1[%] 오차저항 사용, 정전압·정전류 회로를 설계하여 시스템에 추가하였다.

1) 온도잡음 처리

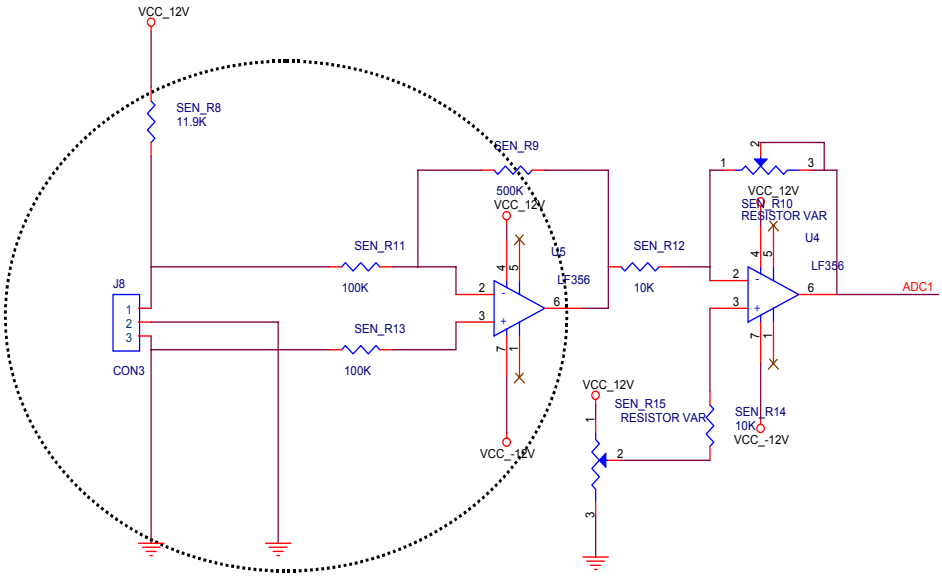

그림 33 온도잡음 처리

그림 33은 온도잡음 처리 회로를 나타낸 것이다. 증폭부에서 입력된 값을 처리하는 과정에서 2번과 3번을 접지하여 잡음을 줄여 정밀측정이 가능하도록 하였다.

2) 1[%] 오차저항 사용

일반적으로 많이 사용되는 탄소피막저항은 가격이 저렴하지만 잡음이 심하고 5[%] 내외의 오차를 가지고 있어 정밀한 측정을 하는데 한계가 있다.

본 연구에서는 1[%] 오차저항을 사용하여 보다 정밀한 측정을 할 수 있도록 설계하였다.

3) 정전압·정전류 회로 추가

부하의 유무 혹은 입력전압, 전류의 변동 그리고 주변 환경에 의한 영향에 의해서 전압과 전류의 출력값은 변화될 수 있다. 출력값의 변화는 제어시스템에 주요한 오작동의 원인이 될 수 있으므로, 본 연구에서는 정전압과 정전류 회로를 추가하여 정확한 온도와 습도를 측정할 수 있도록 하였다.

라. 온도·습도 센서의 정밀도 향상을 위한 처리 알고리즘 구현

1) 평균값 처리 알고리즘

측정값이 경우에 따라서는 같은 환경 상태에서도 다르게 나타날 수 있다, 이러한 경우 빈도를 가중치로 택하여 평균값을 계산한다.

측정값들이 x1, x2, x3, … 이고, 빈도가 각각 f1, f2, f3 … 이라면 평균값은 아래와 같이 계산할 수 있다.

$$\bar{X} = \sum_{I=1}^{N} fi\ xi$$

평균값은 온도와 습도 각각 500회씩 입력된 값을 산술평균으로 하여 계산하였다.

2) 프로그램 구현

본 제어시스템에서 사용된 평균값 처리 알고리즘은 다음과 같다

```
long ave_temp(int data)
{
    while(ave_temp_flag != 500)      {
            func_ave1 = func_ave1 + data;
            ave_temp_flag ++;
    }
    if(ave_temp_flag == 500)         {
            ave_temp_flag = 0;
            func_ave1 = func_ave1 / 500;
            return func_ave1;
            func_ave1 = 0;
    }
}

long ave_humi(int data)
{
    while(ave_humi_flag != 500)      {
            func_ave2 = func_ave2 + data;
            ave_humi_flag ++;
    }
    if(ave_humi_flag == 500)         {
            ave_humi_flag = 0;
            func_ave2 = func_ave2 / 500;
            return func_ave2;
            func_ave2 = 0;
    }
}
```

5. 제어・측정시스템과 PC간 통신 프로토콜

가. 측정 시스템 통신 프로토콜 정의

제어기간 디지털 유선 통신을 위해 많이 사용되고 있는 물리적인 계층 규약으로서 RS232, RS422, RS485가 있다. 이들 통신방법은 비동기식 통신 콘트롤러인 UART(Universal Asynchronous Receiver/ Transmitter)에서 나오는 TTL레벨의 신호를 노이즈에 강하고 원거리 전송을 할 수 있도록 Line Driver/Receiver를 사용한다.

본 연구에서 사용한 통신시스템은 1대 N(1 : 다수) 통신을 위해 RS485 통신규격을 사용하였다. 각각의 제어기들은 고유의 번호를 가지고 있으며, PC와 같은 Master에 해당하는 기기에서 제어기의 상태를 실시간으로 감시하고 제어한다.

1) 측정시스템 통신 프로토콜 정의

본 연구 개발에서 제작한 저온저장고용 제어시스템에는 RS485 통신을 수행할 수 있도록 SN75176 칩이 설치되어 있으며, 자신의 어드레스를 지정할 수 있도록 dip 스위치가 설치되어 있다. 제어시스템은 16개까지 network로 통합 연결이 가능하다. 본 연구에서는 여러 개의 제어시스템들이 통신하여 데이터를 처리하기 위해 통신규격으로 RS485통신을 사용하였다. 전체 시스템에서 사용되는 데이터 포멧은 패킷(packet)의 형태로 메세지가 전달되어지고 있다. 메세지 전달방식은 주로 상위에서 폴링하는 방식을 사용하였다.

가) 통신 패킷

비동기 통신에서는 단말기간의 통신속도 차이에 의해 통신 에러가 발생하므로 에러 보정이 필수적으로 필요하게 된다. 에러 보정을 위하여 8bit LRC 체크루틴을 각 패킷의 마지막에 전송하여 에러가 발생할 경우 데이터를 재전송 하는 방법을 택하였고 패킷은 다음과 같이 구성하였다.

표 8 패킷정보 형식

STA	SAddr	DAddr	FCode	DATA	LRC	END

DATA를 제외하고는 모두 1 바이트 코드이며 DATA의 길이는 데이터 성격에 따라 크기가 다르다. 각각의 코드 설명은 다음과 같다.

STA : 패킷의 첫 시작부분을 나타내는 코드
SAddr : 소스번지
DAddr : 목적지 번지
FCode : 기능 코드
DATA : 온도 또는 습도 데이터
LRC : 패킷의 LRC를 계산한 값
END : 패킷의 끝

STA로서는 'R'과 'Q'가 있다. 'Q'는 소스번지의 기기가 목적지 번지 기기에게 FCode의 내용을 요청하는데 사용되며, 이때 데이터 영역은 시간 정보를 전송한다. 'R'은 소스번지의 기기가 목적지 번지 기기에게 FCode의 내용에 대한 데이터를 전송하고 있음을 나타낸다. LRC는 STA를 제외한 SAddr에서 DATA까지의 정보를 Xor한 결과를 ASCII로 표현한 것이다. END로서는 0x0d값을 사용하였다. 온도 및 습도 데이터에 해당하는 기능코드는 표 9와 같다.

통신의 예를 들어보면

s	1	2	3	0	1	2	1	2	0	↵	
0x53	0x31	0x32	0x33	0x30	0x31	0x32	0x31	0x32	0x30	0x0d	(1)

R	2	1	3	0	1	2	1	2	0	↵	
0x52	0x32	0x31	0x33	0x30	0x31	0x32	0x31	0x32	0x30	0x0d	(2)

위의 정보 중에 (1)은 번지 1번의 기기가 2번의 기기에게 12시 12분에 기능코드 3(평균 온도)의 정보를 묻고 있는 내용이다. (2)는 (1)의 응답에 해당한 것으

로 번지 2번이 번지 1번의 기기에게 온도 센서 3의 현재 데이터는 1212임을 나타낸 것이다.

표 9 기능코드

기능코드	기능
0	온도 1
1	온도 2
2	온도 3
3	온도 4
4	평균 온도
5	습도 1
6	습도 2
7	습도 3
8	습도 4
9	평균 습도
10	온도 설정
11	습도 설정

나. 제어 시스템 통신 프로토콜 정의

제어시스템 또한 측정시스템 통신 방법과 마찬가지로 여러 개의 Controller들이 통신하여 데이터를 처리하기 위해 통신규격으로 RS485 통신을 사용하였으며, 사용되는 데이터 포멧은 패킷(packet)의 형태로 메세지가 전달되어지고 있다.

1) 통신 패킷

제어시스템 패킷도 표 10과 같이 측정 통신용 패킷과 같은 구조이다.

표 10 패킷 정보 형식

STA	SAddr	DAddr	FCode	DATA	LRC	END

제어 패킷에서도 STA로서는 'R'과 'Q'를 사용한다. 'Q'는 소스번지의 기기가 목적지 번지 기기에게 FCode의 내용을 요청하는데 사용되며, 이때 데이터 영역은 시간 데이터를 전송한다. 'R'은 소스번지의 기기가 목적지 번지 기기에게 FCode의 내용에 대한 데이터를 전송하고 있음을 나타낸다. END로는 0x0d값을 사용하였다. 온도 및 습도 데이터에 해당하는 기능코드는 표 11과 같다.

표 11 기능코드

기능코드	기능
0	Auto/Manual
1	comp ON/OFF
2	heat MC ON/OFF
3	SV ON/OFF
4	EV ON/OFF
5	DT ON/OFF
6	DH ON/OFF
7	OP ON/OFF
8	SP1 ON/OFF
9	ALARM ON/OFF

Master에 해당하는 PC에서 제어 데이터를 전송하고 Slave에서 제어 데이터를 오류 없이 잘 받았다면 "11111"을 데이터 영역에 포함하여 회신하고 오류 발생시에는 "00000"을 회신한다.

통신의 예를 들어, PC에서 수동 모드에서 SV와 ALARM을 ON 하고 나머지는 OFF하는 명령을 제어기에 송신하여 잘 받았다는 회신 신호 교신과정을 표현하면 아래와 같다.

9	8	7	6	5	4	3	2	1	0	
1	0	0	0	0	0	1	0	0	1	=> 1+8+512 = 521

S	1	2	3	0	0	5	2	1	6	↵	
0x53	0x31	0x32	0x33	0x30	0x30	0x35	0x32	0x31	0x36	0x0d	(1)

R	2	1	3	1	1	1	1	1	1	↵	
0x52	0x32	0x31	0x33	0x31	0x31	0x31	0x31	0x31	0x31	0x0d	(2)

다. 통신을 위한 하드웨어시스템 설계

1) RS485 규격

대표적인 비동기 직렬통신 규약으로 RS232, RS422, RS458가 있으며, 표 12에 각각의 규약에 대한 특징을 정리하였다. 표 12에서 알 수 있듯이 RS-232와 RS-423(Single-Ended 통신방식) 통신방식은 RS422와 RS485에 비해서 통신속도가 늦고 통신거리가 짧은 단점이 있으나 동작모드에서 알 수 있듯이 하나의 신호전송에 하나의 전송선로가 필요하기 때문에 비용절감의 장점이 있다. 현재의 RS422 또는 RS485 칩의 경우 표 12에 나와 있는 Driver와 Receiver의 수보다도 훨씬 많이 지원하고 있으며, RS485인 경우 최대 256의 노드를 갖는 칩도 있다

본 연구에서는 많은 노드 연결과 빠른 통신 속도를 구현하기 위해 RS485 규격을 사용하였다.

RS485는 EIA에 의해서 전기적인 사양이 규정되어 있으나 물리적인 코넥터 및 핀에 대한 사양은 아직 규정되어 있지 않다. RS485인 경우 RS232나 RS422처럼 Full Duplex가 아닌 Half Duplex 전송방식만 지원하기 때문에 RS422의 Multi-Drop 모드의 슬레이브 처럼 RS485의 모든 마스터는 TXD신호를 멀티포인트 버스(RS485의 모든 마스터가 공유하는 신호라인)에 접속 또는 단락시켜야만 할뿐만 아니라 RXD신호 역시 모드에 따라서는 접속, 단락의 제어를 하여야 한다.

표 12 비동기 직렬통신 규약

Specification	RS232C	RS423	RS422	RS485
동작 모드	Single-Ended	Single-Ended	Differential	Differential
최대 Driver/Receiver 수	1 Driver 1 Receiver	1 Driver 10 Receivers	1 Driver 32 Receivers	32 Drivers 32 Receivers
최대 통달거리	약 15 m	약 1.2 km	약 1.2 km	약 1.2 km
최고 통신속도	20 Kb/s	100 Kb/s	10 Mb/s	10 Mb/s
지원 전송방식	Full Duplex	Full Duplex	Full Duplex	Half Duplex
최대 출력전압	±25V	±6V	-0.25V to +6V	-7V to +12V
최대 입력전압	±15V	±12V	-7V to +7V	-7V to +12V

2) 결선도

멀티포인트 버스를 사용하는 시스템은 하나의 버스에 여러 개의 마스터를 연결하여 사용한다. 이 때문에 하나의 마스터가 다른 마스터와 통신을 할 경우에는 반드시 출력 개폐를 하여야 한다.

동시에 여러 개의 마스터가 출력하기 때문에 데이터가 충돌하는 현상이 발생하게 되는데 이러한 문제는 소프트웨어에 의해 해결될 수 있다. 이렇게 충돌 여부를 확인하는 방법 중 하나가 자기가 보낸 정보를 자기가 받아 보고 충돌여부를 확인하는 것인데 이것을 RS485 Echo 모드라 한다.

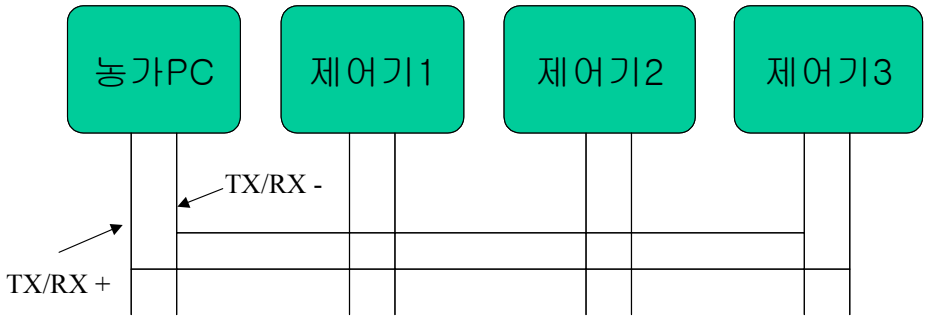

그림 34 결선도

3) RS485 통신 규약을 위한 하드웨어 구성

본 연구에서는 Network를 구성하기 위한 통신 드라이브 소자로서 SN75176A를 사용하였다. SN75176A는 표준 EIA/TIA-422-B와 ITU에서 권장하는 V.11. 규약을 준수하는 소자이다. SN75176A는 3상태 차분 구동기와 수신기를 내장하고 있다. 구동기능과 수신기능 방향 제어 핀에 의해 능동적 High와 Low를 설정할 수 있다.

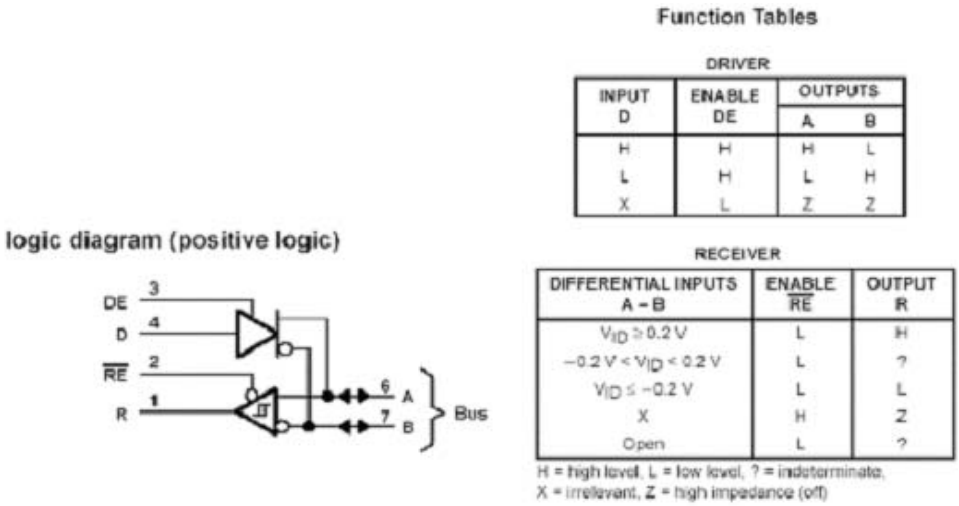

그림 35 SN75176 IC의 논리도 및 진리표

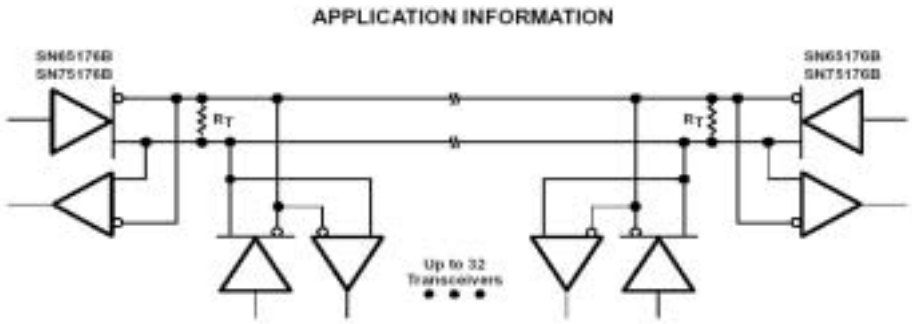

그림 36 SN75176의 응용회로 결선도

본 연구에서는 그림 36에서 볼 수 있는 바와 같이 마이크로프로세서

(ATMega128)의 포트 3을 이용하여 SN75176의 쓰기 Enable을 제어하도록 설계하였다. SN75176은 TX/RX 단자가 0일 때는 수신모드로 동작을 하며, 1일 경우 발신 모드로 동작을 수행하도록 회로가 설계되어 있는 것을 볼 수 있다.

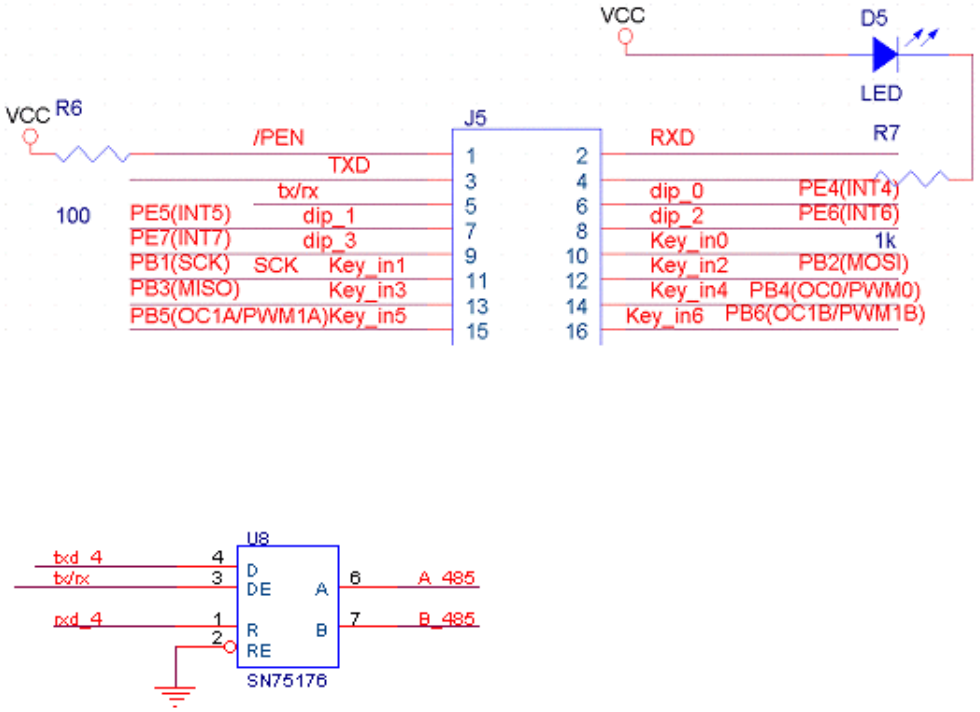

그림 37 본 연구에서 사용한 485통신 회로도

라. 통신프로그램 작성

1) 데이터 교신 과정

저온저장고 제어기와 PC간의 데이터 교신과정을 그림 38에 나타내었다. 그림에서 볼 수 있는 바와 같이 PC와 같은 Master의 정보요구 신호가 선행되어야 교신 동작이 이루어진다. 동작은 정보요구와 정보전송의 두 단계로 나뉜다.

Master의 정보요구 신호 후 Slave는 50[ms] 이내에 응답을 하도록 설계하였다. 그림 39와 같이 Master의 정보요구 신호 후 50[ms] 이내에 응답이 없는 경우 다음 slave node로 정보요구 작업을 옮겨 수행한다.

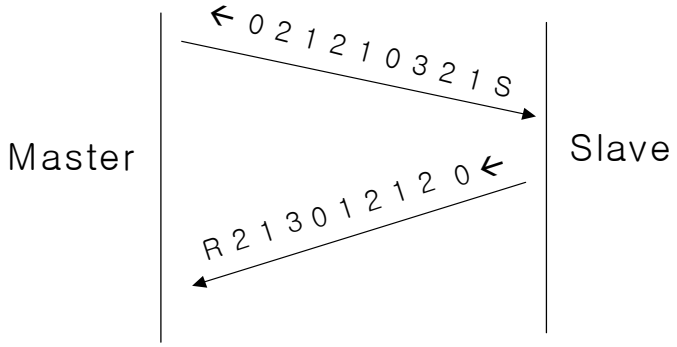

그림 38 데이터 교신과정1

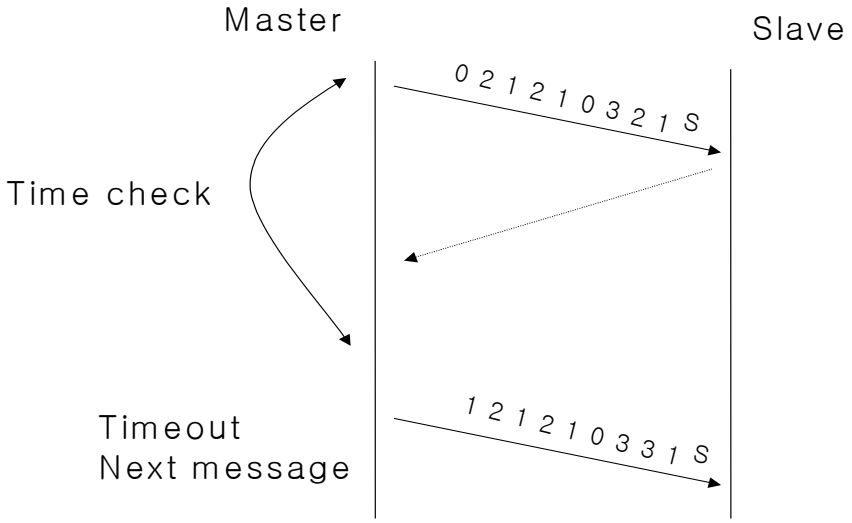

그림 39 데이터 교신과정2

2) Slave 시리얼 인터럽트 및 데이터처리 개요

ATmega128 내의 직렬 수신 버퍼 레지스터에 데이터가 1 byte 수신되면 데이터 수신 인터럽트가 걸리게 하여 이 인터럽트 루틴에서 데이터 처리 루틴을 호출하는 방법으로 데이터를 처리하였다. 이럴 경우 미처 데이터 처리가 종료되지 않은 상태에서 수신 버퍼가 존재할 경우, 이 인터럽트는 무시되며 데이터가 손실되는 상황이 발생할 수 있다. 이것을 해결하기 위해 인터럽트 루틴을 간략하

게 작성하고 링 버퍼 형식의 큐를 만들어 인터럽트가 호출될 때마다 수신버퍼에 들어온 데이터를 큐에 저장하는 방식을 택하였다. 큐의 데이터를 처리하는 방식은 큐에 데이터가 일정 개 이상 저장되면 큐의 데이터를 처리하는 방법으로 데이터 손실을 막을 수 있다.

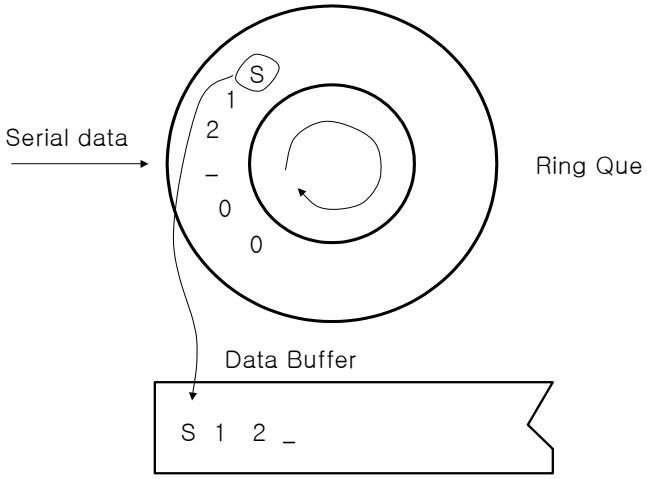

그림 40 Slave 시리얼 인터럽트 및 데이터처리

3) 통신관련 프로그램

컴퓨터와 통신프로토콜을 맞춰서 설정과 데이터를 전송하기 위한 부분이다.

```
SIGNAL(SIG_UART_RECV)
{
        R_buffer = inp(UDR);
        if ( bit_is_set(USR, 4) ) return;
        if(R_buffer != 0)
        {
                if((P_com == 9) || (P_com > 9 ))
                {
                        P_com = -1;
```

```
            signal_flag = 1;
}

P_com ++;

in_data[0] = r_com[4];
in_data[1] = r_com[5];
in_data[2] = r_com[6];
in_data[3] = r_com[7];
in_data[4] = r_com[8];

communi_r = atol(in_data);

if(r_com[3] == 0 + '0')              {
r_com[8] = (act_temp % 10) + '0';
                    temp_value = (act_temp % 10);
r_com[7] = (((act_temp % 100) - temp_value) / 10) + '0' ;
            r_com[6] = 0 + '0';
            r_com[5] = 0 + '0';
            r_com[4] = 0 + '0';        }

else if(r_com[3] == 1 + '0')         {
r_com[8] = (act_humi % 10) + '0';
                    temp_value = (act_humi % 10);
r_com[7] = (((act_humi % 100) - temp_value) / 10) + '0' ;
            r_com[6] = 0 + '0';
            r_com[5] = 0 + '0';
```

```
            r_com[4] = 0 + '0';        }

else if(r_com[3] == 2 + '0')        {
r_com[8] = (set_temp % 10) + '0';
                    temp_value = (set_temp % 10);
r_com[7] = (((set_temp % 100) - temp_value) / 10) + '0' ;
            r_com[6] = 0 + '0';
            r_com[5] = 0 + '0';
            r_com[4] = 0 + '0';        }

else if(r_com[3] == 3 + '0')        {
r_com[8] = (set_humi % 10) + '0';
                    temp_value = (set_humi % 10);
r_com[7] = (((set_humi % 100) - temp_value) / 10) + '0' ;
            r_com[6] = 0 + '0';
            r_com[5] = 0 + '0';
            r_com[4] = 0 + '0';        }

else if(r_com[3] == 4 + '0')        {
r_com[8] = (set_dh % 10) + '0';
                    temp_value = (set_dh % 10);
r_com[7] = (((set_dh % 100) - temp_value) / 10) + '0' ;
            r_com[6] = 0 + '0';
            r_com[5] = 0 + '0';
            r_com[4] = 0 + '0';        }

else if(r_com[3] == 5 + '0')        {
```

```
r_com[8] = (set_dm % 10) + '0';
                    temp_value = (set_dm % 10);
r_com[7] = (((set_dm % 100) - temp_value) / 10) + '0' ;
          r_com[6] = 0 + '0';
          r_com[5] = 0 + '0';
          r_com[4] = 0 + '0';       }

else if(r_com[3] == 6 + '0')       {
r_com[8] = (set_dw % 10) + '0';
                    temp_value = (set_dw % 10);
r_com[7] = (((set_dw % 100) - temp_value) / 10) + '0' ;
          r_com[6] = 0 + '0';
          r_com[5] = 0 + '0';
          r_com[4] = 0 + '0';       }

else if(r_com[3] == 7 + '0')       {
r_com[8] = (set_ddm % 10) + '0';
                    temp_value = (set_ddm % 10);
r_com[7] = (((set_ddm % 100) - temp_value) / 10) + '0' ;
          r_com[6] = 0 + '0';
          r_com[5] = 0 + '0';
          r_com[4] = 0 + '0';       }

else if(r_com[3] == 8 + '0')       {
r_com[8] = (set_dds % 10) + '0';
                    temp_value = (set_dds % 10);
r_com[7] = (((set_dds % 100) - temp_value) / 10) + '0' ;
```

```
            r_com[6] = 0 + '0';
            r_com[5] = 0 + '0';
            r_com[4] = 0 + '0';        }

else if(r_com[3] == 15 + '0')       {
r_com[8] = (set_dh - my_timer_h % 10) + '0';
            temp_value = (set_dh - my_timer_h % 10);
            r_com[7] = (((set_dh - my_timer_h % 100) -
                        temp_value) / 10) + '0' ;
            r_com[6] = 0 + '0';
            r_com[5] = 0 + '0';
            r_com[4] = 0 + '0';        }

else if(r_com[3] == 16 + '0')       {
            r_com[8] = (set_dm - my_timer_m % 10) + '0';
            temp_value = (set_dm - my_timer_m % 10);
            r_com[7] = (((set_dm - my_timer_m % 100) -
                        temp_value) / 10) + '0' ;
            r_com[6] = 0 + '0';
            r_com[5] = 0 + '0';
            r_com[4] = 0 + '0';        }

else if(r_com[3] == 20 + '0')       set_temp = communi_r ;
else if(r_com[3] == 21 + '0')       set_humi = communi_r ;
else if(r_com[3] == 22 + '0')       set_dh   = communi_r ;
else if(r_com[3] == 23 + '0')       set_dm   = communi_r ;
else if(r_com[3] == 24 + '0')       set_dw   = communi_r ;
```

```
                else if(r_com[3] == 25 + '0')       set_ddm   = communi_r ;
                else if(r_com[3] == 26 + '0')       set_dds   = communi_r ;

                r_com[P_com] = R_buffer;

                /*out_data[0] = 'R';
                out_data[1] = r_com[2] ;
                out_data[2] = r_com[1] ;
                out_data[3] = r_com[3] ;

                out_data[4] = r_com[4] ;
                out_data[5] = r_com[5] ;
                out_data[6] = r_com[6] ;
                out_data[7] = r_com[7] ;
                out_data[8] = r_com[8] ;
                out_data[9] = r_com[9] ;

                sbi(PORTE, 3);
                puttext(out_data);
                cbi(PORTE, 3);*/
        }
}
```

마. 통신장애를 위한 대안수립

본 연구에서는 멀티 통신에 따른 통신 충돌 발생 또는 왜란에 의한 통신 장

애 시 이를 감시하고 그에 대한 오작동을 방지할 수 있는 대안을 다음과 같이 수립하였다.

1) 제어장치 통신응답시간 감시

Master의 정보요구 신호 후 50ms 이내에 해당 Slave는 응답을 하도록 설계되어 있으므로 통신응답 시간을 감시하여 50ms이후의 신호는 잡음으로 간주하도록 프로그램을 작성하였다.

2) 동일 Address 감시 프로그램

멀티통신을 수행함으로 제어장치의 어드레스 설정이 중복되지 않도록 하여야 한다. 만일 중복 어드레스 설정 시 데이터의 충돌을 야기할 수 있다. 이를 확인하기 위해 node 확인 과정 프로그램을 작성하였다.

3) LRC 오류 감시

왜란에 의한 데이터 왜곡을 감지하기 위해 본 연구에서는 LRC 오류 감시 byte를 추가하여 통신패킷을 구성하였다. LRC 계산 방법은 STA를 제외한 SAddr에서 DATA까지의 정보를 Xor한 결과를 ASCII로 표현한 것이다.

6. PC기반 제어 프로그램 개발

본 연구에서는 저온 저장고의 감시 및 제어를 위하여 별도의 PC용 프로그램을 작성하여 원격지에서도 저온저장고에 대한 감시 및 제어를 실행할 수 있도록 하였다. 또한 장기간의 데이터를 저장하여 저온저장고의 주기적의 변화를 관찰할 수 있도록 하였다.

가. PC기반의 제어 프로그램 설계 및 구현

1) 제어 프로그램의 역할

본 연구에서 저온저장고는 제어시스템에 의해 감시 및 제어 되어진다. 따라서 사용자가 직접 저온창고에 가지 않고 가정이나 사무실에서 PC를 통해 저온저장

고의 현재 상태를 모니터링하고 제어하기 위한 프로그램이 필요하게 된다.

제어프로그램의 역할은 주기적으로 저온저장고를 제어하는 마이크로 컨트롤 시스템과의 통신을 통하여 원격지에서 저온저장고의 현재 상태를 모니터링 및 제어하게 된다.

2) 제어 프로그램의 구조

제어프로그램은 크게 3부분으로 구분할 수 있다. 제어시스템과 통신을 하는 통신 부분과, 사용자를 위한 사용자 인터페이스 부분 그리고 데이터 저장을 위한 데이터베이스 부분으로 구분할 수 있다.

■ 통신부분

제어시스템과 PC와의 통신은 RS-232C를 기본으로 하는 RS-485을 이용하였다.

■ 사용자 인터페이스 부분

저온저장고가 농가에 많이 보급되어져 있고 사용자가 고령층을 이루고 있으므로 조작이 쉽게 설계하였으며 한눈에 현재 상황을 볼 수 있도록 그래픽하게 설계하였다. 또한 사용자가 모니터링 하는 부분의 글씨의 폰트는 12 폰트 이상으로 하였다.

■ 데이터베이스 부분

제어시스템에서 주기적으로 전송되어져 오는 데이터를 저장하고 또한 여러 가지 설정값을 저장하기 위해서 데이터베이스를 사용하였다. 그러나 본 프로그램에서는 Oracle이나 SQL Server와 같은 대용량 DB를 다루는 데이터베이스를 사용하지 않고 델파이(delphi)에서 호환이 가능한 PARADOX 파일 데이터베이스를 사용하였다. 이를 통하여 별도의 데이터베이스 서버를 운영하지 않으면서 적은 용량으로 데이터베이스를 사용하여 프로그램 할 수 있다.

3) 제어 프로그램의 작동

프로그램의 작동은 저온저장고의 제어시스템과 타이머에 의한 주기적인 통신을 통하여 이루어진다. 제어시스템은 PC용 제어프로그램에서 데이터 요청이 있을 경우에만 데이터를 전송하게 된다. 제어 프로그램은 저온저장고에서 전송되어

진 데이터를 데이터베이스에 저장하며 제어시스템의 설정값을 관리자가 변경할 수 있도록 하였다. 이를 통하여 관리자는 원격지에서 저온저장고의 현재 상태를 모니터링 할 수 있으며 또한 설정값 변경 등을 위하여 저온저장고를 제어할 수 있다.

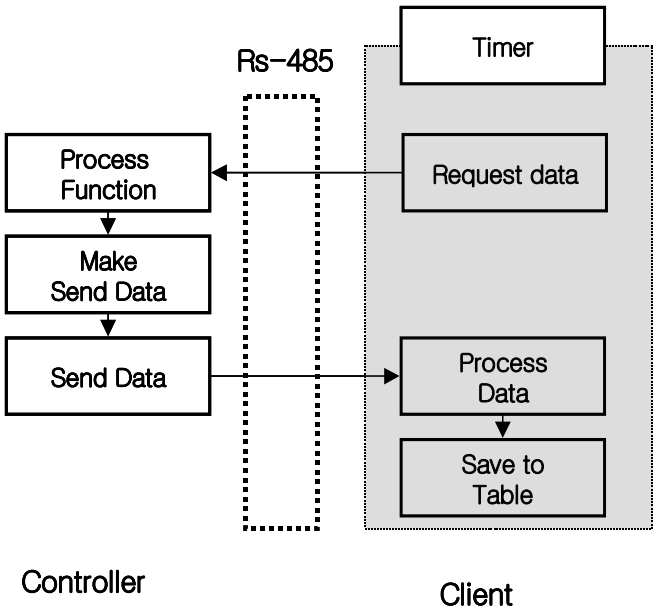

그림 41 제어프로그램의 작동 다이어그램

나. 저온저장고 관리 프로그램 작성

본 연구에서는 저온저장고에 대한 온도 및 습도의 변화특성 등을 통하여 저온저장고의 주기적인 상태 변화를 관찰함으로써 효과적인 운영을 할 수 있도록 하였다.

1) 관리 프로그램의 역할

제어시스템에서 전송되어진 데이터를 통하여 저온저장고의 온도 및 습도의 평균 값 및 시간대에 따른 변화량과 같은 주기적인 변화를 관찰 하고 이를 통하

여 합리적인 저온저장고의 운영을 할 수 있도록 하였다.

2) 관리 프로그램의 구조

데이터베이스에 저장되어진 데이터를 읽어 와서 변화량 및 주기에 따른 평균값 등을 그래프로 표현해 준다.

3) 화면

Login 화면: 사용자의 아이디와 비밀번호를 입력받아서 데이터베이스의 사용자 정보와 비교하여 등록된 사용자인가를 인증한다. 아이디와 비밀번호는 대소문자를 구분하며 다를 경우에는 프로그램을 종료한다.

그림 42 Login 화면

■ 초기화면

현재 온도와 습도는 게이지와 텍스트 두 가지 방법으로 표현 하였다. 게이지와 텍스트기는 타이머 설정 값에 따라 변화한다. 또한 설정값에 따라 게이지의 색띠 위치가 변화하여 현재의 온도와 습도가 설정값 보다 큰 값인가 작은 값인가를 직관적으로 판단할 수 있도록 하였다.

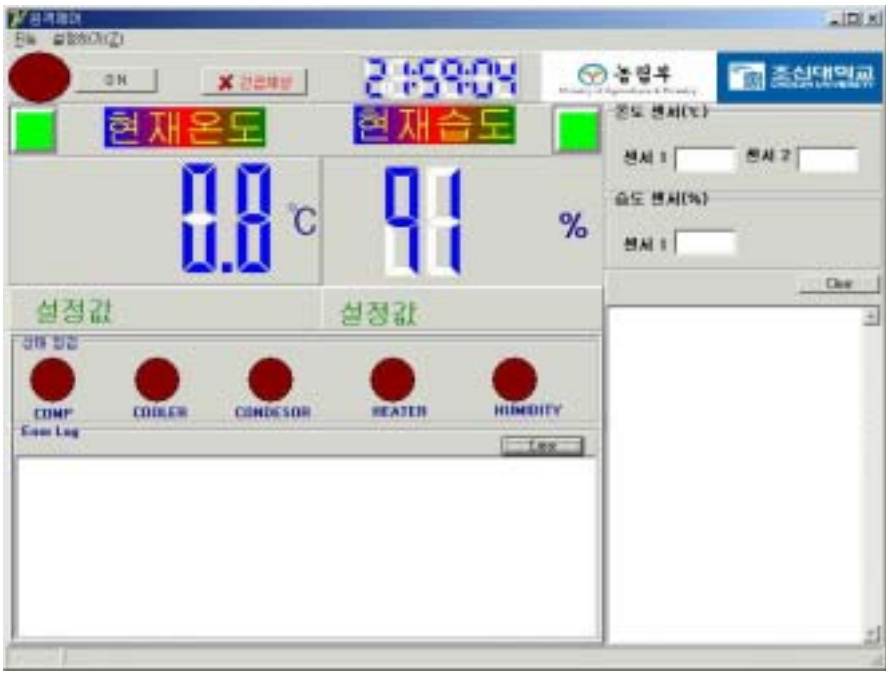

그림 43 초기화면

■ 변화량 관찰 화면

온도와 습도의 변화량을 나타내는 화면이다. 변화량은 데이터베이스에서 데이터를 참조로 그래프로 표현한다. 설정한 관찰 주기에 따라 저장고 내의 온도와 습도를 확인할 수 있으며, 또한 해당 주기 동안의 평균값을 볼 수 있다.

그림 44 변화량 관찰 화면

4) 데이터베이스의 설계

저온저장고를 관리하고 사용자 및 저장고의 상태 정보를 저장하기 위하여 데이터베이스를 설계하였다. 데이터베이스로는 PARADOX을 사용하여 별도의 데이터베이스 서버를 운영하지 않고 DELPHI의 BDE(Boland Database Engine)를 이용하여 테이블을 각각의 파일로 관리하였다. 본 시스템에서의 데이터베이스는 관리는 Query의 복잡성이나 복잡한 Transaction 관리가 필요하지 않으므로 파일을 이용한 데이터베이스의 Query문 이용이 효과적이다.

■ 사용자 관리 테이블

PC용 프로그램을 사용할 수 있는 사용자에 대한 정보를 저장하는 테이블이다. 사용자 ID와 비밀번호는 10자리로 되어 있으며 대소문자를 구분하도록 하였다. 또한 사용자의 정보가 변경될 경우에는 변경 연 월 일을 갱신하도록 하였다.

표 13 USERINFO Table

USERINFO	
USER_ID	ALPHA(10)
PASSWD	ALPHA(10)
UPYYYYMMDD	ALPHA(10)

■ 저장고 설정값 저장테이블

현재 저온저장고를 제어하는 마이크로 컨트롤러의 설정값을 저장하는 테이블로 각각의 설정값에 코드를 부여하였다. WH_CD는 여러 개의 저온저장고가 있을 경우 저장고에 부여되는 코드이며 WH_KD는 각각의 저온저장고의 설정파라미터에 대한 코드값이며 SET_VALUE는 설정코드에 대한 설정값이다.

표 14 WAREHOUSE_INFO Table

WAREHOUSE_INFO	
WH_CD	ALPHA(5)
WH_KD	ALPHA(2)
SET_VALUE	INTEGER

■ 저장고 설정 파라미터 설명테이블

설정 파라미터에 대한 설명을 저장하는 테이블이다. WH_KD는 WAREHOUSE_INFO의 WH_KD에 대한 값이며 WH_KD_DESC는 각각의 파라미터에 대한 설명을 나타내고 있다.

표 15 WAREHOUSE_INFO Table

WAREHOUSE_INFO	
WH_KD	ALPHA(2)
WH_KD_DESC	ALPHA(30)

통신 프로토콜 중 통신 함수를 정의하는 테이블이다. PARAM_ID는 함수 ID를 나타내고, PARAM_DESC는 함수에 대한 설명을 나타내며 IND는 함수에 대한 INDEX를 나타낸다.

표 16 COMM_PARAM Table

COMM_PARAM	
PARAM_ID	ALPHA(3)
PARAM_DESC	ALPHA(30)
IND	INTEGER

■ 온도데이터 저장테이블

마이크로 컨트롤러에서 전송되어진 온도데이터를 저장한다. WH_CD는 여러 개의 저온저장고가 있을 경우 저온저장고의 ID를 나타내며 YYYYMMDD는 데이터가 저장된 연 월 일을, HHMMSS는 데이터가 저장된 시 분 초를 나타낸다. 그리고 CURR_VALUE는 실제로 전송되어진 온도 값을 나타낸다.

표 17 TEMP Table

TEMP	
WH_CD	ALPHA(5)
YYYYMMDD	ALPHA(8)
HHMMSS	ALPHA(6)
CURR_VALUE	INTEGER

■ 습도데이터 저장 테이블

마이크로 컨트롤러에서 전송되어진 습도데이터를 저장한다. WH_CD는 여러 개의 저온저장고가 있을 경우 저온저장고의 ID를 나타내며 YYYYMMDD는 데이터가 저장된 연 월 일을, HHMMSS는 데이터가 저장된 시 분 초를 나타낸다. 그리고 CURR_VALUE는 실제로 전송되어진 습도 값을 나타낸다.

표 18 HUMID Table

HUMID	
WH_CD	ALPHA(5)
YYYYMMDD	ALPHA(8)
HHMMSS	ALPHA(6)
CURR_VALUE	INTEGER

7. 저온저장고 온·습도 제어실험

가. 온도와 습도의 상관관계에 관한 연구

본 연구에서 개발한 저온저장고용 컨트롤러의 검증을 위하여 온·습도 제어실험을 수행하였다. 온·습도 제어실험은 원예연구소 나주 배 시험장의 배 저온저장고에서 수행하였다. 본 연구에서 수행한 온·습도 제어실험은 온도와 습도의 특성 실험, 온도와 습도의 상관관계에 관한 실험을 수행하였다. 본 실험을 통하여 개발한 저온저장고용 컨트롤러는 안정하게 온·습도를 제어할 수 있음을 확인하였고, 원격 감시 및 제어 기능을 수행할 수 있음을 확인하였다.

1) 컨트롤러의 온·습도 특성 및 제어 실험

본 연구에서 개발한 저온저장고용 컨트롤러를 원예연구소 나주 배 시험장의 시험용 배 저온저장고에 적용할 수 있도록 설치하였다. 전자접촉기는 기존의 컨트롤러 패널에 부착된 것을 사용하고, 전자접촉기의 제어기능은 본 연구에서 개발한 컨트롤러로 수행하였다.

실험을 위한 준비 과정은 다음과 같다.

가) 제어패널 설치

나) 기존의 컨트롤러에 결선

다) 온도계수 값 보정 : Calibration

라) 습도계수 값 보정 : Calibration

마) 온도 및 습도 설정 : 온도 1[℃], 습도 90[%]

바) 제상주기 설정 : 5시간

사) 제상시간 설정 : 20분

아) 지연시간 설정 : 10분

설정 값들은 마이크로프로세서의 비 휘발성 메모리부에 저장되므로 전원을 차단하여도 그 내용은 지워지지 않는다. 먼저 이런 설정값 저장 기능에 대하여 검증하였다. 검증결과 모든 설정 정보들이 안정적으로 저장될 수 있음을 확인하였다.

2) 온도와 습도의 상관관계에 대한 연구

본 연구에서는 온도와 습도의 상관관계를 조사하였다 조사 대상 저온저장고의 상태는 다음과 같다.

저장고의 크기 : 2.4[m]×4.7[m]×3.4[m](가로×세로×높이)

배 저장량 : 공간의 50[%] 정도 점유

가) 온도와 습도의 계측

그림 45에서는 시간에 따른 온·습도의 변화 그래프를 표현하였다. 그림에서 볼 수 있는 바와 같이 온도의 경우 약 0.2[℃]에서 1.8[℃]까지 변화하는 것을 확인하였다. 또한 습도의 경우는 약 78[%]에서 99[%]까지 변화함을 확인하였다. 실험에서 알 수 있는 것은 냉장장치의 동작 및 정지 시에 온·습도는 지연시간을 갖고 변화함을 확인하였다.

나) 저온저장고 온도와 습도의 상관관계

저장고의 냉장장치 운전 및 정지 시의 온도와 습도의 변화 특성을 보다 자세히 알아보기 위해 그림 45의 시간영역을 확대하여 그림 46에 나타내었다. 그림 46에서 볼 수 있는 바와 같이 실험결과 온도와 습도는 반비례 변화 특성을 나타냄을 확인하였다. 즉 온도가 상승할 경우 온도 상승에 따른 상대습도가 저하되었

으며, 냉장장치 동작의 경우 온도 하강에 따른 상대습도의 상승을 확인하였다.

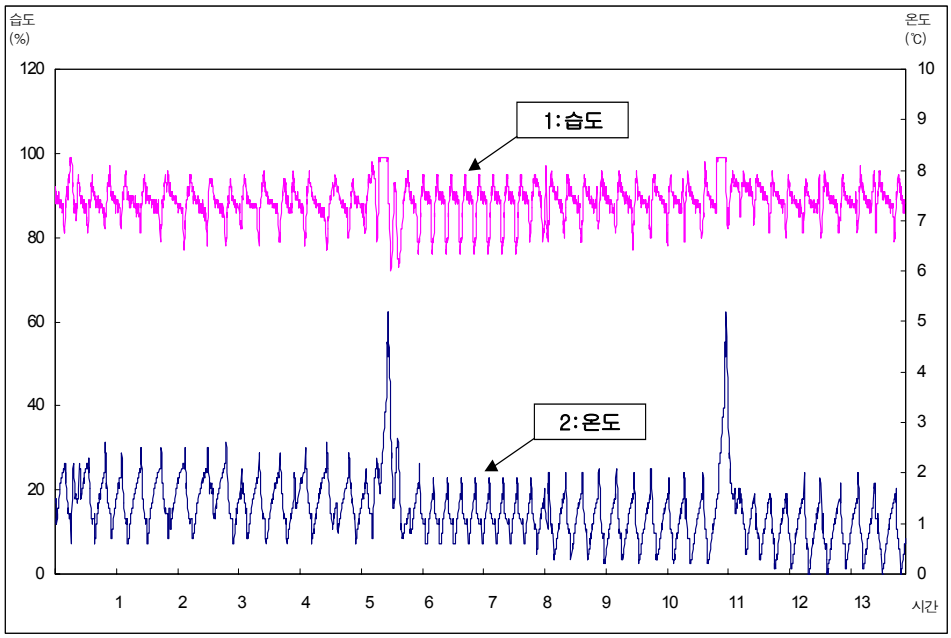

그림 45 시간에 따른 온·습도 변화 그래프

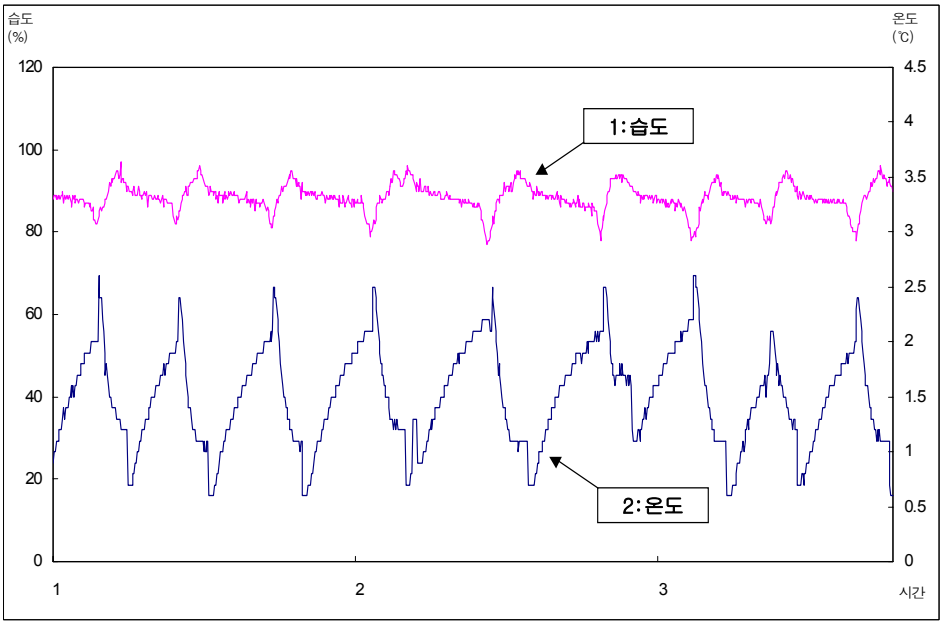

그림 46 저온저장고 온도와 습도의 상관관계

다) 제상시 온도와 습도의 상관관계

저장고의 제상운전시의 온도와 습도의 변화 특성을 보다 자세히 알아보기 위해 그림 45의 시간영역을 확대하여 그림 47에 표시하였다. 그림 47에서 볼 수 있는 바와 같이 온도와 습도가 냉장장치 운전 시와는 달리 반비례 관계를 나타내지 않음을 확인하였다. 즉 제상기간 동안에 고내 온도가 상승하였고 습도 역시 증가하였다. 일반적으로 저장고의 규모가 큰 경우는 제상기간 동안 온도와 습도가 반비례하여 상대습도가 감소하여야 하나, 본 실험의 경우는 저장고의 크기가 비교적 작기 때문에 제상 시에 성애의 증발로 발생한 수분에 의해 상대적으로 습도가 많이 증가하였음을 확인하였다. 이러한 현상은 좁은 공간에서는 온도 상승의 영향보다 수분 증가의 영향이 더 우세하게 작용하기 때문으로 사료된다.

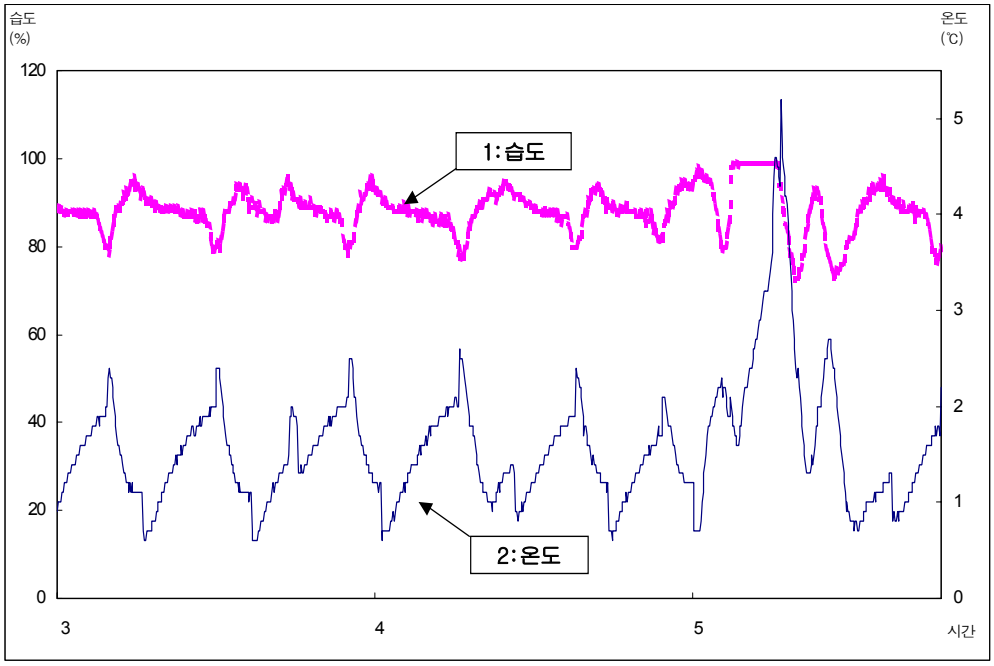

그림 47 제상시 온도와 습도의 상관관계

나. 시스템 오작동에 대한 안정성 확보

본 연구에서는 시스템 오작동에 대해 하드웨어적인 측면과 소프트웨어적인 측면에서 오작동에 대한 대안을 수립하였다.

1) 하드웨어적인 측면

본 저온저장고 관리 시스템은 크게 서버 부분, 클라이언트 부분, 컨트롤러 부분으로 나뉘어져 있다. 본 연구에서는 서버 부분이나 클라이언트 부분에 이상이 있거나 컨트롤러 부분에서 클라이언트 부분으로 통신이 이루어지지 않더라도 저장고의 온·습도 제어는 정상적으로 수행되어질 수 있도록 컨트롤러 부분을 구성하였다.

컨트롤러 부분에 원하는 온도 및 습도를 설정할 수 있도록 입력 장치를 구성하였으며 또한 관리자 및 사용자가 허용할 수 있는 경계 값을 지정할 수 있도록 하였다. 이는 온도 및 습도 변화가 있더라도 경계 값을 넘지 않으면 정상적으로 운영되어지는 것으로 인정하고 냉장장치에 서리가 과다하게 형성되어 온도 경계 값을 넘어 설 경우 알람을 표시해 준다. 알람 경보가 있을 시 관리자 및 저장고 소유주는 서리 제거를 위해 '긴급제상' 버튼을 작동하여 관리할 수 있게 된다.

2) 소프트웨어적인 측면

소프트웨어적인 측면에서는 통신의 장애 및 경계 값을 통한 관리가 이루어지도록 하였다. 클라이언트 프로그램과 컨트롤러간의 통신 장애 및 클라이언트 프로그램과 서버간의 통신 장애가 발생 할 경우 알람을 알리도록 하였다. 클라이언트 프로그램과 컨트롤러간의 시리얼 통신에 장애가 발생할 경우에는 클라이언트 프로그램이 실행되어지는 OS(Operation System)에서 이벤트가 발생하는데 클라이언트 프로그램에서 이 이벤트를 통해서 통신의 장애를 감지하고 이를 서버로 전송한다. 서버와 클라이언트 프로그램간의 통신 장애는 클라이언트 프로그램은 일정 주기로 서버에 접속하여 데이터를 전송하는데 서버에서 데이터가 주기적으로 관리자가 설정한 일정 시간 동안 들어오는지를 점검하여 데이터 전송주기가 일정 시간이 아닌 경우에는 에러로 간주하고 알람을 알려준다.

3) 시스템의 오작동에 대한 안정성 확보 실험

시스템에 오작동이 발생하면 저온저장고에 저장된 작물에 해를 미친다. 따라서 본 연구에서는 각종 오류에 대한 대책을 수립하였으며, 이들에 대한 검증 실험을 수행하였다.

가) 시스템 설정 오류에 대한 대책

온도 및 습도의 설정치를 터무니없는 수치(온도 -5℃이하 또는 10℃이상, 습도 50%이하 또는 100%이상)로 입력하여 보았다. 이러한 경우 시스템에서는 즉각 이상 설정임을 감지하고 설정의 이상 상태를 알람으로 알려주었으며, 이상 설정을 하여 냉장시스템을 가동시켜도 설정을 무시하고 안정적으로 동작하면서 이상 상태임을 계속적으로 표시하는 기능이 작동함을 확인하였다.

나) 온도 및 습도의 동작범위 초과 오류에 대한 대책

배의 동결 온도는 -1.6[℃]이고 고내의 최적 상대습도는 90[%]~95[%]이다. 냉해를 피하기 위해 온도의 하한선을 0[℃]로 설정하였으며, 고내의 상대습도 상한선을 95[%]로 설정하였다. 따라서 온도가 0[℃] 이하의 온도가 되었음에도 불구하고 냉장기가 동작하는 경우가 발생되어서는 안 되며, 습도가 95[%]에 도달되었음에도 불구하고 가습장치가 동작되어서는 안 되므로 동작범위의 초과 시에 이를 차단하는 기능 즉 강제제어 기능을 제어기 프로그램에 삽입하여 실험을 수행하였다. 실험결과 오류 없이 0[℃] 이하일 경우는 냉장기가 상대습도 95[%]일 경우는 가습기의 동작이 정지함을 확인하였다.

표 19 온도와 습도의 정상제어 및 강제제어 구간

강 제 제 어				정 상 제 어		
	동결점		하한점		설정온도	
-1.6℃이하	-1.6℃	-1.6~0℃	0℃	0~1℃	1℃	1℃이상

정 상 제 어			강 제 제 어	
	설정습도		상한점	
90%이하	90%	90-95%	95%	95%이상

다) 센서 오류에 대한 대책

본 연구에서는 컨트롤러에 4개의 온도센서와 4개의 습도센서를 부착할 수 있

도록 설계하였으며, 이중 한 개의 센서값이 임계 오차값을 벗어나면 해당 센서의 수치를 무시하고 나머지 3개로서 정확한 온도의 유추를 수행하며, 센서 이상상태임을 알려주는 기능을 추가하였다. 이에 대한 현장 실험을 수행하였다. 시험결과 이상 없이 동작함을 확인하였다.

표 20 온도 및 습도 센서의 오류 처리(임계 오차값 5℃, 30%)

	센서1	센서2	센서3	센서4
온도[℃]	1	1.2	1.1	7.5
처리	정상	정상	정상	무시

	센서1	센서2	센서3	센서4
상대습도[%]	90	92	55	89
처리	정상	정상	무시	정상

라) 통신기능 오류에 대한 대책

클라이언트 프로그램과 컨트롤러 사이의 통신 이상은 통신장애 이벤트를 이용하여 장애를 감지하였고 서버와 클라이언트 프로그램간의 통신 장애는 클라이언트 프로그램에서 서버 프로그램으로 관리자에 의해 설정되어진 주기 마다 데이터를 전송하게 되어져 있는데 그 주기 시간에 데이터 전송이 되어지지 않을 경우 서버와 클라이언트 간에 통신 장애가 발생한 것으로 간주하고 재 통신이 이루어 질 때까지 알람을 표시하도로 구성하였다. 이를 통하여 클라이언트 프로그램과 컨트롤러간의 통신장애 및 클라이언트 프로그램과 서버 프로그램간의 통신 장애를 모두 관찰할 수 있게 된다. 실험결과 작동이 원활히 수행됨을 확인하였다.

8. 데이터베이스 구축

가. 업무 분석

저온저장고 시스템의 운영절차를 분석하고 안정적인 시스템의 운영과 데이터

의 안정된 저장 및 효과적인 데이터 확인을 위한 테이블의 정의를 위하여 업무 분석을 선행하였다.

1) 저온저장고 시스템의 구성 분석

본 연구에서 구성되어진 저온저장고 제어시스템은 물리적인 5가지 구성 요소와 업무적인 2가지 구성요소로 정의할 수 있다.

가) 물리적인 5가지 구성 요소

저온저장고의 원격 관리를 위해서 저온저장고를 직접 제어하는 컨트롤러, 컨트롤러와 통신하여 서버에 데이터를 전송하는 클라이언트 프로그램, 클라이언트 프로그램에서 전송되어지는 데이터를 처리하고 관리자의 제어 명령을 클라이언트 프로그램에 전송하는 서버 프로그램, 데이터를 저장하는 DBMS 그리고 저온저장고를 제어하고 상태를 파악할 수 있는 웹브라우져 등 5가지 구성 요소를 정의하였다.

(1) 컨트롤러

컨트롤러는 저온저장고에 있는 온도 센서와 습도 센서의 값을 피드백 받아 압축기, 증발기, 응축기, 제상히터, 습도기 등을 제어한다. 또한 현재 온도, 습도, 상태 값을 주기적으로 클라이언트 프로그램의 요청에 의해 전송하며 클라이언트 프로그램에서 제어명령을 전송하기 위한 명령을 수행한다.

(2) 클라이언트 프로그램

클라이언트 프로그램은 컨트롤러와 서버간의 통신을 중재하는 부분으로 컨트롤러에서 전송되어진 온도, 습도, 상태 값을 서버에 전송한다.

(3) 서버 프로그램

서버 프로그램은 웹 어프리케이션 서버인 Weblogic 기반 하에서 운영되어지며 클라이언트 프로그램에서 전송되어지는 값을 데이터베이스에 저장하고 저온저장고에 전송할 컨트롤 데이터를 전송해 준다. 또한 사용자 및 관리자가 시스템을 사용할 수 있는 웹페이지가 운영되어진다.

(4) DBMS

저온저장고의 온도, 습도, 상태 값 및 여러 데이터를 저장하고 시스템의 요구가 있을 시 안정적이고 빠르게 데이터를 제공해준다.

(5) 웹브라우저

관리자 및 저온저장고 소유주는 언제 어디에서나 웹브라우져를 통해 시스템에 접근할 수가 있다. 관리자는 전체 저온저장고의 환경 및 상태를 모니터링 할 수 있으며 또한 각각의 저온저장고를 제어할 수 있다. 소유주는 자신의 저온저장고의 온도, 습도 및 상태를 확인할 수 있으며 개별적으로 제어할 수 있다.

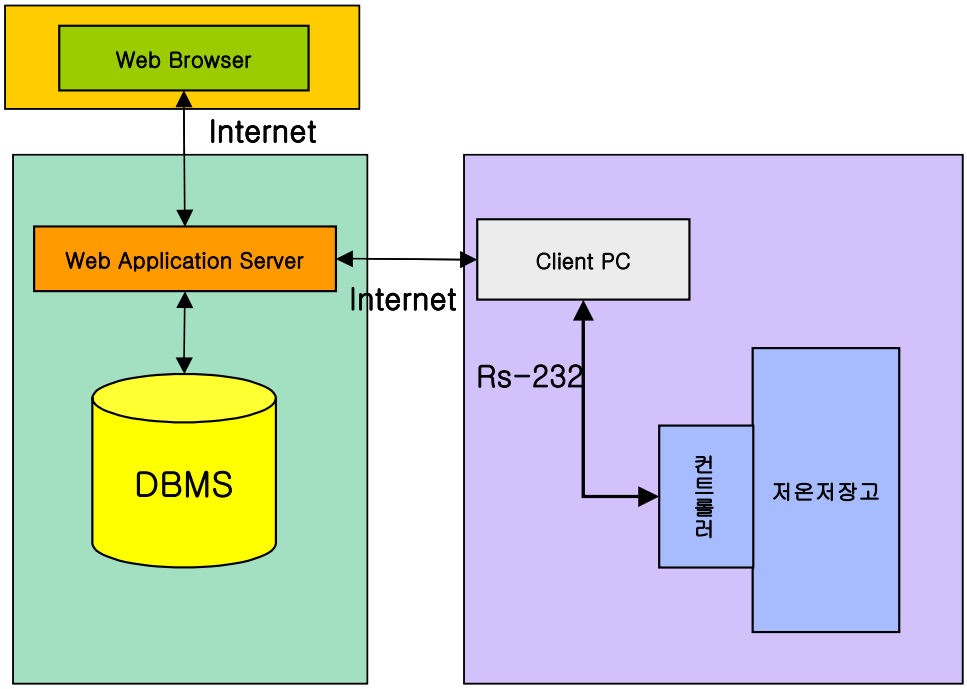

그림 48 저온저장고 시스템의 물리적 구성

나) 업무적인 2가지 구성 요소

(1) 저온저장고의 모니터링

■ 저온저장고에서 보내진 데이터를 기준으로 온도, 습도, 상태를 확인한다.

■ 저온저장고에서 주기적으로 데이터를 전송한다.

■ 저온저장고에서 전송되어지는 데이터는 온도, 습도, 상태값이다.

■ 관리자 및 소유주는 웹브라우져를 통해 데이터를 확인한다.

■ 데이터의 입력과 데이터의 확인은 DBMS를 통해 인터페이스가 이루어진다.

(2) 저온저장고의 제어

■ 관리자 및 소유주는 웹브라우져를 통해 각각의 저온저장고를 제어할 수 있다.

■ 제어 데이터를 클라이언트 프로그램이 서버에서 주기적으로 가져간다.

■ 제어 데이터는 DB에 저장되어진다.

■ 새로운 제어 데이터가 있을 경우 이전 데이터를 삭제하고 새로운 제어 데이터만을 보유하고 있다.

■ 제어 데이터는 온도, 습도, 제상주기, 제상시간, 제상 후 지연시간, 강제제상으로 이루어진다.

다) 기타

■ 게시판, 공지사항, Q&A, 사용자 관리

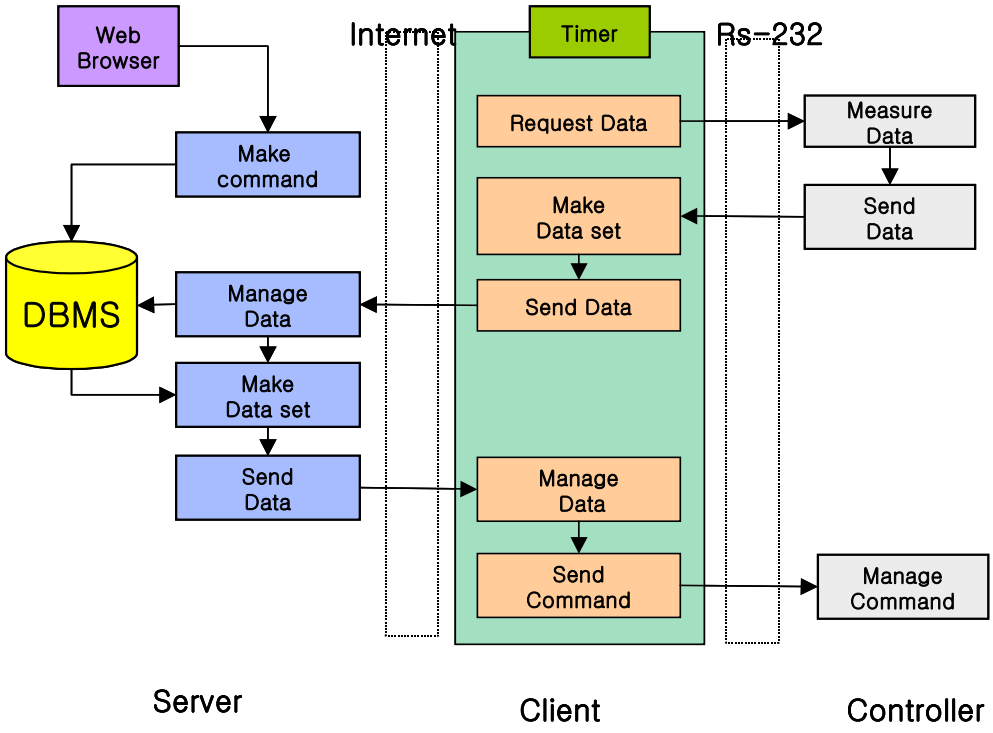

그림 49 시스템간 데이터 전송 흐름도

나. 데이터베이스 설계

1) 데이터베이스의 필요성

데이터를 저장하는 일반적인 방법으로는 크게 '파일시스템을 이용한 데이터의 저장방법'과 '데이터베이스를 이용한 저장방법'으로 구분할 수 있다. 파일시스템을 이용한 데이터의 저장은 별도의 시스템을 사용하지 않고 단지 파일 형태로 데이터를 저장할 수 있다는 장점이 있지만 데이터의 저장 및 유지 그리고 데이터의 추출에 있어 여러 가지 문제점을 가지고 있다. 파일시스템에서 응용 프로그램은 논리적 파일 구조와 물리적 파일 구조가 1:1로 대응될 것을 요구한다.

이러한 환경 하에서 데이터의 공용은 어렵게 되어 결국 하나의 응용만을 위해 사용되게 된다. 파일 시스템에서의 문제점은 크게 데이터의 종속성과 데이터의 중복성으로 집약시킬 수 있다.

표 21 파일 설계와 DB 설계의 차이

파일 설계	DB설계
· 데이터는 개별 프로그램에 의존 · 특정 프로그램의 부속물 · 특정 프로그램을 처리하기 위한 처리 지향적 관점에서 설계	· 데이터는 공동재원으로 다수 사용자의 요구를 충족시켜야함 · 현재 및 미래 요구사항을 충족시키기 위해 필요 데이터를 수집 가공하는 데이터 지향적 관점에서 설계

2) 데이터 베이스의 특성

데이터베이스는 다음과 같은 특성을 가지고 있다.

가) 실시간 접근성

컴퓨터가 접근할 수 있는 저장 장치에 수록된 데이터베이스는 수시적 이고 비정형적인 질의(query)에 대하여 실시간 처리로 응답할 수 있어야 한다. 실시간 처리는 생성된 데이터를 즉시 컴퓨터에 보내어 그 처리 결과를 보고 다음 의사결정에 바로 반영할 수 있게 하는 처리 방식을 말한다. 따라서 실시간에서의 응

답 시간은 단 몇 초를 넘지 않아야 한다. 일반적으로 온라인 처리라 하면 보통 이 실시간 처리를 의미한다. 데이터베이스는 주로 실시간 처리를 위해 활용되는 특성을 가지고 있다.

나) 계속적인 변화

어느 한 시점에 데이터베이스가 저장하고 있는 내용은 곧 데이터베이스의 상태를 의미한다. 그런데 이 데이터베이스의 상태는 정적이 아니고 동적이다. 즉 데이터베이스는 새로운 데이터의 삽입, 기존 데이터의 삭제, 갱신으로 항상 변하고, 또 그 속에서 현재의 정확한 데이터를 유지해야 된다. 특히 데이터베이스는 항상 변하는 현실을 반영하는 것이기 때문에 계속적으로 변하지 않으면 안 된다. 이러한 계속적인 변화 특성 때문에 데이터베이스를 정확하게 관리한다는 것이 더욱 어려운 문제가 된다.

다) 동시 공용

데이터베이스는 서로 다른 목적을 가진 응용들을 위한 것이기 때문에 여러 사용자가 동시에 자기가 원하는 데이터를 접근 이용할 수 있어야 한다. 이것은 한 가지 특정 목적으로 하나의 응용 프로그램에 의해 접근되는 데이터와 다르며, 여러 프로그램이 같은 데이터를 직렬적으로 공용하는 개념과도 전혀 다르다. 데이터베이스의 서로 다른 부분을 여러 사람이 서로 다른 방법으로 동시에 공용한다는 것은 관리와 조직 면에서 복잡하게 된다. 그것은 모든 데이터베이스의 데이터 요소들이 처음부터 항상 동시 공용이 가능하도록 조직 관리되어야 하기 때문이다.

라) 내용에 의한 참조

데이터베이스 환경 하에서 데이터의 참조는 수록되어 있는 데이터 레코드들의 주소나 위치에서가 아니라 사용자가 요구하는 데이터의 내용, 즉 데이터가 가지고 있는 값에 따라 참조된다. 일반적으로 사용자가 참조하기를 원하는 데이터의 조건을 명시하면 이 조건을 만족하는 모든 레코드들은 그들이 어디에 위치하든지 간에 하나의 논리적인 단위로 취급되고 접근된다. 이것은 전통적인 컴퓨터 시스템에서 모든 데이터가 기본적으로 주소로 참조되는 것과는 다르다. 예를 들

어, “기말성적이 90점 이상인 학생을 검색하라”는 요청이 있다면, 학생 레코드의 기말 성적 필드 값이 90점 이상이 되는 학생 레코드들은 그 수나 위치에 관계없이 모두 하나의 논리적 단위 속에 포함되어 검색된다는 것이다.

3) 데이터베이스 관리 시스템

데이터베이스 관리시스템(DBMS: DataBase Management System)은 파일 시스템에서 야기된 데이터의 종속성과 중복성의 문제를 해결하기 위해 제안된 시스템이다. DBMS는 응용 프로그램과 데이터의 중재자로서 모든 응용 프로그램들이 데이터베이스를 공용할 수 있도록 관리해 주는 소프트웨어 시스템으로 정의할 수 있다. 이러한 시스템 하에서 응용 프로그램들이 데이터베이스를 이용하기 위해서는 데이터베이스 관리 시스템을 통해서만 가능하다. 이것은 DBMS가 데이터베이스의 구성, 접근 방법, 관리 유지에 대한 모든 역할을 수행하고 있다는 것을 의미한다.

그림 50에 나타난 것처럼 응용 프로그램은 DBMS를 이용하여 데이터베이스의 생성, 접근방법, 처리절차, 보안, 물리적 구조 등에 대해서 관여할 필요 없이 원하는 데이터와 처리 작업만을 DBMS에 요청하면 된다. DBMS는 데이터베이스를 종합적으로 조직하고 접근하며 전체적으로 통제할 수 있는 프로그램들로 구성되어 있기 때문에 응용 프로그램의 요청을 수행시켜줄 수 있다.

DBMS의 필수 기능은 다음과 같이 정의할 수 있다.

가) 정의 기능

다양한 응용 프로그램과 데이터베이스가 서로 인터페이스를 할 수 있는 방법을 제공하는 것이다. 즉 구현된 하나의 물리적 구조의 데이터베이스로 여러 사용자들의 다양한 형태의 데이터 요구를 지원해 줄 수 있도록 가장 적절한 데이터베이스 구조를 정의할 수 있는 기능을 말한다. 이 데이터 구조의 정의는 DBMS의 가장 중요한 기능 중의 하나로서 각 응용 프로그램은 이 데이터 정의를 통해서 원하는 데이터를 DBMS에 표현한다.

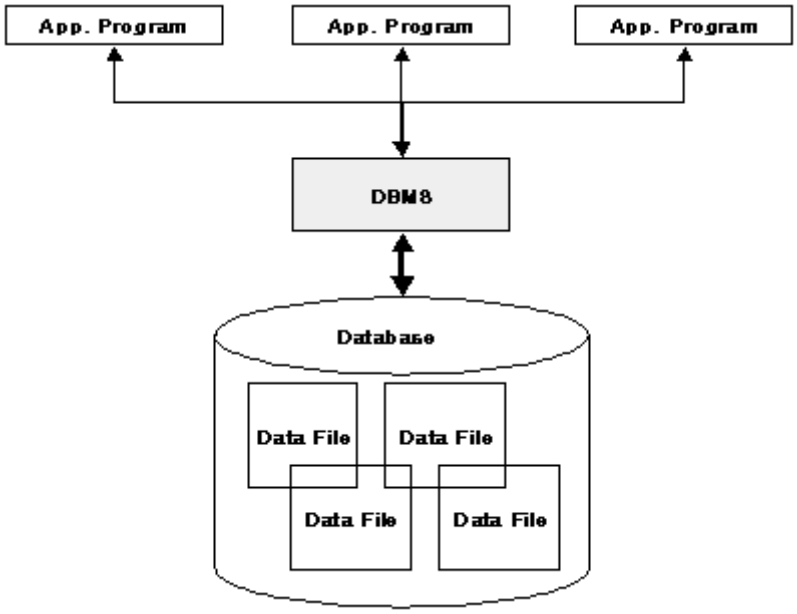

그림 50 DBMS와 응용 Program의 관계

나) 조작 기능

사용자와 데이터베이스 사이의 인터페이스를 위한 수단을 제공한다. DBMS는 데이터베이스를 이용하는 사용자의 요구에 따라 체계적으로 데이터베이스를 접근하고 조작할 수 있어야 한다. 이것은 데이터의 검색, 갱신, 삽입, 삭제 등 데이터베이스 연산을 지원하는 도구 언어를 통해서 구현될 수 있다.

다) 제어 기능

DBMS는 공용 목적으로 관리되는 데이터베이스의 내용에 대해 항상 정확성과 안전성을 유지할 수 있는 제어기능을 가지고 있어야 한다. 이 정확성은 데이터 공용의 기본적인 가정이며, 관리의 제약 조건이 된다.

4) Table 명세서

- 테이블 명: wh_cmd

  * 서버에서 클라이언트프로그램에 제어명령을 전송시 사용하는 임시테이블
  * 저장고 번호를 PRIMARY KEY로 사용하며 서버에서 제어명령이 전송되어지면 기존의 명령 row를 update 한다.

* field 설명
  - wh_cd
    + 저장고 번호
    + 데이터 타입 varchar
    + 크기 5
  - wh_cmd_cd
    + 제어명령 코드
    + 데이터 타입 char
    + 2
  - wh_cmd_value
    + 제어명령에 따른 파라미터
    + 데이터 타입 varchar
    + 4 default
  - yyyymmdd
    + 제어명령이 내려진 연 월 일
    + 데이터 타입 varchar
    + 크기 8
  - hhmmss
    + 제어명령이 내려진 시 분 초
    + 데이터 타입 varchar
    + 크기 6

▪ 테이블 명: wh_cmd_info

* 서버에서 클라이언트 프로그램에 제어명령의 설명 테이블
* 제어명령코드를 PRIMARY KEY로 사용
* field 설명
  - wh_cmd_cd
    + 제어명령 코드
    + 데이터 타입 char
    + 크기 2

- wh_cmd
  + 제어명령 설명
  + 데이터 타입 varchar
  + 크기 20

▪ 테이블 명:wh_comm

* 사용자의 클라이언트 프로그램과 저온저장고를 제어하는 컨트롤러 간 의 통신 프로토콜 파라미터 저장

* 클라이언트 프로그램은 처음 기동 시에 서버에서 이 통신 파라미터를 전송 받는다.

* field 설명

- IND
  + 통신 파라미터의 index
  + 데이터 타입 tinyint
  + 크기 3
- ID
  + 통신 파라미터
  + 데이터 타입 char
  + 크기 3
- DES
  + 통신 파라미터의 설명
  + 데이터 타입 varchar
  + 크기 30

▪ 테이블 명: wh_curr_value

* 데이터의 효율적 처리를 위하여 현재의 온도, 습도, 상태 값을 한 곳에 저장

* field 설명

- wh_cd
  + 저장고 번호

+ 데이터 타입 varchar
+ 크기 5

- wh_temp
+ 저장고의 현재 온도
+ 데이터 타입 float

- wh_humid
+ 저장고의 현재 습도
+ 데이터 타입 float

- wh_state
+ 저장고의 현재 상태 값
+ 데이터 타입 varchar
+ 크기 10

▪ 테이블 명: wh_humid

* 저장고의 현재 습도 값을 저장한다.

* field 설명

- WH_CD
+ 저온저장고 번호
+ 데이터 타입 varchar
+ 데이터 크기 5

- YYYYMMDD
+ 습도값이 저장된 연 월 일
+ 데이터 타입 varchar
+ 크기 8

- HH
+ 습도값이 저장된 시
+ 데이터 타입 char
+ 크기 2

- mm
+ 습도값이 저장된 분

+ 데이터 타입 char

+ 데이터 크기 2

- ss

+ 습도값이 저장된 초

+ 데이터 타입 char

+ 크기 2

- CURR_VALUE

+ 현재 습도값

+ 데이터 타입 float

▪ 테이블 명: wh_temp

* 저장고의 현재 온도 값을 저장한다.

* field 설명

- WH_CD

+ 저온저장고 번호

+ 데이터 타입 varchar

+ 데이터 크기 5

- YYYYMMDD

+ 온도 값이 저장된 연 월 일

+ 데이터 타입 varchar

+ 크기 8

- HH

+ 온도 값이 저장된 시

+ 데이터 타입 char

+ 크기 2

- mm

+ 온도 값이 저장된 분

+ 데이터 타입 char

+ 데이터 크기 2

- ss

+ 온도 값이 저장된 초
+ 데이터 타입 char
+ 크기 2

- CURR_VALUE
+ 현재 온도 값
+ 데이터 타입 float

▪ 테이블 명: wh_info

* 저온저장고 관리를 위한 데이터의 저장

* field 설명

- WH_CD
+ 저온저장고 번호
+ 데이터 타입 varchar
+ 데이터 크기 5

- WH_KD
+ 관리 코드
+ 데이터 타입 tinyint
+ 크기 2

- SET_VALUE
+ 설정 값
+ 데이터 타입 tinyint
+ 크기 3

▪ 테이블 명: wh_kd_info

* 저온저장고 관리코드에 대한 설명

* field 설명

- WH_KD
+ 저온저장고 관리코드
+ 데이터 타입 tinyint
+ 크기 2

- WH_KD_DESC
  + 저온저장고 관리코드 설명
  + 데이터 타입 varchar
  + 크기 30

▪ 테이블 명: wh_location

* 지역코드 관리

* field 설명

- wh_cd
  + 지역코드
  + 데이터 타입 char
  + 크기 5
- wh_loc
  + 지역
  + 데이터 타입 char
  + 크기 50

▪ 테이블 명: wh_production

* 저온저장고에 관리 되어지는 품목 코드에 대한 관리

* field 설명

- wh_cd
  + 품목 코드
  + 데이터 타입 char
  + 크기 5
- wh_prod
  + 품목
  + 데이터 타입 char
  + 크기 30

▪ 테이블 명: wh_state

* 저온저장고의 상태값 저장
* field 설명
  - wh_cd
    + 저온저장고 번호
    + 데이터 타입 varchar
    + 크기 5
  - yyyymmdd
    + 상태 값이 저장되어진 연 월 일
    + 데이터 타입 varchar
    + 크기 8
  - hh
    + 상태 값이 저장되어진 시
    + 데이터 타입 char
    + 크기 2
  - mm
    + 상태 값이 저장되어진 분
    + 데이터 타입 char
    + 크기 2
  - ss
    + 상태 값이 저장되어진 초
    + 데이터 타입 char
    + 크기 2
  - curr_value
    + 현재 상태 값
    + 데이터 타입 varchar
    + 크기 10

▪ 테이블 명: wh_user_info
* 저온저장고 소유주 정보
* field 설명

- ID
  + 소유주 ID
  + 데이터 타입 varchar
  + 크기 10
- PW
  + 비밀번호
  + 데이터 타입 varchar
  + 크기 10
- UPYMD
  +변경 연 월 일
  + 데이터 타입 varchar
  + 크기 8
- user_lo_cd
  + 거주지역 코드
  + 데이터 타입 char
  + 크기 3
- wh_lo_cd
  + 저장고 위치 코드
  + 데이터 타입 char
  + 크기 3
- reg_num
  + 주민등록번호
  + 데이터 타입 varchar
  + 크기 14
- hp_num
  + 무선전화 번호
  + 데이터 타입 varchar
  + 크기 12
- home_num
  + 집전화 번호

+ 데이터 타입 varchar
+ 크기 13

- email
+ 전자메일 주소
+ 데이터 타입 varchar
+ 크기 20

- admin_yn
+ 관리자 여부
+ 데이터 타입 char
+ 크기 1

- name
+ 성명
+ 데이터 타입 varchar
+ 크기 20

▪ 테이블 명: wh_wh_info

* 저온저장고에 대한 일반적 정보

* field 설명

- wh_cd
+ 저온저장고 번호
+ 데이터 타입 varchar
+ 크기 5

- usr_id
+ 소유주 ID
+ 데이터 타입 varchar
+ 크기 10

- wh_location
+ 저온저장고 위치
+ 데이터 타입 varchar
+ 크기 30

- wh_product
  + 저온저장고에 저장되어진 품목 코드
  + 데이터 타입 varchar
  + 크기 30
- wh_am
  + 수량
  + 데이터 타입 tinyint
  + 크기 3
- wh_use
  + 저장고의 사용 여부
  + 데이터 타입 char
  + 크기 1

▪ 테이블 명: user_location

* 소유주의 거주 지역 코드 관리

* field 설명

- wh_cd
  + 지역 코드
  + 데이터 타입 char
  + 크기 5
- wh_loc
  + 지역
  + 데이터 타입 char
  + 크기 50

▪ 테이블 명: free

* 자유게시판 관리

* field 설명

- no
  + 게시물 번호

+ 데이터 타입 int

+ 크기 10

- name

+ 이름

+ 데이터 타입 varchar

+ 크기 20

- email

+게시자 전자메일 주소

+ 데이터 타입 varchar

+ 크기 70

- title

+ 게시물 제목

+ 데이터 타입 varchar

+ 크기 100

- re_date

+ 재 게시 날짜

+ 데이터 타입 varchar

+ 크기 30

- passwd

+ 비밀번호

+ 데이터 타입 varchar

+ 크기 30

- contents

+ 내용

+ 데이터 타입 text

▪ 테이블 명: notice

* 공지사항 관리

* field 설명

- no

+ 게시물 번호
+ 데이터 타입 int
+ 크기 10

- name
+ 이름
+ 데이터 타입 varchar
+ 크기 20

- email
+게시자 전자메일 주소
+ 데이터 타입 varchar
+ 크기 70

- title
+ 게시물 제목
+ 데이터 타입 varchar
+ 크기 100

- re_date
+ 재 게시 날짜
+ 데이터 타입 varchar
+ 크기 30

- passwd
+ 비밀번호
+ 데이터 타입 varchar
+ 크기 30

- contents
+ 내용
+ 데이터 타입 text

▪ 테이블 명: qna

* Q&A 관리

* field 설명

- no
  + 게시물 번호
  + 데이터 타입 int
  + 크기 10
- name
  + 이름
  + 데이터 타입 varchar
  + 크기 20
- email
  +게시자 전자메일 주소
  + 데이터 타입 varchar
  + 크기 70
- title
  + 게시물 제목
  + 데이터 타입 varchar
  + 크기 100
- re_date
  + 재 게시 날짜
  + 데이터 타입 varchar
  + 크기 30
- passwd
  + 비밀 번호
  + 데이터 타입 varchar
  + 크기 30
- contents
  + 내용
  + 데이터 타입 text

다. 데이터베이스 구축

본 연구에서는 데이터의 관리, 유지 및 사용을 위하여 DBMS를 사용하였다. DBMS로는 현재 무료로 배포되고 있으면서도 성능에 있어 여러 가지 장점을 가지고 있는 MySQL을 사용하였다.

1) MySQL

가) 빠른 응답속도

MySQL은 초기의 개발 목표인 가볍고 '빠른 속도의 데이터베이스 추구'에서 보여주듯이 타 DBMS 보다 빠른 처리속도를 보여 주고 있다. 그림 51은 온라인 벤치마킹 싸이트인 Eweek(www.eweek.com)에서 제시한 벤치마킹 자료로 MySQL이 타 DBMS에 비해 우수한 성능을 보여줌을 알 수 있다.

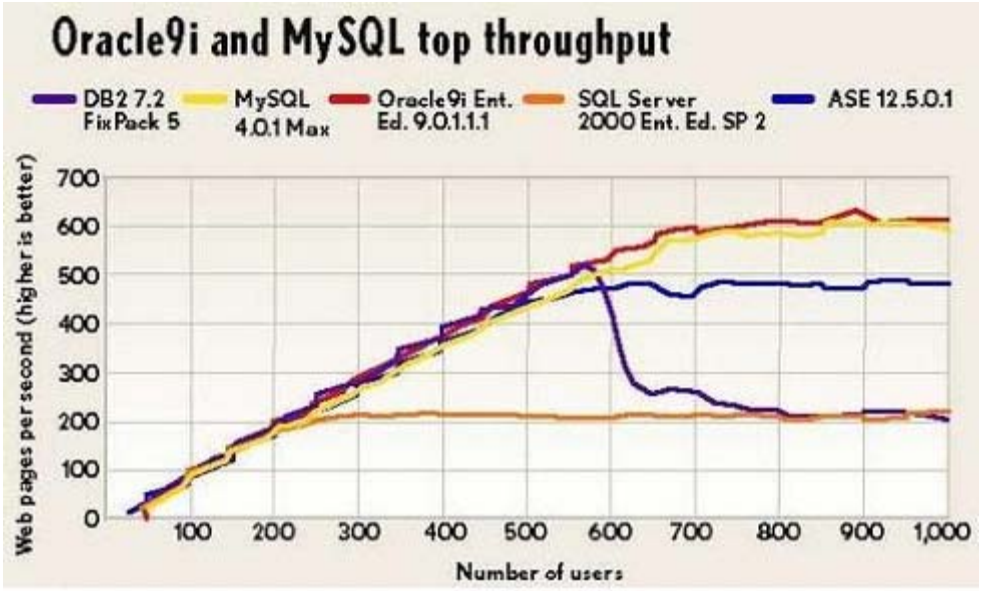

그림 51 MySQL 과 타 DBMS와 Throughput 비교 출처: Eweek

나) 안정적인 데이터베이스

MySQL은 이미 많은 사용자로부터 속도는 물론 안정성까지 입증 및 확인되어져 왔다. MySQL은 설치 및 관리가 쉽고, 작은 용량에, 리소스를 많이 사용하

지 않으며, 속도가 빠르고, 공개된 프로그램이라는 많은 장점에도 불구하고 안정성 검증을 이유로 주로 소규모 시스템에서 웹과의 연동으로 밖에 사용되지 못했다. 그러나 MySQL4.0은 많은 기능이 추가되어 MySQL의 단점 중 하나였던 SQL의 기능 부재를 극복하여, 부하가 아주 높은 DB에서도 충분히 사용될 수 있도록 개발되고 있으므로 안정성은 더 이상 문제가 되지 않는다.

다) 대용량 데이터의 안정적 처리

MySQL은 한 테이블 당 OS에서 지원하는 하나의 파일이 지원하는 최대 크기까지 데이터를 저장할 수 있다. MySQL은 테이블의 데이터를 실제 OS 파일에 저장하기 때문에 하나의 테이블에 저장할 수 있는 데이터의 최대 크기는 OS에서 지원하는 최대 파일 크기가 된다. SunOS에서는 하나의 테이블에 4GB를 저장할 수 있고, 커널 2.2의 리눅스에서는 2GB, 커널 2.4의 리눅스에서는 파일 시스템에 따라 조금씩 다르다. 어떤 경우라 해도 하나의 테이블 당 2GB 또는 4GB이므로 매우 큰 용량으로 실제로 MySQL 개발팀에서는 50,000,000 레코드를 가진 데이터베이스도 다루고 있으며, MySQL 메일링 리스트에는 수백만 레코드를 가진 테이블에서도 속도에 관계없이 안정적으로 운용되어지고 있다. 최대 파일크기를 넘는 데이터에 대해서는 MySQL에서 지원하는 RAID기법을 활용하여 여러 파일에 나누어 데이터를 저장할 수 있으므로, 저장 장치가 지원하는 한 거의 무한대의 자료를 저장할 수 있다.

라) 적은 시스템 리소스 사용

2001년 12월부터 MySQL에서는 다른 DBMS에서 지원하는 SUB-SELECT, SELECT INTO TABLE, 트랜잭션, 저장 프로시져(Stored Procedure), 트리거(Trigger), 외부 키(Foreign Key), 뷰(View) 등의 기능을 제공하지 않는다. 이러한 기능은 전반적으로 시스템을 느리게 할뿐만 아니라 구현을 위해서는 높은 시스템 성능을 요구한다. MySQL 개발의 목표 중 하나가 '가볍고 빠른 속도의 데이터베이스 추구'이다. 따라서 속도를 느리게 하는 요소는 과감하게 포기하게 되었다.

마) 우수한 보안성

MySQL은 보안성이 뛰어날 뿐만 아니라 보안 정책이 명확하다. 리눅스의 사용자 계정을 MySQL의 DB에 저장하려는 연구도 진행 중이고, 이미 어느 정도 실제 활용할 수 있는 단계에 와있는 상태이다. 즉 etc/passwd 파일에 사용자 계정 정보가 저장되는데 계정이 수십만 개가 되는 경우 속도가 느려진다는 것이다. 이러한 정보를 MySQL에 저장해 속도를 빠르게 하고 보안성을 높이려는 연구 역시 수행되었다.

이와 같은 많은 장점을 가지고 있으나 다양한 업무요건이 요구 되어지고 복잡한 업무 로직(Logic)이 요구되어지는 Enterprise환경에서는 사용되어지지 않고 있다. 그것은 질의(Query)에서 사용되어지는 고기능 함수의 부재와 다양한 트랜젝션(Transaction) 관리의 부재, 데이터 백업 관리, 데이터 복구 관리 등의 취약성과 같은 여러 가지 요인 때문이다.

본 연구에서는 데이터의 안정적인 저장과 빠른 응답속도가 요구 되어지나 복잡하지 않는 질의(Query)를 사용하기 때문에 MySQL로서 데이터를 충분히 처리할 수 있었다. 또한 Table의 설계에 있어서도 Query의 복잡성을 최소화 하는 방향으로 설계하였다.

2) DBMS 구성

가) 그림 52는 MySQL 설정 후 관리자 화면을 나타내고 있다. 관리자 화면에서는 사용자의 설정 초기변수 DBMS 서버의 구성 등 DBMS의 전반적인 사항을 관장한다. 본 연구에서는 DBMS 서버를 Web Application Server와 같이 사용하였으며 관리자 ID와 프로그램 상에서 사용하는 사용자 ID 등 두개의 ID를 등록하여 사용하였다.

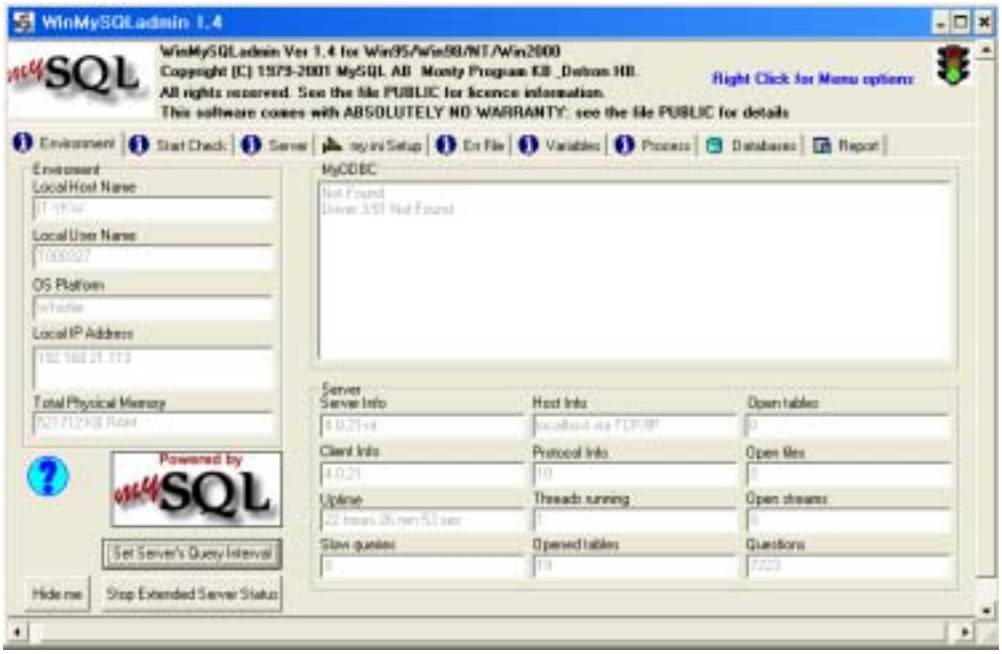

그림 52 MySQL 관리자 화면 설정

나) 초기화일 설정

초기화일에는 사용자 ID와 비밀번호 그리고 서버 IP, 통신을 위한 포트, DBMS에서 사용하는 디렉토리 등을 지정하도록 되어있다. MySQL DBMS 사용을 위한 초기화일을 다음과 같이 설정하였다.

```
#This File was made using the WinMySQLAdmin 1.4 Tool
#2003-10-01 오전 11:09:53

#Uncomment or Add only the keys that you know how works.
#Read the MySQL Manual for instructions

[mysqld]
basedir=C:/mysql
#bind-address=192.168.21.112
datadir=C:/mysql/data
#language=C:/mysql/share/your language directory
#slow query log#=
```

```
#tmpdir#=
port=3306
#set-variable=key_buffer=16M
[WinMySQLadmin]
Server=C:/mysql/bin/mysqld-nt.exe
user=admin
password=admin
QueryInterval=10
```

9. Client와 Server간 프로토콜

본 연구에서 클라이언트 프로그램과 서버간의 통신을 위하여 Webservice를 이용하였다. Webservice 통신의 기본은 SOAP(Simple Object Access Protocol)으로 이는 HTTP를 기본 프로토콜로 사용한다. HTTP는 반 이중방식의 통신이 이루어 져야 한다. 서버에서는 클라이언트에서 보내온 데이터를 처리하기 위해서 데이터 프레임을 정의하여야 한다. 본 연구에서는 반 이중방식을 채택하였음으로 클라이언트 프로그램에서 서버에 데이터를 전송할 때와 서버에서 내려진 명령을 가져가는 부분이 서로 독립적으로 존재한다. 클라이언트는 정의되어진 주기 마다 서버에 접속하여 자신의 현재 데이터를 서버에 전송하고 또한 서버에서 처리되어진 명령 데이터 중 자신에게 할당되어진 값을 가져와서 이를 다시 컨트롤러에 전송하게 된다.

가. Server와 Client간 통신 프로토콜 정의

1) 클라이언트 프로그램과 컨트롤러 간의 통신 규약

클라이언트 프로그램과 컨트롤러 간의 통신은 RS-232를 이용한 아스키코드 기반 통신이 이루어진다. 전송방식은 서버에서 클라이언트 기기에 데이터를 요청하면 클라이언트 기기는 각각의 커맨드에 맞도록 데이터를 반송해 주거나 전송되어진 데이터를 통해 정해진 커맨드를 수행한다. 데이터의 구성은 전체 데이터가 Start Byte를 시작으로 2Byte의 ID, Function 정의를 위한 2Byte, 데이터를

위한 4Byte 그리고 parity Byte와 End Byte로 총 11바이트로 구성 되어져있다. 통신프로토콜의 데이터 구성은 다음과 같다.

표 22 클라이언트와 컨트롤러간의 통신 프로토콜 데이터 구성(요청)

Start Byte	ID		Function		Data				P	End Byte
1	2	3	4	5	6	7	8	9	10	11
S	From	To	1	2	1	2	3	4	1	E

표 23 클라이언트와 컨트롤러간의 통신 프로토콜 데이터 구성(반환)

Start Byte	ID		Function		Data				P	End Byte
1	2	3	4	5	6	7	8	9	10	11
R	From	To	1	2	1	2	3	4	1	E

■ StartByte(1) : 제정한 프로토콜의 시작을 나타낸다.(S: 클라이언트 프로그램에서 컨트롤러에 데이터를 요청, R: 컨트롤러에서 클라이언트 프로그램으로 반환되어지는 데이터)

■ ID(2) : 데이터를 보내고 받는 컨트롤러의 내부적인 ID.

■ Function(2) : 데이터를 통해 수행되는 기능(온도, 습도)으로 각각의 기능은 숫자로 정의되어 있으며 크게 요청기능, 제어기능, 리턴기능으로 구분되어진다.

■ Data(4) : 기능에 맞는 데이터(온도 값, 습도 값), 클라이언트 프로그램에서 컨트롤러에 데이터를 요청할 경우에는 사용하지 않는다.

■ P(1) : 전송되어지는 전체 프로토콜의 에러 체크를 위한 패리티 바이트

■ EndByte(1) : 제정한 프로토콜의 끝을 나타낸다.

* 프로토콜 사용의 예

- 1번 서버에서 2번 클라이언트로 전체 온도데이터를 요청할 경우 (Return)

- 전체 온도 데이터를 요청 < - > 현재 전체 온도 데이터 반송 (3도일 경우)
- 전송 프로토콜의 구성: S121100005E <-> R211100038E

2) 클라이언트프로그램과 컨트롤러간의 데이터 요청/리턴을 위한 기능정의

클라이언트 프로그램에서 컨트롤러에 데이터 요청 시 컨트롤러에서 클라이언트 프로그램으로 데이터를 전송한다.

표 24 컨트롤러에서 클라이언트 프로그램간의 기능 정의

Function		설 명
1	1	현재 전체 온도를 요청하는 부분 (첫 번째 데이터가 0이면 상온이고, 1이면 영하이다)
1	2	첫 번째 온도계의 온도를 요청하는 부분 (첫 번째 데이터가 0이면 상온이고, 1이면 영하이다)
1	3	두 번째 온도계의 온도를 요청하는 부분 (첫 번째 데이터가 0이면 상온이고, 1이면 영하이다)
1	4	현재 전체 습도를 요청하는 부분
1	5	습도계의 요청하는 부분
1	6	현재 설정된 온도를 요청하는 부분
1	7	현재 설정된 습도를 요청하는 부분
2	0	현재 설정된 가동 시간(시)을 요청하는 부분
2	1	현재 설정된 가동 시간(분)을 요청하는 부분
2	2	현재 설정된 제상 시간(분)을 요청하는 부분
2	3	현재 설정된 딜레이 시간(분)을 요청하는 부분
5	0	현재 보드의 상태 값을 전송하는 부분 (압축기, 증발기, 응축기, 제상히터, 가습기 등)
5	2	기기 시스템의 이상 유무를 요청하는 부분 (과전류 및 시스템 이상)

3) 클라이언트 프로그램과 컨트롤러간의 제어 기능의 정의

표 25는 클라이언트 프로그램이 컨트롤러의 데이터를 변경하거나 특정 명령을 전송할 경우 사용되어지는 Function의 정의이다.

표 25 제어 명령 기능 정의

Function		설명
3	0	전체 온도를 설정하는 부분 (첫 번째 데이터가 0이면 상온이고, 1이면 영하이다)
3	1	전체 습도를 설정하는 부분
3	2	제상 주기(시)를 설정하는 부분
3	3	제상 주기(분)를 설정하는 부분
3	4	제상 시간(분)를 설정하는 부분
3	5	딜레이 시간(분)를 설정하는 부분
5	1	긴급 제상으로 제어하는 부분

4) 클라이언트 프로그램과 서버간의 통신 프로토콜

본 연구에서는 클라이언트 프로그램과 서버간의 통신을 위해 Webservice의 기반이 되는 SOAP(Simple Object Access Protocol)을 사용하였다. 이를 통하여 Web 기반의 프로그램뿐 아니라 Application 기반의 프로그램도 자유로이 통신을 할 수 있도록 하였다. 서버프로그램은 JAVA를 기반으로 하였으며 J2EE(Java 2 Enterprise Edition)의 EJB(Enterprise Java Bean)를 기반으로 Webservice를 담당하는 부분은 EAR(Enterprise Achaive)로 하여 서버에 배포(deploy)하였고, 이를 통해 클라이언트 프로그램과 서버 프로그램간의 안정적인 통신을 보장하였다.

5) 저온저장고의 데이터를 가져오는 서비스

저온저장고의 데이터를 가져오는 서비스는 다음과 같은 함수가 정의 되어져 있다.

■ public void saveData(String tableNM, String cd, String ymd, String h, String m, String s, String value)

이 함수는 클라이언트 프로그램에서 DB에 온도와 습도의 데이터를 저장할 때 사용한다. tableNM변수는 온도를 위한 것인지 습도를 위한 것인지를 나타내며, 온도와 습도가 별도의 테이블을 사용한다. cd는 저온저장고 번호이며, ymd는 데이터가 전송되어진 연 월 일을 나타내며, h는 데이터가 전송되어진

시간, m은 데이터가 전송되어진 분, s는 데이터가 전송되어진 초, value는 전송되어진 데이터 값을 나타낸다.

■ public void saveState(String cd, String ymd, String h, String m, String s, String value)

이 함수는 저온저장고의 상태 값을 저장하는 함수이다. cd는 저온저장고 번호이며, ymd는 데이터가 전송되어진 연 월 일을 나타내며, h는 데이터가 전송되어진 시간, m은 데이터가 전송되어진 분, s는 데이터가 전송되어진 초, value는 전송되어진 데이터 값을 나타낸다.

■ public String[][] selectData()

이 함수는 서버에서 정의 되어진 통신 값을 클라이언트 프로그램에서 전송받는 부분이다. 통신 코드값과 통신 값을 전송 받기 위해 2차원 배열로 정의하였다.

■ public String[][] getBoundSel(String whcd)

이 함수는 서버에 저장되어진 각 저온저장고의 온도와 습도의 경계 값을 클라이언트 프로그램에서 가져가는 부분이다. 코드값과 경계 값을 위해 2차원 배열로 리턴 값이 정의 되어져 있다. whcd는 저온저장고 번호를 나타낸다.

■ public boolean getAuthSel(String id,String pw)

이 함수는 사용자 인증을 위한 함수이다. 각각의 클라이언트 프로그램이 기동되어질 때 먼저의 서버를 통해 인증을 받은 후 다음을 시행 한다. id는 사용자 id를 나타내며 pw은 비밀번호를 나타낸다.

■ public void setWhInfo(int value,String whCd,String whKd)

이 함수는 각각의 저온저장고의 설정 값을 저장하는 부분이다. 각각의 저온저장고의 설정값이 변경되어졌을 경우 이 함수를 통해 서버에 변경된 값을 전송한다. value는 변경되어진 값, whCd는 저온저장고 번호, whKd는 변경되어진 설정값 코드를 나타낸다.

■ public int getInfoData(String whCd,String whKd)

각각의 저온저장고의 설정값을 클라이언트 프로그램이 서버로부터 전송 받을 때 사용하는 함수이다. whCd는 저온저장고의 번호를 나타내고 whKd는 설정 파라미터의 코드를 나타내며 리턴 값은 해당 저온저장고의 해당 설정 파라미터 코드에 대한 설정 값이 전송되어진다.

6) 제어 데이터 전송 서비스

본 연구에서의 제어 데이터는 관리자에 의해 제어 데이터가 서버를 통해 데이터베이스에 저장되어지며 클라이언트 프로그램이 서버에 저온저장고의 모니터링 데이터를 저장하기 위해 통신 요구가 이루어지면 자신에 해당하는 제어 데이터를 가져간다.

■ public void setCmd(String cd, String cmd,String ymd, String hms, String value)

이 함수는 제어 데이터를 설정하는 함수이다. cd는 저온저장고 번호, cmd는 명령 종류, hms는 명령이 내려진 시 분 초를 나타내며 value는 제어 값을 나타낸다.

■ public String[][] getCmd(String whcd)

이 함수는 클라이언트 프로그램에서 명령 값을 가져가는 함수이다. 클라이언트 프로그램은 서버로부터 전체 명령을 같이 가지고 있다. 서버에서 클라이언트에 보내는 명령은 총 6가지가 있다. whcd는 저온저장고의 번호를 나타내며 이 번호를 이용하여 6가지의 명령을 동시에 보낸다. 표 26에 명령을 정의하였다.

표 26 Command 설명

wh_cmd_cd	wh_cmd
01	temp_command
02	humid_command
03	defrost_command
04	deice start time
05	deice holding time
06	system stoptime

■ public void delCmd(String whcd)

이 함수는 각각의 저온저장고의 명령을 지우는 함수이다. 본 연구에서는 클라이언트 프로그램에 전송되어지는 명령어는 명령이 지워지는 즉시 지워지도록 하였다. 이를 통해 클라이언트 프로그램은 자신에게 전송되어질 명령이 있는

지를 파악하고 있을 경우만 명령을 가지고 간다. whcd는 저온저장고의 번호를 나타낸다.

나. 통신 프로토콜 구현

통신 프로토콜의 구현을 위해 WSDL(Web Service Definition Language)을 정의하였다. 이를 통하여 서버와 클라이언트 프로그램간의 SOAP(Simple Object Access Protocol)을 통한 통신을 가능하도록 하였다. 기본적인 통신을 위한 프로토콜은 HTTP(Hyper Text Transfer Protocol)를 따르며 전송되어지는 데이터의 포맷은 XML(Extensible Markup Language)을 통해 정의되어 진다. 이를 통해 클라이언트 프로그램과 서버간의 원격객체 통신이 가능해진다.

1) SOAP의 특징

SOAP은 다른 분산객체 시스템에 비해 다음과 같은 장점을 지니고 있다.

■ SOAP은 방화벽을 통해서 쉽게 전달될 수 있다.

■ SOAP 데이터는 XML을 사용해서 구조화되어 있다.

■ SOAP은 잠재적으로 HTTP, SMTP 및 JMS 같은 여러 전송 프로토콜과 함께 사용될 수 있다.

■ SOAP은 가볍다.

■ Microsoft, IBM, SUN을 포함한 많은 벤더들이 지원한다.

SOAP의 주된 장점 중 하나는 SOAP 어플리케이션은 HTTP를 전송 프로토콜로 사용할 수 있기 때문에 방화벽을 쉽게 통과할 수 있다는 점이다. 따라서 한번 설치되면 내부적으로든(인트라넷) 외부적으로든(인터넷) SOAP 어플리케이션이 쉽게 접근할 수 있다. 그러나 이런 속성은 허가되지 않은 사용자도 SOAP 어플리케이션에 접근할 수 있게 되므로 매우 심각한 보안 문제가 될 수도 있다.

2) SOAP의 단점

SOAP의 단점은 다음과 같다.

■ SOAP 툴킷간의 상호운영성 문제: SOAP이 많은 벤더의 지원을 받고 있기는 하지만, 서로 다른 SOAP 구현 간에는 여전히 상호운용성 문제가 존재한

다.

■ 보안 메커니즘의 미성숙: SOAP에는 메시지를 처리하기 전에 인증하기 위한 메커니즘이 정의되어 있지 않다. 또한 메시지의 내용을 다른 사용자가 알아볼 수 없도록 내용을 암호화하는 메커니즘도 존재하지 않는다.

■ 확실한 메시지 전송을 보장하지 못함: 메시지를 보내는 시스템이 다운된다면 SOAP 시스템은 메시지를 어떻게 다시 보내야 할지 모른다.

■ 공개 및 구독 모델 지원의 부재: SOAP 클라이언트가 여러 서버에 요청을 하려면 반드시 요청하려는 모든 서버에 직접 요청을 보내야 한다.

3) SOAP 통신 구조

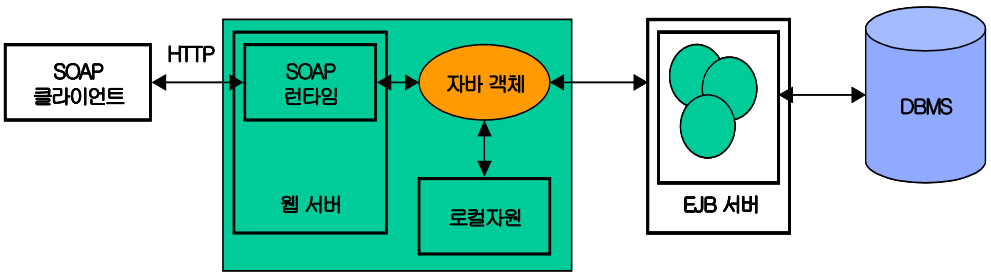

그림 53 SOAP 통신 구조

4) WSDL의 정의

SOAP통신을 위해서는 XML로 되어진 WSDL 파일이 필요하다. 저온저장고의 데이터를 가져오는 서비스의 WSDL과 제어 데이터 전송 서비스를 위한 WSDL를 다음과 같이 정의하였다.

■ 저온저장고의 데이터를 가져오는 서비스의 WSDL

표 27 저온저장고의 데이터를 가져오는 서비스의 WSDL

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions xmlns: tns=" ttp://www.acai.com/servers/wls/samples
/examples/webservices/basic/javaclass"
 xmlns:wsr="http://www.openuri.org/2002/10/soap/reliability/"
```

```
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
 xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
 xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:soap12enc="http://www.w3.org/2003/05/soap-encoding"
 xmlns:conv="http://www.openuri.org/2002/04/wsdl/conversation/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.acai.com/servers/wls/samples/examples
/webservices/basic/javaclass">
- <types>
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:stns="java:language_builtins.lang"
 elementFormDefault="qualified"
 attributeFormDefault="qualified"
 targetNamespace="java:language_builtins.lang">
  <xsd:import namespace=
                    "http://schemas.xmlsoap.org/soap/encoding/" />
- <xsd:complexType name="ArrayOfArrayOfString">
- <xsd:complexContent>
- <xsd:restriction xmlns:soapenc="
                    http://schemas.xmlsoap.org/soap/encoding/"
                  base="soapenc:Array">
  <xsd:attribute xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                 ref="soapenc:arrayType" wsdl : arrayType="
                 xsd:string[,]" />
```

```
  </xsd:restriction>
  </xsd:complexContent>
  </xsd:complexType>
  </xsd:schema>
  </types>
- <message name="getBoundSel">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string" />
  </message>
- <message name="getBoundSelResponse">
  <part xmlns:partns="java:language_builtins.lang"
            type="partns:ArrayOfArrayOfString" name="result" />
  </message>
- <message name="getAuthSel">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string0" />
  </message>
- <message name="getAuthSelResponse">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:boolean" name="result" />
  </message>
  <message name="selectData" />
- <message name="selectDataResponse">
  <part xmlns:partns="java:language_builtins.lang"
            type="partns:ArrayOfArrayOfString" name="result" />
```

```
  </message>
- <message name="setWhInfo">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:int" name="intVal" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string0" />
  </message>
  <message name="setWhInfoResponse" />
- <message name="saveState">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string0" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string1" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string2" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string3" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string4" />
  </message>
  <message name="saveStateResponse" />
- <message name="saveData">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
```

```
                type="partns:string" name="string" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string0" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string1" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string2" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string3" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string4" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string5" />
  </message>
  <message name="saveDataResponse" />
- <message name="getInfoData">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string0" />
  </message>
- <message name="getInfoDataResponse">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:int" name="result" />
  </message>
- <portType name="divWSPort">
- <operation name="getBoundSel">
```

```
  <input message="tns:getBoundSel" />
  <output message="tns:getBoundSelResponse" />
  </operation>
- <operation name="getAuthSel">
  <input message="tns:getAuthSel" />
  <output message="tns:getAuthSelResponse" />
  </operation>
- <operation name="selectData">
  <input message="tns:selectData" />
  <output message="tns:selectDataResponse" />
  </operation>
- <operation name="setWhInfo">
  <input message="tns:setWhInfo" />
  <output message="tns:setWhInfoResponse" />
  </operation>
- <operation name="saveState">
  <input message="tns:saveState" />
  <output message="tns:saveStateResponse" />
  </operation>
- <operation name="saveData">
  <input message="tns:saveData" />
  <output message="tns:saveDataResponse" />
  </operation>
- <operation name="getInfoData">
  <input message="tns:getInfoData" />
  <output message="tns:getInfoDataResponse" />
  </operation>
```

```
  </portType>
- <binding type="tns:divWSPort" name="divWSPort">
  <soap:binding style="rpc"  transport=
"http://schemas.xmlsoap.org/soap/http" />
- <operation name="getBoundSel">
  <soap:operation style="rpc" soapAction="" />
  <wsr:reliability persistDuration="60000" />
- <input>
  <soap:body namespace  =  "http://www.acai.com/servers/wls
                    /samples/examples/webservices/basic/javaclass"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </input>
- <output>
  <soap:body namespace   =  "http://www.acai.com/servers/wls
                    /samples/examples/webservices/basic/javaclass"
     encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </output>
  </operation>
- <operation name="getAuthSel">
  <soap:operation style="rpc" soapAction="" />
  <wsr:reliability persistDuration="60000" />
- <input>
  <soap:body namespace= " http://www.acai.com/servers/wls
                    /samples/examples/webservices/basic/javaclass"
     encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
use="encoded" />
  </input>
- <output>
  <soap:body namespace = "http://www.acai.com/servers/wls
                    /samples/examples/webservices/basic/javaclass"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </output>
  </operation>
- <operation name="selectData">
  <soap:operation style="rpc" soapAction="" />
  <wsr:reliability persistDuration="60000" />
- <input>
  <soap:body namespace = "http://www.acai.com/servers/wls
                    /samples/examples/webservices/basic/javaclass"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </input>
- <output>
  <soap:body namespace = "http://www.acai.com/servers/wls
                    /samples/examples/webservices/basic/javaclass"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </output>
  </operation>
- <operation name="setWhInfo">
  <soap:operation style="rpc" soapAction="" />
```

```
  <wsr:reliability persistDuration="60000" />
- <input>
  <soap:body namespace = "http://www.acai.com/servers/wls
                         /samples/examples/webservices/basic/javaclass"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </input>
- <output>
  <soap:body namespace = "http://www.acai.com/servers/wls
                         /samples/examples/webservices/basic/javaclass"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </output>
  </operation>
- <operation name="saveState">
  <soap:operation style="rpc" soapAction="" />
  <wsr:reliability persistDuration="60000" />
- <input>
  <soap:body namespace = "http://www.acai.com/servers/wls
                         /samples/examples/webservices/basic/javaclass"
       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </input>
- <output>
  <soap:body namespace = "http://www.acai.com/servers/wls
                         /samples/examples/webservices/basic/javaclass"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
use="encoded" />
  </output>
  </operation>
- <operation name="saveData">
  <soap:operation style="rpc" soapAction="" />
  <wsr:reliability persistDuration="60000" />
- <input>
  <soap:body namespace = "http://www.acai.com/servers/wls
                         /samples/examples/webservices/basic/javaclass"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </input>
- <output>
  <soap:body namespace = "http://www.acai.com/servers/wls
                         /samples/examples/webservices/basic/javaclass"
     encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </output>
  </operation>
- <operation name="getInfoData">
  <soap:operation style="rpc" soapAction="" />
  <wsr:reliability persistDuration="60000" />
- <input>
  <soap:body namespace = "http://www.acai.com/servers/wls
                         /samples/examples/webservices/basic/javaclass"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
```

```
  </input>
- <output>
  <soap:body namespace = "http://www.acai.com/servers/wls
                        /samples/examples/webservices/basic/javaclass"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </output>
  </operation>
  </binding>
- <service name="divWS">
- <port name="divWSPort" binding="tns:divWSPort">
  <soap:address location=
                    "http://203.237.111.208:7001/web-services/divWS" />
  </port>
  </service>
  </definitions>
```

■ 제어 데이터 전송 서비스를 위한 WSDL

표 28 제어 데이터 전송 서비스를 위한 WSDL

```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions xmlns: tns= "http://www.acai.com/servers/wls
                        samples/examples/webservices/basic/javaclass"
 xmlns:wsr="http://www.openuri.org/2002/10/soap/reliability/"
```

```
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
 xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
 xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:soap12enc="http://www.w3.org/2003/05/soap-encoding"
xmlns:conv="http://www.openuri.org/2002/04/wsdl/conversation/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.acai.com/servers/wls/samples/examples
/webservices/basic/javaclass">
- <types xmlns:tns =  "http://www.acai.com/servers/wls
                        /samples/examples/webservices/basic/javaclass"
xmlns:wsr="http://www.openuri.org/2002/10/soap/reliability/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soap12enc="http://www.w3.org/2003/05/soap-encoding"
xmlns:conv="http://www.openuri.org/2002/04/wsdl/conversation/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/">
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:stns="java:language_builtins.lang"
                    elementFormDefault="qualified"
attributeFormDefault="qualified"
```

```
                    targetNamespace="java:language_builtins.lang">
  <xsd:import namespace
          ="http://schemas.xmlsoap.org/soap/encoding/" />
- <xsd:complexType name="ArrayOfArrayOfString">
- <xsd:complexContent>
- <xsd:restriction xmlns:soapenc=" http://schemas.xmlsoap.org/soap
/encoding/" base="soapenc:Array">
  <xsd:attribute xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                          ref="soapenc:arrayType"
wsdl:arrayType="xsd:string[,]" />
  </xsd:restriction>
  </xsd:complexContent>
  </xsd:complexType>
  </xsd:schema>
  </types>
- <message name="setCmd">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
          type="partns:string" name="string" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
          type="partns:string" name="string0" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
          type="partns:string" name="string1" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
          type="partns:string" name="string2" />
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
          type="partns:string" name="string3" />
  </message>
```

```
  <message name="setCmdResponse" />
- <message name="getCmd">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string" />
  </message>
- <message name="getCmdResponse">
  <part xmlns:partns="java:language_builtins.lang"
                type="partns:ArrayOfArrayOfString" name="result" />
  </message>
- <message name="delCmd">
  <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
                type="partns:string" name="string" />
  </message>
  <message name="delCmdResponse" />
- <portType name="contWSPort">
- <operation name="setCmd">
  <input message="tns:setCmd" />
  <output message="tns:setCmdResponse" />
  </operation>
- <operation name="getCmd">
  <input message="tns:getCmd" />
  <output message="tns:getCmdResponse" />
  </operation>
- <operation name="delCmd">
  <input message="tns:delCmd" />
  <output message="tns:delCmdResponse" />
  </operation>
```

```
  </portType>
- <binding type="tns:contWSPort" name="contWSPort">
  <soap:binding style="rpc"
                transport="http://schemas.xmlsoap.org/soap/http" />
- <operation name="setCmd">
  <soap:operation style="rpc" soapAction="" />
  <wsr:reliability persistDuration="60000" />
- <input>
  <soap:body namespace = "http://www.acai.com/servers/wls
                /samples/examples/webservices/basic/javaclass"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </input>
- <output>
  <soap:body namespace = "http://www.acai.com/servers/wls
                /samples/examples/webservices/basic/javaclass"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </output>
  </operation>
- <operation name="getCmd">
  <soap:operation style="rpc" soapAction="" />
  <wsr:reliability persistDuration="60000" />
- <input>
  <soap:body namespace = "http://www.acai.com/servers/wls
                /samples/examples/webservices/basic/javaclass"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
use="encoded" />
  </input>
- <output>
  <soap:body namespace = "http://www.acai.com/servers/wls
                        /samples/examples/webservices/basic/javaclass"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </output>
  </operation>
- <operation name="delCmd">
  <soap:operation style="rpc" soapAction="" />
  <wsr:reliability persistDuration="60000" />
- <input>
  <soap:body namespace = "http://www.acai.com/servers/wls
                        /samples/examples/webservices/basic/javaclass"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </input>
- <output>
  <soap:body namespace = "http://www.acai.com/servers/wls
                        /samples/examples/webservices/basic/javaclass"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" />
  </output>
  </operation>
  </binding>
- <service name="contWS">
```

```
- <port name="contWSPort" binding="tns:contWSPort">
  <soap:address
   location="http://203.237.111.208:7001/ctrl-service/contWS" />
  </port>  </service>
  </definitions>
```

10. Client용 통신 프로그램 개발

본 연구에서 클라이언트 통신프로그램의 주 역할은 컨트롤러와 서버간의 통신을 중재하고 사용자에게 PC를 통한 저온저장고의 상태를 확인할 수 있도록 하는 것이다. 클라이언트용 프로그램은 Borland사의 Delphi 7을 사용하였다.

가. Client 정보 자료구조 정의

클라이언트의 정보통신 데이터의 구조는 원격 객체를 사용함으로 서버와 클라이언트간의 통신이 웹서비스를 위해 서버에서 정의한 WSDL에 의해 기술되어진 서비스로 이루어진다. 델파이7에서는 WSDL을 통해 SOAP통신을 할 수 있는 클래스를 생성할 수 있다. 본 연구에서 사용되어지는 서비스는 크게 컨트롤을 담당하는 서비스와 데이터의 수집을 담당하는 서비스로 구분되어져 있으며 각각의 서비스에 대해 별도의 unit 화일을 생성하고 메인 프로그램에서 첨부 unit로 설정하여 사용한다. 이 unit 화일은 웹서비스에서 제공하는 메소드에 대한 인터페이스만 정의되어져 있다. 실제로 통신이 이루어지는 부분에 대해서는 델파이7 자체에 내포한 클래스를 사용한다.

다음은 2개의 서비스에 대한 unit 화일이다.

표 29 contWS.pas

```
// ************************************************************ //
// The types declared in this file were generated from data read from the
// WSDL File described below:
```

```
// WSDL        : C:\yikw\webclient3\wsdl\contWS.wsdl
// Encoding : UTF-8
// Version   : 1.0
// (2004-08-28 오후 6:30:54 - 1.33.2.5)
// ************************************************************** //

unit contWS;

interface

uses InvokeRegistry, SOAPHTTPClient, Types, XSBuiltIns,divWS;

type

  //    ************************************************************
//
  // The following types, referred to in the WSDL document are not being
represented
  // in this file. They are either aliases[@] of other types represented or
were referred
  // to but never[!] declared in the document. The types from the latter
category
  // typically map to predefined/known XML or Borland types; however,
they could also
  // indicate incorrect WSDL documents that failed to declare or import a
schema type.
  //    ************************************************************
//
  // !:string             - "http://www.w3.org/2001/XMLSchema"
//   ArrayOfArrayOfString = array of array of WideString;    {
    "java:language_builtins.lang" }
  //    ************************************************************
//
  //Namespace                                                          :
http://www.acai.com/servers/wls/samples/examples/webservices/basic/javaclas
s
```

```
  // transport : http://schemas.xmlsoap.org/soap/http
  // style     : rpc
  // binding   : contWSPort
  // service   : contWS
  // port      : contWSPort
  // URL       : http://203.237.111.208:7001/ctrl-service/contWS
  //   ************************************************************************
//
  contWSPort = interface(IInvokable)
  ['{CE5DAC6F-8BF5-21EC-08E1-A8838FEF72DB}']
    procedure setCmd(const string_: WideString; const string0: WideString;
const string1: WideString; const string2: WideString; const string3:
WideString); stdcall;
    function getCmd(const string_: WideString): ArrayOfArrayOfString;
stdcall;
    procedure delCmd(const string_: WideString); stdcall;
  end;

function GetcontWSPort(UseWSDL: Boolean=System.False; Addr: string='';
HTTPRIO: THTTPRIO = nil): contWSPort;



implementation

function GetcontWSPort(UseWSDL: Boolean; Addr: string; HTTPRIO:
THTTPRIO): contWSPort;
const
  defWSDL = 'C:\yikw\webclient3\wsdl\contWS.wsdl';
  defURL  = 'http://203.237.111.208:7001/ctrl-service/contWS';
  defSvc  = 'contWS';
  defPrt  = 'contWSPort';
var
  RIO: THTTPRIO;
begin
  Result := nil;
  if (Addr = '') then
```

```
  begin
    if UseWSDL then
      Addr := defWSDL
    else
      Addr := defURL;
  end;
  if HTTPRIO = nil then
    RIO := THTTPRIO.Create(nil)
  else
    RIO := HTTPRIO;
  try
    Result := (RIO as contWSPort);
    if UseWSDL then
    begin
      RIO.WSDLLocation := Addr;
      RIO.Service := defSvc;
      RIO.Port := defPrt;
    end else
      RIO.URL := Addr;
  finally
    if (Result = nil) and (HTTPRIO = nil) then
      RIO.Free;
  end;
end;


initialization
  InvRegistry.RegisterInterface(TypeInfo(contWSPort),
'http://www.acai.com/servers/wls/samples/examples/webservices/basic/javacla
ss', 'UTF-8');
  InvRegistry.RegisterDefaultSOAPAction(TypeInfo(contWSPort), '');
  InvRegistry.RegisterExternalParamName(TypeInfo(contWSPort),    'setCmd',
'string_', 'string');
  InvRegistry.RegisterExternalParamName(TypeInfo(contWSPort),    'getCmd',
'string_', 'string');
  InvRegistry.RegisterExternalParamName(TypeInfo(contWSPort),    'delCmd',
```

```
'string_', 'string');
  RemClassRegistry.RegisterXSInfo(TypeInfo(ArrayOfArrayOfString),
'java:language_builtins.lang', 'ArrayOfArrayOfString');

end.
```

표 30 divWS.pas

```
//  ************************************************************************
//
// The types declared in this file were generated from data read from the
// WSDL File described below:
// WSDL       : C:\yikw\webclient3\wsdl\divWS.wsdl
// Encoding : UTF-8
// Version  : 1.0
// (2004-10-10 오후 10:59:58 - 1.33.2.5)
//  ************************************************************************
//

unit divWS;

interface

uses InvokeRegistry, SOAPHTTPClient, Types, XSBuiltIns;

type

  //  ************************************************************************
//
  // The following types, referred to in the WSDL document are not being
represented
  // in this file. They are either aliases[@] of other types represented or
were referred
  // to but never[!] declared in the document. The types from the latter
category
  // typically map to predefined/known XML or Borland types; however,
they could also
```

```
  // indicate incorrect WSDL documents that failed to declare or import a schema type.
  //  ************************************************************************//
  // !:string             - "http://www.w3.org/2001/XMLSchema"
  // !:boolean            - "http://www.w3.org/2001/XMLSchema"
  // !:int                - "http://www.w3.org/2001/XMLSchema"

  ArrayOfArrayOfString = array of array of WideString; { "java:language_builtins.lang" }

  //  ************************************************************************//
  //                                Namespace : http://www.acai.com/servers/wls/samples/examples/webservices/basic/javaclass
  // transport : http://schemas.xmlsoap.org/soap/http
  // style     : rpc
  // binding   : divWSPort
  // service   : divWS
  // port      : divWSPort
  // URL       : http://203.237.111.208:7001/web-services/divWS
  //  ************************************************************************//
  divWSPort = interface(IInvokable)
  ['{9DD1DCAD-2576-DBE0-75F4-992BC256DD39}']
    function getBoundSel(const string_: WideString): ArrayOfArrayOfString; stdcall;
    function getAuthSel(const string_: WideString; const string0: WideString): Boolean; stdcall;
    function selectData: ArrayOfArrayOfString; stdcall;
    procedure setWhInfo(const intVal: Integer; const string_: WideString; const string0: WideString); stdcall;
    procedure saveState(const string_: WideString; const string0: WideString; const string1: WideString; const string2: WideString; const string3: WideString; const string4: WideString); stdcall;
```

```
    procedure saveLog(const string_: WideString; const string0: WideString);
stdcall;
    procedure saveData(const string_: WideString; const string0: WideString;
const string1: WideString; const string2: WideString; const string3:
WideString; const string4: WideString; const string5: WideString); stdcall;
    function getInfoData(const string_: WideString; const string0:
WideString): Integer; stdcall;
  end;

function GetdivWSPort(UseWSDL: Boolean=System.False; Addr: string='';
HTTPRIO: THTTPRIO = nil): divWSPort;



implementation

function GetdivWSPort(UseWSDL: Boolean; Addr: string; HTTPRIO:
THTTPRIO): divWSPort;
const
  defWSDL = 'C:\yikw\webclient3\wsdl\divWS.wsdl';
  defURL  = 'http://203.237.111.208:7001/web-services/divWS';
  defSvc  = 'divWS';
  defPrt  = 'divWSPort';
var
  RIO: THTTPRIO;
begin
  Result := nil;
  if (Addr = '') then
  begin
    if UseWSDL then
      Addr := defWSDL
    else
      Addr := defURL;
  end;
  if HTTPRIO = nil then
    RIO := THTTPRIO.Create(nil)
  else
```

```
    RIO := HTTPRIO;
  try
    Result := (RIO as divWSPort);
    if UseWSDL then
    begin
      RIO.WSDLLocation := Addr;
      RIO.Service := defSvc;
      RIO.Port := defPrt;
    end else
      RIO.URL := Addr;
  finally
    if (Result = nil) and (HTTPRIO = nil) then
      RIO.Free;
  end;
end;


initialization
  InvRegistry.RegisterInterface(TypeInfo(divWSPort),
'http://www.acai.com/servers/wls/samples/examples/webservices/basic/javacla
ss', 'UTF-8');
  InvRegistry.RegisterDefaultSOAPAction(TypeInfo(divWSPort), '');
  InvRegistry.RegisterExternalParamName(TypeInfo(divWSPort),
       'getBoundSel', 'string_', 'string');
  InvRegistry.RegisterExternalParamName(TypeInfo(divWSPort),
       'getAuthSel', 'string_', 'string');
  InvRegistry.RegisterExternalParamName(TypeInfo(divWSPort),
       'setWhInfo', 'string_', 'string');
  InvRegistry.RegisterExternalParamName(TypeInfo(divWSPort),
       'saveState', 'string_', 'string');
  InvRegistry.RegisterExternalParamName(TypeInfo(divWSPort), 'saveLog',
       'string_', 'string');
  InvRegistry.RegisterExternalParamName(TypeInfo(divWSPort), 'saveData',
       'string_', 'string');
  InvRegistry.RegisterExternalParamName(TypeInfo(divWSPort),
       'getInfoData', 'string_', 'string');
```

```
RemClassRegistry.RegisterXSInfo(TypeInfo(ArrayOfArrayOfString),
        'java:language_builtins.lang', 'ArrayOfArrayOfString');

end.
```

나. Client 정보 수집 프로그램 개발

컨트롤러의 데이터를 수집하고 서버에서 내려진 명령을 수행하는 클라이언트 프로그램의 개발을 위해 본 연구 1차년도에 기 개발되어진 클라이언트 프로그램에 서버와의 통신을 위한 프로그램을 개발하여 추가하였다. 또한 이 프로그램은 복잡하지 않으면서도 안정적으로 운영되어지도록 개선하였다.

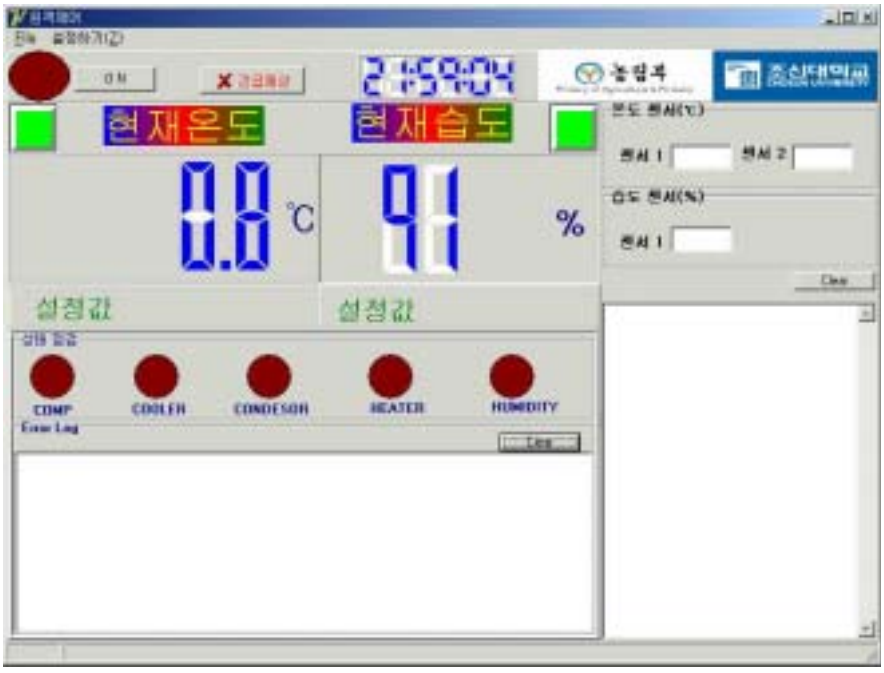

그림 54 클라이언트 프로그램

다. 네트워크 접속 프로그램 개발

클라이언트 프로그램과 서버와의 통신을 위해 SOAP을 이용한 통신이 이루어

지도록 하였다. SOAP은 원격객체 통신을 하며 HTTP 기반 통신이므로 네트워크의 부하를 줄이면서도 안정적인 통신이 가능하다. 본 연구에서는 델파이7의 웹서비스 지원 클래스를 이용하여 통신 프로그램을 제작하였다.

다음 프로그램은 클라이언트 프로그램에서 서버로의 데이터 전송을 하는 부분으로써 기 정의한 WSDL에 의해 생성되어지는 conWS.pas와 divWS.pas 클래스의 인터페이스를 사용하였다. 이는 웹서비스를 사용함으로써 웹기반 프로그램과 C/S 기반 프로그램이 서버 인터페이스를 구성하고 동일한 비즈니스 로직을 사용할 수 있음을 보여주고 있다.

```
procedure TForm1.SaveData(temp:boolean;value:double);
var
    tableNm:String;
    data:divWSPort;
    i:integer;
    dd:ArrayOfArrayOfString;
begin
     data:=GetdivWSPort;

     if not temp then
     begin
          if value > StrToInt(Form2.edit26.Text) then
               Panel17.Color:=clRed
          else if value < StrToInt(Form2.edit27.Text) then
               Panel17.Color:=clBlue
          else Panel17.Color:=clLime;
          tableNm:='wh_humid';
     end
     else
     begin
          if value > StrToInt(Form2.edit24.Text) then
               Panel16.Color:=clRed
          else if value < StrToInt(Form2.edit25.Text) then
               Panel16.Color:=clBlue
          else Panel16.Color:=clLime;
```

```
            tableNm:='wh_temp';
        end;
        data.saveData(tableNm,whcd,FormatDateTime('yyyymmdd',
                                now),FormatDateTime('hh', now),
        FormatDateTime('nn', now),FormatDateTime('ss', now),
                                FloatToStr(value));

end;
```

라. 제어・측정시스템 상태정보 전송 프로그램 개발

클라이언트 프로그램에서 수집되어진 저온저장고의 상태정보와 모니터링 데이터는 주기적으로 서버로 전송되어진다. 클라이언트에서 서버로의 데이터 전송을 위해 클라이언트 프로그램은 우선 컨트롤러와 통신을 통해 데이터를 수집하고 수집한 데이터를 WSDL에 의해 기술되어진 저장함수를 호출하여 서버에 데이터를 저장하고 해당 저장고에 할당되어진 제어명령을 가져와 컨트롤러에 전송한다. 이러한 일련의 절차가 클라이언트 프로그램의 자체 타이머에 의해 동작되어지므로 저온저장고에 대한 데이터의 지연이 발생한다. 이는 저온저장고의 온도와 습도가 급격한 변화가 일어나지 않으므로 수 십초의 시간 지연에서는 시스템이 안정적으로 운영되어진다. 다음은 개발한 상태정보 전송 프로그램 이다.

```
//***
 각각의 저온저장고에 전송되어질 상태값을 저장한다.
 본 연구에서는 서버가 클라이언트에 상태값을 전송하는 것이 아니라
 서버가 상태값을 데이터 베이스에 저장하면 클라이언트는 자신에게
 할당되어진 상태값을 가지고 가도록 되어져있다.
***//

  public void setCmd(String cd, String cmd,String ymd, String hms,
                          String value)
  {
```

```
            Statement stmt=null ;
            ResultSet rst =null;
            String ss,query;
            Connection conn=null;

            try
           {
           ctx = new InitialContext();
           if(ctx == null )        throw new Exception("Boom - No Context");
           ds =  (DataSource)ctx.lookup("MYSQL_JNDI_WH");
             if (ds != null)    conn = ds.getConnection();
           }catch(Exception e) {e.printStackTrace();}
      try {
                stmt = conn.createStatement();
                query="insert into wh_cmd
                        values( '"+ cd+"','"+cmd+"','"
                                +value+"','"+ymd+"','"+hms+")";
                 stmt.executeUpdate( query);
                 stmt.close();
                 conn.close();
           } catch(Exception e){e.printStackTrace(); ss="error";}
}

//**
클라이언트가 서버에서 할당되어진 상태값을 가지고 간다.
**//

public Hashtable getCmd(String whcd)
{
     Statement stmt=null ;
     ResultSet rst =null;
     Hashtable ht;
     Context ctx;
     DataSource ds;
     Connection conn=null;
     ht=new Hashtable();
```

```
String sql;
boolean deiceYN;

try{
        System.out.println("==public Hashtable getBoundSel()==");
        ctx = new InitialContext();
        if(ctx == null )        throw new Exception("Boom - No Context");

        ds =  (DataSource)ctx.lookup("MYSQL_JNDI_WH");
        if (ds != null)    conn = ds.getConnection();
     }catch(Exception e) { System.out.println(e.toString());}
     try{
         stmt = conn.createStatement();
         sql="select wh_cmd_cd,wh_cmd_value from wh_cmd where
                  wh_cd='"+whcd+"' order by wh_cmd_cd";
          rst = stmt.executeQuery(sql );
          while ( rst.next())
          {
              ht.put(rst.getString(1),rst.getString(2));
              if ((rst.getString(1)="03") && (rst.getString(2)<>"-1"))
                    deiceYN=true;
           }
           rst.close();
} catch(Exception e){System.out.println(e.toString());}
try
{
         if(deiceYN)
         {
                    stmt.close();
                    stmt=conn.createStatement();
                    sql="update wh_cmd set wh_cmd_value=-1
                           where wh_cmd_cd='03' and wh_cd='"+whcd+"'";
                    stmt.executeUpdate(sql);
                    System.out.println(sql);
          }
          stmt.close();
```

```
            conn.close();
    }catch(Exception e){System.out.println(e.toString());}
                System.out.println("----public Hashtable getBoundSel()
|||||||end-----------");
          return ht;
      }

//**
   저장고 번호가 whcd인 저장고의 상태값을 삭제 한다.
**//

 public void delCmd(String whcd)
 {
            Statement stmt=null ;
            Connection conn=null;
            ResultSet rst =null;
            String ss,query;

            try
           {
              ctx = new InitialContext();
              if(ctx == null ) throw new Exception("Boom - No
Context");
              ds =  (DataSource)ctx.lookup("MYSQL_JNDI_WH");
              if (ds != null)
              {
                  conn = ds.getConnection();
              }
             }catch(Exception e) {e.printStackTrace();}
             try {
                    stmt = conn.createStatement();
                    query="delete from wh_cmd where wh_cd='"+whcd+"'";
                    stmt.executeUpdate( query);
                    stmt.close();
                    conn.close();
          } catch(Exception e){e.printStackTrace(); ss="error";}
```

```
    }
```

마. 통신 장애를 위한 대안수립

통신장애는 크게 2부분으로 분리될 수 있다. 우선은 컨트롤러와 클라이언트 프로그램간의 통신장애와 클라이언트 프로그램과 서버간의 통신장애를 들 수 있다. 본 연구에서는 만약 통신장애가 발생할 경우 그 상황을 관리자에게 알려주는데 주안점을 두었다.

1) 컨트롤러와 클라이언트 프로그램간의 통신 장애

본 연구에서는 컨트롤러와 클라이언트 프로그램간의 통신장애에 대비해서 컨트롤러의 자체 동작과 클라이언트 프로그램에 의한 동작을 동시에 수행할 수 있도록 하였다. 이는 기본적으로 컨트롤러에 의해 저온저장고가 운영되며 주기적으로 클라이언트 프로그램에 데이터를 전송하고 클라이언트 프로그램에서 제어 데이터가 전송되어져 오면 그 데이터에 우선권을 주어 컨트롤러에서 실행한다. 이를 통하여 컨트롤러에 의한 저온저장고의 독립적 운영이 가능해 지고 저온저장고와 클라이언트 프로그램간의 연관성을 최소화 하여 컨트롤러와 클라이언트 프로그램간의 에러 전파가 최소화 되도록 하였다. 그리고 클라이언트 프로그램과 컨트롤러간의 통신 장애가 발생할 경우 서버에 장애를 알려 관리자에 의해 조치되도록 구성하였다. 또한 컨트롤러에서 과전류에 의한 저온저장고의 이상이 발견될 경우 클라이언트 프로그램에 이상 유무를 전송하고 클라이언트 프로그램은 서버로 재전송하여 관리자에 의한 모니터링 및 제어가 가능 하도록 하였다.

2) 클라이언트 프로그램과 서버간의 통신장애

클라이언트 프로그램과 서버간의 통신장애에 대한 감지를 위하여 클라이언트 프로그램에 의해 데이터 전송이 일정 시간 이상으로 전송되어지지 않을 경우에는 통신장애로 인정하고 조치할 수 있도록 하였다. 이는 클라이언트에서 서버로 주기적으로 데이터를 전송하는 구조에서 데이터의 전송이 없다면 이는 통신장애

로 인정 할 수가 있다. 그림 55는 서버에 나타나는 시스템의 전반적인 에러사항을 보여주고 있다. '저장고'는 컨트롤러에 의해 저온저장고에 과전류가 흐르고 있을 경우에 적색으로 나타나며 '클라이언트 통신'은 클라이언트 프로그램과 컨트롤러간의 통신에 장애가 있을 경우 적색으로 나타나고 서버 통신은 일정시간 동안 클라이언트에서 서버로의 데이터 전송이 이루어지지 않고 있을 경우 적색으로 나타난다.

그림 55 관리자화면에서 경고사항 보기

11. Server용 프로그램 개발

본 연구에서는 WAS(Web Application Server)로 Java 기반의 BEA사의 Weblogic 8.0을 사용하였다. Weblogic은 이미 안정성, 편의성, 기능성에 대해 인정을 받고있다. 데이터베이스와 인터페이스를 위해 Java의 JDBC(Java DataBase Connectivity)를 사용하였으며 비즈니스 로직을 위해 EJB(Enterprise JavaBeans)를 사용하였고 클라이언트 프로그램의 원격 객체 접속을 위해 웹서비스 개념을 위한 프로토콜인 SOAP(Simple Object Access Protocol)을 사용하였다.

1) WebLogic 구성

BEA WebLogic Server는 확장성 있는 엔터프라이즈 수준의 자바 표준을 따르는 애케이션 서버를 필요로 한다. BEA WebLogic Server는 EJB 컴포넌트, 자바 메시징 및 이벤트 서비스, 마이크로소프트 COM 통합, J2EE표준 스펙을 준수한 완벽한 플랫폼을 제공한다.

2) WebLogic Enterprise Server

고도의 확장성과 코바를 지원하는 대규모의 애플리케이션의 개발 및 메인프

레임과의 연동을 필요로 하는 경우에는 이 팩키지를 선택할 수 있다. BEA WebLogic Enterprise는 웹 기반의 애플리케이션과 기존의 애플리케이션 및 메인 프레임 애플리케이션을 하나의 통합된 시스템으로 구축할 수 있다. BEA WebLogic Enterprise는 BEA Tuxedo와 BEA M3에 탑재되었던 동일한 엔진을 갖추게 되어 강력한 성능을 발휘하면서도 EJB, 코바 및 BEA Tuxedo의 애플리케이션 프로그래밍 인터페이스를 제공한다. 따라서 BEA WebLogic Server, BEA M3 또는 BEA Tuxedo로 개발된 애플리케이션은 수정 없이 BEA WebLogic Enterprise에서 운영할 수 있다. 이러한 자동적인 상향 호환성(upward compatibility)이 의미하는 것은 BEA의 어떠한 미들웨어를 사용하는 경우이든 향후에 BEA WebLogic Enterprise 상에서 운영할 수 있게 되어 투자에 대한 보장을 받게 된다.

3) WebLogic 기능 및 장점

표 31 WebLogic 기능 및 장점

기능	장점
■ 웹 서비스	
- SOAP - WSDL - UDDI	별도의 추가적인 프로그래밍 없이 EJB를 웹 서비스로서 알리거나, 다른 플랫폼에서 호스팅 되어지는 웹 서비스들을 리모트에서 접근할 수 있다.
■ 프리젠테이션 서비스	
- Built-in 웹서버 - Apache, Microsoft IIS, Netscape통합 - 서블릿, JSP 엔진 - 향상된 웹 캐싱	무선과 웹 애플리케이션에 동적, 정적인 컨텐츠를 제공하는 플랫폼을 자체 내장하고 있으며, 성능과 확장성을 증가시키기 위해 고속의 페이지 캐싱 기능을 제공한다.
■ 웹 프리젠테이션 서비스 클러스터링	
- 서블릿 상태 캐싱을 제공하는 향상	서버의 클러스터링을 통해 웹서버의

된 페일 오버 - 웹 서비스의 동적인 발견	확장성과 가용성을 향상시키며, 메모리내 중복기능으로 고성능의 페일오버를 제공한다.
■ 비즈니스 로직 서비스	
- EJB 컨테이너 - 2PC를 지원하는 분산 트랜잭션관리	EJB 서버는 트랜잭션, 데이터베이스 연결 같은 미들웨어 서비스를 자동적으로 제공해줌으로써 애플리케이션을 개발하는 복잡성을 줄여준다.
■ 비즈니스로직서비스클러스터링	
- 로드발란싱 - 중복된네이밍, 스마트스텁, 메모리내 EJB 상태 캐싱 등의 향상된 페일오버 기능	서버 클러스터링을 통해 애플리케이션 단계의 확장성과 고 가용성을 가능하게 한다. 특히 메모리내 중복기능은 비즈니스로직의 페일오버 성능을 향상시킨다.
■ 데이터접근서비스	
- JDBC 지원과드라이버 - 네이밍과디렉토서비스 - XML	백엔드시스템과 통합이 가능하다.
■ 데이터 접근 서비스 클러스터링	
- JDBC 멀티풀(Multipool)	고 가용성과 확장성을 증가시키기 위해 향상된 클러스터링 서비스를 제공한다.
■ 엔터프라이즈 메시징 플랫폼	
- JMS 클러스터링 - JavaMail	메시지통합은 비즈니스데이터와 이벤트간의 비동기적인 데이터교환을 위한 신뢰성 있고 유연한 서비스를 제공한다. 엔터프라이즈 메세징시스템은 Point-to-point나 publish/subscribe 구조에서 메시지를 생성하고 처리한

	다.
■ 애플리케이션통합	
- Java Connector Architecture(JCA) 지원	Java Connector Architecture (J2EE CA) 지원은 JCA를 준수하는 리소스 아답터를 가진 어떤 애플리케이션과도 WebLogic Server가 연결될 수 있도록 해준다. 더 향상된 통합기능은 BEA WebLogic Integration에서 제공된다.
■ 인증된 J2EE 준수	
- EJB 1.1, 2.0 - J2EE CA 1.0 - JDBC 2.0 - JSP 1.21 - Servlet 2.32 - JTA 1.01 - JMS 1.0.2 - JNDI 1.2 - Java RMI 1.0 - RMI/IIOP 1.0 - JAAS 1.0 - JMX 1.0 - JavaMail 1.1	산업표준인 J2EE 플랫폼에서 개발함으로써 투자를 보호하고, 기업이나 개발자가 단일한 방법으로 작업할 수 있게 한다.
■ 기타 인터넷 테크놀러지	
- JHTTP 1.1 - SSLv3 - LDAPv2	표준 인터넷 프로토콜과 통합
■ 개발툴 통합	
- 웹게인스튜디오, Visual Age for	선택한 개발 툴을 사용하여 빠르게

Java, JBuilder 같은 개발 툴들과의 통합	개발할 수 있다.
■ 엔터프라이즈 관리	
- 웹기반 관리콘솔 - Java Management Extensions (JMX) - SNMP	개발자와 관리자들을 위한 웹기반의 구성과 관리, 모니터링을 위한 툴을 제공하고, 다른 우수관리 프레임웍과도 통합 된다.
■ 보안	
- 유연성 있는 권한과 인증 - 통합된 보안기능과 방화벽지원 - 통합된 로깅	WebLogic Server는 SSL 이나 X.509, 디지털 서명, ACL 기반의 선택적인 암호와 권한, 인증 등을 사용하여 네트워크 애플리케이션에 보안기능을 장착할 수 있다. 모든 WebLogic Server 서비스는 HTTP나 HTTPS 터널링을 통해서 방화벽을 통한 보안이 가능하다.

4) WebLogic 장점

가) 표준을 따른 최초의 웹 어플리케이션 서버

복잡한 웹 어플리케이션의 쉬운 구현을 위한 중요한 산업 표준인 EJB, JSP, JMS, JDBC, XML, WML 등을 지원하고, 표준 솔루션을 제공함으로써 개발을 쉽게 한다.

나) 높은 성능과 확장성

높은 확장성과 클러스터 아키텍쳐 기반의 BEA WebLogic Server는 부하 분산, 커넥션 풀링, 결과 캐싱을 제공하며, 웹 서버, 오퍼레이팅 시스템, 가상 머신, 데이터베이스 연결 등을 최적화한다.

다) 높은 가용성

BEA WebLogic Server는 미션크리티컬한 e-비즈니스 애플리케이션을 신뢰성 있게 배포할 수 있게 하여 준다. BEA WebLogic Server는 웹과 비즈니스 로직

계층 양측의 자동 페일오버를 제공한다.

라) 신속한 개발

BEA WebLogic Server의 EJB, JSP, 서블릿 컴포넌트 아키텍쳐로 인해 신속하게 시장에 대응할 수 있도록 개발 속도를 높여 준다. 이러한 개방형 표준들은 WebGain Studio와 밀접하게 결합되어 개발을 더욱 간결하게 해주고, 빠르게 어플리케이션을 배치할 수 있도록 도와준다.

마) 다양한 클라이언트 지원

BEA WebLogic Server는 다양한 웹 브라우저, 무선 디바이스, 프로그램 가능한 클라이언트 등 복잡한 e-비즈니스 애플리케이션을 지원하는 복잡한 e-비즈니스 애플리케이션을 위해 설계되었다.

바) 유연성

BEA WebLogic Server는 각 부문에서 선두를 차지하고 있는 대부분의 데이터베이스, 운영체제, 웹 서버, 웹 브라우저, 무선 디바이스와 통합할 수 있는 유연성을 제공한다.

사) 기존 시스템과의 유연한 통합

BEA WebLogic Server는 기존 레거시 시스템과의 유연한 통합을 위하여 J2EE 커넥터 아키텍쳐 기반을 이용하여 모든 EIS 시스템과의 연동을 가능하게 한다.

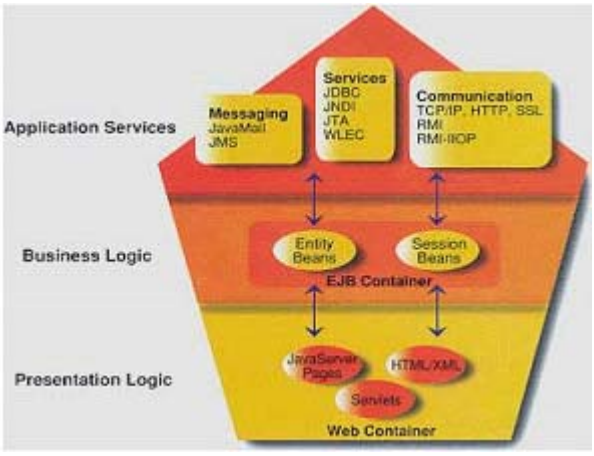

그림 56 Weblogc Platform의 구성

가. 요구 Client의 인증 처리 프로그램 개발

본 연구에서는 클라이언트 프로그램의 인증처리를 위해 웹 서비스를 사용하였다. 클라이언트 프로그램에서 ID와 Password를 입력하면 이를 네트워크를 통해 서버로 전송하게 되고 서버에서는 이에 대한 인증 처리를 한 후 클라이언트에 그 정보를 전송한다. 이는 전적으로 SOAP의 통신방식을 따르며 인증 모듈의 변경이 서버에서만 이루어지면 됨으로 추후 변경에 효과적이라 할 수 있다. 다음은 DivWS 서비스에 정의되어진 인증을 위한 메소드이다. 이 인증 메소드에서는 SelectBean이라는 EJB를 통해 Query를 실행하고 그 결과 값을 다시 클라이언트 프로그램에 전송한다.

그림 57은 인증처리의 개념도이며, 표 32와 표 33은 요구 클라이언트의 인증처리를 위해 개발한 인증 메소드에 관한 프로그램 이다.

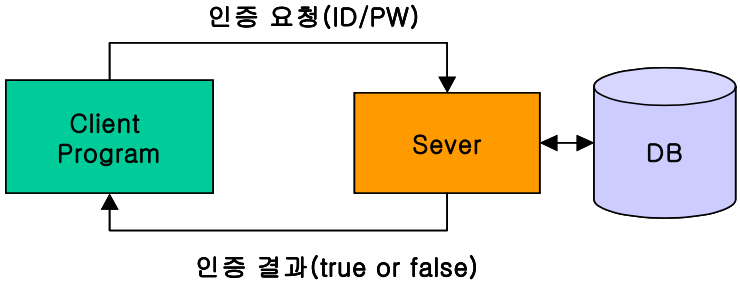

그림 57 인증처리 개념도

표 32 인증 메소드

```
public boolean getAuthSel(String id,String pw)
 {
       boolean autf;
       SelectHome home=null;
       Select select;
       int i=0;
       try{
```

```
            InitialContext ctx=new InitialContext();
            Object objref=ctx.lookup("SelectBean");
            home=(SelectHome)PortableRemoteObject.narrow(objref,
              SelectHome.class);
            select=home.create();
            autf=(boolean)select.getAuthSel(id,pw);
        } catch(Exception e)
        {
            e.printStackTrace();
            autf=false;
        }
        return autf;
    }
```

표 33 EJB에서 인증 처리하는 메소드

```
public boolean getAuthSel(String id,String pw)
{
            Statement stmt ;
            ResultSet rst =null;
            String ss;
            Context ctx;
            DataSource ds;
            Connection conn=null;
            boolean ht;

            try{
             System.out.println("================public boolean
                        getAuthSel(String id,String pw)=============");
             ctx = new InitialContext();
             if(ctx == null )          throw new Exception("Boom - No
Context");

             ds =  (DataSource)ctx.lookup("MYSQL_JNDI_WH");
             if (ds != null)  conn = ds.getConnection();
           }catch(Exception e) { System.out.println(e.toString());}
```

```
            try{
              stmt = conn.createStatement();
             String sql=" select distinct id from wh_user_info where id='"+id+"'
                  and pw='"+pw+"'";
              rst = stmt.executeQuery( sql);
              if ( rst.next()){
                       ht=true;
                  }else ht=false;

              rst.close();

        } catch(Exception e){ht=false; System.out.println(e.toString());}
        try{
              conn.close();
         }catch(Exception e){System.out.println(e.toString());}
         System.out.println("-----public boolean getAuthSel(String id,String
pw)||||| end----------");
         return ht;
        }
```

나. 다중 접속 처리에 대한 연구

본 연구에서는 클라이언트 프로그램 및 사용자들의 웹을 이용한 다양한 접근을 처리하기 위해 Web Application Server와 인터넷 서비스 요청을 받는 부분을 분리하여 구성하였다. WAS 앞단에 Apache WebServer를 사용하여 사용자의 Connection 관리를 수행하였다.

1) Apache WebServer

아파치는 1995년 그 당시에 가장 영향력 있었던 웹 서버중의 하나인 NCSA HTTPD 1.3 버전을 기반으로 탄생하였다. 그 후 기존의 NCSA 웹 서버에 더욱 향상된 기능들을 탑재하여 오늘의 Apache 웹 서버로 발전하였다. 지속적으로 패

치파일을 제공하고 최고의 퍼포먼스를 내고 있기 때문이다.

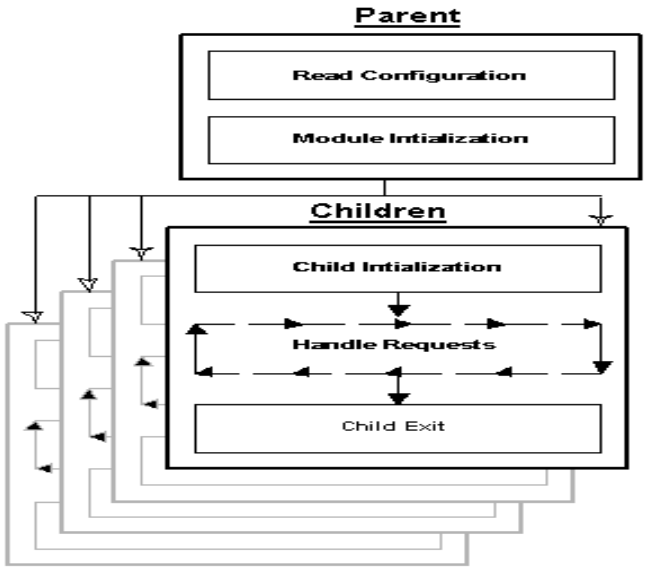

그림 58 Apache 프로세스 모델

2) Weblogic과 아파치 웹서버의 통합

Weblogic을 단독으로 사용하는 데에는 다음과 같은 문제점이 있으므로 본 연구에서는 아파치 웹서버와 연동하여 사용하였다.

■ 아파치 웹서버에 비해 정적 페이지 서비스가 느리다.
■ 아파치 만큼 다양한 웹서버 환경을 제공하지 않는다.
■ 아파치 만큼 웹서버로서 안정적이지 않다.
■ 많은 사이트들이 기존의 웹서버 환경에서 서비스하고 있는 것들이 많다.
■ 웹서버의 기본은 클라이언트의 HTTP(Hypertext Transfer Protocol) 요청을 기다리다가 요청이 들어왔을 때 필요한 컨텐트를 제공함으로써 이 요청을 서비스한다.

서블릿 컨테이너를 추가함으로써 웹 서버는 다음과 같은 행동을 한다.
□ 서블릿 요청이 있기 전에 서블릿 컨테이너 어댑터 라이브러리를 로딩하고

초기화한다.

□ 요청이 들어오면 서블릿에 대한 요청인지를 체크해서, 어댑터가 이 요청을 받아 처리한다. 어댑터는 HTTP(Hypertext Transfer Protocol) 요청 중에 서블릿에 대한 요청을 구분해내야 하는데, 일반적으로 요청 URL 패턴에 근거한다.

이와 같이 WAS을 단독으로 사용하는 데에는 효과적인 연결 관리 및 성능 관리를 하는데 문제점이 있으므로 본 연구에서는 WAS와 Webserver를 같이 사용함으로써 다중접속 처리의 성능을 향상시킬 수 있었다.

다. 네트워크 접속 프로그램 개발

네트워크 접속 프로그램은 크게 두 부분으로 나눌 수 있다. 클라이언트 프로그램에서 서버로 데이터를 전송하도록 하는 부분과 관리자 및 웹을 통해 접속하는 사용자들의 접속을 관리하는 부분으로 나눌 수가 있다.

1) 클라이언트 프로그램에서 서버로의 데이터 전송 관리

가) 클라이언트와 서버 간 통신

클라이언트는 서버와의 통신을 위해 SOAP를 통한 통신을 한다. 본 연구에서는 별도의 클라이언트 프로그램의 접속을 관리하는 프로그램을 사용하지 않고 기존의 Web Application Server인 Weblogic을 사용하여 안정적이고 효과적인 운영을 지향하였다. web Application Server는 HTTP 통신 포트인 80 포트를 사용하여 통신이 이루어지며 WWW(World Wide Web)을 사용하는 곳에서는 항상 열려 있는 포트이다. 또한 HTTP기반의 SOAP을 통한 원격 객체 통신을 통하여 안정적인 어플리케이션 관리가 이루어지도록 하였다. SOAP은 전송 프로토콜로 HTTP를 사용함으로 데이터 전송 시 HTTP의 특성을 그대로 따른다. HTTP는 특성상 서버와 클라이언트의 연결 지속성이 유지되지 않는다. 따라서 모니터링 데이터의 전송과 제어 데이터를 가져오는 부분은 별개로 이루어진다. 이는 HTTP가 단 방향 통신만이 가능하기 때문이다. 이는 서버의 리소스 및 네트워크 리소스를 절약할 수 있고 서버와 클라이언트의 연결 지속에 의해 발생하는 문제

점들을 해결할 수 있으며 방화벽을 통해서도 쉽게 전달되므로 인트라넷과 인터넷을 통한 어플리케이션 접속이 가능해진다.

나) HTTP(Hypertext Transfer Protocol)

(1) HTTP(Hypertext Transfer Protocol)통신은 TCP/IP 통신상의 어플리케이션 프로토콜의 일종으로 웹 클라이언트와 웹 서버가 서로 데이터를 주고받는 데 필요한 약속이다. HTTP 통신은 Connectionless와 Stateless 통신의 특징을 갖고 있어 초기에는 인터넷 통신에 가장 적합한 통신 구조로 인정받았다.

(2) Connectionless 통신 구조: HTTP 통신은 HTTP 요청에 대해 TCP통신을 설정한 후 요청에 대한 응답이 처리되면 TCP 연결을 끊어 버리는 형태의 통신으로 트랜잭션이 연결된 지속적인 통신에는 부적합 하지만, 인터넷과 같이 다수의 사용자를 대상으로, 사용자가 원할 때 필요한 HTML 문서 서비스를 전달하는 클라이언트 Puul 방식의 서비스에는 네트웍의 오버헤드가 없는 적합한 통신 방식이다. 그러나 단방향 C/S 아키텍쳐에서 양방향 분산통신으로 발전하면서 Connectionless 방식의 HTTP 통신은 또 다른 문제를 제기하고 있어 새로운 프로토콜(Socket, RMI, CORBA 등)이 웹 통신에 적용되고 있다.

(3) Stateless 통신 방식: 한번의 요청에 대해 한 번의 응답으로 HTTP 트랜잭션이 종료되므로 연속적인 작업 처리에 필요한 트랜잭션 상태 정보를 관리하기 위한 웹 서버측의 Overhead가 필요 없다. 그러나 Stateless통신 구조로 인해 사용자 로그인, 쇼핑 정보 보관 등 사용자 상태 정보를 관리하기 위해 현재는 Cookie 또는 CGI 스크립트 상에서 지원하는 Session정보 등 Overhead를 부여하고 있다.

(4) HTTP 요청과 응답 메시지 구조: HTTP 통신은 클라이언트의 HTTP요청 메시지에 대해 서버의 HTTP 응답 메시지를 전달 받는 형식이다. 따라서 요청 및 응답 메시지는 HTTP 통신의 핵심이라 할 수 있다. 각 메시지는 시작 라인, 메시지 헤더, 메시지 바디 세 부분으로 구성되어 있다.

■ 시작 라인 : 요청 메시지인 경우는 클라이언트 요청사항에 대한 정보가 응답 메시지의 경우는 응답 내용의 상태 정보로 구성된다. 시작 라인만으로 구성된 HTTP 메시지도 존재한다.

■ 메시지 헤더 : HTTP 메세지에 대한 부가적인 정보를 담는다. 수행 날짜,

프로그램 이름, 버전, 쿠키, 사용자 인증, 캐시 등의 정보가 포함된다. 각 헤더 정보의 끝은 항상 CRLF로 하나의 헤더정보의 끝임을 나타낸다.

■ 메시지 바디 : 요청이나 응답에 필요한 내용을 담고 있다. POST 방식의 경우 요청 메시지에 엔코딩된 데이터 스트림으로 구성되고, 응답 메시지의 경우엔 응답 문서 내용이 들어온다.

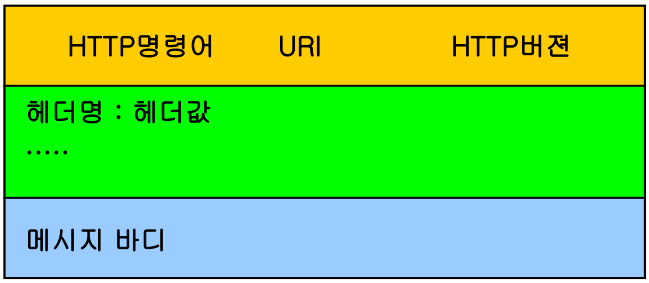

그림 59 클라이언트 요청 메시지 구조도

(5) 서버 응답 메시지: 서버응답 메시지는 다음과 같은 구조를 갖는다.

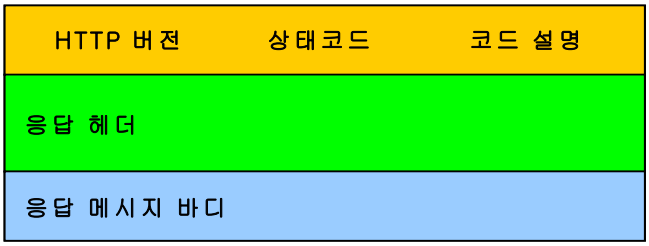

그림 60 서버응답 메시지 구조

서버의 응답 메시지는 특별한 경우가 아니고는 에러가 난 경우에도 메시지 바디를 갖고 있다. 상태 코드는 표 34와 같은 범위를 갖는다.

웹 서버에 따라 상태 코드와 설명을 자체 커스트 마이징 할 수 있는 방법을 제공하고, CGI 프로그램에서도 자체 에러를 처리해 사용자에게 친근한 메시지를 발행할 수 있도록 만드는 것이 좋다. 대부분의 웹 브라우저는 100에서 300번대 영역의 응답상태는 사용자에게 보여주지 않고 결과를 처리하며, 400과 500번 대의 일부 에러 코드들은 사용자에게 직접 보여준다.

표 34 상태 코드

참고정보	100 ~ 199
요청 성공	200 ~ 299
방향 재지정	300 ~ 399
요청 불안전	400 ~ 499
서버 에러	500 ~ 599

다) Object Web

ASP, Java Servlet, JSP 등에서 서버측의 Java Beans, EJB, COM+ 컴포넌트 접근 기술 등이 제공되고 있다 하더라도 여전히 HTTP/CGI 통신 구조 안에서는 클라이언트의 상태 비 보존성과 Connectionless 서비스로 인해 기업형 Application 서비스를 웹으로 이동시키기에는 많은 문제점이 있다. 이것을 해결하는 방법은 통신 기반구조를 HTTP/CGI에서 IIOP나 DCOM같은 통신기반 구조로 이동하여 클라이언트의 객체와 서버의 객체가 직접 컴포넌트 통신을 하고, 이들 컴포넌트 통신간의 고품질 통신 서비스 중재를 담당할 중재자가 필요하게 되었다. 또한 기존의 Application을 손쉽게 웹 환경으로 통합하기 위해서는 Application 플랫폼과 상관없이 모든 서비스를 서비스 컴포넌트로 손쉽게 묶어줄 수 있는 통합 기술이 필요하게 되었다. 이것이 오늘날 CORBA와 EJB, DCOM 등이 WEB을 Application 프레임웍으로 발전시키는 확실한 차세대 기술로 자리 잡게 하는 것이고, 여기서 Java Applet, Java Beans, Active X기술은 클라이언트를 컴포넌트 환경으로 구성하는 중요한 역할을 수행한다. 그림 61은 전형적인 CORBA/Java 기반의 Object Web 서비스 형태를 보여준다.

(1) 클라이언트가 HTML/XML서비스인 경우: JSP와 Java Servlet을 사용해 동적으로 웹문서를 생성한다. JSP/Servlet은 Business Logic을 EJB나 CORBA 서비스객체로부터 서비스 받는다.

EJB/CORBA 서비스 객체는 JDBC, 또는 Legacy 연동 Java API 모듈을 이용

해 시스템 서비스를 접근한다.

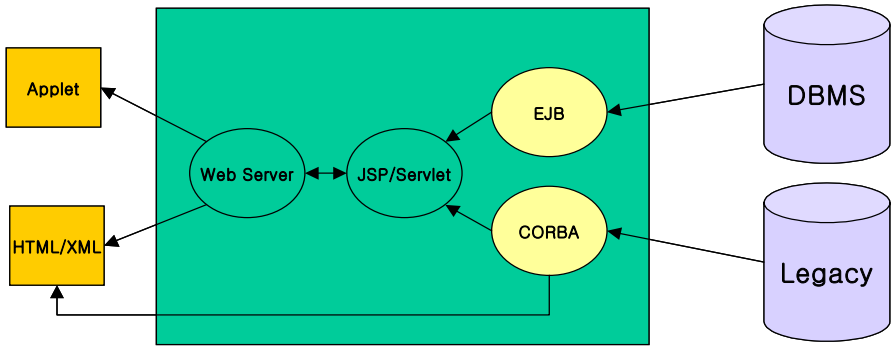

그림 61 HTTP, CORBA/Java 통신 방법

(2) 클라이언트가 Applet일 경우: 웹 브라우져를 통해 Applet이 포함된 HTML페이지를 다운로드 받는다. 웹 브라우져는 서버로부터 Applet을 다운로드 받아 Applet을 실행시킨다. Applet은 브라우져의 내장 ORB 또는 Applet 다운로드시 Jar에 묶어서 같이 다운로드 받은 ORB로 서버쪽 CORBA서버와 IIOP 트랙잭션을 맺고, 지속적인 Business 트랜잭션 서비스를 요청하여 서비스 한다.

본 연구에서는 HTTP 기반의 object 통신인 SOAP를 이용하여 WEB상에서 원격 객체통신이 가능하도록 하였다.

라) 클라이언트 프로그램과 서버의 통신 절차

클라이언트 프로그램에서 서버로의 데이터의 전송 시 다음과 같은 절차를 거쳐 저온저장고의 모니터링 데이터가 서버에 전송되어진다.

(1) 클라이언트 프로그램은 컨트롤러에서 수집한 모니터링 데이터를 포맷에 맞추어 정리한다.

(2) 클라이언트 프로그램은 WSDL에 정리되어진 서비스로 구현되어진 객체를 통해 특정 함수를 호출한다.

(3) 함수는 XML에 따라 SOAP 전송을 위한 데이터 포맷에 맞추어 전송되어질 데이터를 정리한다.

(4) 클라이언트 프로그램은 HTTP를 통해 서버에 연결을 시도한다.

(5) 서버에서 연결을 수락한다.

(6) 클라이언트 프로그램은 HTTP를 통해 데이터를 서버로 전송한다.

(7) 서버는 데이터를 수신하고 연결을 끊는다.

2) 웹을 통해 접속하는 사용자들의 접속 관리

웹을 통한 접속 관리는 Web Application Server에 의해 이루어진다. 웹을 통한 통신은 HTTP를 이용한다.

라. 클라이언트 전송정보 관리 프로그램 개발

1) EJB(Enterprise JavaBeans)

Enterprise JavaBeans (EJB)는 분산환경하의 컴포넌트 아키텍처이다. 대표적인 분산환경 으로는 Corba를 들 수 있으며 Corba는 ORB를 통해 분산 환경하의 객체호출 메커니즘, 트랜잭션 처리 등 여러 가지 서비스를 제공하고 있지만 분산 작업의 기본단위가 되는 컴포넌트 모델은 제공하지 못하고 있다. 따라서 Corba3.0 에서는 EJB가 Corba의 표준 컴포넌트 모델이 될 것이다.

가) EnterPrise Beans의 특징

■ enterprise Bean's instances들은 Container에 의해 RunTime에 생성되고 관리된다.

■ enterprise Bean은 environment properties를 수정함으로써 deployment time에 최적화 될 수 있다.

■ 트랜잭션모드, 보안속성 등 다양한 Metadata는 EnterPrise Beans 클래스로부터 외부에 분리해서 위치할 수 있다. 이것은 design and deployment time에 Container Tools를 이용해 다루어 질 수 있음을 의미한다.

■ Client Access는 Container(WebLogic의 Thenah서버 등) 혹은 EJB Server(OS, DBMS 등)에 의해 조정될 수 있다.

■ enterprise Bean이 EJB specification에 정의된 standard container services 만 사용한다면 어떤 Container에서도 재 컴파일 없이 사용 가능하다.

■ enterprise Bean는 재 컴파일이나 소스코드 변경없이 애플리케이션 구성에 포함될 수 있다.

나) Enterprise JavaBeans contracts

■ Client's view contract

클라이언트와 컨테이너 사이의 계약을 기술한다. enterprise Bean provider와 Container provider는 (Object identity, Method invocation, Home Interface)과 같은 계약을 이행할 책임을 가진다. 클라이언트는 enterprise Bean object가 유일한 식별자를 가질 것을 기대하며 컨테이너는 각각의 session EJB object에 대하여 이것을 제공해야 한다. 빈 제공자는 컨테이너가 EJB object's identifier 속에 포함할 수 있는 유일한 primary key를 제공(EJB object 생성 시간 동안)한다. 그리고 object activation and/or load time에 EJB의 primary key를 사용한다. 클라이언트는 Java Naming and Directory Interface(JNDI)를 이용하여 EJB Home interface를 찾을 수 있다. Primary key는 Home interface 내에서 각각의 EJB 객체를 식별하기 위해 사용한다. 빈 제공자는 클라이언트에서 호출할 수 있는 Business method를 정의하는 remote interface를 정의한다. 클라이언트가 enterprise Bean을 호출하면 컨테이너는 remote interface를 통해 호출을 받아들이고 실제 비즈니스 메소드의 실행은 enterprise Bean class에 위임한다. 또한 빈 제공자는 javax.ejb.EJBHome을 상속하는 enterprise Bean's home interface를 제공한다. home interface는 0개 이상의 create() 메소드를 제공함으로써 다양한 방법으로 EJB 객체를 생성한다. entity Beans의 홈 인터페이스는 0개 이상의 find...()메소드 또한 제공함으로써 여러 방법으로 DB 등에 저장되어 있는 EJB 객체를 찾을 수 있다. 빈 제공자가 홈 인터페이스 내에 정의한 create() 메소드는 enterprise Bean class내에 같은 시그네처를 가지는 ejbCreate()로 대응되며 컨테이너는 클라이언트에서의 create() 호출을 enterprise Bean class 내의 ejbCreate() 호출로 위임한다. 엔티티 빈의 find..() 메소드의 경우도 마찬가지이다.

■ Component contract

Enterprise Bean과 컨테이너 사이의 계약이다. 세션빈은 javax.ejb.SessionBean, javax.ejb.SessionSynchronization interfaces에 의해 엔티티 빈은 javax.ejb.EntityBean interface에 의해 정의된 상태관리 callback을 포함하

고 컨테이너는 객체의 생명주기 상에서 중요한 이벤트가 발생했을 때 이를 알리기 위해 이들 인터페이스에 의해 정의된 콜백 메소드를 호출한다. 컨테이너는 세션빈의 인스턴스를 생성할 때 javax.ejb.Session Context interface를 전달하며 SessionContext는 컨테이너로부터 다양한 정보와 서비스를 얻기 위해 사용된다. 엔티티 빈은 javax.ejb.EntityContext를 이용한다.

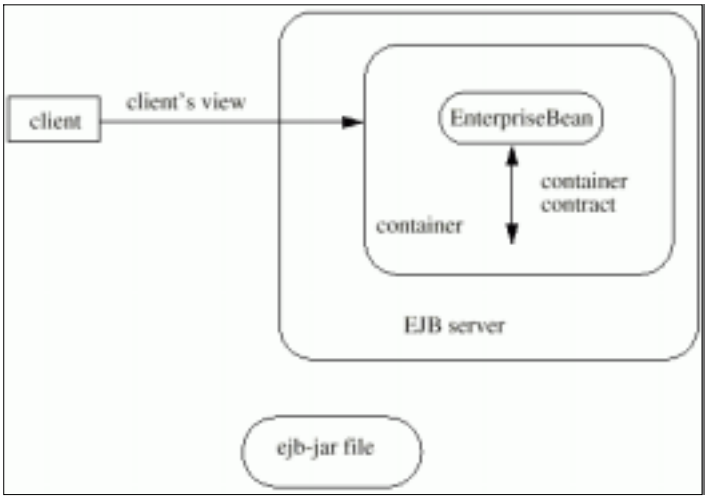

그림 62 Enterprise JavaBeans contracts

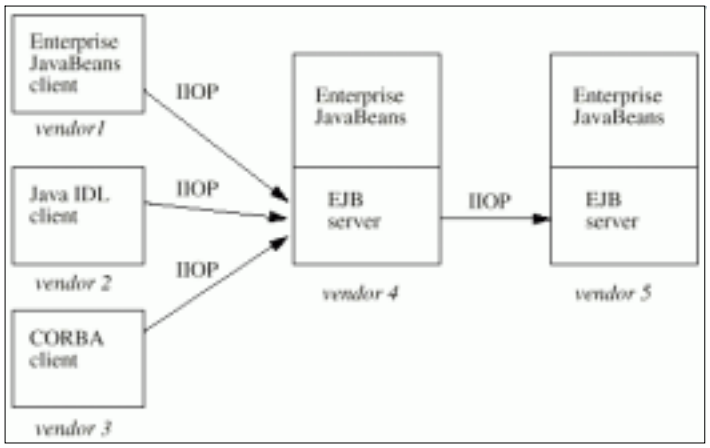

그림 63 Standard CORBA mapping

다) session bean

session Bean은 remote access, security, concurrency, transactions 등의 컨테이너가 제공하는 모든 기능들과 빈의 비즈니스 로직을 실행한다. session enterprise Bean은 서버 상에서 실행하는 비즈니스 로직을 다루는 비지속성 객체이며, multiple clients 사이에서 공유되지 않는 객체이다. 클라이언트는 session Bean's remote interface를 통해 세션빈에 접근하며 리모트 인터페이스를 구현하는 객체는 EJB Object 이다. EJB Object는 생명주기 동안 컨테이너 내에서 존재하며 클라이언트에 투명하다. 일반적으로 세션빈은 컨테이너가 실행중지 또는 재시작하면 생명을 잃게 된다. Multiple EJB classes들이 동일한 컨터이너 속에 포함될 수 있으며 클라이언트는 JNDI를 통해 인스톨된 EJB classes들의 Home interface를 찾을 수 있다. 컨테이너 내의 enterprise Bean object들은 데이터베이스, 파일시스템 등에 저장되어 지속성을 가진다.

■ enterprise Bean's home interface 찾는 방법

```
// 클라이언트는 JNDI를 통해 홈 인터페이스를 찾는다.
Context initialContext = new InitialContext(); CartHome cartHome
= javax.rmi.PortableRemoteObject.narrow(
                initialContext.lookup("applications/mall/freds-carts"),
                CartHome.class);
```

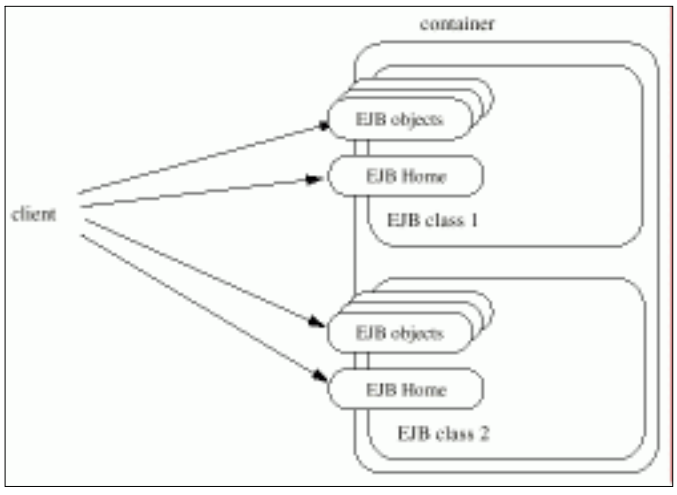

그림 64 세션 컨테이너가 클라이언트에 제공하는 것

■ Home interface: 컨테이너는 각각의 enterprise Bean을 위한 홈인터페이스를 실행하고 클라이언트가 JNDI를 통해 홈인터페이스를 호출할 수 있도록 만든다. 홈인터페이스를 얻은 클라이언트는 Create a new EJB object, Remove an EJB object, Get the jvax.ejb.EJBMetaData interface for the enterprise Bean과 같은 일을 한다.

■ EJB object: 클라이언트는 enterprise Bean's class에 직접 접근하지 않고 enterprise Bean's remote interface를 통해 접근한다. enterprise Bean's remote interface를 실행하는 클래스는 컨테이너가 제공한다. EJB object는 객체의 비즈니스 로직 메소드를 제공하며 클라이언트로부터의 비즈니스메소드 호출을 enterprise Bean instance로 위임하거나 클라이언트가 javax.ejb.EJBObject interface를 통해 'EJB object's container 얻기', 'EJB object handle 얻기', 'EJB object가 다른 EJB object에 대해 유일한지 테스트', 'EJB object 제거' 등을 제공한다. javax.ejb.EJBObject interface 내의 메소드 실행은 컨테이너가 제공한다.

■ Session object identity: 세션빈은 그들을 생성한 클라이언트로부터 private resources들을 다루는 경우가 많기 때문에 클라이언트의 시각에서는 익명이다. 즉 엔티티빈은 primary key로서 식별자를 공개하지만 세션빈은 숨긴다. 따라서 세션빈의 홈 인터페이스는 어떠한 find..() 메소드도 제공하지 않아야 한다. 또한 session EJB object handle을 지속성 저장장소에 저장함으로써 클라이언트의 생명주기를 넘어서 보관될 수 있다.

■ Client's view of session Bean's life cycle: 객체 참조를 얻은 클라이언트는 세션빈의 리모트 인터페이스를 통해 객체상의 애플리케이션 메소드를 호출하거나 객체의 홈 인터페이스에 대한 참조를 얻을 수 있으며 객체를 위한 handle을 얻을 수 있다. 또한 클라이언트의 범위 안에서 객체를 파라미터나 리턴값으로 보낼 수 있으며 객체를 제거할 수 있다. 또한 컨테이너는 객체의 생명이 다했을 때 자동으로 제거할 수 있다. 존재하지 않는 개체에 대한 참조는 사용할 수 없다. 만약 호출한다면 컨테이너는 java.rmi.NoSuch ObjectException을 발생할 것이다.

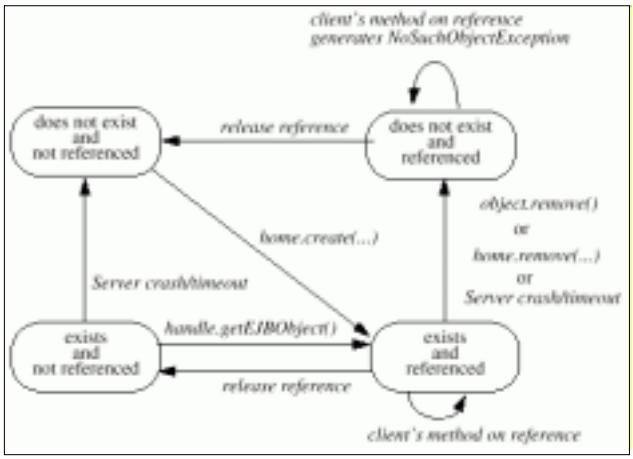

그림 65 session Bean's life cycle

■ Creating and using a session Bean: 클라이언트는 Cart's home interface 의 create() 메소드를 호출하여 Cart 세션객체를 생성한다.

```
CartHome cartHome =
javax.rmi.PortableRemoteObject.narrow(
    initialContext.lookup(...), CartHome.class);
Cart cart = cartHome.create(...);
    cart.addItem(66); cart.addItem(22);
```

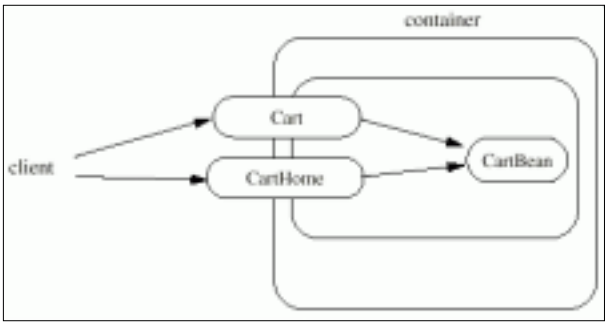

그림 66 An example of the session Bean runtime objects

라) entity Beans

business methods는 enterprise Bean 자신이 제공해야 한다. 엔티티빈은 지속

성 저장매체(데이터베이스, 파일시스템...)에 저장되는 지속성 객체이다. 클라이언트는 엔티티빈의 홈인터페이스를 통해 접근한다. 리모트 인터페이스를 구현하는 객체는 EJB object 이다. EJB object는 컨터이너 내에서 생활하며 클라이언트에 투명하다. 컨테이너는 EJB object를 위해 security, concurrency, transactions, persistence 등의 서비스를 제공한다. 컨테이너 또한 클라이언트에 투명하며 클라이언트가 컨테이너를 다룰 수 있는 API가 존재하지 않는다. Multiple clients는 한 엔티티 객체에 동시에 접근할 수 있다. 일반적으로 각각의 엔티티 객체는 식별자를 가지며 컨테이너가 재 시작하거나 작동을 멈추어도 계속 생존한다. 객체 식별자는 컨테이너에 의해 구현된다. enterprise Bean's home interface 클라이언트가 EJB Object를 create, look up, remove하는 것을 허락한다. 클라이언트는 JNDI를 통해 enterprise Bean's home interface를 찾을 수 있다. 컨테이너는 JNDI 이름 공간에서 enterprise Bean's home interface가 사용가능하도록 할 책임이 있다.

■ EJB container : 컨테이너는 enterprise Bean object가 생존하는 공간이다. 다음과 같이 참조할 수 있다.

```
// Account enterprise Bean을 위한 홈인터페이스를 찾는방법
    Context initialContext = new InitialContext();
    AccountHome accountHome =
javax.rmi.PortableRemoteObject.narrow(
initialContext.lookup("applications/bank/accounts",
                                    AccountHome.class);
```

■ Enterprise Bean's home interface : 컨테이너는 각각의 enterprise Bean의 홈 인터페이스 구현을 제공한다. 그리고 모든 클라이언트가 JNDI를 통해 홈 인터페이스에 접근할 수 있도록 한다. 홈 인터페이스의 구현 클래스는 EJB home 이다. 엔티티빈의 홈 인터페이스는 Create new EJB objects, Look up existing EJB objects, Remove an EJB object. Get the javax.ejb.EJBMetaData interface for the enterprise Bean과 같은 사항을 클라이언트에게 허락한다. 홈 인터페이스는 javax.ejb.EJBHome interface를 상속해야 한다.

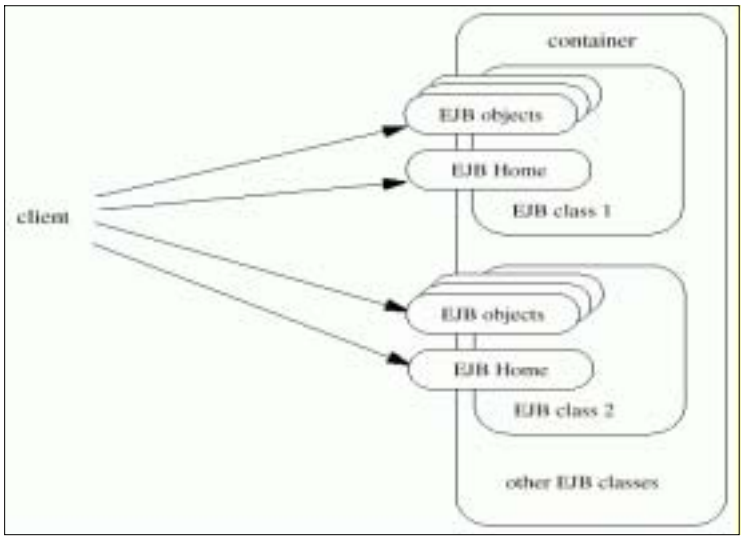

그림 67 컨테이너가 제공하는 것

■ Entity EJB object life cycle : EJB object는 생성(create())되기 전까지 존재하지 않으며 식별자도 가지지 않는다. 클라이언트는 컨테이너에 의해 실행되는 enterprise Bean's home interface를 이용해 EJB 객체를 생성한다.
entity EJB는 create() 메소드를 이용 하는 것 외에 다른 메커니즘 (데이터베이스 레코드에 직접 삽입하는 방법 등)을 사용하여 객체를 생성할 수 있다. 이 경우에 클라이언트는 find~() method만을 이용할 수 있다.
또한 remove()를 사용하지 않고 데이터베이스의 레코드를 직접 삭제함으로써 제거될 수도 있다.
존재하지 않는 객체에 대한 참조를 가지고 호출할 경우 java.rmi.NoSuchOb jectExcep -tion을 발생한다. 모든 엔티티빈은 지속성 객체이다. 따라서 생명주기는 자바 가상머신, 컨테이너들의 상태에 영향을 받지 않는다. Multiple clients는 같은 EJB 객체에 동시에 접근할 수 있으며 트랜잭션은 각각으로부터 클라이언트의 요청을 분리하기 위해 사용된다.
■ Primary key and object identity : 모든 엔티티 객체를 그것의 홈 내에서 유일한 식별자를 가진다. 컨테이너내의 객체 식별자는 EJB object's home 그리고 primary key에 의해 결정된다. 만약 두 객체가 같은 home과 primary key를 가진다면 두 객체는 동일하다. 그리고 primary key object 직렬화가 가

능해야 하며 primary key class는 enterprise Bean class에 특화된다. 즉 각각의 enterprise Bean class는 primary key를 위해 다른 클래스를 가질 수 있다. 클라이언트는 getPrimaryKey() 메소드를 호출하여 EJB 객체의 primary key를 얻을 수 있다. entity EJB object의 primary keys를 알고 있는 클라이언트는 컨테이너에 의해 실행되는 홈 인터페이스의 findByPrimaryKey(key)를 호출해서 객체의 참조를 획득할 수 있다.

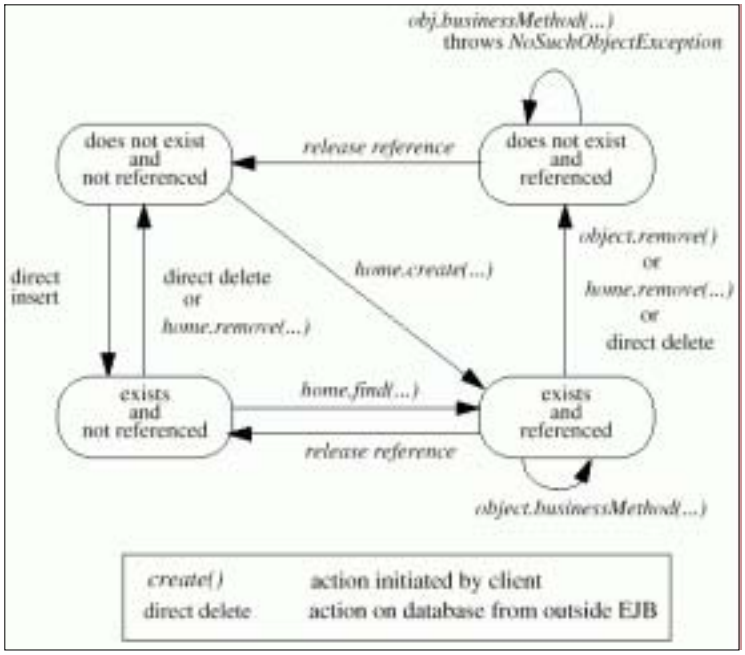

그림 68 Entity EJB object life cycle

■ Enterprise Bean's remote interface : 클라이언트는 리모트 인터페이스를 통해 Entity Bean에 접근할 수 있고 리모트 인터페이스는 business methods를 정의한다. 리모트 인터페이스의 정의 예는 다음과 같다.

```
public interface Account extends javax.ejb.EJBObject
                    {
                     void debit(double amount) throws
                          java.rmi.RemoteException,
                        InsufficientBalanceException;
                     void credit(double amount) throws
```

```
                    java.rmi.RemoteException;
                double getBalance() throws
                 java.rmi.RemoteException;
                }
```

■ Enterprise Bean's handle : handle은 EJB 객체를 식별하기 위한 객체이며, EJB 객체의 참조를 가지는 클라이언트가 참조 상에서 getHandle() 메소드를 호출함으로써 객체의 핸들을 얻을 수 있다. handle은 일반적으로 컨테이너에 의해 제공되지만 클래스를 직접 구현할 경우는 반드시 직렬화 가능하기 위해 java.io.Serializable interface를 구현해야만 한다. 그리고 클라이언트는 이후에 직렬화된 핸들을 사용할 수가 있다.

마) Query의 구성

본 시스템에서 사용되어지는 EJB는 DBMS와의 통신을 통한 안정된 데이터 관리를 수행하며, Query는 EJB에서 수행되어지도록 구성되어져 있다. EJB는 SessionBean을 사용하였다. 다음은 Weblogic에서 EJB를 deploy하는 화면이다.

그림 69 Weblogic에서 EJB deploy 화면

(1) CtrlBean

저온저장고의 제어를 위한 메소드들을 정의하였다.

- Mathod: setCmd

  return: void

  parameter

  + String cd: 저온저장고 번호
  + String cmd: 명령어
  + String ymd: 연 월 일
  + String hms: 시 분 초
  + String value: 명령어 값

Query: insert into wh_cmd values(cd,cmd,value,ymd,hms)

설명: 클라이언트 프로그램에 전송할 명령어를 설정하는 메소드이다.

- Mathod: getCmd

  return: Hashtable

  parameter

  + String whcd: 저온저장고 번호

Query: select wh_cmd_cd,wh_cmd_value from wh_cmd where wh_cd=:whcd order by wh_cmd_cd

설명: 클라이언트 프로그램에서 명령어를 가져가는 메소드이다. 리턴값인 Hashtable은 Java에서 정의되어진 함수이다. 따라서 델파이에서는 Array of array of String 변수로 바꾸어서 사용한다.

- Mathod: delCmd

  return: void

  parameter

  + String whcd: 저온저장고 번호

  Query: delete from wh_cmd where wh_cd=:whcd

설명: 클라이언트 프로그램에 명령어가 전송된 후에 명령어를 삭제 할 경우 사용하는 메소드이다.

▪ SelectBean.java

조회를 위한 메소드를 정의 하였다.

- Mathod: getSel

return: Hashtable

Query: select ind,id from wh.wh_comm order by ind

설명: 통신 파라미터를 가지고서 Hashtable로 만들어 전송한다.

- Mathod: getBoundSel

return: Hashtable

parameter

+ String whcd: 저온저장고 번호

Query: select wh_kd, set_value from wh_info where (wh_kd='8' or wh_kd='9' or wh_kd='10' or wh_kd='11') and wh_cd=:whcd order by wh_kd

설명: 각각의 저온저장고의 설정되어진 온도 및 습도에 대한 경계값을 가져오는 메소드이다.

- Mathod: getInfoData

return: int

parameter

+ String whcd: 저온저장고 번호

+ String whkd: 저온저장고 설정 정보 코드

Query: select set_value from wh_info where wh_cd=:whcd and wh_kd=:whkd"

설명: 저온저장고의 특정 설정 정보를 가져온다.

- Mathod: getAuthSel

return: boolean

parameter

+ String id: 저온저장고에 할당된 id

+ String pw: 비밀번호

Query: select distinct id from wh_user_info where id=:id and pw=:pw

설명: 저온저장고의 ID와 비밀 번호를 받아 인증 정보를 전송한다. 전송되어진 ID와 비밀번호가 DB에 저장되어진 정보와 일치할 경우에는 true를 존재하지 않거나 틀릴 경우에는 false를 전송한다.

▪ InsertBean

삽입을 위한 메소드를 정의 하였다.

- Mathod: setTableData

return: void

parameter

+ String tableNM: 테이블 이름(wh_temp 또는 wh_humid)

+ String cd: 저장고 번호

+ String ymd: 연 월 일

+ String h: 시간

+ String m: 분

+ String s: 초

+ String value: 측정값

Query: insert into :tableNM values(:cd,:ymd,:h,:m,:s,:value)

설명: 저온저장고의 현재 측정 온도와 습도를 저장하는 함수

- Mathod: setStateData

return: void

parameter

+ String cd: 저장고 번호

+ String ymd: 연 월 일

+ String h: 시간

+ String m: 분

+ String s: 초

+ String value: 측정값

Query: insert into wh_state values(:cd,:ymd,:h,:m,:s,:value)

설명: 저온저장고의 현재 상태값을 저장하는 함수

▪ UpdateBean

업데이트를 위한 메소드를 정의하였다.

- Mathod: setWhInfo

return: void

parameter

+ int value: 설정값

+ String whCd: 저장고 번호

+ String whKd: 설정값 코드

Query: update wh_info set set_value="value) where wh_cd=:whCd and wh_kd=:whKd

설명: 저온저장고의 설정값을 변경하는 메소드이다.

3) Webservice 프로그램 개발

EJB와 클라이언트 프로그램과의 통신을 이어주고 제어하는 역할을 하는 부분은 SOAP통신을 기반으로 하는 Webservice로 구현하였다.

그림 70 Weblogic에서 Webservice의 함수 정의 부분

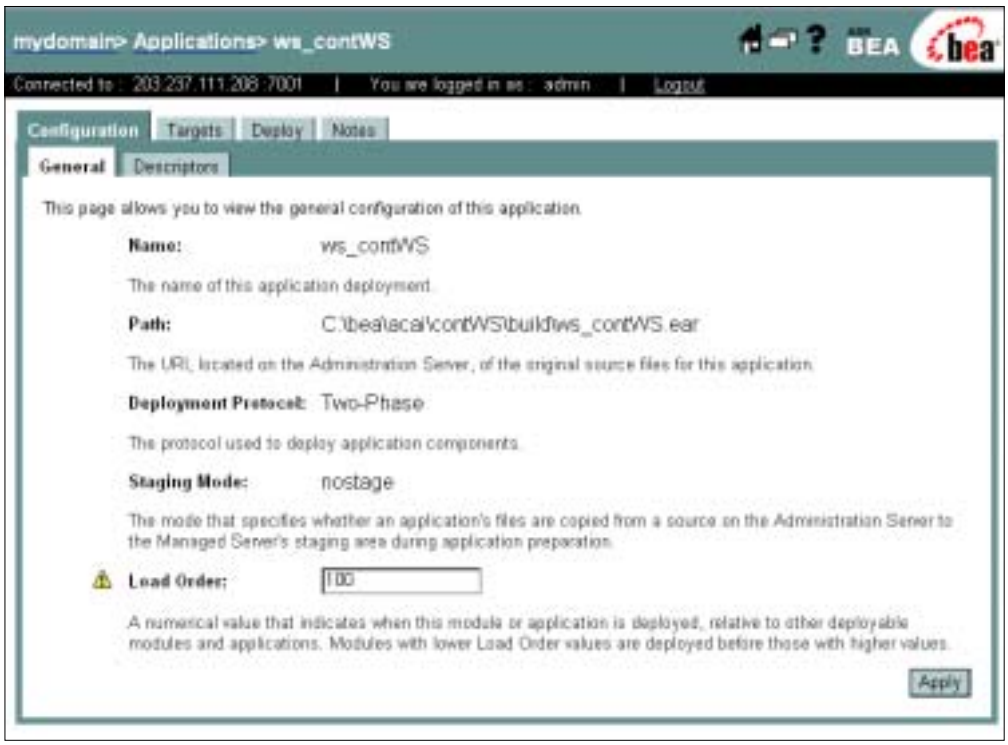

그림 71 Weblogic에서 Webservice의 일반적인 정의

마. 데이터베이스 연동 프로그램 개발

본 연구에서는 Java의 J2EE 기반에서 개발하였다. 데이터베이스와 연동을 위하여 Java에서 제공하는 JDBC를 사용하였으며 데이터의 관리를 위하여 데이터베이스와의 연결하여 데이터를 저장 및 가져오는 부분은 Java의 EJB를 사용하여 안정된 데이터 관리가 이루어지도록 하였다. EJB는 데이터의 저장 및 전송정보 관리를 위해 JDBC를 사용하여 데이터베이스와 연동되어진다.

1) JDBC(Java DataBase Connectivity)

본 연구에서는 클라이언트에서 전송되어진 데이터를 MySQL 데이터베이스에 저장하여 데이터를 보존하고 있다. 이를 위해서 Java Technology에서 제공하는 JDBC를 사용하여 데이터베이스의 데이터를 관리하였다.

JDBC는 자바 프로그램내에서 SQL문을 실행하기 위한 자바 API이다. JDBC는 자바로 작성되어진 클래스와 인터페이스들로 구성되어있다. 툴/데이터베이스 개발자들을 위한 표준 API를 제공하고 pure 자바 API를 사용하여 데이터베이스 어플리케이션을 만들게 해준다. JDBC를 사용하면, 어떠한 관계 데이터베이스(relational database)로도 SQL문을 전송하기 쉽다. 즉, JDBC API를 사용하면 Sybase, Oracle, Informix에 접근하는 프로그램을 따로 만들 필요가 없다. 단지 하나의 프로그램을 작성하고 그 프로그램에서 SQL 문을 적당한 데이터베이스에 전송할 수 있다. 또한 어플리케이션을 자바로 작성한다면, 어플리케이션을 플랫폼에 따라 다르게 작성하지 않아도 되기 때문에 자바와 JDBC의 결합은 하나의 프로그램이 어디에서나 동작할 수 있게 해준다. 자바는 사용하기에 견고하고 안전하고 이해하기 쉬우며 네트워크 상에서 자동적으로 다운로드 되기 때문에 데이터베이스 어플리케이션을 만드는데 있어서 최적의 언어이다. 단지 필요한 것은 다양한 데이터베이스에 연결하는 방법이다. JDBC는 이러한 것을 위한 메카니즘이다. JDBC는 자바의 기능을 확장한다.

가) JDBC의 동작 범위

JDBC는 다음 세 가지 일을 할 수 있다.

(1) 데이터베이스와 연결한다.

(2) SQL문을 전송한다.

(3) 결과를 처리한다.

나) 2-티어(tier) 그리고 3-티어(tier) 모델들

JDBC API는 데이터베이스 접근에 대해서 2-tier와 3-tier 모델 모두를 지원한다. 2-tier 모델에서는 자바 애플릿 또는 어플리케이션이 데이터베이스와 직접 통신한다. 이것은 접근한 특정 DBMS와 통신할 수 있는 JDBC 드라이버가 필요하다. 사용자의 SQL문을 데이터베이스로 전달하고, 그 SQL문의 결과를 사용자에 되돌려 준다. 데이터베이스는 사용자가 네트워크를 통해 연결한 다른 머신 상에 위치할 것이다. 이것은 사용자의 머신이 클라이언트, 그리고 데이터베이스를 가지고 있는 머신이 서버로써 클라이언트/서버 구성에 속한다. 네트워크는 인트라넷이 될 수도 있고, 인터넷이 될 수도 있다. 3-tier 모델에서는 명령들을 서비스의 미들티어(middle-tier)로 보내고, 그리고 나서 SQL문을 데이터베이스로 보낸다. 데이터베이스는 SQL문을 처리하고 결과를 미들티어로 되돌려주며, 그런 다음 그것을 사용자에게 전송한다. MIS 관리자는 미들티어가 기업 데이터로 만들 수 있는 일종의 최신 정보와 억세스에 대한 관리를 유지하는 것이 가능하기 때문에 3-tier 모델을 매우 선호한다. 다른 장점은 미들티어가 있을 때에 사용자는 미들티어에 의해 해석되어지는 사용하기 쉬운 하이레벨 API를 가지고 적절한 로우레벨을 호출할 수 있다는 것이다. 결국 많은 경우에서 3-tier 구조는 성능적 이점을 제공할 수 있다.

다) DB Connection 분리

본 연구에서는 JDBC를 사용하여 데이터베이스와 통신한다. DB(DataBase)에 접속을 위한 부분을 완전히 분리하였다. 이렇게 함으로써 DB의 환경 변화나 DB 자체의 변화에 능동적으로 대처하고 빠른 안정화를 유도할 수 있다. 또한 시스템 확장 시에도 DB 접속과 관련하여 새로이 고려하지 않아도 된다.

라) Query문 분리

프로그램에서 Query문을 추출하여 하나의 Class로 생성하였다. 이는 각각의

프로그램에서 Query문의 변경이 요구될 경우 프로그램에서 Query문이 포함된 부분을 찾아 변경하는 것보다 Query문을 모아둔 Class만을 변경함으로써 유지보수 및 System의 정형화에 기여할 수 있다.

마) Business Logic의 구현

본 연구에서는 데이터의 저장 및 조회와 같은 데이터베이스와의 연결이 이루어지는 부분 즉 Business Logic을 Stateless EJB를 사용하여 구현하였다. 이는 모든 Query문을 EJB에서 구현함으로 Query문의 분리를 시행 하였고, JDBC를 통하여 DB Connection을 관리하는 부분을 분리하였으며 Weblogic을 통해 Datasource 및 connection pool을 생성하여 JNDI를 통해 접근 하도록 구성하였다.

(1) Connection pool: Java Programming에서 Database로 Connection을 맺는 일은 매우 느리며 자원을 많이 소모하는 작업이다. 그러므로 불특정 다수의 사용자들이 동시에 Database의 Connection을 요구한다면 최악의 경우 server가 down되기도 한다. 이것을 해결하기 위해 Connection Pool을 이용한다. Connection Pool은 Database와의 연결을 효율적으로 관리하는 역할을 한다. 사전에 일정량의 Connection 객체를 만들어 공유된 장소에 모아둔다. 시간이 걸리는 Connection 객체를 사전에 생성해 둠으로써 속도향상을 기대할 수 있다.

Java Programming에서 사용이 끝난 Connection 객체를 다시 공유된 장소에 넣어둔다. 예전에 이와 같은 Connection Pool을 직접 개발하였으나 최근에는 JDBC Driver 내에 자체적으로 내장되어 있는 경우가 많다. 본 연구에서는 Weblogic에서 제공하는 Connection pool을 사용하였다. 그림 72는 Weblogic에서 Connection pool을 설정하는 화면이다.

■ Name: JDBC Connection pool의 이름을 지정한다. 여기서는 MySQL로 구현하였다.

■ URL: 데이터베이스의 주소와 포트 번호를 나타낸다. URL의 포맷은 JDBC 드라이버에 따라 다르다. 여기서는 MySQL의 JDBC 드라이브의 포맷에 따라 다음과 같이 설정하였다. jdbc:mysql://localhost:3306/WH +Driver Classname: MySQL에서 사용하는 드라이브의 이름을 지정한다. 이것은 이미 MySQL을 위한 class파일이 있어야 한다. org.gjt.mm.mysql.Driver

■ user: 데이터베이스를 사용할 수 있는 계정: user=root

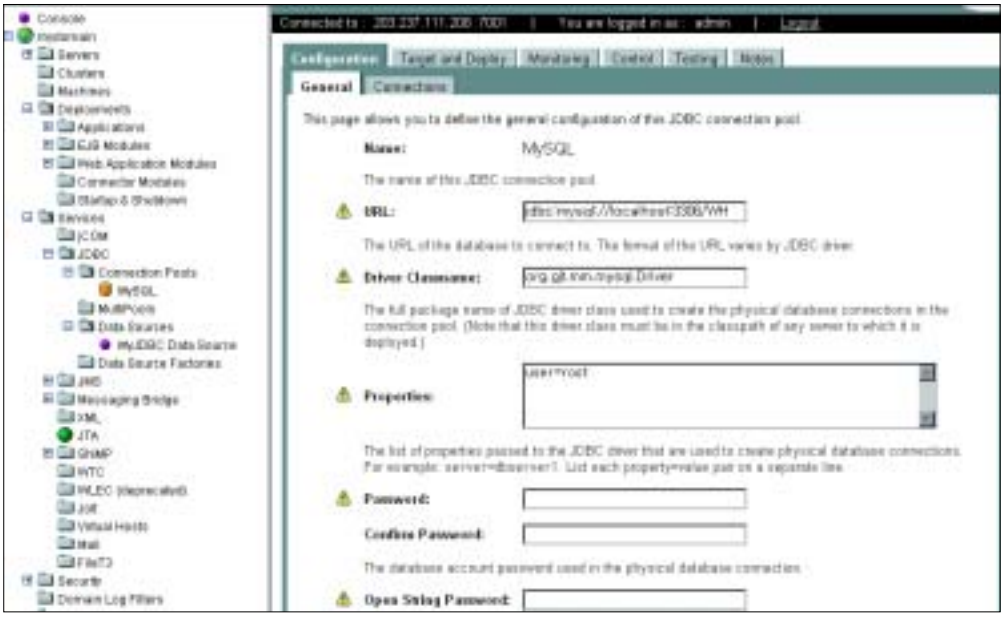

그림 72 Weblogic에서의 Connection pool 설정

(2) Datasource: JDBC 2.0의 javax.sql 패키지에 포함되어 도입되었으며 DataSource 인터페이스는 데이터베이스 커넥션을 만들거나 사용하는데 좀더 유연한 아키텍처를 제공하기 위해 도입되었다. DataSource를 이용할 경우, 클라이언트 코드는 한 줄도 바꾸지 않고서도 다른 데이터베이스에 접속할 수 있도록 해준다. DataSource 인터페이스 구현방식에 따른 클래스 타입들은 다음과 같다.

■ 기본적인 DataSource 클래스: javax.sql.DataSource 인터페이스를 구현한다.

■ 커넥션 풀링 기능을 갖추고 있는 DataSource 클래스:
javax.sql.Connection -PoolDataSource 인터페이스를 구현 한다

■ 분산 트랜잭션을 지원하는 DataSource 클래스:
javax.sql.XADataSource 인터페이스를 구현 한다

그림 73은 Weblogic에서 datasource를 설정하는 화면이다.

■ Name: Datasource의 이름을 정의한다. MyJDBC Data Source

■ JNDI Name: Datasource를 의미하는 JNDI이름. MYSQL_JNDI_WH

■ Pool Name: 이 Datasource와 관계있는 JDBC connection pool을 나타낸다. 이 connection Pool을 이용하여 클라이언트의 요청을 처리한다.

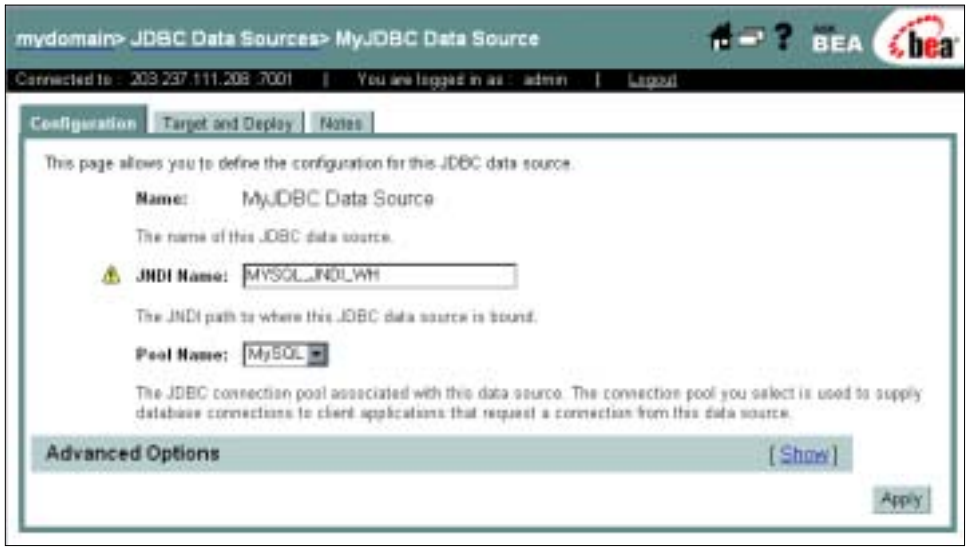

그림 73 Weblogic에서 Datasource 설정화면

바. 데이터 무결성을 위한 방법 연구

본 연구에서는 데이터의 무결성 확보를 위해 온도와 습도의 데이터 저장을 별도의 테이블로 구성하였다. 이는 온도와 습도의 데이터를 별도로 관리하여 서로의 데이터가 연관성을 갖지 않게 함으로써 데이터의 오류를 최소화 하였다. 웹서비스를 사용하여 데이터를 저장하므로 온도와 습도의 저장은 별도의 트랜젝션을 사용한다. 또한 데이터 저장 로직을 하나의 Query문으로 처리함으로써 Rollback의 개념을 배제한 상태에서 단지 Query의 성공 실패 여부만 판단하면 된다. 또한 모든 테이블의 키 값으로 저장고의 번호를 사용하였다. 이는 데이터의 효과적인 처리를 가능하게 한다.

12. 조회 및 관리 프로그램 개발

가. 관리자 관리 프로그램 개발

본 연구에서는 저온저장고 시스템의 전반적인 관리를 위한 프로그램을 Java의 JSP로 구성되어진 웹 기반 프로그램으로 구성하였다. 웹기반 시스템 구조로 인하여 관리자 및 사용자는 언제 어디서나 네트워크가 연결된 곳에서는 시스템에 접속하여 관리할 수 있게 된다. 이는 관리의 안정성과 신속성을 높임과 동시에 관리자 및 사용자의 시스템 운영 및 관리에 있어 시간적 공간적 한계를 벗어날 수 있다.

1) 첫화면

저온저장고 관리 시스템의 컨텐트는 '사업소개', '관리자', '저장고보기', '커뮤니티'로 구성되어져 있다. 그리고 농가에 중요한 정보 중의 하나인 날씨에 대한 정보를 실시간으로 볼 수 있도록 하였다.

그림 74 저온저장고 관리 시스템의 첫화면

2) 관리자 프로그램

‘관리자’ 컨텐트가 관리자를 위한 부분으로서 전체 저온저장고의 상태를 볼 수 있도록 구성되어져 있다. 관리자는 전체 저장고에 대해 제어 권한과 모니터링 권한을 갖는다. 전체 저온저장고의 상태를 한눈에 볼 수 있도록 온도와 습도 그리고 그 밖의 현재 상태를 나타내도록 하였다.

그림 75 관리자 화면

설정 온도와 습도에 대해 사용자 및 관리자가 설정한 경계값을 넘었을 경우 알람 색을 변경하여 관리자에게 상태를 알려 준다. 연두색일 경우에는 정상상태를 적색일 경우에는 상한 값을 초과 한 경우 파란색일 경우에는 하한 값을 초과했을 경우를 나타낸다. ‘전체보기’를 클릭하면 등록되어져 있는 전체 저장고를 볼 수 있도록 하였다. 또한 사용자가 해당 저장고를 클릭 할 경우 저장고에 대한 상세 정보로 넘어가게 된다. 그림 76은 ‘전체보기’를 접속하였을 때 나타나는 화면이다. 동시에 150여개의 저온저장고를 관리할 수 있도록 하였다.

그림 76 전체보기 화면

나. 사용자 조회프로그램 개발

일반 사용자는 자신의 저온저장고에 대하여 제어할 수 있는 권한과 모니터링 할 수 있는 권한을 갖는다. 일반 사용자는 '저장고보기' 컨텐트를 통하여 자신의 저온저장고를 관리할 수 있다. 그림 77은 '저장고보기'의 로그인 화면을 나타내고 있다.

그림 77 저장고 보기의 로그인 화면

그림 78은 저온저장고의 전체 상태를 보여주는 화면으로 기본정보, 경고보기, 현재상태, 설정값 보기, 통계 항목으로 나눌 수 있다. '기본정보' 항목은 저온저장고의 소유주 등 저온저장고의 기본적인 정보를 알려준다. 이 정보 값들은 관리자

에 의해서만 변경이 가능하다. ‘경고보기’ 항목은 저장고의 컨트로러, 클라이언트, 서버에 대한 제어 정보를 알려주는 부분이다. 저장고의 컨트롤러에 이상이 있거나 클라이언트 프로그램과 컨트롤러간의 통신 이상, 서버와 클라이언트간의 통신 이상 등을 나타내는 부분이다.

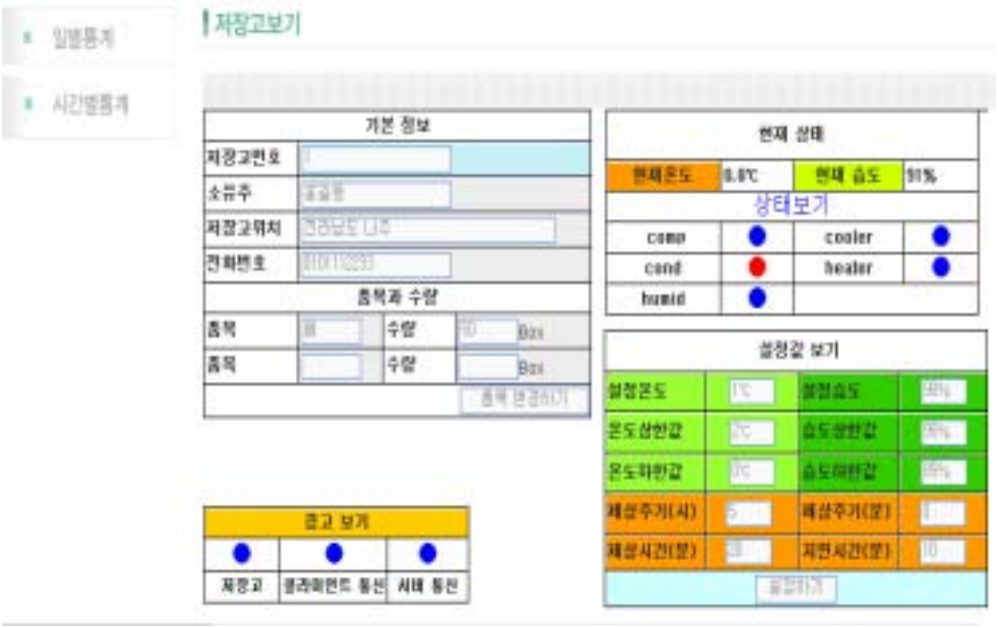

그림 78 저장고 보기 화면

‘현재상태’ 항목은 현재온도 현재습도 그리고 각각의 기기들의 상태를 알려주는 부분이다. 이 부분은 정해진 시간 마다 리로딩(reloding)을 통해서 최신의 데이터를 보여준다.

‘설정값 보기’ 항목은 컨트롤러에 대해 제어 기준이 되는 값을 설정하고 설정된 값을 확인하기 위한 부분이다. 관리자나 저온저장고 소유주가 각 과일 및 농산물의 특성에 따라 적정한 온도 및 습도를 설정하면 그에 따라 저온저장고의 컨트롤러 등이 작동하게 된다.

‘통계’ 항목은 개별 저온저장고의 일별 시간별 통계치를 볼 수 있다. 일별 통계치는 일일 변화량을 시간 단위로 볼 수 있으며 시간별 통계치는 해당 시간동안 저온저장고에서 서버로 보낸 전체데이터를 볼 수 있다.

1) 일별통계

일별 통계는 연 월 일을 선택할 수 있으며 선택되어진 연 월 일의 데이터를 시간별 평균치로 보여준다. 이를 통하여 하루 동안의 온도와 습도 변화량의 세밀한 관찰은 할 수 없으나 전반적인 온도 습도 변화의 패턴을 파악할 수 있다.

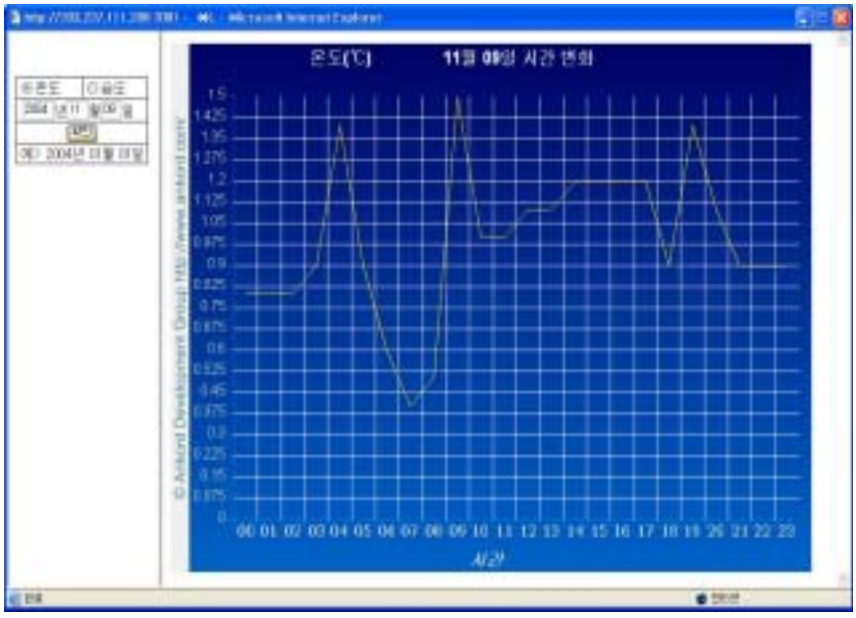

그림 79 일별 온도

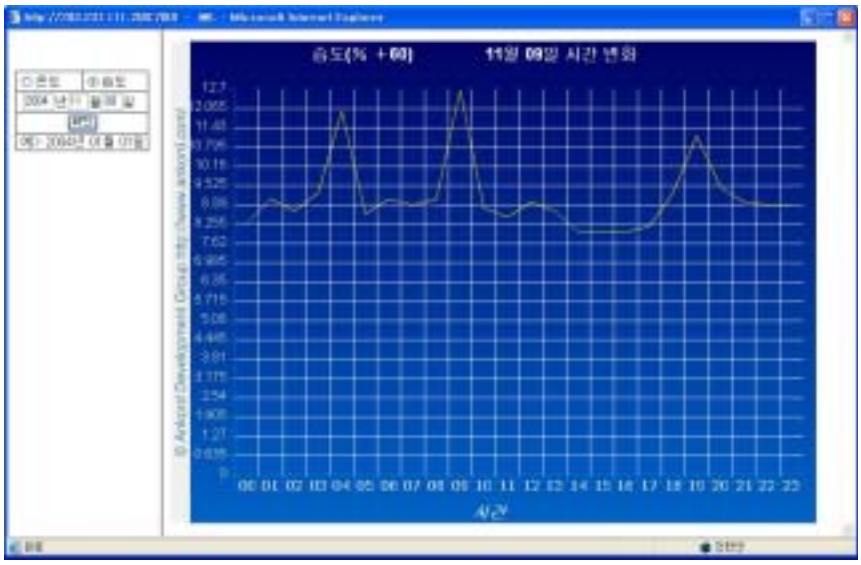

그림 80 일별 습도

2) 시간별 통계

시간별 통계는 연 월 일을 선택하고 시간을 선택할 수 있게 설계하였다. 선택되어진 연 월 일 시간동안 저온저장고에서 서버로 보낸 모든 데이터를 확인할 수 있다. 이를 통하여 온도 및 습도의 세밀한 변화를 관찰할 수 있다.

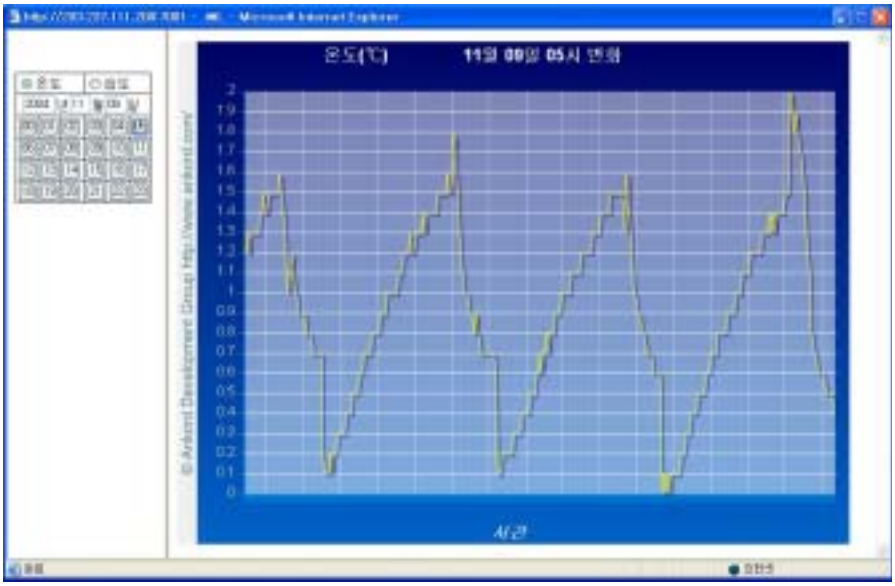

그림 81 시간별 온도

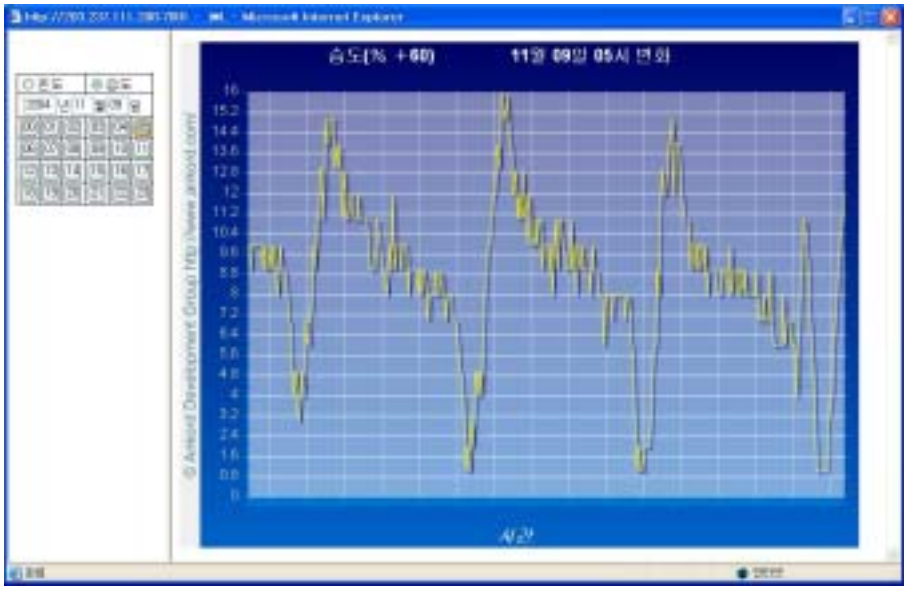

그림 82 시간별 습도

13. 시작품제작

가. 제어기 설계 및 제작

1) 하드웨어 설계 및 제작

가) 회로설계

그림 83은 본 연구를 위해 설계한 제어시스템의 전체 회로도 이다.

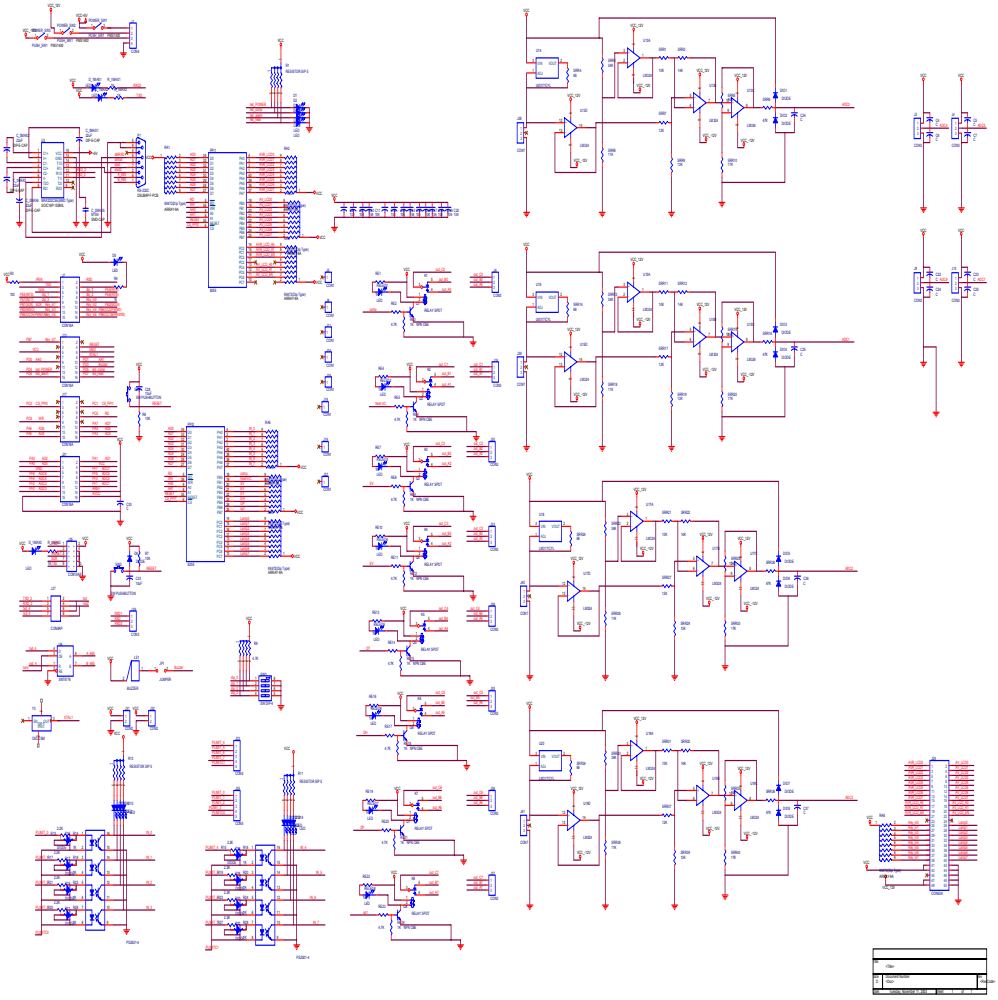

그림 83 전체 회로도

나) 기판설계

그림 84는 본 연구에서 사용된 PCB 기판을 나타내고 있다.

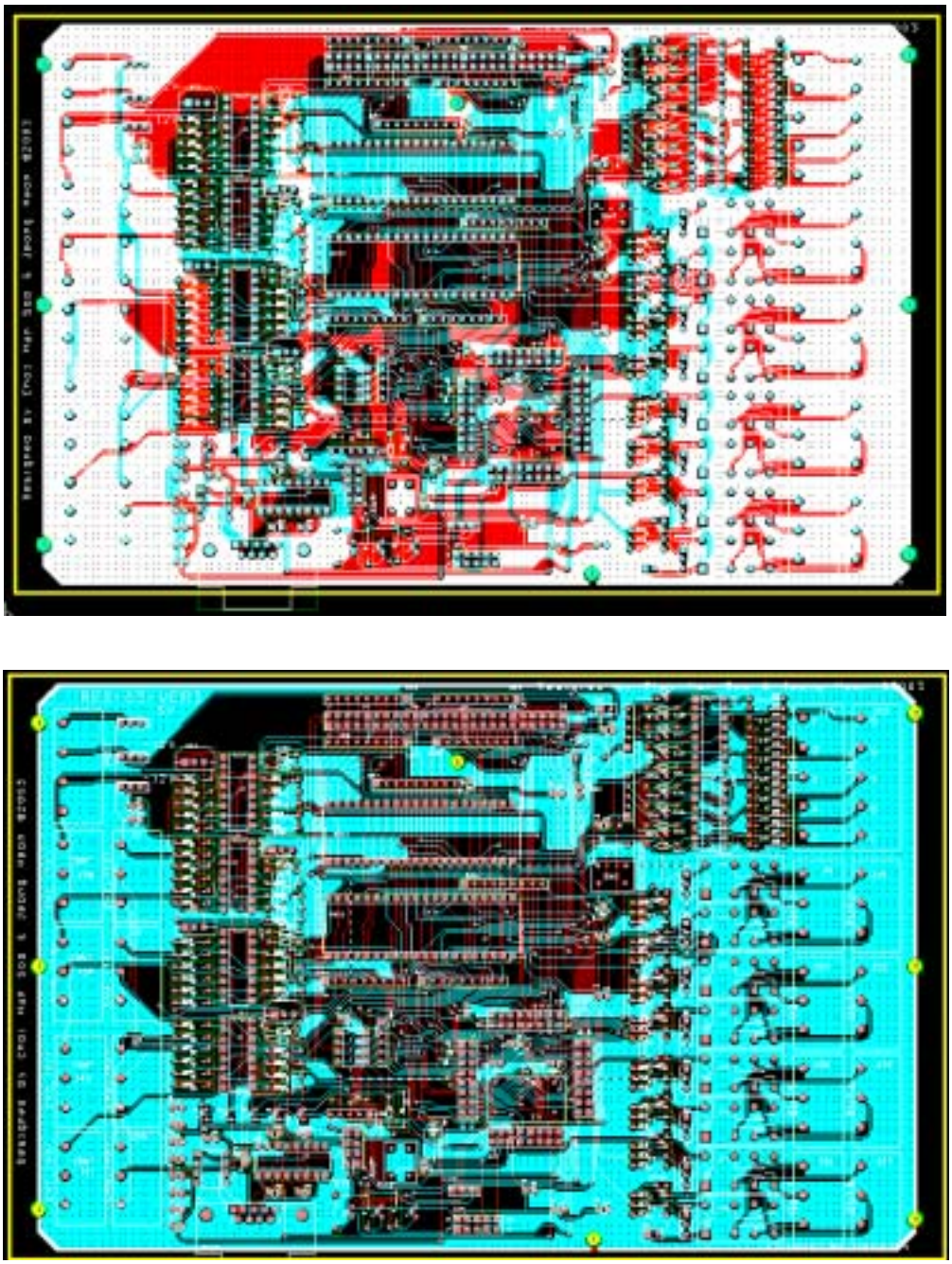

그림 84 PCB기판

다) 하드웨어 제작

그림 85는 회로도와 PCB를 기반으로 제작된 하드웨어를 나타내고 있다.

그림 85 하드웨어

라) 제어 측정 시스템 외형

그림 86 제어 측정 시스템 외형

마) 제어 측정 시스템 내부

그림 87 제어 측정 시스템 내부

2) 소프트웨어 개발

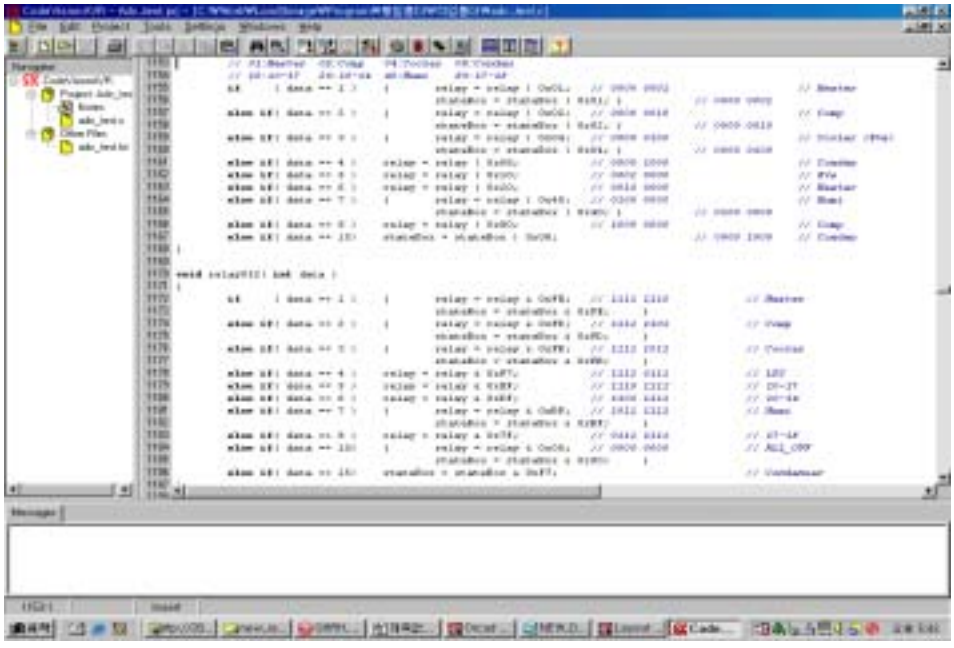

그림 88 소프트웨어 개발 툴

나. Client 화면

그림 89 Client 화면

다. Server 화면

그림 90 Server 화면

라. 실험 결과

본 연구에서는 원격지에 있는 저온저장고를 HTTP기반의 SOAP 통신을 통하여 제어하고 모니터링 하는 시스템을 구축하였다. 본 연구는 원격지에 있는 저온저장고를 제어하기 위해 일반 인터넷 망을 사용하였다. 따라서 데이터의 안정적 전송과 전송속도에 대한 실험을 통하여 본 시스템의 가용성을 알아보았다.

1) 실험 조건

- 일시: 2004년 11월 14일 10:00 ~13:00
- 클라이언트: 전라남도 나주시 나주배시험장
- 서버: 조선대학교 인공지능실험실
- 통신 주기: 10초

2) 데이터 전송속도 실험

본 연구에서는 클라이언트와 서버에 각각 시간과 측정값을 저장하여 시간 지연을 측정하였다. 또한 서버와 클라이언트의 시간 동기를 위해 한국표준과학연구원에서 제공하는 UTCk3을 사용하였다. UTCk3은 설정되어진 주기 마다 인터넷을 통해 한국표준과학연구원의 서버에 접속하여 컴퓨터의 시간을 동기화 한다. 따라서 서버와 클라이언트 PC에 각각 UTCk3을 설치하여 서버와 클라이언트를 동기화 하였다.

3시간 동안(10:00~13:00)의 데이터를 저장하여 분석한 결과 클라이언트와 서버간의 데이터 전송에 있어 시간지연은 1초 이상의 지연은 발생하지 않았으며 총 1080개의 데이터 중 13개에서 지연이 발생하였다. 이는 약 1.2%의 최대 1초 지연 데이터 발생으로 저온저장고 시스템에 대한 모니터링 및 제어를 효과적으로 수행할 수 있음을 보여주고 있다.

그림 91은 클라이언트와 서버간의 Ping 테스트로 서버와 클라이언트간의 통신은 최대 1초을 넘지 않으며 일반적으로 100ms 이하임을 나타내고 있다. 이는 농산물의 감시 및 제어 시스템에서는 서버와 클라이언트간에 1초의 지연이 허용되어지므로 본 시스템이 효율적으로 사용되어 질 수 있음을 보여주고 있다.

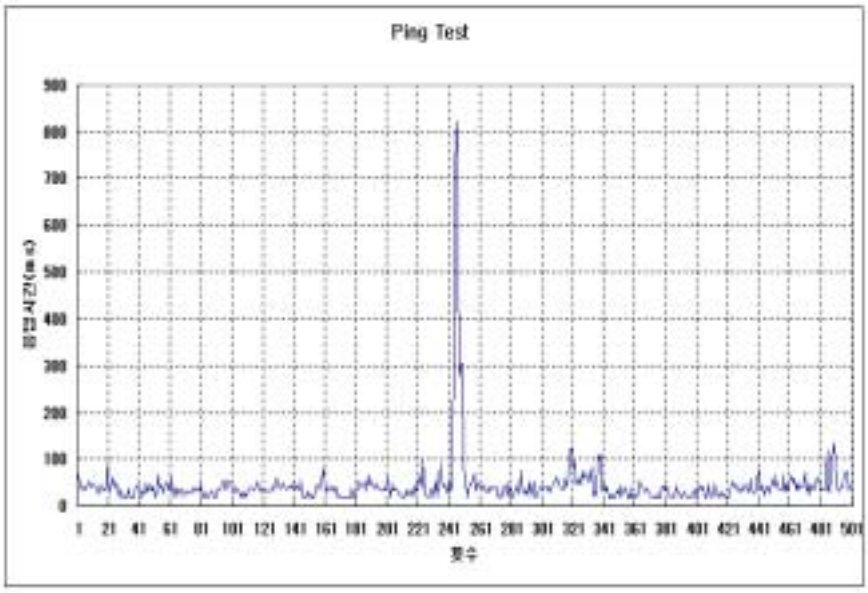

그림 91 클라이언트 서버간의 PING 테스트

3) 온・습도 제어실험

그림 92는 컨트롤러에서 전송되어지는 온도와 상대습도의 변화를 나타내고 있다. 그림에서 알 수 있듯이 일정 온도 이상일 경우 제어되어지는 것을 볼 수 있다. 습도 역시 일정 습도 이하에서 제어 되어지고 있음을 알 수 있다.

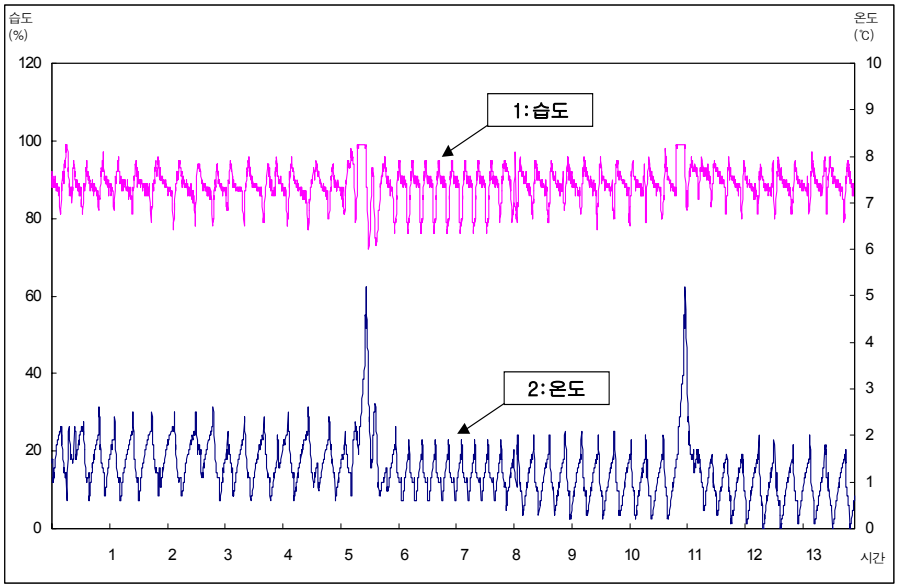

그림 92 시간에 따른 온・습도 변화 그래프

제 4 장 목표달성도 및 관련분야에의 기여도

제 1 절 연구개발 목표의 달성도

연 구 개 발 목 표	달 성 도
1. 자료조사	완 료
2. 제어시스템 구성	완 료
3. 측정시스템 구성	완 료
4. 제어·측정시스템과 PC간 통신 프로토콜	완 료
5. PC기반 제어 프로그램 개발	완 료
6. 저온저장고 온·습도 제어실험	완 료
7. 데이터베이스 구축	완 료
8. Client와 Server간 프로토콜	완 료
9. Client용 통신프로그램 개발	완 료
10. Server용 프로그램 개발	완 료
11. 조회 및 관리 프로그램 개발	완 료
12. 시작품제작	완 료

제 2 절 관련분야에의 기여도

정보통신기술의 발달과 무선 공중 통신망의 확충 그리고 인터넷 통신이 가능하도록 초고속 광케이블 가설지역이 면 단위까지 확대됨에 따라 국내 정보통신 관련 인프라는 확보되어 있는 상태이다. 따라서 정보화는 국내 산업전반에 확대 적용될 것이며 농업관련 산업 역시 예외일 수는 없다. 저온저장, 저온수송의 환경관리, 유리 온실의 발육 환경관리 등에 정보 통신을 이용한 원격 감시 및 제어 시스템을 도입하는 것은 자연스럽고 당연한 일이다.

본 연구에서 제안한 인터넷을 이용하여 원거리에서 중앙 집중 관리하는 중앙 집중 감시 및 제어 시스템은 전문가가 저온저장고를 위탁 관리하는 시스템으로서 저온저장고의 환경을 최적의 상태로 유지 관리할 수 있어서 저온저장고의 관리상의 어려움을 해결함과 동시에 효율을 높일 수 있을 것이다.

앞으로 저온저장고의 관리는 농가에서 직접 관리하는 것보다는 위탁 관리를 받은 전문 업체에서 전문가에 의해 원격 감시 및 제어하는 체제로 전환되어질 것으로 전망된다. 따라서 본 시스템의 개발은 저온저장고를 위탁 관리할 수 있는 기반을 조성하는 데에 기여하게 된다.

1. 기술적 측면

가) 컴퓨터로 저온저장고의 환경상태를 감시하고 최적의 환경상태를 유지할 수 있도록 온·습도를 원격 제어할 수 있는 시스템을 농가에 설치하면, 저장고까지 가지 않고도 농가의 거실이나 안방에서 저장고의 환경을 감시하고 제어할 수 있다.

나) 온·습도 측정에 있어 기존의 하나의 센서에 의존하는 방식에서 다수의 센서를 사용하여 저온저장고 내부 공간 여러 곳의 환경을 측정함으로써 신뢰성 있는 감시 및 제어를 수행할 수 있다.

다) 저온저장고에 대한 전문지식이 없는 농가에서 저장고를 관리하는 것보다는 전문가가 인터넷을 이용하여 각 농가에 설치된 저온저장고의 환경 상태를 감시하고 최적의 환경으로 제어할 수 있는 인터넷을 이용한 원격 감시 및 제어

기능을 갖는 중앙 집중 저온저장고 관리시스템을 구축하여 저장고 관리의 효율성을 높일 수 있는 기술을 개발함으로써 저온저장 분야의 기술을 향상시킬 수 있다.

라) 저온저장고에 대한 컴퓨터 제어 계측시스템 기술과 온·습도 관련 인터페이스 기술을 개발하여 저온저장 기술 수준을 향상시킬 수 있다.

마) 인터넷을 이용한 원격 감시 및 제어기술, 인터페이스 관련기술 등은 본 연구 결과 얻어질 수 있는 기술이다. 이 기술들은 정보화 시대에 산업전반에 적용되는 기술로서 본 연구 결과의 성과물이 타 분야에 미치는 파급효과 역시 기대된다.

2. 경제 · 산업적 측면

가) 농산물의 신선도를 유지하여 상품을 고급화하고 수확 후 상태를 장기간 유지시킴으로써 출하시기를 조절하여 가격이 높은 시기에 출하할 수 있어서 농가소득을 높일 수 있다.

나) 우리나라는 인터넷 인프라 구축 및 보급률이 높은 정보 선진국이다. 이러한 장점을 충분히 활용하여 저온저장고에 원격 감시 및 제어 시스템을 구축함으로써 과수 농산물의 품질을 고급화할 수 있어 수입 상품과 차별화 할 수 있다.

다) 저온저장고의 관리 잘못으로 인한 냉해를 막아 농가의 손실을 줄일 수 있다.

라) 저온저장고를 보유한 농가는 저장고를 편안하고 안전하게 관리할 수 있다.

마) 중앙 집중 관리시스템은 안전함은 물론 전문가의 관리를 받을 수 있어 저장고의 환경 상태를 최적의 상태로 유지할 수 있다.

제 5 장 연구개발결과의 활용계획

1. 본 연구에 의해 개발된 핵심기술은 인터넷을 이용한 원격 감시 및 제어 기술이다. 인터넷을 이용한 원격 감시 및 제어 기술은 인터넷 통신망을 통해 원거리에서 저온저장고의 환경상태를 감시하고 최적의 환경상태를 유지할 수 있도록 온·습도 장치를 원격 제어하는 기술이다.
2. 활용분야는 저온저장고를 관리하는 유통 분야에 적용할 수 있다.
3. 활용유형은 주관기관 명의로 특허를 출원하고, 관련업체에 기술을 이전하여 상품화한 후, 저온저장고를 보유한 농가 또는 관리전문 업체에 상품을 보급하여 농가의 애로사항을 해결할 수 있도록 한다.
4. 활용방안은 저온저장고를 보유한 농가에서 관리하는 방안과, 저온저장고의 관리를 관리 전문업체에 위탁하여 위탁받은 전문업체가 원격 중앙 집중관리를 위해 활용하는 방안이 있다.
5. 현장보급 방안 및 산업화계획 방안은 기술을 이전 받은 업체가 산업화하며 개발된 상품을 농가와 관리 대행업체에 보급한다.
6. 추가기술 방안 및 기술이전 방안은 정보 통신 관련업체 또는 저온저장고를 시공하는 업체에 기술을 이전한 후 현장감각에 맞도록 추가되는 기술은 기술을 이전 받은 업체가 현장감을 살린 기술을 추가로 개발한다.

제 6 장 연구개발과정에서 수집한 해외과학기술 정보

해당사항 없음

제 7 장 참고문헌

1. 원예 생산물 저장, 농민신문사, 박윤문, 이승구, 1997
2. 원예산물 수확 후 관리, 농촌진흥청, 2001
3. Building Intranets with Intermet Information Server and Frontpage, 사이버 출판사, Steve Waterhouse, 1997
4. Intranets Unleashed, 정보문화사 David Garrette, 1997
5. 인트라넷 구축과 관리, 성안당, Tobin Anthony, 1997
6. 웹서버를 이용한 인트라넷 구축하기, 인포북, Desborough, 1997
7. Matering Internets, 성안당, Coleman Dyson, 1997
8. Internet Explorer 4.0, 도서출판 문화사, 이준영, 1997
9. 인터넷 파워 유틸리티 101, 사이버출판사 고인현, 1997
10. PC통신 사용자를 위한 인터넷 홈페이지 만들기, 인포북, 김형로, 1997
11. WEB programing, 성안당 Laura Lemag, 1997
12. Windows 95용 웹사이트 구축, 도서출판 대림, Scott Zimmerman, 1997
13. 인터넷 인증시험 바이블, 도서출판 대림 김정환외 3인, 1997
14. Netscape Server Source Book, Wiley, Ted Coombs, 1997
15. 원예산물의 고품질 저장을 위한 신기술·신소재, 한·일 심포지엄 발표자료, 농업기계화 연구소, 2003

부 록
(메뉴얼)

홈페이지

1. 접속방법

인터넷 주소 창에 http://203.237.111.208/index.jsp 로 접속한다.

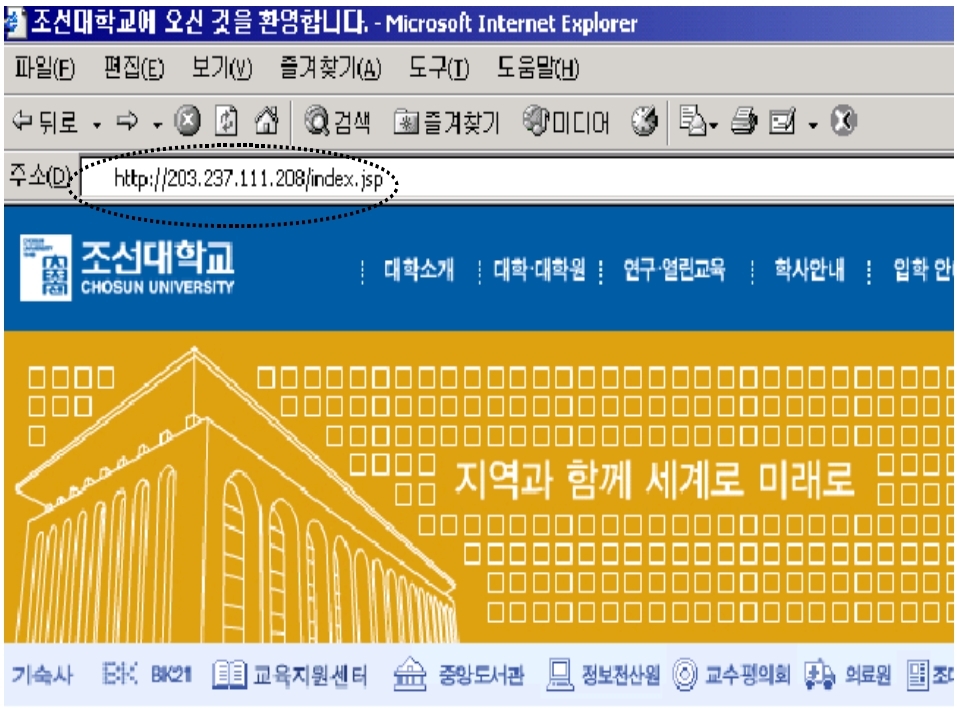

그림 93 인터넷 접속 방법

2. 초기화면

그림 94 초기화면

저온저장고 홈페이지를 접속하면 그림 94와 같은 초기화면이 나타난다.

3. 홈페이지 화면 구성

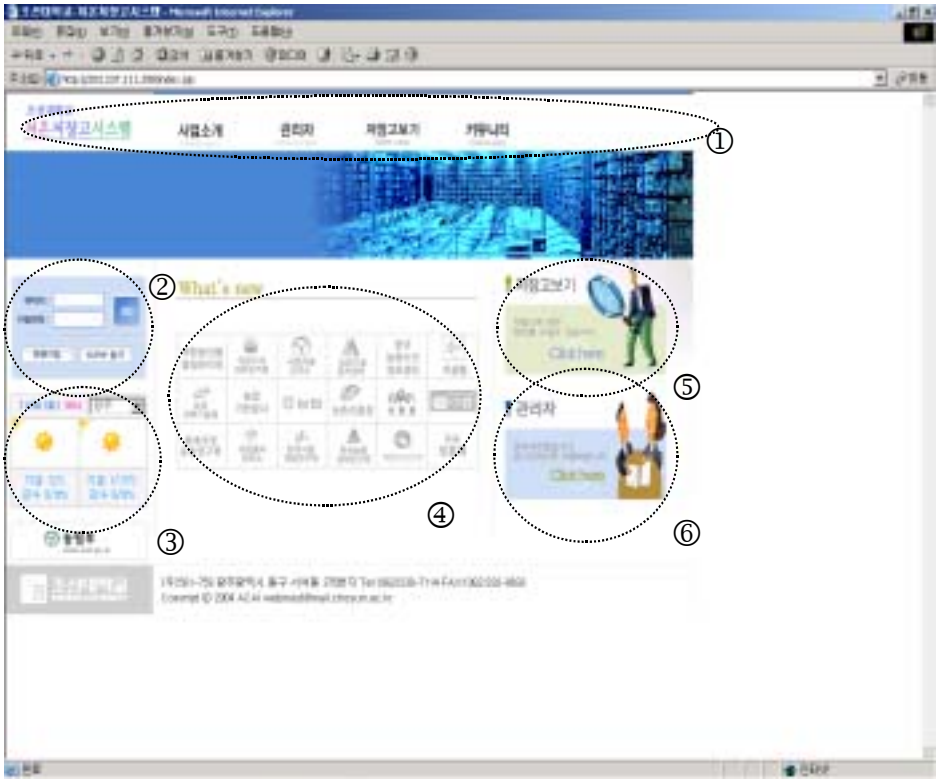

그림 95 홈페이지 화면 구성

초기화면은 그림 95와 같이 6개의 항목으로 구성되어 있다.

가. ①번 항목(메인메뉴)

그림 96 메인메뉴

①번 항목은 그림 96과 같이 홈페이지 메인메뉴 화면이다. 메인메뉴는 '사업

소개', '관리자', '저장고보기', '커뮤니티'로 구성되어 있다. 각 메인메뉴에 대한 서브메뉴는 다음과 같다.

메인메뉴	서브메뉴
사업소개	인사말, 사업개요, 연혁
관리자	저장고상태보기, 공지사항관리, Q&A관리, 자유게시판관리
저장고보기	일별통계, 시간별통계
커뮤니티	공지사항, Q&A, 자유게시판

1) 사업소개

그림 97 사업소개 화면(사업소개/인사말)

메인메뉴 '사업소개' 란에는 사업자의 인사말, 사업개요, 연혁을 수록한다.

2) 관리자

메인메뉴 '관리자'란은 서버를 관리하는 관리자가 조작하는 페이지로 아이디

창과 비밀번호 창에 관리자의 ID와 비밀번호로 로그인하여 접근할 수 있다.

그림 98 관리자 화면(관리자/)

가) 저장고 상태보기

그림 99 저장고상태보기 화면(관리자/저장고상태보기)

그림 98에서 관리자의 아이디와 비밀번호로 접속을 하면 그림 99와 같은 창이 나타난다.

메인메뉴 '관리자'란에는 네 개의 서브메뉴 '저장고상태보기' '공지사항관리' 'Q&A관리' '자유게시판관리'가 있다. 그림 99의 우측 상단에 있는 전체보기를 클릭하면 다음과 같은 창이 나타난다.

나) 전체보기

그림 100 전체보기 화면(관리자/저장고상태보기/전체보기)

위 화면은 서버에 연결되어 있는 저장고들의 전체 상태를 관리할 수 있도록 모든 저장고들의 상태를 보여주고 있다. 각 저장고들의 해당 숫자를 클릭하면 그림 101과 같은 창이 나타난다.

다) 저장고 보기

그림 101은 해당 저장고의 상태를 나타낸다. 이 곳에서 각 저장고에서 보관하고 있는 농산물의 품목, 수량, 현재의 환경상태, 각종 설정값, 경고 여부 등을 확인할 수 있다.

그림 101 저장고보기(관리자/저장고상태보기/해당숫자)

3) 저장고보기

그림 102 저장고보기 화면(저장고보기/)

저장고보기 화면은 아이디와 비밀번호를 부여받은 사용자가 PC상에서 저장고의 상태를 확인할 수 있는 화면이다. 아이디와 비밀번호를 입력하면 다음과 같은 창이 나타난다.

그림 103 저장고보기 로그인 화면

그림 103 화면에서 저장고 내에 있는 농산물의 품목, 수량 등을 수정할 수 있다. 설정온도와 설정습도 값을 본 화면 상에서 수정하여 원거리 제어 기능을 수행할 수 있다.

좌측의 서브메뉴 '일별통계'와 '시간별통계'를 클릭하면 다음과 같은 창이 나타난다.

가) 일별통계

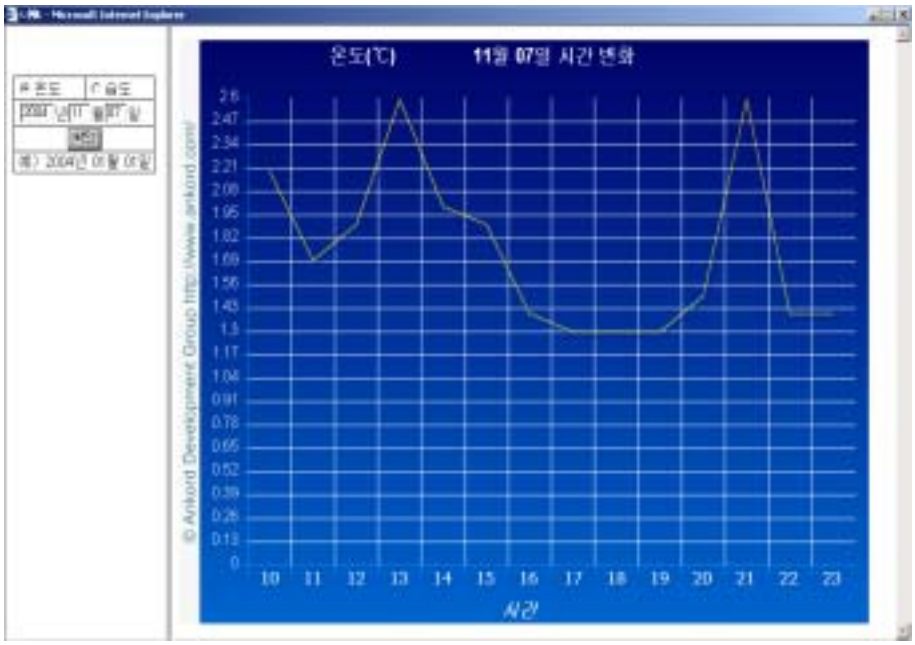

그림 104 일별 온도

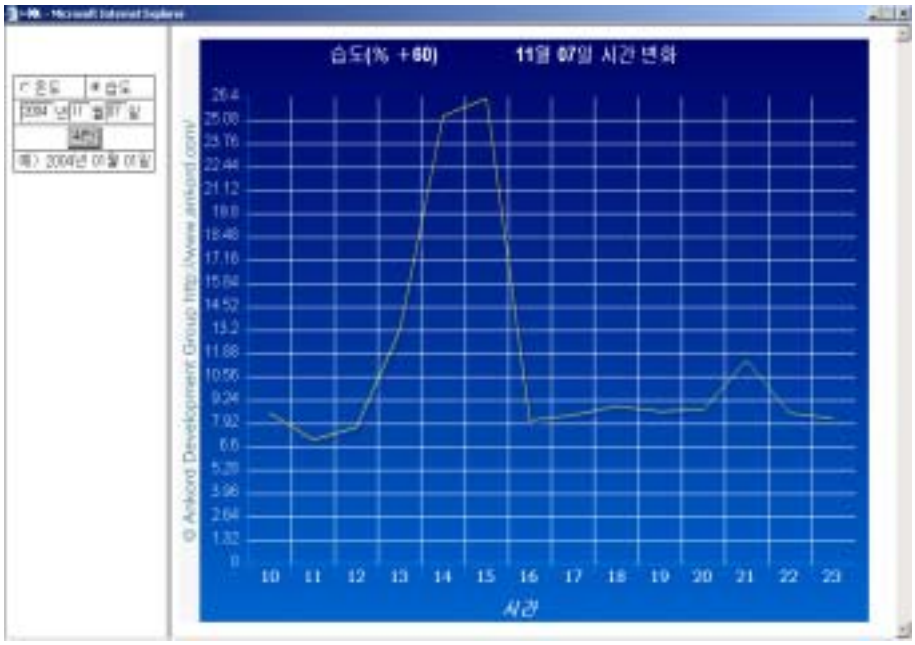

그림 105 일별 습도

나) 시간별통계

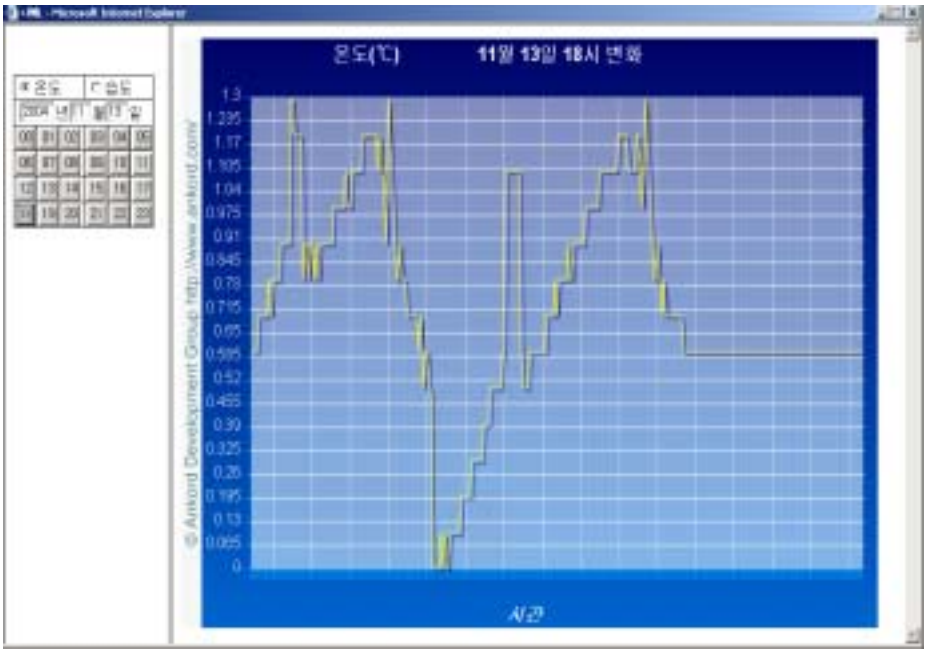

그림 106 시간별 온도

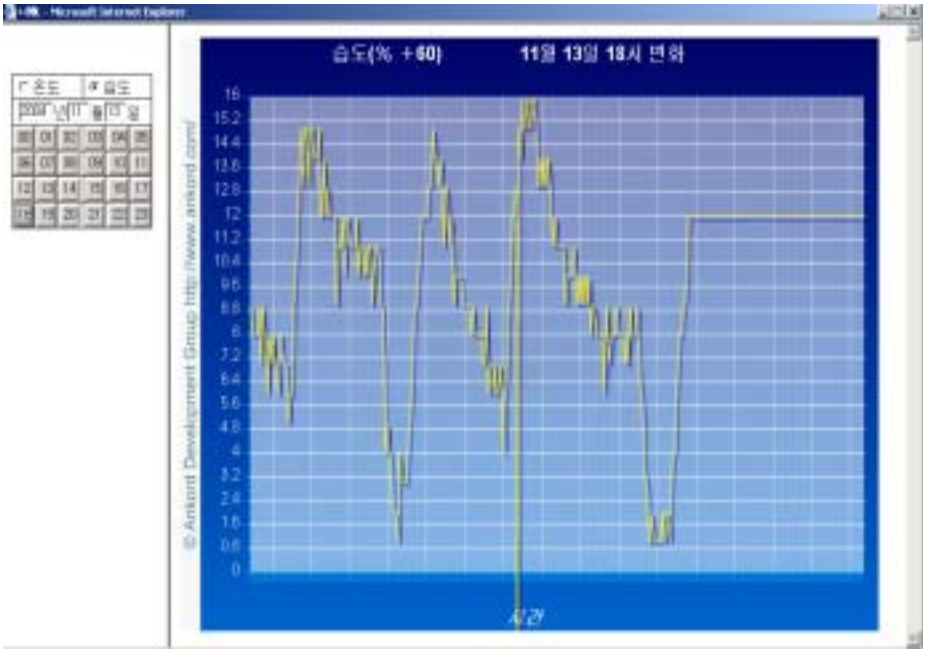

그림 107 시간별 습도

4) 커뮤니티

그림 108 커뮤니티 화면(커뮤니티/공지사항)

커뮤니티 메인메뉴는 공지사항, Q&A, 자유게시판의 부메뉴를 갖는다.

나. ②번 항목

그림 109 Login 화면

②번 항목은 접근경로를 나타낸다. 회원가입을 통해 아이디와 비밀번호를 부여받은 사용자는 ②번 항목을 통하여 본 웹페이지를 자유롭게 사용할 수 있다. 비회원은 회원가입을 클릭하여 회원으로 가입할 수 있고, 아이디와 비밀번호를 잊어버린 회원은 ID/PW 찾기를 클릭하여 아이디와 비밀번호를 찾을 수 있다.

다. ③번 항목

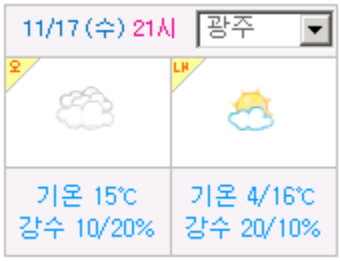

그림 110 기상정보

③번 항목에서는 실시간 전국의 기상정보를 알 수 있다.

그림 111 기상정보2

해당 지역을 클릭하여 전국 원하는 곳의 기상정보를 알 수 있으며. 기상 그림 밑의 기온(15℃) 강수(10/20%)를 클릭하면 기상정보 웹사이트로 직접 연결된다.

그림 112 연결된 기상정보 사이트

라. ④번 항목

그림 113 농업관련 홈페이지

④번 항목은 농업과 관련된 웹 사이트들을 모아둔 곳이다. 각각의 사이트를 클릭하면 해당 홈페이지로 접속된다.

마. ⑤번 항목

그림 114 저장고보기 바로가기

⑤번 항목은 ①번 항목의 메인메뉴 '저장고보기'의 바로가기 이다.

바. ⑥번 항목

그림 115 관리자 바로가기

⑥번 항목은 ①번 항목의 메인메뉴 '관리자'의 바로가기 이다.

컨트롤러

1. 컨트롤러의 외형

그림 116 컨트롤러의 외형

저온저장고용 컨트롤러의 외형은 그림 116과 같다.

① 운전/정지 스위치

컨트롤러의 전원을 운전/정지(ON/OFF)하는 스위치

② 설정화면

고내의 온도와 습도의 설정값과 현재값을 보여 주는 창

③ 상태화면

고내의 온도와 습도 상태, 기기들의 동작 상태, 제상주기 제상시간 지연시간 상태, EVA 모드 상태, 센서모드 상태, 보정값 상태, 데드존 상태 등을 보여 주고 수정하는 창. 각각의 상태들을 보거나 설정값을 수정할 때 Up(▲)버튼과 Down (▼)버튼을 사용한다.

상태화면은 각 페이지로 구성 되어있으므로 해당 상태화면을 Up(▲)버튼과 Down(▼)버튼을 이용하여 볼 수 있음.

설정값을 수정할 때는 해당 화면을 Up(▲)버튼과 Down(▼)버튼으로 찾은 후 '수정버튼'을 누르고 설정값을 수정하고 '완료버튼'을 누른다

④ Up(▲) 버튼

각 상태화면의 상태들을 보거나 설정값을 수정할 때 사용

⑤ Down(▼) 버튼

각 상태화면의 상태들을 보거나 설정값을 수정할 때 사용

⑥ 수정 버튼

각 설정 값들을 수정하기 위해 수정 직전에 사용하는 버튼

⑦ 완료 버튼

각 설정 값들을 수정한 후 수정 값들을 입력하는 버튼

⑧ 습도 램프

습도기의 동작상태를 표시하는 램프

- 평상시에는 설정화면의 S와 A가 대문자로 표시, 설정값 수정 시에는 s와 a가 소문자로 표시됨

- 모든 상태화면에서 수정 후에 완료버튼을 눌러 주어야함

2. 컨트롤러 조작방법

가. 초기화면

그림 117 전원을 ON 시켰을 때의 초기 설정화면

그림 118 전원을 ON 시켰을 때의 초기 상태화면

그림 119 그림 117이후 초기 설정화면

그림 120 그림 118이후 초기 상태화면

그림 119는 초기 설정화면이며 그림 120은 초기 상태화면이다. 평상시에는 이 화면을 유지한다.

초기 설정화면

설정 온도와 현재 온도를 표시, 설정 습도와 현재 습도를 표시

S는 설정(setting) 값을 의미함

A는 실행(application) 값, 즉 현재 값을 의미함

초기 상태화면 : COMP.와 HUMI.(습도), HEAT.(히터), EVA.(증발기)의 동작 상태.

'*' 표시는 해당 기기가 동작 중임을 의미함

나. 고내 기준온도 설정

기준온도 설정 창 선택→수정버튼→▲▼버튼으로 수정→완료버튼→기준온도 설정 창

■ Up(▲)버튼과 Down(▼)버튼을 이용하여 상태화면에서 '기준온도 설정 창 선택'

상태화면에 기존의 기준온도가 표시되어 있음

그림 121 기준온도 설정 창

그림 122 기준온도 설정 창

그림 123 온도수정 상태화면

■ ‘수정버튼’을 누르면 그림 121 또는 그림 122의 상태화면이 그림 123의 상태화면으로 바뀜

■ 이 화면에서 ‘Up(▲)버튼과 Down(▼)버튼’으로 온도값을 수정함

o 02.0(현재설정온도) n 02.0(설정할 온도)

n 02.0에서 새로운 기준온도가 설정됨. 02.2로 설정함

■ 새로운 설정값의 입력을 위해 ‘완료버튼’을 누름

■ 완료버튼을 누르면 ‘기준온도 설정 창’으로 복귀함

■ 설정화면에 새로 설정한 기준온도가 ‘그림 119 초기 설정화면’의 S에 표시됨

다. 고내 습도설정

습도설정 창 선택→수정버튼→▲▼버튼으로 수정→완료버튼→습도설정 창

■ Up(▲)버튼과 Down(▼)버튼을 이용하여 상태화면에서 ‘습도설정 창 선택’ 상태화면에 기존의 기준습도가 표시되어 있음

그림 124 습도설정 창

그림 125 습도수정 상태화면

■ ‘수정버튼’을 누르면 그림 124의 상태화면이 그림 125의 상태화면으로 바뀜

■ 이 화면에서 ‘Up(▲)버튼과 Down(▼)버튼’으로 습도값을 수정함

o 90%(현재 설정습도) n 90%(설정할 습도)

n 90%에서 새로운 기준습도가 설정됨. 92%로 설정함

■ 새로운 설정값의 입력을 위해 ‘완료버튼’을 누름

■ 완료버튼을 누르면 ‘습도설정 창’으로 복귀함

■ 설정화면에 새로 설정한 습도가 ‘그림 119 초기 설정화면’의 S에 표시됨

라. 제상주기(시간) 설정

그림 126은 제상주기(I), 제상시간(O), 제상 후 지연시간(D)을 설정하는 화면이다.

제상주기(I)는 시와 분을 구분하여 설정하도록 설계하였음. 이 란에서는 시를 설정함.

제상주기(시)설정 창 선택→수정버튼→▲▼버튼으로 수정→완료버튼→제상주기(시)설정 창

■ Up(▲)버튼과 Down(▼)버튼을 이용하여 상태화면에서 ‘제상주기(시)설정 창 선택’

상태화면에 기존의 기준제상 주기(시)가 표시되어 있음

■ ‘수정버튼’을 누르면 그림 126의 상태화면이 그림 127의 상태화면으로 바뀜

■ 이 화면에서 ‘Up(▲)버튼과 Down(▼)버튼’으로 제상주기(시)를 수정함

o 05:00(현재 설정제상주기) n 05:00(설정할 제상주기)

n 05:00에서 새로운 재상주기(시)가 설정됨. 06:00으로 설정함

■ 새로운 설정값의 입력을 위해 ‘완료버튼’을 누름

■ 완료버튼을 누르면 ‘제상주기설정 창’으로 복귀함

그림 126 제상주기(시) 설정 창

그림 127 제상주기(시) 수정 상태화면

마. 제상주기(분) 설정

제상주기는 시간과 분을 구분하여 설정하도록 설계하였음. 이 란에서는 분을 설정함

제상주기(분)설정 창 선택→수정버튼→▲▼버튼으로 수정→완료버튼→제상주기(분)설정 창

■ Up(▲)버튼과 Down(▼)버튼을 이용하여 상태화면에서 '제상주기(분)설정 창 선택'
상태화면에 기존의 기준제상 주기(분)가 표시되어 있음
■ '수정버튼'을 누르면 그림 128의 상태화면이 그림 129의 상태화면으로 바뀜
■ 이 화면에서 'Up(▲)버튼과 Down(▼)버튼'으로 제상주기(분)를 수정함

o 05:00(현재 설정제상주기) n 05:00(설정할 제상주기)
n 05:00에서 새로운 재상주기(분)가 설정됨. 06:00으로 설정함

■ 새로운 설정값의 입력을 위해 '완료버튼'을 누름
■ 완료버튼을 누르면 '제상주기설정 창'으로 복귀함

그림 128 제상주기(분) 설정 창

그림 129 제상주기(분) 수정 상태화면

바. 제상시간 설정

제상시간은 분 단위로 설정한다.

제상시간(분)설정 창 선택→수정버튼→▲▼버튼으로 수정→완료버튼→제상시간(분)설정 창

■ Up(▲)버튼과 Down(▼)버튼을 이용하여 상태화면에서 '제상시간(분)설정 창 선택'

상태화면에 기존의 기준제상 시간(분)이 표시되어 있음

■ ‘수정버튼’을 누르면 그림 130의 상태화면이 그림 131의 상태화면으로 바뀜

■ 이 화면에서 ‘Up(▲)버튼과 Down(▼)버튼’으로 제상시간(분)을 수정함

o 00:30(현재 설정제상시간) n 00:30(설정할 제상시간)

n 00:30에서 새로운 재상시간(분)이 설정됨. 00:20으로 설정함

■ 새로운 설정값의 입력을 위해 ‘완료버튼’을 누름

■ 완료버튼을 누르면 ‘제상시간 설정 창’으로 복귀함

그림 130 제상시간 설정 창

그림 131 제상시간 수정 상태화면

사. 제상 후 지연시간 설정

제상 후 지연시간은 분 단위로 설정한다.

제상 후 지연시간(분)설정 창 선택→수정버튼→▲▼버튼으로 수정→완료버튼→제상 후 지연시간(분)설정 창

■ Up(▲)버튼과 Down(▼)버튼을 이용하여 상태화면에서 '제상 후 지연시간(분)설정 창 선택'
상태화면에 기존의 기준 제상 후 지연시간(분)이 표시되어 있음
■ '수정버튼'을 누르면 그림 132의 상태화면이 그림 133의 상태화면으로 바뀜
■ 이 화면에서 'Up(▲)버튼과 Down(▼)버튼'으로 제상 후 지연시간(분)을 수정함

o 00:05(현재설정 제상 후 지연시간) n 00:05(설정할 제상 후 지연시간)
n 00:05에서 새로운 재상 후 지연시간(분)이 설정됨. 00:01로 설정함

■ 새로운 설정값의 입력을 위해 '완료버튼'을 누름
■ 완료버튼을 누르면 '제상 후 지연시간 설정 창'으로 복귀함

그림 132 제상 후 지연시간 설정 창

그림 133 제상 후 지연시간 수정 상태화면

아. 긴급제상

그림 134는 긴급제상 시 상태화면이다. 저장고 내부 온도가 정상상태 범위를 벗어나면 긴급제상을 요구하는 경고 메시지가 나타난다. 이때 수정버튼을 눌러주면 제상이 시작되고 제상이 종료되면 초기화상태로 돌아간다.

긴급제상 상태화면→수정버튼→제상시작→제상종료→제상상태 초기화

그림 134 긴급제상 시 상태화면

자. EVA 팬 모드설정

증발기(evaporator) 팬의 작동을 네 가지 모드로 구분하여 선택할 수 있도록 설계하였다.

운전 / 모드	냉동기 ON	냉동기 OFF	제상
1	ON	OFF	OFF
2	ON	ON	ON
3	ON	OFF	ON
4	ON	ON	OFF

EVA 팬 모드 설정 창 선택→수정버튼→▲▼버튼으로 수정→완료버튼→EVA 팬 모드 설정 창

■ Up(▲)버튼과 Down(▼)버튼을 이용하여 상태화면에서 'EVA 팬 모드 설정 창' 선택'

상태화면에 기존의 EVA 팬 모드 번호가 표시되어 있음

■ '수정버튼'을 누르면 그림 135의 상태화면이 그림 136의 상태화면으로 바뀜

■ 이 화면에서 'Up(▲)버튼과 Down(▼)버튼'으로 모드번호를 수정함

MODE o : 4(현재 설정모드), n : 4(설정할 모드)

n : 4에서 새로운 모드가 설정됨. 1로 설정함

■ 새로운 설정값의 입력을 위해 '완료버튼'을 누름

■ 완료버튼을 누르면 'EVA 팬 모드 창'으로 복귀함

그림 135 EVA 팬 모드 설정 창

그림 136 EVA 팬모드 수정 상태화면

차. 온도 센서의 Mode 설정

온도 습도 센서의 선택을 15가지 모드로 구분하여 선택할 수 있도록 설계하

였다.

센서모드 설정 창 선택→수정버튼→▲▼버튼으로 수정→완료버튼→센서모드 설정 창

■ Up(▲)버튼과 Down(▼)버튼을 이용하여 상태화면에서 ‘센서모드 설정 창’ 선택’

상태화면에 기존의 센서모드 번호가 표시되어 있음

■ ‘수정버튼’을 누르면 그림 137의 상태화면이 그림 138의 상태화면으로 바뀜

■ 이 화면에서 ‘Up(▲)버튼과 Down(▼)버튼’으로 모드번호를 수정함

MODE o : 15(현 설정모드), n : 15(설정할 모드)

n : 15에서 새로운 모드가 설정됨. 08로 설정함

■ 새로운 설정값의 입력을 위해 ‘완료버튼’을 누름

■ 완료버튼을 누르면 ‘센서모드 설정 창’으로 복귀함

그림 137 센서모드 설정 창

그림 138 센서모드 수정 상태화면

MODE	센서의 위치	사용된 센서수	MODE	센서의 위치	사용된 센서수
1	0001	1	9	1001	2
2	0010	1	10	1010	2
3	0011	2	11	1011	3
4	0100	1	12	1100	2
5	0101	2	13	1101	3
6	0110	2	14	1110	3
7	0111	3	15	1111	4
8	1000	1			

카. 보정값 개선

보정값 개선 창 선택→수정버튼→▲▼버튼으로 수정→완료버튼→보정값 개선 창

■ Up(▲)버튼과 Down(▼)버튼을 이용하여 상태화면에서 '보정값 개선 창' 선택'

상태화면에 기존의 보정값이 표시되어 있음

■ '수정버튼'을 누르면 그림 139의 상태화면이 그림 140의 상태화면으로 바뀜

■ 이 화면에서 'Up(▲)버튼과 Down(▼)버튼'으로 보정값을 수정함

o : 04.0℃(현 보정값) n : 04.0℃(개선할 보정값)

n : 04.0℃에서 새로운 보정값이 설정됨. 04.2℃로 설정함

■ 새로운 보정값의 입력을 위해 '완료버튼'을 누름

■ 완료버튼을 누르면 '보정값 개선 창'으로 복귀함

그림 139 보정값 개선 창

그림 140 보정값 수정 상태화면

타. 데드존 개선

냉장시스템의 가동 온도를 지정한다. 기준온도에서 상승하는 범위 즉 허용오

차 범위를 설정한다.

데드존 개선 창 선택→수정버튼→▲▼버튼으로 수정→완료버튼→데드존 개선 창

■ Up(▲)버튼과 Down(▼)버튼을 이용하여 상태화면에서 '데드존 개선 창' 선택'
상태화면에 기존의 값이 표시되어 있음
■ '수정버튼'을 누르면 그림 141의 상태화면이 그림 142의 상태화면으로 바뀜
■ 이 화면에서 'Up(▲)버튼과 Down(▼)버튼'으로 데드존을 수정함

o : 00.2℃(현 설정 데드존) n : 00.2℃(수정할 데드존)

n : 00.2℃에서 새로운 데드존값이 설정됨. 00.3℃로 설정함

■ 새로운 데드존값의 입력을 위해 '완료버튼'을 누름
■ 완료버튼을 누르면 '데드존 개선 창'으로 복귀함

그림 141 데드존 개선 창

그림 142 데드존 수정 상태화면

파. 이상상태 발생

그림 143은 이상상태 발생시 상태화면이다. 냉장시스템이 고장상태임을 의미한다. 전문가에게 수리를 의뢰해야 한다.

그림 143 이상상태 발생시 상태화면