제 2 차년도
최종 보고서

버섯재배 수조식 종합관리 및 재배상 이동장치

태안군 버섯작목회

농 림 부

제 출 문

농림부 장관 귀하

본 보고서를 "버섯재배 수조식종합관리 및 재배상이동장치에 관한 연구"과제의 중간보고서로 제출합니다.

1996. 12.

주관연구기관명 : 태안군 버섯작목회
총괄연구책임자 : 이 택 철
책 임 연 구 원 : 고 동 섭
연 구 원 : 이 상 용
연 구 원 : 김 춘 원
연 구 원 : 김 자 환
연 구 원 : 이 영 주
연 구 원 : 이 승 룡

요 약 문

Ⅰ. 제 목

버섯 재배 수조식 종합 관리 및 재배상 이동 장치

Ⅱ. 연구 개발의 목적 및 중요성

버섯은 아미노산, 섬유질, 회분 등 건강 유지에 필요한 영양분이 대량 함유되어 있으며, 특히 다른 작물에 비하여 항암 물질과 노화 방지 물질이 월등하게 함유되어 있어서 최근에 그 수요가 폭발적으로 증가하고 있다. 여러 종류의 버섯 중에서도 느타리 버섯은 한국인의 기호에 가장 알맞고, 각종 음식물의 첨가제로 많이 사용하고 있다. 느타리버섯은 4 계절 재배가 가능한 농산물이지만, 품종에 따라서 적절한 재배 환경을 만들어 주어야 한다.

그러나 재래식 방법에 의해서는 생산량이 저조하여 버섯 수요에 미치지 못하고 있을 뿐만 아니라 온·습도를 유지 관리하는데 많은 비용이 들어 재배 단가가 높다. 따라서 본 연구에서는 연료 비용과 인건비를 절감하고, 병충해 방지에 있어서 그 효과가 기대되는 새로운 버섯 재배 방법의 연구에 그 목적을 두고 있다. 이 연구 결과가 농가에 보급되므로써 버섯의 다수확을 통한 농가 소득의 증대와 함께 물량의 원활한 공급으로 소비자 가격을 적정 수준으로 유지하고 국민 보건에도 크게 기여할 것으로 기대한다.

Ⅲ. 연구 개발 내용 및 범위

1. 재배사의 밀폐식 보온 시스템 채택
2. 재배상의 배치 : 일광 소독이 가능한 이동식

3. 컴퓨터 자동화 시스템

1) 원격 중앙 집중식 제어 시스템

2) 온·습도 유지 방법 : 지하수 수조식 또는 퇴수로식

3) 온·습도의 조정 : 온·습도 감지와 제어 장치의 자동화

4) 탄산 가스 : 탄산 가스 검지와 환기 시설의 자동화

Ⅳ. 연구 개발 결과 및 활용에 대한 건의

1. 재배상의 효과

1) 주요 결과

구 분	제작비 (%)	살균 연료 소모량 (%)	인력(입·폐상시) (%)	비닐 사용량 (%)
고정식	100	100	100	100
이동식	300	40 ~ 60	30	120

2) 살균과 발효시에 공간 압축 효과에 의해서 균일한 효력을 얻을 수 있으며

3) 병충해 전염을 방지할 수 있음.

4) 일정한 살균과 발효에 효과가 좋음

2. 95년도 동절기 재배 시험 결과

1) 방법: 지하수 수조식, 난방 시설 비사용

2) 성과: 실내 온도 적정 유지 (7 - 15℃)

실내 습도 적정 유지 (95 - 98%)

실내 습도 95% 유지로 무관수 재배 가능

동절기 재배 가성능 확인

3. 96년도 하절기 시험 재배 결과

1) 방법: 지하수 수조식, 냉방 시설없이 재배

2) 종균: 원형(저온성 종균)

3) 결과: 실내 온도 다소 높음 (24℃)

배지 온도 (28℃ 이상)

실내 습도 (90 - 98%)

발이는 되나 배지 온도가 높아 생장 불가능

품종 선택이 잘못된 것으로 판단

4. 96년도 동절기 시험 재배 결과

1) 방법: 지하수 동굴형 퇴수로식

2) 성과: 퇴수로 배출 온도 적정 (11℃)

실내 온도 적정 유지 (7-15℃)

실내 습도 적정 유지 (95 - 99%)

지하수 수조식에 비하여 동굴형 퇴수로식이 경제성 있음

연료 소비 無

5. 컴퓨터 제어 시스템의 특징

제어용량	데이터 수집 및 제어 장치	제 어 방 법		
		온 도	습 도	CO_2
2동 이상	컴퓨터 인터페이싱 A/D converter	보 일 러 순환펌프	지하수 순환에 의해 적정 수준 유지 (제어 불필요)	송풍 모터

목 차

제 1 장 서 론

버섯은 건강 유지에 필요한 영양분이 대량 함유되어 있으며, 특히 다른 작물에 비하여 항암 물질과 노화 방지 물질이 다량으로 함유되어 있어서 최근에 그 수요가 폭발적으로 증가하고 있다. 버섯은 4 계절 재배가 가능한 농산물이지만, 품종에 따라서 적절한 재배 환경을 만들어 주어야 한다.

그러나 재래식 방법에 의해서는 온·습도를 유지 관리하는데 많은 비용이 들 뿐만 아니라, 24 시간 관리를 위한 노동력의 소모가 많아서 재배 단가가 높다. 또한 인력에 의한 재배 환경의 관리하에서는 생산량의 증대와 대량 생산을 크게 기대할 수 없다.

한국인의 기호에 가장 알맞은 느타리버섯은 가장 보편적으로 재배하고 있는 버섯이다. 현재 사용하고 있는 재배 방법과 시설들은 매우 다양하다. 어느 것이 가장 효과적인지 구체적으로 연구된 바는 없으나 기본적인 재배 조건은 같으며 생산량들도 대동소이하다. 그러나 급증하는 수요량을 충족시키고, 생산 단가를 낮추며, 소비자 가격를 안정시키기 위해서는 기존의 방법과는 그 개념이 다른 재배 시설을 연구해야 할 단계에 있다.

우선 연료를 절약할 수 있는 방향에서 연구가 수행되어야 할 것이다. 느타리버섯 재배에 소모되는 경유의 量이 일반 재래식 재배사에서 동절기(10月~4月)에 2,000~3,000 ℓ 정도 된다. 연료 사용량을 절약하면 농가 소득의 증대와 함께 연료의 수입 대체 효과를 얻어 내고 공해 방지에도 일조를 할 것으로 기대한다. 유료 전량을 수입에 의존하는 현 실정에서 지열이나 태양열을 이용하는 방법도 강구해야 할 것이다.

원격 자동화를 통해 효율적으로 재배 관리를 하는 것도 매우 중요한 연구 과제

일 것이다. 컴퓨터를 이용하는 관리 체제에서 기대할 수 있는 효과는 대단히 많다. 가장 중요한 것으로는 재배 조건들을 자료화할 수 있다는 것이다. 버섯 재배의 전 단계에서 주어졌던 온도, 습도, 기타 가스들에 대한 수치를 저장하므로써, 재배 후에 그 결과를 진단할 수있는 근거를 마련할 수 있다. 이것은 버섯 재배의 과학화를 위한 중요한 첫단계로 볼 수 있다.

또한 재배상을 효율적이고 과학적으로 운영할 수 있는 시설로 전환해야 한다. 지금까지 사용하고 있는 고정식 재배상에서는 작업시 불편한 점이 많다. 고령화로 노동력이 부족해지는 농촌에서 작업의 능률을 극대화하기 위한 재배상의 배치에 대해서도 연구해야 할 것이다.

앞에서 기술한 문제점들을 보완 또는 해결하기 위해서 다음과 같은 개발 목표를 수립하였다.

첫째, 인위적인 냉·난방 시설을 사용하지 않고 지하수를 이용하여 온·습도를 적정 수준으로 유지한다.

둘째, 일광 소독을 할 수 있는 이동식 재배상을 채택하고 그 효과를 검증한다.

세째, 컴퓨터를 이용하여 온도, 습도, 탄산가스를 원격 감지하고, 제어할 수 있도록 구상하였다. 또한 이 데이터들은 컴퓨터 모니터를 통해 실시간(real-time)으로 관측할 수 있으며, disk에 저장하도록 설계, 제작한다.

제 2 장 연구 배경

제 1 절 버섯 재배의 일반 현황과 연료 소비성에 대한 견해

버섯 재배에서는 현재 동절기에 집중적으로 재배를 하고 있고 하절기 재배를 기피하고 있는 실정이며, 동절기 재배시에 연료의 다량 소비에 대한 우려가 많다. 느타리는 동절기에 소비가 되는 계절 식품이며, 하절기에는 재배, 보관, 수송 등의 문제와 품종의 선택 문제로 재배를 기피하고 있는 실정이다.

95년도 현재 8,126 재배 농가가 1,789,222평의 재배 면적을 가지고 있으며(영농교재 발췌), 이를 棟收로 환산한다면 재배상 면적을 평균 50坪으로 가정하면 약 35,700棟, 70坪이면 25,000棟이지만 대부분 겨울철 재배에 임하고 있다.

지역적으로 차이는 있겠으나 재배 기간에 棟當 2,000~3,000 ℓ의 연료를 사용하고 있다. 10℃ 내외의 보온을 유지하면서 산소 공급을 위한 환기가 적정 수준 이상으로 이루어져야만 하는 이율배반적인 관리를 하여야 하기 때문에 많은 연료가 소비된다.

표 1. 재배 방법에 따른 연료 사용량과 인건비의 상대 비교표.

구 분		연 료 사 용 량	인 건 비
일반 재배사		100%	100%
본과제	栽 培 舍	20 ~ 30%	100%
	栽 培 床	40 ~ 60% (空間 壓縮 殺菌 方法)	30% (入廢床時)

현재는 재배 경험에 의해 적당한 환기구 개방과, 타이머에 의한 송풍 모터로 불어넣기식 투입 방식에 의한 환기 시설로 재배를 하고 있다. 이 방법도 외부 기온과 버섯의 생장 과정별로 적당한 기준 설정이 별도로 필요한 사항이다.

이와 같은 문제를 해결하는 방법으로써 밀폐식 재배와 수조식 재배 환경 종합 관리 방법을 구상하였다. 즉 폐광에서의 버섯 재배를 모방 참조하여 연료를 70~80 % 이상 절감하고 재배상 이동식 시설과 컴퓨터의 자동화 시설로 인력 절감, 소모성 機資材의 마모 방지, 품질 향상, 그리고 생산 증대에 기여하게 한다.

제 2 절 재배사의 문제점과 보온의 주안점

1. 재배사

현재 국내의 재배사는 하우스용 파이프에 의한 재배사로 견딤성과 화재에 의한 피해 등의 문제가 있다. 부록크 造의 栽培舍를 건축하여 일반 재배사에서의 미완 부분을 보완하여야 하겠다. 본 과제는 이를 밀폐식 재배로 전환할 수 있는 계기가 되도록 하여야 한다고 구상하였다.

2. 밀폐식 재배사의 설계도

현행 창문을 통하거나 外氣를 닥터를 이용하여 환기를 시키는 방법을 개방식으로 보고 있으며 본 과제에서는 개방식보다는 밀폐식으로 선택했다.

개방사의 경우 연료의 절약이 불가능하다. 外氣 그대로 주입되기 때문에 재배에 필요한 적정 온도까지 가온을 해야만 한다. 또한 精製되지 못한 공기의 유입으로 병과 충해에 노출된다. 환기창 등으로 설치된 개방식 재배사에서는 밀폐식 재배가 적용되지 못하기 때문에 병충해로부터 격리되지 못한다.

반면에 밀폐식 재배사의 경우에는 연료의 절약이 가능하다. 외기의 온도를 지하수 15℃에 근접하는 온도로 (수조와 퇴수로를 통과시킴으로서) 재배사에 강제 투입

재배 관리하는 방법이기 때문이다. 특히 精製된 공기만을 주입시킬 수 있으며 外氣에 노출되어 있지 않음으로 해서 병충해로부터 격리된다.

외기의 공기가 직접 창문이나 환기구를 통해서 재배사로 투입되지 못하도록 밀폐한 상태의 재배사를 밀폐식 재배사라 한다.. 일반 재래식 재배사는 연료의 절약형이지 못하여 단열 시설이 효과가 적다. 가온과 환기가 동시에 이루어져야 한다는 二律背反的 상황에 의해 2,000ℓ이상의 연료 소비가 이루어진다. 밀폐식 재배사를 개발하고 본 과제의 洞窟形 퇴수로(즉 지열)를 이용하여 버섯을 재배하고자 한다.

3. 재배상

일반 栽培床은 고정되어 있기 때문에 立床과 廢床에 어려움이 있고 살균시에 연료의 소비가 800ℓ이상으로 과다함은 물론 전체적으로 배지 조성을 함에 있어 일률적인 조건으로 만들지 못한다. 본 과제에서는 연료의 40% 이상을 절감하면서 공간 압축 살균의 효과로 일률적인 배지 조성이 이루어질 수 있는 조건을 성립시켰다.

4. 관리의 자동화

현행 재배 관리는 인위적으로 조작 관리하여야 하기 때문에 버섯 생산 과정과 外氣의 변화에 적절히 대처하지 못하여 기계의 마모를 비롯한 연료, 전력의 절감에 부적합하다. 본 과제는 1 대의 컴퓨터로 2棟을 일시에 재배 관리가 사용 가능하도록 구상하였다.

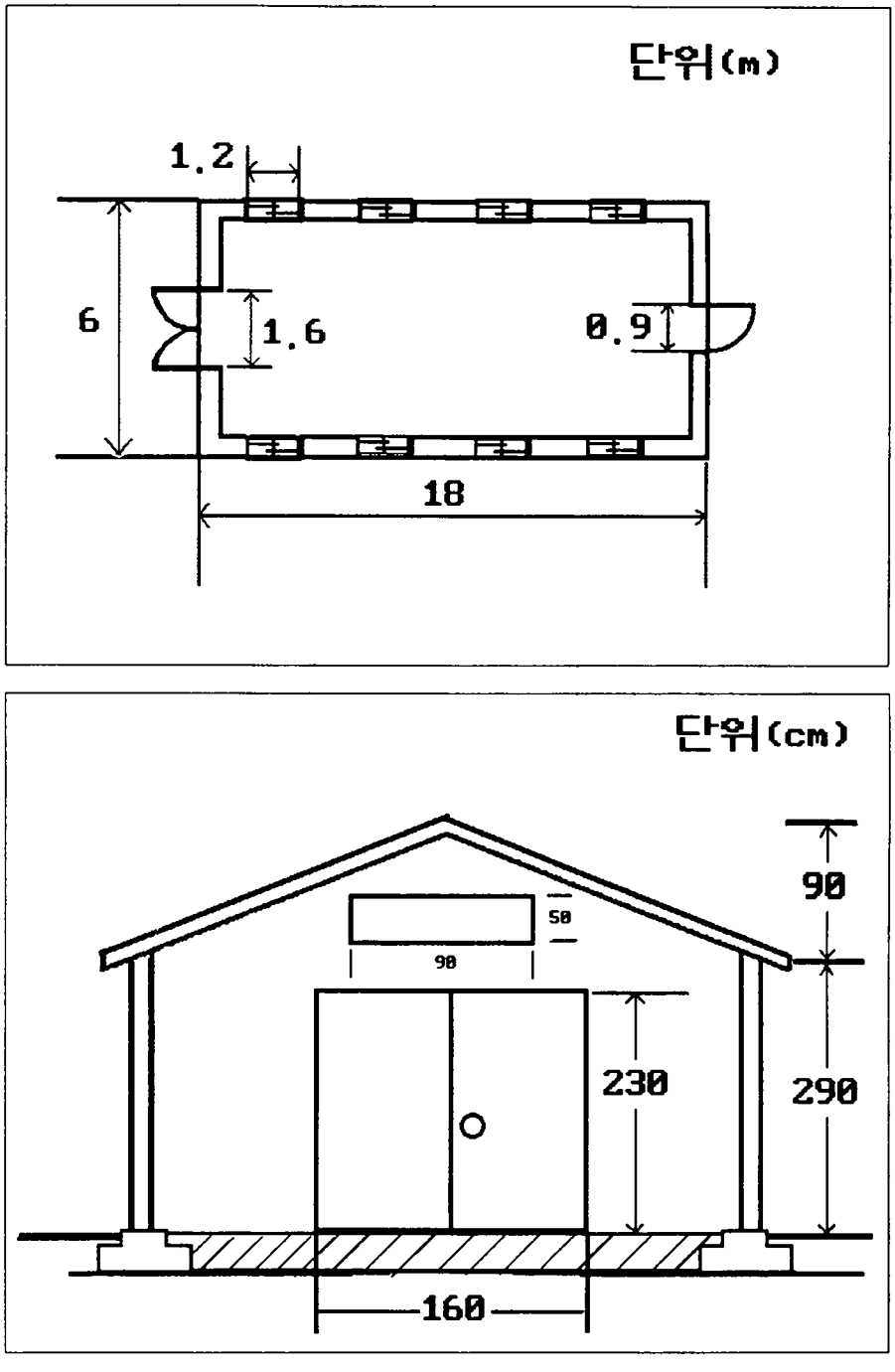


그림 1. 밀폐식 제배사의 평면도 및 정면도.

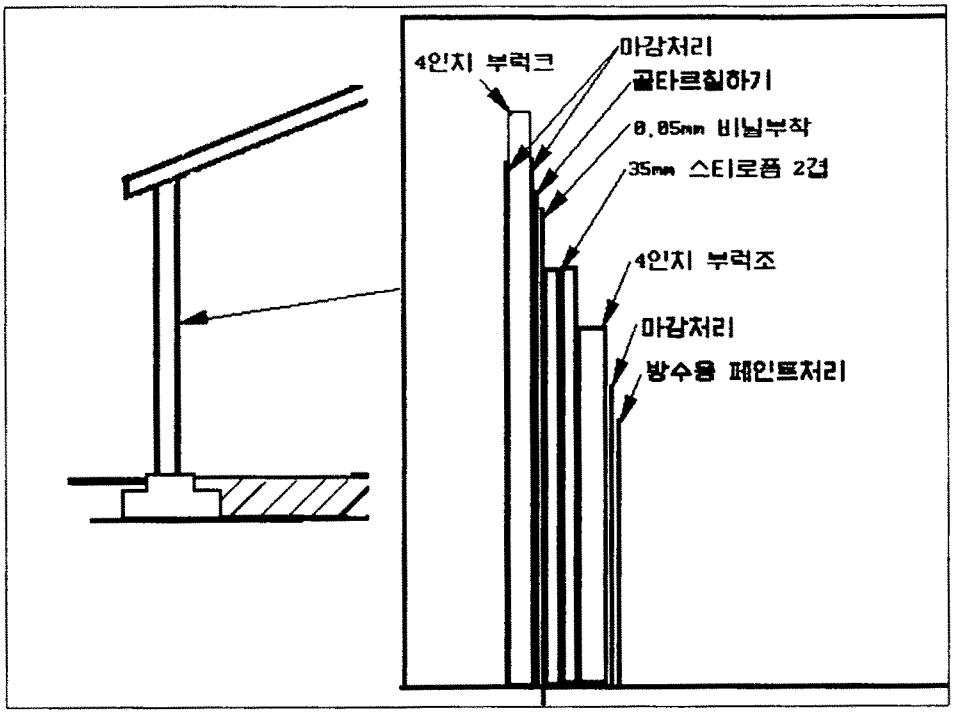


그림 2. 벽체 단면도.

제 3 장 연구 방법 및 결과

제 1 절 수조식 구조 및 원리

1. 수조식 구조와 작동 원리

위탁 연구팀에서 1/10 축소 건물모형 내에서의 기포 발생에 의한 온· 습도 측정 실험을 실시하였다. 그 결과 10 m의 동관을(ϕ5 ㎜)이용하여 기포를 발생시켜 실험했을 때 3℃의 온도를 減溫시킬 수 있었고 90% 이상의 습도를 얻어낼 수 있음을 확인하였다. 수조를 노출형에서 지하 매립형으로 변형시키고 주름관에서 동관으로 교체, 콤푸레샤에서 링부로아 모터로, 0.5 HP, 1 HP에서 2 HP로 증가시키면서 수조식의 서능을 향상시켰다.

현재 단열과 공기 양을 대폭 늘리고 퇴수로에서 공기를 뽑아 쓰는 등의 조치로 70% 이상의 습도와 18℃ 정도의 온도를 얻어내고 있어 희망적이라 할 수 있다. 다음은 수조 설계이다. ϕ 700㎜ 관을 지하에 묻어 외기의 영향에서 벗어나고 이용에 편리하도록 설치를 했으나 물이 새는 재료여서 직경 600 ㎜ 철심관으로 교체 매설하였다. 본래 폐광에서 흘러나오는 동굴 바람을 만들어 버섯 재배에 사용하겠다는 본 과제의 계획에 의해 직경 200 ㎜ 주름관을 묻고 그 관으로 지하수를 흘려 보냈다. 이과정에서 생기는 습도와 온도를 품은 공기를 퇴수구 옆의 흡입구를 통해 링브로아가 흡입하여 흡입된 공기를 기포 발생 장치로 보내어 적당한 온·습도를 가진 공기를 만들어 재배사로 투입하여 이 공기만으로 버섯 재배를 실시하였다.

링부로아에서 보내는 공기가 네가닥의 12m 관을 지나면서 물의 온도를 냉각시키는 구성을 하였다. 그 온도가 유속과 관계가 된다. 공기의 유속과 온· 습도와는 반비례한다고 보아야겠다. 따라서, 종래의 관의 길이보다 곱으로 늘려 주고 공기의 조절 장치는 버섯이 생산될 때 조건을 보아 가며 보완할 계획이다.

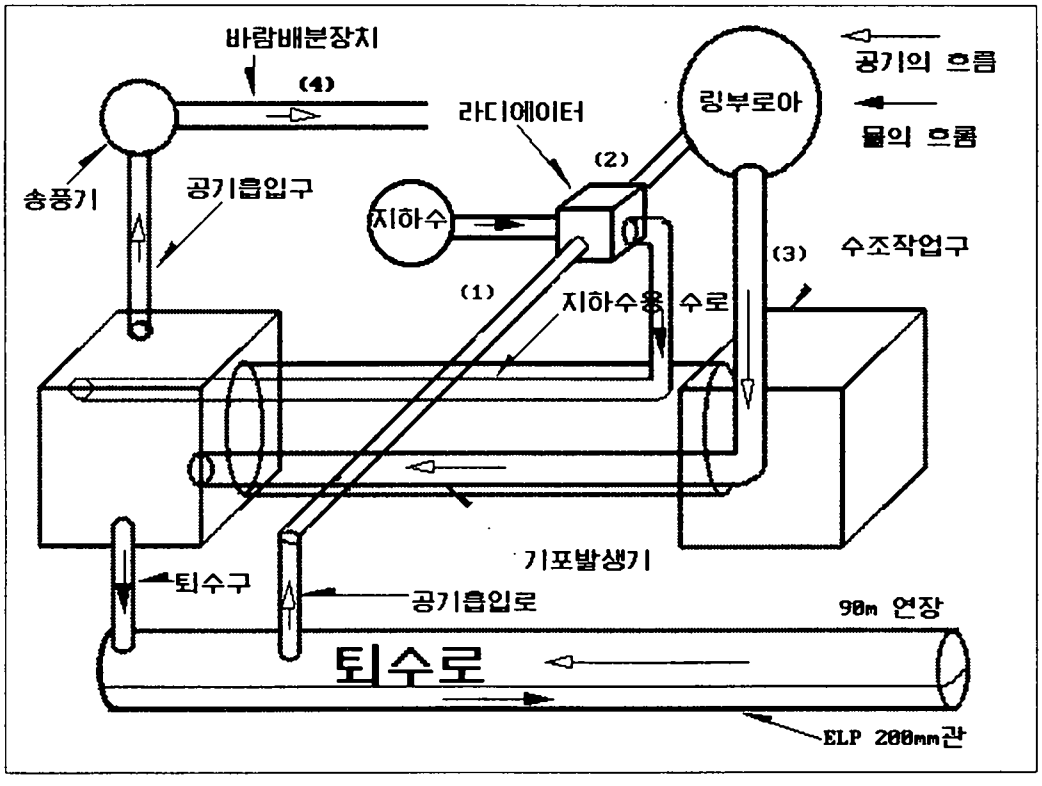


그림 3. 수조 구성을 설명하는 개략도.

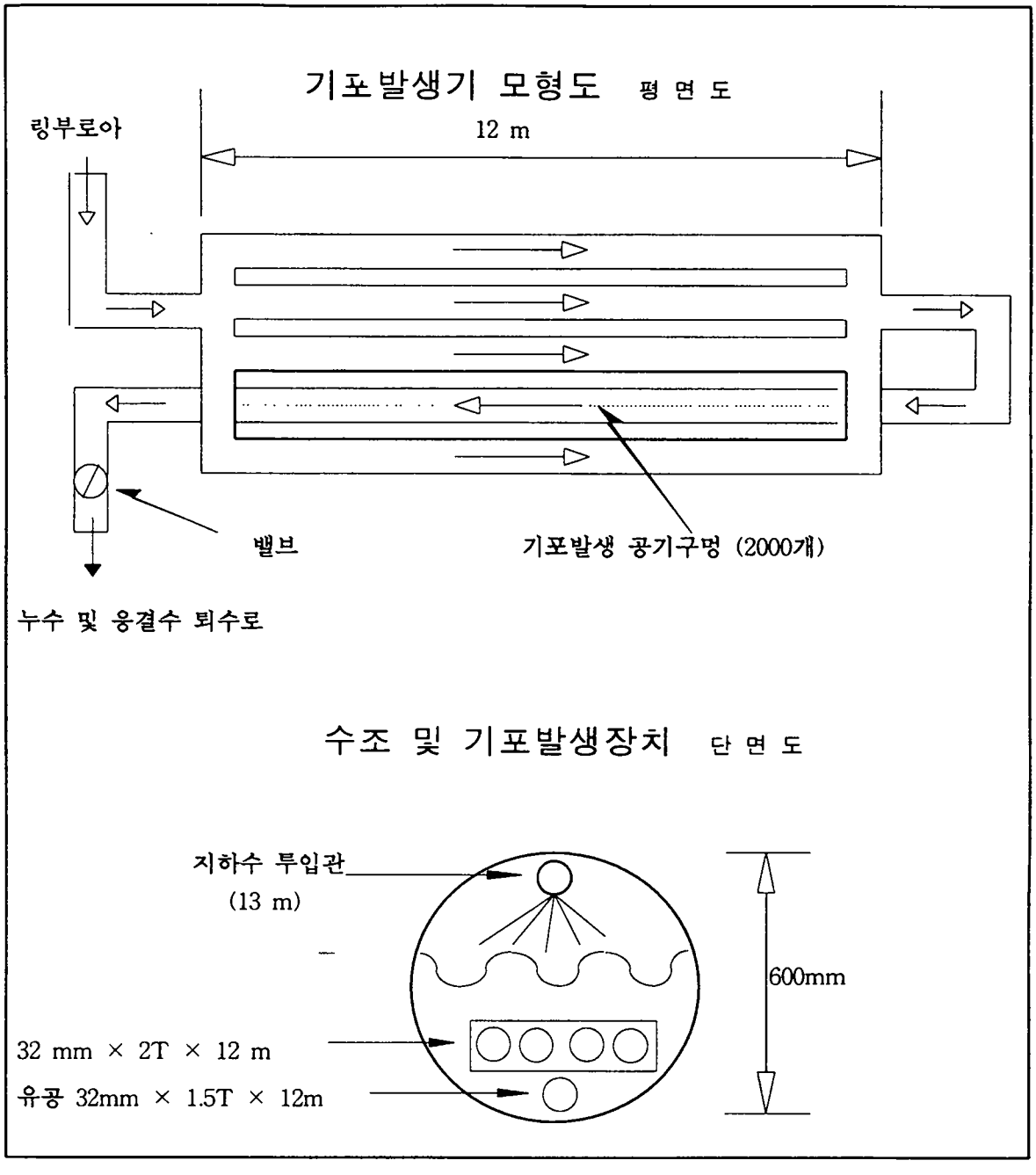


그림 4. 수조 및 기포 발생 장치의 개략도.

퇴수로 200 ㎜관의 역할은 다음과 같다. 현재 1 m 깊이의 지하에 매설 하였다. 관의 굵기를 더 늘려야 좋겠고 또한 관의 경사각도 커서 물의 흐름이 빨라 물방울로 굴러 떨어지는 정도의 경사각이 되었으면 더욱 효과적이겠다는 생각이 든다. 깊이도 또한 깊게 묻히는 것이 효과적일 것이다. 길이도 또한 길게 묻히는 것이 효과면에서 좋다.

문제점으로는 계속해서 외기 기온을 빨아들일 때 퇴수로가 얼거나 더워져 무용지물이 되지 않을까 하는 우려도 있다. 그러나 이 문제는 지하수의 온도로 극복할 수 있을 것으로 보고 퇴수로 상단에 지하수를 흘러내리게 함으로써 문제를 보완하였다.

표 2. 수조식 온도 조절 장치의 각 위치에서의 온도.

(1995. 11. 16. 오전 6시 현재)

구 분	온도 (℃)	그림 ( ) 참조
외부 온도	-2	
퇴수로 통과된 흡입온도	11	1번
라지에이터의 통과온도	13	2번
링부로아 통과온도	25	3번
실내 투입 온도	16	4번

현 온·습도 변화는 물론 공기 방울의 흡착, 산소 공급과 먼지나 잡균 등 이물질 제거하는 효과가 있으며, 하절기 재배시 퇴수되는 물이 논으로 들어가는데 냉수가 아닌 상태이기에 논농사에도 도움이 될 것이다.

2. 주요부 역활및 특징

1) 수조

- 직경 600 mm 길이 12 m의 철심관

- 역할 : 45 cm 높이로 물을 담는 물통, 기포발생기를 설치하여 공기의 온도와 습도를 地下水 溫度에 가깝게 조절 · 이용하는 장치이다.

2) 용수로

15℃의 지하수를 수조에 보내는 30 mm PVC파이프로 라디에이터를 거쳐 들어오는 물 배관임

3) 퇴수로

계속 들어오는 지하수를 퇴수시키는 시설로써 직경 200 mm의 주름관이며 150 m를 설치하였다. 이 퇴수로를 통과해서 흘러 내려가는 물은 작은 파도를 치면서 공기를 변온 시킨다. 변온될 때 생기는 습기를 머금은 공기가 라지에이터를 거쳐 링부로아의 압력으로 기포 발생기를 통해 재배사에 투입된다.

4) 퇴수로의 물리적인 역할

14℃의 지하수가 퇴수되면서 잔물결을 치면서 내려가고 반대로 흡입되는 공기 바람과 마찰에 의해 습도와 온도가 생성한다.

5) 퇴수로의 효과

폐광의 효과를 기대했던 동굴형 퇴수로는 아직까지 효과를 평가하기는 이르나 표 2에 보는 바와 같이 대단한 효과가 있으며 이 효과만으로도 충분히 버섯 재배에 도움이 된다고 본다. 그 크기와 길이가 길게 매설될 경우 그 동굴 바람이 나온다고 보겠다. 하절기에 담배 연기가 위에서 아래로 흘러 들어가는 것을 관찰할 수가 있었고 겨울에는 아래에서 위로 대류하여 폐광에서와 같은 작용이 발생하고 있음을 확인하였다.

6) 라지에이터

보통 쓰는 방법으로 지하수를 이용한 에어컨을 만드는 방법을 이용하는 것이

다.

7) 링부로아 모터

물 속에 공기를 불어넣어 기포를 발생시키는데 이용되는 송풍 모터이다. 목욕탕에서 쓰고 있는 기포 발생 장치와 원리상 유사하다. 분당 3 m'의 용량의 2 HP 모터를 사용하였다.

8) 기포 발생기

링부로아 모터에서 강제 주입된 공기를 지하수의 온도를 이용하기 위해 직경 32 mm 동관을 수조에 4줄로 분산시켜 통과시킨 후 이를 다시 모아서 직경 32 mm 외가닥 동관 상단에 5 mm 간격으로 2,000 개의 구멍을 뚫어 이 구멍으로 공기를 배출시키는 장치이다.

9) 공기 방울 형성기의 역할

1 mm 크기의 2,000 개 구멍으로 공기가 나와 공기 방울이 형성되어 약 45 cm의 물깊이를 통과할 때 공기 방울들은 15℃의 지하수와 만나는데 실험 결과를 통해서 공기의 변온에 있어서 지하수가 주동적인 작용을 하고 있음을 알 수 있다.

10) 송풍기

- 링부로아에서 송풍된 공기가 공기 방울 형성기를 통과할 때 저항에 의해 공기량이 감소되어 효과적으로 분산되지 못하기 때문에 수조 안에서 공기를 흡입하여 이를 재배사에 분산 공급하는 모터이다.
- 용량 : 1/4마력, 1/2마력
- 공기 송풍량 : 자연 상태에서 분당 6 m'

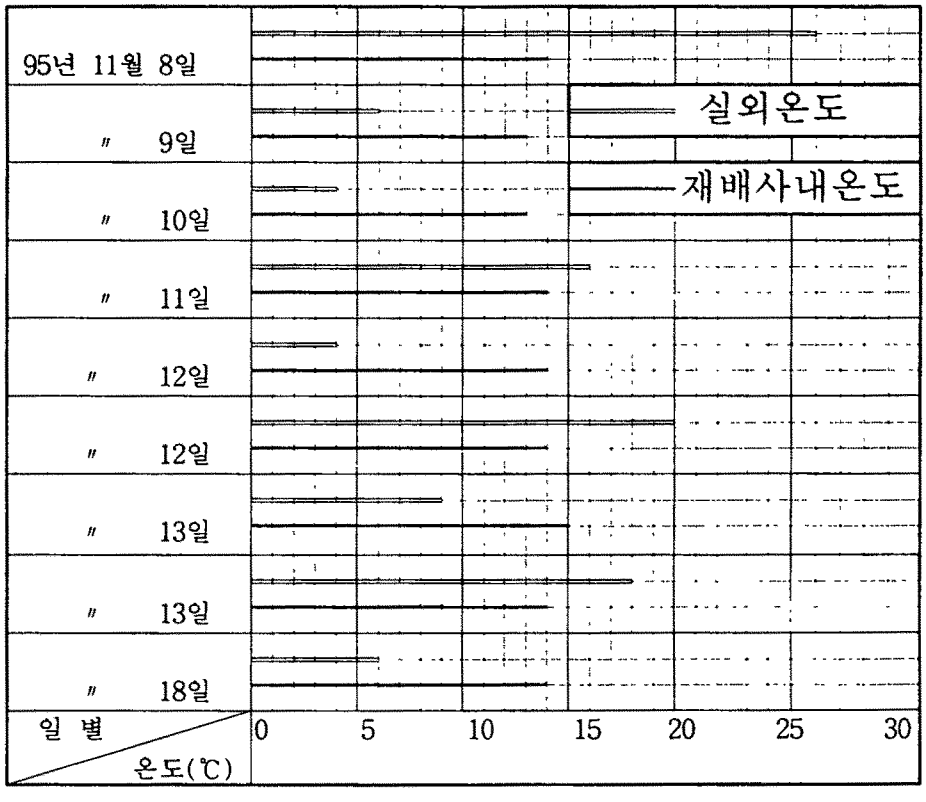
11) 바람 배분 장치

-송풍기로부터 오는 바람을 전재배사내에 고르게 배분시켜 주는 P.V.C 관이다.

-크기 : 직경 100 mm, 길이 : 16 m × 2

2. 수조식 온 · 습도 장치의 조절 능력 평가

표 3. 동절기 실외 온도와 재배사내 온도의 비교표.



수조식과 보일러를 함께 사용하여 겨울 재배를 시도하려 했으나 수조식만으로도 충분한 가능성이 입증이 되어 가온없이 재배했다. 가온없이도 실내 온도는 실외 온도에 영향을 거의 받지 않는다. 표 3에서 보는 바와 같이 동절기에 실외 온도가 4-26℃ 까지 변화하였으나, 실내 온도는 13-15℃의 작은 범위에서 유지되고 있으며, 이 온도는 버섯 재배시 적정 온도에 해당한다.

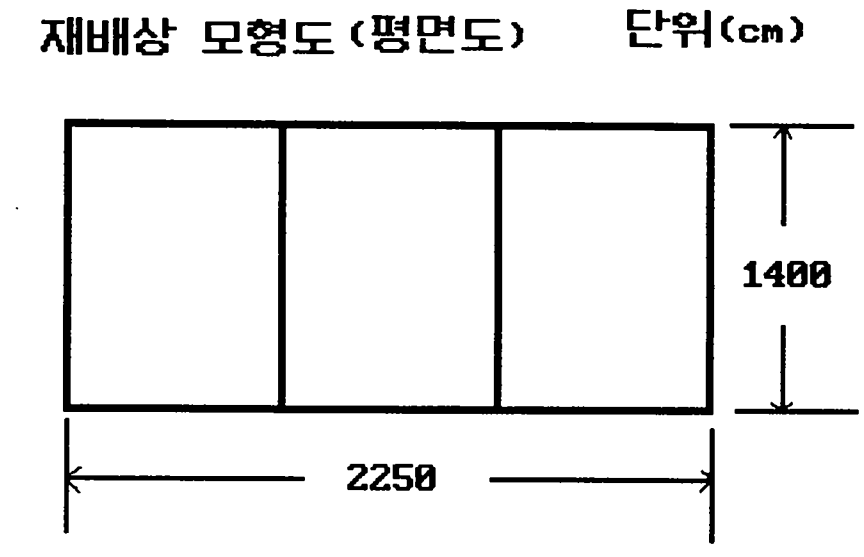
표 4. 동절기 실외 온도와 실내 온도의 비교.

구 분		온 도 (℃)		습 도 (%)	
		밤	낮	밤	낮
실외온도	최 고	6	27	99	60
	최 저	1	20	85	30
실내온도	최 고	14.5	15	95 - 98%	
	최 저	13.5	14		

표 5. 하절기 실외 온도와 실내 온도의 비교.

구 분	온 도 (℃)	습 도 (%)	퇴수로 유입 온도 (℃)
실외 온도	34	50~80	
실내 온도	24	90~98	19

표 4와 표 5는 각각 동절기와 하절기에 측정한 결과를 보여 주고 있다. 동절기에 외부 온도와 습도의 많은 변화에도 불구하고 실내 온도는 평균 13-15℃, 습도는 95-98%의 적정 수준을 유지하고 있다. 퇴수로 유입 온도가 19℃임에도 불구하고, 하절기에는 외부 온도가 높아서 실내 온도는 다소 높은 24℃를 나타내고 있다. 약간의 냉방 시설 가동만으로 버섯 재배 온도까지 낮출 수 있었다. 습도는 역시 90-98%로 적정 습도를 유지하고 있다.



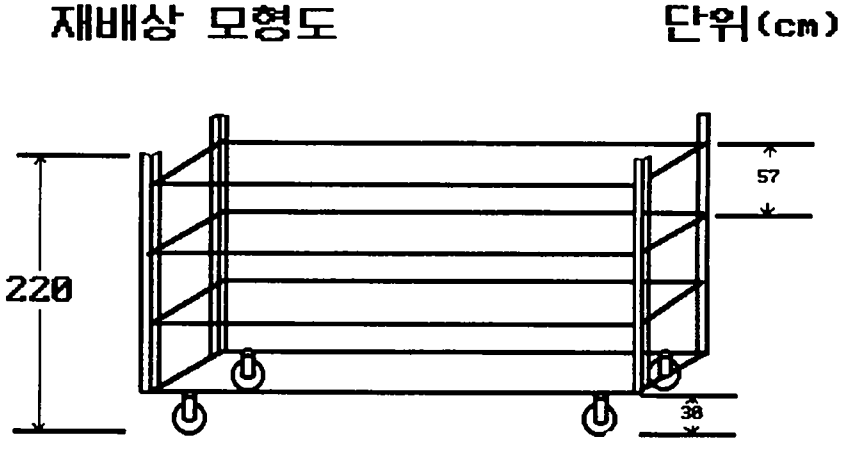


그림 5. 이동식 재배상의 구조.

제 2 절 재배상 이동 장치

1. 이동식 재배상 제작

이동 시설에 대한 결과는 매우 만족스럽게 생각한다. 능률면에서 이상적이나 본 재배사에 들어 있는 배지의 양이 상자 재배용 용기로는 약 1,000 개 이다. 기존 시설의 재배상의 양과 같다. 4 단 구조의 14 개가 제작되었으며 이 제작을 위해 여러 가지 모형을 설계하였다. 여러 곳의 제작사와 상의한 결과 예산의 풍념 산업에서 제작을 맡아 주어 견고하고 고급스럽게 제작이 되었다. 바퀴는 약간 작다는 생각이 들긴 하지만 이상은 없다. 표 6은 고정식에 비해서 이동식이 효율적임을 보여 주고 있다. 고정식에 비하여 제작비와 비닐 사용량은 많으나, 소모성 경비인 연료비와 노동력에 있어서는 상당한 절감 효과를 기대할 수 있었다.

표 6. 고정식과 이동식 제배상의 특성.

구 분	제 작 비(%)	살균 연료 소모량(%)	인력(입 · 폐상시) (%)	비닐 사용량 (%)
고 정 식	100	100	100	100
이 동 식	300	40~60	30	120

제 3 절 자동화 시스템의 역할과 방법

컴퓨터에 의하여 종합적으로 관리함으로서 효과적이고 능률적으로 버섯 재배가 가능하겠다. 자세한 내용은 부록에 수록하였다.

1. 연료의 절대적인 절약

① CO_2 센서에 의한 환기로 열 손실이 없기 때문에 낭비성 가온이 필요없다.

② 마모성 기계 작동 절제로 연료 및 전력 낭비 방지 효과를 기대할 수 있었다.

2. 버섯 품질의 향상과 다수확

① 원하는 품질의 생산이 가능해 진다.⇒버섯의 크기와 모양을 센서에 의한 환기량 조절이 가능해진다.

② 수확과 관리를 위한 출입 이외에 출입이 불필요하며 인력 절감과, 변함이 없는 환경 조건으로 병충해 감염률이 감소하였다.

제 4 장 시험 재배와 실제

표 7은 수조식을 이용한 경우 실내 온도를 버섯 재배 적정 온도로 유지시킬 수 있음을 보여주고 있다. 표 8은 퇴수로를 이용하였을 때의 결과이다. 수조식에 비하여 퇴조식을 이용하였을 때 더 낮은 실내 온도를 얻을 수 있어서 보다 효과적임을 입증하였다. 표 9는 난방 시설을 이용하여 가온하였을 때 습도가 감소함을 나타내고 있다.

표 7. 95년 동절기 평균 실내 온도.

외부 온도(℃)	흡입 온도(℃)	실내 온도(℃)
1	12	9

표 8. 96년도 동절기 평균 실내 온도 및 습도.

외부 온도(℃)	투입되는 온도(℃)	실내 온도(℃)	습 도 (%)
-10	11	7	100~95

표 9. 자연 상태와 난방 장치를 사용하였을 때의 평균 실내 습도의 비교.

외부 습도(%)	실내 습도(%)	가온시 실내 습도(%)
30 이하	~100	85~95

연료의 절감과 재배床이동 장치에 의한 인력의 절감으로 농가의 소득 증대는 물론 재배사와 수조의 종합 관리 시설의 보급형 설계도를 작성하였다. 95년도 시험 재배 결과를 통해서 본 과제의 성능을 보면 습도는 95%로 가습이 필요없다. 이는 공기 방울이 물 속을 통과할 때 자연스럽게 습기를 머금은 공기가 되어 舍內로 투

입되고 이때 탁한 공기는 물 속의 작은 공기 방울 형태에서 방울 벽면에 흡착되는 精製 현상에 의해 산소를 간직한 맑은 공기가 舍內로 투입이 된다고 본다.

外氣 溫度가 -10~4℃일 때 동굴형 퇴수로를 거쳐 라지에이터에 흡입될 때 영상 11±1℃에 이르고 라지에이터에서 1℃ 상승하며 링부로아를 거쳐 실내 투입 온도가 15.5±1℃로써 가온없이 재배가 이루어진다.

그러나 外氣가 계속 저온 상태일 때 재배사의 단열 효과가 떨어져 실내 온도가 5-7℃로 떨어졌으며 이때 버섯의 생장 속도는 더디고 버섯 대가 짧아 상품성이 떨어졌다.

문제점으로는 링부로아 모터의 전기 소모가(2HP) 많아 문제되기도 하지만 모터의 소음도 높아 대단히 부산하고, 가격이 32만원 고가로 그 수명도 짧아 본 과제에 부합되는 전용 모터가 아니면 실용성이 떨어져 일반 버섯 재배 농가에서 사용하기가 어렵다고 생각된다.

모터의 성능이 3 ㎥/분의 흡입 능력으로 재배의 환기 요구량에 못미처 퇴수로에서 40 ㎜파이프를 통해 직접 흡입 혼용 투입하였고, 모자라는 공기량을 보충하기 위하여 수조식을 보완, 보충시키는 계획이였으나 퇴수로를 60 m연장 매설하고 매설부 上端에 小形관정(20 L/분)을 뚫어 약 14℃의 지하수를 흘려 내리게 하여 퇴수로의 역할을 즉, 冬夏節期에 흡입구로부터 얼거나 더워짐을 방지하여 주는 역할을 하도록 하였다. 퇴수로의 직경과 길이의 관계는 지열 이용에 있어 함수 관계가 있다고 생각한다. 습도는 外氣의 온도가 실내의 온도보다 온도차가 많을수록 높아짐을 본다. 96년 하절기 동절기에 버섯 시험 재배에서는 라지에터, 링부로아와 수조를 생략하고 퇴수로에서 송풍모타가 직접 공기을 흡입하여 배분 장치를 이용하여 재배사에 투입하는 방법으로 이용하였으며, 모터의 용량은 12 ㎥/분의 ½ 마력으로 교체 이용하였다. 링부로아 이용시의 실내 온도와 같은 결과가 되었다. 15℃의 온도가 풍량이 적어 재배사에 미치는 영향은(재배사 표면의 단열 효과 관계로) 별 차이가 없었고 습도는 100%로 높게 나타난다.

제 5 장 결 론

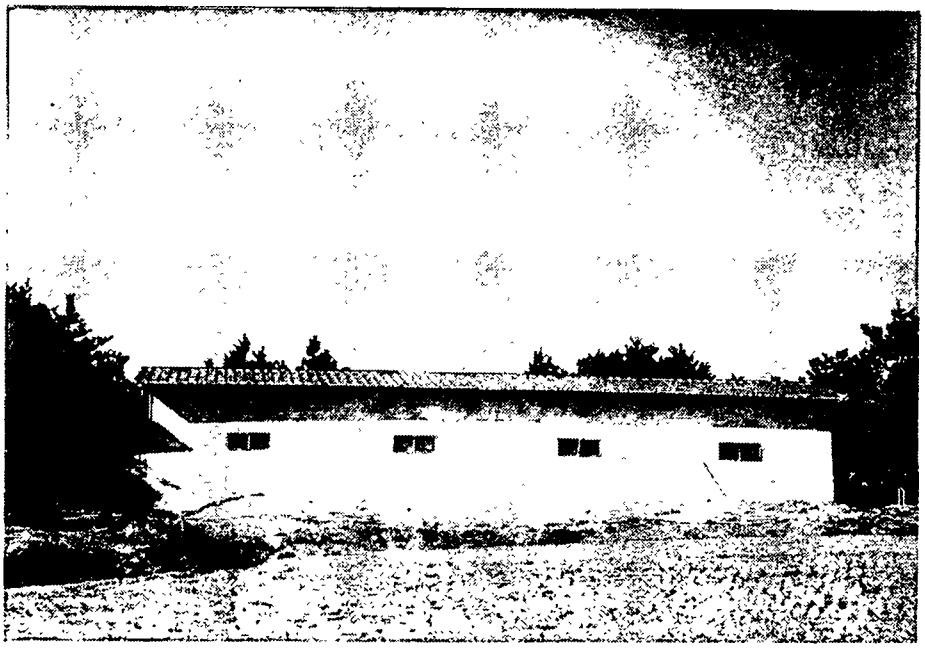
수입에 의존하고 있는 유류의 대량 소비가 국가 경제 발전에 부담을 주고 있으며, 버섯 재배에서 소비하는 연료비도 상당한 액수로 묵과할 수 없는 수준에 있다. 이로인한 공기 오염 뿐만이 아니라 연료 구입비에 따른 농가의 부담이 커지고 있으며, 버섯의 생산 단가도 상승하고 있다.

본 과제에서는 지하수를 이용하여 재배사의 온도와 습도를 유지할 수 있는 방법을 연구하였다. 수조식 종합 관리 체제를 도입하여 난방 시설의 가동률을 낮추어 재배에 성공하였으며, 그 결과로 연료비를 약 70-80% 정도 절약할 수 있었다.

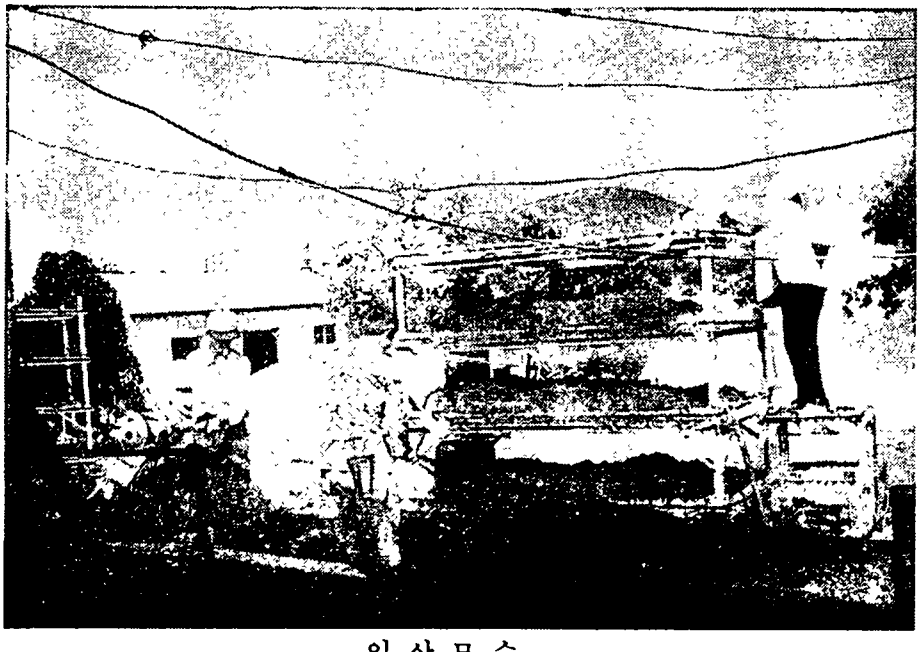
또한 재래식 버섯 재배에서 사용하여 온 고정식 재배상의 단점을 보완하고자 이동식 재배상을 설계, 제작하였다. 이동식 재배상에서는 공기의 유통이 원활하였으며, 살균시에 사용하는 연료의 소비량을 40-60% 정도 절약할 수 있었다. 재배상 연면적 50 평을 기준으로 하여 800-1000 L의 경유가 소비되는 점을 감안하면, 상당한 절약 효과가 있음을 알 수 있다.

마지막으로 컴퓨터를 이용하여, 재배사의 환경을 원격 감지 및 제어할 수 있는 자동화 시스템을 구성하였다. 컴퓨터를 이용하여 온도, 습도, 탄산가스를 원격 감지하고, 이 데이터들은 컴퓨터 모니터를 통해 실시간(real-time)으로 관측하고, disk에 저장하도록 하였다. 따라서 재배사에 출입하는 빈도수가 감소하였으며, 노동력을 절약하는 효과를 보았다. 이 데이터들은 재배 결과를 진단하는 기초 자료로 사용하고자 한다. 또한 다년간 데이터를 수집하고 재배 결과와 비교 분석하여, 최적의 재배 조건을 제시하고자 한다.

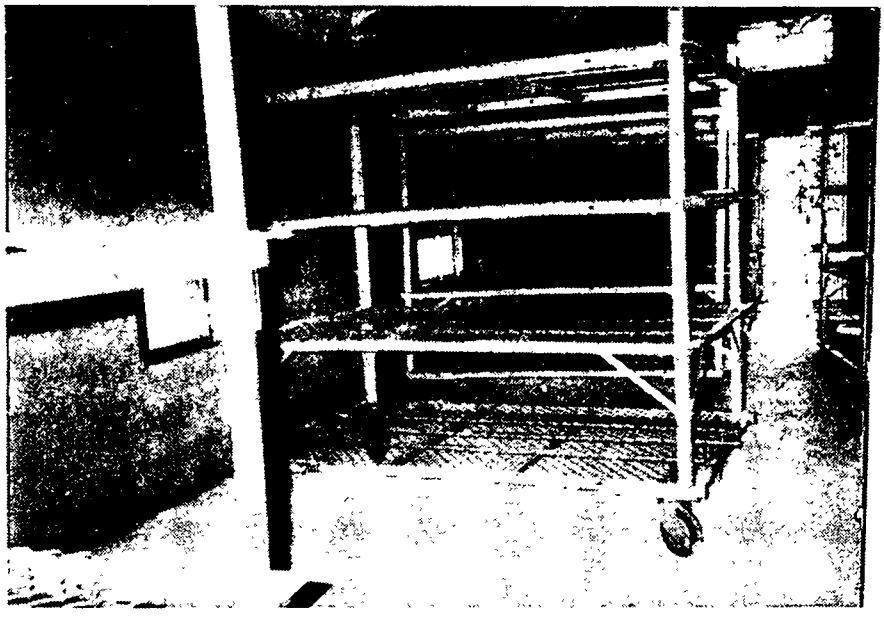
사진으로 보는 개발 내용



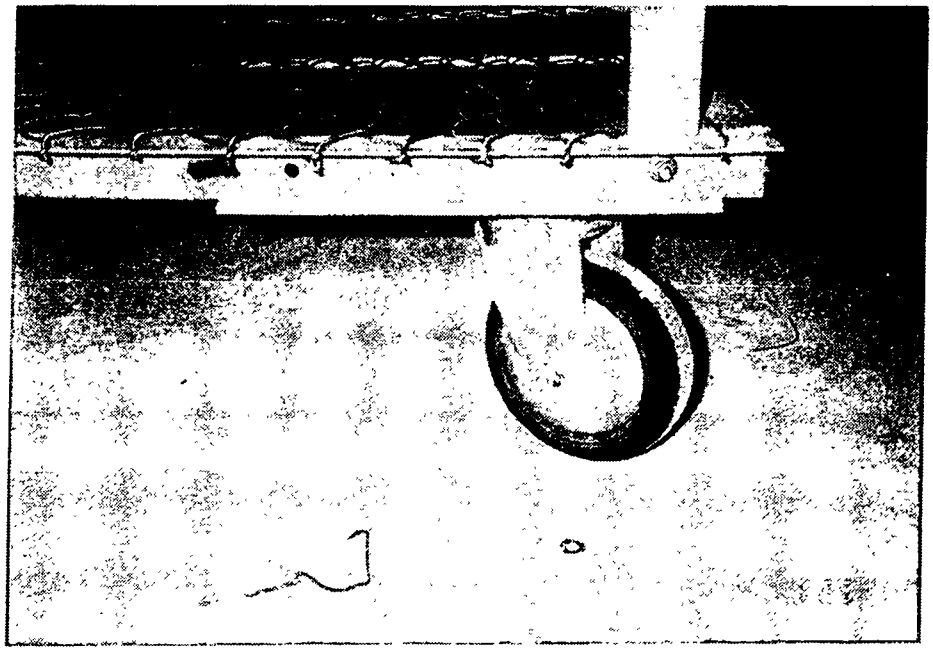
재배사 전경



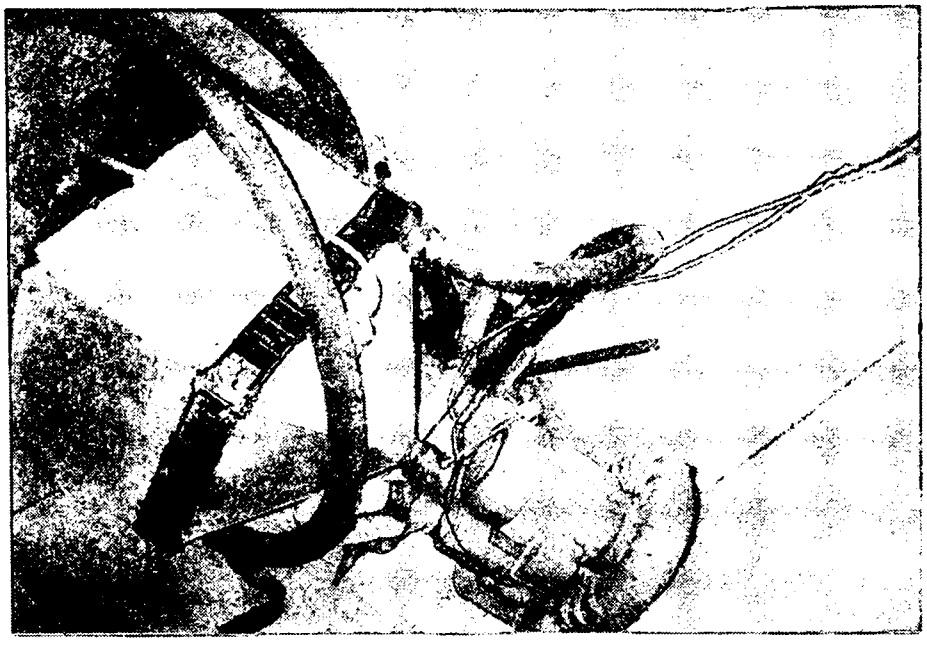
입 상 모 습



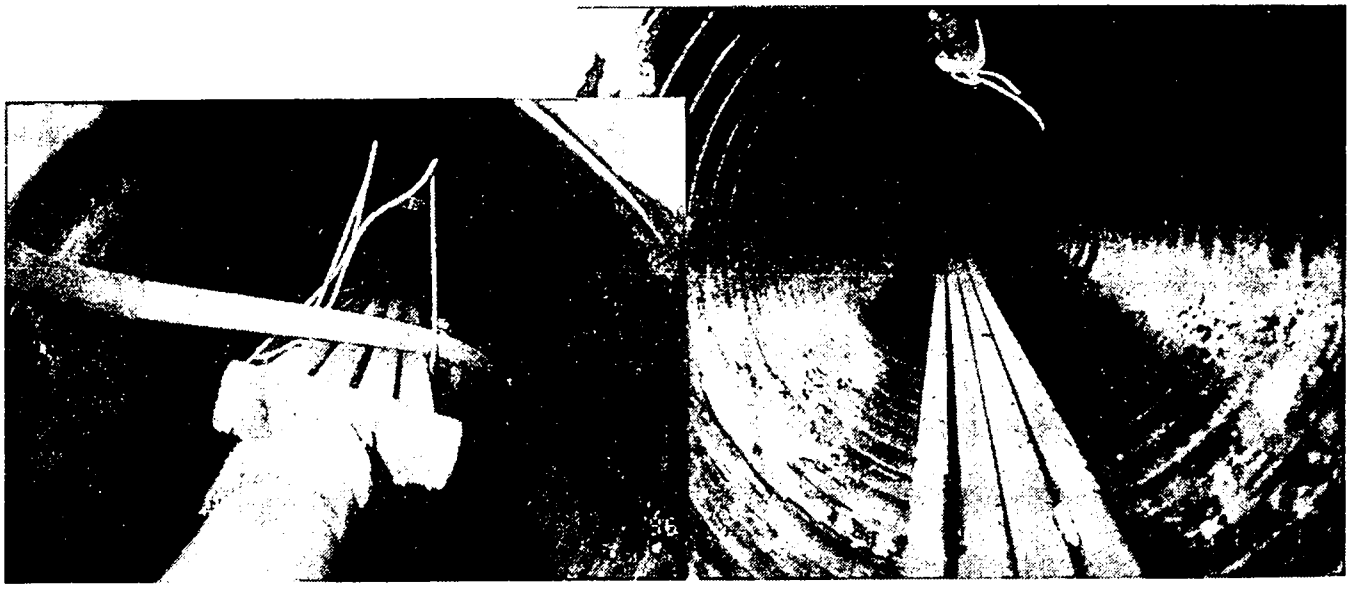
재 배 상



이동식 재배상 조립

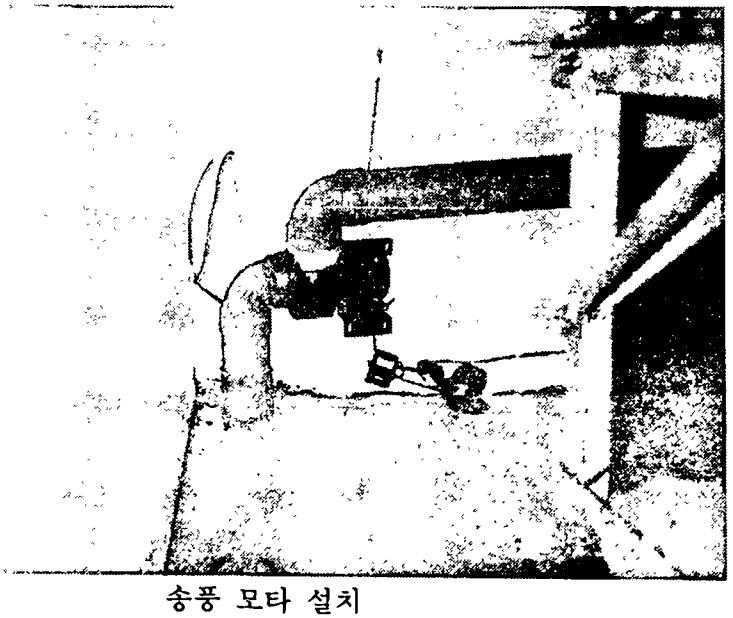


기계실 전경

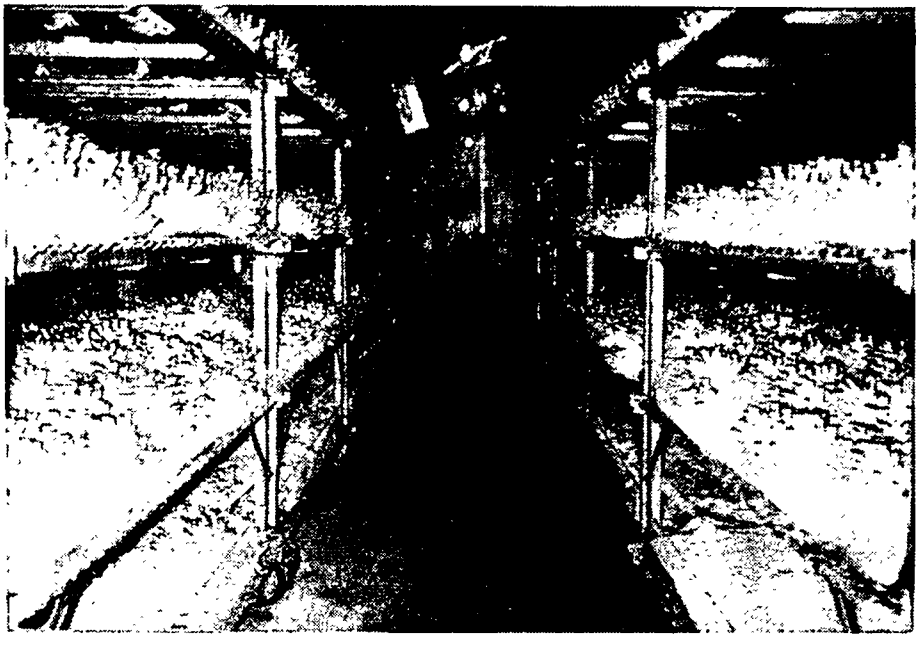


수조와 기포발생장치

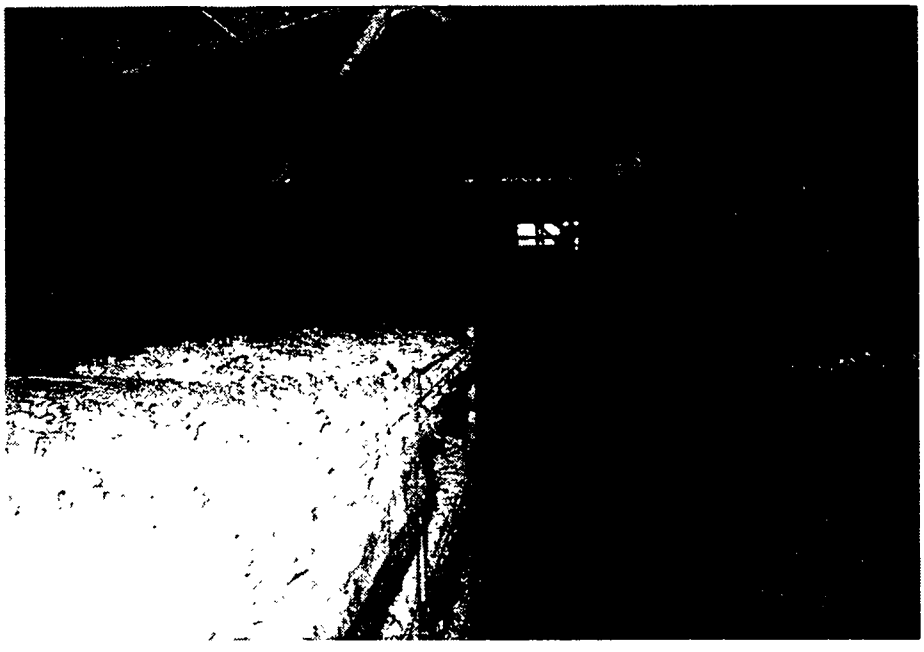
수조 및 퇴수로 매설

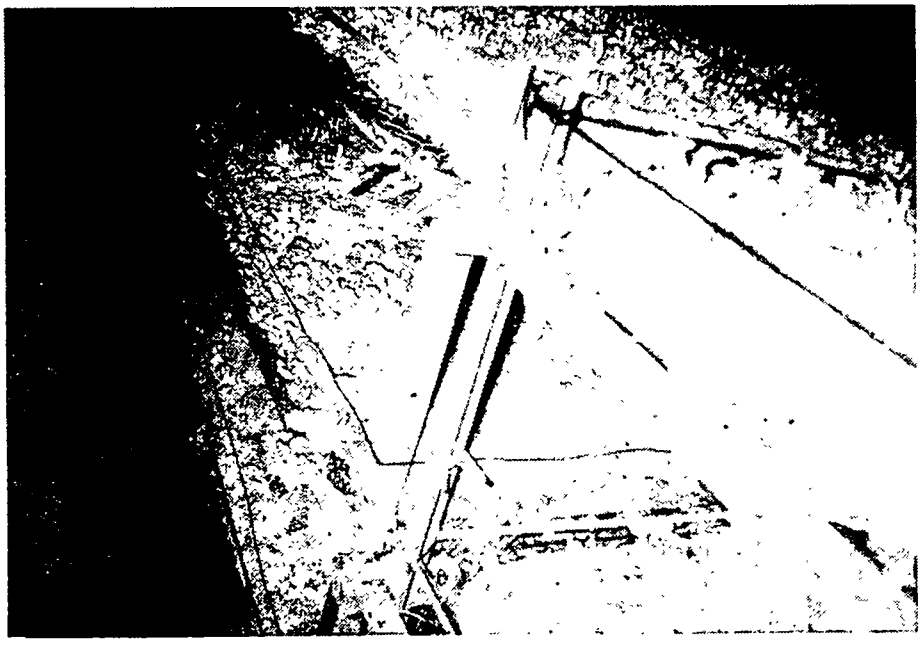


송풍 모타 설치



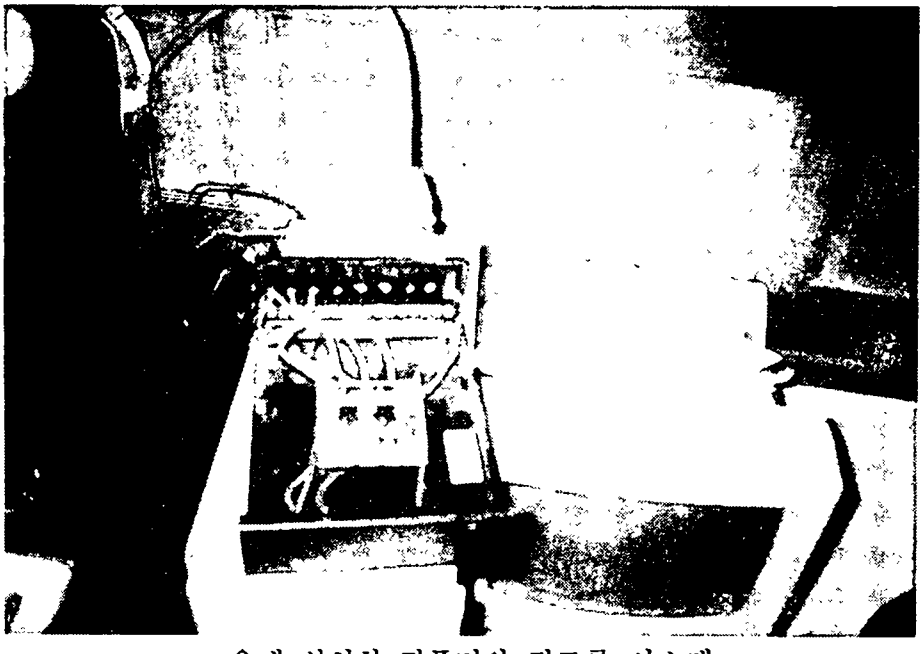
버섯 재배 전경



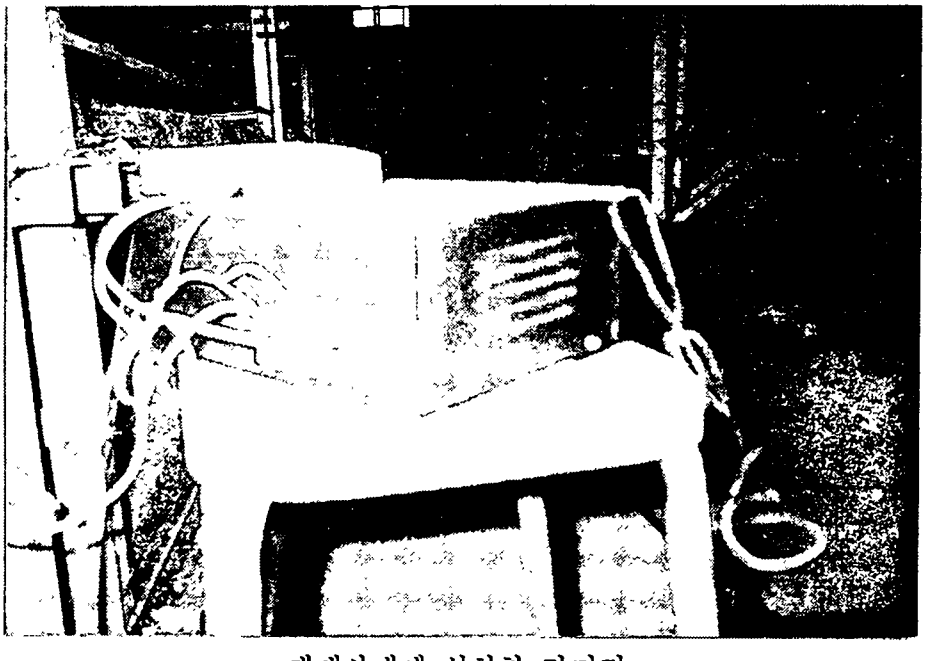


96년 동절기 재배 모습

컴퓨터 자동화의 장면



옥내 설치한 컴퓨터와 컨트롤 시스템



재배사내에 설치한 감지기

참 고 문 헌

[1] 이병식, 온도, 습도 *센서 활용 기술* (세운, 서울, 1988).

[2] 차영배, *Micro Controller 80196* (다음세대, 서울, 1996).

[3] D. Pippenger, Electronic Products, November, 64-70 (1985)

[4] 조순탁, *통계물리학* (교학연구사, 서울, 1984).

[5] 김성천, *논리회로의 설계 및 응용* (생능, 서울, 1989).

[6] 황규섭, *센서 활용 기술* (기전연구사, 서울, 1989).

별 첨 1

요 약 문

Ⅰ. 제목

버섯 재배 자동화 시스템

Ⅱ. 연구개발의 목적 및 중요성

1. 목 적

1) 컴퓨터에 의한 다수 원격 재배지의 자동화 및 무인화

2) 각종 센서에 의한 재배지의 과학적 분석

3) 센서로 인한 각종 데이터의 영구 저장과 분석

2. 중요성

1) 무인화로 인한 인력절감

2) 농가의 컴퓨터 보급과 함께 저가격의 원격 관리 가능

3) 동시에 여러 재배지의 상태 관찰이 용이하여 보다 효율적인 관리가 가능하다.

Ⅲ. 연구개발 내용 및 범위

본 연구에서는 컴퓨터로 다수 재배지의 환경을 제어하고 감시하여 원거리에 있는 PC에 데이타를 보내어 집안에서 데이타를 분석 및 재배지의 환경을 제어할 수 있도록 시스템을 구현했다. 센서로는 온도, 습도, CO_2센서를 사용하였다. 각종 센서에서의 값을 컴퓨터에서는 A/D 컨버터로 값을 변환하여 컴퓨터 모니터상에서 사용자가 재배지의 상태를 쉽게 파악할 수 있다. 또한 사용자가 원하는 재배지의 상

태를 컴퓨터에 지시하면 원하는 상태까지 재배지의 환풍기와 보일러를 가동하게 설계되었다.

Ⅳ. 연구개발 결과 및 활용에 대한 건의

센서의 정확도는 기존의 상품화된 제품과 비슷하며 원거리의 데이터 전송률은 100%이며 데이터를 변환하는 A/D컨버터는 12 bits의 분해능을 갖어 기존의 상용 제품의 8 bits나 10 bits의 제품보다 정확한 재배지의 상태를 알 수 있다. 또한 기존의 제품은 재배지안에서만 사용 가능하게 되어 있지만 본 연구에서 사용한 것은 원거리의 재배지를 갖고 집안에서 재배지의 상태를 정확히 파악할 수 있다. 그리고 데이터의 처리과정에서도 기존의것은 영구저장이 거의 불가능하지만 이것은 데이터의 영구저장과 데이터의 정확한 분석이 가능하고 사용자가 쉽게 이해할 수 있게 컴퓨터 모니터상에 그래프로 상태를 표현하고 있다.

21세기 정보화 시대에서의 능동적으로 대처하는 농가를 실현하기 위해서 컴퓨터의 보급이 절대적이다. 이러한 컴퓨터의 보급으로 다양한 형태로 농가에 접근할 수 있다. 본 연구는 농가에서의 컴퓨터의 활용의 촉진 방향과 이로인한 직접적인 효과를 거둘수 있는 것 중의 하나로 버섯재배의 자동화 및 무인화를 추구한다. 컴퓨터가 널리 보급되면 본 실험장치는 저가에서의 최대효과를 볼 수 있을 것으로 사료된다.

I. 과제명 : 버섯 재배 자동화 시스템

II. 현장애로 기술개발사업을 추진하게 된 사유

현재 국내의 버섯재배는 시설재배사를 이용하여 버섯생육에 적합한 최적환경을 조성하는 것으로 가장 중요시되고 있는 것으로는 재배사내의 온도, 습도, 환기라 할 수 있다. 그 중에서 가장 관심 있는 사항이 온도, 습도이며 관리하기도 까다로운 것이 아닐 수 없다. 문제는 습도나 탄산가스를 측정하는 것이 문제의 지름길인데, 신뢰성 및 설치 비용면에서 해결의 과제가 된다.

우리의 일상생활에서 장마 때가 되면 곰팡이가 생기거나 인체가 나른해져서 습도의 영향이 현저히 나타난다. 전자 기기에서도 같은 상황에서 신뢰성이 높다는 공업용 계기에서도 장마철에 고장이 자주 있다. 즉 내습성은 기기의 신뢰성을 좌우하는 중요한 척도이고, 최근 전자기기 등의 신뢰성 시험 항목에 습도가 포함되어 있다. 이러한 이유에서 센서의 설치부는 습도에 강한 센서를 사용해야 한다. 온도 센서부는 저가이면서 고정확도를 자랑하는 백금저항체를 사용하였다.

공기중에 포함되는 수분량이 공기량에 비해 현저히 적은 양이고 수중기가 측정재료에 미치는 과정이 물리적, 화학적으로 복잡하여 고정확도의 측정이 불가능하다. 하지만 습도센서로는 약간의 비선형성을 갖지만 저가이지만 재배지의 특성상 70%에서 그 이상의 습도 측정만을 요구하므로 외장형 H204C를 사용하였다

채소의 재배에 있어서 탄산가스 관리는 적은 투자로 생산량을 늘릴 수 있는 확실한 방법이다. 버섯재배는 동일인이 재배한 버섯재배사도 방향에 따라 생산량에 차이있다고 보고된다. 그것은 환기량의 차이를 뜻하며 버섯재배에 있어서 환기는

탄산가스의 관리를 뜻한다. 버섯재배사내 탄산가스 관리는 연속적 온도, 습도관리와 함께 버섯 품질을 높일수 있는 유일한 조건이다. 탄산가스 분석기는 첨단 광학장치로 조합구성된 정확도가 높은 측정방법을 사용하였다.

III. 연구배경

최근 농작물을 재배하기 위한 자동화 시스템이 많이 개발되었지만 원거리에서의 제어는 아직 미약하고 특히 PC와 PC간의 통신을 하는 시스템은 고가의 장비를 요구하므로 경제적인면에서 농가의 입장에서는 투자하기 어려운 실정이다. 이러한 약점을 보완하기 위하여 본 연구에서는 컴퓨터로 재배지의 환경을 제어하고 감시하여 원거리에 있는 PC에 데이타를 보내어 집안에서 데이타를 분석 및 재배지의 환경을 제어할 수 있도록 시스템을 구현했다.

IV. 시스템구성

1. 하드웨어

하드웨어는 크게 센서부, 전송부, 데이터획득부, 데이터처리부, 자동화부로 구분된다. 센서부는 온도센서, 습도센서, CO_2센서로 구성되는데 온도센서는 온도의 변화에 따라 저항의 변화를 검출하는 PT100을 사용하였으며 선형적인 출력을 갖는 것이 장점이다. 이 센서를 이용하여 구동회로를 구성하였는데 10,0 mV/℃의 출력특성을 갖게 열특성과 안정동작을 하는 OP400의 OP-AMP를 사용하였다. 습도센서 H204C는 공기 조절기기 등의 습도제어 시스템용으로 개발된 습도센서로서 인가된 바이어스에 의해서 저항이 가변되고 이 변화값을 검출하여 증폭하게 된다. CO_2 센서는 Vaisalae사의 GMW120을 사용하였다. 이것은 마이크로 프로세서가 내장되어 있어 정확한 값을 검출하고 원거리에서 필요한 전류전송방식을 취하고 있다.

전송부는 기존의 공장 내에서의 데이터 전송방식에서 널리 사용하고 있는 전류 전송방식을 사용하였다. 원거리에서의 전압 전송방식으로 취하면 전선에서의 감소 특성 때문에 정확한 데이터를 얻기가 불가능하다. 이러한 점을 보완한 것이 전류 전송방식인 4-20 mA 전송루프방법이다. 전송부에서는 센서부에서 전압으로 출력되는 값을 전류로 바꾸어 전송가능하도록 설계되었다.

표 1. CIO - DAS 800의 특성.

전력 소모	+5V	107mA / 180mA
	+12V	8mA
	-12V	6mA
아날로그 입력	Channels	8 Single Ended
	Resolution	12Bits, 4095 divisions of full scale
	Accuracy	0.01% of reading +/- 1 bit
	Type	Successive approximation
	Speed	20uSec - AD674
	Monotonicity	Guaranteed over operating temp
	Linearity	+/- 1bit
	Ranges	Programmable - See charts
	Overvoltage	+/- 30Volts Continuous
	Input Current	100nA max @ 25 deg. C.
	Inout Impedance	10 Meg Ohms
	Gain Temp Coef	+FS +/- 25 ppm/deg C
		-FS +/- 10 uV/deg C
	Zero Drift	10 ppm/deg C max
	Gain Drift	50 ppm/deg C max
샘플 & 홀드	Acquistion Time	15 uSec to 0.01 %
	Dynamic error	1 bit @ 2000 V/Sec
Reference 전압 출력	Reference	+10 Volts +/- 0.1 V
	Temp Coef	50 ppm/deg C max
	Load Current	2 mA max
Digital I/O	OP1 - OP4 low	0.5 V max @ 8 mA current sink
	OP1 - OP4 high	2.7 V min @ -0.4 mA current source
	IP1 - IP3 low	0.8 V max
	IP1 - IP3 high	2 mA @ 20 uA
Interrupt Input	Type	Positive edge triggered
	PC bus IRQ	IRQ2 - IRQ7

표 2. CO_2 transmitter GMW120의 특성.

Carbon dioxide	Measument range	0 - 2000 ppm Co2
	Accuracy at 20℃	<±[20ppm+2% of the reading]
	Temperature dependence of output	<2ppm/℃
	Long term stability	<1100ppm/2 years
	Responce time	<60 seconds
Operating conditions	Operating temperature	+10 - +35℃
	Storage temperature	-20 - +70℃ (non-condensing)
	Sensor Protection	difusion filter
Electronics	Analogue outputs	4 - 20mA, 0 - 10V
	Recommended external load : current output	max. 500Ω
	voltage output	min 1K
	Power suppy	24 VAC/VDC (16 - 30 VAC/VDC)
	Power consumption	< 3W
	Warm-up time	< 30 minutes
Mexhanics	Housing materal	ABS plastic
	Dimensions	80 x 140 x 35 mm
	Weight	110g

데이터획득부에서는 원거리에서 전류로 전송되는 값을 다시 전압으로 바꾸어 컴퓨터에서 인식할 수 있는 디지탈값으로 바꾸어 주어야 하는데 이것을 위해서 A/D 컨버터를 사용하였다. 이것은 12 bits의 분해능을 갖는 CIO-DAS800을 사용하였다. 데이터 처리부는 A/D 컨버터에서 검출된 값을 사용자가 알아 볼 수 있도록 표현하는 방법으로 C-언어를 사용하여 모니터상에서 쉽게 파악할 수 있도록 그래픽모드로 표현을 하였다. 자동화부는 사용자가 원하는 재배지의 상태를 얻고자 재배지에 사용된 보일러나 환풍기를 동작할 수 있게 컴퓨터와 연결된 장치부이다.

이 자동화부는 컴퓨터에서 지시하면 A/D 컨버터에서 릴레이에 전원을 켜라는

신호가 인가되고 이 릴레이는 대전류를 쉽게 제어가능한 마그네틱 스윗치를 작동하게 되고 이로 인해서 재배지의 보일러나 환풍기를 동작하게 된다. 여기서 사용된 것은 LG사의 마그네틱 스윗치이다

전체적인 시스템 구성도는 다음 그림 1과 같다.

2. 소프트웨어

1) 흐름도

그림 6 참조

2) 프로그램 설명

하드웨어적으로 제작된 시스템을 구동하기 위하여 프로그램 언어인 C 언어로 프로그램을 했다. 이 프로그램은 일반 사용자가 사용하기 편리하게 하기 위해서 기능키 몇 개만 가지고도 조작이 가능하도록 구성하였다. 또한 각 센서의 데이터 변화를 한 눈에 볼 수 있도록 그래프화하였으며, 과거의 데이터를 저장하여 앞으로 버섯을 재배하는 데에 참고 자료로 활용할 수 있도록 하였다.

프로그램 동작 순서는 먼저 시스템 날짜를 읽어 들여서 읽어들인 날짜가 정확한지를 확인한 후 온도, 습도, CO_2센서 값이 버섯재배에 적당한 기준온도를 설정한다. 그리고 재배사에 여러개의 센서를 두었을 경우 확장이 가능하도록 했고, 여러개의 재배사를 두었을 경우 이를 위해 확장이 가능하도록 프로그램화할 수 있다. 또한 데이터 검출 시간을 자유롭게 조절할 수 있도록 하였다(10 - 3600 초).

이러한 환경 설정이 끝나면은 바로 각 센서로부터 데이터를 읽어들여 모니터상에 디스플레이하고 데이터를 저장하고 설정한 기준치와 비교하여 각종 모터를 제어할 수 있도록 프로그램화하였다.

참고로 PC의 사양에 따라 그래픽용 프로그램이 동작이 잘 안될 수도 있다. 이를 해결하기 위해서 텍스트용 프로그램을 프로그램하였다. 그리고 보다 안정적인 시스템을 갖춘 PC를 사용하여야 한다.

3) 프로그램 리스트

별첨 2 참조

V. 개발 가능성

1. 통신 방법에서 RS-422의 통신방법을 이용하면 원거리 약 2 km의 거리를 갖는 재배지의 원격제어가 가능하다.
2. 다채널 A/D 변환기를 사용하면 다수의 재배지의 환경의 검출과 제어가 가능하다.
3. 소프트웨어의 윈도우즈 환경에서 실행이 가능하면 재배지의 제어와 동시에 다른 작업 예를 들면 워드프로세서 또는 통신등 여러 가지 작업이 가능하여 농가의 컴퓨터 보급에 영향을 줄것임.
4. 다수의 센서를 재배지의 각부분에 위치하면 정확한 재배지의 환경을 검출할 수 있다.

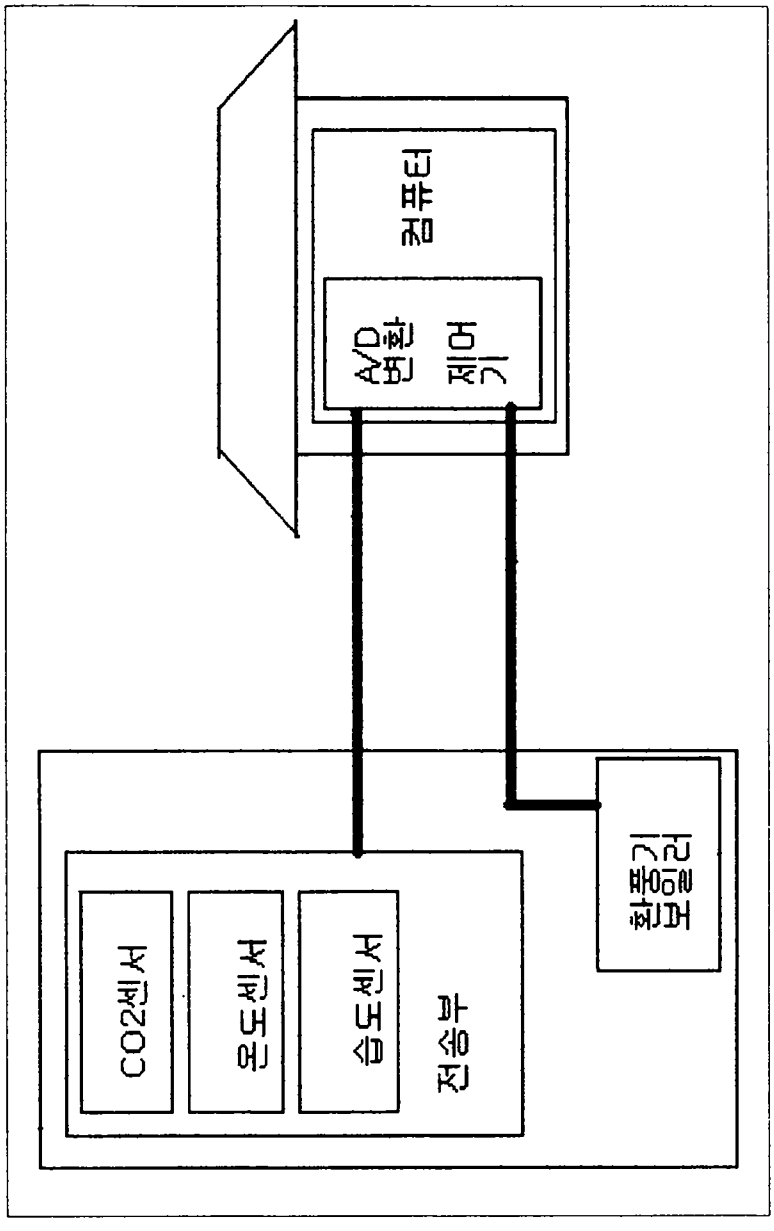


그림 1. 시스템 구성도.

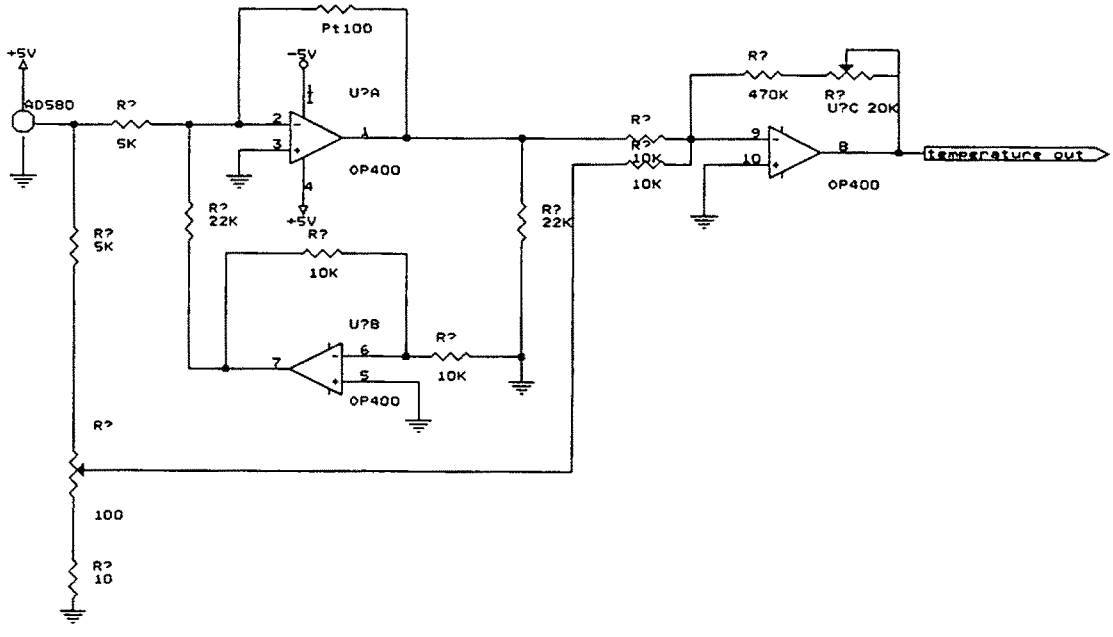


그림 2. 온도 검지기의 회로도.

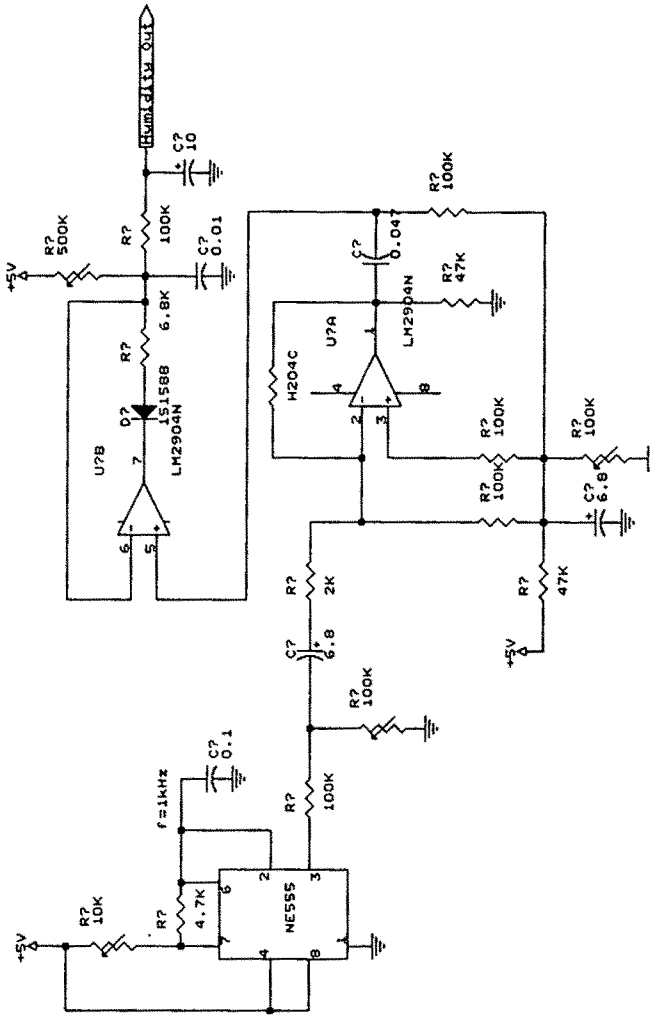


그림 3. 습도 검지기의 회로도.

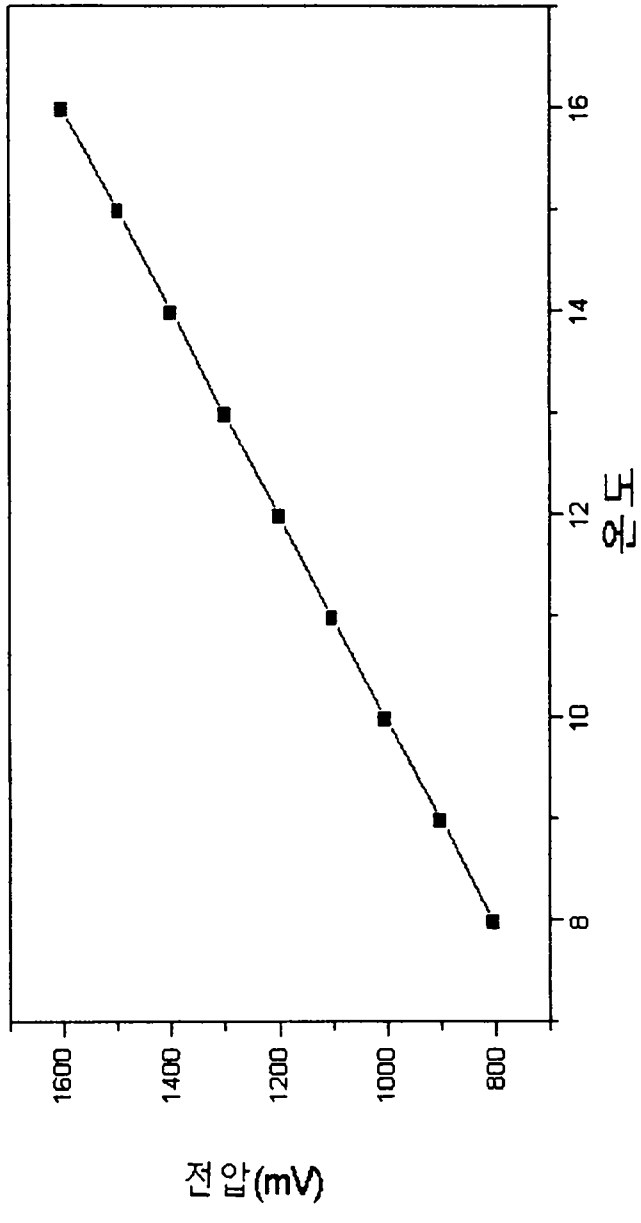


그림 4. 온도 검지기의 온도에 따른 출력 전압.

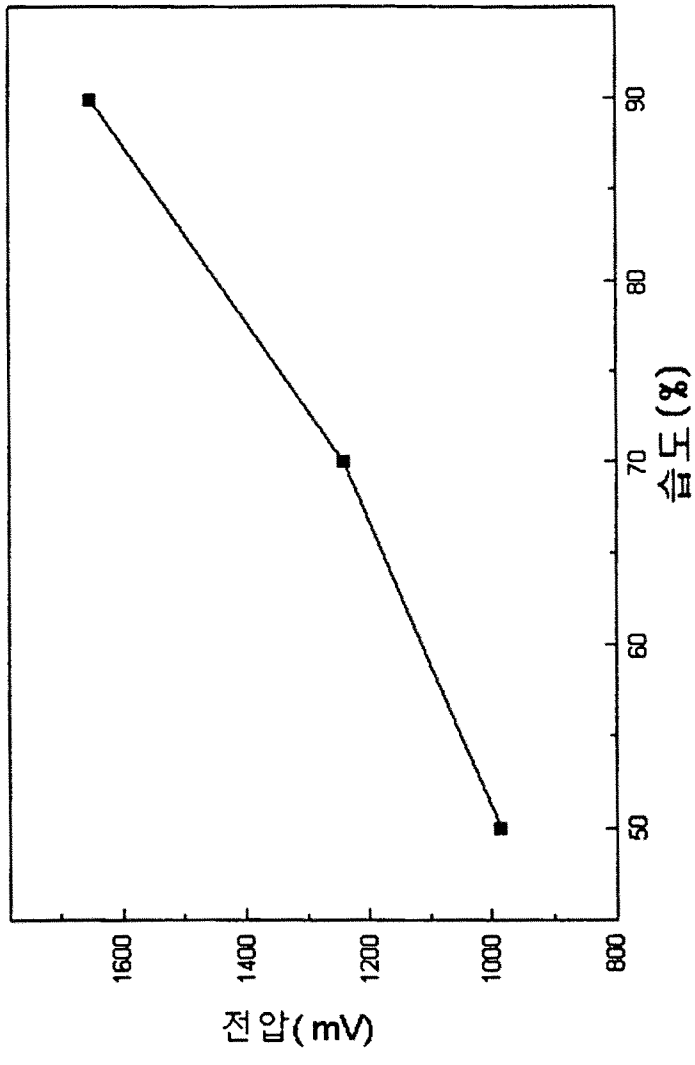


그림 5. 습도 검지기의 습도에 따른 출력 전압.

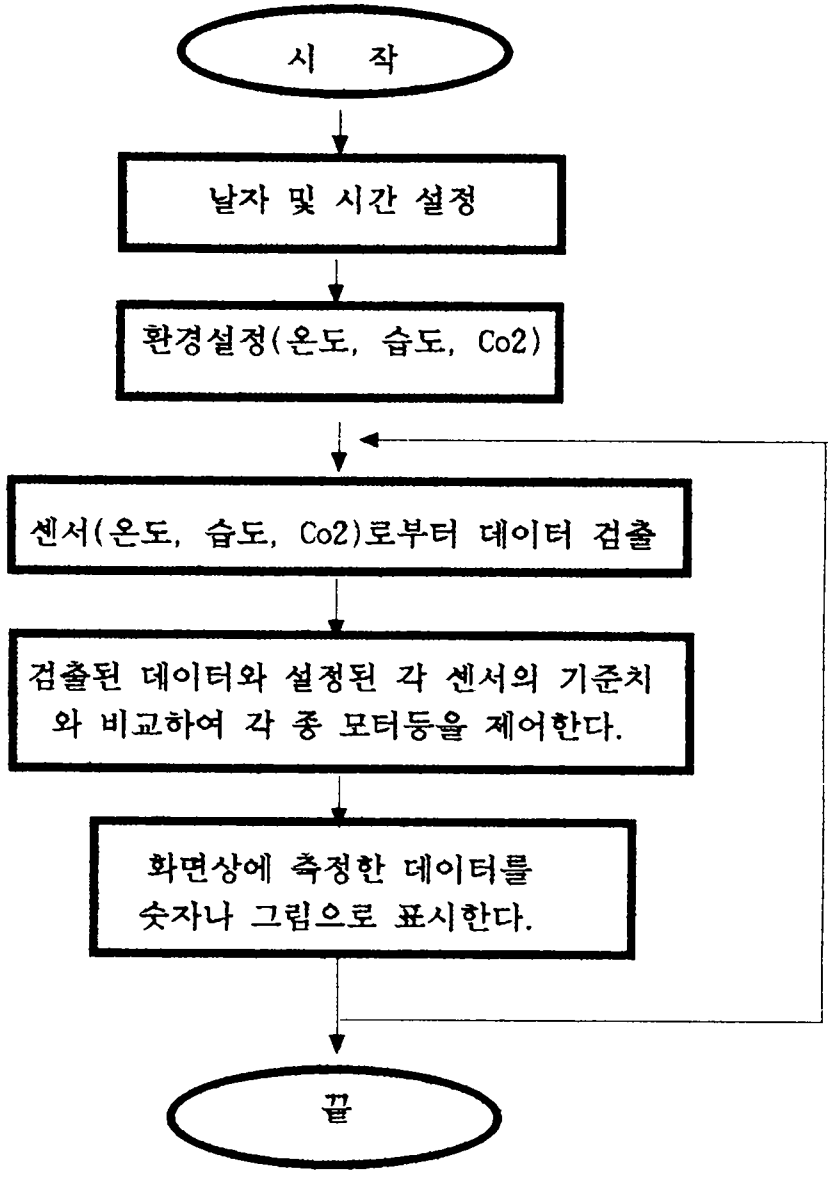


그림 6. 컴퓨터 운영 프로그램의 흐름도.

별 첨 2

```
/* adg.c */
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <dir.h>
#include <graphics.h>
#include "seg-time.h"
#include "adlogo.h"
#include "xyplane.h"

int detect_time_interval = 10;

void main(void)
{
    int gd=DETECT, gm;
    int key;

    initgraph(&gd, &gm, "");
    setviewport(0, 0, getmaxx(), getmaxy(), 0)
    logo_driver();
    system_date_time_init();
    sys_config_init();
    logo_screen(1, CYAN);
    while(1) {
      key = xy_plane();
      if(key != -1) break;
      }
    closegraph();
}


/* adlogo.c */
#include <string.h>
```

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <dir.h>
#include <alloc.h>
#include <bios.h>
#include <graphics.h>
#include "keys.h"
#include "adlogo.h"
#include "seg-time.h"
#include "xyplane.h"

extern int seg_xy[2][2];

int temp_xy[2][2];

extern int detect_time_interval;
extern void da(int chn);

int sensor_range[6][3];
int sensor_select_state[6];

static char *onoff[] = {" OFF ", " ON  "};
static char *yesno[] = {"YES", "NO"};


void color(int fg, int bg)
{
    setbkcolor(bg);
    setcolor(fg);
}

void textxy(int x, int y, char *st)
{
    outtextxy(x, y, st);
}
```

```
int get_key(void)
{
    int lo, hi,key;

    key = bioskey(0);
    lo = key & 0x00FF;
    hi = (key & 0xFF00) >> 8;
    return ((lo==0) ? 256 + hi : lo);
}

void key_press_wait(long int length)
{
    long int i = 0L;
    int done = 0;

    while(1) {
      if(++ i > length) {done = 1; break;}
      else if(kbhit()) break;
      }
    if(!done) get_key();
}

void logo_title(int x1, int y1, int x2, int y2)
{
    win_box3d(0, 0, x2 - x1, y2 - y1, BLUE, 1);
    color(14, YELLOW);
    textxy(20,  30,  " A Mushroom Automatic System Version 1.0 ")
    textxy(25,  70,  " Copyright  Kim Jahwan & Lee YoungJu ");
    textxy(15,  110, " Graduate Dept. of Electronic & Computer ");
    textxy(50,  130, " Engineering  Univ. of Mokwon ");
    color(RED, 1);
    textxy(110, 160, " Press any key.... ");
}

void logo_screen(int fill, int color)
{
    setviewport(0, 0, getmaxx(), getmaxy(), 0);
    win_box3d(0, 0, getmaxx(), getmaxy(), color, fill);
```

```
}

void logo_driver(void)
{
    int x1, y1, x2, y2;

    logo_screen(1, CYAN);
    x1 = getmaxx() / 2 - 180;
    y1 = getmaxy() / 2 - 90;
    x2 = getmaxx() / 2 + 180;
    y2 = getmaxy() / 2 + 90;
    setviewport(x1, y1, x2, y2, 0);
    logo_title(x1, y1, x2, y2);
    key_press_wait(10000L);
    setviewport(0, 0, getmaxx(), getmaxy(), 0);
    logo_screen(1, CYAN);
}

void win_box3d(int x1, int y1, int x2, int y2, int color, int fill)
{
    if(fill == 1) {
      setwritemode(COPY_PUT);
      setfillstyle(1, color);
      bar(x1, y1, x2, y2);
      }

    setcolor(BLACK);
    rectangle(x1, y1, x2, y2);

    setcolor(WHITE);
    line(x1, y2, x1, y1);
    line(x1 + 1, y1 + 1, x1 + 1, y2 - 1);

    line(x1, y1, x2, y1);
    line(x1 + 1, y1 + 1, x2 - 1, y1 + 1);

    setcolor(DARKGRAY);
    line(x1, y2, x2, y2);
```

```
    line(x1 + 1, y2 - 1, x2 - 1, y2 - 1);

    line(x2, y2, x2, y1);
    line(x2 - 1, y2 - 1, x2 - 1, y1 + 1);
}

int system_date_time_init(void)
{
    int x1, y1, x2, y2;
    int key;

    x1 = getmaxx() / 2 - 180;
    y1 = getmaxy() / 2 - 90;
    x2 = getmaxx() / 2 + 180;
    y2 = getmaxy() / 2 + 90;
    setviewport(x1, y1, x2, y2, 0);
    key = dt_display(x1, y1, x2, y2);
    setviewport(0, 0, getmaxx(), getmaxy(), 0);
    return key;
}

int dt_display(int x1, int y1, int x2, int y2)
{
    long int i = 0L;
    int hour, min, sec, ssec, year, month, day;
    int key, dt_select_mode = 1;
    int x11, y11, x22, y22;
    void *map;

    seg_date_time_cordinate(0);
    dt_text_display(x2 - x1, y2 - y1);
    xy_trans(&x11, &y11, &x22, &y22, dt_select_mode)
    seg_date_time_init();

    segtotemp_xy();
    seg_date_time_cordinate(1);
    get_time(&hour, &min, &sec, &ssec);
    seg_time(hour, min, sec, ssec);
```

```
get_date(&year, &month, &day);
seg_date(year, month, day);
temptoseg_xy();

map = malloc(imagesize(x11, y11, x22, y22));
getimage(x11, y11, x22, y22, map);
win_box3d(x11, y11, x22, y22, BLUE, 0);
while(1) {
  get_time(&hour, &min, &sec, &ssec);
  seg_time(hour, min, sec, ssec);
  get_date(&year, &month, &day);
  seg_date(year, month, day);
  key = set_date_time(year, month, day, hour, min, sec, ssec, &dt_select_mode)
  if(key == KEY_RIGHT || key == KEY_LEFT) {
    putimage(x11, y11, map, COPY_PUT);
    free(map);
    xy_trans(&x11, &y11, &x22, &y22, dt_select_mode);
    map = malloc(imagesize(x11, y11, x22, y22));
    getimage(x11, y11, x22, y22, map);
    win_box3d(x11, y11, x22, y22, BLUE, 0);
    seg_date_time_init();
    }
  else if(key == ENTER) {
    putimage(x11, y11, map, COPY_PUT);
    free(map);
    segtotemp_xy();
    seg_date_time_cordinate(1);
    if(dt_select_mode == 1)
      date_time_input(year, month, day, hour, min, sec,
        seg_xy[dt_select_mode][0], seg_xy[dt_select_mode][1], dt_select_mode);
    else date_time_input(year, month, day, hour, min, sec,
        seg_xy[dt_select_mode][0], seg_xy[dt_select_mode][1], dt_select_mode);

    temptoseg_xy();
    map = malloc(imagesize(x11, y11, x22, y22));
    getimage(x11, y11, x22, y22, map);
    win_box3d(x11, y11, x22, y22, BLUE, 0);
    i = 0L;
```

```
        }
      i ++;
      if(key == SPACE || key == ESC || i >= 500L) break;
      }
    putimage(x11, y11, map, COPY_PUT);
    free(map);
    return key;
}

void date_display(int n, int date)
{
    int year_date[4];

    if(n == 0) {
      year_date[0] = date / 1000;
      year_date[1] = (date - 1000 * year_date[0]) / 100;
      year_date[2] = (date - 1000 * year_date[0] - 100 * year_date[1]) / 10;
      year_date[3] = date - 1000 * year_date[0] - 100 * year_date[1] - 10 *
year_date[2];
      seg_countor(1, 0, seg_xy[1][0], seg_xy[1][1], year_date[0]);
      seg_countor(1, 1, seg_xy[1][0], seg_xy[1][1], year_date[1]);
      seg_countor(1, 2, seg_xy[1][0], seg_xy[1][1], year_date[2]);
      seg_countor(1, 3, seg_xy[1][0], seg_xy[1][1], year_date[3]);
      }
    else if(n == 1) {
      seg_countor(1, 4, seg_xy[1][0], seg_xy[1][1], date / 10);
      seg_countor(1, 5, seg_xy[1][0], seg_xy[1][1], date % 10);
      }
    else if(n == 2) {
      seg_countor(1, 6, seg_xy[1][0], seg_xy[1][1], date / 10);
      seg_countor(1, 7, seg_xy[1][0], seg_xy[1][1], date % 10);
      }
}

void time_display(int n, int time)
{
    if(n == 0) {
      seg_countor(0, 0, seg_xy[0][0], seg_xy[0][1], time / 10);
```

```
      seg_countor(0, 1, seg_xy[0][0], seg_xy[0][1], time % 10);
      }
    else if(n == 1) {
      seg_countor(0, 2, seg_xy[0][0], seg_xy[0][1], time / 10);
      seg_countor(0, 3, seg_xy[0][0], seg_xy[0][1], time % 10);
      }
    else if(n == 2) {
      seg_countor(0, 4, seg_xy[0][0], seg_xy[0][1], time / 10);
      seg_countor(0, 5, seg_xy[0][0], seg_xy[0][1], time % 10);
      }
}

void date_time_input(int year, int month, int day, int hour, int min, int sec
int x1, int y1,
        int dt_select_mode)
{
    int number = 0, key, i, temp, temp1[4], key_done = 1;
    int select_number[3] = {0, 0, 0};
    int date[4], seg_date_count[2][3] = {{4, 6, 8}, {15, 17, 18}};
    int time[4], seg_time_count[2][4] = {{2, 4, 6, 8}, {15, 16, 17, 17}};
    int month12[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    int x11, y11, x22, y22;
    void *map;

    if(dt_select_mode == 1) {
      temp1[0] = year; temp1[1] = month; temp1[2] = day;
      }
    else {
      temp1[0] = hour; temp1[1] = min; temp1[2] = sec; temp1[3] = 0;
      }
    while(1) {
      y11 = y1 - 1;
      y22 = y1 + 25;
      if(dt_select_mode == 1) {
        date_display(number, temp1[number]);
        if(number == 0) {
          x11 = x1 - 2;
                x22 = x1 + seg_date_count[0][number] * seg_date_count[1][number]
```

```
- 2;
          }
        else {
          x11 = x1 + seg_date_count[0][number - 1] * seg_date_count[1][number
1] + 4;
          x22 = x1 + seg_date_count[0][number] * seg_date_count[1][number] - 3
          }
        }
      else {
        time_display(number, temp1[number]);
        if(number == 0) {
                x11 = x1 - 2;
            x22 = x1 + seg_time_count[0][number] * seg_time_count[1][number]
2;
          }
        else {
          x11 = x1 + seg_time_count[0][number - 1] * seg_time_count[1][number
1] + 8;
          x22 = x1 + seg_time_count[0][number] * seg_time_count[1][number] + 3
          }
        }
      if(key_done) {
        map = malloc(imagesize(x11, y11, x22, y22));
        getimage(x11, y11, x22, y22, map);
        win_box3d(x11, y11, x22, y22, 0, 0);
        key_done = 0;
        }
      while(!kbhit()) {
        if(dt_select_mode == 1) {
          get_time(&time[0], &time[1], &time[2], &time[3]);
          seg_time(time[0], time[1], time[2], time[3]);
          get_date(&date[0], &date[1], &date[2]);
          for(i = 0; i < 4; i ++) {
            if(select_number[i] == 0) temp1[i] = date[i];
            seg_date(temp1[0], temp1[1], temp1[2]);
            }
          }
        else {
```

```
            get_time(&time[0], &time[1], &time[2], &time[3]);
            for(i = 0; i < 4; i ++) {
              if(select_number[i] == 0) temp1[i] = time[i];
              seg_time(temp1[0], temp1[1], temp1[2], temp1[3]);
              }
            }
          }
        if(kbhit()) key_done = 1;
        if(key_done) {
          key = get_key();
          putimage(x11, y11, map, COPY_PUT);
          free(map);
          }
        switch(key) {
          case KEY_LEFT  : number = (-- number < 0) ? 2 : number;
                   break;
          case KEY_RIGHT : number = (++ number > 2) ? 0 : number;
                   break;
          case KEY_UP    : if(dt_select_mode == 1) {
                    if(number == 0) temp = 9999;
                    else if(number == 1) temp = 12;
                    else if(number == 2) temp = month12[temp1[1] - 1];
                    }
                   else {
                    if(number == 0) temp = 23;
                    else temp = 59;
                    }
                    temp1[number]  =  (++  temp1[number]  >  temp)  ?  temp
temp1[number];
                   select_number[number] = 1;
                   break;
          case KEY_DOWN  : if(dt_select_mode == 1)
                    temp1[number] = (-- temp1[number] < 1) ? 1 : temp1[number];
                   else
                    temp1[number] = (-- temp1[number] < 0) ? 0 : temp1[number];
                   select_number[number] = 1;
                   break;
          case ENTER       : if(dt_select_mode == 1) set_date(temp1[0], temp1[1]
```

```
temp1[2]);
                    else set_time(temp1[0], temp1[1], temp1[2], temp1[3])
                    break;
          }
        if(key == ESC || key == ENTER) break;
        }
}

void segtotemp_xy(void)
{
    int i;

    for(i = 0; i < 2; i ++) {
      temp_xy[i][0] = seg_xy[i][0];
      temp_xy[i][1] = seg_xy[i][1];
      }
}

void temptoseg_xy(void)
{
    int i;

    for(i = 0; i < 2; i ++) {
      seg_xy[i][0] = temp_xy[i][0];
      seg_xy[i][1] = temp_xy[i][1];
      }
}

void seg_date_time_cordinate(int n)
{
    if(n == 0) {
      seg_xy[0][0] = 210;
      seg_xy[0][1] = 60;
      seg_xy[1][0] = 20;
      seg_xy[1][1] = 60;
      }
    else if(n == 1) {
      seg_xy[0][0] = 210;
```

```
        seg_xy[0][1] = 120;
        seg_xy[1][0] = 20;
        seg_xy[1][1] = 120;
        }
    else if(n == 2) {
        seg_xy[0][0] = 509;
        seg_xy[0][1] = 453;
        seg_xy[1][0] = 0;
        seg_xy[1][1] = 453;
        }
}

void seg_date_time_init(void)
{
    seg_inital(0);
    seg_inital(1);
    segtotemp_xy();
    seg_date_time_cordinate(1);
    seg_inital(0);
    seg_inital(1);
    temptoseg_xy();
}

int set_date_time(int year, int month, int day, int hour, int min, int sec, int ssec
        int *dt_select_mode)
{
    int key;

    segtotemp_xy();
    seg_date_time_cordinate(1);
    seg_time(hour, min, sec, ssec);
    seg_date(year, month, day);
    if(kbhit()) {
      key = get_key();
      switch(key) {
        case KEY_RIGHT :
        case KEY_LEFT  : *dt_select_mode = (*dt_select_mode == 0) ? 1 : 0;
                 break;
```

```
        }
      }
    temptoseg_xy();
    return key;
}

void dt_text_display(int x, int y)
{
    win_box3d(0, 0, x, y, BLUE, 1);
    color(3, 1);
    textxy(120, 15, " Date / Time Set ");
    textxy(40,  40, " Current Date            Modify Date ");
    textxy(40,  95, " Current Time            Modify Time ");
    textxy(85, 160, " Press ESC or SPACE ");
}

void xy_trans(int *x11, int *y11, int *x22, int *y22, int dt_select_mode)
{
    *x11 = seg_xy[dt_select_mode][0] - 10;
    *y11 = seg_xy[dt_select_mode][1] + 60 - 7;
    if(dt_select_mode == 1) *x22 = seg_xy[dt_select_mode][0] + 146;
    else *x22 = seg_xy[dt_select_mode][0] + 140;
    *y22 = seg_xy[dt_select_mode][1] + 60 + 30;
}

void default_config(void)
{
    detect_time_interval = 10;

    sensor_range[0][1] = 0;
    sensor_range[0][2] = 30;
    sensor_range[0][0] = 20;

    sensor_range[1][1] = 70;
    sensor_range[1][2] = 100;
    sensor_range[1][0] = 85;

    sensor_range[2][1] = 500;
```

```
    sensor_range[2][2] = 700;
    sensor_range[2][0] = 600;

    sensor_range[3][1] = 0;
    sensor_range[3][2] = 20;
    sensor_range[3][0] = 10;

    sensor_range[4][1] = 70;
    sensor_range[4][2] = 100;
    sensor_range[4][0] = 90;

    sensor_range[5][1] = 500;
    sensor_range[5][2] = 700;
    sensor_range[5][0] = 600;

    sensor_select_state[0] = 1;
    sensor_select_state[1] = 1;
    sensor_select_state[2] = 1;
    sensor_select_state[3] = 0;
    sensor_select_state[4] = 0;
    sensor_select_state[5] = 0;
}

void read_config_file(FILE *cfile)
{
    int i;

    fscanf(cfile, "%d ", &detect_time_interval);
    for(i = 0; i < 6; i ++) {
      fscanf(cfile, "%d ", &sensor_range[i][0]);
      fscanf(cfile, "%d ", &sensor_range[i][1]);
      fscanf(cfile, "%d ", &sensor_range[i][2]);
      }
    for(i = 0 ; i < 6; i ++)
      fscanf(cfile, "%d ", &sensor_select_state[i])
}

void write_config_file(FILE *cfile)
```

```
{
    int i;

    fprintf(cfile, "%d ", detect_time_interval);
    for(i = 0; i < 6; i ++) {
      fprintf(cfile, "%d ", sensor_range[i][0]);
      fprintf(cfile, "%d ", sensor_range[i][1]);
      fprintf(cfile, "%d ", sensor_range[i][2]);
      }
    for(i = 0 ; i < 6; i ++)
      fprintf(cfile, "%d ", sensor_select_state[i]);
}

void config_text_display(void)
{
    color(3, 1);
    textxy(180,  10, " Configuration Set ");
    setcolor(DARKGRAY);
    line(3,  29, 557,  29);
    line(3,  30, 557,  30);
    line(3, 310, 557, 310);
    line(3, 311, 557, 311);
    color(3, 1);
    textxy( 20,  50, " Detect Sensor ");
    textxy( 50,  70, " Detect time interval :        Sec ");
    textxy( 20, 100, " Select Sensor ");
    textxy( 50, 120, " Temperarure 1 :      Standard :      Temp.  Range
   ");
    textxy( 50, 140, " Humidity    1 :      Standard :      %      Range
   ");
    textxy( 50, 160, " Co2         1 :      Standard :      ppm    Range
   ");
    textxy( 50, 180, " Temperature 2 :      Standard :      Temp.  Range
   ");
    textxy( 50, 200, " Humidity    2 :      Standard :      %      Range
   ");
    textxy( 50, 220, " Co2         2 :      Standard :      ppm    Range
   ");
```

```
    textxy(170, 320, " Press ESC or SPACE ");
}

static int cfg_onoff_cordinate[6][2] = {
    {180, 120}, {180, 140}, {180, 160}, {180, 180},
    {180, 200}, {180, 220}};

static int cfg_range_cordinate[6][4] = {
    {318, 475, 523, 120}, {318, 475, 523, 140}, {318, 475, 523, 160},
    {318, 475, 523, 180}, {318, 475, 523, 200}, {318, 475, 523, 220}};

static int cfg_other_cordinate[2][2] = {{247,  70}, {195, 270}};

static int cfg_item_cordinate[25][4] = {
    {240,  65, 283,  83},
    {177, 115, 220, 133}, {312, 115, 350, 133}, {470, 115, 500, 133}, {519, 115, 551
133},
    {177, 135, 220, 153}, {312, 135, 350, 153}, {470, 135, 500, 153}, {519, 135, 551
153},
    {177, 155, 220, 173}, {312, 155, 350, 173}, {470, 155, 500, 173}, {519, 155, 551
173},
    {177, 175, 220, 193}, {312, 175, 350, 193}, {470, 175, 500, 193}, {519, 175, 551
193},
    {177, 195, 220, 213}, {312, 195, 350, 213}, {470, 195, 500, 213}, {519, 195, 551
213},
    {177, 215, 220, 233}, {312, 215, 350, 233}, {470, 215, 500, 233}, {519, 215, 551
233}
    };

void config_data_display(int count, int i)
{
    char *st;

    color(YELLOW, 1);
    if(i == 0 || i == count - 1) {
      st = malloc(sizeof(st));
      if(i == 0) {
        itoa(detect_time_interval, st, 10);
```

```
        textxy(cfg_other_cordinate[0][0], cfg_other_cordinate[0][1], st);
        }
      free(st);
      }
    else {
      if(sensor_select_state[i - 1] == 0) {
        textxy(cfg_onoff_cordinate[i - 1][0], cfg_onoff_cordinate[i - 1][1],
            onoff[0]);
        setfillstyle(1, BLUE);
        bar(cfg_item_cordinate[i * 4 - 1][0] + 3, cfg_item_cordinate[i * 4 - 1][1] +
3,
            cfg_item_cordinate[i * 4 - 1][2] - 3,cfg_item_cordinate[i * 4 - 1][3] -
3);
        bar(cfg_item_cordinate[i * 4 - 2][0] + 3, cfg_item_cordinate[i * 4 - 2][1] +
3,
            cfg_item_cordinate[i * 4 - 2][2] - 3,cfg_item_cordinate[i * 4 - 2][3] -
3);
        bar(cfg_item_cordinate[i * 4][0] + 3, cfg_item_cordinate[i * 4][1] + 3,
            cfg_item_cordinate[i * 4][2] - 3,cfg_item_cordinate[i * 4][3] - 3);
        }
      else {
        textxy(cfg_onoff_cordinate[i - 1][0], cfg_onoff_cordinate[i - 1][1],
            onoff[1]);
        st = malloc(sizeof(st));
        itoa(sensor_range[i - 1][0], st, 10);
        textxy(cfg_range_cordinate[i - 1][0], cfg_range_cordinate[i - 1][3], st);
        free(st);
        st = malloc(sizeof(st));
        itoa(sensor_range[i - 1][1], st, 10);
        textxy(cfg_range_cordinate[i - 1][1], cfg_range_cordinate[i - 1][3], st);
        free(st);
        st = malloc(sizeof(st));
        itoa(sensor_range[i - 1][2], st, 10);
        textxy(cfg_range_cordinate[i - 1][2], cfg_range_cordinate[i - 1][3], st);
        free(st);
        }
      }
}
```

```
void config_display(void)
{
    int count = 8, i;

    config_text_display();
    for(i = 0; i < count; i ++)
      config_data_display(count, i);
}

void read_system_config(void)
{
    FILE *cfile;

    cfile = fopen("ausys.cfg", "r");
    if(cfile == NULL) {
      default_config();
      cfile = fopen("ausys.cfg", "w+");
      write_config_file(cfile);
      fclose(cfile);
      }
    else {
      read_config_file(cfile);
      fclose(cfile);
      }
}

void write_system_config(void)
{
    FILE *cfile;

    cfile = fopen("ausys.cfg", "w");
    if(cfile == NULL) {
      default_config();
      cfile = fopen("ausys.cfg", "w");
      write_config_file(cfile);
      fclose(cfile);
      }
    else {
```

```
        write_config_file(cfile);
        fclose(cfile);
        }
}

int yes_no_select(int x1, int y1)
{
    void *map1;
    int yn = 0, key;
    int x11, y11, x22, y22;
    int yn_xy[2][4] = {{260, 277, 293, 297}, {310, 277, 335, 297}}

    color(3, 1);
    textxy(x1 + 140, y1 + 15, " Save ? ");
    textxy(yn_xy[0][0] + 6, yn_xy[0][1] + 6, yesno[0]);
    textxy(yn_xy[1][0] + 6, yn_xy[1][1] + 6, yesno[1]);
    while(1) {
      x11 = yn_xy[yn][0]; x22 = yn_xy[yn][2];
      y11 = yn_xy[yn][1]; y22 = yn_xy[yn][3];
      map1 = malloc(imagesize(x11,  y11, x22, y22));
      getimage(x11,  y11, x22, y22, map1);
      win_box3d(x11, y11, x22, y22, BLACK, 1);
      color(3, 0);
      textxy(x11 + 6, y11 + 6, yesno[yn]);
      key = get_key();
      putimage(x11, y11, map1, COPY_PUT);
      free(map1);
      switch(key) {
        case KEY_LEFT  : yn = (yn == 1) ? 0 : 1;
                 break;
        case KEY_RIGHT : yn = (yn == 0) ? 1 : 0;
                 break;
        }
      if(key == ENTER) {yn = 1; break;}
      else if(key == ESC) {yn = 0; break;}
      }
    return yn;
}
```

```
int system_config_set(void)
{
    int i= 0, item_count = 24, count = 9, key = 0, done = 0;
    void *map;
    int j = 0, s = 1;

    while(1) {
      if(s) {
        map = malloc(imagesize(cfg_item_cordinate[i][0],  cfg_item_cordinate[i][1],
            cfg_item_cordinate[i][2], cfg_item_cordinate[i][3]));
        getimage(cfg_item_cordinate[i][0],  cfg_item_cordinate[i][1],
            cfg_item_cordinate[i][2], cfg_item_cordinate[i][3], map);
        win_box3d(cfg_item_cordinate[i][0],  cfg_item_cordinate[i][1],
            cfg_item_cordinate[i][2], cfg_item_cordinate[i][3], BLACK, 0);
        s = 0;
        }
      if(kbhit()) {
        key = get_key();
        putimage(cfg_item_cordinate[i][0],          cfg_item_cordinate[i][1],      map,
COPY_PUT);
        free(map);
        s = 1;
        j = 0;
        }
      switch(key) {
        case KEY_RIGHT : if(i > 0 && sensor_select_state[(i - 1) / 4] == 0) i +=
4;
                 else i = (++ i > item_count) ? 0 : i;
                 if(i > item_count) i = 0;
                 break;
        case KEY_LEFT  : if(i > 2 && sensor_select_state[(i + 1) / 4 - 1] == 0) i
-= 4;
                 else i = (-- i < 0) ? item_count : i;
                 if(sensor_select_state[(i - 1) / 4] == 0)
                  i = (i % 4 == 0) ? (i -= 3) : (i % 4 == 1) ? i :
                    (i % 4 == 2) ? (-- i) : (i -= 2);
                 break;
        case KEY_UP    : if(i == item_count) i -= 4;
```

```
            else if(i == 0) i = item_count;
            else if(i > 4 ) i -= 4;
            else i = 0;
            if(i > 0 && sensor_select_state[(i - 1) / 4] == 0)
             i = (i % 4 == 0) ? (i -= 3) : (i % 4 == 1) ? i :
                (i % 4 == 2) ? (-- i) : (i -= 2);
            break;
case KEY_DOWN  : if(i == 0) i ++;
            else if(i < item_count) i += 4;
            else i = 0;
            if(i > item_count) i = 0;
            if(i != 0 && sensor_select_state[(i - 1) / 4] == 0)
             i = (i % 4 == 0) ? (i -= 3) : (i % 4 == 1) ? i :
                (i % 4 == 2) ? (-- i) : (i -= 2);
            break;
case KEY_PGDN  :
case KEY_PGUP  : setfillstyle(1, BLUE);
            bar(cfg_item_cordinate[i][0] + 3, cfg_item_cordinate[i][1] + 3,
             cfg_item_cordinate[i][2] - 3,cfg_item_cordinate[i][3] - 3);
            if(i == 0) {
             if(key == KEY_PGUP) detect_time_interval =
                (detect_time_interval > 590)?600:(detect_time_interval += 5)
             else detect_time_interval = (detect_time_interval <= 10) ?
                10 : (detect_time_interval -= 5);
             config_data_display(count, i);
             }
            else if(i % 4 == 1 && i < item_count) {
             sensor_select_state[i / 4] =
                (sensor_select_state[i / 4] == 0) ? 1 : 0;
             if(i > 2 && ((i / 4) + 1) % 3 == 0) {
                sensor_range[i / 4][0] = 600;
                sensor_range[i / 4][1] = 500;
                sensor_range[i / 4][2] = 700;
                }
             else if(((i / 4) + 1) % 3 == 2) {
                sensor_range[i / 4][0] = 90;
                sensor_range[i / 4][1] = 80;
                sensor_range[i / 4][2] = 100;
```

```
    }
  else {
    sensor_range[i / 4][0] = 0;
    sensor_range[i / 4][1] = 0;
    sensor_range[i / 4][2] = 30;
    }
  config_data_display(count, i / 4 + 1);
  }
 else if(i % 4 == 2 && i < item_count) {
  if(sensor_select_state[i / 4] == 1) {
    if(key == KEY_PGUP) {
      if(i > 2 && ((i + 2) / 4) % 3 == 0)
      sensor_range[i / 4][0] = (sensor_range[i / 4][0] > 998) ?
          999 : (++ sensor_range[i / 4][0]);
      else sensor_range[i / 4][0] = (sensor_range[i / 4][0] >
          99) ? 100 : (++ sensor_range[i / 4][0]);
      }
    else sensor_range[i / 4][0] = (sensor_range[i / 4][0] < 1) ?
          0 : (-- sensor_range[i / 4][0]);

    }
  config_data_display(count, i / 4 + 1);
  }
 else if(i % 4 == 3 && i < item_count) {
  if(sensor_select_state[i / 4] == 1) {
    if(key == KEY_PGUP)
      if(i > 2 && ((i + 1) / 4) % 3 == 0)
      sensor_range[i / 4][1] = (sensor_range[i / 4][1] > 998) ?
          999 : (++ sensor_range[i / 4][1]);
      else sensor_range[i / 4][1] = (sensor_range[i / 4][1] >
          99) ? 100 : (++ sensor_range[i / 4][1]);
    else sensor_range[i / 4][1] = (sensor_range[i / 4][1] < 1) ?
          0 : (-- sensor_range[i / 4][1]);
    }
  config_data_display(count, i / 4 + 1);
  }
 else if(i % 4 == 0 && i < item_count) {
  if(sensor_select_state[(i - 1) / 4] == 1) {
```

```
                    if(key == KEY_PGUP)
                      if(i > 2 && ((i + 1) / 4) % 3 == 0)
                          sensor_range[i / 4 - 1][2] =
                            (sensor_range[i / 4 - 1][2] > 9998) ?
                            9999 : (++ sensor_range[i / 4 - 1][2]);
                      else sensor_range[i / 4 - 1][2] =
                            (sensor_range[i / 4 - 1][2] > 998) ?
                            100 : (++ sensor_range[i / 4 - 1][2]);
                    else sensor_range[i / 4 - 1][2] =
                          (sensor_range[i / 4 - 1][2] < 1) ?
                          0 : (-- sensor_range[i / 4 - 1][2]);
                    }
                  config_data_display(count, i / 4);
                  }
                break;
        case KEY_F1    :
                break;
        case ENTER     : done = yes_no_select(50, 270);
                break;
        }
      j ++;
      if(key == ESC || key == SPACE || done == 1 || j >= 30000) break
      }
    if(done) write_system_config();
    return key;
}

int sys_config_init(void)
{
    int x1, y1, x2, y2;
    int key;

    read_system_config();
    x1 = getmaxx() / 2 - 280;
    x2 = getmaxx() / 2 + 280;
    y1 = getmaxy() / 2 - 170;
    y2 = getmaxy() / 2 + 170;
    win_box3d(x1, y1, x2, y2, BLUE, 1);
```

```
    setviewport(x1, y1, x2, y2, 0);
    config_display();
    key = system_config_set();
    setviewport(0, 0, getmaxx(), getmaxy(), 0);
    return key;
}

void quit(void)
{
    da(0);
    logo_screen(1, CYAN);
    last_screen();
    closegraph();
    exit(1);
}

void last_title(void)
{
    color(14, YELLOW);
    textxy(20,  30,  " A Mushroom Automatic System Version 1.0 ")
    textxy(25,  70,  " Copyright  Kim Jahwan & Lee YoungJu ");
    textxy(15,  110, " Graduate Dept. of Electronic & Computer ");
    textxy(50,  130, " Engineering  Univ. of Mokwon ");
    color(RED, 1);
    textxy(110, 160, " Press any key.... ");
}

void last_screen(void)
{
    int x1, y1, x2, y2;

    setviewport(0, 0, getmaxx(), getmaxy(), 0);
    win_box3d(0, 0, getmaxx(), getmaxy(), CYAN, 0);
    x1 = getmaxx() / 2 - 180;
    y1 = getmaxy() / 2 - 90;
    x2 = getmaxx() / 2 + 180;
    y2 = getmaxy() / 2 + 90;
    win_box3d(x1, y1, x2, y2, BLUE, 1);
```

```
    setviewport(x1, y1, x2, y2, 0);
    last_title();
    key_press_wait(50000L);
    setviewport(0, 0, getmaxx(), getmaxy(), 0);
}


/* seg-time.c */
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include "xyplane.h"
#include "seg-time.h"

static int seg_dot_db[11][7] = { {1, 1, 1, 0, 1, 1, 1},        /* 0 */
                {0, 0, 1, 0, 0, 1, 0},        /* 1 */
                {1, 0, 1, 1, 1, 0, 1},        /* 2 */
                {1, 0, 1, 1, 0, 1, 1},        /* 3 */
                {0, 1, 1, 1, 0, 1, 0},        /* 4 */
                {1, 1, 0, 1, 0, 1, 1},        /* 5 */
                {1, 1, 0, 1, 1, 1, 1},        /* 6 */
                {1, 0, 1, 0, 0, 1, 0},        /* 7 */
                {1, 1, 1, 1, 1, 1, 1},        /* 8 */
                {1, 1, 1, 1, 0, 1, 1},        /* 9 */
                {0, 0, 0, 0, 0, 0, 0}
                };

static int seg[7][6] = {{ 2,  1,  9,  1,  1},                  /* up          */
            { 1,  3,  1, 10,  2},              /* up-left      */
            {10,  3, 10, 10,  3},              /* up-right     */
            { 1, 12, 10, 12,  4},              /* middle       */
            { 1, 14,  1, 21,  2},              /* bottom-left  */
            {10, 14, 10, 21,  3},              /* bottom-right */
            { 2, 23, 10, 23,  5}                /* bottom       */
            };

extern int seg_xy[2][2];
```

```
static int seg_time_gap[8] = {15, 15, 20, 18, 19, 18, 18, 17};
static int seg_date_gap[8] = {15, 15, 15, 15, 17, 17, 18, 18};
static int point_db[3][4] = {{ 1,  1,  0,  0}, { 1, 10, 0, 0}, { 1,  1,  0,  0}}
static int point_time_xy[2][2] = {{31,  0}, {68, 20}};
static int point_date_xy[2][2] = {{60, 20}, {100, 20}};

void dis_seg(int seg_x, int seg_y, int seg_dot, int k)
{
    int x1, y1, x2, y2;
    int j, i;

    for(i = 0; i < 7; i ++) {
      if(seg_dot_db[seg_dot][i] == 0) setcolor(DARKGRAY);
      else setcolor(LIGHTRED);
      x1 = seg_x + seg[i][0] + k;
      y1 = seg_y + seg[i][1] ;
      x2 = seg_x + seg[i][2] + k;
      y2 = seg_y + seg[i][3] ;
      for(j = 0; j < 3; j ++) {
        switch(seg[i][4]) {
          case 1 : line(x1 + j, y1 + j, x2 - j, y2 + j);
                   break;
          case 2 : line(x1 + j, y1 + j, x2 + j, y2 - j);
                   break;
          case 3 : line(x1 - j, y1 + j, x2 - j, y2 - j);
                   break;
          case 4 : line(x1 + j / 2, y1 - j / 2, x2 - j / 2, y2 - j / 2);
                   line(x1 + j / 2, y1 + j / 2, x2 - j / 2, y2 + j / 2);
                   break;
          case 5 : line(x1 + j, y1 - j, x2 - j, y2 - j);
                   break;
        }
      }
    }
}

void dis_point(int seg_x, int seg_y, int x, int y, int n, int color)
{
```

```
    int y1, y2;

    setfillstyle(1, color);
    switch(n) {
      case 1 : y1 = y;
           y2 = y;
           bar(seg_x + point_db[0][0] + x, seg_y + point_db[0][1] + y1,
               seg_x + point_db[0][2] + x, seg_y + point_db[0][3] + y1);
           bar(seg_x + point_db[1][0] + x, seg_y + point_db[1][1] + y2,
               seg_x + point_db[1][2] + x, seg_y + point_db[1][3] + y2);
           break;
      case 2 : bar(seg_x + point_db[2][0] + x, seg_y + point_db[2][1] + y,
               seg_x + point_db[2][2] + x, seg_y + point_db[2][3] + y);
           break;
      }
}

void seg_countor(int td, int n, int seg_x, int seg_y, int seg_dot)
{
    int k;

    if(td == 0)k = n * seg_time_gap[n];
    else k = n * seg_date_gap[n];
    dis_seg(seg_x, seg_y, seg_dot, k);
}

void init_seg_time_log(void)
{
    int inc;

    inc = 6;
    seg_countor(0, 0, seg_xy[0][0], seg_xy[0][1], 10);
    seg_countor(0, 1, seg_xy[0][0], seg_xy[0][1], 10);
    dis_point(seg_xy[0][0], seg_xy[0][1], point_time_xy[0][0], point_time_xy[0][1] +
inc, 1, DARKGRAY);
    seg_countor(0, 2, seg_xy[0][0], seg_xy[0][1], 10);
    seg_countor(0, 3, seg_xy[0][0], seg_xy[0][1], 10);
    dis_point(seg_xy[0][0], seg_xy[0][1], point_time_xy[1][0], point_time_xy[1][1], 2,
```

```
LIGHTRED);
    seg_countor(0, 4, seg_xy[0][0], seg_xy[0][1], 10);
    seg_countor(0, 5, seg_xy[0][0], seg_xy[0][1], 10);
    seg_countor(0, 6, seg_xy[0][0], seg_xy[0][1], 10);
    seg_countor(0, 7, seg_xy[0][0], seg_xy[0][1], 10);
}

void seg_inital(int td)
{
    int i;

    for(i = 0; i < 3; i ++) {
      point_db[i][2] = point_db[i][0] + 3;
      point_db[i][3] = point_db[i][1] + 3;
      }
    if(td == 0) init_seg_time_log();
    else init_seg_date_log();
}

void seg_time(int hour, int min, int sec, int ssec)
{
    int inc;

    inc = 6;
    if(ssec / 10 <= 5) dis_point(seg_xy[0][0], seg_xy[0][1], point_time_xy[0][0]
        point_time_xy[0][1] + inc, 1, LIGHTRED);
    else dis_point(seg_xy[0][0], seg_xy[0][1], point_time_xy[0][0],
        point_time_xy[0][1] + inc, 1, DARKGRAY);
    seg_countor(0, 0, seg_xy[0][0], seg_xy[0][1], hour / 10);
    seg_countor(0, 1, seg_xy[0][0], seg_xy[0][1], hour % 10);
    seg_countor(0, 2, seg_xy[0][0], seg_xy[0][1], min  / 10);
    seg_countor(0, 3, seg_xy[0][0], seg_xy[0][1], min  % 10);
    seg_countor(0, 4, seg_xy[0][0], seg_xy[0][1], sec  / 10);
    seg_countor(0, 5, seg_xy[0][0], seg_xy[0][1], sec  % 10);
    seg_countor(0, 6, seg_xy[0][0], seg_xy[0][1], ssec / 10);
    seg_countor(0, 7, seg_xy[0][0], seg_xy[0][1], ssec % 10);
}
```

```
void init_seg_date_log(void)
{
    seg_countor(1, 0, seg_xy[1][0], seg_xy[1][1], 10);
    seg_countor(1, 1, seg_xy[1][0], seg_xy[1][1], 10);
    seg_countor(1, 2, seg_xy[1][0], seg_xy[1][1], 10);
    seg_countor(1, 3, seg_xy[1][0], seg_xy[1][1], 10);
    dis_point(seg_xy[1][0], seg_xy[1][1], point_date_xy[0][0], point_date_xy[0][1], 2
LIGHTRED);
    seg_countor(1, 4, seg_xy[1][0], seg_xy[1][1], 10);
    seg_countor(1, 5, seg_xy[1][0], seg_xy[1][1], 10);
    dis_point(seg_xy[1][0], seg_xy[1][1], point_date_xy[1][0], point_date_xy[1][1], 2
LIGHTRED);
    seg_countor(1, 6, seg_xy[1][0], seg_xy[1][1], 10);
    seg_countor(1, 7, seg_xy[1][0], seg_xy[1][1], 10);
}

void seg_date(int year, int month, int day)
{
    int year_date[4];

    year_date[0] = year / 1000;
    year_date[1] = (year - 1000 * year_date[0]) / 100;
    year_date[2] = (year - 1000 * year_date[0] - 100 * year_date[1]) / 10;
    year_date[3] = year - 1000 * year_date[0] - 100 * year_date[1] - 10 *
year_date[2];
    seg_countor(1, 0, seg_xy[1][0], seg_xy[1][1], year_date[0]);
    seg_countor(1, 1, seg_xy[1][0], seg_xy[1][1], year_date[1]);
    seg_countor(1, 2, seg_xy[1][0], seg_xy[1][1], year_date[2]);
    seg_countor(1, 3, seg_xy[1][0], seg_xy[1][1], year_date[3]);
    seg_countor(1, 4, seg_xy[1][0], seg_xy[1][1], month / 10);
    seg_countor(1, 5, seg_xy[1][0], seg_xy[1][1], month % 10);
    seg_countor(1, 6, seg_xy[1][0], seg_xy[1][1], day / 10);
    seg_countor(1, 7, seg_xy[1][0], seg_xy[1][1], day % 10);
}


/* xyplane.c */
#include <graphics.h>
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <math.h>
#include <dos.h>
#include <dir.h>
#include <alloc.h>
#include "xyplane.h"
#include "seg-time.h"
#include "adlogo.h"
#include "keys.h"
#include "cb.h"

int tday;
static float detect_data[3];
int seg_xy[2][2];
int tc_state[2] = {1, 1};

unsigned datavalue = 0;
float engunits;
float revlevel = (float) CURRENTREVNUM;

void sub_init_xystate_dis(void);

int init_xystate_display[3] = {1, 1, 1};
float chn_init_xy[6][4], chn_current_xy[6][4], chn_xy_range[6][2];
int check_time_interval;


/*****************************************************************************
*
 reading configure file
*****************************************************************************
*/

extern int detect_time_interval;
extern int sensor_range[6][3];
```

```
extern int sensor_select_state[6];

void xy_grid(int n, int x1, int y1, int x_size, int y_width)
{
    setcolor(LIGHTGRAY);
    setlinestyle(DOTTED_LINE, 0, 0);
    if(n == 0) line(x1, y1, x1, y1 - y_width);
    else line(x1, y1, x1 + x_size, y1);
    setlinestyle(SOLID_LINE, 0, 0);
    setcolor(WHITE);
}

void y_text_display(int x1, int x2, int y2)
{
    int i, x11;
    char *y_title[] = {"Temperature", "Humidity(%)", "Co2 (ppm)"}

    setfillstyle(1, DARKGRAY);
    bar(x1 - 35, y2, x2 + 16, y2 + 15);
    bar(x1 - 35, y2, x2 + 16, y2 + 15);
    x11 = getmaxx() / 2 - 50;
    settextstyle(0, 0, 1);
    for(i = 0; i < 3; i ++) {
      if(sensor_select_state[i] == 1) {
         setcolor(10 + i);
         setfillstyle(1, 10 + i);
         bar(x11, y2 + 5 , x11 + 7, y2 + 7);
         setcolor(10 + i);
         textxy(x11 + 12, y2 + 2, y_title[i]);
         x11 = x11 + 17 + strlen(y_title[i]) * 8 + 20;
         }
      }
}

void band_rate(int x1, int y2, int y_width, int x_size, int chn)
{
    int i, j, k = 1, inc = 1;
    char *temp;
```

```
float total_time, hour_gap;
float x_gap, y_ran, y_gap, yy_width;
int hour, min, sec, ssec;

check_time_interval = 0;
total_time = (float)60.0 * 60.0 / (float)detect_time_interval;
i = 1;
while(1) {
  if(total_time * i ++ < x_size) check_time_interval += 1;
  else break;
  }
total_time *= check_time_interval;
x_gap = x_size / total_time;
chn_xy_range[chn][0] = total_time  ;
chn_init_xy[chn][2] = chn_current_xy[chn][2] = (int)x_gap;
hour_gap = 3600.0 / (float)detect_time_interval;
for(j = 0, i = 0; i <= (int) x_size; i += inc) {
  if(i % (int)hour_gap == 0) {
    setcolor(WHITE);
    if(j % inc == 0) {
      get_time(&hour, &min, &sec, &ssec);
      total_time = hour / check_time_interval;
      temp = malloc(sizeof(temp));
      itoa((j + ((int)total_time * check_time_interval)) % 24, temp, 10)
      settextstyle(0, 0, 1);
      if(strlen(temp) <= 2) outtextxy(x1 + inc * i - 5, y2 + 3, temp);
      else outtextxy(x1 + inc * i - 10, y2 + 3, temp);
      free(temp);
      if(i != 0) xy_grid(0, x1 + inc * i, y2 - 2, x_size, y_width - 25);
      }
    j ++;
    }
  else setcolor(LIGHTGRAY);
  if((i % (int)((hour_gap / 60.0) * 10)) == 0) setcolor(WHITE);
  line(x1 + inc * i, y2 + 2, x1 + inc * i, y2 - 2);
  }
y_ran = sensor_range[chn][2] - sensor_range[chn][1];
chn_xy_range[chn][1] = yy_width = y_width - 25;
```

```
    y_gap = y_width / y_ran;
    chn_init_xy[chn][3] = chn_current_xy[chn][3] = y_gap;
    if(y_gap < 2) k = 2;
    for(i = 0;  ; i ++) {
      if((float) i * y_gap > yy_width) break;
      if(i % (5 * k) == 0) {
        setcolor(WHITE);
        temp = malloc(sizeof(temp));
        itoa(i + sensor_range[chn][1], temp, 10);
        outtextxy(x1 - strlen(temp) * 8 - 2, y2 - y_gap * (float)i, temp);
        free(temp);
        if(i != 0) xy_grid(1, x1 + 2, y2 - y_gap * (float)i, x_size, y_width - 25)
        }
      else setcolor(LIGHTGRAY);
      line(x1 - 2, y2 - y_gap * (float)i, x1 + 2, y2 - y_gap * (float)i);
      }
}

void xy_draw(int x1, int x2, int y1, int y2, int x_center)
{
    float y_size, y_width, x_size;
    int hour, min, sec, ssec, t_time;
    int current_lo, current_lo_sec;
    int i, te;
    char *st = "ESC:Quit  F1:Date/Time  F2:Configure";
    char *st1 = "Time(Hour)";

    setcolor(WHITE);
    settextstyle(2, 0, 1);
    outtextxy(x_center - strlen(st) * 4, getmaxy() - 15, st);
    y_size = y2 - y1;
    y_width = y_size / 3.0;
    x_size = x2 - x1;
    y_text_display(x1, x2, y2);
    get_time(&hour, &min, &sec, &ssec);
    check_time_interval = 0;
    te = (float)60.0 * 60.0 / (float)detect_time_interval;
    i = 1;
```

```
    while(1) {
      if(te * i ++ < x_size) check_time_interval += 1;
      else break;
      }
    te = hour / check_time_interval;
    current_lo_sec = (hour - check_time_interval * te) * 60 * 60 + min * 60 + sec;
    for(i = 0; i < 3l; i ++) {
      if(sensor_select_state[i] == 1) {
        current_lo = (current_lo_sec / detect_time_interval) * chn_init_xy[i][2];
        y2 = y1 + y_width;
        setcolor(WHITE);
        settextstyle(0, 0, 1);
        outtextxy(x2 - strlen(st1) * 8, y2 - 10, st1);
        chn_init_xy[i][0] = chn_current_xy[i][0] = current_lo + x1;
        if(init_xystate_display[0] == 1)
            chn_init_xy[i][1] = chn_current_xy[i][1] = y2 - 25;
        else if(init_xystate_display[0] != 1)
            chn_init_xy[i][1] = y2 - 25;
        if(init_xystate_display[1] == 1)
            chn_init_xy[i][1] = chn_current_xy[i][1] = y2 - 25;
        else if(init_xystate_display[2] != 1)
            chn_init_xy[i][1] = y2 - 25;
        line(x1, y2 - 25, x2, y2 - 25);
        line(x1,  y1, x1,  y2 - 25);
        band_rate(x1, y2 - 25, y_width, x_size, i);
        y1 = y2;
        current_lo_sec = (hour - check_time_interval * te) * 60 * 60 + min * 60 +
sec;
        t_time = current_lo_sec / detect_time_interval;
        if(init_xystate_display[0] == 1)
          chn_current_xy[i][0] = chn_init_xy[i][0] + t_time * chn_current_xy[i][2];
        if(init_xystate_display[2] == 1)
          chn_current_xy[i][0] = chn_init_xy[i][0] + t_time * chn_current_xy[i][2];
        }
      }
    if(init_xystate_display[0] == 1) init_xystate_display[0] = 0;
    if(init_xystate_display[2] == 1) init_xystate_display[2] = 0;
}
```

```
void bkground_box(void)
{
    setcolor(WHITE);
    setlinestyle(0, 0, 1);
    rectangle(1, 0, getmaxx() - 0, getmaxy() - 27);
    rectangle(2, 1, getmaxx() - 1, getmaxy() - 28);
    rectangle(3, 2, getmaxx() - 2, getmaxy() - 29);
}

void data_value_display(int chn, float data)
{
    char *temp;
    char *st = "Current Value = ";

    temp = malloc(sizeof(temp));
    gcvt(data, 5, temp);
    strncpy(temp, temp, 7);
    setfillstyle(1, DARKGRAY);
    bar(getmaxx() / 2, chn_init_xy[chn][1] + 14,
        getmaxx() / 2 + 10 * 8, chn_init_xy[chn][1] + 23);
    setcolor(WHITE);
    outtextxy(getmaxx() / 2 - strlen(st) * 8, chn_init_xy[chn][1] + 15, st);
    setcolor(10 + chn);
    outtextxy(getmaxx() / 2, chn_init_xy[chn][1] + 15, temp);
    free(temp);
}

void data_display(int chn, float data)
{
    int x1, y1, x2, y2;

    x1 = chn_current_xy[chn][0];
    y1 = chn_current_xy[chn][1];
    x2 = x1 + chn_current_xy[chn][2];
    y2 = chn_init_xy[chn][1] - ((data - sensor_range[chn][1]) *
(float)chn_current_xy[chn][3]);
    if(y2 > chn_xy_range[chn][1]) y2 = chn_xy_range[chn][1];
    if((data > sensor_range[chn][1]) &&  data < sensor_range[chn][2]) {
```

```
        y2 = chn_init_xy[chn][1] - ((data - sensor_range[chn][1]) *
(float)chn_current_xy[chn][3]);
        line(x1, y1, x2, y2);
        chn_current_xy[chn][1] = y2;
        }
    chn_current_xy[chn][0] = x2;
}

int date_time_display(void)
{
    int hour, min, sec, ssec, year, month, day;

    seg_date_time_cordinate(2);
    get_time(&hour, &min, &sec, &ssec);
    seg_time(hour, min, sec, ssec);
    get_date(&year, &month, &day);
    seg_date(year, month, day);
    return sec;
}

int all_data_display()
{
    int i, state = 1, time;
    int key, done = 1;
    char *filename;
    char st[11];

    filename = (char *)malloc(sizeof(filename));
    file_creat(filename);
    for(i = 0; i < strlen(filename); i ++) st[i] = 0x00;
    for(i = 0; i < strlen(filename); i ++) st[i] = filename[i];
    st[i] = 0x00;

    while(done) {
      time = date_time_display();
      read_system_config();
      if(time % detect_time_interval == 0) {
        if(state == 1) {
```

```
            setcolor(LIGHTRED);
            ad(st);
            for(i = 0; i < 3; i ++) {
              if(sensor_select_state[i] == 1) {
                data_value_display(i, detect_data[i]);
                setcolor(10 + i);
                data_display(i, detect_data[i]);
                if(chn_current_xy[i][0] > 40 + chn_xy_range[i][0] *
                     chn_current_xy[i][2]) {
                     done = 0;
                     key = -1;
                     }
                }
              }
            state = 0;
            }
          }
        else if(time % detect_time_interval != 0) state = 1;
        if(kbhit()) {
          key = get_key();
          if(key == ESC || key == KEY_F1 || key == KEY_F2 || key == KEY_F4)
            done = 0;
          }
        }
    free(filename);
    return key;
}

void shift_xy_plane(void)
{
    setfillstyle(1, BLACK);
    bar(3, 3, getmaxx() - 3, getmaxy() - 30);
    xy_plane();
}

void sub_init_xystate_dis(void)
{
    init_xystate_display[0] = 1;
```

```
    init_xystate_display[1] = 1;
    init_xystate_display[2] = 1;
}

int xy_plane(void)
{
    int key;

    seg_xy[0][0] = getmaxx() - 130;
    seg_xy[0][1] = getmaxy() - 26;
    seg_xy[1][0] = 0;
    seg_xy[1][1] = getmaxy() - 26;
    seg_inital(0);
    seg_inital(1);
    key = KEY_F3;
    while(1) {
      logo_screen(1, DARKGRAY);
      setfillstyle(1, DARKGRAY);
      bar(0, getmaxy() - 26, getmaxx(), getmaxy());
      bkground_box();
      switch(key) {
        case KEY_F3 : xy_draw(40, getmaxx() - 20, 5, getmaxy() - 45, getmaxx()
/ 2);
                key = all_data_display();
                break;
        case KEY_F1 : da(0);
                tc_state[0] = tc_state[1] = 1;
                key = system_date_time_init();
                 if(key == SPACE || key == ESC || key == ENTER) key =
KEY_F3;
                seg_date_time_cordinate(2);
                break;
        case KEY_F2 : da(0);
                      tc_state[0] = tc_state[1] = 1;
                key = sys_config_init();
                 if(key == SPACE || key == ESC || key == ENTER) key =
KEY_F3;
                break;
```

```
        case ESC     : quit();
                break;
        }
      if(key != ESC && key != KEY_F1 && key != KEY_F2) key = KEY_F3
      }
    return key;
}

void set_date(int year, int month, int day)
{
    union REGS r;

    r.h.ah = 0x2b;
    r.x.cx = year;
    r.h.dh = month;
    r.h.dl = day;
    int86(0x21, &r, &r);
}

void get_date(int *year, int *month, int *day)
{
    union REGS r;

    r.h.ah = 0x2a;
    int86(0x21, &r, &r);
    *year = r.x.cx;
    *month = r.h.dh;
    *day = r.h.dl;
}

void set_time(int hour, int min, int sec, int ssec)
{
    union REGS r;

    r.h.ah = 0x2d;
    r.x.bx = hour;
    r.x.bx = r.x.bx << 8;
    r.x.cx = min;
```

```
    r.x.cx = r.x.bx | r.x.cx;
    r.x.bx = sec;
    r.x.bx = r.x.bx << 8;
    r.x.dx = ssec;
    r.x.dx = r.x.bx | r.x.dx;
    int86(0x21, &r, &r);
}

void get_time(int *hour, int *min, int *sec, int *ssec)
{
    union REGS r;

    r.h.ah = 0x2c;
    int86(0x21, &r, &r);
    *hour = r.h.ch;
    *min = r.h.cl;
    *sec = r.h.dh;
    *ssec = r.h.dl;
}

float getdata(int chan)
{
    cbAIn(0, chan, UNI5VOLTS, &datavalue);
    cbToEngUnits(0, UNI5VOLTS, datavalue, &engunits)
    return engunits;
}

void ad_init(void)
{
    cbDeclareRevision(&revlevel);
    cbErrHandling(PRINTALL, STOPALL);
}

void file_creat(char *filename)
{
    int year, month, day;
    long td;
    char *st;
```

```
    get_date(&year, &month, &day);
    td = (long)(year % 100) * 10000 + (long)month * 100 + (long)day
    st = (char *)malloc(sizeof(st));
    ltoa(td, st, 10);
    strcat(st, ".dat");
    strcpy(filename, st);
    free(st);
}

void data_read(void)
{
    int i, j;
    static float temp[3][13];

    for(i = 0; i < 3; i ++) detect_data[i] = 0.0;
    for(i = 0; i < 3; i ++) {
      for(j = 0; j < 13; j ++) temp[i][j] = getdata(i);
      }
    for(i = 0; i < 3; i ++)
      for(j = 3; j < 13; j ++)
        detect_data[i] += temp[i][j];
    for(i = 0; i < 3; i ++)
        detect_data[i] /= 10.0;

    detect_data[0] = detect_data[0] * 100.0 - 1.0;
    detect_data[1] = detect_data[1] / 0.0213;
    detect_data[2] = detect_data[2] * 500.0 - 500.0;
}

void data_write(char *filename)
{
    int year, month, day;
    int hour, min, sec, ssec;
    FILE *f;

    get_date(&year, &month, &day);
    if(tday != day) {
      file_creat(filename);
```

```
        f = fopen(filename, "w");
        fclose(f);
        tday = day;
        }
    get_time(&hour, &min, &sec, &ssec);
    f = fopen(filename, "a");
    fprintf(f, "%d %d %d ", hour, min, sec);
    fprintf(f, " %.2f %.2f %.2f", detect_data[0], detect_data[1], detect_data[2])
    fprintf(f, "\n");
    fclose(f);
}

int file_find(char *filename)
{
    struct ffblk ffblk;

    return findfirst(filename, &ffblk, 0);
}

void da(int chn)
{
    float RevLevel = (float)CURRENTREVNUM;

    cbDeclareRevision(&RevLevel);
    cbErrHandling (PRINTALL, STOPALL);
    cbDOut(0, 1, chn);
}


void ad(char *filename)
{
    int year, month, day;
    int chn;
    FILE *f;

    ad_init();
    if(file_find(filename)) {
      f = fopen(filename, "w");
```

```
        fclose(f);
        }
    get_date(&year, &month, &day);
    tday = day;
    data_read();
    data_write(filename);
    if(tc_state[0])
      if(detect_data[0] < sensor_range[0][0] - 1) tc_state[0] = 0;
    else
      if(detect_data[0] > sensor_range[0][0] + 1) tc_state[0] = 1;
    if(tc_state[1])
      if(detect_data[2] > sensor_range[2][0] + 20) tc_state[1] = 0
    else
      if(detect_data[2] < sensor_range[2][0] - 20) tc_state[1] = 1
    if(tc_state[0] == 0 && tc_state[1] == 0) chn = 3;
    else if(tc_state[0] == 0 && tc_state[1] == 1) chn = 1;
    else if(tc_state[0] == 1 && tc_state[1] == 0) chn = 2;
    else if(tc_state[0] == 1 && tc_state[1] == 1) chn = 0;
    da(chn);
}
```