보행로보트를 위한 관리제어 시스템

°권호열*, 이도남*, 신유식*, 변증남*, 서일홍**, 임준홍***

* 한국과학기술원 전기및전자공학과, ** 한양대학교 전자공학과, *** 한국항공대학 전자공학과

A Supervisory Control System for Walking Robot

H.Y. Kwon*, D.N. Lee*, Y.S. Shin', Z Bien*, I.H. Suh**, J.H. Lim***

* KAIST, ** HanYang Univ., *** Korea Aviation College

요 약

다각 보행로보트를 효율적으로 운용하기 위한 총괄적인 관리제어 시스템이 설계 되었다. 운영 체제인 XINU를 시스템내의 프로세서간의 통신 및 작업 스케쥴링을 효율적으로 하기 위해 채용하였다. 보행로보트의 제어프로그램이 개발되었으며, 1 각의 기구 모형에 대한 실험이 수행되었다.

1. 서 론

일반적으로 산업용 로보트는 잘 정돈된 작업환경하에서 일하는 것으로서 몸체는 고정된 채 팔과 손을 이용하는 것이 대부분이다. 이와는 달리 해저작업, 원자력 발전소, 위험한 토목공사 현장등 작업 환경이 불균형이고 험난한 곳에서 사람대신 작업을 하는 로보트를 위해서는 사람과 같이 두다리 또는 여러개의 다리를 부착한 보행가능한 로보트의 개발이 필수적이다. 이러한 다각 보행로보트는 20여년전부터 미국, 일본등지와 영국, 소련등 유럽권에서 꾸준히 개발되어 왔다. 그러나 기계적 구조의 복잡성과 제어의 어려움 때문에 아직도 기술 수준은 초기단계에 머물러 있는 형편이지만 최근에는 컴퓨터등 관련기술의 급격한 발전에 힘입어 다각보행로보트의 개발은 매우 활발히 진행되고 있다.[Hir,84][Tod,85]

본 논문에서는 보행로보트의 관리제어 시스템의 구조적 설계및 제작된 1 각의 다리를 이용한 보행 실험이 수행되었다. 보행로보트의 관리제어시스템은 동작계획부터 각 다리를 움직이는 동작실행부까지 관리하며 여러 센서들로부터의 궤환신호를 처리하여야 하므로 많은 계산량이 요구되어 다수의 마이크로 프로세서를 사용하여 일을 분담처리할 수 있도록 다중 CPU를 이용한 관리제어시스템이 설계되었다. 이 시스템의 운영체제로 XINU를 채택하여 각각의 프로세서간의 작업조정과 통신을 용이하게하며, 이러한 환경하에서 보행로보트의 제어 프로그램이 수행되도록 하여 관리제어 시스템이 구조화 및 확장성을 갖도록 하였다.

2. 관리제어 시스템의 구성

전형적인 다각 보행 이동체에 있어서 인간과 기계의 상호 작용을 관리 제어적인 구조로 표시하면 그림 1과 같다[Ori, 82][신유식, 87].

인간 조종자는 목표 임무를 보다 고차의 체계로 부터 받으며, 이동체의 현재 상태에 대한 시각적 궤환을 갖고, 전후진, 회전, 좌우이동등의 동작모우드 명령과 이동 방향, 속도를 지정한다. 때로는 조종자가 각각의 다리를 직접 조종하기도 한다. 동작 계획부에서는 하나, 혹은 다수의 컴퓨터에 의해 구현되는 동작계획 알고리즘을 통하여, 이동체의 몸체나 다리의 이동 궤적에 대한 목표치를 결정한다. 이 단계에서의 주된 문제는 첫째로 발을 들거나 놓는 순서를 포함하는 보조패턴을 결정하여야 하며, 둘째로 몸체나 다리의 동작이 지형에 적응하도록 계획되어야 하고, 끝으로 목표로 하는 이동체의 궤적을 얻기 위한 기계적 작동력을 하중평형과 에너지 소비를 최적화하는 조건하에서 제어하여야 한다. 동작 실행부에서는 실제의 관절 각도, 속도및 발의 힘을 목표치와 비교하여, 기계적 작동부를 직접 제어하는 부분으로 서보 제어부를 포함한다.

위에서 살펴본 동작 계획부와 동작 실행부의 임무는 많은 계산량이 실시간 처리되어야 하는 것을 포함하고 있으므로, 실제의 구성에 있어서는 다수의 마이크로프로세서를 사용하여 일을 분담 처리하고, 각각의 프로세서간의 작업조정과 통신을 위한 실시간 운영체제를 채용할 필요가 있다[McG, 84]. 본 연구에서는 다리마다 3개의 자유도를 갖는 4각 보행 로보트에서 50 개이상의 센서들로부터의 정보를 사용하여 12개의 모터를 적절히 제어함으로써 설계된 걸음새를 구현하도록, 이에 필요한 각 다리관절의 동시 제어 및 이들 간의 작업 조정과 통신을 위한 총괄적인 관리기능을 지원하기 위하여 마이크로컴퓨터용의 운영 체제인 XINU를 채용하였다.

3. 지원 운영체제 XINU

XINU는 마이크로 컴퓨터 시스템을 위한 최소한의 운영 체제로서, XINU는 원시 코드가 공개되어 있기 때문에 응용 목적에 따라 수정하기가 용이할 뿐만 아니라, 기본적으로 UNIX 환경과 유사한 기능의 풍부한 System Call Function을 제공하고 있으므로 사용자는 친숙한 C 언어 환경하에서 프로그램을 쉽게 작성할 수 있을 뿐 아니라, Multi-Task에 적합한 Concurrent Programming을 할 수 있고, 통신용 프로토콜이 확립되어 있어서 다중 프로세서 시스템으로의 확장이 용이하다는 것등의 장점을 갖고 있기 때문에, 본 연구에서는 XINU를 보행로보트의 관리제어 시스템을 지원하는 운영 체제로 채용하고 XINU 환경하에서 보행로보트의 제어 프로그램이 설계및 실행되도록하여 관리제어 시스템의 구조화및 확장성을 갖도록 하는 한편, 시스템의 개발기간을 단축하도록 하였다.[Com,84]

운영체제 XINU는 그림 2와 같은 계층구조로 되어 있으며 System Call Function으로 제공되는 기능은 표준 입력및 출력으로서 keyboard로 부터의 문자 입력 기능과 terminal 상에 문자를 표시하는 기능, Multi-Task를 위한 다수의 계산 프로그램의 동시 수행과 이들 사이의 동기화를 위한 프로그램들 사이의 message 전달 기능, Interrupt 처리및 예약된 일정한 시간의 지연을 위한 Timer 기능, Disk와 Memory등의 자원관리및 File 처리 기능, 그리고 다른 시스템과의 통신을 위한 Network기능등 이다.

4. 관리제어의 State별 설계

본 관리제어는 Top-Down 방식의 설계 기법을 이용하여 먼저 원하는 Function을 Identify 하고 나서 그

Function을 수행하기 위해 필요한 더 작은 Function으로 단계적으로 점점 세분화시키면서 전체 관리 제어를 구현하는 방법을 사용한다.[이제혁,88]

보행 로보트 시스템이 일반적으로 갖추어야 하는 기능을 크게 분류해서 State라 정의하였다. 먼저 Power On직후 시스템이 초기화되고 난후, 인간 조종자로부터의 명령을 입력하는 Key를 기다리는 Idle_State, 보행 동작의 Parameter를 바꿀 수 있는 기능을 갖는 Para_State, 보행 동작을 실행하는 기능의 Exec_State, 기계가 오동작을 하거나 고장 났을때 기계의 상태를 체크해 볼 수 있는 기능의 Diag_State, 동작 순서를 명령화하여 이를 편집하는 명령어 파일에 의해 동작 순서를 미리 기억시킬 수 있는 Edit_State 등이 있다. 보행 로보트의 관리제어 시스템의 전 구조를 그림 3에 나타내었다. 구현된 각 State의 구조는 및 기능을 살펴보면 다음과 같다.

4.1 시스템의 초기화

Power On직후에 이루어지는 시스템의 초기화 과정은 물리적 원점(Physical Home)과 논리적 원점(Logical Home)을 찾는 과정으로 이루어져 있다. 물리적 원점은 로보트가 미리 정해진 방향을 따라 이동하여 Limit Sensor, 또는 Motor Encoder의 펄스에 의해 감지될 수 있는 한계 위치에서 정지하는 동작이다. 논리적 원점은 로보트의 운동을 표현하는 좌표의 원점으로서 물리적 원으로부터 일정한 거리에 있으며, 로보트 운동영역의 좌우한계 위치의 중간에 위치하는 것이 보통이다. 로보트가 물리적 원점으로부터 논리적 원점으로 이동한 후, 몸체및 각 발끝의 좌표가 초기화되며, Timer, 각 입출력 port및 RAM, Motor등의 하드웨어와 시스템의 status및 각 변수에 관한 데이타등의 소프트웨어의 초기화가 일어난다.

4.2 Idle_State

시스템웨어이 초기화된 후에 Main Menu를 단말기상에 표시하며, State 천이 조건에 따른 Key 입력에 의하여 Para_, Exec_, Diag_, Edit_State등으로 천이를 일으킨다. 시스템 Reset에 대하여 논리적 원점으로 로보트를 이동시키며, Power Off시에 시스템 상태를 보존하여 차후에 회복할 수 있도록 한다.

4.3 Para_State

이 State는 Machine의 Flexibility를 위한 State 이다. 즉 한가지 동작에만 고정된 보행로보트가 아니라 여러가지 다른 걸음새에 대해서도 Machine이 원하는 기능을 할 수 있게 Parameter를 조정할 수 있다. 여기에서 걸음새의 선택, 발끝의 이동 궤적, 몸체 이동 조건인 속도와 방향및 거리등의 Parameter 들과 시스템의 상태를 나타내는 status 데이타들을 수집하여 단말기의 화면에 표시하고, 이의 수정및 저장을 한다.

Para_State에서 결정된 값에 대하여 Motion Planner라고 하는 프로세스는 걸음새의 알고리즘에 의하여 보행 로보트의 각 관절의 제어 동작 순서를 계산하고, 이 결과는 Exec_State에서 읽도록 Memory에 저장한다.

4.4 Exec_State

Go명령에 의하여 Memory에서 미리 Motion Planner에서 설계해 놓은 각 다리의 제어 신호의 시간에 따른 데이타를 읽어 내어 Servo 제어기로 보낸다. 몸체 이동 동작중에 Hold명령에 의하여 순간적 정지자세를 취하며, Stop명령 또는 미리 지정된 명령의 수행 완료 시점에서서 Idle_State로 복귀한다.

4.5 Diag_State

Diag_State는 Machine이 오동작을 하거나 고장이 났을 경우 고장난 부위의 진단을 용이하게 하기 위해 Man-Machine Interface 기능을 강화시켜 구현한 State 이다. 로보트 각 모듈의 자기진단(Self Diagnosis) 기능을 수행시켜 이상이 있는가를 조사한 후, 각 Sensor의 현재 상태를 단말기의 화면상에 표시한 후 Sensor를 직접 손으로 작동시켜 그 상태 표시의 변화를 보고 각 Sensor의 정상 동작 여부를 확인할 수 있게 하고 또 각 Actuator들을 움직여 봄으로써 기계적인 고장 유무를 체크할 수도 있게 하여 기계의 고장이나 오동작시 그 원인을 빨리 찾아내어 수리 시간을 줄일 수 있게 한다.

또한 긴급한 상황에 대처할 수 있는 기능을 추가하여 예기치 않은 사용자의 잘못, 시스템 일부의 고장 등으로 기계가 정상 동작 범위를 벗어나는 경우에 특정 Actuator의 정상 동작 범위 밖에 설치한 Limit Sensor로부터 Interrupt가 걸리면 그 Actuator및 이에 관련된 다른 Actuator를 멈춘 후 Interrupt가 걸린 Sensor를 찾아내어 적절한 정보 단말기상에 표시한다. 대부분의 경우 Limit Sensor Error는 심각한 경우이므로 기계의 안전을 위해선 기계를 완전히 멈추도록 구성한다.

4.6 Edit_State

Edit_State에서는 로보트의 기본적인 동작에 관한 명령들을 프로그램 언어화 하여 이들을 사용하여 작성된 명령어 화일을 작성하도록 하여 상위수준에서의 로보트 제어가 가능하게하며, 이를 위해 단순한 구조의 편집 기능과 함께 명령의 수행을 위한 인터프리터 기능및 프로그램 디버깅등이 구현될 예정이다.

5. 1 각 보행 실험

구성된 보행 로보트의 관리제어를 위한 하드웨어 시스템은 그림 4 와 같다. Camera를 통하여 입력된 시각 정보는 Chain Coder와 Vision CPU에서 영상의 특징량을 추출하여 Vision Memory에 저장하거나 VME Bus를 통해 Supervisor CPU로 보낸다.[이병일,87] Supervisor CPU는 보행제어 프로그램에 의해 시각 정보를 분석하여 로보트의 이동 경로를 결정하여 Servo CPU로 보내고, Servo CPU는 Motor를 제어하기 위한 신호를 발생시켜 Leg Machine을 구동하게 된다. Display CPU는 영상 처리 결과및 Graphic 데이타를 적절히 Monotor 화면상에 표시하기 위한 것이며, 시스템내의 공통적인 정보들은 Common Memory에 저장한다. 또한 관리제어 프로그램을 개발하기 위한 Host Computer로서 Micro-VAX II가 직렬 통신 인터페이스인 ACIA를 통하여 Supervisory CPU와 연결되어 있다. 시스템내의 각 CPU들은 Motorola의 VME110 module을 사용하였다. 실험을 위해 제작된 2 축의 자유도를 갖는 Pantograph 구조의 1 각 보행기구는 사진 1과 같다.[최병욱,88]

기본적인 보행동작의 확인을 위한 실험으로서 Para_State에서 보행 로보트의 자세 조정을 위한 Jog Mode로 쓰이는 미소 구간 이동명령인 Posture와 함께, 발 끝의 이동 궤적이 3각형, 4각형, 사다리꼴로 주어진 Triangle, Square, Trapezoidal의 Menu가 마련되어 수행되었다. 특히 관리제어 시스템에 필요한 Concurrency의 실험을 위하여, Exec_State에서 보행 동작을 위한 Motor의 구동 Process와 함께 Motor의 현재 위치를 일정시간 간격으로 Position Sensor로부터 읽어서 Console로 출력시키는 기능의 위치감지 Process를 만들어 이러한 2개의 독립된 Process를 동시에 수행시키도록 하였다. 이 과정에 대한 Flow Chart는 그림 5와 같다.

실제적으로 DC Motor의 제어에 있어서 4 각 보행로보트의 몸체 진행속도를 1[m/min], 보폭(Stride)이 25[cm] 이라고 할때, 평탄지형에서 가장 큰 세로안정도를 확보하는 상대위상이 0.5인 물결걸음새및 정적 균형을 유지하기에 최적인 0.75의 더딤율로 보행하는 경우[박성혁,88], Supervisor CPU가 Servo CPU의 역할까지 수행할 때 발 끝이 정사각형의 이동 궤적을 갖는다면 Motor를 제어하는 명령은 약 1[sec]마다 필요로 한다. 이 때 Motor의 현재위치를 100[msec]마다 감지하도록 한다면, Motor의 구동및 현재 위치감지 라는 두가지 Process에 할당된 CPU 사용시간 단위가 XINU의 Context Switching 주기인 2[msec]로 되어 있으므로, 외형상으로 동시에 수행되는 2개의 Concurrent Process가 존재한다. 이러한 기능을 이용하면 보행로보트 의 관리제어에 특히 요구되는 능력인 복잡한 보행의 수행 과정을 독립적인 작은 단위의 Process들로 나누어, 이들을 동시에 병렬 수행시키는 한편 이들 Process간에 적절한 통신기능을 부여함으로써 총괄적인 보행 로보트 시스템의 동작을 관리제어하는 용도및 보행 로보트의 동작 중에 동작 상태를 항상 점검하는 Process를

통한 Man-Machine Interface의 편리성등을 얻을 수 있다.[Don,87]

본 관리제어 시스템의 실험을 위하여 197 KByte의 RAM을 갖는 Supervisor CPU Card 안에 XINU, 보행 관리제어 프로그램, 간단한 걸음새 발생 프로그램 및 서보 회로 제어 프로그램이 함께 들어 있는데, 앞으로 Servo CPU내의 걸음새 및 서보 관련 프로그램이 보강되고 여기에 Vision CPU및 Display CPU의 비젼시스템및 별도의 On-Line 시뮬레이터까지 연결되어 전체 보행 로보트 제어 시스템이 구성되는 경우, 각각의 프로세서마다 XINU Kernel을 갖도록 하고, 프로세서간의 통신은 Common RAM을 이용하는 방식, 즉 Tightly Coupled, Separate Supervisor, Multiprocessor System의 구조를 취하는 것이 효율적일 것으로 전망된다.[Par,87] 한편, XINU는 원래 범용성을 위하여 설계되어 Process간의 독립성 확보에 중점을 두기 때문에, Process간의 스위칭이 일어날때 불필요하게 많은 정보를 바꾸게 되어 Process들의 대기 시간이 불필요하게 길어지 는 문제점이 있다. 따라서 보행 로보트 시스템에서 요구되는 실시간 처리를 위하여 중요한 일부의 프로세스에 대해서는 우선 순위를 높게 배정하거나 Interrupt Level을 작업의 중요도에 따라 달리하여 가능한한 대기 시간을 짧게하므로써 처리속도를 빠르게 하는 방안등의 Process Scheduling 문제가 앞으로 연구되어야 할 과제이다.[Cho,86]

6. 결 론

본 논문에서는 보행로보트를 위한 관리제어 시스템의 개략적인 구조를 제시하고, 이를 지원하기 위한 Multi-Task 운영체제로서 XINU에 대해서 설명을 하였다. 관리제어 시스템은 기본적인 기능에 따라 다수의 State로 나누어 개별 State에 대한 동작을 State Transition의 개념으로 정리하였으며, 이러한 구조하에서 이루어진 System의 Control Flow에 대해 서술하였다. 관리 제어 시스템의 동작을 확인하기 위하여 간단한 보행 패턴들에 대한 실험을 행하였고, XINU가 제공하는 Concurrent Programming의 보행로보트의 제어에 대한 유용성을 검토하였다.

차후로 해결해야 할 과제는 최적 보행 알고리즘을 구현하기 위한 다리 제어 프로그램의 모듈화및 이들 간의 효율적인 통신 방법의 연구, Real-Time을 보장하기 위한 운영체제의 Process Scheduler 기능의 조정과, 4각 보행로보트의 완전한 구현및 시각장치의 결합을 위한 Multi-Processor System으로의 확장등이다.

참고 문헌

[박성혁,87]
박성혁, 황승구, "다각보행로보트의 제어 방법에 관한 연구", 87한국자동제어학술회의, Vol. 1. pp. 69 - 73

[신유식,87]
신유식, 이정수, 권호열, "다각보행로보트의 관리 제어 및 시각센서의 응용", 전기학회지, Vol.36, No.5, 1987.5

[이병일,87]
이병일, "물체 인식을 위한 산업용 비전 시스템의 설계", 한국과학기술원 석사논문, 1987.2

[이제혁,88]
이제혁, "자동 조립 시스템을 위한 진단기능을 갖는 구조적 관리제어에 관한 연구", 한국과학기술원 석사논문, 1988.2

[최병욱,88]
최병욱, "Pantograph 구조의 다리를 갖는 4각 보행로보트의 Graphic Simulator의 개발", 한국과학기술원 석사논문, 1988.2

[Cho,86]
H.K. Choi, "RTOS: A Real-Time Operating-System for Industrial Process Control", MS Thesis, KAIST, 1986.

[Com,84]
Douglas Comer, *Operating System Design: The XINU Approach*, Prentice-Hall, 1984.

[Don,87]
Marc D. Donner, *Real-Time Control of Walking*, Birkhauser, Boston,1987

[Hir,84]
Shigeo Hirose, "A Study on Design and Control of a Quadruped Walking Vehicle", Int. J. of Robotics Reserch., Vol. 3, no. 2, pp. 113-133, 1984.

[McG,84]
McGhee, *Advances in Automation and Robotics: Theory and Application*, Vol.1, pp259-284, JAI press Inc., London England, 1985

[Ori,82]
David E. Orin,"Supervisory Control of a Multilegged Robot", In'tl Jounal of Robotics Research, Vol.1, No.1, Spring 1982.

[Par,87]
K. Park, *Digital System Architecture*, Course Notes, Electrical Science Dept. KAIST, 1987

[Tod,85]
D. J. Todd, *Walking Machins - A Introduction to Legged Robots*, Anchor Press, 1985.

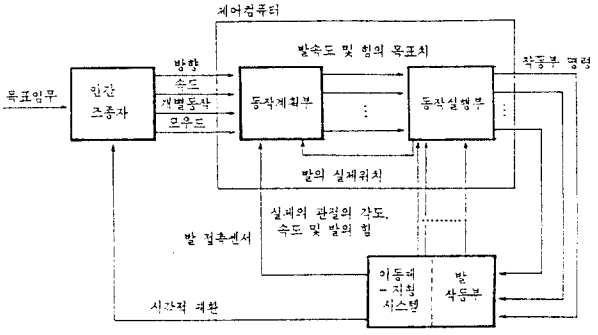


그림 1. 다각 보행로보트의 관리제어 시스템 구조

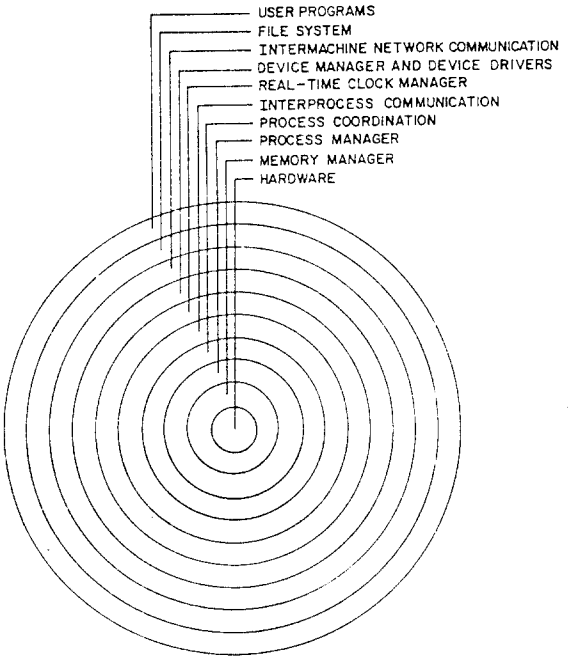


그림 2. XINU의 계층적 구조

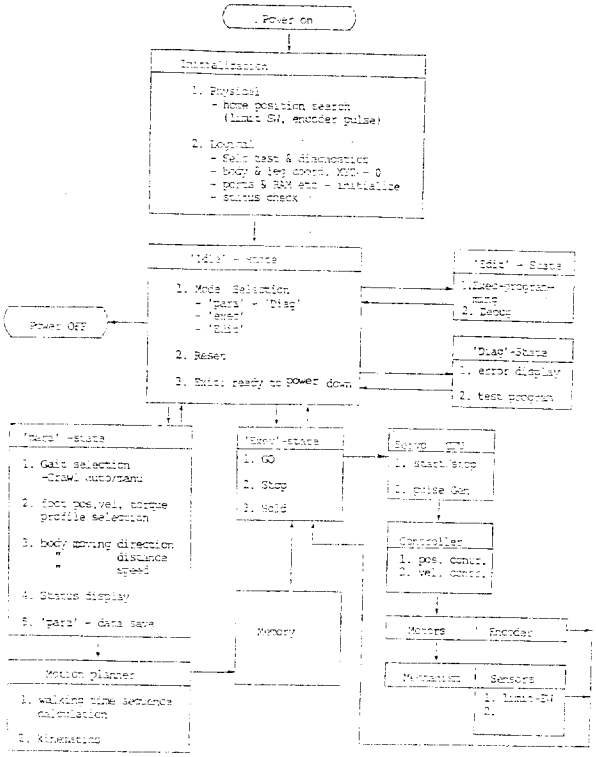


그림 3. 관리제어 시스템의 소프트웨어 구조

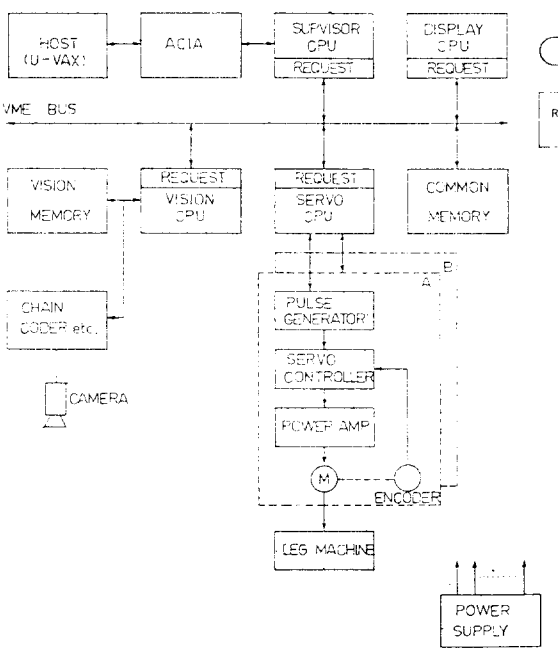


그림 4. 관리제어 시스템의 하드웨어 구조

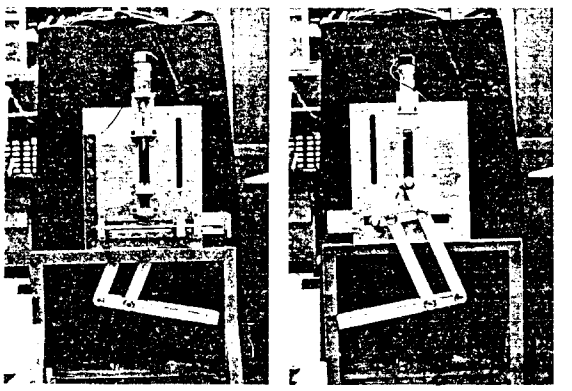
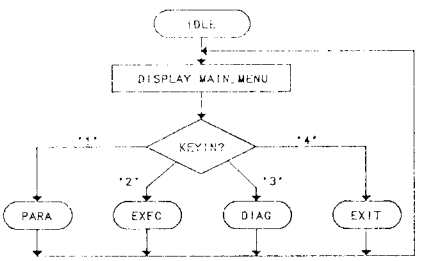


사진 1. Pantograph 형태의 1각 기구 모델



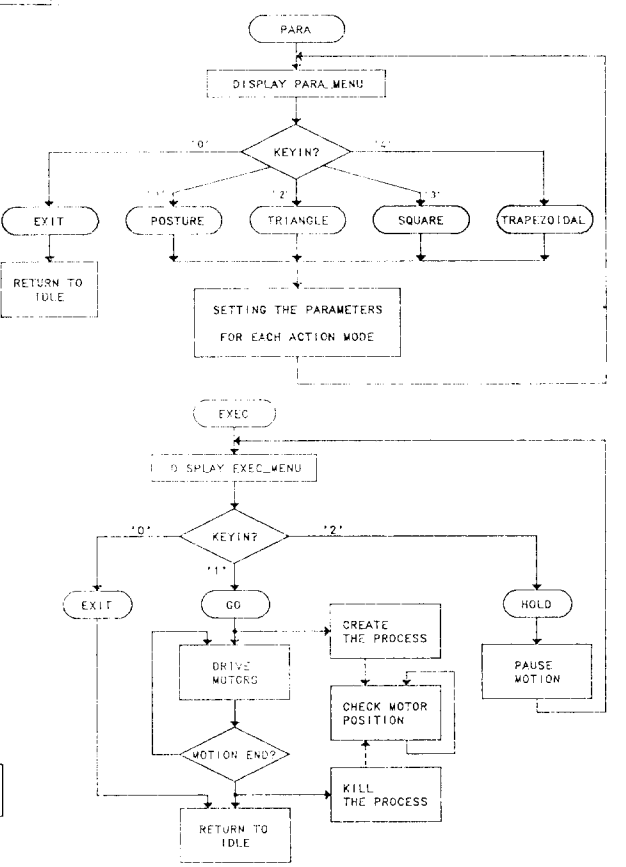


그림 5. 1각 보행실험을 위한 Flow Chart