GOVP1200511068

01-M-DU-29-C-01

소형 무인항공기용 영상장치 Gimbal 시스템 개발

Development of Low Cost Stabilized Gimbal System for Small UAV

(응용연구단계 보고서)

(주)세종무인항공기술

과 학 기 술 부

제 출 문

과학기술부 장관 귀하

본 보고서를 " 소형 무인항공기용 영상장치 Gimbal 시스템 개발" 과제의 응용연구 단계 보고서로 제출합니다.

2003. 10. 31

주관연구기관명 : (주)세종무인항공기술

주관연구책임자 : 김 정 엽

연 구 원 : 박 주 혁
: 홍 성 경

공동연구기관명 : (주)쎄트렉아이

공동연구책임자 : 이 훈 구

공동연구기관명 : 한국항공우주연구원

공동연구책임자 : 안 이 기

요 약 문

Ⅰ. 제 목 : 소형 무인항공기용 영상장치 Gimbal 시스템 개발
(응용연구단계 : 2001. 9.10 ~ 2003. 7. 31)

Ⅱ. 연구개발의 목적 및 필요성

본 연구의 궁극적인 목적은 무인항공기(Unmanned Air Vehicle)용 저가형 소형 김발 시스템을 개발하여 국산화하는 것이다. 여기서 이번 응용연구단계의 목적은 김발 시스템의 형상설계 및 각 구성요소의 개발이다. 또한, 1차 시제품을 제작하여 성능평가를 하는 것이다.

김발 시스템은 영상센서 시선 안정화 및 자동추적 등의 고급기술을 필요로 하는 항공정찰 및 사격 통제 시스템에 필수적인 장비로서, 무인항공기의 소형화에 따라 소형 경량화가 요구되어지고 있다. 현재 김발 시스템은 대부분이 수입에 의존하고 있으며, 또한 매우 고가의 장비이다. 따라서 이 시스템의 국산화시 외화 유출을 최소화 할 수 있을 뿐 아니라 장기적인 측면에서의 수입 대체 효과는 매우 클 것이다. 그리고 광학계, 전자회로, 정밀기계, 계측/제어 및 신뢰도 기술 등의 부가 가치가 큰 기술 확보를 통한 핵심 소요기술의 산업계로의 파급효과가 클 것으로 예상된다. 또한 전문 업체 개발을 통한 국내 타 분야의 장기적인 소요에 대비 할 수 있을 것으로 기대된다.

Ⅲ. 연구개발의 내용 및 범위

본 연구에서 개발하고자 하는 김발의 사양은 안정정확도가 100μrad 이하, 기구물의 중량이 5kg이하, 표적탐지 거리 1km이상, 구동각도 (Azimuth) 0°~360°, (Elevation) 0°~120°, 환경규격 (진동 3G (RMS, random signal, 충격 15G, 11 msec, HALF SINE, 동작온도 -32°C ~ +40°C 이다.

이번 응용연구단계의 연구내용 및 범위는 김발 구조물을 경량, 소형화 하기 위해서 최적설계를 하고 김발의 수학적 동력학 모델링 및 제어 알고리즘을 개발하는 것이다. 또한, 동작 시뮬레이션 및 성능을 예측하고 RFP를 충족하는 영상장치와 기계부품을 선정, 구매하여 시험하는 것이다.

그리하여, 김발 구조물의 설계 및 구성요소의 개발이 완료가 되면, 제어기의 하드웨어 및 소프트웨어를 개발하고 영상장치의 신호 전송부 및 모터 구동회로를 설계하여 제작하는 것이다. 그리고, 조종기 연결부 및 EMI/EMC 케이스를 제작하고 Sensor의 신호 처리를 체크한다. 또한, 김발 상태 모니터 기능과 실시간 고장 진단 기능도 체크한다. 그래서 이 모든 김발 시스템을 구성하여 시험평가 및 해석하는 것이다.

Ⅳ. 연구개발결과

본 연구의 응용연구단계에서 각 분야별 연구개발 결과는 다음과 같다.

첫째, 김발 기구설계 및 제작과 센서/구동기 시스템 구성 측면에서는, 경량 및 소형 김발 구조물을 설계하여 1차 Prototype을 제작하였다. 그리고, 자이로 및 가속도계의 혼합 알고리즘을 개발하였으며, 요구사항에 부합하는 구동기를 선정하여 시험하였다. 또한 센서의 특성 시험을 하였다.

둘째, 제어기 개발 측면에서는, 김발 제어용 소프트웨어 및 하드웨어를 개발하여 제작하였다. 그리고 카메라 제어회로 및 파워 분배기를 제작하였다.

셋째, 시험 및 성능평가 측면에서는, 안정 정확도, 마찰 시상수 및 관성모멘트와 Gyro 모델 변수의 측정방법과 열 환경 시험기법 및 진동 시험기법을 수립하였다.

Ⅴ. 연구개발결과의 활용계획

본 연구개발결과는 총 4년의 연구기간 중 응용연구단계 2년의 결과 산출물이다. 이번 단계에 정립된 김발 시스템에 관한 기본연구를 바탕으로 차기 시험연구단계는 좀더 세분화된 시스템 종합기술로 최종목표인 소형 무인항공기용 영상장치 김발 시스템 완제품을 완성할 계획이다. 개발품은 영상센서 시선 안정화 및 자동추적 등의 고급기술을 필요로 하는 항공정찰 및 사격통제 시스템에 활용됨과 동시에 첨단 정보화의 전략이 필요한 군사작전에 적용되어 비접근 지역의 공간정보, 인공지물 및 목표물 감시등에 적용 할 계획이다. 이와 더불어 사회적인 엔터테인먼트 및 스포츠 중계 등 무궁무진하게 넓어지는 민간분야의 적용 방안도 시대의 진보에 발맞추어 계획하고 있다. 또한 세부적 기술인 제어시스템, 지향 구동기 기술 및 센서/소프트웨어 통합기술들은 산업계(자동화, 로봇, 정밀계측, S/W) 및 방위산업 기술의 핵심 기반 기술로 기여도가 클 것이다.

S U M M A R Y

Development of Low Cost Stabilized Gimbal System for Small UAV

The final object of this study is to develop the light and cheap gimbal system for small UAV(unmanned air vehicle). And the object of 1st stage (2002~2003) is to design the optimization of gimbal structure and make first prototype for test. A gimbal system is needed for an aerial reconnaissance, surveillance and a shooting range control system with the high-class technology of line-of-sight stabilization and auto-tracking. In addition, The UAVs are becoming more small so that those size are miniaturizing.

At present almost of the gimbal systems imported from abroad, and those are expensive equipment. therefore if those would be localized, we could save budget and get the enormous substitute effects of import in the long run. Besides we can get the various techniques which are optics, electronic circuit, precision machine, measurement/control and reliability. Such the technological know-hows of the gimbal system will affect on the other core technical industry. And we expect to prepare for the anticipated needs through the development of professional company.

The final specification of developed gimbal is as follows.

- The line-of-sight stabilization: up to 100μrad
- Weight of gimbal: 5 Kg
- The rang of detection: 1 Km
- Operated angle: (Azimuth) 0°~360°, (Elevation) 0°~120°
- The environmental standard: vibration 3G, impact 15G, temperature(-32°C~+40°C)

Development areas are separated 3 parts in this study. That results are as follows.

1. Machinery design/manufacture
- Design and manufacture a 1st prototype product
- Select and test motor, sensor
- Make a mixture algorism

2. Controller development
- Manufacture of controller circuit, hardware and software
- Make power distributor

3. Test and performance evaluation method

- Make the test methods of stabilization error, friction coefficient, MOI, thermal environment, vibration

This result is 2 year output about total 4 years. From this builded fundamental production we plan to apply for the rest period study to make more detail and complete thing as the state-of-the art gimbal system for small UAV.

CONTENTS

목 차

제 1 장 서론

제 1 절 연구의 목표 및 내용

1. 최종목표

본 과제는 소형 무인기용(Unmanned air vehicle) 영상장치 김발(Gimbal) 시스템 개발이다. 김발 시스템은 영상센서 시선 안정화와 자동추적 등의 고급기술을 필요로 하는 항공정찰 및 사격 통제 시스템에 필수적인 장비이다. 현재 세계적인 무인 항공기의 소형화에 따라 소형경량의 김발 시스템이 요구되고 있다. 이에 따라서 본 과제의 최종목표는 소형 경량의 영상장치 김발의 개발이며 내용은 다음과 같다.

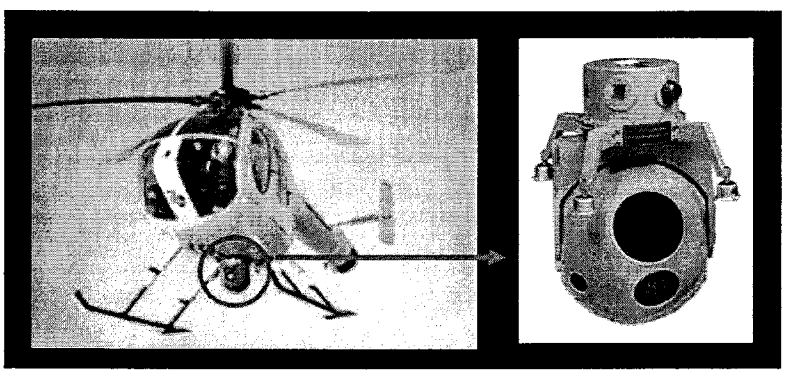


그림 1.1 상용 헬리콥터에 장착된 김발 시스템

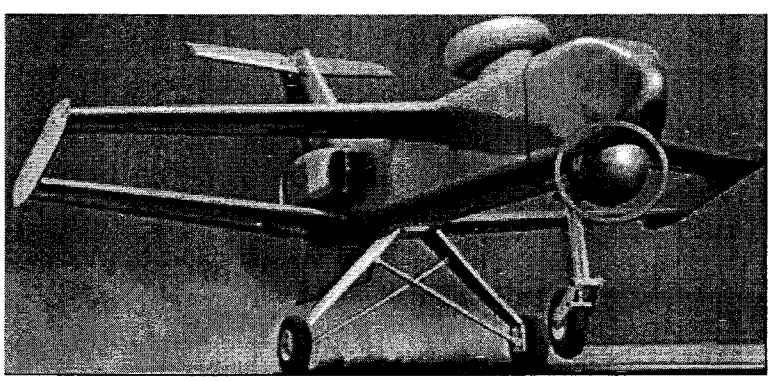


그림 1.2 Unmanned Air Vehicle (TUAV)

가. 시선안정화/자동추적 소형 정밀 김발 시스템 개발

o 소형 경량의 기구설계/해석 및 제작

o 시선안정화 강건 제어기 설계 (PID, Adaptive, Fuzzy Logic Controller)

o 자동추적 알고리즘 설계 (Kalman Filter)

o 영상송수신장치 및 지상조종장치 개발

o 보급형 센서/구동기 응용 기술개발

- Rate Gyro 개발 :
  1,000 deg/s (full scale rate), 100Hz(bandwidth),±0.3 deg/s(max zero offset)
- Position 센서 : potentiometer, resolver, encoder
- 구동기 : brushless torque motor
- 베어링 : angular contact ball bearing

나. 모의시험 및 비행시험 평가

o 비행 모션 테이블을 이용한 제어시스템 시험적 해석

- Rate Gyro 특성 시험 / Moment of Inertia 측정
- 김발 시스템 시험 및 데이터 분석

o 시험비행평가

- 무인항공기 탑재 비행시험
- 무인헬리콥터 탑재 비행시험

o 환경 적응성 시험 : 온/습도, 방수, 진동, 충격 등

다. 정량적인 목표 성능

o 안정정확도(외부 진동에 대하여 시선이 흔들리는 정도) : 100μrad 이하

o 김발 중량 : 5 kg 이하

o 표적탐지거리: 1 km 이상

o 구동각도 : (Azimuth) 0°~ 360°(Elevation) 0°~ 120°

o 환경 규격 만족

- 진동 : 3G (RMS), random signal
- 충격 : 15 G, 11 msec, HALF SINE
- 동작온도 : -32℃ ~ +40℃

2. 응용연구단계 목표 및 내용

총 4년의 연구단계과정 중 본 연구기간2년은 응용연구단계로서 기본적인 김발 개발 기술의 기본을 정립하는 것에 목표를 두고 있다.

표 1.1 연구 목표 및 내용

구 분		연구개발 목표	연구개발내용 및 범위
1차년도 (2001.9.10 ~2002.9.9)	응용연구	Gimbal형상 설계 및 구성 요소개발	· 김발 수학적 동력학 모델링 · 경량/소형을 위한 최적 설계 · 김발 제어 알고리즘 개발 · 동적 시뮬레이션/성능 예측 · RFP를 충족하는 영상장치/기계부품 선정 · 영상장치/기계부품 구매 및 시험
2차년도 (2002.9.10 ~2003.9.9)		1차 시제품 제작 및 성능평가	· 제어기 하드웨어 개발 · 제어기 소프트웨어 개발 · 영상 장치 신호 전송부 설계 및 제작 · 모터 구동 회로 설계 및 제작 · 조종기 연결부 제작 · EMI/EMC Case 제작 · Sensor 신호 처리 · 김발 상태 모니터 기능 · 실시간 고장 진단 기능 · 김발 시험평가 및 해석

제 2 절 연구의 필요성

본 개발 연구의 필요성을 기술적, 군사적, 사회/경제적 측면에서 알아보면 다음과 같다.

1. 기술적 측면

항공정찰 및 사격통제 시스템에 필수인 고성능 영상장치 김발은 영상센서의 시선 안정화 및 자동추적 등의 고급기술과 항공무기체계의 핵심기술인 자세제어, 영상처리 및 관성센서 종합설계기술 등이 필요하다. 이러한 항공기용 영상 장치 김발 시스템은 20 ~ 40만$로 고가의 장비로 무게 또한 12~30Kg로 상당히 무거워서 소형 무인 항공기에 적용하기 많은 어려움이 따른다. 그러므로 저가형 센서를 이용하여 상업적으로 경쟁력이 있는 소형 경량의 영상장치 김발 시스템 개발이 필요하다.

3. 군사적 측면

항공정찰 및 사격통제 시스템을 위한 소형 영상장치의 필요성이 대두되고 있으며, 전투 직전 및 전투 중에 산 너머 적정을 실시간으로 파악하는데 사용되는 초저가형 소모성 소형 무인항공기용 영상장치의 기술개발이 필요하다. 게다가 영상기술과 제어기술을 이용하여 각종 무기체계가 자동화되어 가는 추세이며 이러한 무기체계 개발의 핵심 기반기술인 자세제어, 영상처리 및 관성센서 종합설계기술을 본 연구는 제공할 수 있다.

2. 사회적/경제적 측면

센서, 제어계측, 신호처리 및 전자기술의 발달로 사람이 수행해야 할 부분을 기계 및 전자시스템으로 수행이 가능해지고 있으며 산불관측, 교통방송, 항공촬영 등 관측을 위한 영상장치는 국내에 일정한 수요를 갖고 있으나 수입에 따른 가격이 높다. 더불어 무인 항공기, 유인항공기, 헬리콥터 등 Gimbal 시스템이 갖추어진 영상장치의 예상 수요는 매우 많으나, 고가의 가격이 시장 발전의 주 장애가 되고 있다. 그리고 세계적으로 확대되고 있는 엔터테인먼트 산업에 첨단기술인 영상과 모션의 통합 시뮬레이션 기술을 제공하여 가상현실을 이용한 게임기 등의 고부가가치 제품의 개발로 기술과 상품의 수출이 가능하다.

제 3 절 연구개발 방향

응용연구기간 동안 축적된 기술과 결과물을 이용하여 추후 시험연구단계에서는 보다 세부적이고 종합적인연구를 수행하여 최종 목표 성능을 만족시키려고 한다.

표 2.2 연구개발 방향

구 분		연구개발목표	연구개발내용 및 범위
3차년도 (2003.8.1 ~2004.7.31)	시험개발	성능개선 및 제품설계	· 경량/소형을 위한 최적 설계 · 김발 성능향상에 따른 성능재해석 · 영상장치 신호 수신부/조종기 제작 · 영상처리 및 도시 장치 제작/조립 · 지상 제어 시스템 구축 · EMI/EMC Case 제작
최종년도 (2004.8.1 ~2005.7.31)		제품원형 제작 및 성능 수치도출	· 김발 상태 모니터 제작 · 실시간 고장 진단 기능시험 · 영상 김발 시스템 제작/조립 · 지상 시뮬레이션 시험 . 비행시험 및 성능 종합평가

제 4 절 연구개발 기대성과 및 파급효과

1. 연구개발 기대성과

군작전시 UAV에 탑재하여 정보획득을 통한 전략적 우위에 설 수 있도록 하는 경량 저가의 영상장치 김발(Gimbal) 시스템의 국산화이다.

2. 파급효과

가. 기술적 측면

본 과제 제안의 핵심은 경쟁력 있는 상용기술을 최대한 활용하여 소형 무인항공기용 영상장치 Gimbal 시스템 개발에 있으며 특히 영상센서 시선 안정화 및 자동추적 등의 고급기술을 필요로 하는 항공정찰 및 사격통제 시스템에 필수적인 영상시스템을 구현하기 위한 기반연구와 실용화 연구를 수행함으로써 항공방산기술의 핵심인 제어시스템, 지향 구동기 기술, 및 센서/소프트웨어 통합기술을 확보하는데 있다. 이와 같은 기술 파급(응용) 분야는 벤처 산업형의 핵심기술과 직접 연계되며 산업계(자동화, 로봇, 정밀계측, S/W) 및 방위산업 기술(무인정찰기, 센서 및 구동기)의 핵심 기반 기술의 확보에 기여도가 클 것으로 판단된다.

나. 군사적 측면

비접근 지역의 공간 정보, 인공지물, 토지 활용 현황 등의 탐지, 목표물 감시 및 요격 등을 위한 기술은 군사작전(방위) 계획 수립, 군사 시설 변환 감지 시스템 등에 근간이 된다. 따라서 정밀하고 안정된 영상정보를 얻기 위한 시선안정화 및 자동추적 기술은 군사적 측면에서 필수적인 확보 대상기술이다. 한편 국방 무기체계 개발 사업의 특성상 기술 개발은 막대한 개발비와 첨단기술의 확보가 요구되어 이에 대한 핵심기술의 자체 확보가 필수적이며 해외 기술 종속 탈피를 위해 중장기적인 기술 개발 전략이 필요하다. 또한 갈수록 심화되는 선진 기술 보유국의 기술이전 회피 정책에 능동적인 대응이 요구되며, 따라서 반드시 국가적으로 자체 확보해야할 첨단 민군 겸용 기술을 보유하게 된다는 의미가 크다고 하겠다. 한편 궁극적으로 본 제안과제의 소형 군사용 무인항공기용 영상 장치 실용화시 군 전력 증대에 기여할 것으로 기대된다.

다. 경제적 측면

항공정찰 및 사격통제 시스템에 필수적인 영상장치 Gimbal 시스템은 현재 전량 수입에 의존하고 있으며, 또한 매우 고가인 장비이다. 따라서 이 시스템의 국산화시 외화 유출을 최소화 할 수 있을 뿐 아니라 장기적인 측면에서의 수입 대체 효과

는 매우 클 것으로 예상되며 광학계, 전자회로, 정밀기계, 계측/제어 및 신뢰도 기술 등의 부가 가치가 큰 기술 확보를 통한 핵심 소요 기술의 산업계로의 파급효과가 클 것으로 예상된다. 또한 전문 업체 개발을 통한 국내 타 분야의 장기적인 국내 소요에 대비 할 수 있을 것으로 기대된다.

제 2 장 국내외 기술개발 현황

제 1 절 국내 기술현황

1. 지금까지의 연구개발 실적

- o 삼성톰슨CSF에서 안정정확도 50μrad, 직경 354mm, 길이 510mm, 무게 37Kg의 고가형 정밀 영상감지기(EOTS-31)을 개발한 바 있음
- o 한국항공우주연구원에서는 1993년 Expo 지상관측 무인비행선용 안정정확도 50μrad급 영상안정화장치를 개발한 바 있음
- o 소형 경량(5kg이하)의 저가(2만$)형 안정화 Gimbal시스템은 아직 국내에서 개발된 바가 없으며 소요기술에 대한 기반기술은 산학연에 축적되어 있음
- o 안정화 구동기 기구구조물 및 전자회로 ASIC 등에 대한 소형화 기술개발 필요함
- o 김발 제어기 분야
  - .제어기 Electronics는 각종 분야에 공통성이 높고 현재 국내에서 독자 개발이 가능
  - .소형 위성 시스템의 신호처리부 및 자세제어 시스템 구축 경험 보유
  - .IMT-2000용 위성탑재 transponder 및 지상시스템, 단말기 등의 시험 모델 개발 경험 보유
- o 영상장치 분야
  - . 소형위성 탑재용 카메라 시스템 개발 경험 보유
  - . 현재 말레이시아의 ATSB와 구경 30 cm 정도의 위성탑재용 카메라를 공동개발 중에 있음
  - . SPOT, Ikonos, KOMPSAT 등의 위성영상 처리기술 경험 보유

2. 현(現) 기술상태의 취약성(문제점)

- o 안정화정밀도 100 μrad급의 소형경량 초저가의 Gimbal 시스템 국내 개발능력이 빈약함
- o 센서기술 등의 기반기술이 취약하며 기구물의 초정밀 기계가공 기술의 확보가 시급함

3. 앞으로의 전망

- o 센서, 소재, 항공기술의 발달로 무인항공기 및 항공기 탑재시스템이 소형 경량화를 요구하고 있으며 산불재해, 기상예보, 원양어선 어군탐지, 해양정찰 및 감시등의 다양한 분야의 활용이 예상됨

o 저가제품으로 개발 시 국내외에 시장수요 증대가 예상됨

제 2 절 국외 기술개발 현황

1. 지금까지의 연구개발 실적

o 영국 Cranfield 대학에서는 소형무인기에 탑재된 3개의 영상카메라로 원/중/장거리 영상을 동시에 받아 복합영상을 구성하고 항공기 종축에 대해서만 회전시키면서 영상획득을 하는 시스템을 개발함

o 미국의 Insitu Group에서는 2개의 Rate Gyro를 이용한 저가형 Gimbal시스템을 개발하고 있음

o 영국 BAI에서 무게 6 lb, 크기 1/3"의 Pan/Tilt/Zoom이 가능한 CCD 카메라를 전자적으로 안정화시킨 소형 영상시스템을 개발하고 있음

o 사격통제 시스템으로 중량 30 kg급 초정밀 영상감지 시스템을 운용하고 있으며, 정찰용 소형 영상감지 시스템을 개발 중임

2. 연구개발 또는 사업화 단계

o 지금까지 개발되어 활용되고 있는 것은 영상 시선안정화 기술보다는 원하는 방향으로 추적하는 기술이 대부분임

o 가격이 저가이고 소형경량이며 시선안정화와 추적이 가능한 고성능 제품은 연구개발 단계임

o 초저가형 소형경량 영상장치 김발 시스템은 상업적인 제품이 개발되어 있지 않은 실정임.

제 3 절 기술도입 가능성

- 현재 국내 기반기술 확보는 미비한 실정이나 기술경쟁력 제고와 군수측면에서 매우 중요한 핵심기술이며, 선진국이 기술이전을 회피하고 있는 분야이기 때문에 핵심적인 기술의 도입 가능성이 희박하고 초정밀 제품의 획득에도 어려움이 많다.

제 3 장 연구개발 수행 내용 및 결과

제 1 절 서론

1. 김발 시스템의 정의

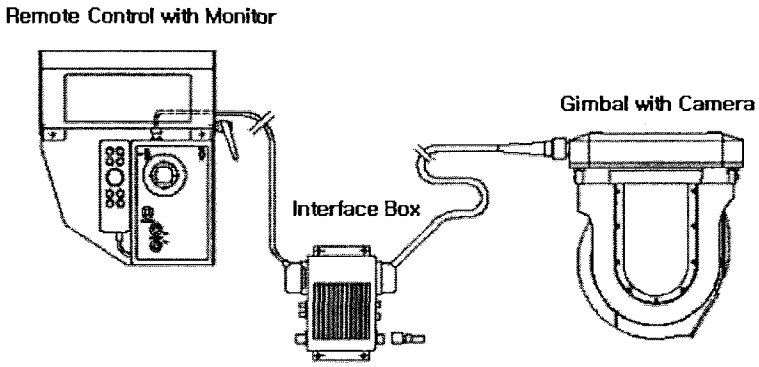


그림 3.1 김발 시스템의 전체 구성도

가. 김발 시스템

영상장치 김발(gimbal) 시스템은 영상센서 시선안정화 및 자동추적 등의 고급기술을 필요로 하는 항공정찰 및 사격통제 시스템에 필수적인 장비이다. 구성은 그림과 같이 영상센서인 카메라를 탑재하여 시선을 안정시키면서 표적방향으로 구동시키는 안정화 구동부인 김발기구와 목표물방향으로 김발 구동부를 움직여 영상을 얻으려고 하는 조정자의 조정장치, 그리고 이 둘의 사이에서 신호전달 및 전원공급을 하는 인터페이스 장치로 크게 3부분으로 나눌 수 있다.

나. 시선안정화(Line of sight stabilization)

광학계의 시선을 진동으로부터 차단시켜서 고정된 영상을 얻는 기술(장치)로 외부진동에 대하여 시선이 흔들리는 정도인 안정정확도 (Stabilization error, Stabilization accuracy)의 단위 1μrad은 다음과 같다.

- Theta = l / R (radian)

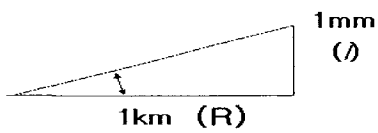


그림 3.2 1마이크로 레디안

다. 광학추적기

광학추적기 기술은 일반적인 김발 시스템에서 한 단계 보다 우위에 있는 기술로 김발로부터 획득된 영상 데이터를 처리하여 이동하는 목표물을 자동으로 추적하는 기술이다.

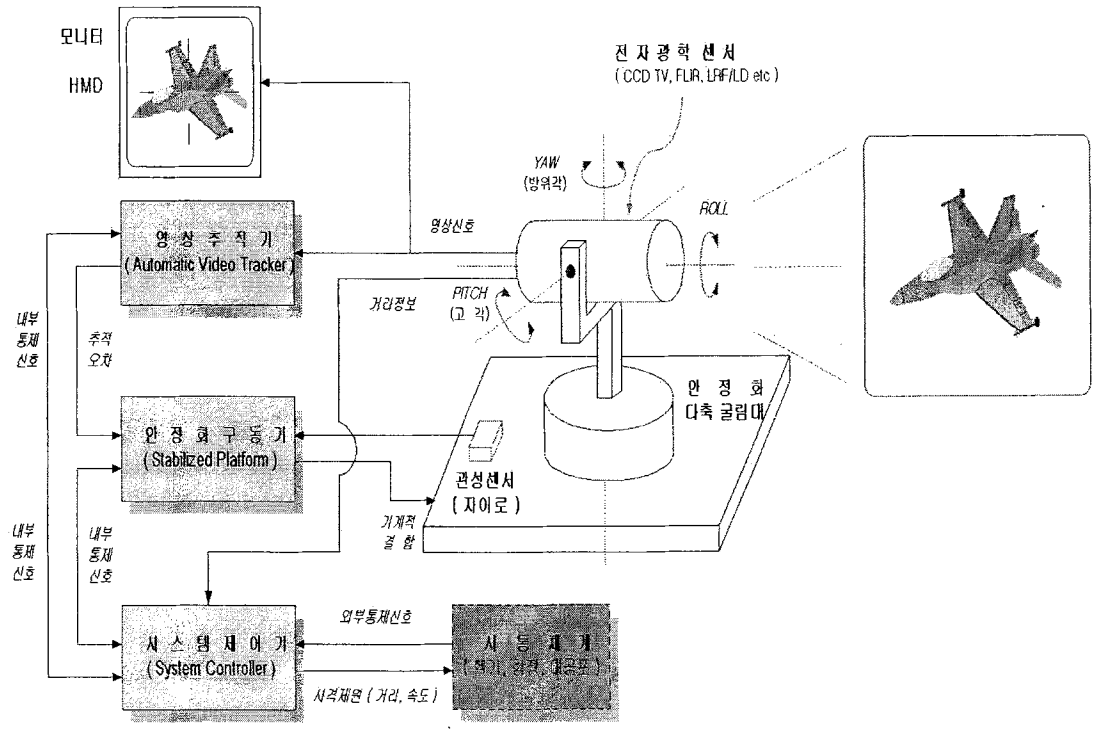


그림 3.3 광학추적기 개념도

제 2 절 김발 기구설계 및 제작

1. 상용 모델 자료 조사 및 분석

소형 무인기용 영상장치 김발 기구설계에 앞서서 상용화된 기존 모델을 조사하고 분석하는 일이 우선 되어야 한다. 본 연구에서는 이러한 조사를 통하여 전반적인 김발 시스템의 구조, 구동방법, 동작성능, 구동기 및 센서, 제작 방법 및 재료, 최신 기술 등 기구설계에 필요한 정보를 획득하였다.

가. 상용 모델의 구조 및 구동 방식

실제 김발 시스템에 관한 기술은 국외 개발업체들이 공개를 꺼려하므로 이에 관한 자료를 찾기가 쉽지는 않았으나 상용 WSCAM사의 김발 14PS모델을 입수하여 직접적인 분해를 통하여 조사를 시행하였다. 아래 그림에 나와 있는 실제 김발모델을 구할 수 있어서 김발 구조물에 관한 기초연구에 많은 도움을 얻을 수 있었다.

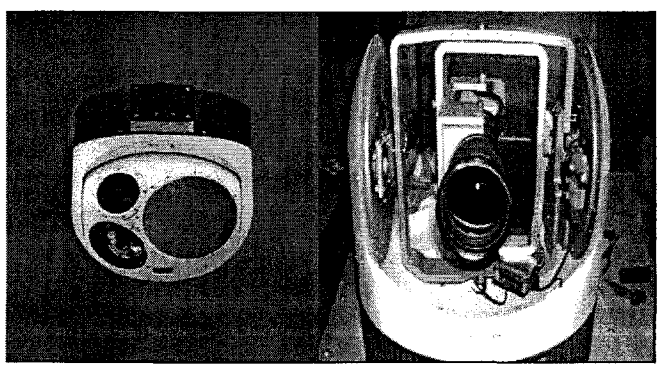


그림 3.4 상용 김발외형과 내부구조

- Specification
  - Sensor 1 - 3.5μm 6 Field-of-View (FOV), Thermal Imager, available in PtSi or InSb detector formats
  - Sensor 2 - Colour Daylight CCD Camera with 955 mm , Long-Range /spotter lens
  - Sensor 3 - Colour Daylight CCD Camera with 10x Zoom lens
  - Sensor 4 - Eye-safe Laser Range Finder (LRF)
  - Gimbal (Dia * H) 36 * 42 cm/34 kg (74 lb) 28 VDC, 10amp (with no ancillary equipment operating)
  - Autotracker : RS232/422 : ARINC 429 : GPS: Video enhancer : Microwave TX : MIL 1553: moving map.

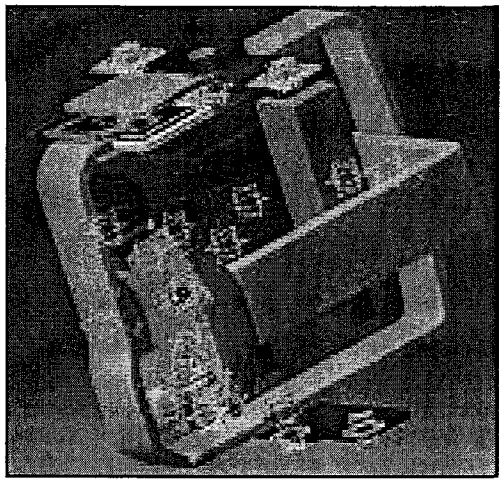


그림 3.5 3D CAD로 구현된 내부구조

- 구조 : 총 4축 김발로 Elevation과 Azimuth로 각각 2축으로 외부(Outer)김발은 위치제어(Position control)이고 내부(Inner)김발은 외란(disturbance)에 대한 안정화(Isolation)로 임무가 분담된 구조이다.
- 재료 : 대부분 알루미늄 계열이 사용되어졌으며 주조 제작 방식으로 본체를 가공하였다. 표면 정밀도와 강도를 높이기 위해서 특수표면처리가 되어 있다.
- 적용된 센서 및 구동기 : 2개의 DC모터로 Outer 김발을 구동하게 되어있다. 구동 각도에 대한 센서는 엔코더를 사용하였고 동력전달 기구로 피니언 축과 베벨기어를 42:1의 비율로 사용하였다. Inner 김발의 경우 정밀 자이로로 외부 외란을 센싱하고 이에 대하여 기계적 시간상수가 작고 제한된 구간 내에서 작동 성능이 우수한 로터리식 보이스 코일 액튜에이터(Rotary Voice Coil Actuator)가 적용되었다.

2. 김발 기구설계 및 제작(1차 시제품)

주로 항공기에 장착되어 사용되는 김발은 항공기 Payload로써 무게가 가볍고 소형이여야 한다. 현 김발 개발업체들의 추세도 고성능이면서도 무게 및 크기를 줄이는 방향으로 개발을 하고 있다. 특히 기구물의 재료로 주로 항공용 알루미늄 소재가 사용되고 몇몇 업체에서는 복합소재를 적용하여 무게를 줄이는 성과를 보고 있다. 게다가 적용되는 센서 및 구동기도 소형이면서 고성능의 제품을 적용하여 기구 배치 공간을 절약하고 무게도 줄이는 방향으로 나가고 있다.

가. 형상설계 (설계도)

김발 기구설계는 상용캐드 프로그램을 통하여 3차원 상에서 이루어졌다. 이번 1차

시제품의 경우 진보된 CAD/CAM/CAE 기술을 적용하여 3D상에서 설계 따른 여러 가지 시험을 통하여 많은 수정과정을 지나 완성되었다. 특히 여러 가지 기능 중 주요하게 사용된 방법은 DMU기능으로 각 부품간의 조립 시 발생할 수 있는 간섭, 충돌, 불균형 등을 찾아내었다. 또한 각 단품의 형상에 적용 재료의 물성치를 부과하여 제작 전 대략적인 전체 무게를 추정할 수 있었으며 적용형상의 무게를 최소화 하기 위한 형상의 최적화 및 구동축을 기준으로 질량의 균형을 맞추었다.

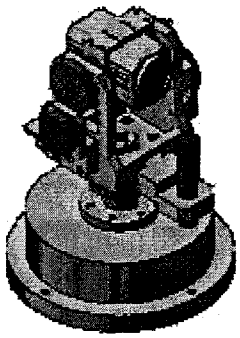


그림 3.6 김발 1차 시제품 3D

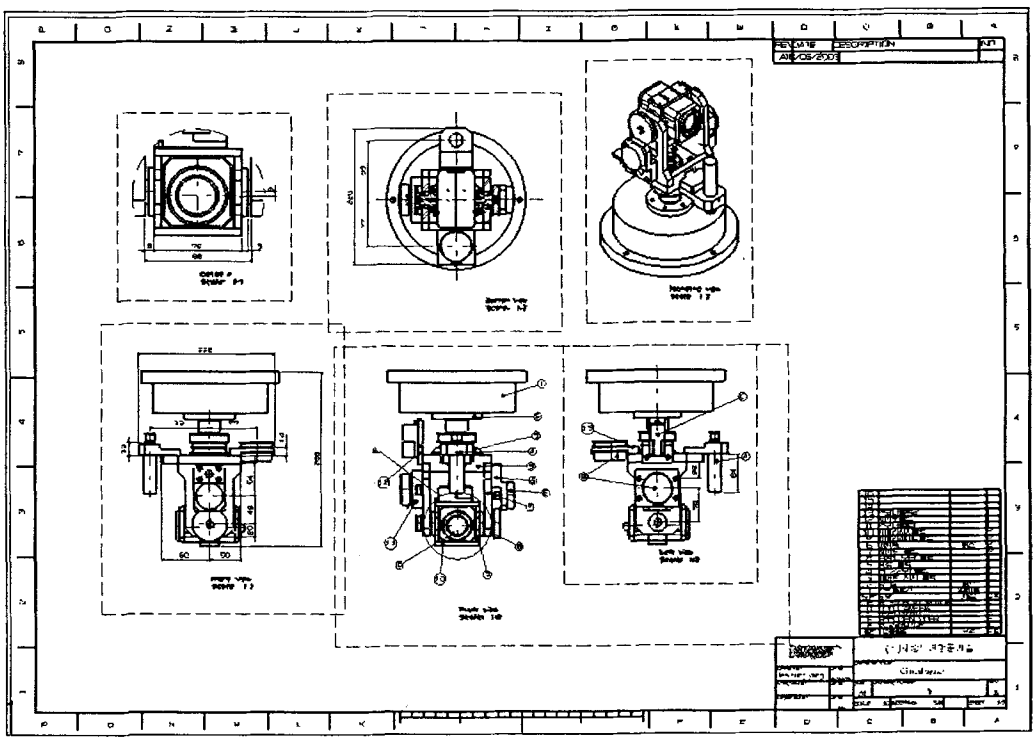


그림 3.7 전체형상 설계도 (2D)

상용프로그램의 DMU기능을 사용하여 설계의 타당성과 적합성을 검증
(space analysis, fitting, crash inspection, inertia moment measure)

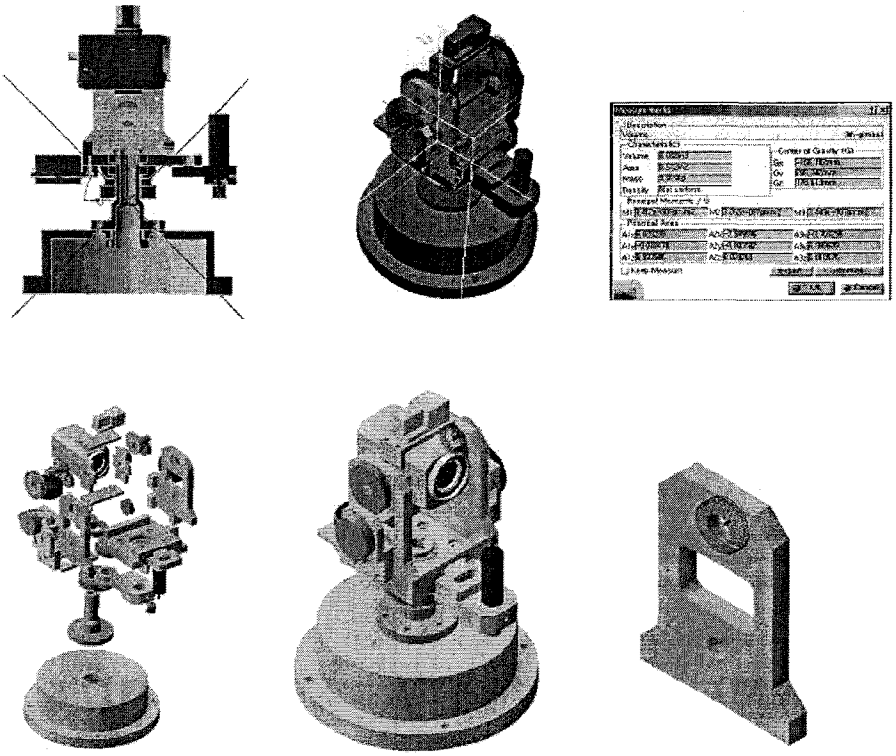


그림 3.8 설계과정 중 DMU 기능 적용

표 3.1 1차 시제품의 Specification

Specification			
Weight	5.24kg (Aluminum 80% Stainless 20%)		제어기별도
Size	Diameter: 229 Height: 288(mm)		
Control axis	2축 (Elevation, Azimuth)		
Inertia moment	Elevation : 5.3kg.cm^2 =>45mNm(토크) Azimuth : 72.4kg.cm^2 => 180mNm(토크)		
Actuator & Sensor			
Azimuth Motor	50.3 mNm(30Watt)	4400 rpm	4:1
	14.2 mNm(20Watt)	9500rpm	76:1
Potentiometer	Resolution 0.007°		
Slip ring	56 ring @ 2amp	250rpm	
Gyro	Threshold/Resolution ≤0.004° /sec		

나. 제작 방법

1차 시제품의 재료는 주로 알루미늄(Aluminum 2001-T3)을 사용하였다. 알루미늄의 선정 배경에는 우선 비중이 적어 경량한 기구제작에 적합하고 또한 가공성이 우수하여 가공비의 절감 효과 및 제작기간을 단축되는 이점이 있다. 그리고 강성 및 정밀도가 요구되는 축은 스테인리스 스틸을 사용하여 최소부피에 큰 힘을 견딜 수 있도록 하였다. 특히 1차 시제품의 제작 의도는 기본 김발 구조물의 형상을 제작해 봄으로써 구조적 특성을 이해하고 선정된 구동기 및 센서 등을 종합적으로 시험해 볼 수 있게 함에 의의를 두고 있었기에 전체적 가공방법은 주조가 아닌 절삭 가공에 의하여 이루어 졌다. 실제 설계도를 기반으로 제작함에 있어 특히 축 정렬의 문제가 중요시 대두되었다. 김발 굴림대의 축은 양쪽에 베어링으로 지지되는데 양쪽의 위치 정밀도가 0.5㎜의 공차로 제작하여 축의 편심을 최대한 줄이려고 노력하였다. 만일 축이 편심 되면 베어링의 마찰 저항이 늘어나고 카메라의 구동에 편향이 생겨 시선의 안정화에 많은 문제가 발생하게 된다.

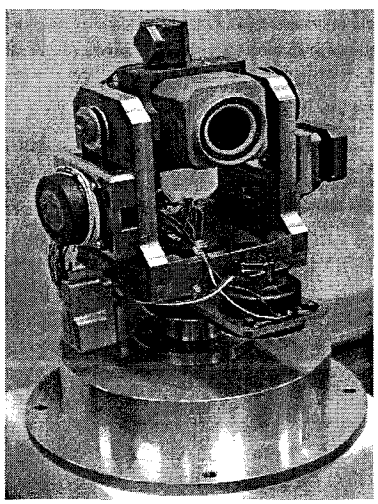


그림 3.9 1차 시제품

제 3 절 센서/구동기 시스템 구성

1. 연구 수행내용

o센서 선정 및 시험

-각속도 Gyro 시험 : CRS03-02, RRS-75

- 센서 요구조건 설정
- 선정대상 자이로(CRS03-02, RRS-75)의 특성(Specification)과 실험을 통한 분석
- CRS03-02와 RRS-75의 성능 비교

-가속도계를 이용한 변위측정 기법 연구

- 수학적 증명을 통한 가속도계의 자세 각 유도
- Matlab 이용한 자세 각 추정 알고리즘 설계
- dSpace Device를 이용한 자세 각 추정 알고리즘 성능 실험

-자이로/가속도계 혼합 알고리즘 시험

- 자이로/가속도계 혼합 알고리즘 설계
- 자이로의 Bias drift 제거 logic 설계 및 실험
- 자이로/가속도계 혼합 알고리즘 검증 실험

o구동기 선정 및 시험

-소요 토오크/정격 출력 계산

-BLDC 1028T 024B 구매

2. 센서 검증시험

가. 신호처리 (dSpace Device, 14 bit A/D)

센서 검증 실험을 하기 위한 신호처리의 방법은 Matlab Simulink와 호환되는 dSpace Device를 사용하여 아래의 그림3.10과 같이 구성하였다.

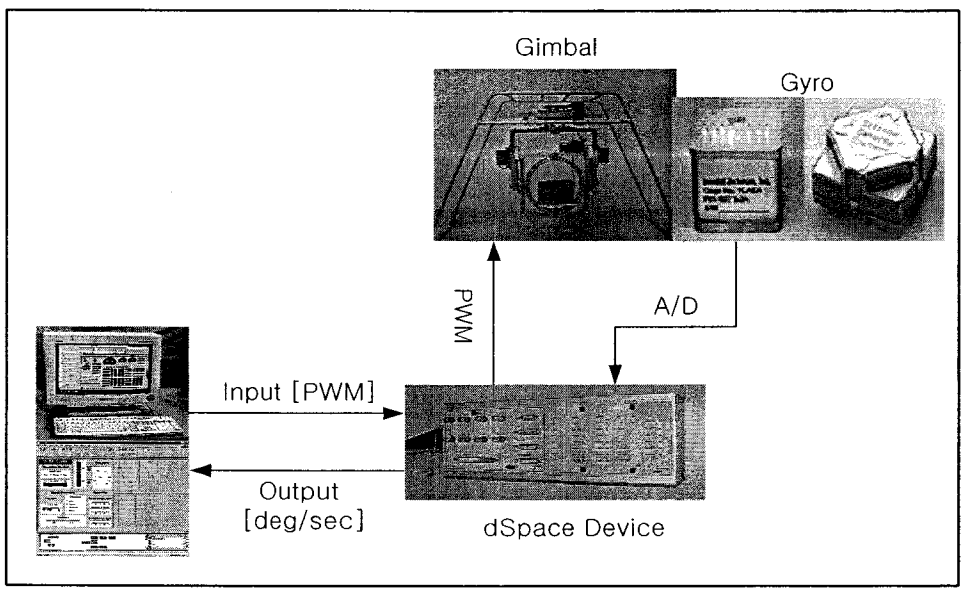


그림 3.10 자이로 특성 검증 실험 구성도

① PC의 Matlab simulink에서 구성된 알고리즘에서 원하는 입력 값을 준다.
② dSpace Device에서 PWM(입력 신호)의 Signal이 실험장치의 모터를 구동시킨다.
③ 실험장치의 움직임에 따라 자이로(ISI RRS-75, CRS03-02)가 반응한다.
④ 자이로의 반응이 다시 dSpace Device A/D converter를 통해 PC의 dSpace Device

프로그램에 구성한 계기판에 응답이 실시간으로 보여 진다.

나. 잡음 및 바이어스 특성 시험

자이로 구매 시 제품 매뉴얼에 나온 특성(Specification)을 확인하기 위한 실험을 아래 그림3.11 과 같이 Simulink로 구성하여 수행하였다.

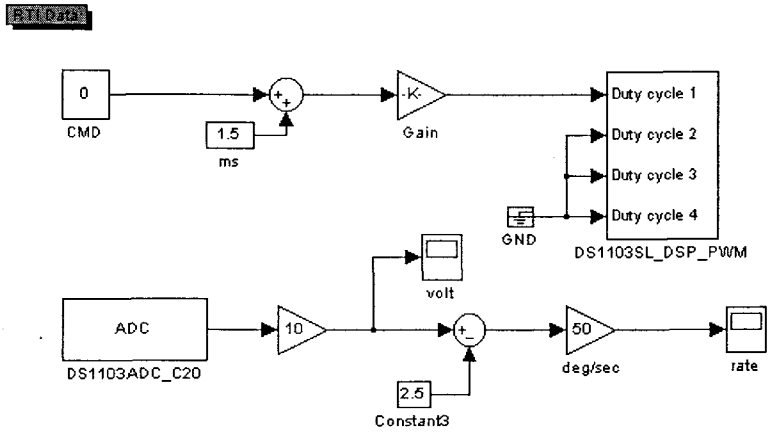
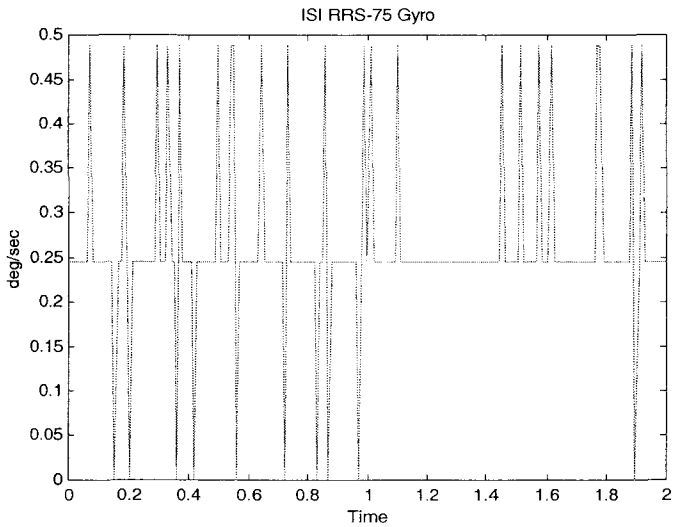
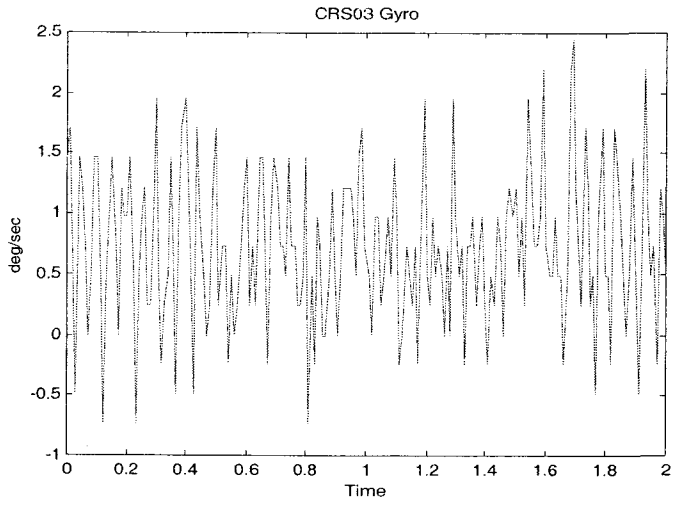


그림 3.11 CRS03-02 Gyro 특성 실험 블록도

아래의 그래프는 실험 결과이다.

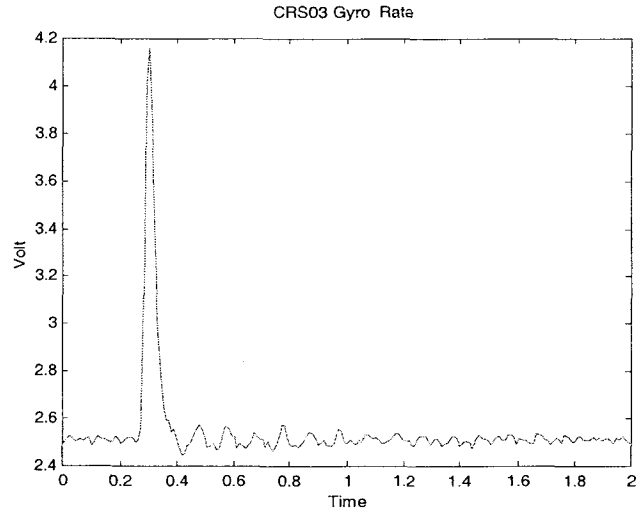


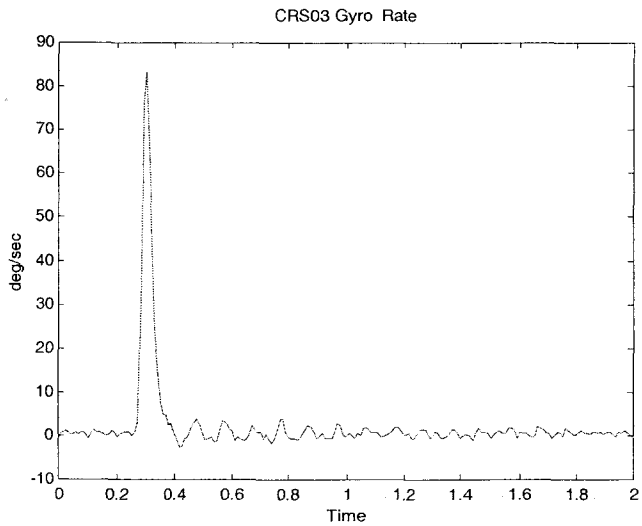


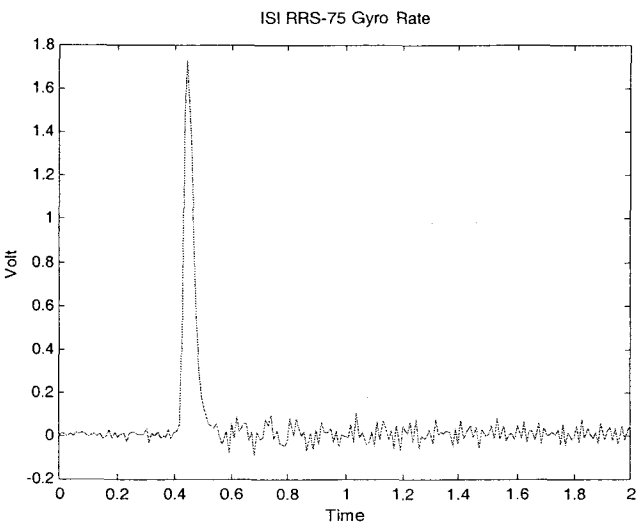
ISI RRS-75 gyro의 Bias (Zero offset)는 0.3deg/sec.Max이고, 실험결과 측정된 값은 약 0.25deg/sec이다. CRS03-02 gyro의 Bias (Zero offset)는 20mV이고 이 값을 deg/sec으로 환산하면 약 1deg/sec이다. 실험 결과 측정된 값은 약 0.65deg/sec 이다. 아래의 실험 그래프는 Scale Factor(Sensitivity)가 매뉴얼의 값과 실험치의 값과 부합되는지의 여부를 보여준다.
먼저, 임의의 입력 값을 주고 위의 그림13 같은 시뮬링크 블록을 사용하여 실험하였다. CRS03-02 gyro의 경우 매뉴얼의 값은 20mV/(deg/sec)이다. 이 자이로의 경우 측정할 수 있는 Rate Range는 0deg/sec에서 100deg/sec이므로 Volt 당 환산된 각속도 값은 50deg/sec이다. 아래의 그래프에서 보는 바와 같이 임의의 입력에 대한 자이로의 출력 Voltage는 약 4.17V이지만, 초기 Zero offset volt가 2.5V이기 때문에 4.17V에서 2.5V를 빼면 약 1.67Volt이다. 이 값을 각속도(deg/sec) 값으로 표현하면 그래프의 값과 비슷한 약 84deg/sec의 값이 나온다.

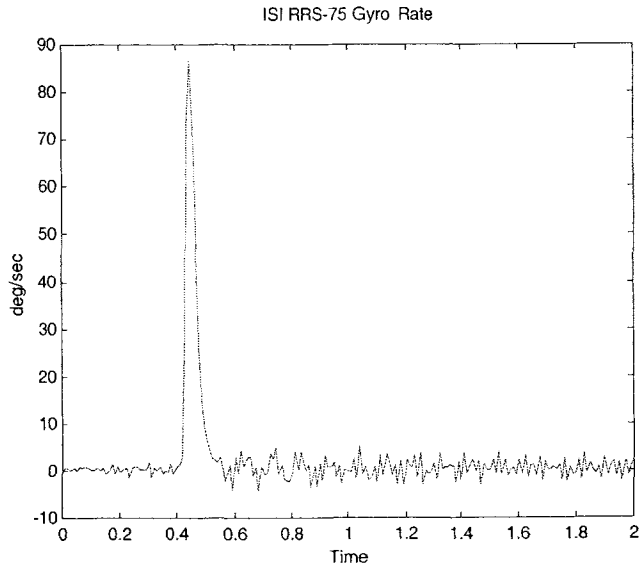
ISI RRS-75 gyro의 경우에서도 이 제품의 Scale Factor(Sensitivity)는 20mV/(deg/sec)이다. 그러나 출력 Volt의 값이 다른 이유는 CRS03-02 gyro의 경우 Supply Voltage가 +5Volt이므로 2.5Volt에서 Zero offset이 된다. 그러나 ISI RRS-75 gyro는 Supply Voltage가 ±5V 이므로 0Volt에서 Zero offset이 된다. 이런 이유로 인해 초기 volt 출력 값의 차이를 보이는 것이다.

ISI RRS-75 gyro의 경우 측정할 수 있는 Rate Range는 CRS03-02 gyro와 같은 0deg/sec에서 100deg/sec이므로 Volt 당 환산된 각속도 값은 50deg/sec이다. 출력 volt의 값은 약 1.7 volt이기에 이 값을 각속도(deg/sec) 값으로 표현하면 그래프의 값과 비슷한 약 85deg/sec의 값이 나온다.


CRS03 Gyro Rate
Volt
Time
4.2
4
3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


CRS03 Gyro Rate
deg/sec
Time
90
80
70
60
50
40
30
20
10
0
-10
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


ISI RRS-75 Gyro Rate
Volt
Time
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2



ISI RRS-75 gyro와 CRS03-02 gyro는 각각의 특성(Specification)이 거의 비슷함은 위에서 언급하였다. 그러나 CRS03-02 gyro가 가격 면에서 훨씬 저가이면서 소형, 경량의 자이로이다. 그러므로 가격대비 성능 면에서 어떠한 차이를 보이는지에 대해 실험을 수행하였다.

실험을 위한 모든 조건은 위에서 언급되어진 내용과 동일한 방법으로 수행하였다. 단, 시뮬링크에서의 입력 값은 주파수 변경에 따른 Sine wave이다.
아래의 그림 3.12는 자이로 비교 실험 시뮬링크 블록도이다.

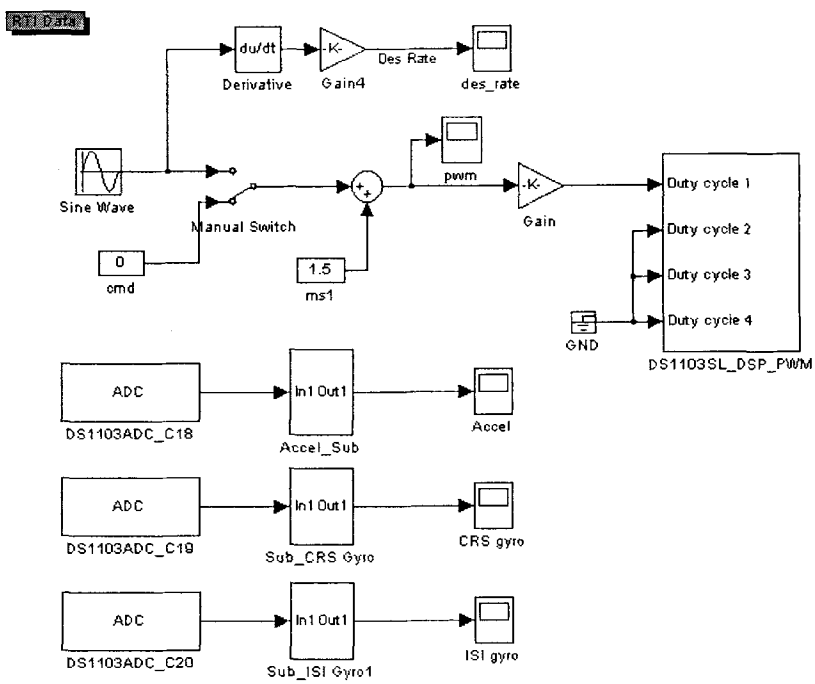


그림 3.12 자이로 비교실험 Simulink Block Diagram

아래의 그림5.는 dSpace Device의 실시간 입, 출력을 보여주는 계기판 구성도이다.

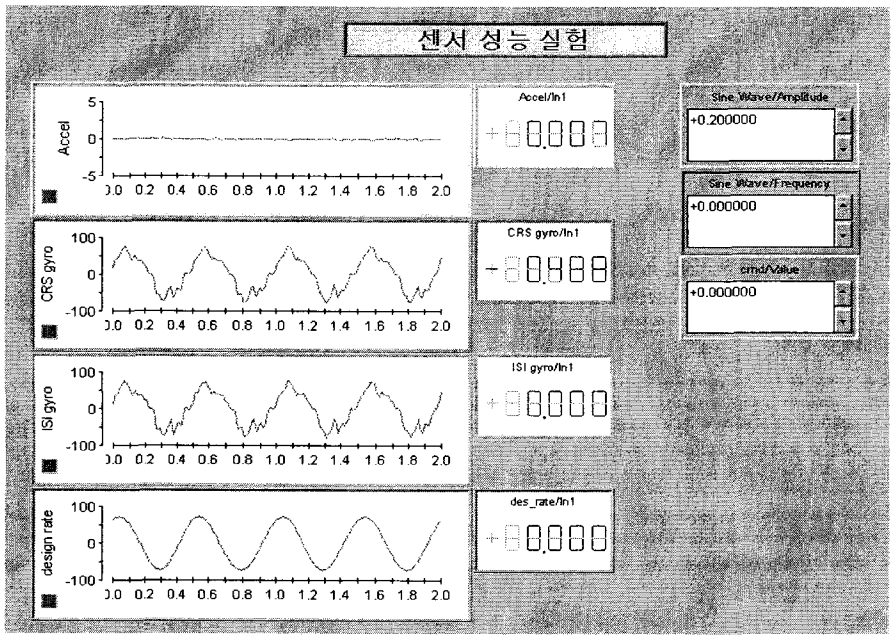
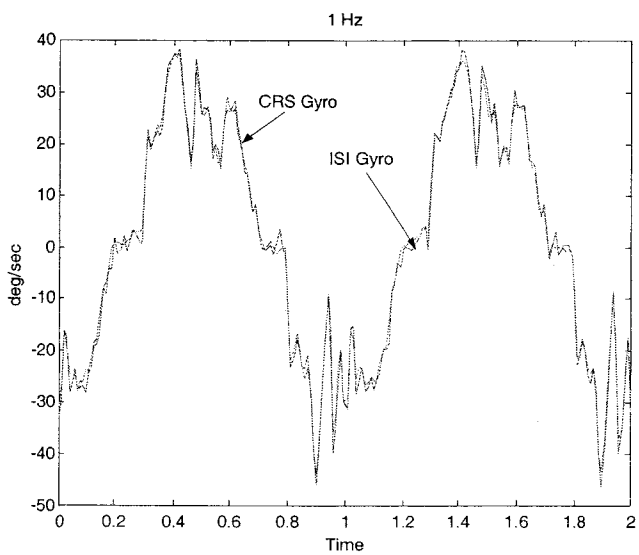
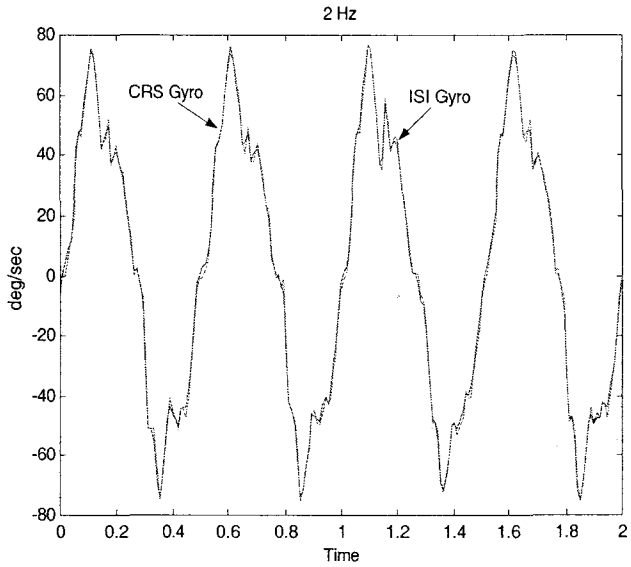
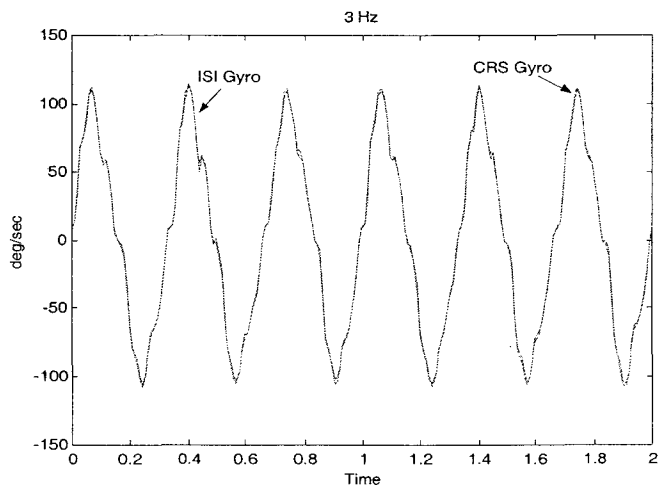


그림 3.13 dSpace Device 계기판 구성도

아래의 그래프는 위의 실험 방법으로 출력된 응답 값이다. 보는 바와 같이 동일한 입력 값에 대한 두 자이로의 응답이 동일함으로 나타났다. 이로서 같은 사양이면서 저가의 소형, 경량의 CRS03-02 gyro의 성능이 고가의 ISI RRS-75 자이로와 견줄 만 하다는 것을 알 수 있다.







센서 오차 모델링 및 보정 기법 연구(보정 logic 부분)
센서 구성 요소의 최적 (자이로/가속도계 혼합 알고리즘 부분)
가속도계를 이용한 변위측정 기법 연구
: Crossbow(USA) CXL-04-LP3

표3.2 Accelerometer CXL-04-LP3 특성

	Crossbow(USA) CXL-04-LP3	비고
Input Range	±4	g (gravity)
Sensitivity	500 ±25	mV/g
Bandwidth	100	Hz
Operatiing Temp. Range	-40 ~ +80	℃
Supply Voltage	5	Volts
Zero g Output	+2.5 ±0.1	Volts
Weight	46	gram

가속도계는 운동체의 가속도를 측정하는 관성 센서(Inertial Sensor)이다. 운동체에 장치된 진자가 가속도의 영향으로 움직이게 되는데 이 때에 진자의 흔들리는 주기는 가속도에 비례하여 나타나게 된다. 이러한 성질을 이용하여 진자의 운동을 확대하여 기록함으로서 계측할 수 있게 된다. 가속도계는 지진의 진동 가속도를 측정할 목적으로 지진계의 일종으로 발달되어서 기계적 진동의 가속도, 항공기 같은 운동체의 출발 시, 운항 중 그리고 정지시의 가속도 측정에 많이 이용된다.
가속도계의 출력에는 운동체의 운동에 의한 가속도와 지구의 중력 가속도, 원심력에 의한 가속도 항, 코리올리 효과에 의한 항이 포함된다. 따라서 계산에 의해 이런 항들을 보정하여 줌으로써 운동체의 운동에 의한 가속도를 얻는다.
또한 가속도계는 아래의 그림3.14와 같이 물체의 기울임이나 언덕의 경사도를 구할 수 있는 Tilt sensor로 사용될 수 있다. 가속도계로 자세 각을 측정할 수 있는 수학적인 유도를 아래에 보여준다.

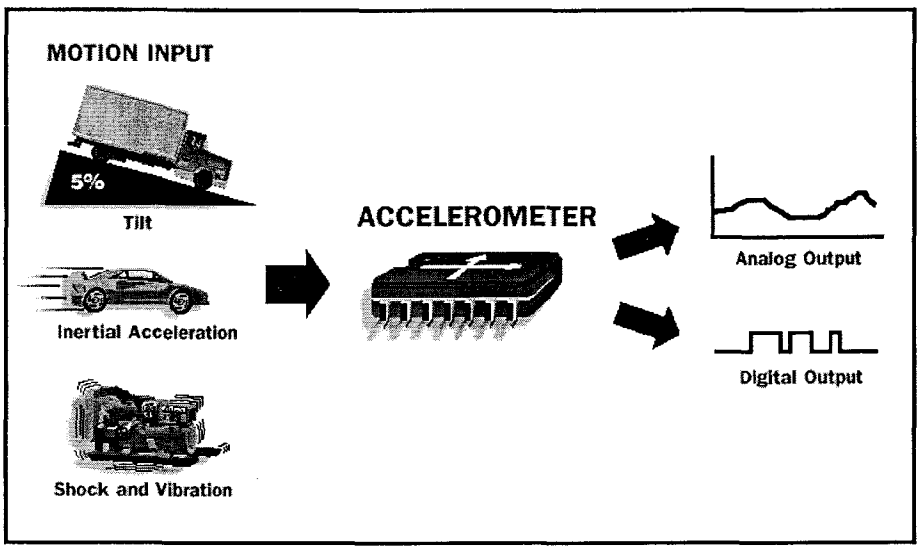


그림 3.14 가속도계 응용 범위

가속도계의 출력 값을 θ 의 값으로 표현하기 위해서는 아래와 같다.

$$\sum \vec{F} = \dot{\vec{G}}$$

여기서 $\vec{G}$는 모멘텀(Momentum) $m\vec{v}$

$$\begin{aligned} &= m\dot{\vec{v}} \\ &= m\frac{d\vec{v}}{dt} \quad (TranferTheorem) \\ &= m(\frac{\delta}{\delta t}\vec{v} + \vec{w}\times\vec{r}) \end{aligned}$$

여기서, $(\vec{v} = u\vec{i} + v\vec{j} + w\vec{k})$동체좌표계 , $(\vec{w} = p\vec{i} + q\vec{j} + r\vec{k})$ 지구고정좌표계 면 $= m(\dot{u}\vec{i} + \dot{v}\vec{j} + \dot{w}\vec{k})$ 로 다시 표현되어진다.

각각의 좌표 i, j, k 벡터로 아래와 같이 표현할 수 있고

$$\begin{vmatrix} i & j & k \\ p & q & r \\ u & v & w \end{vmatrix} = i(qw - rv) + j(pw - ru) + k(pv - qu)$$

X성분, Y 성분, Z 성분에 따라 분류하면, 아래와 같이 각각의 성분에 대해 표현되어진다.

$$\sum F_x = m(\dot{u} + qw - rv)$$

$$\sum F_y = m(\dot{v} + pw - ru)$$

$$\sum F_z = m(\dot{w} + pv - qu)$$

X에 대한 성분인 $\sum F_x = \sum F + mg_x$를 m 으로 나누면,

$$a_x + g_x = \dot{u} + qw - rv$$

$$a_x = \dot{u} + qw - rv - gx$$

$$\begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} = C_3C_2\begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} gsin\theta \\ -gcos\,\theta\sin\phi \\ -gcos\theta\cos\,\phi \end{bmatrix}$$

동체 좌표계 (Body Axis) 각 축의 비행기 가속도로 표현하면, 아래와 같이 표현되어진다.

$$a_x = \dot{u} + qw - rv - gsin\,\theta$$

$$a_y = \dot{v} + pw - ru + gcos\,\theta\sin\,\phi$$

a_x 는 X 축의 비행기 가속도, a_y 는 Y 축의 비행기 가속도이다.

$$\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} 0 & w & -v \\ -w & 0 & u \\ v & -u & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} -r^2 - q^2 & pq - \dot{r} & pr + \dot{q} \\ pq + \dot{r} & -p^2 - r^2 & rq - \dot{q} \\ pr - \dot{q} & rq + \dot{p} & -q^2 - p^2 \end{bmatrix} \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} + g \begin{bmatrix} \sin\theta \\ -\cos\theta \sin\phi \\ -\cos\theta \cos\phi \end{bmatrix}$$

비행자세 각(deg) ϕ 와 θ 는 비행기 가속도에 의해서 얻어질 수 있다. 비행기가 등속도 운동, 또는 정지상태라 가정한다면,

$g \begin{bmatrix} \sin\theta \\ -\cos\theta \sin\phi \\ -\cos\theta \cos\phi \end{bmatrix}$ 을 제외한 다른 부분들은 제거된다.

따라서 가속도의 X축 방향에 대한 θ을 구하면,

$\theta = \sin^{-1} f_x$ 의 형태로 된다.

위에서 최종 유도된 공식을 Matlab을 사용하여 알고리즘을 설계하면 아래 그림3.15 와 같다.

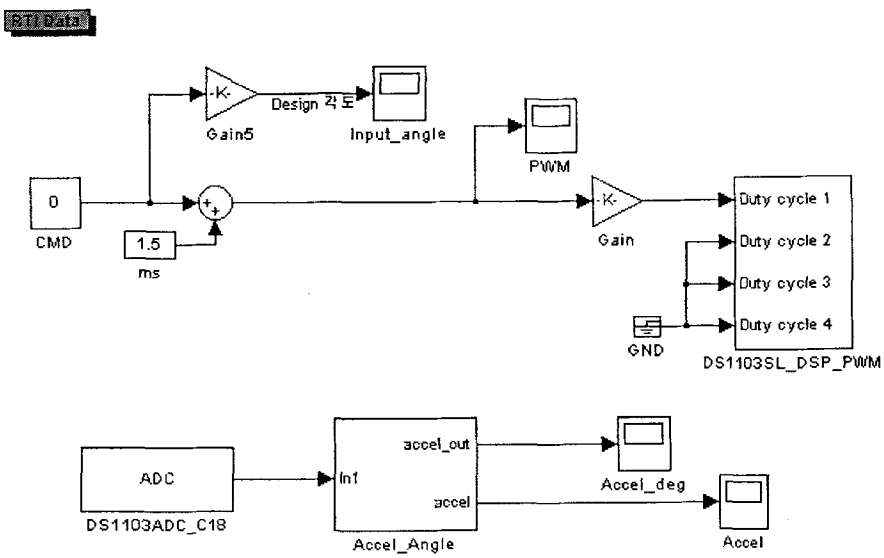


그림 3.15 가속도계 이용 변위측정 알고리즘 블록도

모터 구동 입력을 PWM (Pulse Width Modulation)으로 주어 움직인 속도를 DS1103ADC 블록에서 읽은 다음, 각도 변환 Accel_Angle Sub 블록을 통해 최종적으로 각도의 출력을 얻는다.

여기서, 각도 변환 Accel_Angle Sub 블록은 아래 그림3.16과 같다.

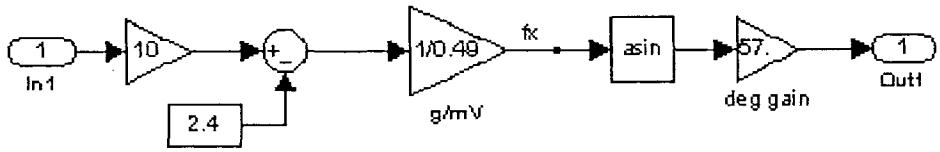


그림 3.16 각도 변환 Accel_Angle Block

이 제품(CXL-04-LP3)이 움직이지 않을 경우의 출력 값(Zero g Output)은 2.4Volt가 나온다. 따라서 0 Volt에 맞추기 위하여 위의 그림3.16에서 보듯이 출력 값에서 2.4Volt를 빼주고, 이 값을 중력(G)의 단위로 환산하는 Sensitivity (g/mv)는 1/0.494이다. 이 신호를 위에서 구한 공식 $\theta = \sin^{-1} f_x$에 근거하여 블록을 형성한다.

아래의 그림3.17은 가속도계 이용 변위측정 실험 구성도이다. 가속도계 이용 변위측정 알고리즘 시뮬링크를 형성하여 입력 값을 주면 PC를 통해 dSpace 장비를 통해 실험 장치를 구동하여 그 움직인 값을 가속도계 센서가 측정, 다시 dSpace 장비를 통해 PC에 구성한 계기판을 통해 그 최종 각도 응답이 실시간으로 그려지게 된다.

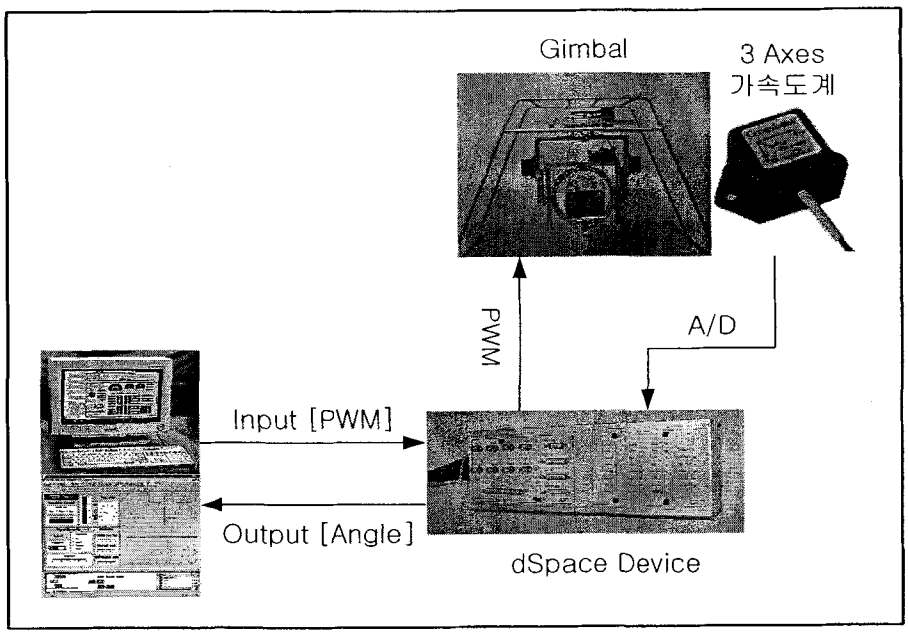
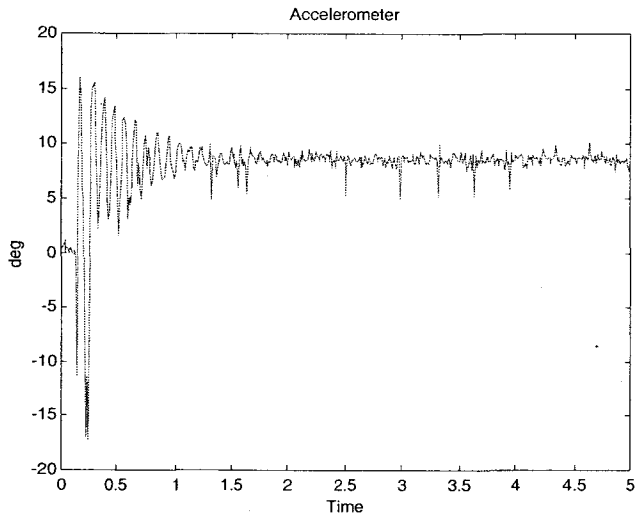


그림 3.17 가속도계 이용 변위측정 실험 구성도

위의 모든 과정을 통해 얻은 값은 아래의 그래프에 보여준다.



다. 자이로/가속도계 혼합 알고리즘 연구

현 개발하는 소형 영상 장치 김발은 무게와 가격면에서 경량과 저가의 개발을 목적으로 하고 있기에 무인항공기 등에 사용되어지는 AHRS 센서를 사용하지 않고 이 센서의 기능을 대신 수행할 자이로/가속도계 혼합 알고리즘을 연구한다.

아래의 그림3.18은 자이로/가속도계 혼합 알고리즘 블록도이다.

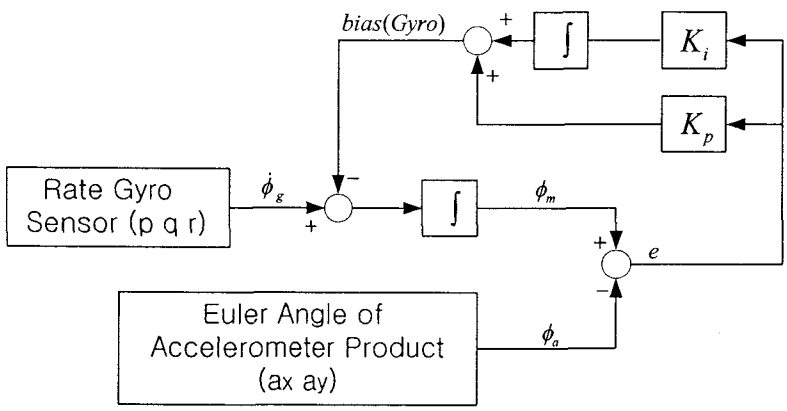


그림3.18 자이로/가속도계 혼합 알고리즘 Block Diagram

이 블록도는 자이로의 각속도 입력과 가속도계의 입력, 그리고 PI controller로 구성된다.

각속도(deg/sec)의 값을 적분하여 그 값을 위에서 언급된 가속도계를 이용한 각도 추출 기법을 이용하여 출력된 가속도계 각도 값과 비교하여 PI controller의 게인

값에 의해 제어하고 그 값을 다시 자이로의 각속도 값과 비교한다. 여기서 PI 게인은 $K_p = 2\zeta\omega_n$, $K_i = \omega_n^2$이므로 두 게인 값은 ω 라는 Cutoff Frequency에 의해 결정된다. 이 값은 Trial & Error의 방법으로 그 값을 0.5로 정한다. 그러나 자이로는 항상 Bias drift를 갖고 있기 때문에 이를 반드시 보정해 줄 필요가 있다.
아래의 그림3.19는 Bias drift의 값을 보정 해 줄 logic(Logic) 이다.
이 logic(Logic)은 Switch에 입력해 놓은 시간만큼 자이로의 출력 값이 저장되어, 저장된 자이로 출력 값의 평균을 구해 그 값만큼 계속 해서 빼준다.
이 로직을 증명하기 위해 자이로 출력 대신 랜덤한 노이즈를 넣어 로직의 최종 출력 값을 본다.
그 결과는 아래의 그래프에 나타나 있다.

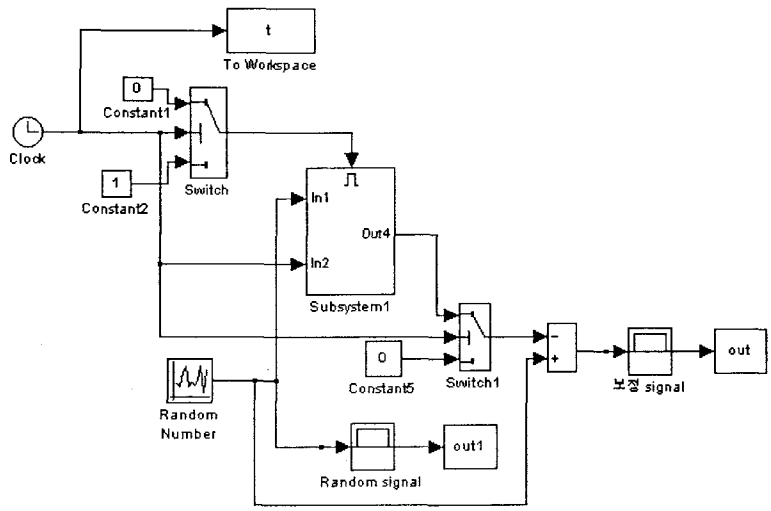
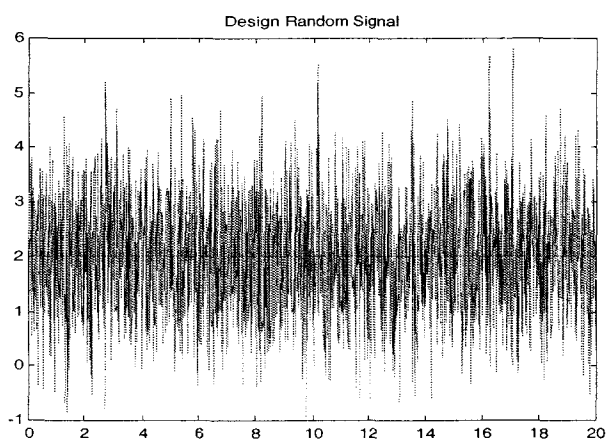
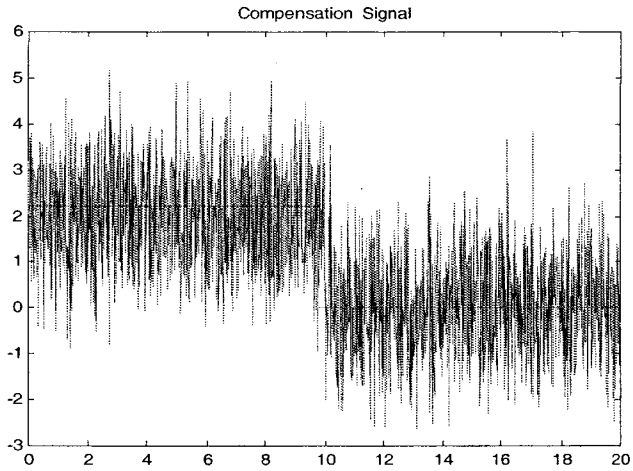


그림 3.19 바이어스 보정 logic





보는 바와 같이 입력 신호는 약 2 정도의 값을 가진다. 그러다 로직에서 정한 시간이 되면 그 평균값을 빼주게 된다. 그 결과 거의 '0'에 가까이 보정됨을 볼 수 있다.

이 로직을 실제 구성한 자이로/가속도계 혼합 알고리즘에 연결하여 자이로의 Bias drift의 값이 보정되는지 알아보자.

실험을 위한 루프는 아래의 그림3.20에 나타나 있다.

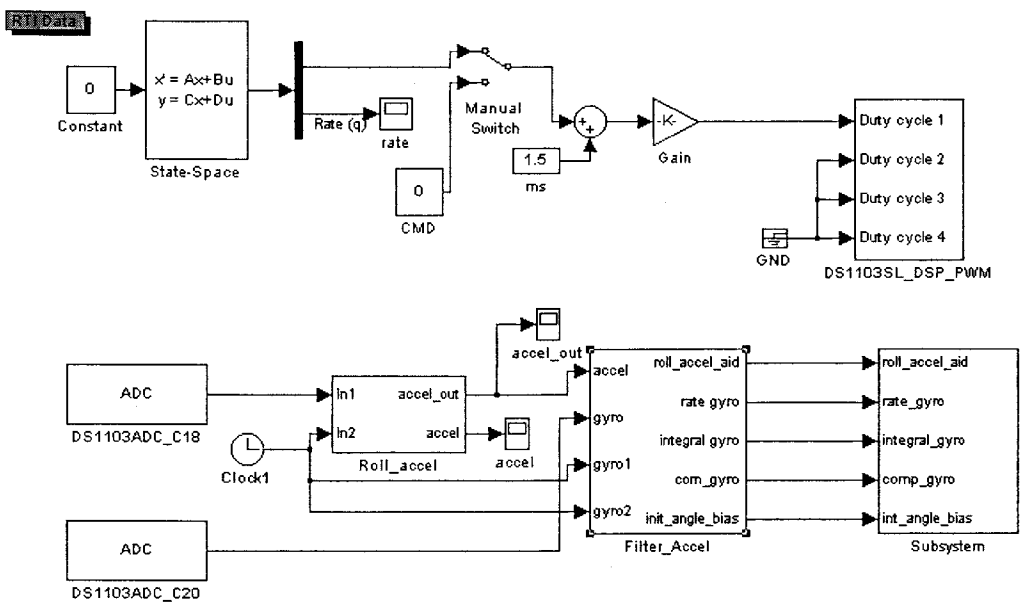
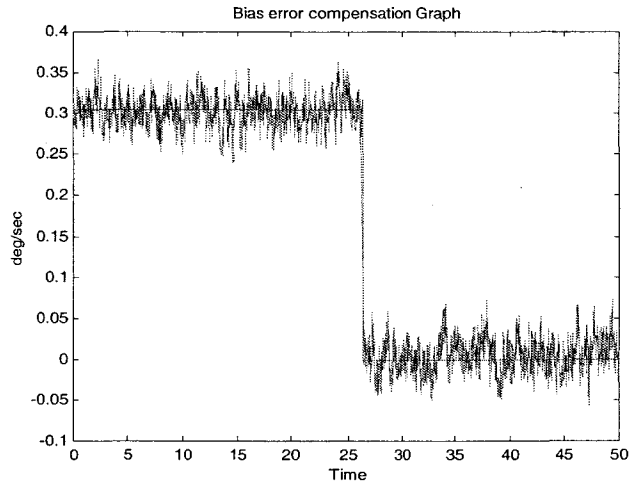


그림3.20 자이로/가속도계 혼합 알고리즘 Simulink

실험기구에 아무런 입력 값을 주지 않고, 자이로를 수평으로 놓고 그 값을 보았다. 아래의 그래프는 그 결과 값이다. 보는 바와 같이 0.3정도의 Bias drift를 갖고 있다가 로직에 입력해 놓은 25초 후에 그 값이 거의 0으로 변함을 볼 수 있다.



자이로 Bias 보정 로직이 위의 실험들을 통해 검증이 되었다. 이 로직을 포함한 자이로/가속도계 혼합 알고리즘 검증 실험을 아래와 같은 방법으로 수행한다.

물리적인 계의 알고 있는 SW Model 중에서 Pendulum model (Reference model)에 대한 입력 값을 자이로/가속도계 혼합 알고리즘에 적용하고 모델 출력과 혼합 알고리즘의 dSpace 출력 값과 비교 검증한다.

아래의 그림3.21은 자이로/가속도계 혼합 알고리즘 검증 실험 블록도이다.

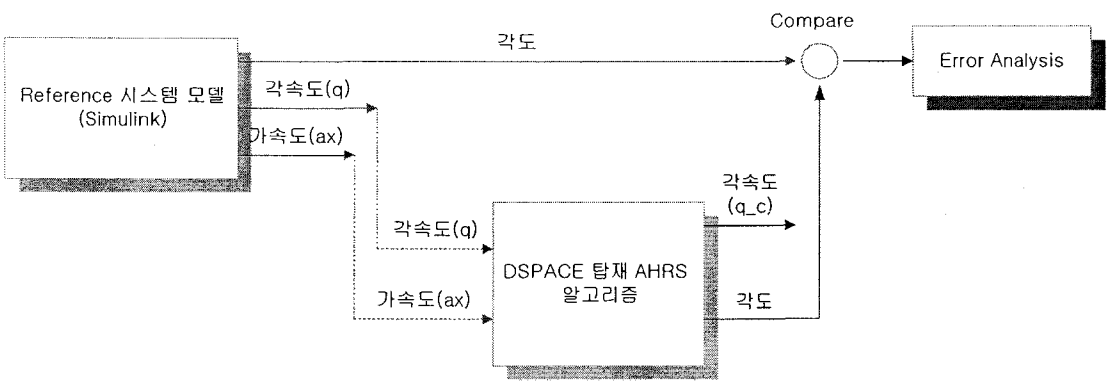


그림3.21 자이로/가속도계 혼합 알고리즘 검증 실험 블록도

아래의 그림3.22는 Pendulum(진자)주기 운동이다. 선형 상태 방정식으로 유도하여 그 Reference 값을 0.5 (rad/sec)로 주었다.

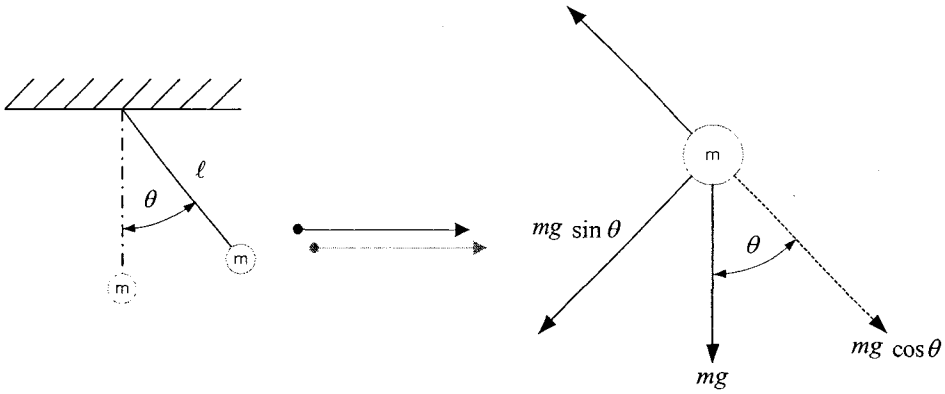


그림3.22 Pendulum 주기 운동

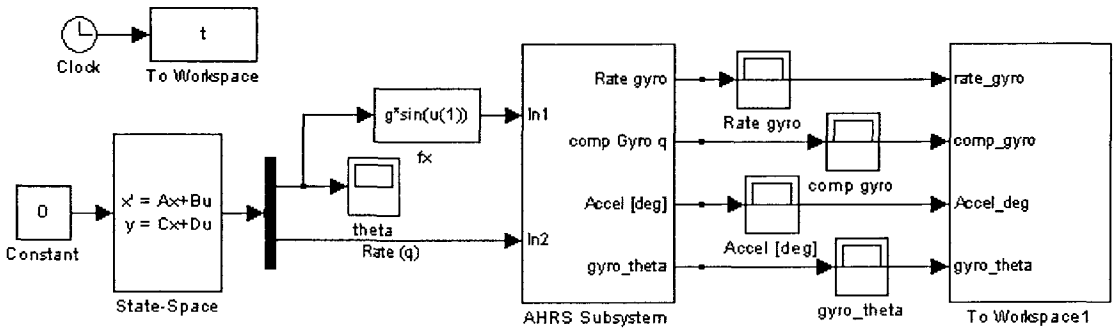
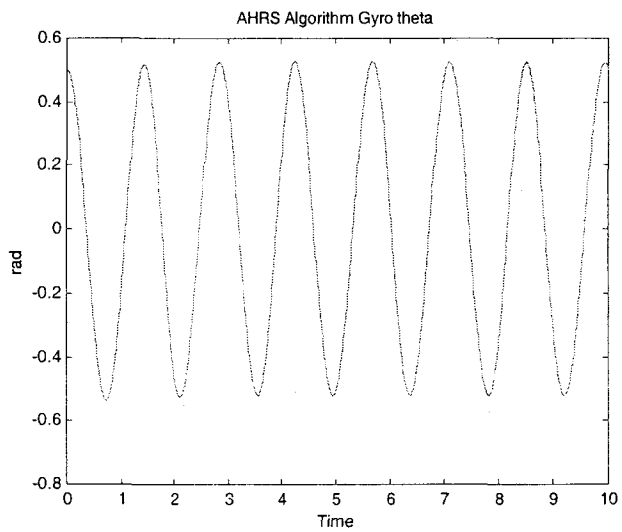
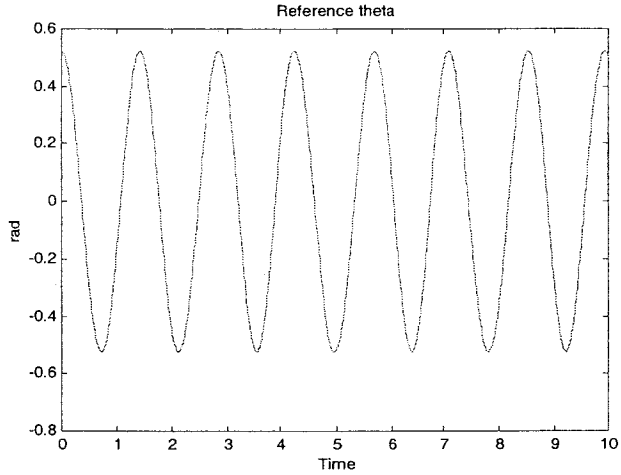
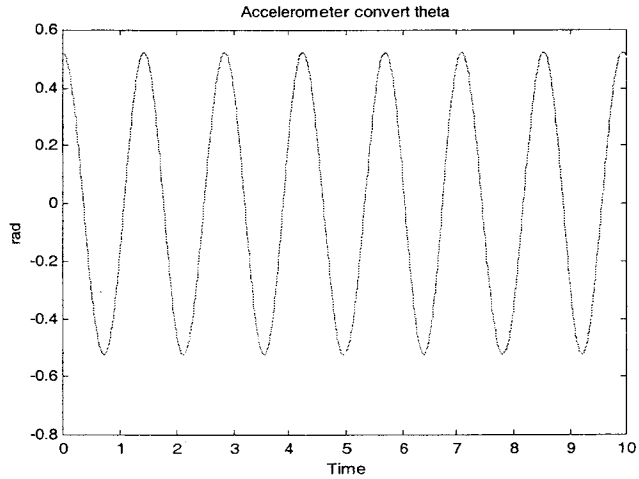


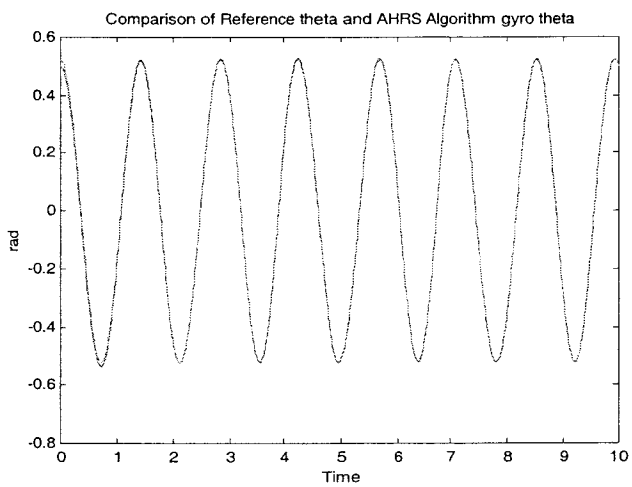
그림 3.23 자이로/가속도계 혼합 알고리즘 검증 Simulink

그 결과 주어진 입력 값에 대한 자이로의 응답 값과 가속도계의 응답 값이 거의 동일하게 나타남을 아래의 그래프에서 볼 수 있다









o 김발 시스템의 구동기 선정 및 시험

• 소요 토오크/정격출력 계산

-소요 토오크

Max. Inertia : Pan < 0.12 Kg-m2

: Tilt < 0.12 Kg-m2

Travel Rate : 60°/s

$$T = J/g \times (\frac{dw}{dt}) + T_f + T_d$$

$$= \frac{0.12\,(Kg-m^2)}{9.8\,(m/s^2)} \times \frac{1.04\,(rad/s)}{0.5\,(s)} + T_f + T_d$$

$$= 0.012\,Kg-m-s^2 \times 2.08\,rad/s^2 + T_f + T_d$$

$$= 0.025\,Kg \cdot m + T_f + T_d$$

여기서,

$\frac{dw}{dt}$: *max. angular acceleration*

T_f : *Friction Torque*

T_d : *Imbalance Torque*

$$T = (0.025\,Kg \cdot m) \times \text{안전률} = (0.025\,Kg \cdot m) \times 2 = 0.051\,Kg \cdot m$$

$$\cong 5\,Kg \cdot cm$$

-정격 출력

정격 출력(Output Power) $P = \frac{\pi}{30000} \times rpm \times (K_{m \times l_e} - C_o - C_v \times rpm)$에서

K_m = *Torque constant*

L_e = *Current* . max

C_o = *Friction torque , statics*

C_v = *Friction torque , dynamic*

$$P = \frac{\pi}{30000} \times 29900 \times (7.42 \times 0.41 - 0.15 - 8 \times 10^{-6} \times 29900)$$

$$= 8.1\ \ Watt$$

3. 구동기 선정 (엔코더, 감속기어 포함)

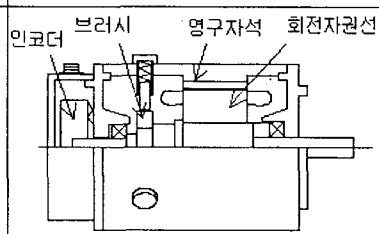
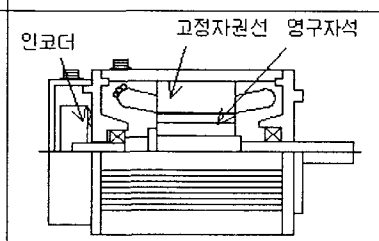
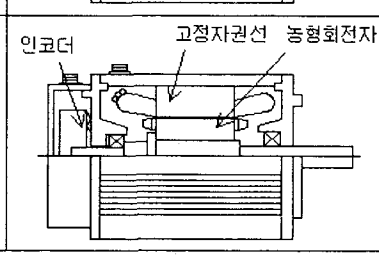
가. 구동기의 종류 및 요구조건

· 구동기 요구조건 설정

- 시정수(Time constant)가 빠를 것
- DC 이득이 클 것
- 최대 Torque
- Bandwidth(대역폭)
- 소형, 경량
- 유지보수가 편리
- 내환경성이 우수

영상 장치 김발은 Outer Gimbal, Inner Gimbal의 형태로 구성하고, 김발의 안정화 제어를 위한 구동기(Actuator)인 모터를 선정 한다. 가장 일반적으로 접할 수 있는 모터로서는 DC 모터, BLDC 모터, Induction 모터 등이 있다. 우선 선정하기에 앞서 아래의 표3.에서 각 모터들의 장, 단점을 살펴본다.

표3.3 구동기 비교표

	구 조	특 징	
		장 점	단 점
영구자석형 DC Motor		➢ 제어구조가 간단 ➢ 정전시 발전제동 가능 ➢ 소용량 및 저가 ➢ 높은 파워레이트	➢ 정류자의 유지보수가 필요 ➢ 브러시 마모에 따른 분진으로 청정지역 사용불가 ➢ 고속,대토그가 곤란 ➢ 마그네트 감자우려가 있음
영구자석형 BLDC Motor		➢ 유지보수가 편리 ➢ 내환경성이 우수 ➢ 대토크가 가능 ➢ 정전시 발전제동 가능 ➢ 소형 및 경량 ➢ 높은 파워레이트	➢ 제어가 비교적 복잡 ➢ 모터와 Amp의 1:1대응이 필요 ➢ 마그네트 감자우려 있음
Induction Motor		➢ 유지보수가 편리 ➢ 내환경성이 우수 ➢ 고속, 대토크가 가능 ➢ 대용량화가 용이 ➢ 구조가 견고	➢ 제어가 복잡 ➢ 소용량시 효율이 낮음 ➢ 정전시 제동불가 ➢ 온도에따라 특성이 변함

DC 모터는 제어구조가 간단하고, 가격 면에서 저가라는 장점을 보유하고 있지

만, 브러시(Brush) 마모에 따른 분진, 그리고 고속회전이나 대 토크가 곤란한 단점을 가지고 있다. Induction 모터는 위의 표에 보듯이 영상 장치 김발에 사용하기엔 온도에 따라 특성이 변하는 큰 단점을 가지고 있다. 그런 반면에, BLDC 모터는 소형 영상 장치 김발의 구동기 요구조건 설정에 부합하는 유지보수의 편리, 내환경성, 그리고 소형, 경량 등의 장점들을 보유하고 있다.

일반적인 BLDC(Brushless) 모터는 DC 모터가 가진 정류 장치를 모터에서 떼어내고, 대신에 전원을 제어 하여 회전자 위치에 맞는 전류를 흘리는 장치로서, DC 모터는 정류자의 개수를 늘림으로써 토오크 리플을 적게 할 수 있는데, BLDC 모터에서는 모터를 3상 권선으로 하고, 각 상의 전류를 구형파 혹은 정현파의 교번 전류를 흘려 구동한다. BLDC 모터의 원리는 아래의 그림3.24와 같다.

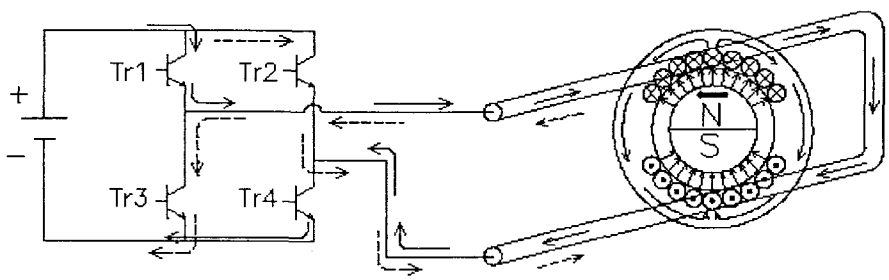


그림3.24 BLDC 모터의 원리

Tr1과 Tr3이 ON상태가 되면 실선의 방향으로 전류가 흐르게 된다. 이 전류는 플레밍의 왼손법칙에 의하여 코일에 시계방향으로 힘을 발생한다. 이때 코일이 고정되어 있으면 반발력에 의하여 회자의 자석이 반시계방향으로 회전하게 된다. 회전자가 반회전하게 되면 Tr2와 Tr4를 ON 상태로 만들어 전류방향을 반대로(점선방향) 하면 연속적인 회전을 하게 된다.

최종적으로 선정한 구동기는 스위스 Minimotor사의 BLDC Servo Motor-1628T 024B이다.

이 제품의 형상은 아래의 그림3.25에 나타나 있다.

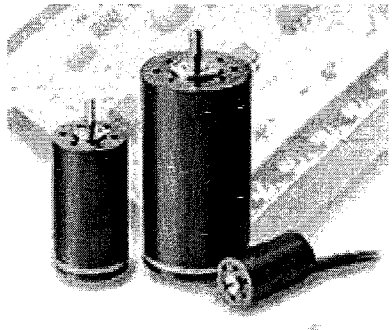


그림3.25 BLDC Servo Motor

이 제품의 모터 자체의 특징은 아래와 같다.

표3.4 BLDC Servo Motor-1628T의 특징

	BLDC Servo Motor-1628T	비고
Norminal voltage	24	V
회전수	29,900	rpm
Torque	0.012	Nm
Weight	31	gram
작동온도범위	-35 ~ 125	℃
정격 출력	11	Watt

위의 특징 중에서 토크(Torque)의 값이 계산된 소요토크에 비해 작으므로 감속기를 연결하여 원하는 토크의 값으로 설정한다.

아래의 그림3.26은 Planetary Gearheads RGHD 16/7 감속기 제품의 형상이다.

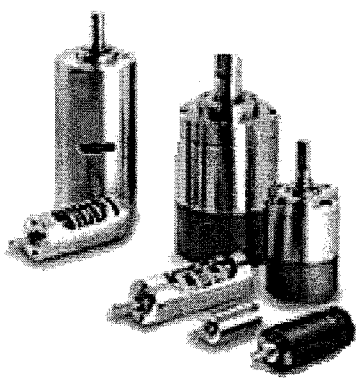


그림3.26 Reduction (RGHD 16/7)

RGHD 16/7 Reduction ratio를 14 : 1로 설정하여, 아래와 같은 구동기 제품에 대한 최종적인 특징을 나타낸다.

- Continuous Operation Output Torque : 0.24 Nm

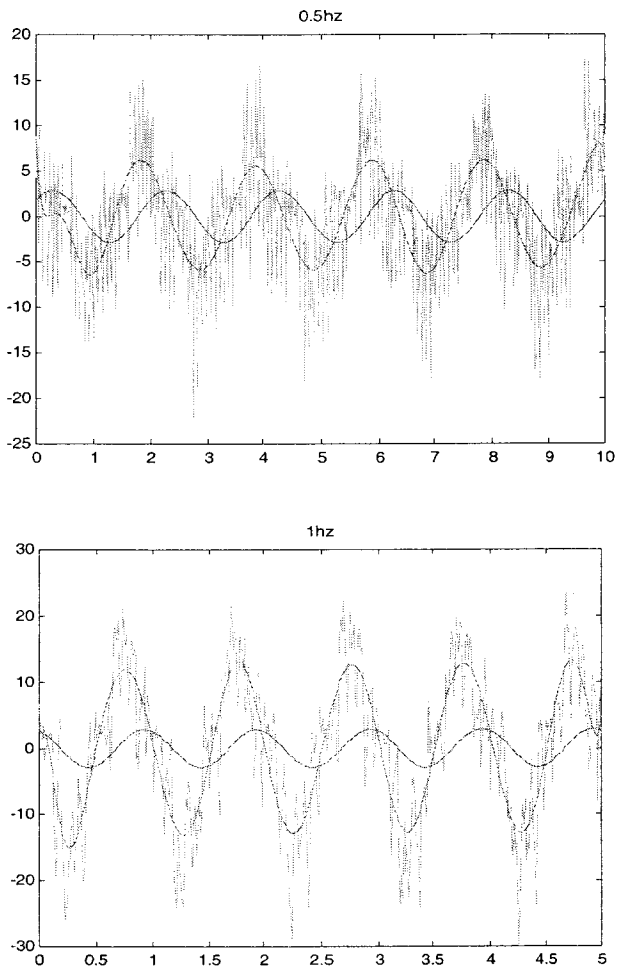
- Intermittent Operation Output Torque : 0.36 Nm
- Reduction RPM : 2200 rpm
- Total Length : 49.1 mm

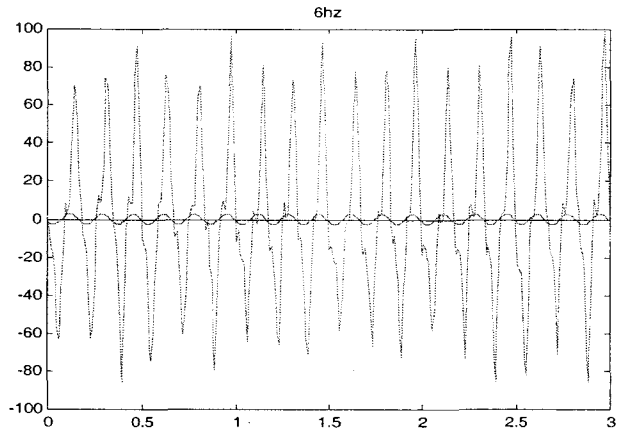
4. 1축 김발 제어 시스템 설계

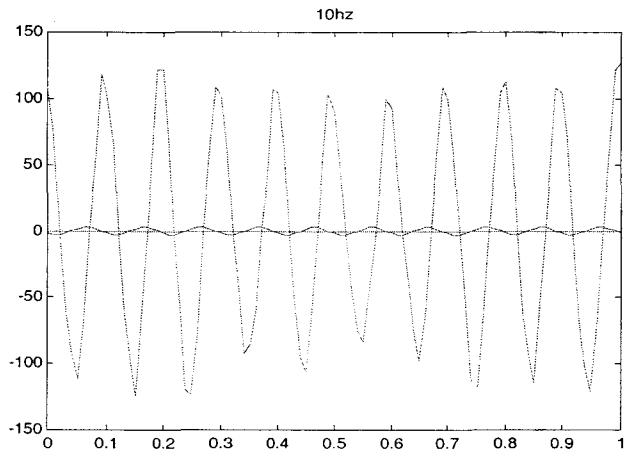
가. 동력학 모델링 추출 기법 연구

1축 김발 동력학 모델링 추출 기법은 Bode Plot 방법으로 연구하였다. 이 방법은 Time domain상이 아닌 Frequency domain상에서 실험 데이터를 기반으로 Bode 그래프를 그려 DC 게인요소, 미분, 적분 요소, 2차 미.적분 요소 등을 찾아 모델링을 추출한다. 입력 각도에 대한 출력 각속도의 데이터로 모델링을 한다.

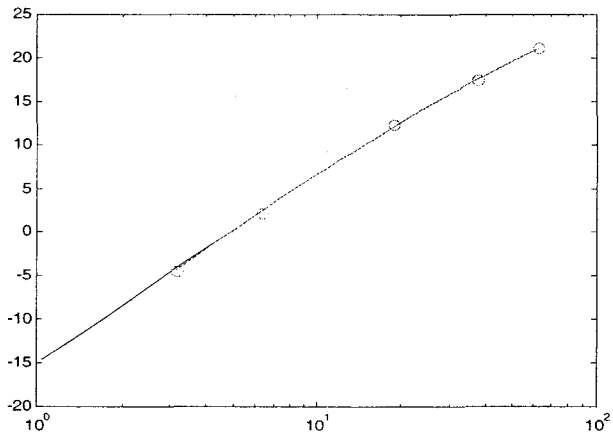
아래의 그래프는 모터 입력 각도에 대한 김발 자이로의 각속도 출력 값을 나타낸 그림이다.





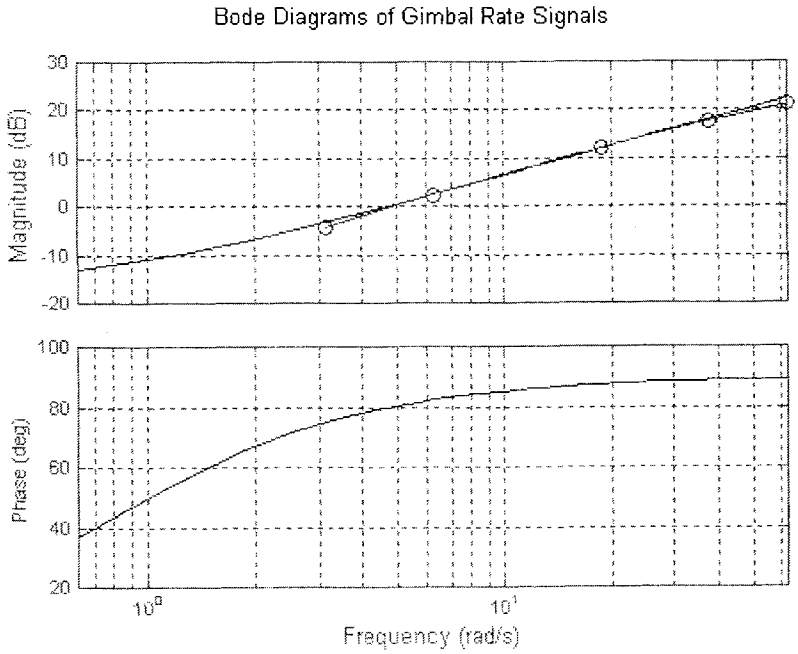


위의 그래프에서 각도와 각속도 값의 비를 semilogx를 취하면 아래와 같은 그래프가 그려진다.



위의 그래프를 보면 X축의 0(zero)과 만나는 Y축의 dB(데시벨 : decibel)값이 DC 게인요소이다. 그리고 미분요소가 존재한다.

이렇게 구해진 전달 함수로 아래와 같은 Bode plot을 한다.



제어대상의 Plant와 모델링의 오차를 확인하기 위해, 위에서 구한 전달 함수로 Simulink를 구성하여 실험 데이터와 비교하여 본다.

아래의 그림3.27는 전달함수 Simulink이다.

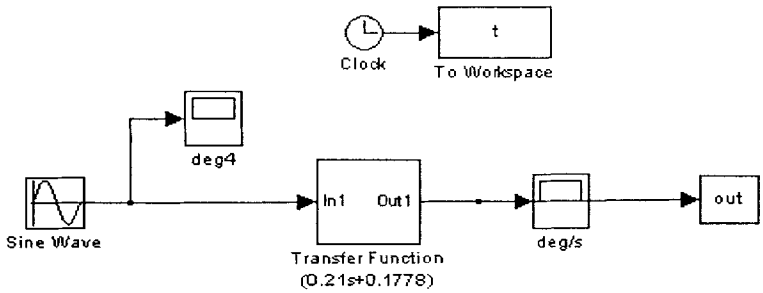
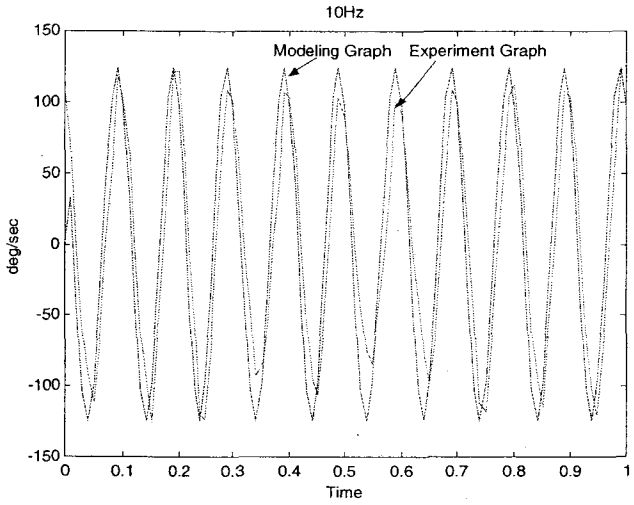
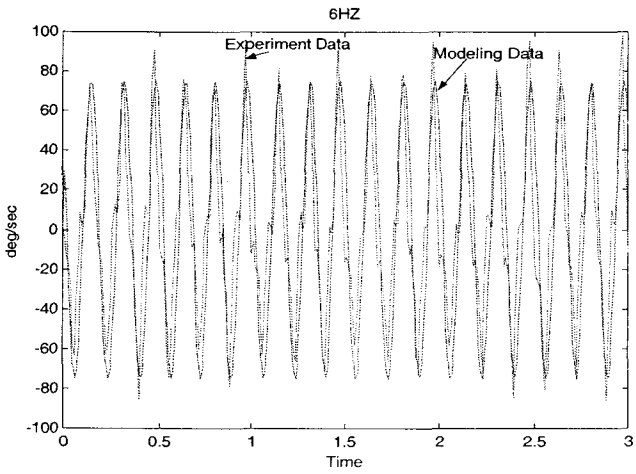


그림3.27 전달함수 Simulink

모델링으로 구성된 그림19.의 출력 값과 실험 데이터 값과 비교한 그래프는 아래에 나타나 있다.


10Hz
Modeling Graph
Experiment Graph
deg/sec
Time
150
100
50
0
-50
-100
-150
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


6HZ
Experiment Data
Modeling Data
deg/sec
Time
100
80
60
40
20
0
-20
-40
-60
-80
-100
0
0.5
1
1.5
2
2.5
3

제 4 절 김발 제어기 개발

1. Gimbal 제어기 요구사항

본 과제의 연구개발계획요구서(RFP: Request For Proposal)에 나타난 Gimbal 제어기 및 탑재 영상 장치의 요구사항은 다음과 같다.

- 안정정확도: 100 mrad
- 구동각도: 요 축 0 ~ 360 도, 피치 축 0 ~ 120 도
- 영상 감지성능: 2km 거리의 2 m 크기의 물체를 감지(detect)할 수 있어야함

이상의 요구사항으로부터 도출된 제어기의 요구사항은 다음과 같다.

- 기능 요구사항: 제어 알고리듬은 Gimbal의 자세 정보를 각속도 자이로 및 각측정 센서 등의 측정값을 사용하여 추정하고 이를 사용하여 Gimbal의 안정화 제어 명령 및 자세 명령을 자동적으로 계산하여야 한다.
- 설치 요구사항: 제어 알고리듬은 Gimbal 제어기 프로세서에 탑재되어 수행되어야 한다.
- 성능 요구사항: Gimbal 제어기는 안정정확도 요구사항을 만족할 수 있도록 100 Hz 이상의 속도로 제어 명령을 생성할 수 있어야 한다.

위의 요구사항 중 성능 요구사항에 명시된 100 Hz 이상의 제어 명령 생성 요구사항은 다음의 경험 법칙(rule of thumb)에 의해 유도되었다.

$$(\text{안정 정확도}) \propto 1 / (\text{제어 대역폭}) \qquad (1)$$

2. Gimbal 제어기 시스템 설계

소형 영상 Gimbal 시스템은 크게 Gimbal 구조물, Gimbal 제어기 및 Gimbal 조종부로 이루어진다. 그림 3.28에는 소형 영상 gimbel 시스템의 구조가 나타나 있다.

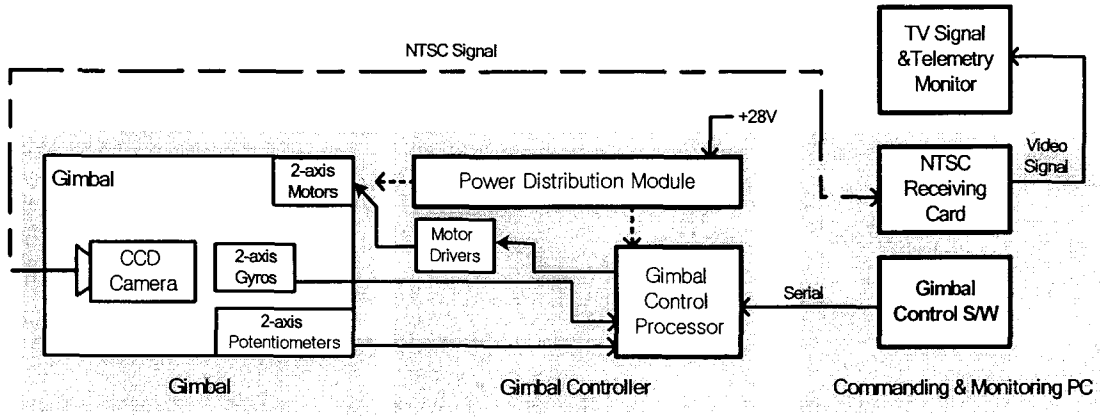


그림 3.28 소형 영상 Gimbal 시스템의 구조도

Gimbal 구조물에는 영상 센서 외에도 2축의 각속도 자이로 및 포텐셔미터(Potentiometer)가 탑재되어 각 축의 각속도와 각 변화를 측정하고 또한 2축의 구동 모터가 탑재되어 Gimbal 제어기로부터 생성된 명령에 따라 Gimbal의 자세를 제어하게 된다. Gimbal 제어기는 조종 명령으로부터 생성된 자세 명령과 현재 측정된 자세 각속도 및 자세 각으로부터 모터의 구동 명령을 실시간으로 생성한다. Gimbal의 조종명령은 Gimbal 제어 소프트웨어에서 입력해 줄 수 있다, 그림 3.28에 나타난 Gimbal 제어기의 구성 모듈의 역할은 다음과 같다.

- Gimbal 제어기 프로세서 (Gimbal Control Processor) : 센서 데이터와 구동기를 바탕으로 Gimbal의 안정화 명령(rate command)과 자세 명령(attitude command)을 생성한다. Gimbal 안정도 요구조건에 따라 빠른 계산을 요구하는 경우가 대부분이므로 DSP 프로세서를 사용하였다.
- PDM (Power Distribution Module) : 외부의 +28V 전원을 받아들여 DC/DCconversion을 통해 필요 전압을 생성하고 각 모듈에 분배하는 역할을 한다. 서보 모터와 탑재 영상 센서의 전원은 PDM에서 공급받지 않고 외부의 +28V를 직접 사용한다.
- 모터 구동기 (Motor Driver) : 서보 모터를 구동(drive)하는 회로로써 보통 모터를 구매할 때 공급자로부터 제공된다.
- 탑재 제어 소프트웨어 : Gimbal 제어 명령을 생성하는 탑재 제어 소프트웨어는 실시간으로 서보 모터의 구동 명령을 생성한다.

3. Gimbal 제어 알고리듬 개발

가. Gimbal 운동 방정식

Gimbal의 간략한 운동 모델은 다음과 같이 유도된다.

$$\ddot{\varphi} = -\alpha_{\varphi}\dot{\varphi} + T_{\varphi}/J_{\varphi}$$
$$\ddot{\theta} = -\alpha_{\theta}\dot{\theta} + T_{\theta}/J_{\theta} \tag{2}$$

여기에서 φ_o, θ_o는 각각 Gimbal의 요 및 피치 각이고 T_{φ}, T_{θ}는 서보 모터에서 생성된 토오크이다. Gimbal의 형상이 대칭에 가깝다고 가정하여 커플링 항들은 무시하였고 ($J_{\varphi\theta} << 1$) $\alpha_{\varphi}, \alpha_{\theta}$는 각각 요 및 피치 축의 베어링의 마찰에 의한 시상수이다. 또한 보통 서보 모터는 1차 시스템으로 표현하므로 제어기의 명령 u_{φ}, u_{θ}에 대한 모터 토오크의 응답은 다음과 같은 식으로 표현되며

$$\dot{T}_{\varphi} = -\tau_{\varphi}T_{\varphi} + u_{\varphi}$$
$$\dot{T}_{\theta} = -\tau_{\theta}T_{\theta} + u_{\theta} \tag{3}$$

(3)식과 (2)식을 합하면 다음과 같은 Gimbal의 운동 방정식이 유도된다.

$$\begin{bmatrix} \ddot{\varphi} \\ \ddot{\theta} \\ \dot{T}_{\varphi} \\ \dot{T}_{\theta} \end{bmatrix} = \begin{bmatrix} -\alpha_{\varphi} & 0 & 1/J_{\varphi} & 0 \\ 0 & -\alpha_{\theta} & 0 & 1/J_{\theta} \\ 0 & 0 & -\tau_{\varphi} & 0 \\ 0 & 0 & 0 & -\tau_{\theta} \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ T_{\varphi} \\ T_{\theta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_{\varphi} \\ u_{\theta} \end{bmatrix} \tag{4}$$

한편 기계적으로 Gimbal의 피치 각은 어느 정도 이상의 값을 갖지 못하게 설계된다. 이는 다음과 같은 inequality constraint로 주어진다.

$$\underline{\theta} \le \theta \le \bar{\theta} \tag{5}$$

나. Gimbal 제어기 설계

Gimbal의 제어는 2축 제어를 기본으로 한다. 제어 알고리듬의 구조는 다음 그림 3.29와 같다.

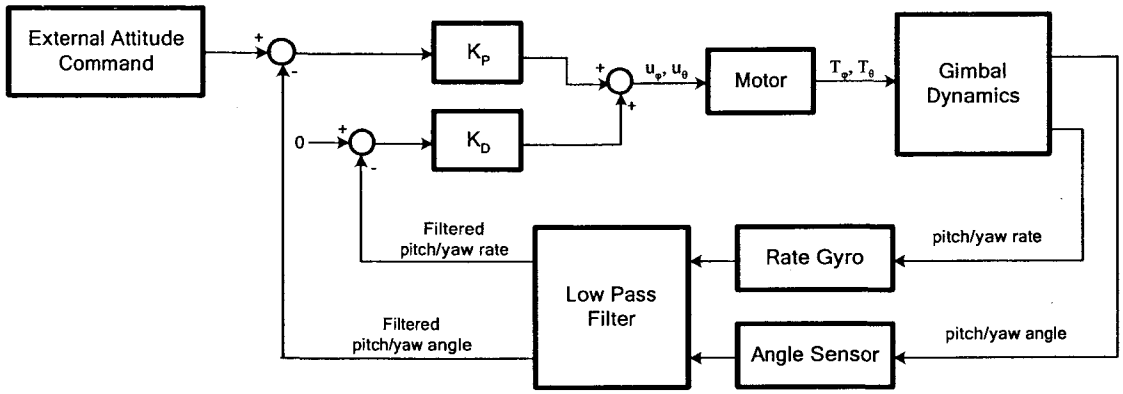


그림 3.29 Gimbal 제어 알고리듬의 구조

그림 3.29에서 보이는 바와 같이 Gimbal을 제어하는 모터의 제어 입력은 자이로로부터 얻어지는 안정화 명령과 외부에서 주어지는 자세 명령을 추종하도록 각 추종 명령의 두 부분의 합으로 나타난다. 이러한 제어기 구조는 비행체의 종운동 제어기 등에서 많이 사용되는 전형적인 PD제어기의 형태이다.

제어 법칙은 다음과 같이 표현된다. 우선 Gimbal 자세각 명령을 각각 φ_c, θ_c 라 하면 PD 제어 법칙에 의해 제어 명령은 다음과 같다.

$$u_\varphi = K_{P\varphi\varphi}(\varphi_c - \varphi) + K_{P\varphi\theta}(\theta_c - \theta) - K_{D\varphi\varphi}\dot{\varphi} - K_{D\varphi\theta}\dot{\theta}$$
$$u_\theta = K_{P\theta\varphi}(\varphi_c - \varphi) + K_{P\theta\theta}(\theta_c - \theta) - K_{D\theta\varphi}\dot{\varphi} - K_{D\theta\theta}\dot{\theta} \quad (6)$$

위 식은 Gimbal 제어기 프로세서의 탑재 소프트웨어에 구현되어 제어 명령을 생성하는데 사용된다. 이 식을 (4)식의 운동 방정식에 대입하면 다음과 같은 폐회로 운동 방정식이 얻어진다.

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \ddot{\varphi} \\ \ddot{\theta} \\ \dot{T}_\varphi \\ \dot{T}_\theta \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\alpha_\varphi & 0 & 1/J_\varphi & 0 \\ 0 & 0 & 0 & -\alpha_\theta & 0 & 1/J_\theta \\ -K_{P\varphi\varphi} & -K_{P\varphi\theta} & -K_{D\varphi\varphi} & -K_{D\varphi\theta} & -\tau_\varphi & 0 \\ -K_{P\theta\varphi} & -K_{P\theta\theta} & -K_{D\theta\varphi} & -K_{D\theta\theta} & 0 & -\tau_\theta \end{bmatrix} \begin{bmatrix} \varphi \\ \theta \\ \dot{\varphi} \\ \dot{\theta} \\ T_\varphi \\ T_\theta \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K_{P\varphi\varphi} & K_{P\varphi\theta} \\ K_{P\theta\varphi} & K_{P\theta\theta} \end{bmatrix} \begin{bmatrix} \varphi_c \\ \theta_c \end{bmatrix} \quad (7)$$

요 축과 피치 축의 커플링 항을 무시하면, 즉 $J_{\varphi\theta} \approx 0$ 이라 가정하면, Gimbal 제어기의 제어 법칙은 다음과 같이 각 축에 대한 제어기의 설계 문제로 단순화된다.

$$u_\varphi = K_{P\varphi\varphi}(\varphi_c - \varphi) - K_{D\varphi\varphi}\dot{\varphi}$$
$$u_\theta = K_{P\theta\theta}(\theta_c - \theta) - K_{D\theta\theta}\dot{\theta} \quad (8)$$

이 식을 사용하면 Gimbal의 운동 방정식은 다음과 같이 요 축 운동 방정식과 피

치 축 운동 방정식으로 분리된다.

요 축 운동 방정식:

$$\begin{bmatrix} \dot{\varphi} \\ \ddot{\varphi} \\ \dot{T}_{\varphi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\alpha_{\varphi} & 1/J_{\varphi} \\ -K_{P\varphi\varphi} & -K_{D\varphi\varphi} & -\tau_{\varphi} \end{bmatrix} \begin{bmatrix} \varphi \\ \dot{\varphi} \\ T_{\varphi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ K_{P\varphi\varphi} \end{bmatrix} \varphi_c \tag{9}$$

피치 축 운동 방정식:

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{T}_{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\alpha_{\theta} & 1/J_{\theta} \\ -K_{P\theta\theta} & -K_{D\theta\theta} & -\tau_{\theta} \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ T_{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ K_{P\theta\theta} \end{bmatrix} \theta_c \tag{10}$$

다. Gimbal 제어기 이득 선정

본 제어기에서 결정해야 할 제어 이득은 각 축의 비례 및 미분 이득이다. 이득 결정 방법은 두 축이 같으므로 여기에서는 요 축에 대한 내용만을 서술하고자 한다.

(9)식에 표현된 요 축에 대한 운동 방정식을 사용하여 요 각 명령으로부터 요 각 응답에 이르는 전달 함수를 유도하면 다음과 같다.

$$\begin{aligned} \frac{\varphi}{\varphi_c}(s) &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} s & -1 & 0 \\ 0 & s+\alpha_{\varphi} & -1/J_{\varphi} \\ K_{P\varphi\varphi} & K_{D\varphi\varphi} & s+\tau_{\varphi} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ K_{P\varphi\varphi} \end{bmatrix} \\ &= \frac{K_{P\varphi\varphi}/J_{\varphi}}{s^3 + (\alpha_{\varphi}+\tau_{\varphi})s^2 + (\alpha_{\varphi}\tau_{\varphi} + K_{D\varphi\varphi}/J_{\varphi})s + K_{P\varphi\varphi}/J_{\varphi}} \end{aligned} \tag{11}$$

(11)식으로부터 요 축의 폐회로 극점의 위치는 다음의 방정식의 해로 주어진다.

$$s^3 + (\alpha_{\varphi}+\tau_{\varphi})s^2 + (\alpha_{\varphi}\tau_{\varphi} + K_{D\varphi\varphi}/J_{\varphi})s + K_{P\varphi\varphi}/J_{\varphi} = 0 \tag{12}$$

극점의 위치가 다음과 같이 정해진다고 가정하면

$$s = -a_{\varphi}, s = -\omega_{n\varphi}(\zeta_{\varphi} \pm j\sqrt{1-\zeta_{\varphi}^2})$$

다항식의 계수 비교 식들로부터 다음의 식들을 얻을 수 있다.

$$2\omega_{n\varphi}\zeta_{\varphi} + a_{\varphi} = \alpha_{\varphi} + \tau_{\varphi}$$
$$2a_{\varphi}\omega_{n\varphi}\zeta_{\varphi} + \omega_{n\varphi}^{2} = \alpha_{\varphi}\tau_{\varphi} + K_{D\varphi\varphi}/J_{\varphi}$$
$$a_{\varphi}\omega_{n\varphi}^{2} = K_{P\varphi\varphi}/J_{\varphi} \tag{13}$$

복소평면에서 커플된 두 극점이 실제적으로 제어 성능을 결정짓는 극점들이므로 이 값들을 결정하면, 즉 $\omega_{n\varphi}, \zeta_{\varphi}$를 결정하면 나머지 하나의 극점의 위치와 제어기 이득 값들을 결정할 수 있다.

피치 축의 경우에도 유사하게 다음과 같이 극점의 위치를 결정하면

$$s = -a_{\theta}, s = -\omega_{n\theta}(\zeta_{\theta} \pm j\sqrt{1-\zeta_{\theta}^{2}})$$

다음의 다항식을 풀어서 제어기 이득 값을 결정할 수 있다.

$$2\omega_{n\theta}\zeta_{\theta} + a_{\theta} = \alpha_{\theta} + \tau_{\theta}$$
$$2a_{\theta}\omega_{n\vartheta}\zeta_{\theta} + \omega_{n\theta}^{2} = \alpha_{\theta}\tau_{\theta} + K_{D\theta\theta}/J_{\theta}$$
$$a_{\theta}\omega_{n\theta}^{2} = K_{P\theta\theta}/J_{\theta} \tag{14}$$

4. 영상 센서

가. 영상 센서 요구사항

본 과제에서 사용될 영상 센서의 요구사항은 2 Km 거리에서 2 m (사람 크기) 정도의 물체를 감지(detect)하는 것이다.

나. 영상 센서 성능 및 규격

선정된 영상 센서는 C&B Technology의 CNB-AN200이다. 본 영상 센서는 CCD를 기반으로 한 컬러 카메라로써 광학 줌 및 디지털 줌 방식을 겸용으로 사용하여 최대 220배까지의 확대가 가능하다. 선정된 영상 센서의 모습은 그림 32에 나타나 있다.

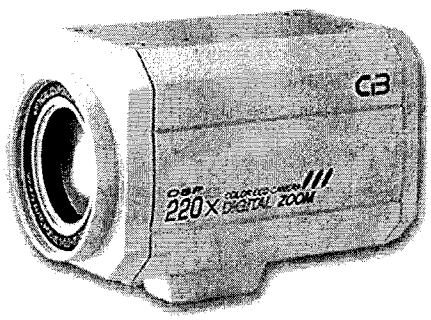


그림 3.30 영상 센서 CNB-AN200의 모습

영상 센서 CNB-AN200의 보다 구체적인 사양은 다음 표 4.1과 같다.

표 4.1 카메라 성능

항목	사양
출력 신호	NTSC : 525 Lines
스캔 주파수	15.734 KHz (H), 59.94 KHz (V)
센서	1/4 inch interline transfer super HAD CCD
전체 픽셀 수	811(H)× 508(V)
유효 픽셀 수	768(H)× 494(V)
비디오 출력 레벨	1.0Vp-p (75 Ohms, composite)
광학 렌즈	22 광학 줌 (F1.6, f=3.9 ~ 85.8 mm) Video AF
디지털 줌	2x ~ 10x variable
관측각 (Field of View)	Horizontal : 47° (wide) / 3° (tele)
최소 촬영 거리	1 cm (wide) / 1 m (tele)
초점 방식	Auto / Manual
I/O	Control & Power : 6-pin circle, Video : BNC
전원	DC 9V ~ 15V
소모 전력	Max 4.2 W / 350 mA
크기 (W×H×D)	60×60.4×103 mm
무게	345 g

다. 인터페이스

CNB-AN200의 후면에는 원형의 6-pin 커넥터가 있다. 이 커넥터는 카메라 제어를 위한 것인데 이 각 핀의 인터페이스는 다음의 표 4.2와 같다.

표 4.2 CNB-AN200의 제어 인터페이스

핀 번호	이름	I/O	라인 색	참고
1	Zoom(-:Tele, +Wide)	Input	Green	±6V (Limit +3~13V, -3~-13V)
2	Focus(-:Near, +Far)	Input	White	±6V (Limit +3~13V, -3~-13V)
3	A/D KEY	Input	Brown	A/D(Voltage Divide) Control Input
4	COM		Yellow	Pair Ground for Zoom, Focus
5	GND		Black	Pair Ground for Power Supply
6	DC 12V	Input	Red	Power Supply 12V Input

이 중 3번의 A/D KEY는 전압에 따른 카메라의 동작을 제어하는 핀으로써 다음의 그림 4.4에 나타난 것과 같은 방식을 통해 동작을 수행한다.

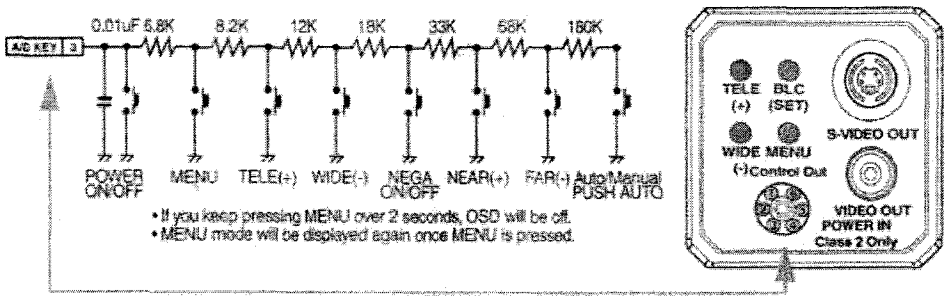
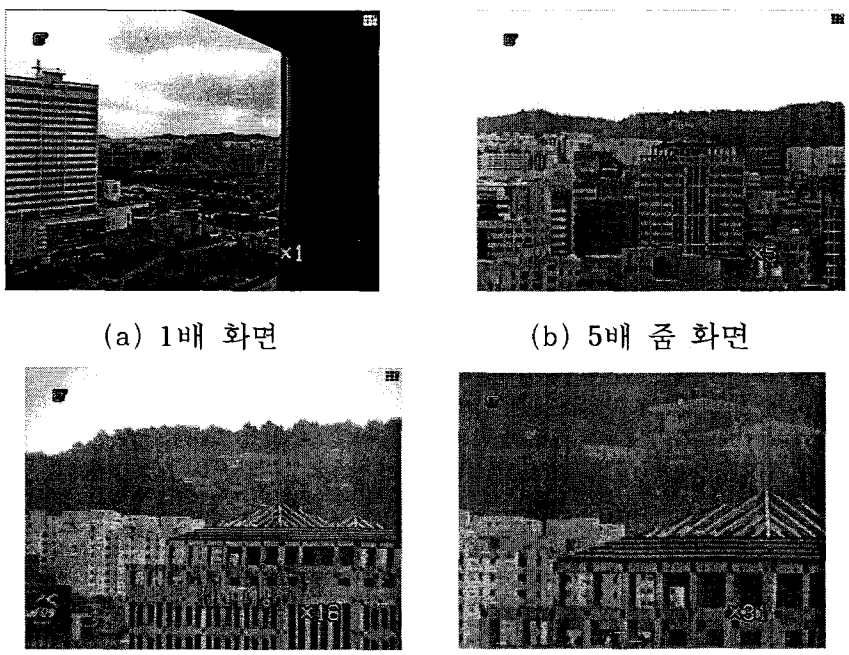


그림 3.31 전압 분배 방식에 따른 KEY의 동작

기계적인 장착을 위해 CNB-AN200은 아랫면에 Gimbal 구조물에 부착 가능한 별도의 인터페이스를 제공하고 있다.

라. 영상 센서 성능 검증

구매한 CNB-AN200 카메라의 성능을 검증하기 위하여 대전시 서구 둔산동 사학연금회관 13층 창문을 통해 약 3 km 전방의 물체를 줌인 하여 촬영하였다. 촬영된 시각은 오후 5시 경이고 기후는 흐린 날이다. 따라서 일조량이 많은 밝은 날 낮에 촬영할 경우 이 영상보다는 훨씬 더 좋은 질의 영상을 얻을 수 있다. 촬영된 영상은 그림 3.32에 나타나 있다.



(a) 1배 화면 (b) 5배 줌 화면

(c) 16배 줌 화면 (d) 31배 줌 화면

그림 3.32 CNB-AN200의 영상 성능

5. Gimbal 제어기 하드웨어

가.1 Gimbal 제어기 프로세서

(1) Gimbal 제어기 보드의 구조

Gimbal 제어기 보드의 구조는 다음 그림 3.33과 같다.

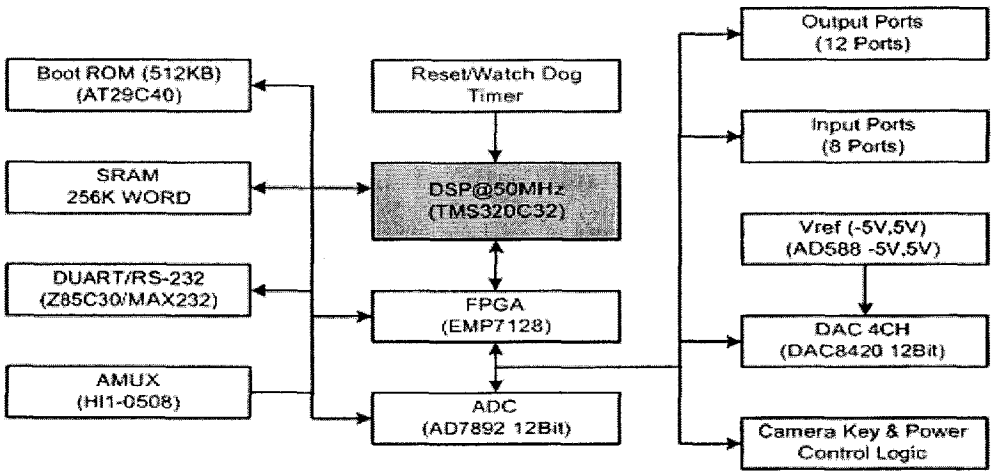


그림 3.33 Gimbal 제어기 시스템 구조도

Gimbal제어기 보드는 중앙 처리 장치로 50MHz로 동작하는 TMS320C32를 사용하며 다음과 같은 자원을 가지고 있다.

- 외부 확장 메모리: 고속 SRAM(K6R4016C1D) (256K x 16) 2개
- 부트용 플래시 메모리: AT29C040A, 512K x8
- DSP의 (POR)Power On Reset과 Watchdog Timer(max706)
- 비동기 통신 소자: Zilog 사의 Z85C30, RS232 통신
- A/D 변환기: AD7892, 12비트 8채널
- D/A 변환기: DAC8420 ,(output:-5V ~ 5V ,12비트, 4채널)
- 입력 포트: 8개
- 출력 포트: 14개
- XDS510 접속, BOOT 커넥터 지원

(2) DSP(TMS320C32)의 사양

TMS320은 다양한 직렬 통신 환경 및 빠른 계산 능력을 갖고 있고 MIL STD 급의 칩도 생산되고 있으므로 본 과제에서 요구되는 제어기 프로세서로 가장 적합하다고 판단된다. 다음의 그림 3.34는 TMS320C32의 구성도를 보여주고 있다.

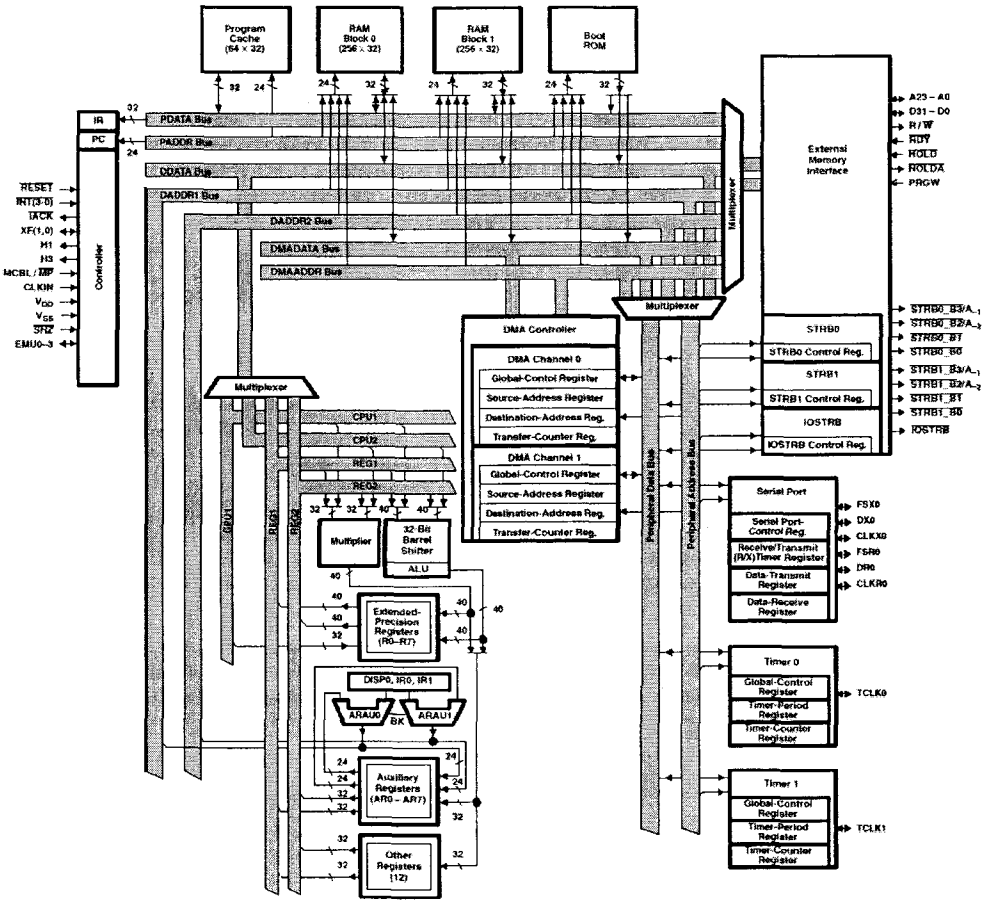


그림 3.34 TMS320C32 의 구성도

TI사의 DSP TMS320C32의 성능을 정리해보면 아래와 같다.

- 32비트 부동 소수점 연산 DSP
- 144핀 PQFP Package
- 처리 속도 : 40/50/60/80 MHz 모델이 있음
- 80MHz에서 40MIPS, 80MFLOPS 처리 속도
- 내부 RAM : 512 x 32 용량
- 내부 프로그램 캐시 : 64 x 32 크기
- 전체 어드레스 공간 : 16M x 32 (24비트 어드레스)
- 외부 ROM 직접 실행 모드(마이크로프로세서 모드)와 부트 모드(마이크로컴퓨터 모드)가 있음. : ROM 부트(3 위치), 시리얼 부트(1) 가능
- 내부장치: 32비트 타이머(2), DMA제어기(1), 클럭 방식 직렬 통신 포트(1개)

(3) TMS320C32 메모리 Map과 Gimbal 제어기의 메모리 Map

Gimbal 제어기에서 사용하는 DSP의 메모리 구성은 아래의 그림 3.35과 같다.

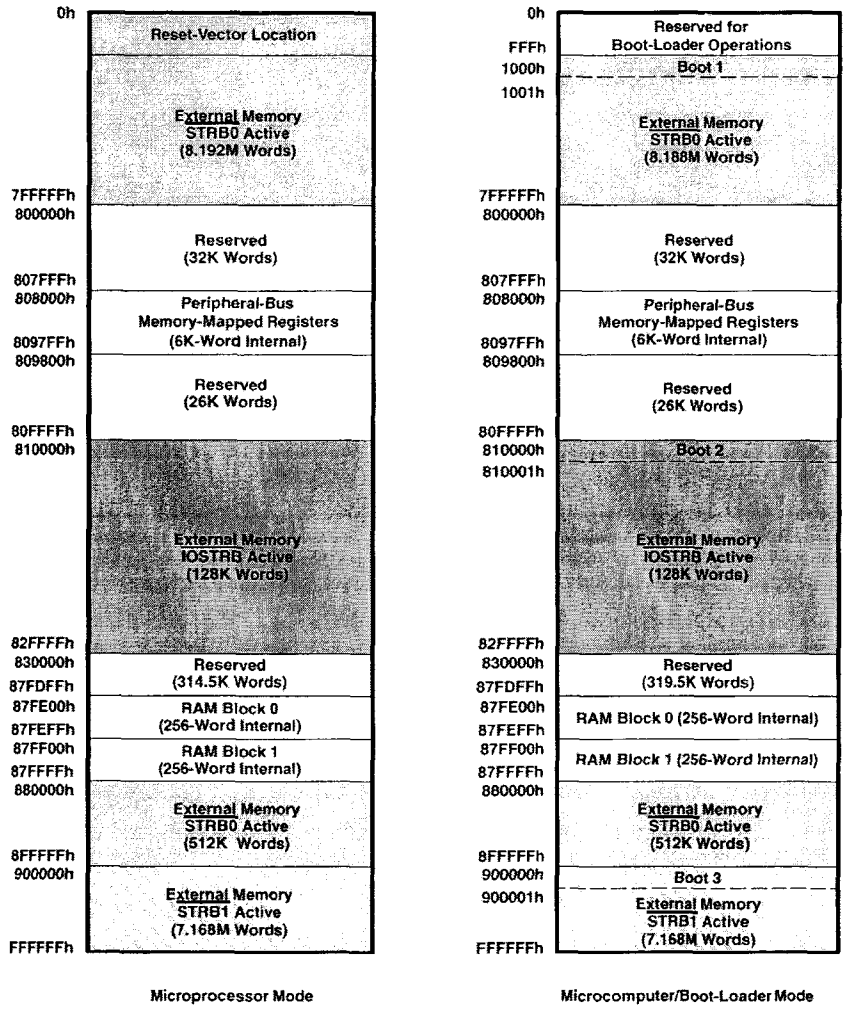


그림 3.35 TMS320C32의 Address Map

Gimbal 제어기에서 사용하는 주변소자의 Address는 다음의 표 4.3과 같다.

표 3.3 Gimbal 제어기 I/O Address Map

선택 신호	A23 A22 A21 A20	A15 ..A12	strobe	사용 영역	Remark
FROM\	1 0 0 1	x x	STRB0&1\	900000H ~ 9FFFFFH	R/W
SRAM\	0 1 0 0	x x	STRB0&1\	400000H ~ 7FFFFFH	R/W
DUART	1 0 0 0	0 0 0 0	IOSTRB\	810000H - 810FFFH	R/W
IOP	1 0 0 0	0 0 1 0	IOSTRB\	812000H ~ 812FFFH	R/W
OUTPUTP	1 0 0 0	0 1 0 0	IOSTRB\	814000H ~ 815FFFH	W
CAMERA	1 0 0 0	0 1 1 0	IOSTRB\	816000H ~ 816FFFH	W
AMUXSEL	1 0 0 0	1 0 0 0	IOSTRB\	818000H ~ 818FFFH	W
ADC	1 0 0 0	1 0 1 0	IOSTRB\	81A000H ~81AFFFH	R
ADCCTRL	1 0 0 0	1 0 1 0	IOSTRB\	81A000H ~81AFFFH	W
DACCTRL	1 0 0 0	1 1 0 0	IOSTRB\	81C000H ~81CFFFH	W
SYSCTRL	1 0 0 0	1 1 1 0	IOSTRB\	81E000H ~81FFFFH	R/W

아래의 내용은 제어기 프로그램에서 정의하여 사용하는 Header파일의 내용이다.

```
#ifndef __include_Gimbal_header_file__
#define __include_Gimbal_header_file__

#include "global.h"

#define MAINCLK         50000000L

/* Watchdog Function */
int g_WatchdogCnt = 0;

#define ClearWatchdogTimer()                                          \
   if((++g_WatchdogCnt) & 0x01)                                       \
       asm("    LDI  06h,IOF    ");          /* IOMr IOMr */              \
   else                                                               \
       asm("    LDI  02h,IOF    ");          /* IOMr IOMr */

#define GetEndOfConvert()          P1.2

#define SCC0CHACTRL ( *(char *)(0x810000 + 0x0002)) /* 10 read & write */
#define SCC0CHADATA ( *(char *)(0x810000 + 0x0003)) /* 11 read & write */
#define SCC0CHBCTRL ( *(char *)(0x810000 + 0x0000)) /* 00 read & write */
#define SCC0CHBDATA ( *(char *)(0x810000 + 0x0001)) /* 01 read & write */

#define IOP         ( *(char *)(0x810000 + 0x2000))    /* read & write */
#define ReadPort()   (IOP & 0x0FF)
#define OUTPORT      ( *(char *)(0x810000 + 0x4000))   /* read & write */
#define CAMERAKEY    ( *(char *)(0x810000 + 0x6000))   /* write */
#define AMUXSEL      ( *(char *)(0x810000 + 0x8000))   /* write */
#define ADCPORT      ( *(int  *)(0x810000 + 0xA000))   /* read  */
#define ReadAdc()    (ADCPORT & 0x0FFF)
#define CvtStart()   ( *(char *)(0x810000 + 0xA000)) = 0x00  /* write */
#define DACPORT      ( *(char *)(0x810000 + 0xC000))   /* write */
#define Start()      ( *(char *)(0x810000 + 0xE000)) = 0x00
#endif
```

(4) 클럭(Clock) 및 리세트(Reset) 회로

클럭 회로는 50MHz의 OSC를 사용하며, DSP의 클럭은 OSC의 출력을 직접 이용하지 않고 74HC14의 출력을 사용한다.

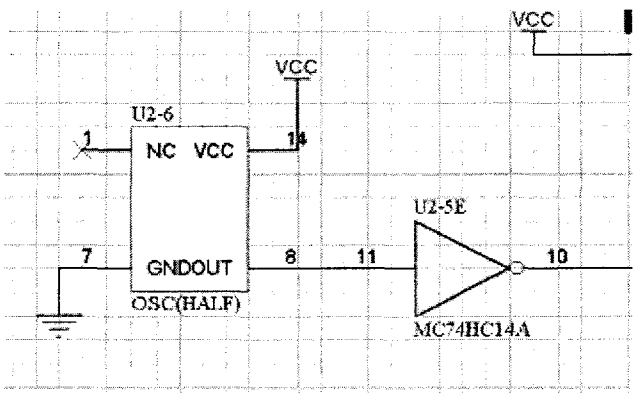


그림 3.36 클럭 회로

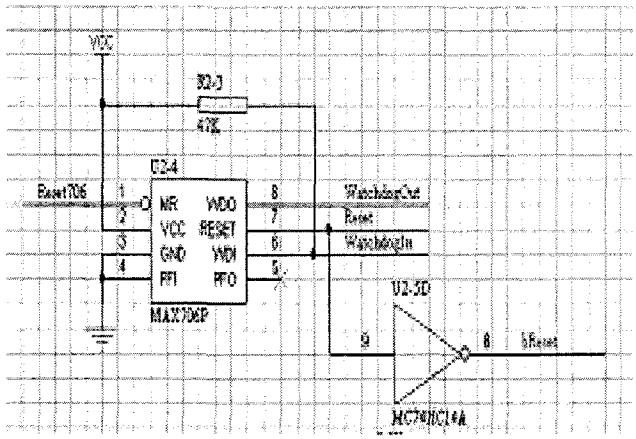


그림 3.37 리세트 회로

(5) 외부 인터럽트(INT0\, INT1\, INT2\, INT3\)/부트선택 입력 신호

TMS320C32에서 사용할 수 있는 외부 인터럽트 신호는 4개가 있는데, 부트모드를 사용하는 경우 부트 선택 신호로도 사용 된다. 제어기보드는 2가지 부트 모드(부트 3,시리얼)를 사용하기 때문에 2개의 인터럽트 핀(INT2\, INT3\)은 부트 선택 핀으로 전담 시키고, 2개의 핀(INT0\, INT1\)을 유저 인터럽트 용으로 사용 하고 있다. 아래의 그림은 DSP의 Booting을 선택하는 점퍼로 1-2를 선택하면 Serial Boot을 선택하게 되고, 2-3를 선택하는 경우는 Boot ROM(AT29C040A)를 선택하여 Booting을 하게 되어 있다.

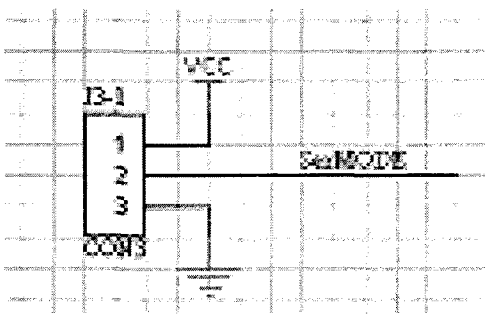


그림 3.38 Boot 선택 점퍼

표 3.4 인터럽트 / 부트 선택

핀	인터럽트 기능	부트 기능	제어보드
INT0\	외부 인터럽트 0	부트 1 (롬, 0x1000)	DUART 인터럽트
INT1\	외부 인터럽트 1	부트 2 (롬, 0x400000)	Reserve
INT2\	외부 인터럽트 2	부트 3 (롬, 0xFFF000)	부트 롬 (AT29C040A)
INT3\	외부 인터럽트 3	시리얼부트	시리얼 통신 부트

(6) RDY\ 입력 신호

RDY\ 신호의 기능은 DSP가 외부 장치에 데이터를 쓰거나 읽는 동작을 할 때 외부 장치의 응답 시간이 느려서 DSP의 액세스 시간을 만족 시키지 못할 때 DSP에게 처리 시간을 좀더 요청 할 때 사용 된다. RDY\ 신호가 High(1) 상태이면 한 사이클 동안 씩 기다리다가, Low(0) 상태가 되면 처리 명령을 완료 한다. 제어보드에서 RDY\ 생성 로직은 FPGA을 사용하여 생성 하는데, 항상 "Low"를 출력하게 되어 있고, DSP의 제어 Register 의 SWW bit를 제어하여, Software적으로 Wait Count를 설정하는 방식을 사용한다.

(7) 메모리 (SRAM & Flash-ROM)

DSP 보드의 외부 확장 RAM으로는 삼성의 16 비트SRAM (K64016C1D) 2개가 사용되는데, 메모리 용량은 각각이 128K×16 비트 (256K 바이트)이므로 RAM 전체 용량은 512 K바이트가 된다. 응답 시간10nsec RAM이 사용되어 0 (Zero) 웨이트 동작이 가능하다. 플래시 롬은 AMD 나 Atmel 사의 29F040 / 29F040이 사용되는데, 각각의 용량은 512KB로 응답 시간 90nsec를 사용하며, DSP32 B/D에서는 7개의 웨이트를 걸어 동작 시킨다. 플래시 롬은 사용자 운용 프로그램을 저장하여 롬 부트 동작을 통해 0 웨이트로 동작이 가능한 내부 메모리나 외부 RAM으로 이동 후 실행 된다.

(8) 비동기 통신 회로 (UART)

일반적으로 비동기 통신용 소자로 8250 / 16C450 / 16C550 등이 많이 사용되는데, 제어기 보드에서는 Zilog사의 SCC 85c30을 사용하였다. 제어기 보드에서는 통신 소자의 클럭 주파수는 14.7456MHz나 7.3728MHz를 사용한다.

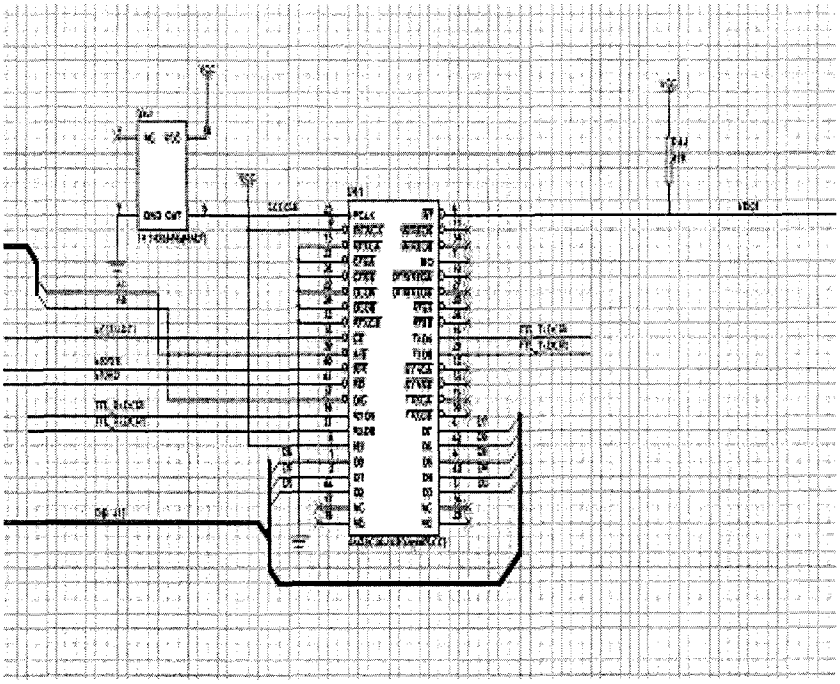


그림 3.39 비동기 통신(UART) 회로

(9) A/D 변환기 회로

아나로그 신호를 디지털 신호로 변환하는 A/D 변환 소자로는 아나로그 디바이스 사의 AD7892를 사용하였다. 이 소자의 특징은 아래와 같다.

- 해상도 : 12비트
- 채널 수 : 1 채널
- 변환 속도: 1.6 usec
- 입력 전압: -2.5V ~ +2.5V
- 인터페이스: 병렬 및 직렬 가능
- 변환 시작 : 하드웨어적 방법
- 내부 기준 전압 및 Hold 회로 및 입력 과전압 보호회로 내장
- 단일 5V 전원만으로도 사용 가능

Analog입력은 5V ~ 5V의 8개의 입력을 받아 아래의 Anlog mux(HI1-0508)의 선택된 출력이 2.5V ~ 2.5V입력으로 변환된 후 ADC에 연결하여 사용하며,2개의 포텐션메터와 2개의자이로 센서의 데이터를 읽어 낸다.

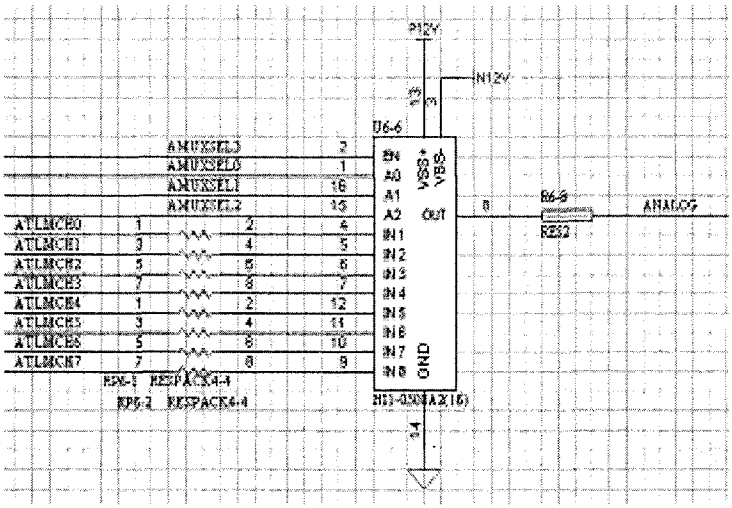


그림 3.40 Analog MUX 회로

(10) D/A 변환기 회로

D/A 변환 소자는 Analog사의 12비트 4채널 DAC8420을 사용하였고 , 기준 전압은 AD588을 사용하여 5V와 5V를 생성하여 사용하였다. D/A출력은 모터의 Speed제어 명령에 사용되었고,.상세한 D/A 변환기 회로는 아래의 그림과 같다.

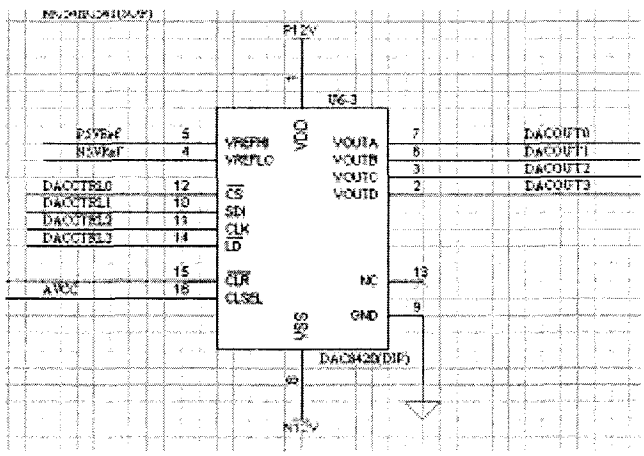


그림 3.41 D/A 변환기 회로

(11) 입출력 포트 회로

74HC541과 FPGA를 사용한 Bi-Level의 12개의 출력포트와 8개의 입력포트가 구성이 되어 있다. 출력포트의 경우는 모터 컨트롤러를 제어하여 방향을 전환하는 기능으로 사용하고 있고, 입력포트는 Gimbal제어기에서는 사용하지 않고 있다.

(12) 카메라 제어 회로

Gimbal에 사용하는 카메라를 제어하는 회로는 카메라의 Power와 Key를 입력하는 회로로 구성이 되어 있다. 카메라의 Key의 경우는 카메라에서 Line수를 줄이기 위해서 Analog입력 방식을 사용한 것 같다.

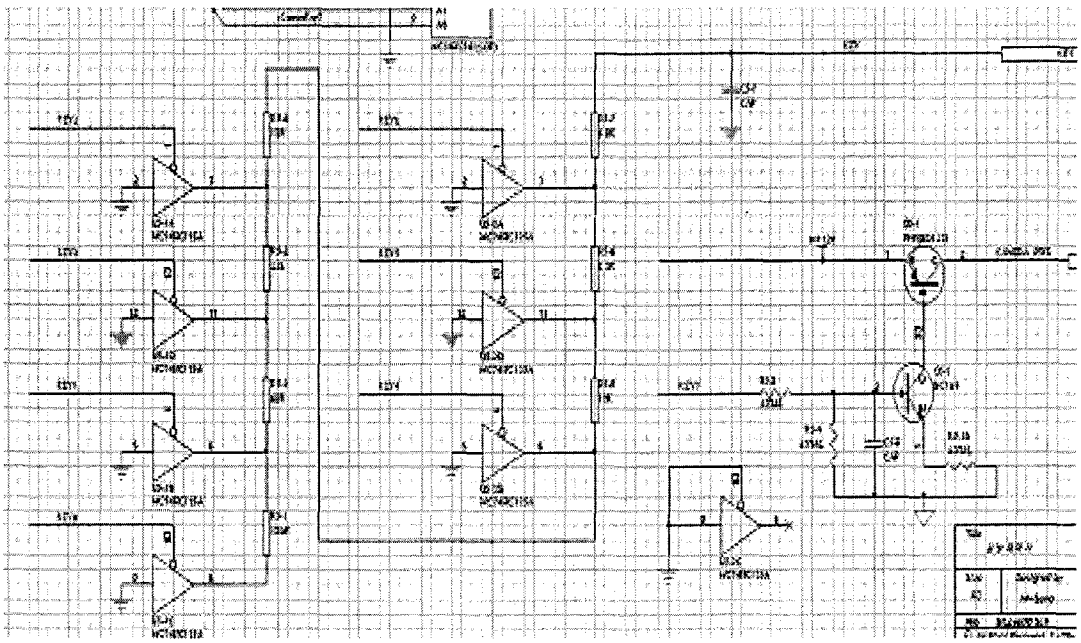


그림 3.42 카메라 제어 회로

(13) FPGA

Gimbal에 사용한FPGA는 Altera 사의 EPM7128S를 사용하였고, MAX+Plus II를 사용하여 VHDL로 coding을 하였다.

아래의 내용은 Gimbal제어기에 사용한 VHDL source code 이다.

```
-- MAX+plus II VHDL Template
-- Clearable flipflop with enable

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY Gimbal IS
        PORT
        (
                SerMODE      : IN  STD_LOGIC;

                bReset       : IN  STD_LOGIC;
                InReset      : IN  STD_LOGIC;
                WatchdogOut  : IN  STD_LOGIC;
                Reset706     : OUT STD_LOGIC;

                Clk          : IN  STD_LOGIC;
        Start        : OUT STD_LOGIC;

                bReady       : OUT STD_LOGIC;
```

```
                H1              : IN STD_LOGIC;
                H3              : IN STD_LOGIC;

                R_W             : IN STD_LOGIC;
                bSTRB0          : IN STD_LOGIC;
                bSTRB1          : IN STD_LOGIC;
        bIOSTRB      : IN STD_LOGIC;

                bRD             : OUT STD_LOGIC;
                bWR             : OUT STD_LOGIC;
                bIORD           : OUT STD_LOGIC;
                bIOWR           : OUT STD_LOGIC;

                A23             : IN  STD_LOGIC;
                A22             : IN  STD_LOGIC;
                A21             : IN  STD_LOGIC;
                A20             : IN  STD_LOGIC;

                A15             : IN  STD_LOGIC;
                A14             : IN  STD_LOGIC;
                A13             : IN  STD_LOGIC;

        Data          : IN  STD_LOGIC_VECTOR(3 downto 0);

                bCSRAM          : OUT STD_LOGIC;     -- STRB0:40 -- STRB1:90
                bCSFROM         : OUT STD_LOGIC;     -- STRB1:90
                bCSUART1        : OUT STD_LOGIC;
--              bCSSpare1         : OUT STD_LOGIC;
-- interrupt line
--              bIRQ0           : IN  STD_LOGIC;
                bIRQ1           : IN  STD_LOGIC;

                INT0            : OUT STD_LOGIC;
                INT1            : OUT STD_LOGIC;
                INT2            : OUT STD_LOGIC;
                INT3            : OUT STD_LOGIC;

-- ADC & DAC Signals
                AMUXSEL         : OUT STD_LOGIC_VECTOR(3 downto 0);
                bCONVST         : OUT STD_LOGIC;
                bEOC            : IN  STD_LOGIC;
                OutEOC          : OUT STD_LOGIC;
                bCSADC          : OUT STD_LOGIC;

                DACCTRL         : OUT STD_LOGIC_VECTOR(3 downto 0);
-- Bilevel Signals
                BiLevelOut1     : OUT STD_LOGIC_VECTOR(7 downto 0);
                BiLevelOut2     : OUT STD_LOGIC_VECTOR(3 downto 0);
                bRDBiLevel      : OUT STD_LOGIC;
                CameraKey       : OUT STD_LOGIC_VECTOR(7 downto 0)
                );
END Gimbal;
-------------------------------------------------------------------------------
--
ARCHITECTURE a OF Gimbal IS
-------------------------------------------------------------------------------
--
COMPONENT EXP
   PORT (a_in:  IN STD_LOGIC;
         a_out: OUT STD_LOGIC);
END COMPONENT;

SIGNAL   sbIOWR     : STD_LOGIC;
SIGNAL   sbIORD     : STD_LOGIC;

SIGNAL   sINT0      : STD_LOGIC;
SIGNAL   sINT1      : STD_LOGIC;
SIGNAL   sINT2      : STD_LOGIC;
SIGNAL   sINT3      : STD_LOGIC;

SIGNAL   sbCSUART1      : STD_LOGIC;
--SIGNAL   sbCSSpare1     : STD_LOGIC;
```

```
SIGNAL   sbCSBilevel1  : STD_LOGIC;
SIGNAL   sbCSCameraKey : STD_LOGIC;
SIGNAL   sbCSADC       : STD_LOGIC;
SIGNAL   sbCSAMUX      : STD_LOGIC;
SIGNAL   sbCSBilevel2  : STD_LOGIC;
SIGNAL   sbCSDAC    : STD_LOGIC;
SIGNAL   sbCSStart     : STD_LOGIC;

SIGNAL   sBANK     : STD_LOGIC_VECTOR(7 downto 0);
SIGNAL   sbBANK    : STD_LOGIC_VECTOR(7 downto 0);

SIGNAL   sAMUXSEL  : STD_LOGIC_VECTOR(3 downto 0);
SIGNAL   sbCONVST  : STD_LOGIC;
SIGNAL   sStart    : STD_LOGIC;
SIGNAL   sDACCtrl  : STD_LOGIC_VECTOR(3 downto 0);
-------------------------------------------------------------------------
--
BEGIN
-------------------------------------------------------------------------
--

bReady <= '0';

--bCSRAM     <= '0' WHEN (A23 = '1' and A22 = '0' and A21 = '0' and A20 =
'1') ELSE '1';
bCSRAM     <= '0' WHEN (A23 = '0' and A22 = '1' and A21 = '0' and A20 = '0')
ELSE '1';
bCSFROM    <= '0' WHEN (A23 = '1' and A22 = '0' and A21 = '0' and A20 = '1')
ELSE '1';

bRD    <= '0' WHEN (R_W = '1' and (bSTRB0 = '0' or bSTRB1 = '0')) ELSE '1';
bWR    <= '0' WHEN (R_W = '0' and (bSTRB0 = '0' or bSTRB1 = '0')) ELSE '1';
sbIORD <= '0' WHEN (R_W = '1' and bIOSTRB = '0') ELSE '1';
sbIOWR <= '0' WHEN (R_W = '0' and bIOSTRB = '0') ELSE '1';

bIORD  <= sbIORD;
bIOWR  <= sbIOWR;

sbCSUART1     <= '0' WHEN (A15 = '0' and A14 = '0' and A13 = '0' and bIOSTRB
= '0') ELSE '1'; -- RD & WR
sbCSBilevel1  <= '0' WHEN (A15 = '0' and A14 = '0' and A13 = '1') ELSE '1';
-- WR: Bilevel Output,RD: Bilevel Input
sbCSBilevel2  <= '0' WHEN (A15 = '0' and A14 = '1' and A13 = '0') ELSE '1';
-- WR: Bilevel Output
sbCSCameraKey <= '0' WHEN (A15 = '0' and A14 = '1' and A13 = '1') ELSE '1';
-- WR: Camera keys

sbCSAMUX      <= '0' WHEN (A15 = '1' and A14 = '0' and A13 = '0') ELSE '1';
-- WR: AMUX selection
sbCSADC       <= '0' WHEN (A15 = '1' and A14 = '0' and A13 = '1') ELSE '1';
-- RD: ADC Read,WR:Convert Start
sbCSDAC       <= '0' WHEN (A15 = '1' and A14 = '1' and A13 = '0') ELSE '1';
-- WR: DAC ctrl
sbCSStart     <= '0' WHEN (A15 = '1' and A14 = '1' and A13 = '1') ELSE '1';
-- WR:

bCSUART1      <= sbCSUART1;

bRDBilevel    <= '0' WHEN(sbIORD = '0' and sbCSBilevel1 = '0') ELSE '1';

-- Create Bank Select Signal
sBANK(0) <= '1' WHEN (Data(3) = '0' and Data(2) = '0' and Data(1) = '0') ELSE
'0';
sBANK(1) <= '1' WHEN (Data(3) = '0' and Data(2) = '0' and Data(1) = '1') ELSE
'0';
sBANK(2) <= '1' WHEN (Data(3) = '0' and Data(2) = '1' and Data(1) = '0') ELSE
'0';
sBANK(3) <= '1' WHEN (Data(3) = '0' and Data(2) = '1' and Data(1) = '1') ELSE
'0';
sBANK(4) <= '1' WHEN (Data(3) = '1' and Data(2) = '0' and Data(1) = '0') ELSE
'0';
sBANK(5) <= '1' WHEN (Data(3) = '1' and Data(2) = '0' and Data(1) = '1') ELSE
```

```
'0';
sBANK(6) <= '1' WHEN (Data(3) = '1' and Data(2) = '1' and Data(1) = '0') ELSE
'0';
sBANK(7) <= '1' WHEN (Data(3) = '1' and Data(2) = '1' and Data(1) = '1') ELSE
'0';

lBANK0:  EXP port map(sBANK(0),sbBANK(0));  -- sbBANK(0) <= not(sBANK(0);
lBANK1:  EXP port map(sBANK(1),sbBANK(1));  -- sbBANK(1) <= not(sBANK(1);
lBANK2:  EXP port map(sBANK(2),sbBANK(2));  -- sbBANK(2) <= not(sBANK(2);
lBANK3:  EXP port map(sBANK(3),sbBANK(3));  -- sbBANK(3) <= not(sBANK(3);
lBANK4:  EXP port map(sBANK(4),sbBANK(4));  -- sbBANK(4) <= not(sBANK(4);
lBANK5:  EXP port map(sBANK(5),sbBANK(5));  -- sbBANK(5) <= not(sBANK(5);
lBANK6:  EXP port map(sBANK(6),sbBANK(6));  -- sbBANK(6) <= not(sBANK(6);
lBANK7:  EXP port map(sBANK(7),sbBANK(7));  -- sbBANK(7) <= not(sBANK(7);

PROCESS(bReset,sbCSStart,sbIOWR)
BEGIN
        if(bReset = '0') then
                sStart <= '0';
        elsif (sbCSStart = '1') then
                null;
        elsif(sbIOWR'event and sbIOWR = '1') then
                sStart <= '1';
        end if;
END PROCESS;

Start <= sStart;

PROCESS(sStart,InReset,WatchdogOut)
BEGIN
        if(sStart = '0') then
                Reset706 <= InReset;
        else
                Reset706 <= (WatchdogOut and InReset);
        end if;
END PROCESS;

-- interrupt line
PROCESS(bReset,H1,bIRQ1,sStart)   -- UART1 interrupt
BEGIN
        if(bReset = '0') then
         sINT0 <= '0';
        elsif (sStart = '0') then
                null;
        elsif (H1'event and H1 = '1') then
         sINT0 <= not(bIRQ1);
        end if;
END PROCESS;
INT0 <= not(sINT0);

--PROCESS(bReset,H1,bIRQ2,sStart)  -- UART2 interrupt
--BEGIN
--      if(bReset = '0') then
--         sINT1 <= '0';
--      elsif (sStart = '0') then
--              null;
--      elsif (H1'event and H1 = '1') then
--         sINT1 <= not(bIRQ2);
--      end if;
--END PROCESS;
--INT1 <= not(sINT1);
INT1 <= '1';

PROCESS(bReset,H1,SerMODE,sStart)        -- boot3 0x900000
BEGIN
        if(bReset = '0') then
                sINT2 <= '0';
        elsif(H1'event and H1 = '1') then
                if(SerMode = '0' and sStart = '0') then
                        sINT2 <= '1';
                else
                        sINT2 <= '0';
```

```
                end if;
        end if;
END PROCESS;
INT2 <= not(sINT2);

PROCESS(bReset,H1,SerMODE,sStart)          -- serial boot
BEGIN
        if(bReset = '0') then
                sINT3 <= '0';
        elsif(H1'event and H1 = '1') then
            if(SerMODE = '1' and sStart = '0') then
                        sINT3 <= '1';
         else
                        sINT3 <= '0';
                end if;
        end if;
END PROCESS;
INT3 <= not(sINT3);

-- ADC signal
PROCESS(bReset,sbCSAMUX,sbIOWR,Data)
BEGIN
        if(bReset = '0') then
                sAMUXSEL <= "0000";
        elsif(sbCSAMUX = '1') then
                null;
        elsif(sbIOWR'event and sbIOWR = '1') then
                sAMUXSEL <= Data;
        end if;
END PROCESS;

AMUXSEL <= sAMUXSEL;

bCSADC   <= '0' WHEN (sbCSADC = '0' and sbIORD = '0') ELSE '1';
sbCONVST <= '0' WHEN (sbCSADC = '0' and sbIOWR = '0') ELSE '1';
bCONVST  <= sbCONVST;

PROCESS(bReset,bEOC,H1,sbCONVST,bIOSTRB)
BEGIN
        if(bReset = '0') then
                OutEOC <= '0';
        elsif(bEOC = '0') then
                OutEOC <= '1';
        elsif (H1'event and H1 = '1') then
                if(sbCONVST = '0' and bIOSTRB = '0') then
                        OutEOC <= '0';
                end if;
        end if;
END PROCESS;

-- DAC ctrl
PROCESS(bReset,sbIOWR,sbCSDAC,sbBANK,Data)
BEGIN
        if(bReset = '0') then
                sDACCtrl <= "0000";
        elsif(sbIOWR'event and sbIOWR = '1') then
                FOR i IN 0 to 3 LOOP
                        if(sbCSDAC = '0' and sbBANK(i) = '0') then
                                sDACCtrl(i) <= Data(0);
                        end if;
                END LOOP;
        end if;
END PROCESS;
DACCTRL <= sDACCtrl;

-------------------------------------------------------------------------------
--
PROCESS(bReset,sbIOWR,sbCSBilevel1,sbBANK,Data(0))
BEGIN
        if(bReset = '0') then
                BiLevelOut1 <= "00000000";
        elsif(sbIOWR'event and sbIOWR = '1') then
```

```
                FOR i IN 0 to 7 LOOP
                        if(sbCSBilevel1 = '0' and sbBANK(i) = '0') then
                                BiLevelOut1(i) <= Data(0);
                        end if;
                END LOOP;
        end if;
END PROCESS;

PROCESS(bReset,sbIOWR,sbCSBilevel2,sbBANK,Data(0))
BEGIN
        if(bReset = '0') then
                BiLevelOut2 <= "0000";
        elsif(sbIOWR'event and sbIOWR = '1') then
                FOR i IN 0 to 3 LOOP
                        if(sbCSBilevel2 = '0' and sbBANK(i) = '0') then
                                BiLevelOut2(i) <= Data(0);
                        end if;
                END LOOP;
        end if;
END PROCESS;

PROCESS(bReset,sbIOWR,sbCSCameraKey,sbBANK,Data(0))
BEGIN
        if(bReset = '0') then
                CameraKey <= "00000000";
        elsif(sbIOWR'event and sbIOWR = '1') then
                FOR i IN 0 to 7 LOOP
                        if(sbCSCameraKey = '0' and sbBANK(i) = '0') then
                                CameraKey(i) <= Data(0);
                        end if;
                END LOOP;
        end if;
END PROCESS;

END a
```

(14) 개발된 하드웨어

1단계 연구 결과로 얻어진 Gimbal 제어기 프로세서 모듈은 그림 3.43에 나타나 있다. 모듈의 크기는 220×150×35 ㎜ 이다.

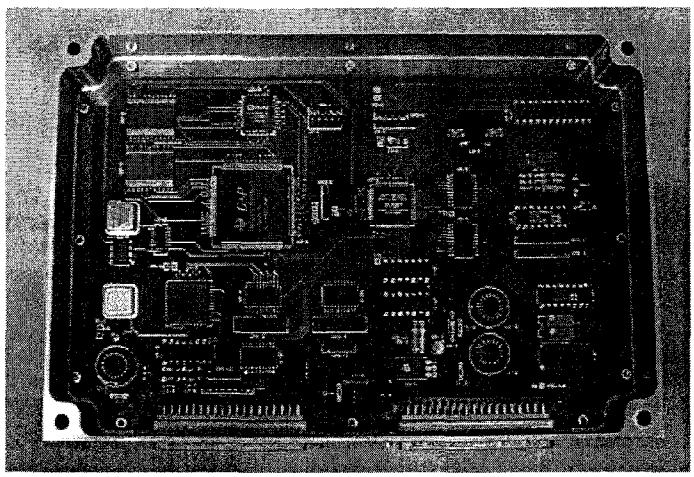


그림 3.43 개발된 제어기 프로세서 모듈

나. 전력 분배기(PDM: Power Distribution Module)

PDM에서 전력을 공급해야 할 장치는 제어기 프로세서 보드, 영상 센서 및 포텐셔미터, 자이로 등의 센서들이다. 전력 분배기에 요구되는 입출력 규격은 다음과 같다.

- 입력: +28V의 unregulated input
- 출력: +5VDC, +12VDC, -12VDC

설계를 간단히 하기 위하여 PDM은 다음과 같이 충분한 용량의 3개의 DC/DC converter를 사용하였다.

표 3.5 PDM에 사용된 DC/DC converter

출력 전압	소자명	최대 전류	용량
+5V	PS10-24-5	2A	10W
+12V	PS25-24-12	1.5A	18W
-12V	PS6-24-12	0.5A	6W
계			34W

1단계 연구 결과로 얻어진 PDM은 그림 3.44에 나타나 있다. 모듈의 크기는 제어기 프로세서와 동일한 220×150×35 mm 이다.

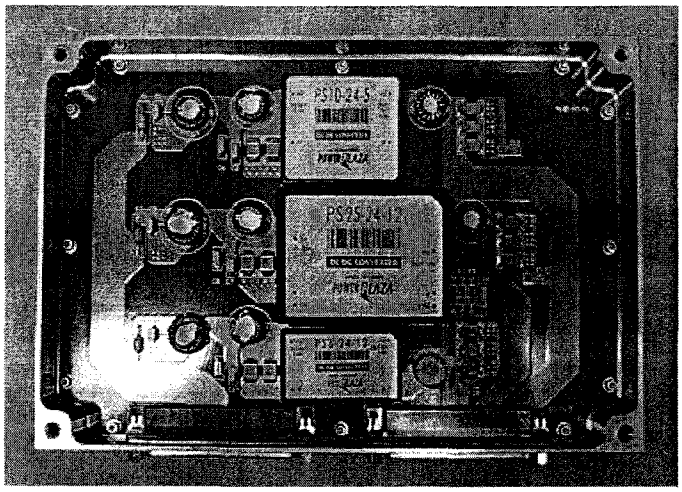


그림 3.44 개발된 전력 분배기

6. 모터 드라이버

가. 요 축 DC 모터 및 모터 드라이버

요 축 모터는 그림 3.45에 나타나 있는 바와 같은 Maxon사의 EC22 Brushless Motor를 사용하였다. 이 모터의 주요 특성치는 다음과 같다.

- Power : 50 W
- Nominal Voltage : 32 V
- Stall Torque : 400 mNm
- Torque Constant : 13.6 mNm/A
- Rotor Inertia : 4.2 × 10-7 Kg m2
- Mechanical Time Constant : 2.5 ms

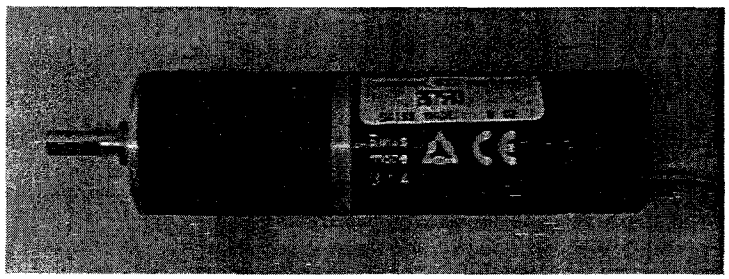


그림 3.45 MAXON EC 22 Brushless Motor

그림 3.46는 EC Motor 드라이버인 DEC50/5의 모습을 보여주고 있다.

그림 3.46 요 축 모터 드라이버 (Maxon 1-Q-EQ Amp. DEC50/5)

나. 피치 축 스테핑 모터 및 모터 드라이버

피치 축에는 최근의 발달한 스테핑 모터 기술에 근거하여 Oriental Motors사의 CFKII 시리즈를 적용하였다. 이 모델은 반응 속도가 매우 빠르고 본 과제에서 개발된 김발을 구동하기에 충분한 토크를 제공하며 특히 Microstep을 적용할 경우 최대 0.00288o(~50mrad)의 step 해상도를 가지므로 100 mrad의 요구사항을 만족시키기 위한 충분한 각 분해능을 제공한다. 사용한 543AT모델의 주요 성능은 다음과 같다.

- Maximum Holding Torque : 0.13 Nm
- Rotor Inertia : 35 × 10-7 Kg m2
- Rated Current : 0.75 A/phase
- Basic Step Angle : 0.72o
- Micro-step Capability : Maximum 250 divisions
- Weight : 210 g

그림 3.47은 CKFII543AT 모터의 모습을 보여주고 있고 그림 3.48에는 CKFII543AT의 토크 특성 곡선을 보여주고 있다.

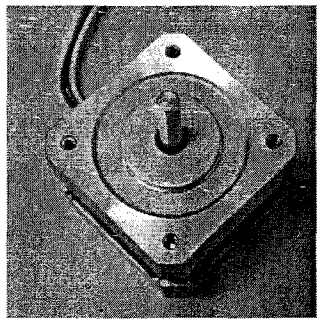


그림 3.47 CFKII543AT Micro-Stepping 모터

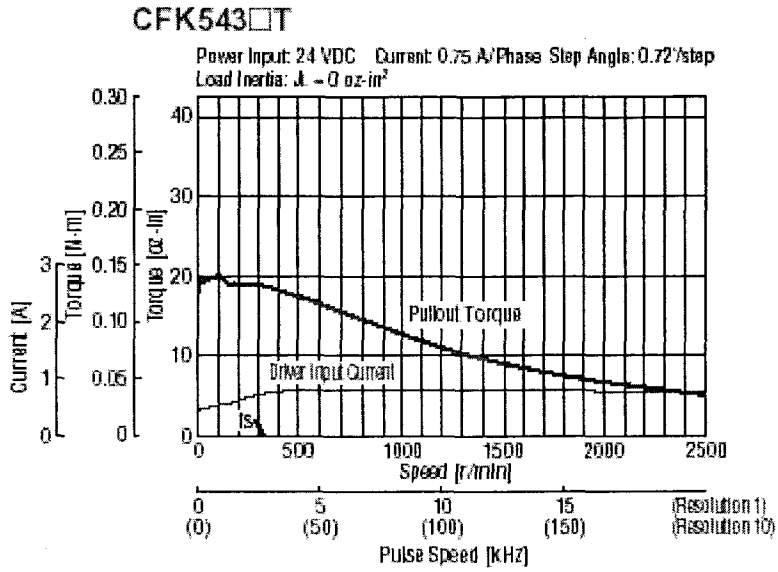


그림 3.48 CKFII543의 토크 특성 곡선

그림 3.49는 CFKII용 모터 드라이버인 DFC5107P의 모습을 보여주고 있다

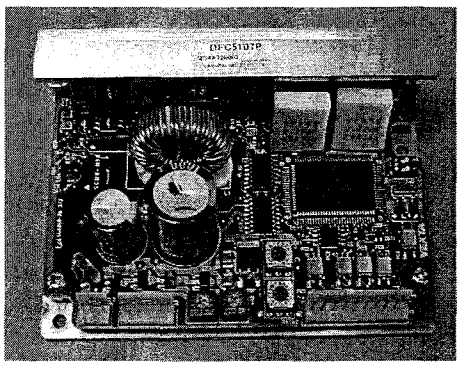


그림 3.49 DFC5107P 모터 드라이버

다. 모터 드라이버 하우징

앞서 소개된 Gimbal 제어기 프로세서 보드 및 PDM의 하우징에 맞도록 모터 드라이버들을 위한 하우징을 따로 개발하였다. 그림 3.50에는 세 모듈의 하우징을 스택 형태로 쌓아올린 모습을 보여주고 있다. 전자부의 크기는 체결나사의 높이를 포함하여 220×150×124 mm 이다.

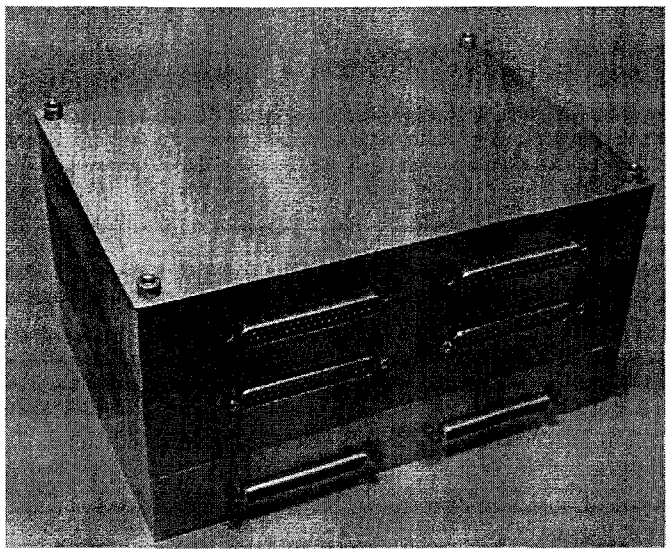


그림 3.50 조립된 제어기 전자부 하우징

7. 제어용 소프트웨어

가. 제어용 PC 소프트웨어

Gimbal 제어기를 제어하는 프로그램은 Visual C++를 사용하여 작성되었고, 기능은 크게 카메라를 제어하는 기능과 모터를 제어하는 기능과 센서 데이터의 현재 상태를 표시해 주는 부분으로 되어 있다.

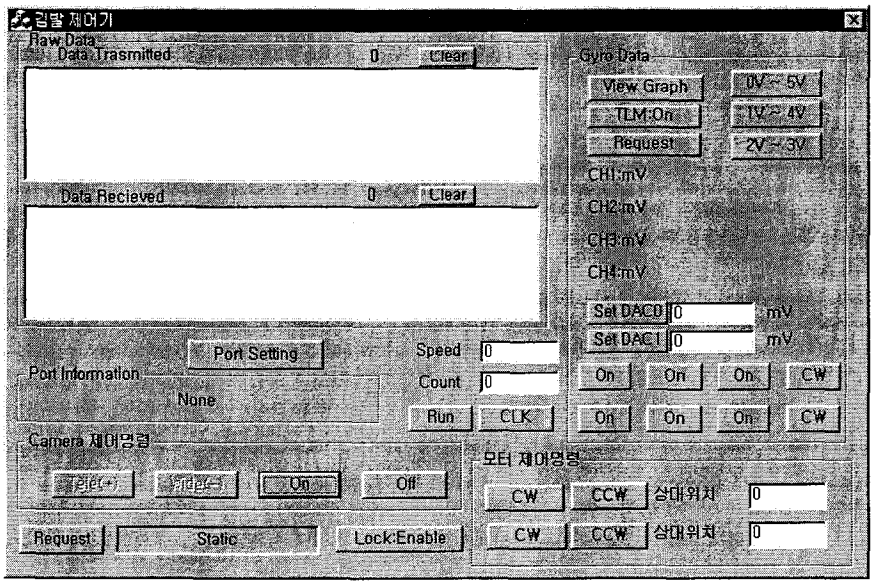


그림 3.51 Gimbal 제어기초기화면

Gimbal 제어용 소프트웨어를 실행을 시키면 그림 3.51과 같은 화면이 표시된다. 우선 Gimbal제어기와 통신을 하기위해서는 화면의 중앙에 있는 "Port Setting"버튼을 누르면 그림 3.52와 같은 통신 포트의 설정 윈도우가 나타난다. 우선 통신 속도를 19200Bps로 설정 하고, Gimbal제어기가 연결되어 있는 COM port, Data Bits, Parity, Stop Bits를 설정 후 "Ok"버튼을 누르면 된다.

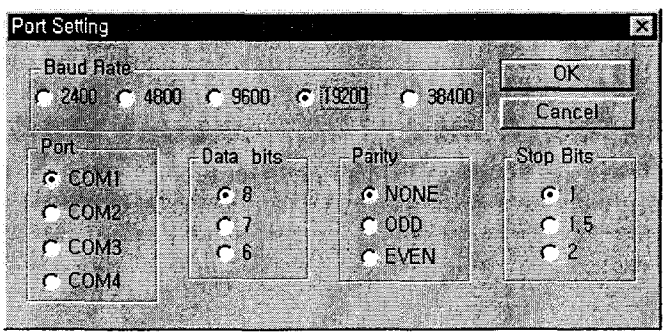


그림 3.52 Serial port 설정 Box

통신포트의 설정이 정상적으로 설정이 된 경우에는 Port Setting명령 단추아래에 설정된 Port의 내용이 아래와 같이 표시된다.

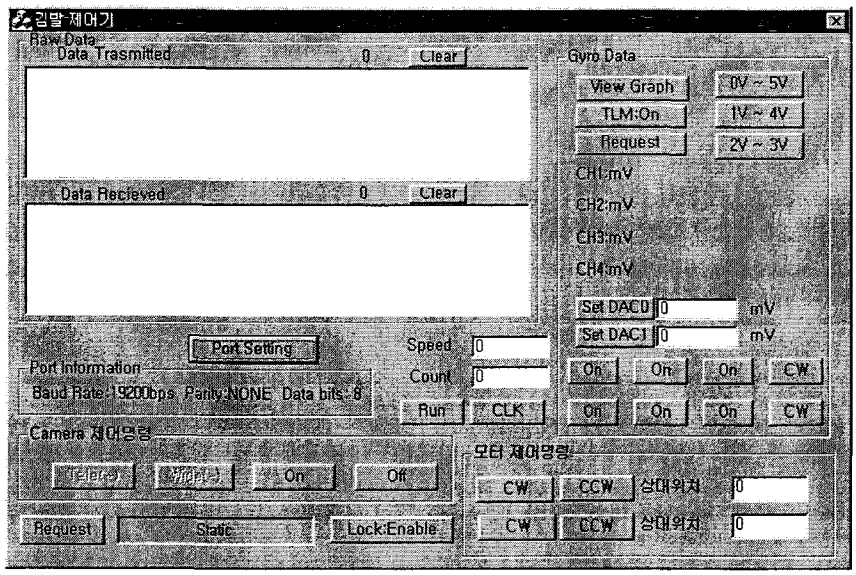


그림 3.53 Port 설정 상태

화면의 상단 좌측의 "Data Transmitted"와 "Data Received" 윈도우는 RS-232를 통해서 제어기와 컴퓨터 간에 통신한 데이터를 Hex data로 표시해준다.

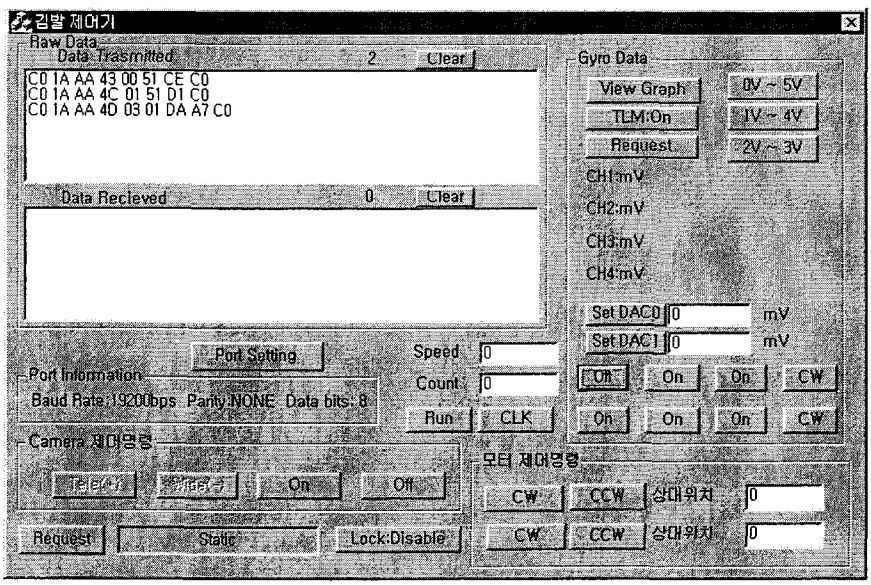


그림 3.54 송수신 데이터 표시 윈도우

Camera 제어명령 Box에 있는 명령들은 카메라의 파워를 On/Off하는 명령과 카메라의 Zoom In/Out하는 명령으로 되어 있다. Tele(+)와 Wide(-)명령 버튼의 경우는 마우스의 왼쪽 버튼을 누르는 경우와 버튼을 놓는 경우에 명령이 전송하도록 되어 있어, 카메라의 출력 화면을 보면서 Zoom In/Out을 컨트롤해야 한다.

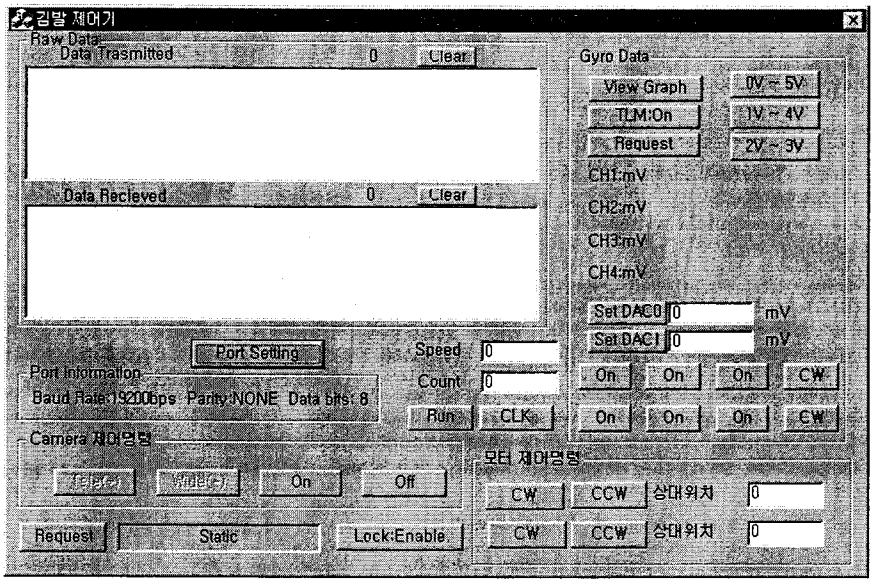


그림 3.55 카메라 제어 명령

아래의 그림은 모터를 제어하는 명령들로 모터의 회전 방향과 위치를 입력하는 부분으로 되어 있다.

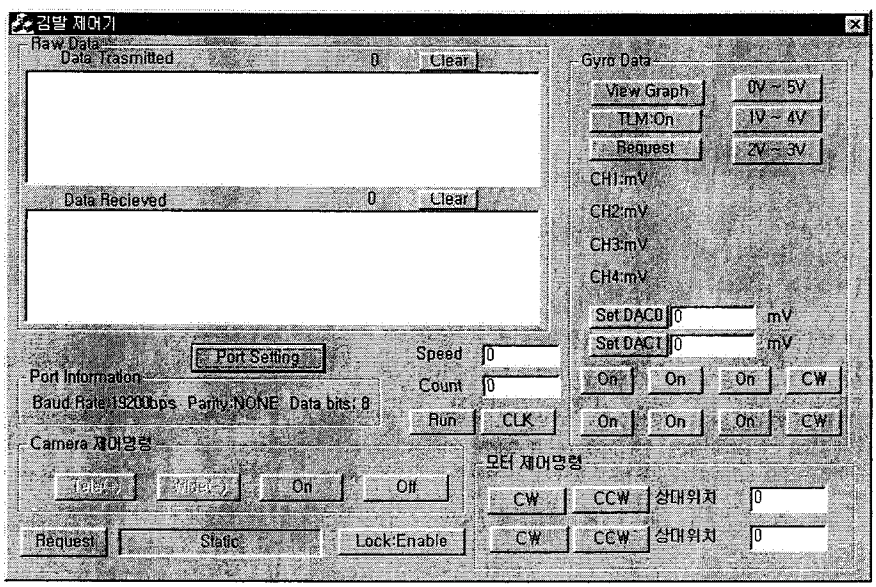


그림 3.56 모터제어 명령

화면의 상단 우측에 있는 Gyro data 부는 Gimbal에 연결되어 있는 자이로와 포텐셔미터의 상태를 표시해주는 부분이다.

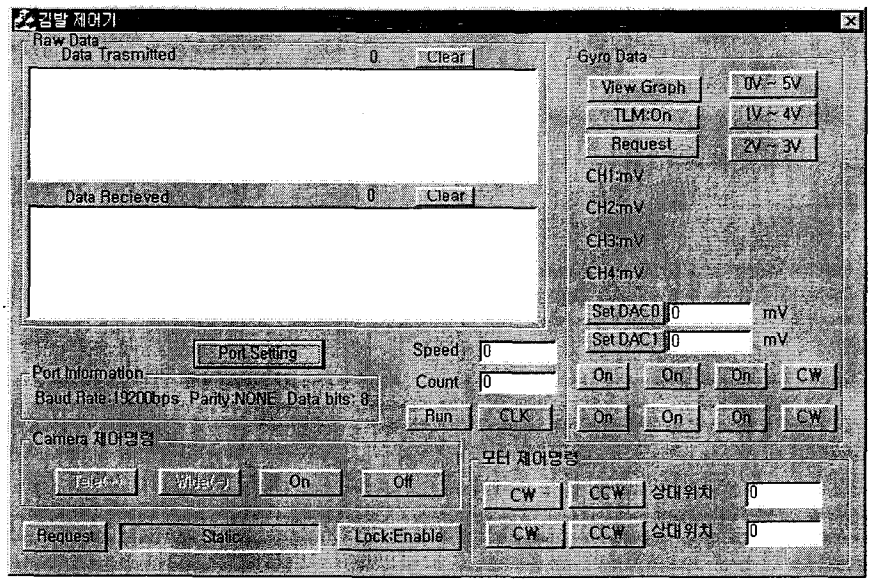


그림 3.57 센서데이터 표시부

"View Graph" 명령 버튼은 센서 데이터를 Graph형태로 보여주는 윈도우가 나타난다.

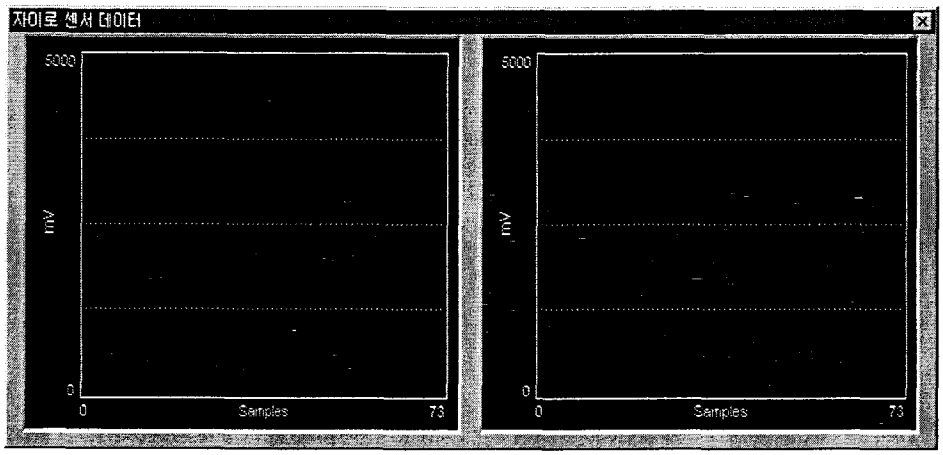


그림 3.58 자이로 데이터 표시 윈도우

나. 통신 프로토콜

Gimbal 제어기와 제어 컴퓨터 간의 RS-232 통신프로토콜은 RFC-1055의 SLIP프로토콜을 사용하며 Gimbal 제어기의 통신 규약을 간단히 정리하면 아래와 같다.

- 통신 방식: UART
- Data : 8Bits
- Start Bit: 1bit
- Stop Bit: 1bit
- Parity: None
- Speed: 9600bps

SLIP 프로토콜은 단지 패킷 화에 관한 것만이 정리되어 있다. 따라서 패킷을 표 3.5와 같은 구조로 정의하여 사용하며, 데이터의 통신상에 문제가 있을 경우 패킷의 정합성을 판단하기 위해서 체크코드로 CCITT-16를 사용한다.

표 3.5 통신제어기간 패킷 형식

패킷 시작	Address	FT	INFO	CCITT-16	패킷 종료
1Byte(0xC0)	2Byte	1Byte	0 ~ 128 Bytes	2Bytes	1Byte(0xC0)

8. Gimbal 제어기 시뮬레이션 소프트웨어

가. 시뮬레이션 소프트웨어의 개요

시뮬레이션 소프트웨어는 사용자 편의 인터페이스(GUI)를 기반으로 한 Windows 소프트웨어로써 Microsoft Visual C++ Ver. 6.0을 사용하여 개발되었다. 그림 3.59은 시뮬레이션 소프트웨어가 실행되고 있는 모습을 보여주고 있고 그림 3.60에는 시뮬레이션 소프트웨어의 구조가 나타나 있다.

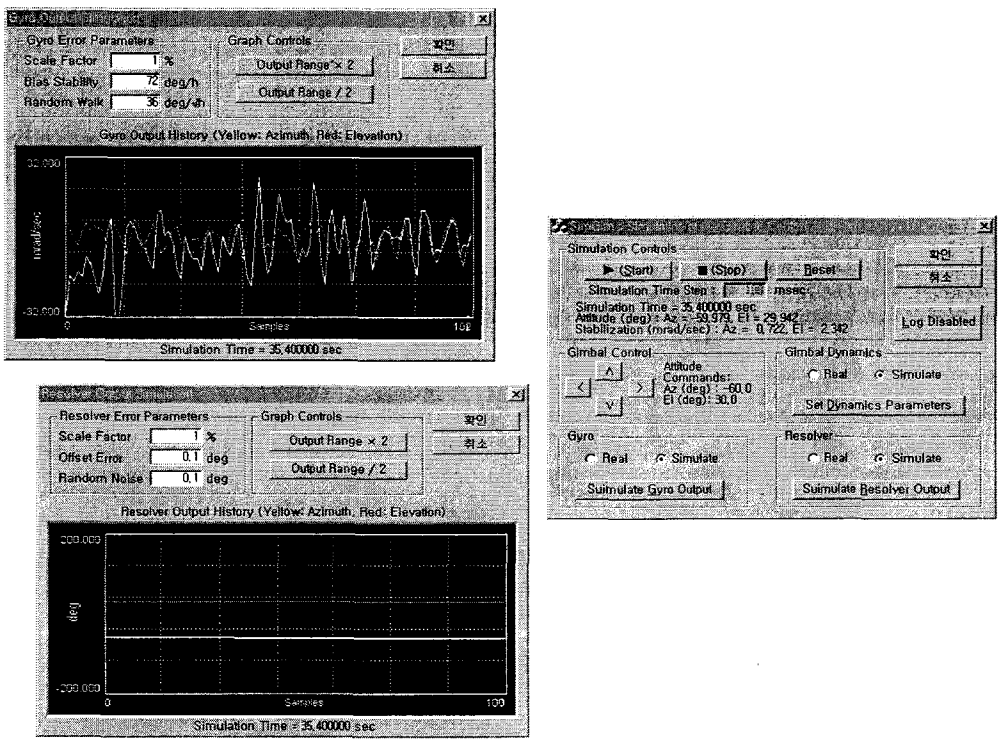


그림 3.59 시뮬레이션 소프트웨어의 실행 모습

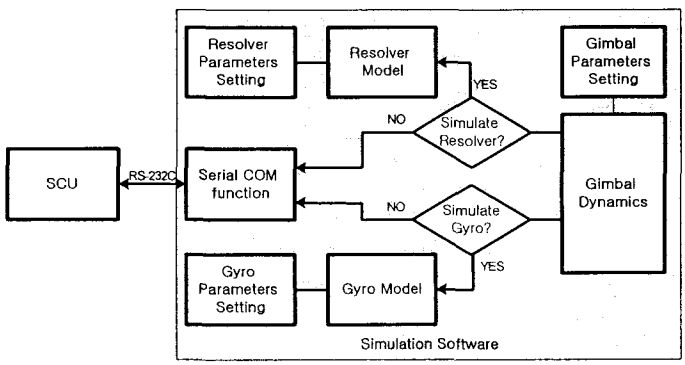


그림 3.60 시뮬레이션 소프트웨어의 구조

앞 그림에서 볼 수 있는 바와 같이 시뮬레이션 소프트웨어는 Gimbal 운동 방정식의 시뮬레이션 뿐 아니라 자이로 및 각 측정 센서의 모델도 함께 포함하고 있다. 이는 센서가 갖추어지지 않은 상황에서의 Gimbal 제어기와의 통신이 무의미하기 때문이기도 하지만 반대로 안정화 요구조건을 만족하기 위해 요구되는 센서의 성능을 결정하는데 사용하기 편리한 형태이기 때문이기도 하다.

나. 센서 모델

Gimbal에는 각 축마다 각속도 자이로가 부착되고 이 자이로들은 Gimbal의 2축 각속도 측정값을 제공한다. 자이로의 경우 실제 값에 단순히 white noise를 더하여 모델링 하는 것이 일반적이지만 실제로는 drift 및 scale factor등의 에러 요인을 고려하여 이 값들을 시험을 통해 파악하거나 칼만 필터를 통해 추정해 사용해 주어야 한다. 자이로의 측정식은 다음과 같이 표현된다.

$$\omega_\varphi = (1+\lambda_\varphi)(\dot{\varphi}_I + b_\varphi) + w_\varphi$$
$$\omega_\theta = (1+\lambda_\theta)(\dot{\theta}_I + b_\theta) + w_\theta \qquad (15)$$

여기에서 $\lambda_\varphi, \lambda_\theta$는 scale factor, b_φ, b_θ는 drift에 의한 bias이고 w_φ, w_θ는 random walk noise이다.

다. 시뮬레이션 소프트웨어 사용법

(1) 소프트웨어의 시동

시뮬레이션 소프트웨어를 실행시키면 다음과 같은 메인 프로그램이 나타나게 된다.

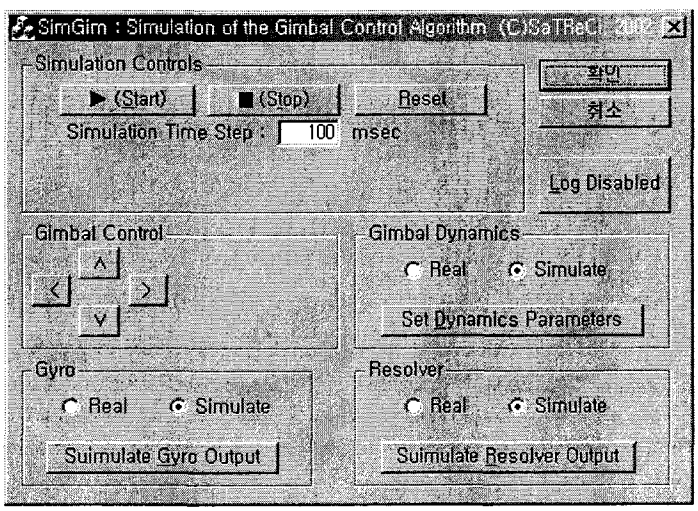


그림 3.61 메인 프로그램의 모습

메인 프로그램은 시뮬레이션의 시작 및 멈춤, 그리고 reset등을 수행할 수 있다.

또한 자이로 및 포텐셔미터 등의 센서 및 Gimbal 운동 방정식의 파라미터들을 설정할 수 있는 윈도우를 불러 올 수 있다.

(2) Gimbal 운동 방정식 파라미터 설정

Gimbal의 운동 방정식 파라미터들은 메인 프로그램의 Set Dynamics Parameters 버튼을 눌러 설정할 수 있다. 이 버튼을 누르면 다음 그림 3.62와 같은 대화 상자가 나타나게 된다.

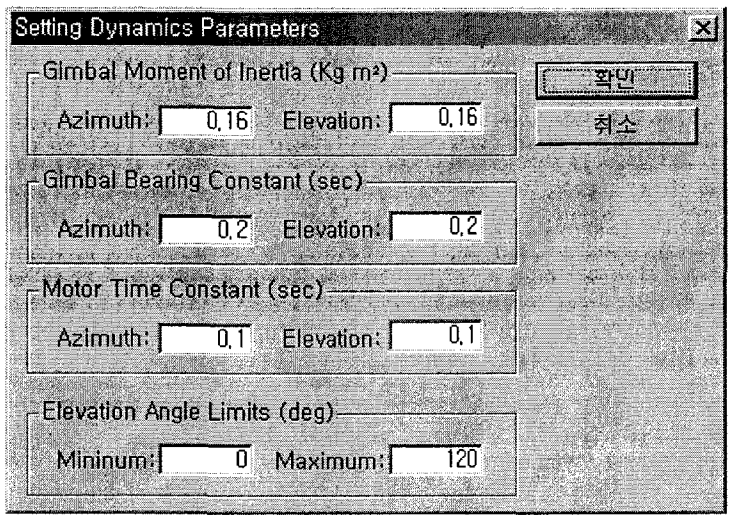


그림 3.62 Gimbal 운동방정식 파라미터 설정 대화 상자

이 대화상자에서는 Gimbal 운동방정식을 시뮬레이션하기 위해 필요한 Gimbal의 관성 모멘트, 베어링의 마찰 계수, 모터의 시 상수 및 elevation angle의 한계 값 등을 설정할 수 있다.

(3) Gimbal 운동 방정식 파라미터 설정과 자이로 출력 시뮬레이션

메인 프로그램의 Simulate 자이로 Output 버튼을 누르면 다음의 그림 3.63과 같은 대화 상자가 나타난다.

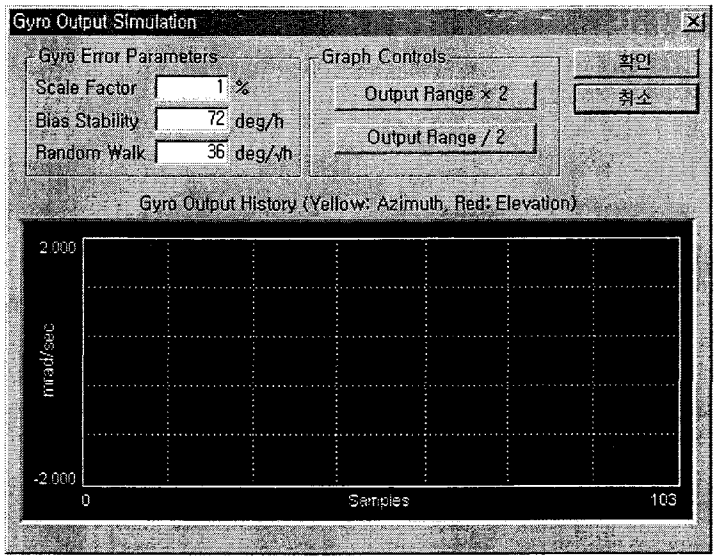


그림 3.63 자이로 출력 시뮬레이션 대화 상자

이 대화 상자는 자이로 오차 파라미터들, 즉 scale factor, bias 및 랜덤 잡음 등의 크기를 설정할 수 있고 시뮬레이션에 의해 생성되는 자이로 출력 값을 그래프로 볼 수 있다. Y축의 자이로 출력 값은 mrad/sec 의 단위로 설정되어 있으며 출력치를 보다 정확히 보기 위해 출력 범위를 Output Range × 2 및 Output Range / 2 버튼을 사용해 조절할 수 있다.

(4) 포텐셔미터 출력 시뮬레이션

메인 프로그램의 Simulate Resolver Output 버튼을 누르면 다음의 그림 3.64와 같은 대화 상자가 나타난다.

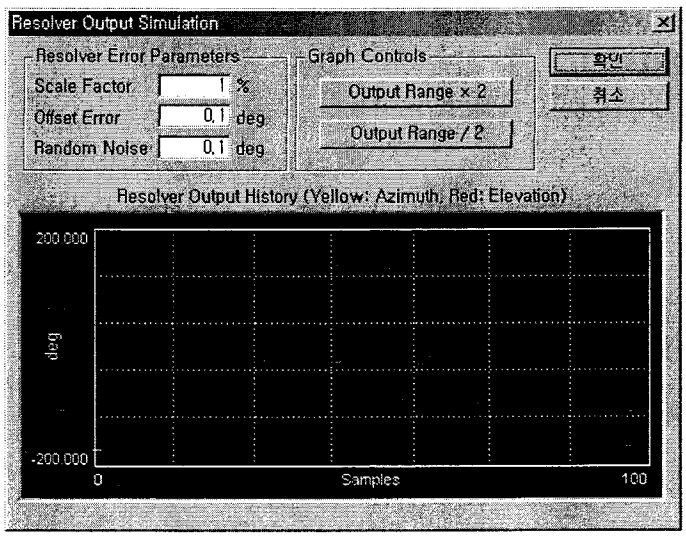


그림 3.64 Resolver 출력 시뮬레이션 대화 상자

이 대화 상자는 Resolver 오차 파라미터들, 즉 scale factor, Offset 에러 및 랜덤 잡음 등의 크기를 설정할 수 있고 시뮬레이션에 의해 생성되는 Resolver 출력 값을 그래프로 볼 수 있다. Y축의 Resolver 출력 값은 deg 의 단위로 설정되어 있으며 출력치를 보다 정확히 보기 위해 출력 범위를 Output Range × 2 및 Output Range / 2 버튼을 사용해 조절할 수 있다.

(5) 시뮬레이션의 수행

이상의 Gimbal의 운동식 및 자이로, resolver 등의 센서 파라미터의 설정이 완료되면 시뮬레이션을 수행할 수 있다. 시뮬레이션을 수행할 때의 적분 간격은 Simulation Time Step에서 msec 단위로 되어 있는 값을 입력 시켜 설정할 수 있다. 기본 값은 100 msec이다. 메인 프로그램의 ▶ (Start) 버튼을 누르면 시뮬레이션이 수행되며 아래 그림 3.65와 같이 시뮬레이션이 수행된 시간 및 그 시간에서의 요 축 및 피치 축 각과 각속도 값이 표시된다.

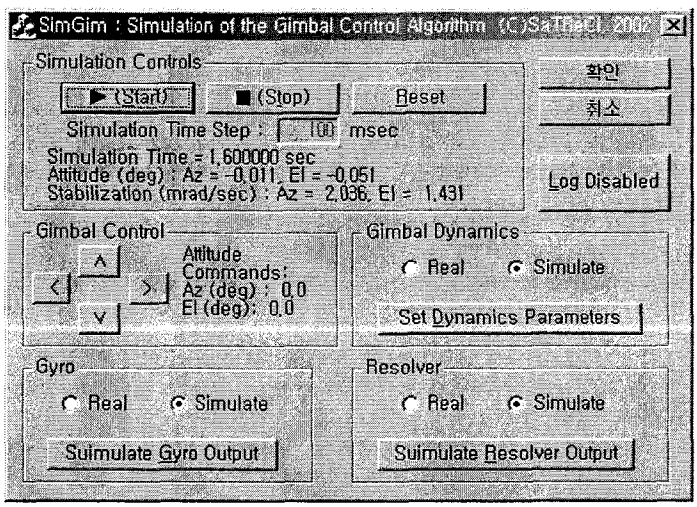


그림 3.65 시뮬레이션 수행 모습

시뮬레이션이 수행되는 도중에는 적분 간격을 변화시키지 않기 위해 Simulation Time Step의 입력 text box는 disable된다.

시뮬레이션을 잠시 멈추려면 ■ (Stop) 버튼을 누르면 된다. 이때 다시 적분 간격을 변화시킬 수 있도록 Simulation Time Step text box는 다시 enable된다. 한편 시뮬레이션을 다시 처음부터 시작하고 싶을 때에는 Reset 버튼을 눌러주면 된다. 이때 메인 프로그램은 다음과 같이 초기화된다.

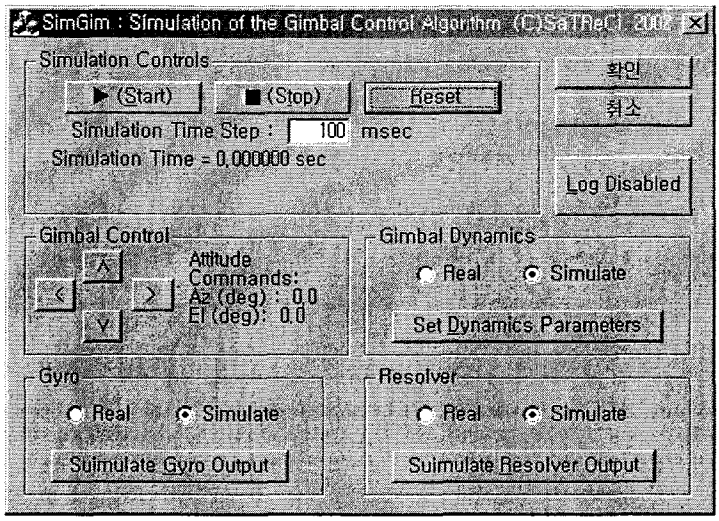


그림 3.66 Reset으로 초기화된 메인 프로그램

Reset기능은 시뮬레이션이 수행되는 도중에도 수행 가능하다.

(6) Gimbal 자세각 명령 입력 기능

Gimbal 자세각 명령은 기본적으로 조이 스틱과 같은 Gimbal 조종기를 통해 이루어져야 하지만 현재 단계에서는 버튼을 사용해 1도 단위로 설정할 수 있도록 하였다.

Gimbal Control의 < > 버튼은 요 축의 자세각 명령을 각각 감소 및 증가시키기 위해 사용하고 ∧ ∨ 버튼은 피치 축의 자세각 명령을 각각 감소 및 증가시키기 위해 사용한다. 각의 증가 혹은 감소 단위는 1도이다. 각 명령을 변경하면 자이로 및 resolver 출력 값이 명령을 따라 변화하는 것을 볼 수 있다. 이를 통해 제어기의 성능을 평가하게 된다.

(7) Log 파일의 생성

Matlab과 같은 다른 프로그램을 사용하여 Gimbal 운동 방정식을 분석하기 위해서는 Gimbal 상태 변수 값을 별도의 텍스트 파일로 출력할 필요가 있다. 메인 프로그램의 Log Diabled 버튼은 이러한 목적으로 사용할 수 있다.

이 버튼은 기본값이 Log Disabled로 설정되어 있다. 이는 시뮬레이션이 수행되더라도 log가 생성되지 않는다는 의미이다. 이 버튼을 누르면 다음의 와 같이 버튼의 text가 Log Enabled로 바뀌고 이때부터는 log가 생성되기 시작한다.

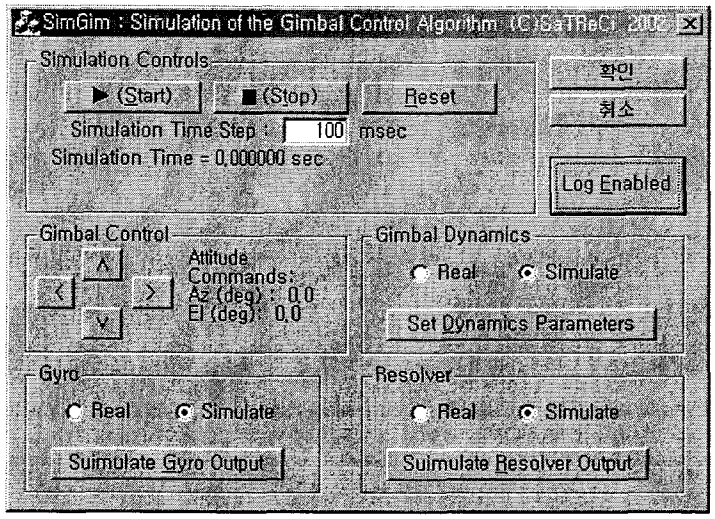


그림 4.40 Log Enabled의 모습

이 버튼을 다시 누르게 되면 버튼의 text가 다시 Log Disabled로 바뀌고 이때부터는 시뮬레이션 결과가 log 파일에 저장되지 않게 된다. Log 파일의 이름은 Gimbal.log이다.

라. 시뮬레이션 소프트웨어를 사용한 Gimbal 제어기 성능 검증

(1) Gimbal 동역학 파라미터 및 센서 파라미터의 설정

시뮬레이션을 위한 Gimbal의 동역학 파라미터는 다음과 같이 설정하였다.

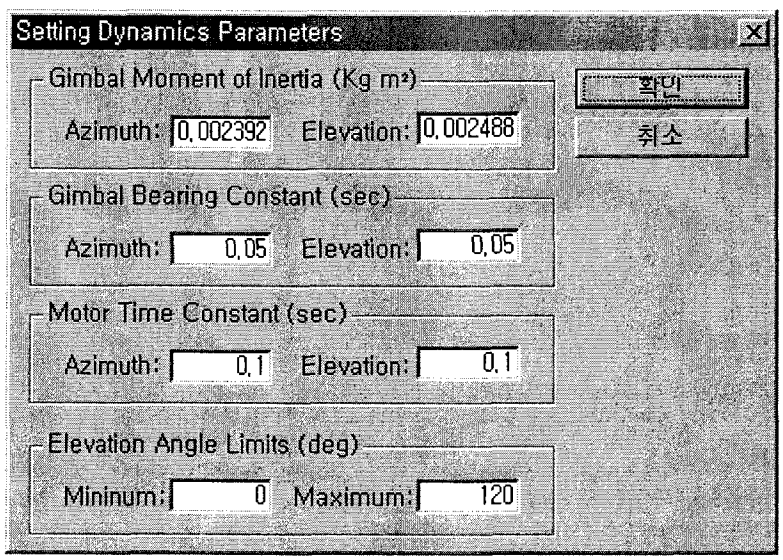


그림 3.67 Gimbal의 동역학 파라미터 설정

자이로와 포텐셔미터의 오차 에러값은 다음 표와 같이 설정하였다.

표 3.6 자이로 및 포텐셔미터 오차 값 설정

항목	값
자이로 scale factor	1 %
자이로 drift	72 deg/h
자이로 랜덤 노이즈	35 deg/$\sqrt{h}$
포텐셔미터 scale factor 오차	1 %

(2) 폐회로 극점의 위치에 따른 성능

극점 위치 기법에 따라 PD 제어기의 이득 값들은 (13)식 및 (14)식에 의해 결정할 수 있다. 빠른 응답을 원하기 때문에 폐회로 극점의 대역폭을 5 Hz로 정하고 다른 damping ratio에 대해 시뮬레이션을 수행하였다. 고려된 damping ratio와 그에 따른 이득 값들은 다음의 표와 같다.

표 3.7 Damping ratio에 따른 이득 값

ζ	$K_{P\varphi}$	$K_{D\varphi}$	$K_{P\theta}$	$K_{D\theta}$
1.0	1.1960	0.0589	1.2440	0.0622
0.8	1.3156	0.0024	1.3684	0.0025
0.5	1.4950	-0.1196	1.5550	-0.1244

요 각 명령은 60도, 피치 각 명령은 30도의 상수로 주었고 명령 생성 주기는 100 Hz로 정했다. 각 축의 명령 추종 성능을 비교한 것은 그림 3.68과 그림 3.69에 주어져 있다.

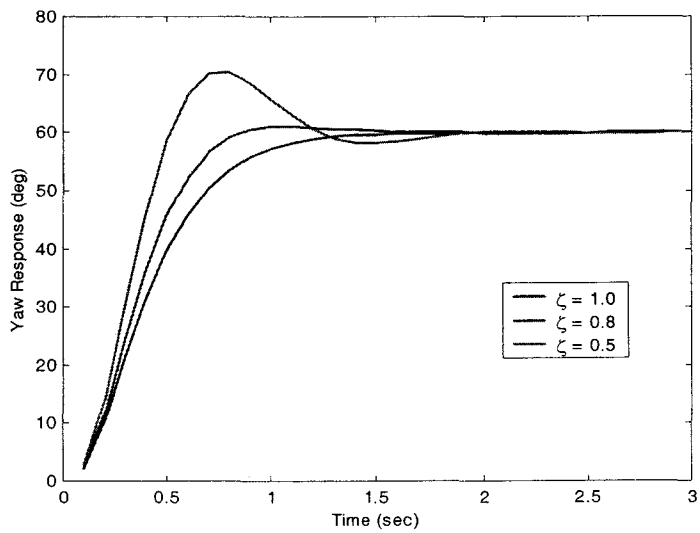


그림 3.68 요 축의 제어 성능 비교

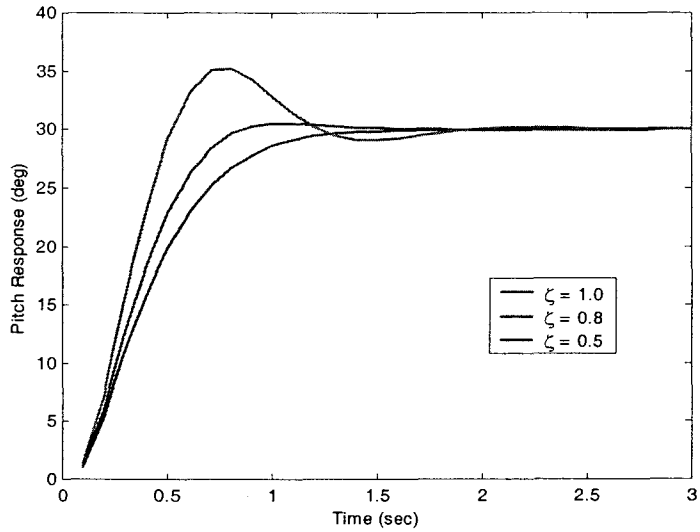


그림 3.69 피치 축의 제어 성능 비교

(3) 안정 정확도 성능 평가

안정 정확도 성능을 평가하기 위해서는 Load Profile과 같은 실제적인 외란의 크기와 주파수 등에 대한 데이터와 정확한 자이로의 특성 데이터가 있어야 한다. 단 여기에서는 외란을 고려하지 않고 순수하게 자이로 데이터의 오차에 의한 시선각 안정 정확도를 제어 대역폭에 따라 평가하였다. 제어 이득은 damping ratio z= 1 이라고 생각하여 결정했다. 그림 3.70및 그림 3.71의 시뮬레이션 결과에서 볼 수 있는 바와 같이 외란에 의한 영향을 고려하지 않더라도 100 Hz 정도의 충분히 빠른 제어 성능이 보장되어야 충분한 안정 정확도를 얻을 수 있다.

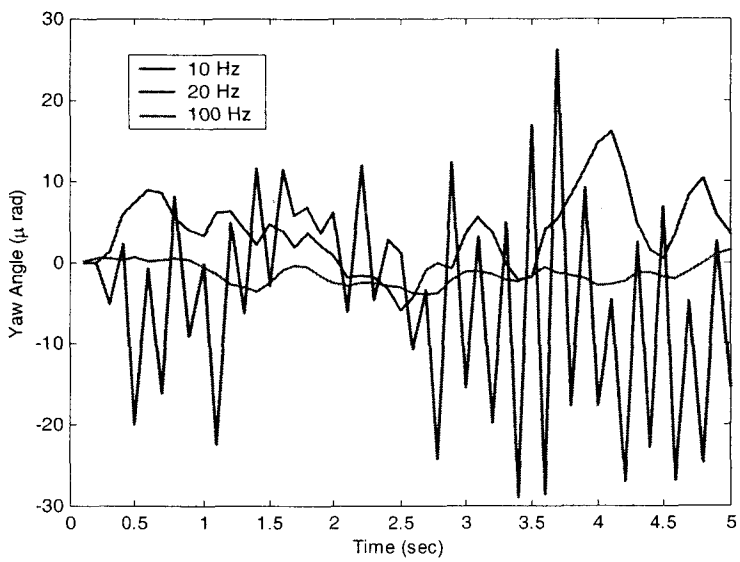


그림 3.70 제어 대역폭에 따른 요축의 안정 정확도

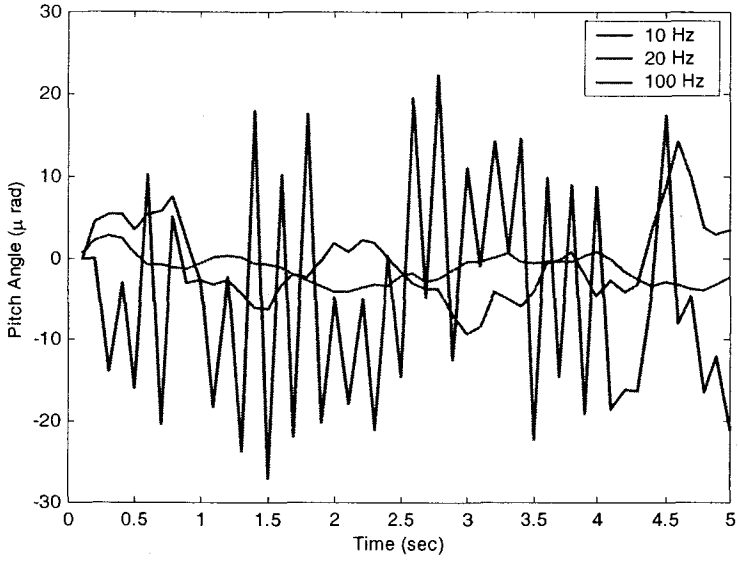


그림 3.71 제어 대역폭에 따른 피치 축의 안정 정확도

제 5 절 시험 및 성능평가

1. Gimbal 시스템의 안정 정확도 측정 방법

가. 시스템의 구성

김발 시스템의 요구 안정 정확도 100 mrad급을 측정 가능하도록 시스템을 구축하여야 한다. 아래 그림은 Gimbal의 안정 정확도를 측정하는 예이다.

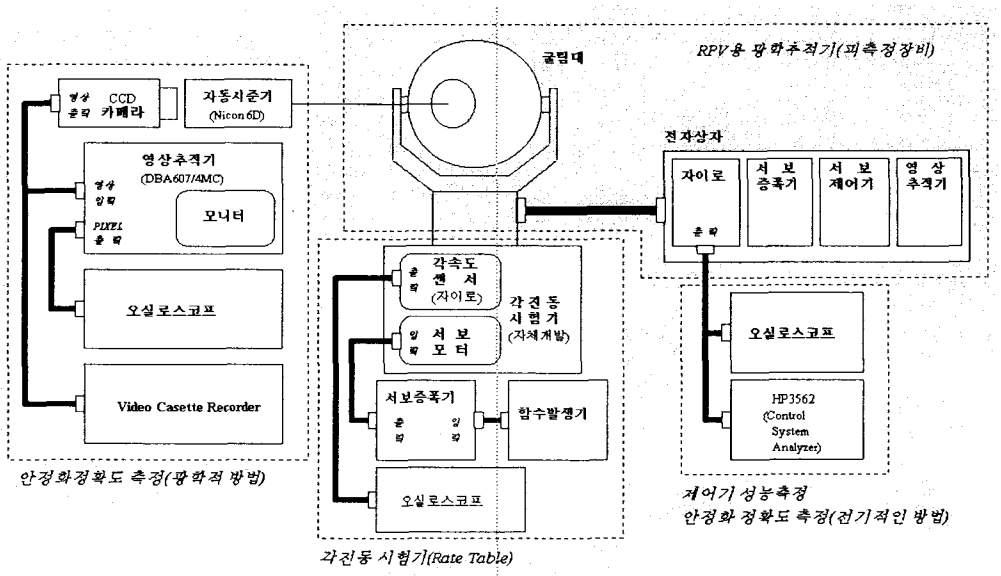


그림3.72 안정정확도 측정(예)

안정 정확도 측정 시스템은 광학적인 방법을 통하여 수행하며, 이를 위하여 영상 획득용 카메라, 각진동 시험기 및 데이터 획득 장치가 요구된다. 김발을 탑재할 비행체의 진동특성데이터를 입력으로 하여 각진동 시험기를 구동하고, 이때 획득한 영상을 분석 처리함으로써, 김발의 성능을 시험/평가한다. 그림 3.73는 안정 정확도 측정 시스템 구성을 보여준다.

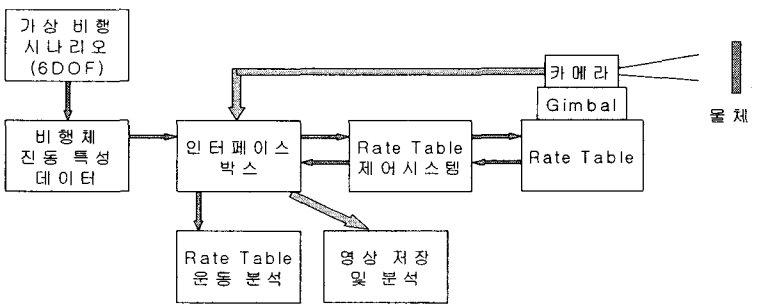


그림 3.73 안정 정확도 측정 시스템 구성

나. 운동판 탑재 시스템 데이터 획득 인터페이스 박스 제작

안정 정확도 측정 시스템 구성을 위하여, 데이터 획득, 각진동시험기 구동에 필요한 인터페이스를 통합하는 박스를 제작하였다. 외부 시스템 인터페이스 및 내부 결선은 각각 그림 3.74, 3.75와 같다.

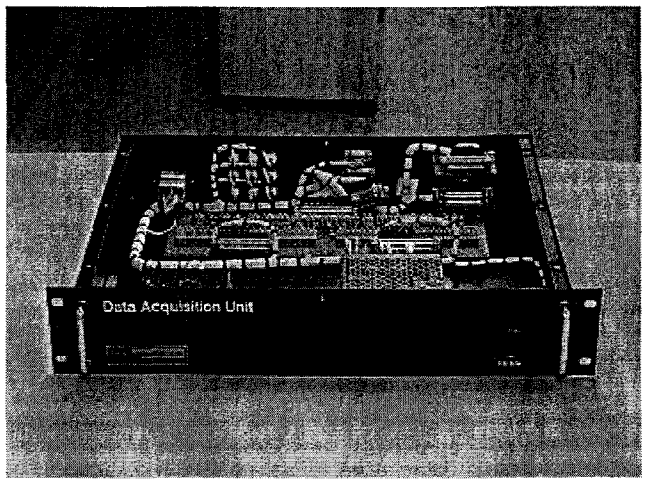


그림 3.74 운동판 탑재 시스템 데이터 획득 인터페이스 박스

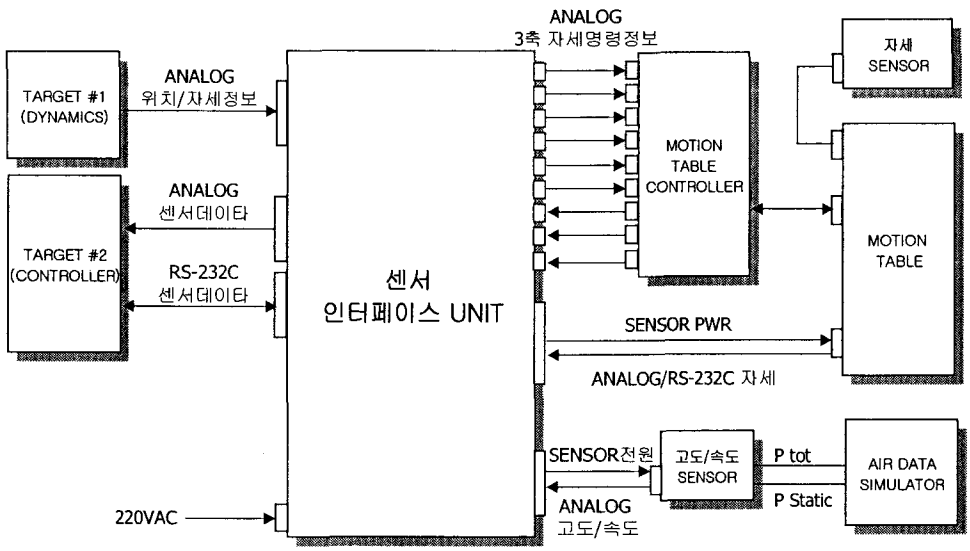


그림 3.75 외부 시스템 인터페이스

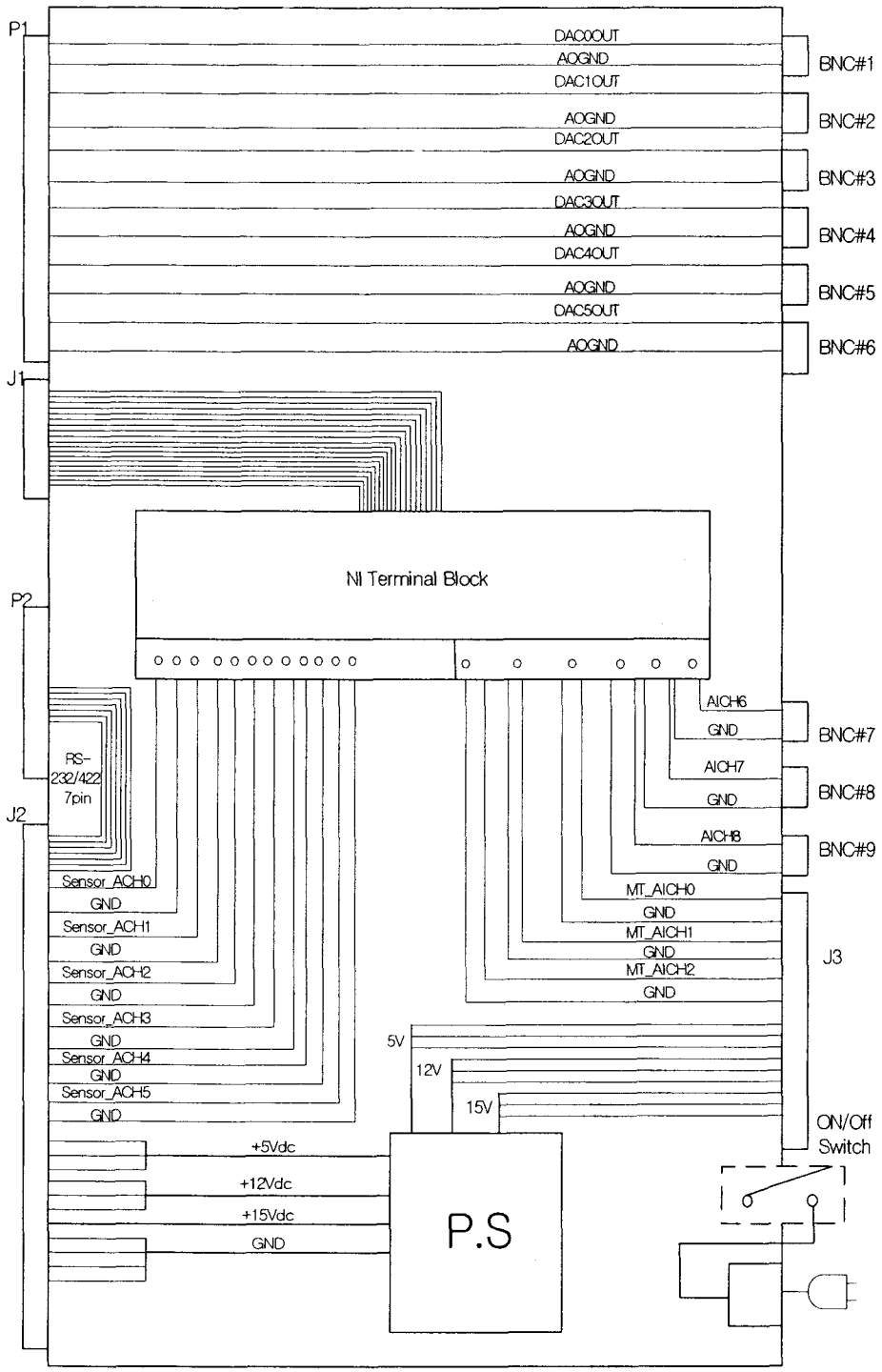


그림 3.76 데이터 획득 인터페이스 박스 내부 배선 개념도

다. Gimbal 시험용 운동판 제어 프로그램

김발 시험 시, 비행체 진동 특성 데이터를 이용하여 시험에 필요한 진동환경을 구현하여야 한다. 이를 위하여, 각진동 시험장치를 구동하기 위한 소프트웨어를 구현하였다.

(1) 프로그램화면

그림 3.77은 각진동 시험장치(운동판) 제어를 위한 프로그램 사용자 인터페이스를 보여준다.

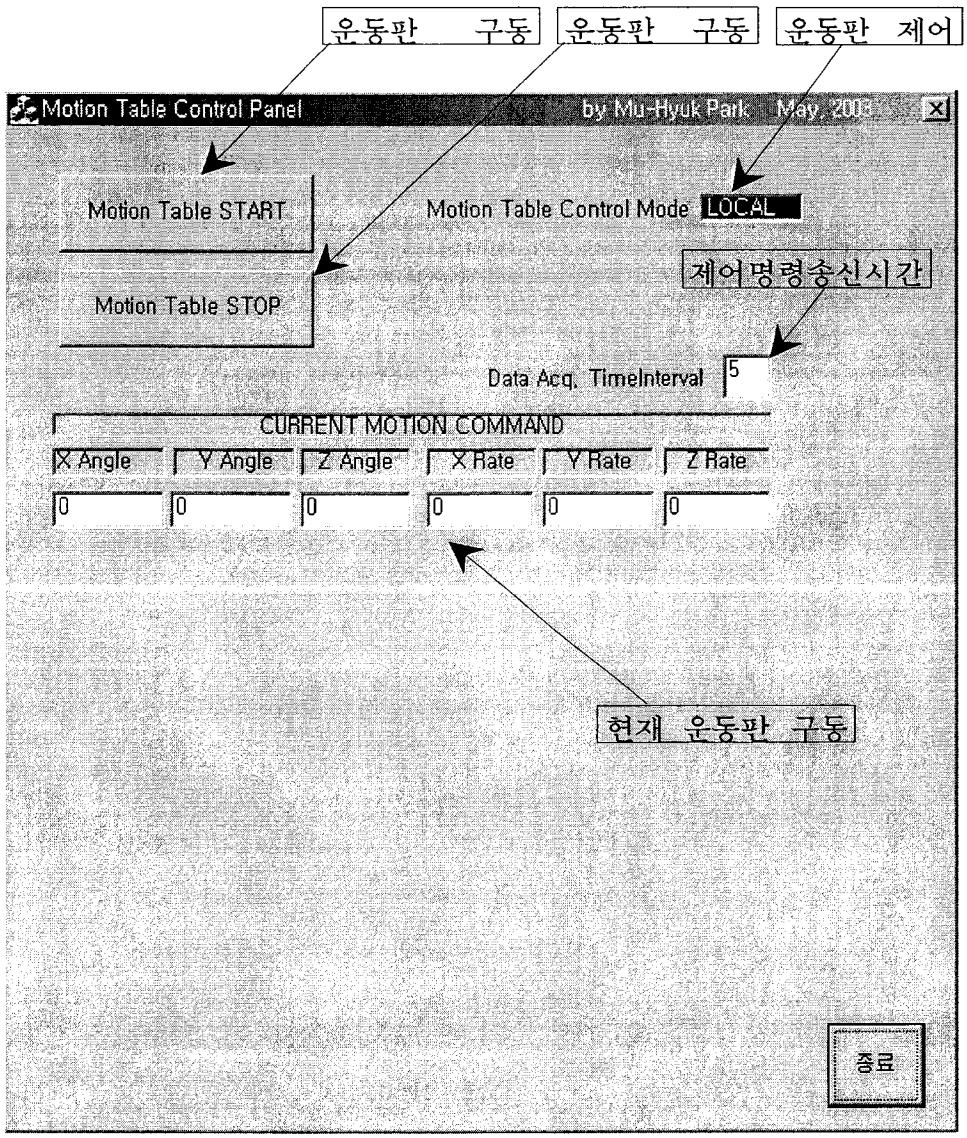


그림 3.77 Gimbal 시험용 운동판 제어 프로그램

(2) 운동판 제어 프로그램

Motion Table Start 명령 시, 다음의 Thread Routine에서 운동판 제어명령 데이터 파일을 읽고 이에 따라 운동판을 제어한다.

```
UINT ThreadRoutine(LPVOID pParam)
{
        unsigned char MsgData[500];
        DWORD MsgNo;
        int i,cnt=0;int InitialTic,FinalTic,CompDuration;
        unsigned int Sum = 0;
        FILE* fpMTCommand, * fpDataSave;
        char Command_filename[30];
        BOOL FIRST=TRUE;
        CFDRDemoDlg Mydlg;
        unsigned char CommandMsg[10];
        unsigned char MainMsg[]="test\r";

        InitialTic = GetTickCount();
        fpMTCommand = fopen("MTCommand.txt","r");
        fpDataSave = fopen("DUMMY_WithoutMotion.txt","w");

        while(THREADRUN)
        {

                FinalTic = GetTickCount();
                CompDuration= FinalTic-InitialTic;

            if(MTOPERATION)
                {
                        if(CompDuration > IntervalTime*1000 || FIRST)
                        {
                                FIRST =FALSE;
                                if(!feof(fpMTCommand))
                                {
                                fclose(fpDataSave);
                                fscanf(fpMTCommand,"%s %f %f %f %f %f %f\n",
                                     Command_filename,&Xp,&Yp,&Zp,&Xr,&Yr,&Zr);
                                ::SendMessage(MainWndHandle,WM_ADD_LIST2,0,0);
                                fpDataSave                                      =
fopen((LPCTSTR)Command_filename,"w");

                                m_pComm.Write((unsigned char *)mode_position,
                                                sizeof(mode_position)-1);
                                write_position(Xp,Yp,Zp);
                                m_pComm.Write((unsigned char *)position_cmd,
                                                sizeof(position_cmd)-1);
                                Delay(5);
                                m_pComm.Write((unsigned char *)mode_rate,
                                                sizeof(mode_rate)-1);
                                write_rate(Xr,Yr,Zr);
                                m_pComm.Write((unsigned char *)rate_cmd,
                                                sizeof(rate_cmd)-1);

                                InitialTic = GetTickCount();
                                }
```

```
                                else
                                {
                                        fclose(fpMTCommand);
                                        MTOPERATION=FALSE;
                                        LAST = TRUE;

::SendMessage(MainWndHandle,WM_ON_BUTTON2,0,0);

                                }
                        }
                        else;
                }
                else
                {
                }


                AcqTime = GetTickCount();
                ::SendMessage(MainWndHandle,WM_ADD_LIST,0,0);
                AHRS.Format("%d %12.4f %12.4f %12.4f %12.4f %12.4f

%12.4f",AcqTime,X_Rate,Y_Rate,Z_Rate,X_Vel,Y_Vel,Z_Vel);
                fprintf(fpDataSave,"%d  %12.4f  %12.4f  %12.4f  %12.4f  %12.4f
%12.4f
                        %12.4f %12.4f \n",AcqTime,  X_Rate,Y_Rate,Z_Rate,X_Vel,
                        Y_Vel,Z_Vel,X_Acc,Y_Acc);
                }

        if(CompDuration > IntervalTime*1000 && LAST)
                fclose(fpDataSave);


        return 0;
}
```

(3) 운동판 제어 명령 데이터 파일(예)

축1각도	축2각도	축3각도	축1각속도	축2각속도	축3각속도
0.	90.	-120.20862	0.	0.	2.
0.	90.	-120.20862	0.	0.	-2.
0.	0.	149.79138	0.	0.	2.
0.	0.	149.79138	0.	0.	-2.
90.	90.	-30.20862	0.	0.	2.
90.	90.	-30.20862	0.	0.	-2.
0.	90.	-120.20862	0.	0.	6.
0.	90.	-120.20862	0.	0.	-6.
0.	0.	149.79138	0.	0.	6.
0.	0.	149.79138	0.	0.	-6.

(4) 운동판 제원

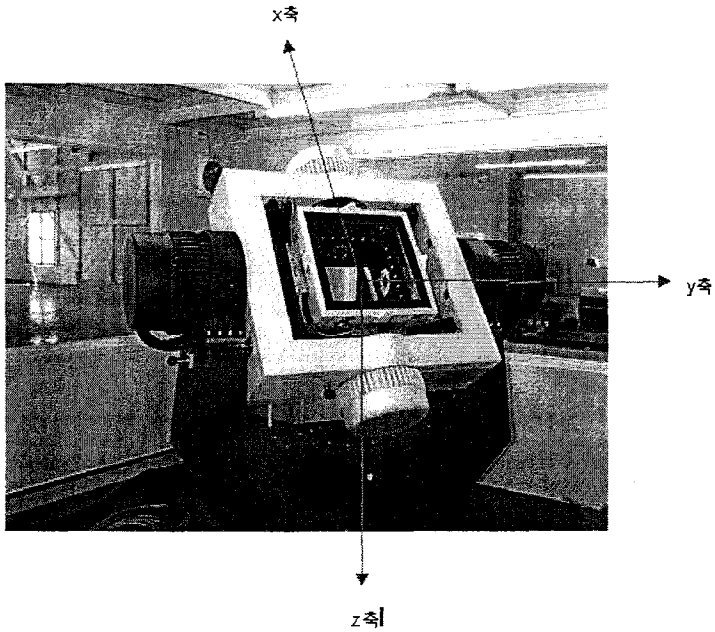


그림 3.78 3축 동시험대

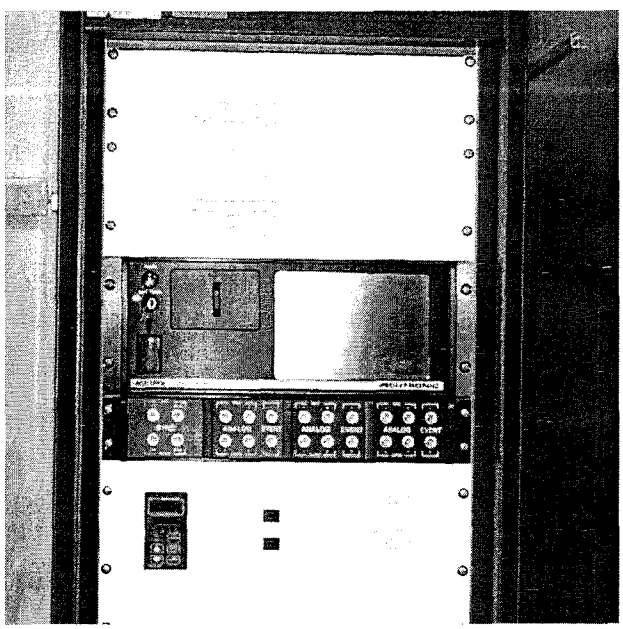


그림 3.79 3축 운동 시험대 콘트롤러

표 3.8 3축 운동 시험대의 재원 및 성능

Contents		Performance		
Dimension	Height of the system	2204 mm		
	Height of axes intersection	1514 mm		
	Width across outer axis	1990 mm		
	Base	800 mm		
Payload	Package size maximum	550x550x550 mm		
	Package weight nominal	60 kg payload(100 kg max.)		
	Electrical access to UUT (Slipring-wiring no. 360.06-0060	60 lines at 2 Amp/150 VDC		
		6 lines at 6 Amp/150 VDC		
		4 lines at 20 Amp/250 VDC		
Mechanical axis accuracy	Axis wobble	outer axis	middle axis	inner axis
		0.5 arc sec	1.4 arc sec	1.3 arc sec
	Orthogonality error			
	between inner and middle axis		1 arc sec	
	between middle and outer axis	2.3 arc sec		
Dynamic performance and control accuracy	Angular Freedom	infinite	infinite	infinite
	Axis inertia without payload	260 $kg\,m^2$	42 $kg\,m^2$	3.6 $kg\,m^2$
	Peak torque nominal	1440 Nm	840 Nm	100 Nm
	Peak acceleration with TC without payload	$\pm 300^\circ/s^2$	$\pm 1500^\circ/s^2$	$\pm 1600^\circ/s^2$
	Position accuracy	< 1.5 arc sec	< 1.5 arc sec	< 1 arc sec
	Rate range with TC	$\pm 400^\circ$/s	$\pm 600^\circ$/s	$\pm 1000^\circ$/s
	Rate range without TC	$\pm 400^\circ$/s	$\pm 600^\circ$/s	$\pm 1000^\circ$/s
	Rate stability (specified max. value)	averaged over 360 deg ±0.0002 %		

2. 2축 김발 시스템의 마찰 시상수 및 관성 모멘트 측정

Gimbal 시스템의 모델은 다음과 같다.

$$\ddot{\varphi} = -\alpha_{\varphi}\,\dot{\varphi} + T_{\varphi}/J_{\varphi}$$
$$\ddot{\theta} = -\alpha_{\theta}\,\dot{\theta} + T_{\theta}/J_{\theta} \qquad (1)$$

가. 2축 김발 시스템의 마찰 시상수 측정

김발 시스템의 모터를 이용하여 고속으로 회전시키다가 전원을 끊는다. 그러면 모터 회전축 및 김발 회전체는 자유롭게 회전하고, 김발의 베어링 마찰에 의하여 회전축은 감소하고 결국은 정지하게 된다.

위의 경우는 수식(1)에서 모터 구동력이 영인 경우로 식(2)가 된다.

$$\ddot{\theta} - \alpha_{\theta}\,\dot{\theta} = 0 \qquad (2)$$

수식(2)를 2번 적분하고 시간 t = 0일 경우 회전각 $\theta = 0$로 하면

$$\theta = C\,(1 - e^{-\alpha_{\theta} t}) \qquad (3)$$

위의 수식에서 미지수는 C, α_{θ} 2개임으로 2개의 경우에 대하여 회전각과 시간을 측정하면 2개의 미지수를 구할 수 있다. 같은 방식 다른 축에 대해서도 측정을 수행하면 된다.

나. 2축 김발 시스템의 관성 모멘트 측정

- 방법 1

김발 시스템의 모터를 이용하여 일정한 속도로 회전을 시킨다.

위의 경우는 수식(1)에서 회전 각속도가 영인 경우로 식(4)가 된다.

$$J_{\varphi}\,\alpha_{\varphi}\,\dot{\varphi} = T_{\varphi} \qquad (4)$$

수식(4)에서 베어링의 마찰 시상수는 알고 있고, 회전각속도와 회전 모멘트를 측정하면 관성 모멘트를 측정할 수가 있다.

회전 각속도는 회전각과 시간을 측정하면 회전각속도를 얻을 수가 있다. 회전 모멘트는 그림과 같이 정밀 로드 셀을 이용하면 정밀하게 측정할 수가 있다.

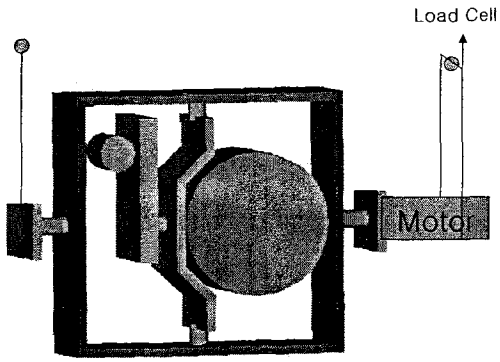


그림 3.79 관성 모멘트 측정 구성도

- 방법 2

Gimbal 시스템에 Mass를 달아서 진동을 시킬 때, 지배하는 방정식은 아래와 같다.

$$\ddot{\varphi} = -\alpha_{\varphi}\dot{\varphi} + T_{\varphi}/J_{\varphi}$$
$$T_{\varphi} = mg*\sin(\varphi)$$

다만 관성모멘트는 진동을 시키는 Mass를 포함하고 있다. 진동하는 Mass는 계산 또는 측정을 통하여 제거한다. 작은 변위에 대한 수식은 아래와 같이 된다.

$$\ddot{\varphi} + \alpha_{\varphi}\dot{\varphi} + \frac{mg}{J_{\varphi}}\dot{\varphi} = 0$$

위의 수식의 진동수를 측정하여 관성 모멘트를 측정할 수 있다.

(1) 회전수 및 시간 측정 장비

- 측정장비명 : 펄스 간격 측정 장비
- 사용 환경 :
  . 사용 온도 : 회전 감지 센서 (-20℃ - 50℃)
  Test Meter(0℃ - 50℃)
- 전원 . DC 12V
- 장비 사양
  (가) 회전 센서
  . Hall 센서, Photo 센서 등 펄스 신호 발생 가능 센서
  . 입력 전원 : 10-30VDC

. 응답 특성 : 0.5msec이하

. 입력 단자 : 입력전원단자/NPN Open Collector 출력단자/GND 단자

(나) 입력 펄스 신호

. 펄스 신호 : Beam Sensor 공급 전원의 정압 정도 펄스

. 펄스 폭 : 0.1msec 내외

. 펄스 간 간격 : 최소 20msec에서 최대 200sec

. 총 실험시간 : 2000sec 이내

(다) 전원

. 불안전한 DC 12V를 승압하여 스위칭 개념으로 DC 12V, DC 5V를 생성

(라) 측정

. 펄스의 Rising Edge간 시간 측정

. 정확도 : 0.1msec 이하

. 최대 펄스 갯수 : 25개

. STOP도 하나의 펄스로 간주

(마) 명령 버튼

. READY, STOP, SEND, RESET

. DISPLAY, NUMBER, MARK, DIVIDER

. 4개의 Arraw Key

(바) 표시

. LCD로 표시

(사) 기타 단자.

. DC 12V 입력 단자(노트북형)

. 전원 스위치

. male 9 pin RS-232C 단자

. 3 Pin Beam Sensor Connector

(아) 데이터 저장 및 전송 포맷

. 2자리 Number + blank + 25*(시간 + blank)

. 위의 포맷으로 100개 데이터 저장

. 전송 전용 프로그램으로 PC에서 데이터 받음.

. Power off 후에도 데이터 기억

(자) Divider

. 1-1000 까지 세팅 가능

3. Gyro 모델 변수 측정

Gyro의 측정 식은 다음과 같이 표현된다.

$$\omega_\varphi = (1+\lambda_\varphi)(\dot{\varphi}_I + b_\varphi) + w_\varphi$$
$$\omega_\theta = (1+\lambda_\theta)(\dot{\theta}_I + b_\theta) + w_\theta \quad (5)$$

여기에서 $\lambda_\varphi, \lambda_\theta$는 scale factor, b_φ, b_θ는 drift에 의한 bias이고, w_φ, w_θ는 random walk noise이다.

표 3.9 제어 알고리즘의 Key Parameters

표현	Typical Valve	설명
$\alpha_\varphi, \alpha_\theta$	5 /sec	김발의 Bearing의 마찰에 의한 시상수
J_φ, J_θ		김발의 관성 모멘트
$\lambda_\varphi, \lambda_\theta$	0.2%	Gyro의 Scale Factor
b_φ, b_θ	1-3deg/h	Gyro의 Bias
w_φ, w_θ	$\leq 10\,\mathrm{deg}/\sqrt{h}$	Gyro의 Random Walk Noise

가. 3축 운동 시험대를 이용한 오차보정 시험

각 시험에서 추정되는 오차계수들을 표2에 나타내었다.

표 3.10 관성항법센서의 오차계수 및 추정

	Notation	Error Coefficient	3-Position Rate Test	24-Position Static Test
Rate Gyro	$\omega_x, \omega_y, \omega_z$	Inertial angular rate (reference value)		
	$\omega_x^m, \omega_y^m, \omega_z^m$	Measured angular rate		
	$\lambda_{\omega_x}, \lambda_{\omega_y}, \lambda_{\omega_z}$	Scale factor error	○	
	$\alpha_{xy}, \alpha_{xz}, \cdots, \alpha_{zx}, \alpha_{zy}$	Misalignment error	○	
	m_1, m_2, m_3	Mass unbalance error		•
	q_1, q_2, q_3	Quadrature term		•
	n_1, n_2, n_3	Anisoelasticity error		•
	$b_{\omega_x}, b_{\omega_y}, b_{\omega_z}$	Bias		•
	$\eta_{\omega_x}, \eta_{\omega_y}, \eta_{\omega_z}$	Random noise		

표 3.11 WGS-84 좌표계에서의 지구 물리량

Parameters		Notation	Value
지구반지름 (Equatorial radius)		a	6,378,137 m
Flattening (ellipticity)		f	1/298.257223563 (0.00335281066474)
지국 자전 각속도 (Angular rate of the earth)		Ω	7.292115×10^{-5} rad/s (0.004178074 deg/s)
Geodetic Position in Taejon	경도 Longitude	λ	127.35438835°(σ=1.252)
	위도 Latitude	Φ	36.37685175°(σ=1.321)
	고도 Height(MSL)	h	60.738 m (σ=2.623)
지국 중력 가속도 (대전) (Gravity in Taejon)		g	9.7982994 m/s^2

* Gravity : 한국표준연구원(KRISS)
* Geodetic Positon : 한국항공우주연구소 비행자세제어실

Rate Test로는 식 (1)에서 정의된 오차모델 중 이득계수 오차(scale factor error)와 비정렬 오차(misalignment error)를 구할 수 있다. 관성센서의 3축에 대하여 측정코자하는 축을 3축 운동 시험대의 회전축(z축)과 동일한 Local Vertical이 되도록 정렬시킨 후 3축 운동 시험대를 시계방향(clockwise)과 반시계방향(counter-clockwise)으로 회전시킬 때 관성센서 축에 인가되는 지구자전속도, 모션테이블의 회전속도, 중력가속도에 대한 출력 값의 적분차이로부터 에러계수를 구할 수 있다. 우선 3축 운동 시험대의 3축을 NED 좌표계로 정렬시킨 후 관성센서의 z축을 3축 운동 시험대의 회전축(z축)과 정렬시키면 다음과 같은 초기조건이 주어진다.

- 초기조건

: $\Phi_3 = \Omega_3 = const$, 주기 = T

: $\Phi_1(t) = 0$, $\Phi_2(t) = 0$, $\Phi_3(t) = \Phi_3 \cdot t = \Omega_3 \cdot t$

: Ω_{NED}

$$\begin{aligned} \Omega_N &= \Omega \cdot \cos \phi \\ \Omega_E &= 0 \\ \Omega_D &= \Omega \cdot \sin \phi \end{aligned}$$

여기서, 3축 운동 시험대에서 임의로 주어지는 회전 각속도는 Ω_3이며, 시간 t에 따른 회전각도는 주어진 시간에서 $\phi_3 \cdot t = \Omega_3 \cdot t$ 가 된다. 반면에 ϕ_1, ϕ_2 두 축은 주어지는 각속도가 0이므로 시간에 따른 회전각도도 0이 된다. 장착된 자이로에서 측정되는 각속도 값은 주어지는 회전 각속도 Ω_3와 위도에 따라 인가되는 지구 자전 속도가 더해진 값이 된다.

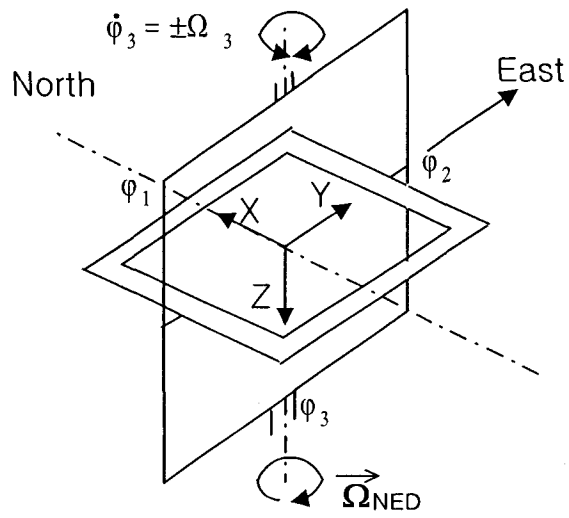


그림 3.80 각속도 시험 정렬상태

(xyz축 : IMU, ϕ_1, ϕ_2, ϕ_3 : 3축 운동 시험대)

회전하는 동안에 각각의 위치에서 측정되는 각속도와 가속도의 참값은 식 (3)와 식 (4)에서 정의된 지구 자전속도와 지구 가속도항을 포함하게 되고 다음 식 (6)으로 정의된다.

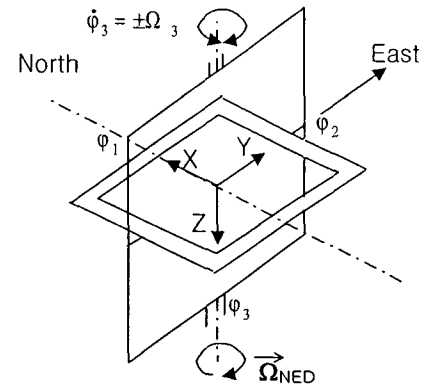
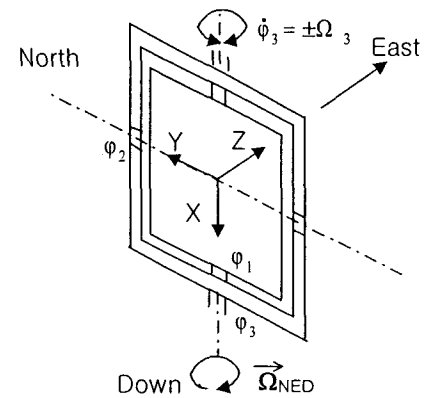
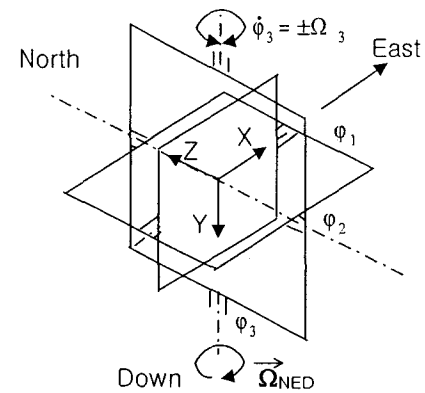
Main Positon No.	Initial Position of the 3-Axis Test Table	IMU Attitude		
		Φ_1 (Roll)	Φ_2 (Pitch)	Φ_3 (Yaw)
1		0°	0°	0°
		$\alpha_{xy} = -\frac{\triangle J_{x,1}}{4\pi(1+\lambda_{\omega_x})}$ $\alpha_{yx} = \frac{\triangle J_{y,1}}{4\pi(1+\lambda_{\omega_y})}$ $\lambda_{\omega_z} = \frac{\triangle J_{z,1}}{4\pi} - 1$		
2		0°	-90°	-90°
		$\alpha_{yz} = -\frac{\triangle J_{y,2}}{4\pi(1+\lambda_{\omega_y})}$ $\alpha_{zy} = \frac{\triangle J_{z,2}}{4\pi(1+\lambda_{\omega_z})}$ $\lambda_{\omega_x} = \frac{\triangle J_{x,2}}{4\pi} - 1$		
3		90°	0°	90°
		$\alpha_{xz} = \frac{\triangle J_{x,3}}{4\pi(1+\lambda_{\omega_x})}$ $\alpha_{zx} = -\frac{\triangle J_{z,3}}{4\pi(1+\lambda_{\omega_z})}$ $\lambda_{\omega_y} = \frac{\triangle J_{y,3}}{4\pi} - 1$		
	Measurement : Sum of the Incremental Signals Constant Rate : ±0.1, ±0.3, ±1.0, ±3.0, ±10.0 。/s			

그림 3.81 각속도 시험 (Rate Test) 정렬 위치

4. 열 환경시험

가. Thermal Chamber 수리

- 가변 온도 : -40℃ - 100℃

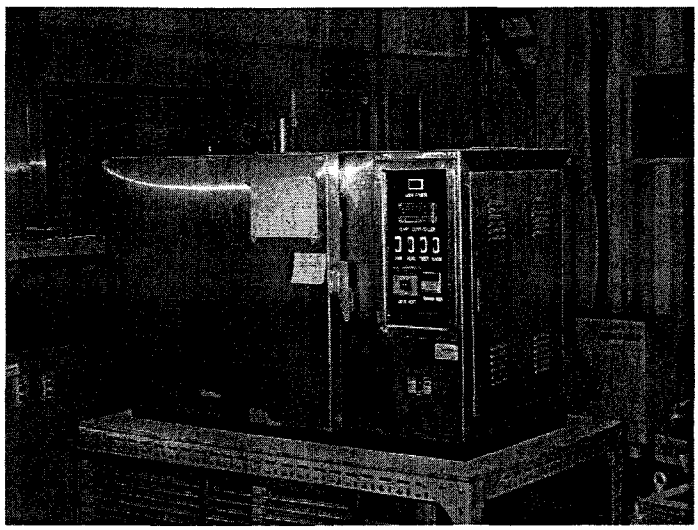


그림 3.82 열 환경시험 장치 모습

- 크기 : 1000 * 600 * 500㎜

나. 데이터 측정 장비 구축

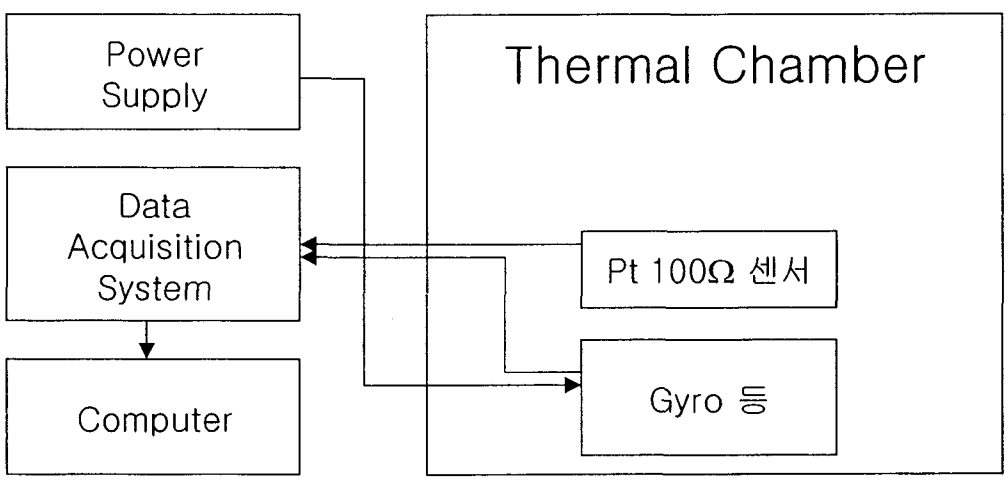


그림 3.82 데이터 측정 시스템 구성도

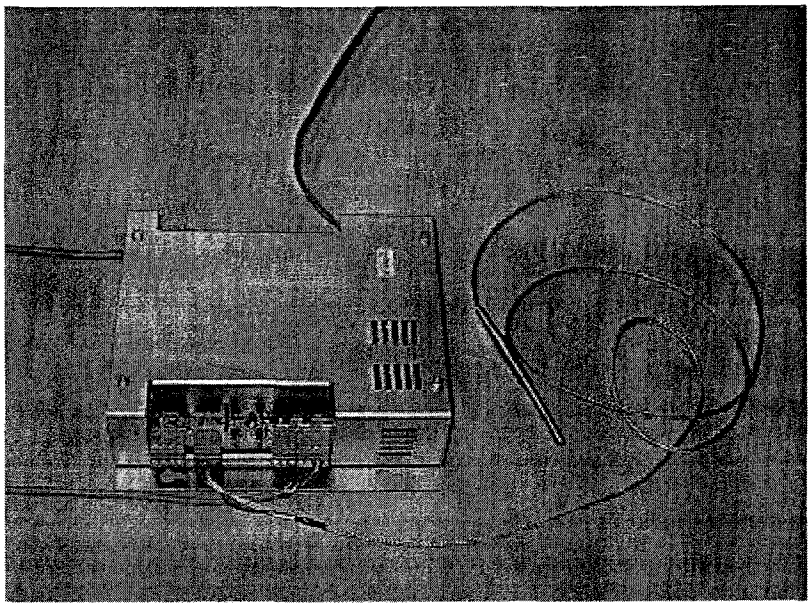


그림 3.84 온도 측정 장비

다. 시험관련 규정

- 고온저장시험(MIL-STD-810E 기준)

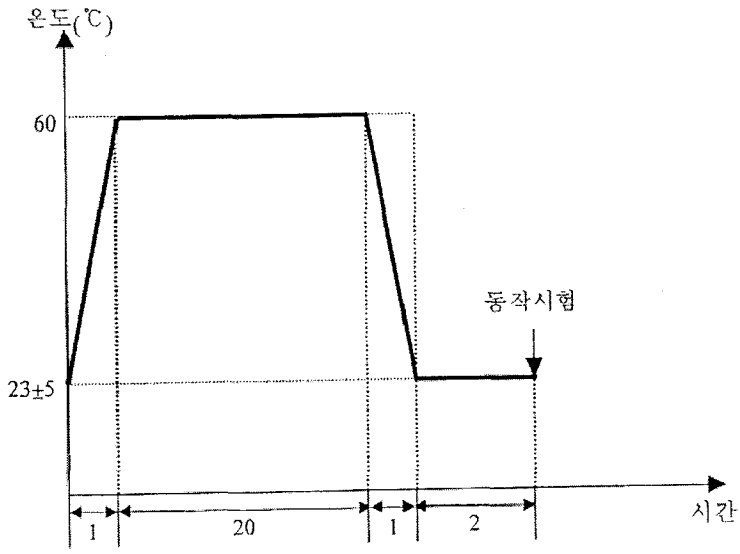


그림 3.85 고온저장시험 프로파일

• 저온저장시험(MIL-STD-810E 기준)

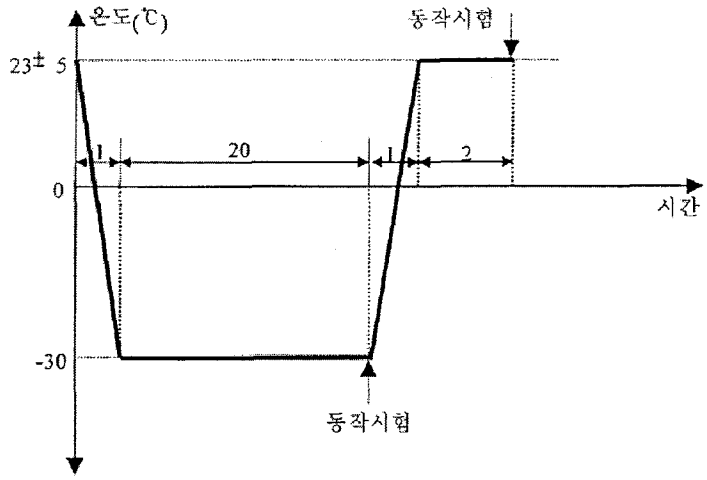


그림 3.86 저온저장시험 프로파일

• 고온동작시험(MIL-STD-810E 기준)

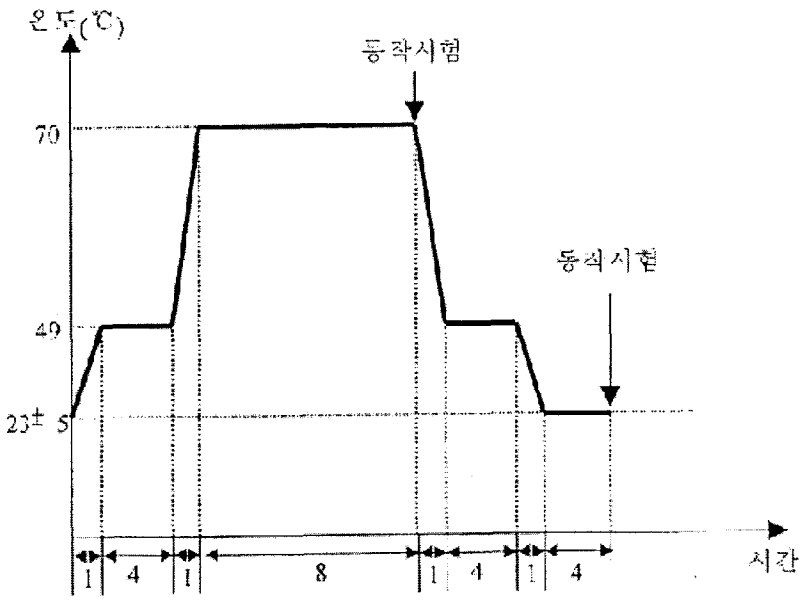


그림 3.87 고온동작시험 프로파일

라. 열 환경시험 데이터 획득 프로그램

(1) 열 환경시험 데이터 획득 프로그램 구동 화면

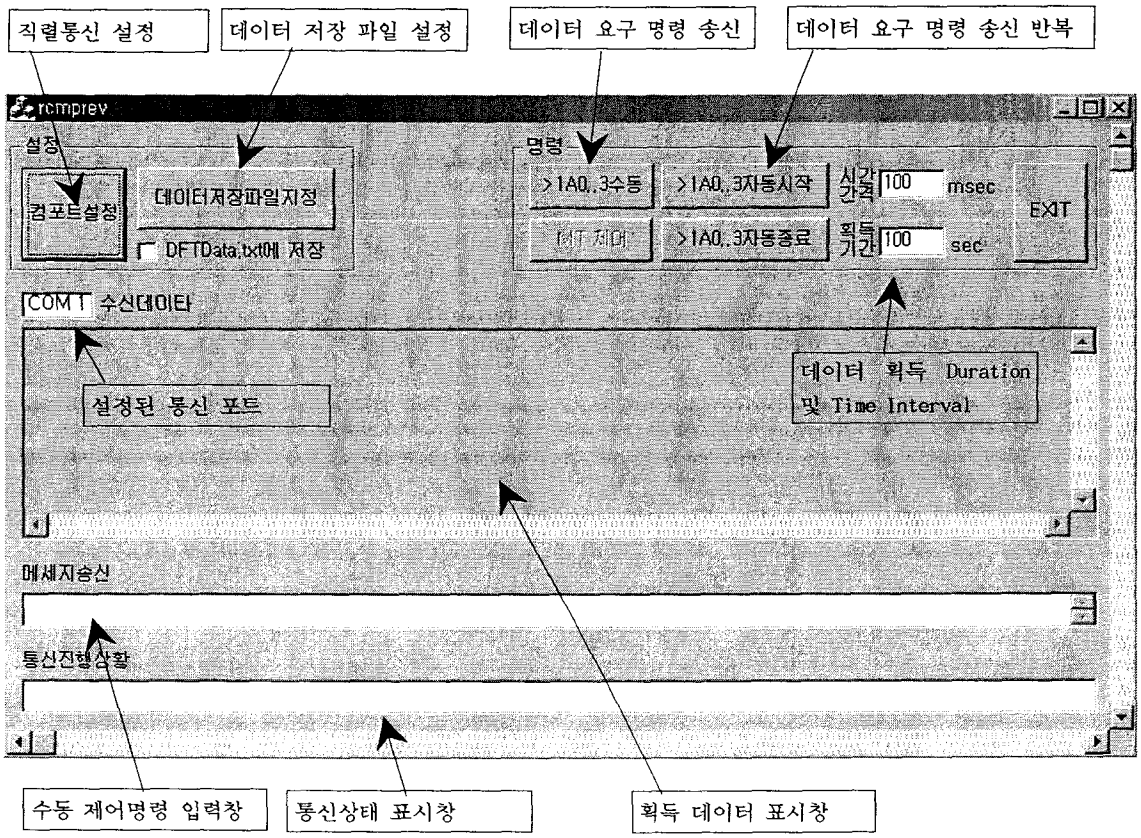


그림 3.88 열 환경시험 프로그램 구동 화면

(2) 측정 프로그램 개요

(가) 데이터 요구 명령 송신 함수

```
void CRcmprevDlg::OnRCStart()
{
        // TODO: Add your control notification handler code here

        if (m_Connected == FALSE) {
                        MessageBox("Can't open Comport. Comport Setup First
Please!");
                        return;
        }
```

```
        char c[] = ">1A0..3\r";

        m_pComm.WriteCommBlock((char *)c, 8);
        m_gndctrl.ResetContent();
        m_gndctrl.AddString(">1A0..3\r 송신완료");

}
```

(나) 자동 데이터 요구 명령 송신 함수

```
LONG CRcmprevDlg::OnTestRoutine(WPARAM wParam, LPARAM lParam)
{
        int FinalTic,CompDuration,cnt=0;
        FinalTic = GetTickCount();
        if (FirstTic)
        {
                InitialTic = FinalTic;
                FirstTic =FALSE;
        }
        CompDuration = FinalTic - InitialTic;
        if(CompDuration > Duration)  MyTimer.KillMulTimer();
        //      TRACE("%d\n", tttt);
        if (m_Connected == FALSE) {
                        MessageBox("Can't  open  Comport.  Comport  Setup  First
Please!");
                        return 0;
        }

        char c[] = ">1A0..3\r";

        m_pComm.WriteCommBlock((char *)c, 8);

        m_gndctrl.ResetContent();
        m_gndctrl.AddString(">1A0..3\r 송신완료");
        return 0;
}
```

(다) 데이터 획득 시 디스플레이 및 저장 함수

```
LONG CRcmprevDlg::OnReceiveData(UINT WParam, LONG LParam)
{
        CString tmp;
        int cnt=0, i;
        m_gndctrl.ResetContent();

        // 컴포트로부터 데이타를 받으면, 버퍼에 저장한다.

        if(hexa == FALSE) {
                cnt = m_rcvdata.GetLength();
                m_rcvdata+=m_pComm.abIn;
                cnt = m_rcvdata.GetLength()-cnt;

                for(i = 0; i < cnt; i++)
                {
                        if (logok == TRUE)
                                fprintf(fpt,"%c", m_pComm.abIn[i]);
```

```
            }
            fprintf(fpt,"\n");
            m_gndctrl.ResetContent();
            m_rcvdatar = m_rcvdata;

        }
        else {

            m_rcvdata=m_pComm.abIn;
            cnt = m_rcvdata.GetLength();
            m_rcvdata.Empty();
            m_rcvdata += m_rcvdatar;
            for(i = 0; i < cnt; i++)
            {
                if (m_pComm.abIn[i] == '<')
                {
                    m_rcvdata = m_rcvdata + "\r\n";
                    tcnt++;
                    if (logok == TRUE)
                    {
                        fprintf(fpt," \n");
                        fprintf(fpt,"%d  ", GetTickCount());
                    }

                }
                m_rcvdatar.Format("%c", m_pComm.abIn[i]);
                if (logok == TRUE)
                    fprintf(fpt,"%c", m_pComm.abIn[i]);

                m_rcvdata += m_rcvdatar;

            m_rcvdatar = m_rcvdata;
            }

        }

            cnt = m_rcvdata.GetLength();
            m_rcvdata.Delete(0,cnt-124);

        UpdateData(FALSE);
        m_gndctrl.ResetContent();
        m_gndctrl.AddString("텔리메트리 데이타 수신 성공");
        return TRUE;
}
```

마. 열 환경 시험 예

Gimbal 개발에 사용되는 레이트 자이로의 온도 특성을 파악하고자 온도 챔버에 센서를 집어넣고 온도 변화에 따른 레이트 자이로 출력을 파악하였다. 사용된 센서는 온도를 자동으로 보상하는 회로가 포함되어 있어 센서 출력 특성이 매우 우수함을 알 수가 있다.

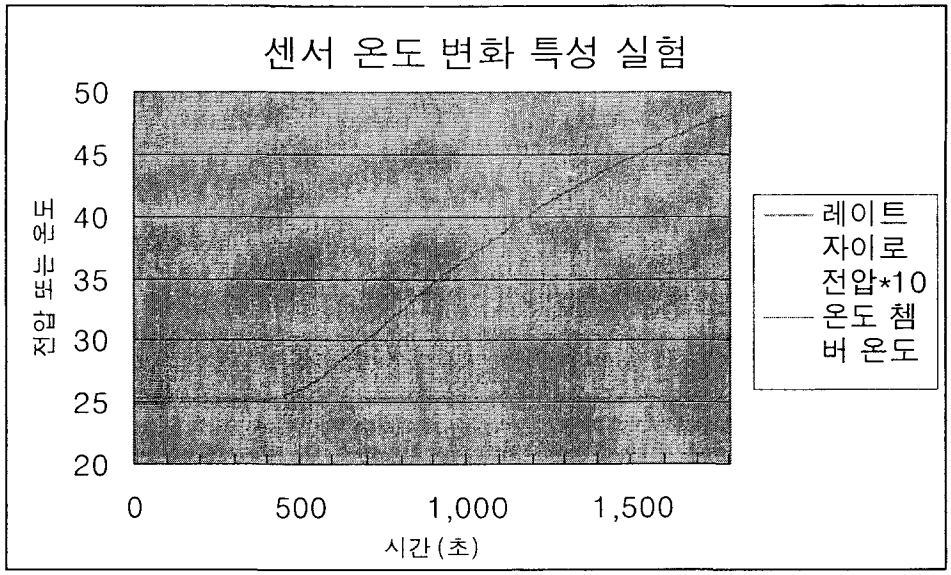


그림 3.89 센서온도변화 특성

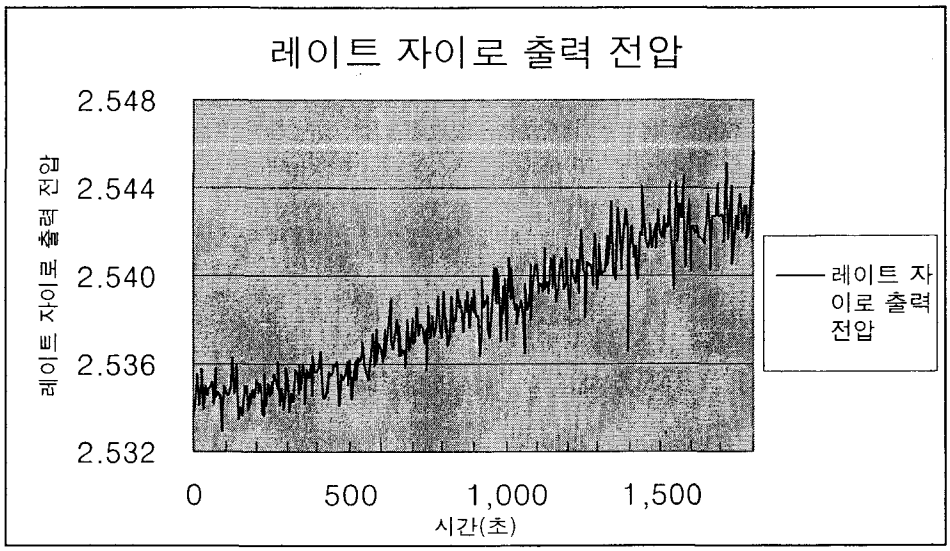


그림 3.90 레이트 자이로 출력

5. 진동시험

가. Shaker System의 사양

- 제 작 사 : LDS (Ling Dynamics System) Co.
- 모델번호 : V964 DPAK 130/140K - Combo Base
- 슬립테이블(Slip table)크기 : 900 mm × 900 mm (HBT 900)
- 용 량(Capability) : 80 KN

표 3.12 Shaker system 사양

	수 평	수 직
Maximum Displacement	- Sine 38 mm - Transient 50.4 mm	- Sine 38 mm - Transient 50.4 mm
Maximum Acceleration	- Random13.21 g(rms) - Sine 13.221 g(rms)	- Random 12.44 g(rms) - Sine 12.44 g(rms)

- 데이터 측정/수집 시스템 (Data measurement system)

항 목	모 델	제 작 사	수 량	비 고
Acquisition System	Maxion 9502	CONCURRENT COMPUTER Co.	1	가속도계용 : Ch.64 Strain gage용 : Ch.32 Microphone용 : Ch.16
Accelerometer	ENDEVOCO 2220D	ENDEVOCO	12	

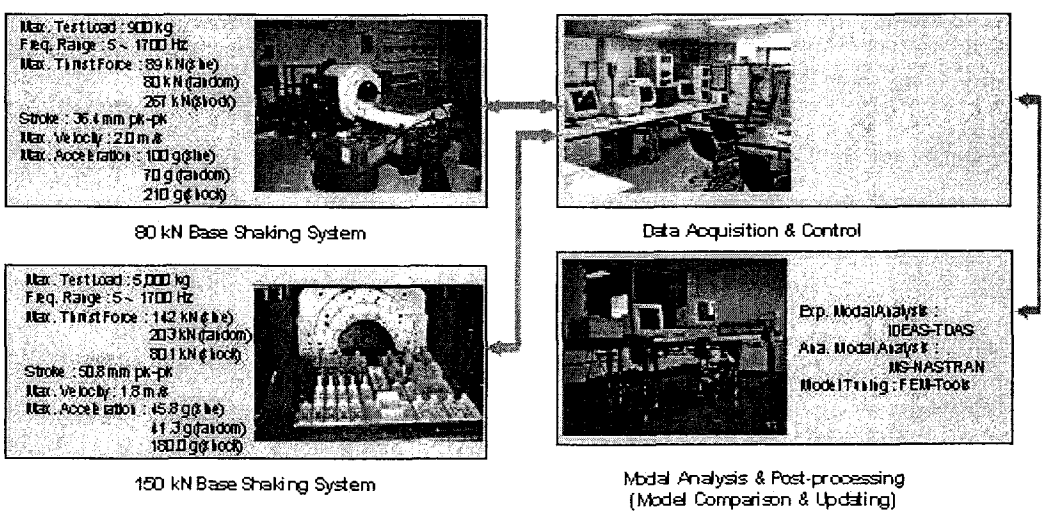


그림 3.91 진동시험장비

나. 진동 시험기 사양

표3.12 진동 시험기 사양

Model	DHV-6000 (진동 시험기)
Plate Size	450 * 450 * 28 t
진 폭	0 ~ 5mm
진 행 방 향	수직·수평방향
주파수 범위	5 ~ 200Hz
시료 무게	MAX : 50kg

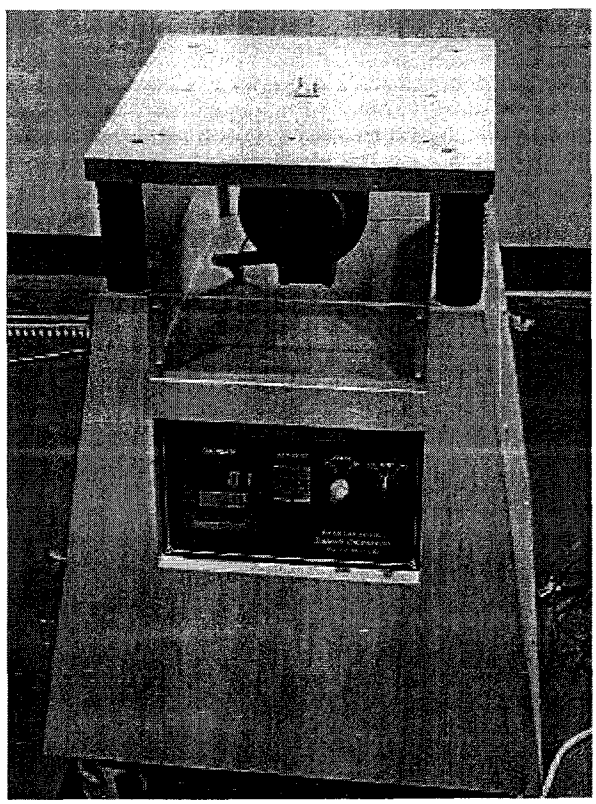


그림 3.92 Vibration Tester

다. 진동시험 규정

- 운용진동시험(MIL-STD-810E 기준)

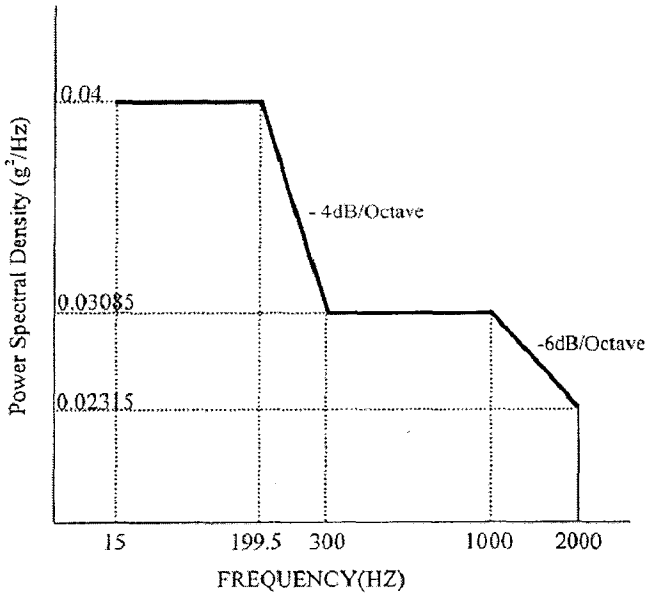


그림 3.93 운용진동시험 프로파일

(1) 진동 시험 예

(가) Vibration Shaker (40Hz)

아래의 그림은 진동을 40Hz로 발생시켜 시간에 따른 전압을 측정한 결과이다. 0.1초에 주기가 약 4번이 반복됨을 확인할 수 있다. 또한 이를 FFT(Fast Fourier Transform)를 수행하여 주파수 영역에서 확인한 결과 40Hz 근방에서 최대 값을 나타내는 것을 확인함으로써 정확한 진동 측정이 이루어졌음을 확인할 수 있다.

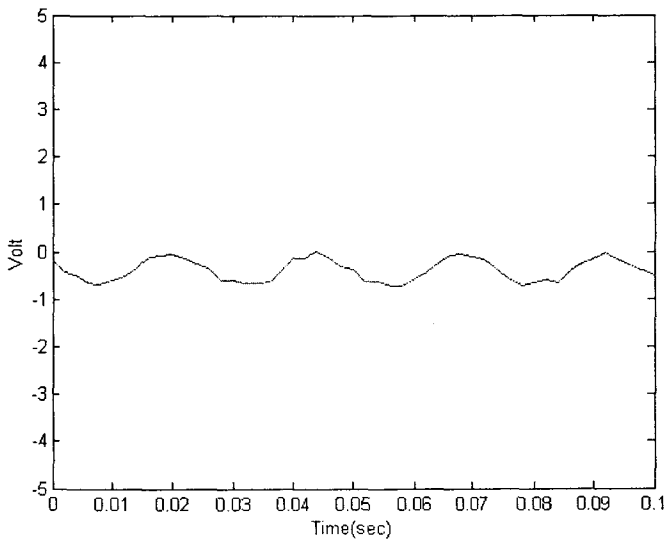


그림 3.94 가속도계 출력

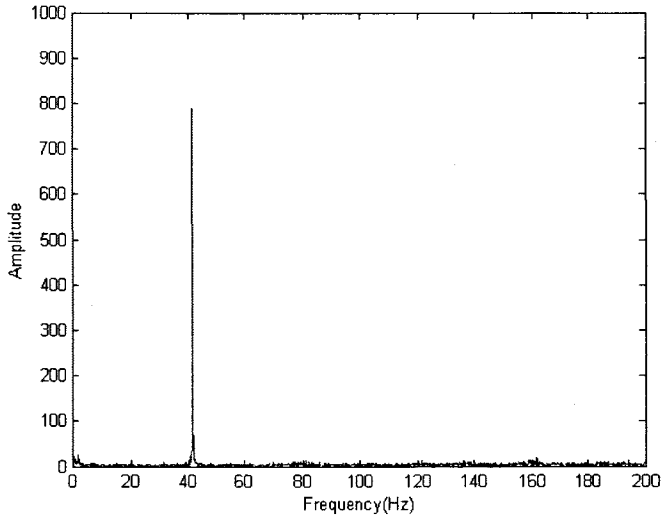


그림 3.95 측정 신호 FFT 결과

(나) Vibration Shaker (80Hz)

아래의 그림은 진동을 80Hz로 발생시켜 시간에 따른 전압을 측정한 결과이다. 0.1초에 주기가 약 8번이 반복됨을 확인할 수 있으며 . 또한 40Hz로 시험하였을 때보다 전압의 진폭이 커진 결과를 확인할 수 있다. 또한 이를 FFT(Fast Fourier Transform)를 수행하여 주파수 영역에서 확인한 결과 80Hz 근방에서 최대 값을 나타내는 것을 확인함으로써 정확한 진동 측정이 이루어졌음을 확인할 수 있다.

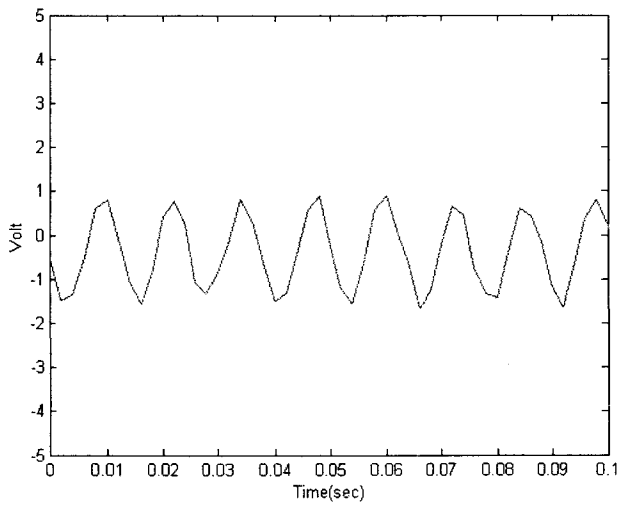


그림 3.96 가속도계 출력

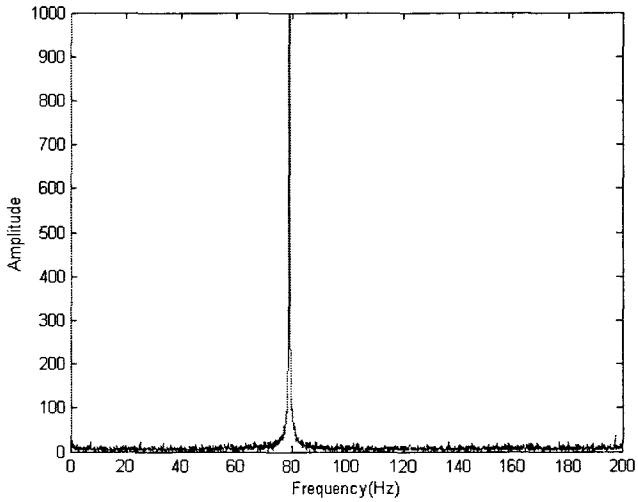


그림 3.97 측정 신호 FFT 결과

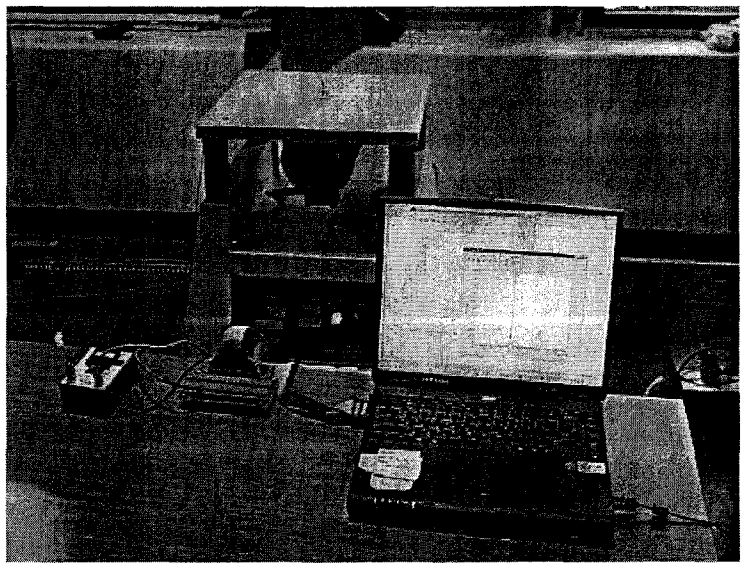


그림 3.98 진동 시험을 수행하는 모습

6. 충격 시험

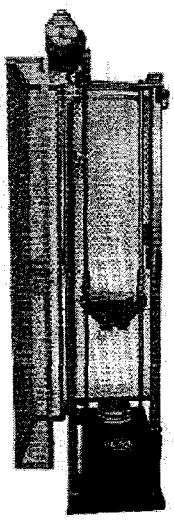


그림 3.99 충격 시험기

- Carriage Size : 30 cm x 30 cm
- Rated Test Load : 15 kg
- Maximum Acceleration : over 3,000 g in time domain over 5,000 g(Q=10) in SRS
- Minimum Pulse Duration : 0.5 msec

• 충격시험 (MIL-STD-810E 기준)

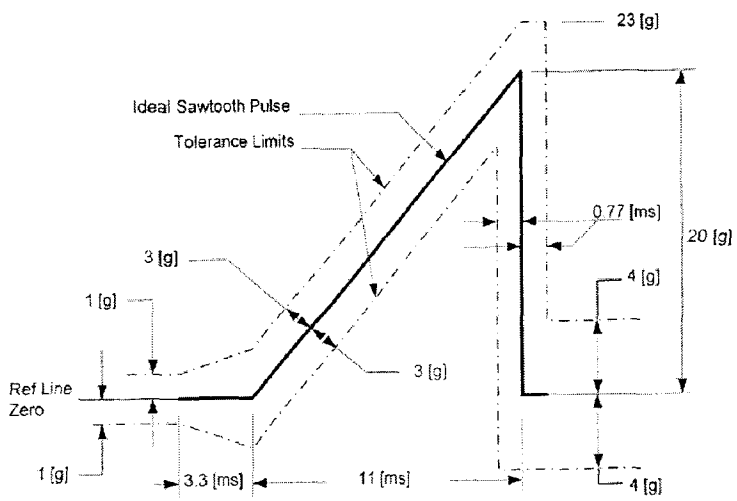


그림 3.100 충격시험 프로파일

7. 인증관련 규정

가. MIL-STD-810F 환경시험 내역

나. RTCA/DO-160환경시험 내역(21종)

번호	시험 제목	시험의 목적	시험의 내역	범위	시험장비
1	온도 및 고도시험 (Temperature and Altitude) Sec 4.5	부품의 항공기내에서의 장착 장소와 항공기 최대운용고도에서의 기기 특성을 확인함.	· 지상잔존 저온시험 및 작동 저온 시험 · 지상잔존 고온시험 및 단시간 작동 고온 시험 · 작동 고온시험 · 비행중 냉각 손실시험 · 고도시험 · 감압시험 · 과압시험	· 온도 : -15℃～+85℃ · 고도 : 4,600m～16,800m	· 시험조
2	온도 변화 시험 (Temperature Variation) Sec 5.3	항공기 운항중에 극한 작동 고온도와 극한 작동 저온도 사이의 정상온도 변화중 기기 성능 특성을 확인함.	· 지상잔존 저온시험 및 작동 저온 시험 · 지상잔존 고온시험 및 단시간 작동 고온 시험 · 작동 고온시험	· 온도변화율 : 2℃～10℃/min	· 시험조
3	습도 시험(Humidity) Sec 6.0	기기가 자연 또는 유발된 습도 환경에 견디는 능력을 확인함.	· 표준습도환경 · 엄격한 습도환경 · 외부습도 환경	· 상대습도 95% 이상 · 온도～50℃ · 상대습도 95% 이상 · 온도～65℃ · 상대습도 95% 이상 · 온도～55℃	· 항온항습기
4	운용 충격 및 파괴착륙 안정성 시험 (Operational Shock and Crash Safety) Sec 7.0	기기가 항공기의 정상운용 중에 조우하는 충격에 노출된 후 성능 규격내에서 기능 상태를 검증함.	· 운용충격	· 충격 펄스 : 2.0～12g	· 충격대 · 가속도계
5	진동 시험(Vibration) Sec 8.0	기기에 규정한 진동을 가했을 때 성능 규격에 적합한가를 확인함.	· 표준 진동시험 - 싸인파 진동시험 - 불규칙 진동시험 · 엄격한 진동시험 - 싸인파 진동시험 - 불규칙 진동시험	· 가속도 파워 스펙트럼 밀도(APSD : Acceleration Power Spectral Density) 측정	· 진동발생기 · 진동대 · 제어용 가속도계

번호	제목	시험의 목적	시험의 내역	범위	시험장비
6	방폭성 시험(Explosion Proofness) Sec 9.0	가연성 유체 및 증기에 접촉하는 것이 있는 탑재기기에 대한 요구 사항.	· 차폐가 없는 가연성 유체 또는 증기가 계속 또는 간헐적으로 존재 · 고장의 결과로 인한 가연성 기체 발생 · 가연성 기체발생 확률이 낮은 경우 · 지정방화구역	· 폭발 시연	· 폭발조 · 혼합조
7	방수성 시험(Water Proofness) Sec 10.0	기기가 물방울로 떨어지는 액체의 영향에 견딜 수 있는 지 여부를 확인함. · 카테고리 X : 물떨어짐 없음 · 카테고리 W : 물떨어짐 있음 · 카테고리 R : 비에 노출 됨 · 카테고리 S : 항공기 제빙, 세척, 청정작업시 유체 노출	· 방적시험(Drip proof Test) · 분수시험(Spray Proof Test) · 연속 유수시험(Continuous Stream Proof Test)	· 50℃의 물 사용	· 급수조 · 살수기
8	유체 감수성 시험(Fluid Susceptibility) Sec 11.0	기기의 구성재가 유체 오염 요인물의 해로운 영향에 견딜 수 있는 지를 확인함. · 카테고리 X : 유체 오염 노출되지 않음 · 카테고리 F : 유체 노출	· 분무시험 · 침적시험	· 시험 유체 : 연료, 작동유, 운활유, 용제 및 세척애, 제빙액, 소화제, 살충제, 오니 · 유체의 온도 : 20~150℃	· 열압력 시험조 · 담금조 · 분무기
9	모래 먼지 시험(Sand and Dust) Sec 12.0	공기의 이동에 의해 운반되는 모래와 먼지의 영향에 대해서 기기의 저항성을 확인한다. · 카테고리 X : 노출되지 않음 · 카테고리 D : 노출	· 노출 시험 - 제1사이클: 온도 25℃, 습도 30% - 제2사이클: 온도 55℃, 습도 30%	· 3.5~8.8g/㎥의 농도로 사진 발생 · 입자 : KS A5101에 따름 · 분류속도0.5~2.5 m/s	· 시험조

번호	제목	시험의 목적	시험의 내역	범위	시험장비
10	곰팡이 저항성 시험(Fungus Resistance) Sec 13.0	기기의 재료가 곰팡이 발육에 좋은 조건(온난 대기+무기염 존재)에서 곰팡이의 나쁜 영향을 받는 지 시험. · 카테고리 X : 노출되지 않음 · 카테고리 F : 노출	· 무기염 용액조제 · 혼합포자 현탁액 조제 · 접종 곰팡이 생육 능력의 관리	· 현탁액 · 강제 통기 1㎜s 이내 · 온도30℃, 상대습도 97±2%에서 4시간 상태 조절후 배양	· 시험조
11	염수 분무 시험(Salt Spray) Sec 14.0	염분을 포함한 대기 또는 정규 운전중에 조우하는 염수분무 노출시 기기가 받는 영향 확인. · 카테고리 X : 노출되지 않음 · 카테고리 S : 노출	· 분무 시험	· 염용액 조제	· 노출 시험조 · 염용액 저장조 · 염용액 분무장치 · 온도제어장치 · 가습장치
12	자기 영향 시험(Magnetic Effect) Sec 15.0	장비 기술자가 항공기 내에서의 기기의 적정한 배치를 선정하는 보조수단으로 자기의 영향을 확인.	· 기기의 클래스(Z, A, B, C, X) 결정	· 수평자력 14.4A/m(0.18Gauss)±10%	· 자력계
13	입력전원 시험(Power Input) Sec 16.0	시험 대상기기의 단자에 가하는 전원의 상태를 정하고 적용할 수 있는 관련기기의 시험절차 나열. · 카테고리 A: 1차전원 교류, 직류는 정류기에서 급전, 축전지는 버스에 접속 · 카테고리 B: 엔진구동의 교류발전기, 정류기 또는 직류발전기로 급전 · 카테고리 E: 교류 입력전원만 필요 · 카테고리 Z: 모든 기기	· 정상작동조건(교류) · 정상작동조건(직류) · 이상작동조건(교류) · 이상작동조건(직류)	· 교류전압 : 104~122V · 주파수 : 360~420 Hz · 직류 : 11~32.2V	· 전원장치(교류 및 직류) · 축전지

번호	제목	시험의 목적	시험의 내역	범위	시험장비
14	전압 스파이크 시험(Voltage Spike) Sec 17.0	기기가 교류 또는 직류의 어느 한쪽의 전원 리드선상에 나타나는 점압 스파이크의 영향에 견딜 수 있는 가를 확인함. · 카테고리 A: 고도의 보호 필요 · 카테고리 B: 낮은 보호	· 과도파형 실험.(1분이내 50회 과도전압 부여) - 간헐과도 시험 - 반복과도상태 시험	· 최소 개회로 전압 : 600V	· 과도 파형발생기 · 공심 변압기 · 오실로스코우프
15	움성 주파 전도 방해 감수성-전원입력(폐회로 시험) 시험(Audio Frequency Conducted Susceptibility) Sec 18.0	기기를 항공기에 장착했을 때 그것이 보통 예상되는 크기의 주파수 성분을 받는 지를 확인. · 카테고리 A: 1차전원 교류, 직류는 정류기에서 급전, 축전지는 버스에 접속 · 카테고리 B: 엔진구동의 교류발전기, 정류기 또는 직류발전기로 급전 · 카테고리 E: 교류 입력전원만 필요 · 카테고리 Z: 모든 기기	· 시험회로에 의한 시험	· 신호주파수 : 750~15000Hz	· 교류 또는 직류 전원 · 출력 30W 음성증폭기 · 신호감시기
16	유도신호방해 감수성 시험(Induced Signal Susceptibility) Sec 19.0	기기의 상호접속 회로의 형태가 임의의 레벨의 유도전압을 받아들이는 가를 확인. · 카테고리 Z: 방해가 없는 계통 · 카테고리 A: 방해가 없는 작동이 바람직한 곳 · 카테고리 B: 방해를 허용레벨로 제어하는 곳	· 기기에 유도하는 자계 · 상호접속 케이블에서 유도하는 자계 · 상호접속 케이블에서 유도하는 전계 · 상호접속 케이블에서 유도하는 스파이크	· 적용기기의 성능규격에 적합여부 확인	· 방사기 · 음성주파수 자계 시험장치 · 전원장치 : 400Hz, 20~30A

번호	제목	시험의 목적	시험의 내역	범위	시험장비
17	무선 주파수 방해 감수성(방사 및 전도) 시험(Radio Frequency Susceptibility) Sec 20.0	기기 및 상호 접속용 배선을 임의의 레벨의 무선주파 변조 전력에 노출 시킬 때 전원 회로상 및 인터페이스 회로 배선상의 방사 무선 주파수 전자계 또는 주입 프로브 유도에 의해서 기기가 성능 시방서의 범위 내에서 작동하고 있는 지를 확인. · 카테고리 W 및 Y: 가혹한 전자환경에 장착 · 카테고리 V: 중정도 환경 · 카테고리 U: 부분적으로 보호된 환경 · 카테고리 T: 충분히 보호된 환경 · 카테고리 X: 전자적 영향이 적은 환경	· 전도방해 감수성 시험(CS) - 10㎑~400㎒ · 방사방해 감수성 시험(RS) - 30㎒~18㎓	· 카테고리 Y: 200V/m · 카테고리 W: 100V/m · 카테고리 V: 50V/m · 카테고리 U: 20V/m · 카테고리 T: 5V/m · 카테고리 X:	· 접합시험장치 · 차폐실(무향실)
18	무선주파수 에너지 방사 시험(Emission of Radio Frequency Energy) Sec 21.0	기기가 규정 레벨을 초과하지 않고 바람직하지 않은 무선주파수 잡음을 방사하지 않는 것을 확인함. · 카테고리 Z: 방해가 없는 계통 · 카테고리 A: 방해가 없는 작동이 바람직한 곳 · 카테고리 B: 방해를 허용 레벨로 제어하는 곳	· 무선주파수 전도 방해 · 무선주파수 방사 방해	· 카테고리 Z: · 카테고리 A · 카테고리 B:	· 시험기기 배치

번호	제목	시험의 목적	시험의 내역	범위	시험장비
19	낙뢰 유도 과도 방해 감수성 시험(Lightning Induced Transient Susceptibility) Sec 22.0	개별 기기 시방서에 정한 낙뢰가 일으키는 영향에 기기가 견딜 수 있는 것을 확인. · 카테고리 J: 부분적으로 보호된 환경 · 카테고리 K: 중환경(조종석 등) · 카테고리 L&M: 가혹한 환경 · 카테고리 X: 낙뢰 근소한 영향	· 장시간파 시험 · 단시간파 시험 · 감쇄 사인파	· 전압: 10kV 이상 · V_p: 125~3,200V · I_p: 10~320A	· 파형발생기
20	낙뢰의 직접영향(Lightning Direct Effects) Sec 23.0	항공기 외부에 장착되는 전기 및 전자 기기가 과혹한 뇌격의 직접영향에 견디는 것을 확인함. · 카테고리 1A: 낙뢰구역 1A에 장착한 기기 · 카테고리 1B: · 카테고리 2A: · 카테고리 2B: · 카테고리 3:	· 고전압 시험 · 대전류 시험	· 전압: 250kV~1500kV · 전류: - 성분A: 200kA - 성분B: 2kA - 성분C: 200A - 성분D: 100kA	· LTF
21	착빙 시험(Icing) Sec 24.0	온도, 고도 및 습도가 급속하게 변화하는 상태에서 조우할 수 있는 착빙 조건에 노출되었을 때 작동하는 기기의 성능 특성을 확인함. · 카테고리 A: 극단 저온 · 카테고리 B: 가동 부품을 가진 기기 · 카테고리 C: 자유빙 및 동결 가능성 지역 · 카테고리 X: 영향 없음	· 착빙시험 - 카테고리 A - 카테고리 B - 카테고리 C	· 카테고리 A - 온도: -10~30℃ - 습도 95% 이상 · 카테고리 B - 온도: -20℃ · 카테고리 C - 온도: -20℃	· 시험조

제 6 절 시험평가용 김발 운동모델 및 알고리즘 기초연구

1. 운동판 운동 방정식

가. 가정 사항

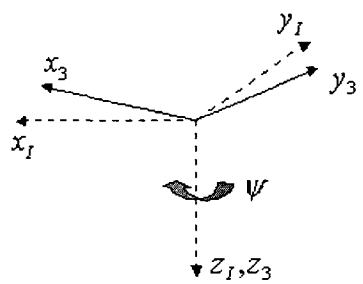
- 운동판은 3개의 개별 테이블로 이루어져있다 : Inner, middle, outer table로 구성되어있음.
- 각 테이블은 테이블 좌표계에 대해 대칭이므로, 관성 모멘트 행렬은 대각행렬의 형태를 띈다.
- 운동판의 구성상 내부의 회전하는 성분이 포함된다. 이에 따라 middle, outer table 좌표계에 대해 관성 모멘트 행렬이 변하게 된다. 본 과제에서는 이에 대한 영향을 무시하도록 한다. 관성 모멘트 행렬은 고정된 값으로 정의한다.
- 운동 테이블을 이루는 재질은 두랄루민을 사용하며, 고르게 분포하고, 결합에 의한 영향은 없다고 한다. 즉, 모터, 기어, 결합부에 따른 관성 모멘트에 대한 영향은 없다고 가정한다.
- 각 테이블의 C.G는 각 테이블 좌표계의 원점에 위치한다고 가정한다. 이는 Balancing에 의해 이루어진다고 가정한다.

나. 좌표계

관성 좌표계와 각 테이블에 대한 좌표계를 정의한다. 관성좌표계와 최종 항공기의 운동과 일치하는 Inner table 좌표계의 관계는 3→2→1의 순서로 이루어진다. 자세 변환 관계를 표현하기 위해 DCM(Direction Cosine Matrix)를 사용하여 정의하면 다음과 같다.

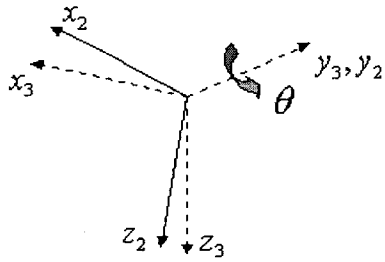
- 관성 좌표계에서 Outer table 좌표계로의 변환.

$$D_3^I = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



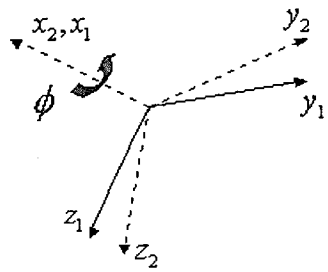
- Outer table에서 Middle table로의 변환

$$D_2^3 = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$



- Middle table에서 Inner table로의 변환.

$$D_1^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}$$



다. 각 테이블 좌표계에 대한 각속도 및 모멘트 식

(1) Outer Table

이후 아래첨자로 3을 사용하면 outer table에 대해 정의된 값이라 한다.

- Angular velocity ω_3

Outer table은 $\dot{\psi}$의 각속도를 가지고 z축에 대해 회전한다. 따라서 관성 좌표계에 대한 outer table의 회전 각속도는 다음과 같다.

$$\omega_3 = \begin{bmatrix} \omega_{3x} \\ \omega_{3y} \\ \omega_{3z} \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + D_3^I \begin{bmatrix} \omega_{Ix} \\ \omega_{Iy} \\ \omega_{Iz} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}$$

여기서, 관성 좌표계는 회전하지 않으므로, $\omega_I = 0$ 이다.

- Angular Momentum H_3

각 운동량은 다음 식을 이용해 구할 수 있다.

$$H_3 = I_3 \cdot \omega_3$$

여기서, 1의 가정에서 각 테이블은 각 평면에 대해 대칭이므로, 대각 행렬이다 따라서 각 운동량은 다음과 같다.

$$H_3 = I_3 \cdot \omega_3 = \begin{bmatrix} I_{3x} & 0 & 0 \\ 0 & I_{3y} & 0 \\ 0 & 0 & I_{3z} \end{bmatrix} \begin{bmatrix} \omega_{3x} \\ \omega_{3y} \\ \omega_{3z} \end{bmatrix}$$
$$= \begin{bmatrix} I_{3x}\omega_{3x} \\ I_{3y}\omega_{3y} \\ I_{3z}\omega_{3z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ I_{3z}\omega_{3z} \end{bmatrix}$$

- 모멘트

모멘트는 다음 식을 만족한다.

$$M_3 = \dot{H}_3 + \omega_3 \times H_3$$

따라서 outer table에 작용하는 모멘트는 다음과 같다.

$$M_3 = \dot{H}_3 + \omega_3 \times H_3$$
$$= \begin{bmatrix} 0 \\ 0 \\ I_{3z}\dot{\omega}_{3z} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega_{3z} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ I_{3z}\omega_{3z} \end{bmatrix}$$
$$= \begin{bmatrix} 0 \\ 0 \\ I_{3z}\dot{\omega}_{3z} \end{bmatrix}$$

(2) Middle table

- Angular velocity ω_2

$$\omega_2 = \begin{bmatrix} \omega_{2x} \\ \omega_{2y} \\ \omega_{2z} \end{bmatrix} = \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + D_2^3 \begin{bmatrix} \omega_{3x} \\ \omega_{3y} \\ \omega_{3z} \end{bmatrix}$$
$$= \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \omega_{3z} \end{bmatrix}$$
$$= \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} -\sin\theta\omega_{3z} \\ 0 \\ \cos\theta\omega_{3z} \end{bmatrix}$$

- Angular Momentum H_2

$$H_2 = I_2 \cdot \omega_2$$
$$= \begin{bmatrix} I_{2x}\omega_{2x} \\ I_{2y}\omega_{2y} \\ I_{2z}\omega_{2z} \end{bmatrix} = \begin{bmatrix} -\sin\theta I_{2x}\omega_{3z} \\ \dot{\theta} I_{2y} \\ \cos\theta I_{2z}\omega_{3z} \end{bmatrix}$$

- 모멘트

$$M_2 = \dot{H}_2 + \omega_2 \times H_2$$
$$= \begin{bmatrix} I_{2x}\dot{\omega}_{2x} \\ I_{2y}\dot{\omega}_{2y} \\ I_{2z}\dot{\omega}_{2z} \end{bmatrix} + \begin{vmatrix} i & j & k \\ \omega_{2x} & \omega_{2y} & \omega_{2z} \\ I_{2x}\omega_{2x} & I_{2y}\omega_{2y} & I_{2z}\omega_{2z} \end{vmatrix}$$
$$= \begin{bmatrix} I_{2x}\dot{\omega}_{2x} \\ I_{2y}\dot{\omega}_{2y} \\ I_{2z}\dot{\omega}_{2z} \end{bmatrix} + \begin{bmatrix} (I_{2z} - I_{2y})\omega_{2y}\omega_{2z} \\ (I_{2x} - I_{2z})\omega_{2z}\omega_{2x} \\ (I_{2y} - I_{2x})\omega_{2x}\omega_{2y} \end{bmatrix}$$

(3) Inner table

- Angular velocity ω_1

$$\omega_1 = \begin{bmatrix} \omega_{1x} \\ \omega_{1y} \\ \omega_{1z} \end{bmatrix}$$
$$= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + D_1^2 \begin{bmatrix} \omega_{2x} \\ \omega_{2y} \\ \omega_{2z} \end{bmatrix}$$
$$= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \omega_{2x} \\ \omega_{2y} \\ \omega_{2z} \end{bmatrix}$$
$$= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \omega_{2x} \\ \cos\phi\omega_{2y} + \sin\phi\omega_{2z} \\ -\sin\phi\omega_{2y} + \cos\phi\omega_{2z} \end{bmatrix}$$

- Angular Momentum H_1

$$H_1 = I_1 \cdot \omega_1$$
$$= \begin{bmatrix} I_{1x}\omega_{1x} \\ I_{1y}\omega_{1y} \\ I_{1z}\omega_{1z} \end{bmatrix}$$

- 모멘트

$$
\begin{aligned}
M_1 &= \dot{H}_1 + \omega_1 \times H_1 \\
&= \begin{bmatrix} I_{1x}\dot{\omega}_{1x} \\ I_{1y}\dot{\omega}_{1y} \\ I_{1z}\dot{\omega}_{1z} \end{bmatrix} + \begin{vmatrix} i & j & k \\ \omega_{1x} & \omega_{1y} & \omega_{1z} \\ I_{1x}\omega_{1x} & I_{1y}\omega_{1y} & I_{1z}\omega_{1z} \end{vmatrix} \\
&= \begin{bmatrix} I_{1x}\dot{\omega}_{1x} \\ I_{1y}\dot{\omega}_{1y} \\ I_{1z}\dot{\omega}_{1z} \end{bmatrix} + \begin{bmatrix} (I_{1z} - I_{1y})\omega_{1y}\omega_{1z} \\ (I_{1x} - I_{1z})\omega_{1z}\omega_{1x} \\ (I_{1y} - I_{1x})\omega_{1x}\omega_{1y} \end{bmatrix}
\end{aligned}
$$

라. 운동 시뮬레이션을 위해 정의해야할 관계식

운동판의 운동을 표현하기 위해서는 기본적으로 $p, q, r, \phi, \theta, \psi$ 의 6개의 state에 대한 시간에 따른 변화를 구해야한다. 그리고 table의 구동이 Motor에 의해 이루어짐으로, 이에 대한 관계 또한 정의되어야 한다.

운동판에 탑재하여 테스트할 물체의 각속도를 p, q, r이라 할 경우, 이는 inner table의 각속도 ω_1과 일치한다. 이후 정의에서는 inner table의 각속도를 이용하여 관계식을 정의하도록 한다.

(1) 탑재물의 각속도 v.s. Euler Angle의 변화율.

탑재물의 각속도를 ω라 할 경우, 이는 inner table의 각속도와 일치한다.

$$
\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \omega_1 = \begin{bmatrix} \omega_{1x} \\ \omega_{1y} \\ \omega_{1z} \end{bmatrix}
$$

1과 2에서 정의한 순서로 관성 좌표계에 대해 inner table의 자세를 표현함으로, 탑재물 각속도와 Euler Angle의 관계는 다음과 같다.

$$
\begin{aligned}
\omega &= \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \omega_{1x} \\ \omega_{1y} \\ \omega_{1z} \end{bmatrix} \\
&= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + D_1^2 \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + D_1^2 D_2^3 \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}
\end{aligned}
$$

(2) Motor축에서의 각속도와 inner table 각속도의 관계식.

가동 토크가 세 개의 Motor에 의해 주어짐으로써, 이에 의한 각속도 성분과 탑재

물의 각속도인 inner table의 각속도에 대한 관계식이 구해져야한다. 각 모터는 각 좌표계에 대해 한 축에 고정되어 사용된다. 본 시스템에서는 기어의 영향을 무시할 경우, 모터 축과 일치하는 가속도 성분은 다음과 같다.

- inner table 모터의 각속도 : ω_{1x}
- middle table 모터의 각속도 : ω_{2y}
- outer table 모터의 각속도 : ω_{3z}

위 세 개의 각속도가 변하고 이에 따라 inner table의 자세가 결정됨으로, 이에 대한 관계를 구해야 한다. 즉, ($\omega_{1x}, \omega_{1y}, \omega_{1z}$) 와 ($\omega_{1x}, \omega_{2y}, \omega_{3z}$)의 관계식을 구해야한다.
우선 ω_{1y}는 다음과 같다.

$$\begin{aligned}\omega_{1y} &= \cos\phi\omega_{2y} + \sin\phi\omega_{2z} \\ &= \cos\phi\omega_{2y} + \sin\phi\cos\theta\omega_{3z}\end{aligned}$$

그리고 ω_{1z}는 다음과 같다.

$$\begin{aligned}\omega_{1z} &= -\sin\phi\omega_{2y} + \cos\phi\omega_{2z} \\ &= -\sin\phi\omega_{2y} + \cos\phi\cos\theta\omega_{3z}\end{aligned}$$

따라서 다음의 관계식을 만족한다.

$$\begin{bmatrix}\omega_{1x} \\ \omega_{1y} \\ \omega_{1z}\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta\end{bmatrix}\begin{bmatrix}\omega_{1x} \\ \omega_{2y} \\ \omega_{3z}\end{bmatrix}$$

(3) 각 모터 축에 대한 모멘트 관계식.

본 시스템에서 정의한 모멘트 식에 의하면 총 9개의 Equation이 구해진다. 이 식들을 이용하여, 모터와의 연결에 의한 제어를 수행할 경우, 세 개의 모터 축에 관한 식으로 변환되어야한다. 이 경우, inner table에 의해 middle, outer table에 영향을 주게 되고, middle table에 의해 outer table에 모멘트 값이 변하게 된다. 즉 다음의 관계식을 구해야 한다.

$M_{dx} = M_{1x}$

$M_{dy} = M_{2y} + D_2^1 M_1$ 의 y 방향 성분

$M_{dz} = M_{3z} + (D_3^1 M_1 + D_3^2 M_2)$의 z 방향 성분

여기서, M_{dx}, M_{dy}, M_{dz}는 inner, middle, outer table을 돌리기 위한 모터의 토크이다. 이에 대해서는 5에서 유도할 것이다.

(4) 모터 회전축에 대한 모멘트 관계식

(가) Inner table에 대한 모멘트 식

inner table에 의한 영향만 포함된다. 이는 M_1의 x 방향 성분이다.

$$I_{1x}\dot{\omega}_{1x} + \omega_{1y}\omega_{1z}(I_{1z} - I_{1y}) = M_{dx}$$

(나) Middle table에 대한 모멘트 식

Middle table에 대해 적용되는 모멘트 식 M_2에서의 y방향 성분과 함께, inner table에 의한 영향 또한 포함된다.

$M_{2y} + D_2^1 M_1$의 y 방향 성분 = M_{dy}

- Middle table에 의한 모멘트 M_2의 y방향 성분.

$$M_{2y} = I_{2y}\dot{\omega}_{2y} + \omega_{2z}\omega_{2x}(I_{2x} - I_{2z})$$

- Inner table에 의한 모멘트 영향.

$$D_2^1 M_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} I_{1x}\dot{\omega}_{1x} + \omega_{1y}\omega_{1z}(I_{1z} - I_{1y}) \\ I_{1y}\dot{\omega}_{1y} + \omega_{1z}\omega_{1x}(I_{1x} - I_{1z}) \\ I_{1z}\dot{\omega}_{1z} + \omega_{1x}\omega_{1y}(I_{1y} - I_{1x}) \end{bmatrix}$$

$$= \begin{bmatrix} I_{1x}\dot{\omega}_{1x} + \omega_{1y}\omega_{1z}(I_{1z} - I_{1y}) \\ \cos\phi\{I_{1y}\dot{\omega}_{1y} + \omega_{1z}\omega_{1x}(I_{1x} - I_{1z})\} - \sin\phi\{I_{1z}\dot{\omega}_{1z} + \omega_{1x}\omega_{1y}(I_{1y} - I_{1x})\} \\ \sin\phi\{I_{1y}\dot{\omega}_{1y} + \omega_{1z}\omega_{1x}(I_{1x} - I_{1z})\} + \cos\phi\{I_{1z}\dot{\omega}_{1z} + \omega_{1x}\omega_{1y}(I_{1y} - I_{1x})\} \end{bmatrix}$$

여기서, y 방향 성분만을 고려하면 Middle table의 모터 축에 작용하는 모멘트는 다음과 같다.

$$M_{dy} = I_{2y}\dot{\omega}_{2y} + \omega_{2z}\omega_{2y}(I_{2x} - I_{2z}) + \cos\phi\{I_{1y}\dot{\omega}_{1y} + \omega_{1z}\omega_{1x}(I_{1x} - I_{1z})\} - \sin\phi\{I_{1z}\dot{\omega}_{1z} + \omega_{1x}\omega_{1y}(I_{1y} - I_{1x})\}$$

(다) Outer table에 대한 모멘트 식

Outer table은 inner table과 middle table에 의해 영향을 받게 된다. 따라서 outer table에 대한 모터 회전축에서의 모멘트 식은 다음과 같이 이루어진다.

$M_{dz} = M_{3z} + (D_3^2 M_2 + D_3^1 M_1)$ 의 z 방향 성분

- Outer table의 모멘트 식에서의 z방향 성분

$M_{3z} = I_{3z}\dot{\omega}_{3z}$

- Middle table에 의한 영향

$$D_3^2 M_2 = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} I_{2x}\dot{\omega}_{2x} + (I_{2z} - I_{2y})\omega_{2y}\omega_{2z} \\ I_{2y}\dot{\omega}_{2y} + (I_{2x} - I_{2z})\omega_{2z}\omega_{2x} \\ I_{2z}\dot{\omega}_{2z} + (I_{2y} - I_{2x})\omega_{2x}\omega_{2y} \end{bmatrix}$$

위 식을 정리하고, z 방향 성분만을 구하면 다음과 같다.

$$M_3^2 = -\sin\theta I_{2x}\dot{\omega}_{2x} - (I_{2z} - I_{2y})\omega_{2y}\omega_{2z}\sin\theta + \cos\theta I_{2z}\dot{\omega}_{2z} + \cos\theta(I_{2y} - I_{2x})\omega_{2x}\omega_{2y}$$

- Inner table에 의한 영향.

$$D_3^2 D_2^1 M_1 = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} M_1$$

$$= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} D_2^1 M_1$$

여기서, $D_2^1 M_1$은 5.2에서 구한 식을 사용한다. 이를 정리하여 z 방향 성분만을 구하면 다음과 같다.

$$M_3^1 = -\sin\theta\{I_{1x}\dot{\omega}_{1x} + \omega_{1y}\omega_{1z}(I_{1z} - I_{1y})\} + \cos\theta\sin\phi\{I_{1y}\dot{\omega}_{1y} + \omega_{1z}\omega_{1x}(I_{1x} - I_{1z})\}$$
$$+ \cos\theta\cos\phi\{I_{1z}\dot{\omega}_{1z} + \omega_{1x}\omega_{1y}(I_{1y} - I_{1x})\}$$

따라서 outer table의 모터축에 작용하는 모멘트는 (1), (2), (3)식의 합이다.

마. 모멘트식 정리하기

3개의 모멘트 관계식을 각 회전축의 각속도 성분 $\omega_{1x}, \omega_{2y}, \omega_{3z}$를 이용하여 정리해보자. 그리고 일반적으로, 각속도의 곱셈 항은 작은 값을 가지므로, 식의 단순화를 위해 무시하도록 하자.

(1) Inner table의 모터축 관계식

inner table의 모멘트 식은 다음과 같다.

$$I_{1x}\dot{\omega}_{1x} + \omega_{1y}\omega_{1z}(I_{1z} - I_{1y}) = M_{dx}$$

여기서, 각속도간의 곱은 이외의 항보다 작다는 가정 하에 무시하면 위 식은 다음과 같다.

$$I_{1x}\dot{\omega}_{1x} = M_{dx}$$

(2) Middle table의 모멘트 관계식.

- Middle table에 의한 모멘트 M_2의 y방향 성분.

$$M_{2y} = I_{2y}\dot{\omega}_{2y} + \omega_{2z}\omega_{2x}(I_{2x} - I_{2z})$$

위 식에서 각 속도 곱을 무시하면, $M_{2y} = I_{2y}\dot{\omega}_{2y}$ 이다.

- Inner table에 의한 모멘트 영향.

$$M_{dy} = \cos\phi\{I_{1y}\dot{\omega}_{1y} + \omega_{1z}\omega_{1x}(I_{1x} - I_{1z})\} - \sin\phi\{I_{1z}\dot{\omega}_{1z} + \omega_{1x}\omega_{1y}(I_{1y} - I_{1x})\}$$

위 식에서 각속도간의 곱인 항을 무시하면 아래와 같다.

$$M_{dy} = \cos\phi I_{1y}\dot{\omega}_{1y} - \sin\phi I_{1z}\dot{\omega}_{1z}$$

여기서, ω_{1y}와 ω_{1z}를 ω_{2y}와 ω_{3z}에 대한 항으로 바꿔보자.
각속도 관계식을 이용하면 아래와 같다.

$$\begin{aligned}\dot{\omega}_{1y} &= \frac{d}{dt}\{\cos\phi\omega_{2y} + \sin\phi\omega_{2z}\} \\ &= \cos\phi\dot{\omega}_{2y} + \sin\phi\dot{\omega}_{2z} - \sin\phi\dot{\phi}\omega_{2y} + \cos\phi\dot{\phi}\omega_{2z}\end{aligned}$$

여기서, 각속도의 곱인 항을 무시하면 아래와 같다.

$$\dot{\omega}_{1y} = \cos\phi\dot{\omega}_{2y} + \sin\phi\dot{\omega}_{2z}$$

그리고 $\omega_{2z} = \cos\theta\omega_{3z}$ 이므로, $\dot{\omega}_{2z} = \cos\theta\dot{\omega}_{3z} - \sin\theta\dot{\theta}\omega_{3z} \simeq \cos\theta\dot{\omega}_{3z}$ 를 이용하면 아래와 같다.

$$\dot{\omega}_{1y} = \cos\phi\dot{\omega}_{2y} + \sin\phi\cos\theta\dot{\omega}_{3z}$$

ω_{1z}의 경우는 아래와 같다.

$$\begin{aligned}\dot{\omega}_{1z} &= \frac{d}{dt}\{-\sin\omega_{2y} + \cos\phi\omega_{2z}\} \\ &= -\sin\phi\dot{\omega}_{2y} + \cos\dot{\omega}_{2z} - \cos\phi\dot{\phi}\omega_{2y} - \sin\phi\dot{\phi}\omega_{2z}\end{aligned}$$

여기서, 각속도의 곱인 항을 무시하면, $\dot{\omega}_{1z} = -\sin\phi\dot{\omega}_{2y} + \cos\phi\dot{\omega}_{2z}$ 이다.
따라서 ω_{2z}와 ω_{3z}와의 관계식을 이용하여 정리하면 아래와 같다.

$$\omega_{1z} = -\sin\phi\dot{\omega}_{2y} + \cos\phi\cos\theta\dot{\omega}_{3z}$$

위의 결과를 이용하여 Middle table에 대한 모멘트 식을 정리하면 다음과 같다.

$$M_{dy} = I_{2y}\dot{\omega}_{2y} + \cos\phi I_{1y}\{\cos\phi\dot{\omega}_{2y} + \sin\phi\cos\theta\dot{\omega}_{3z}\} - \sin\phi I_{1z}\{-\sin\phi\dot{\omega}_{2y} + \cos\phi\cos\theta\dot{\omega}_{3z}\}$$
$$= (I_{2y} + \cos^2\phi I_{1y} + \sin^2\phi I_{1z})\dot{\omega}_{2y} + \cos\phi\sin\phi\cos\theta(I_{1y} - I_{1z})\dot{\omega}_{3z}$$

(3) Outer table에 관한 모멘트 관계식

- middle table에 의한 영향.

$$M_3^2 = -\sin\theta I_{2x}\dot{\omega}_{2x} - (I_{2z} - I_{2y})\omega_{2y}\omega_{2z}\sin\theta + \cos\theta I_{2z}\dot{\omega}_{2z} + \cos\theta(I_{2y} - I_{2x})\omega_{2x}\omega_{2y}$$

위 식에서, 각속도 곱의 항을 무시하고, 위에서 정의한 각속도 관계식과 함께 아래 수식을 이용하여 정리하면 다음과 같다.

$$\omega_{2x} = -\sin\theta\omega_{3z}$$
$$\dot{\omega}_{2x} = -\sin\theta\dot{\omega}_{3z} - \cos\theta\dot{\theta}\omega_{3z}$$
$$\simeq -\sin\theta\dot{\omega}_{3z}$$

$$M_3^2 = -\sin\theta I_{2x}\dot{\omega}_{2x} + \cos\theta I_{2z}\dot{\omega}_{2z}$$
$$= \sin^2\theta I_{2x}\dot{\omega}_{3z} + \cos^2\theta I_{2z}\dot{\omega}_{3z}$$

- Inner table에 의한 영향.

$$M_3^1 = -\sin\theta\{I_{1x}\dot{\omega}_{1x} + \omega_{1y}\omega_{1z}(I_{1z} - I_{1y})\} + \cos\theta\sin\phi\{I_{1y}\dot{\omega}_{1y} + \omega_{1z}\omega_{1x}(I_{1x} - I_{1z})\}$$
$$+ \cos\theta\cos\phi\{I_{1z}\dot{\omega}_{1z} + \omega_{1x}\omega_{1y}(I_{1y} - I_{1x})\}$$

위 식에서 각속도 곱의 항을 무시하고, 정리하면 다음과 같다.

$$M_3^1 = -\sin\theta I_{1x}\dot{\omega}_{1x} + \cos\theta\sin\phi I_{1y}\dot{\omega}_{1y} + \cos\theta\cos\phi I_{1z}\dot{\omega}_{1z}$$

$$= -\sin\theta I_{1x}\dot{\omega}_{1x} + \cos\theta\sin\phi I_{1y}\cos\phi\dot{\omega}_{2y} + \sin^2\phi\cos^2\theta I_{1y}\dot{\omega}_{3z}$$

$$+ \cos^2\theta\cos^2\phi I_{1z}\dot{\omega}_{3z} - \cos\theta\cos\phi\sin\phi I_{1z}\dot{\omega}_{2y}$$

$$= -\sin\theta I_{1x}\dot{\omega}_{1x} + \cos\theta\sin\phi\cos\phi(I_{1y} - I_{1z})\dot{\omega}_{2y} + \cos^2\theta\{\cos^2\phi I_{1z} + \sin^2\phi I_{1y}\}\dot{\omega}_{3z}$$

따라서 위 정리 식을 이용하면, Outer table에 대한 모멘트 관계식은 다음과 같다.

$$M_{dz} = M_{3z} + M_3^2 + M_3^1$$

$$= I_{3z}\dot{\omega}_{3z} + \sin^2\theta I_{2x}\dot{\omega}_{3z} + \cos^2\theta I_{2z}\dot{\omega}_{3z}$$

$$-\sin\theta I_{1x}\dot{\omega}_{1x} + \cos\theta\sin\phi\cos\phi(I_{1y} - I_{1z})\dot{\omega}_{2y} + \cos^2\theta\{\cos^2\phi I_{1z} + \sin^2\phi I_{1y}\}\dot{\omega}_{3z}$$

2. 모터와 기어

가. 모터 제어

(1) 모터 Parameter 측정하기.

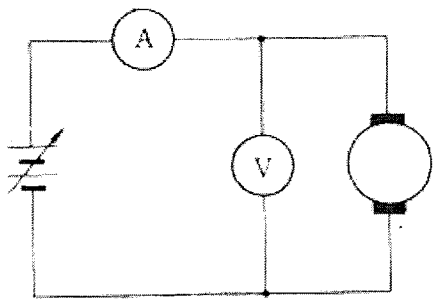
(가) 모터 운동 표현을 위한 중요 Parameters

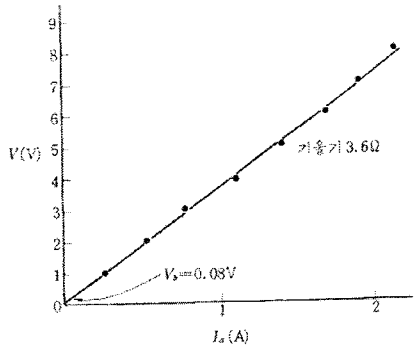
- 내부 저항 : 전기자 저항으로써 R_a
- 모터 상수 : 역기전력 상수 또는 토크 상수라고 불리는 것으로 K라 함.
- 기계적인 시정수 : τ_M

(나) 내부 저항의 측정

- 모터가 동작하지 않도록 (샤프트가 움직이지 않는 상태)하고 다음 그림의 측정회로에 접속해서 전압과 전류의 관계를 조사함.
- 전류와 전압에 대한 관계 그래프를 구함.
- 이때의 직선의 기울기가 내부 저항 R_a 임.

(다) 모터 상수의 측정



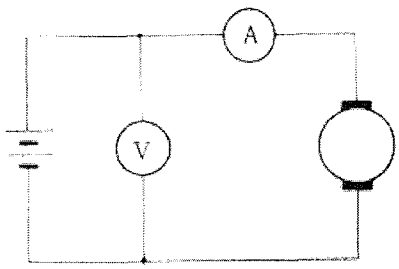


1) 무부하 운전 데이터와 회전수로부터 구하기.

- 적당한 전압 V에 의해서 회전할 때의 회전 속도를 비 접착식 회전계로 측정. 그리고 이때의 전류를 측정함.
- 역기전력 상수 K_E는 다음과 같다.

$$K_E = \frac{V - R_a I_a}{N} \quad (\mathrm{V/rpm})$$

여기서, R_a는 2에서 구한 내부 저항 (Ω) , N은 rpm으로 표시되는 회전속도, I_a는 모터의 전기자에 흐르는 전류이다.



- 이때 모터 상수 K는 다음과 같다.

$K = 9.54K_E$ (V · s/rad or N · m)

2) 회전력 측정에 의한 방법.

- 아래 그림과 같은 장치를 준비하고 적당한 전압에서 모터를 회전시킴.

- 이때의 회전력은 다음 식을 만족한다.

$$T = r(m - g) \quad (\mathrm{g \cdot cm})$$

여기서, r은 pulley의 반지름(cm), g 는 저울의 눈금(g), m은 추의 무게(g)이다.

- 이때의 토크 상수는 다음과 같다.

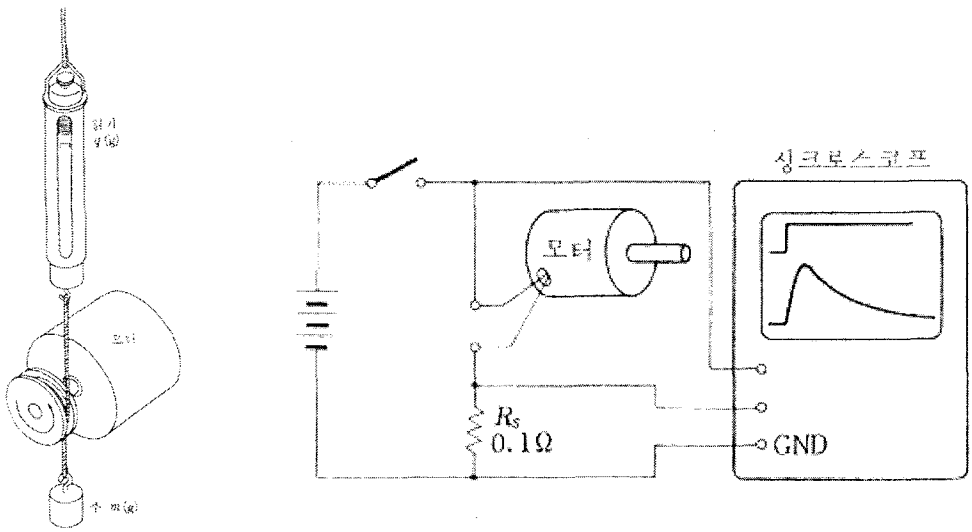
$$K_T = \frac{T}{1000 I_a} \qquad (\mathrm{kg \cdot cm/A})$$

- 모터 상수 K는 다음과 같다.

$$K = 9.81 \times 10^{-2} K_T \ (\mathrm{N \cdot m \ or \ V \cdot s/rad})$$

(라) 기계적 시정수 구하기

- 아래 그림과 같은 측정 장치를 준비한다.

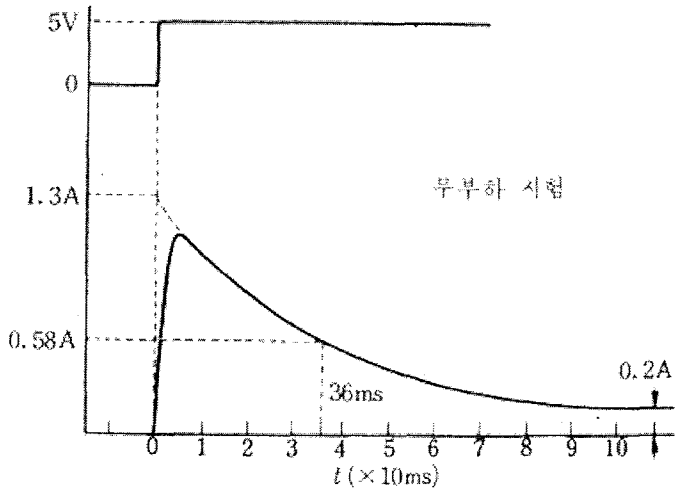


여기서 전류 관측용의 저항 R_s는 모터 내부 저항 R_a보다도 작은 것으로 하고 여기에서는 스위치를 On시킨 직후의 전류 파형을 관측하도록 한다.

- 다음의 저항을 구한다.

$$R_D = \frac{V - i_0}{R_a - R_s}$$

여기서, i_0는 다음 그림 파형의 최종 전류이다.



- τ는 다음 식을 만족한다.

$$\tau = \left(\frac{R_a}{R_a + R_D} + \frac{R_D}{R_a + R_D} \times \frac{1}{2.72} \right) \times 100\ (\%)$$

스위치 On시의 과도전류의 τ가 되는 시간이 기계적인 시정수가 된다.

나. SM-3657 특성을 표현하는 Parameter 구하기

(1) DC 모터의 기본 방정식 (전기자 모터)

$$E_b = L_a \cdot \frac{di_a}{dt} + R_a \cdot i_a + E_c$$

$$E_c = K_e \cdot \omega_m$$

$$T = i_a \cdot K_t$$

$$J\frac{d\omega_m}{dt} = K_t \cdot i_a - T_L$$

여기서, E_b는 인가전압이고, i_a는 모터를 흐르는 전류이다.

위 식에서 DC 모터의 운동을 표현하기 위해 필요한 parameter는 다음과 같음을 알 수 있다.

R_a , L_a , K_e , K_t , J

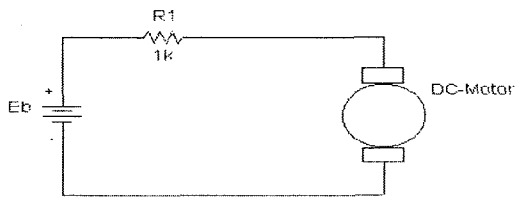
(2) 전기자 저항 R_a 구하기.

DC 모터의 전압 관계식은 다음과 같다.

$$E_b = L_a \cdot \frac{di_a}{dt} + R_a \cdot i_a + K_e \cdot \omega_m$$

위 식에서 만약 모터의 회전축을 고정시켰을 경우, steady state에서의 운동을 구하면, 즉 $\omega_m = 0$, $\frac{di_a}{dt} = 0$ 이면 전압 관계식은 다음과 같다.

$$E_b = R_a \cdot i_a \qquad \Rightarrow \qquad R_a = \frac{E_b}{i_a}$$



위와 같이 회로를 구성한 경우, 이 회로를 흐르는 전류를 i 라 할 경우, 저항 R_1에 걸리는 전압을 v_1이라고 하자.
이때 본 회로를 흐르는 전류 i는 다음과 같다.

$$i = \frac{E_b}{R_1 + R_2}$$

그리고 저항 R_1에 걸리는 전압 v_1는 다음식을 만족한다.

$$v_1 = i \cdot R_1 = \frac{R_1}{R_1 + R_2} E_b$$

따라서 전기자 저항 R_a는 다음과 같다.

$$R_a = \frac{E_b - v_1}{v_1} R_1$$

(3) 토크 상수 K_t 구하기

인가전압에 대해 모터가 정상 상태에 도달한 경우, $\frac{d\omega_m}{dt} = 0$ 이므로, 위 관계식에서 토크 상수를 구할 수 있다.

$$K_t \cdot i_a = T_L$$

위 식에서 T_L는 모터 축에 걸어주는 인가 토크로써 DC 모터 제원표의 x축을 이루는 값이고, 이에 따라 전기자 전류 또한 구할 수 있다. 두 변수의 관계에서 그 기울기가 토크상수이다.

라. 역 기전력 상수 K_e 구하기

무부하 조건에서 모터가 정상상태에 도달 한 경우 다음 조건을 만족한다.

$$\frac{di_a}{dt} = 0 \ , \quad T_L = 0$$

위 조건에 의해 전압 관계식은 다음과 같이 정리할 수 있다.

$$E_b = R_a \cdot i_a + K_e \cdot \omega_m$$

따라서 역 기전력 상수는 다음과 같다.

$$K_e = \frac{E_b - R_a \cdot i_a}{\omega_m}$$

DC 모터의 제원표를 위 식에 대입해서 역 기전력 상수를 구할 수 있다.

(4). 모터 회전자 관성 모멘트 J와 전기자 인덕턴스 L_a 구하기

위 두 값의 경우, J는 회전속도 N에 대해, 인덕턴스 L_a는 전기자 전류에 대해 응답특성에 영향을 주는 값으로 제원표 상에서는 구할 수 없다.
회전자 관성 모멘트의 경우, 코일 뭉치로 가정하여 모터의 전체 지름에 대한 일정 비율로 결정하여 간략히 구할 수 도 있을 것으로 본다. 정상상태 값에는 큰 영향이

없으므로, 시뮬레이션 결과에 의해 적절한 응답특성을 가지도록 선택할 수 있을 것으로 본다.

(5). SM-3657 DC 모터의 Parameters

2 ~ 5에서 언급한 방법을 이용하여 운동판의 구동을 위해 사용할 DC모터 SM-3657의 특성치를 구해보았다.

- 전기자 저항 R_a

현재 실험실에 구비하고 있는 DC 모터의 경우 테이블에 결합되어있는 상태이고, 여분의 모터가 없으므로, 우선 직렬로 연결된 두 개의 모터에 대해 저항을 구해보았다. 즉, 결과로 나온 저항의 반이 실제 DC 모터 하나의 전기자 저항이 된다. 50kΩ의 저항을 모터에 직렬로 연결하고, 3.02V와 5.05V를 연결할 경우, 회전축 고정 조건에서 모터에 인가되는 전압은 0.38mV 와 0.64 mV 이다.
따라서 이 회로를 흐르는 전류를 구하고, 이에 의해 모터의 전기자 저항을 구하면 약 3.1 Ω 이다. 저항측정기로 직접 모터에 연결한 결과 약 3.5Ω이 나왔다. 따라서 회로상의 저항 등을 고려할 경우, 전기자 저항 값으로 약 3.3Ω 근방의 값으로 잡으면 될 것 같다.

- 토크 상수 K_t

제원 표에서 rated load 항의 값을 이용해서 토크상수를 구할 수 있다.

$$T_L = 170\ g \cdot cm = 170 \times 9.8 \times 10^{-3} \times 10^{-2} \simeq 0.01667$$
$$i_a = 680\,mA = 0.68\ A$$

위 조건에서 3에서 주어진 식을 사용해서 토크 상수를 구하면 아래와 같다.

$$K_t = \frac{0.01667}{0.68} = 0.0245 \quad (N \cdot m/A)$$

- 역 기전력 상수 K_e

제원표의 No Load 조건을 이용해서 역 기전력 상수를 구할 수 있다. 위에서 결정한 전기자 저항이 3.3 Ω이라면, 역 기전력 상수 K_e는 아래와 같다.

$$K_e = \frac{24 - 3.3 \times 0.22}{5870 * 2\pi \times \frac{1}{60}} = 0.03786 \quad (V \cdot s/rad)$$

- 모터 회전자 관성 모멘트 J와 전기자 인덕턴스 L_a

DC 모터 관련 식에서 알 수 있듯이, 관성 모멘트 J는 회전각속도에 대해 Damping항으로 영향을 주고, 전기자 인덕턴스는 전류변화에 영향을 준다. 시간에 대한 응답특성에 영향을 주어 과도 응답에서의 지연으로 나타나고, 실제 정상상태에 대한 영향은 미소하다. 따라서 관성 모멘트 J는 Motor 지름에 대해 일정 비율의 지름을 가지는 코일 뭉치로 가정하여 구한 값을 그대로 사용하기로 한다. 그리고 전기자 인덕턴스는 LCR 미터를 이용하여 기본 값을 구하고, 이에 대해 응답 특성 등을 고려해서 구하도록 한다.
이 과정을 통해 구한 관성 모멘트 및 전기자 인덕턴스 값은 다음과 같다.

$$J = 0.00001 \quad (kg \cdot m^2)$$
$$L_a = 0.00334 \quad (H)$$

(6) 시뮬레이션

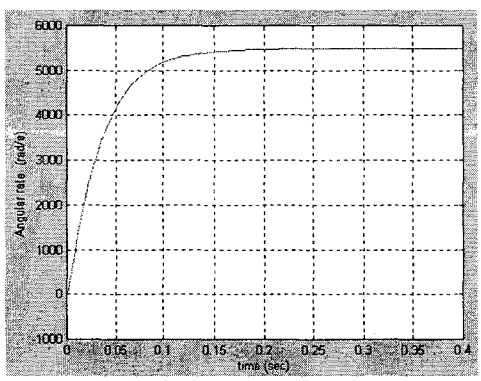


그림 6.1 회전 각속도 변화

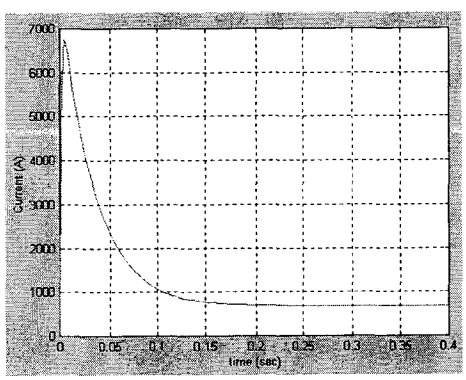


그림 6.2 전류 변화

위 결과에서 Rated Load 조건에서 정상상태일 경우, 회전 각속도는 5486rpm 이었고, 전류는 680 mA 였다. 따라서 전류는 제원표와 동일하게 나왔고, 회전 각속도의 경우, 약 7%의 오차가 난다. 이는 모터 제작 상에서 발생하는 오차로 분석된다. 따라서 이는 개별 모터에 대해 추후 그 계수를 구해보고, 평균을 구하여 적용하도록 해야 할 것이다.
그리고 이 편차에 대해서는 오차로써 판별해야 할 것이다.

- 적용 모터 제원 데이터

```
L_a=0.00334;      % 모터의 인덕턴스      (H)%
R_a=3.3;          % 모터의 저항          (Ohm)%
K_t=0.0245;         % 모터 토크 상수      (N-m/A)%
K_e=0.03786;      % 모터 역기전력 상수    (V-sec/rad)%
J=0.00001;        % 모터의 관성 모멘트    (kg-m^2)%
T_L = 0.01666;
```

다. 모터 제어

(1) 직류 모터의 전기적 모델

계자의 자기 회로의 포화를 무시할 경우 다음과 같은 관계를 가진다

$$v_a = R_a i_a + L_a \frac{di_a}{dt} + K \cdot i_f \cdot \omega_m$$

$$v_f = R_f i_f + L_f \frac{di_f}{dt}$$

$$\tau = K \cdot i_f \cdot i_a$$

여기서, v_a, v_f : 전기자 전압과 계자 전압

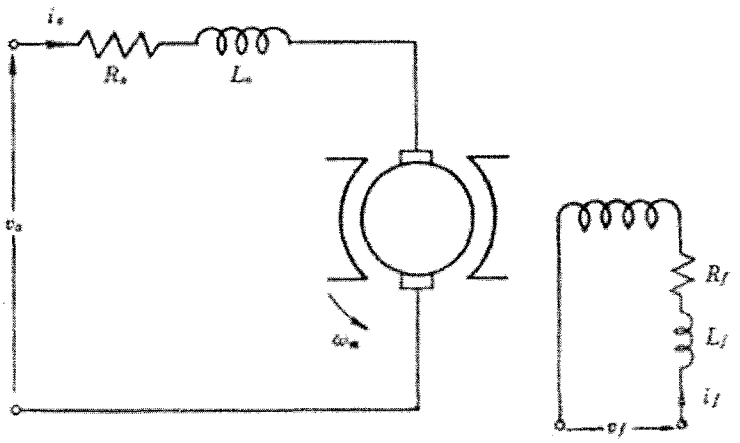
i_a, i_f : 전기자 전류와 계자 전류

R_a, L_a : 전기자 회로 저항과 인덕턴스

R_f, L_f : 계자 회로 저항과 인덕턴스

K, ω_m : 상수와 모터 속도

τ : 모터의 발생 토크



(2) 모터의 정상 상태.

정상 상태일 경우 전류의 시간에 대한 변화율은 0이다.

$$V_a = R_a I_a + K I_f \Omega_m$$

$$V_f = R_f I_f$$

$$T = K I_f \cdot I_a$$

- 모터 속도

$$\Omega_m = \frac{V_a - R_a I_a}{K I_f} = \Omega_{m0}(1 - \frac{R_a I_a}{V_a})$$

여기서, $\Omega_{m0} = \frac{V_a}{K I_f}$

- Ω_{m0} : 모터의 무부하시 속도. 전기자 전압과 계자 전류의 함수이다

(3) 전기자 제어법

응답성이나 정밀도가 요구되는 직류 모터의 제어에 주로 상용.

- 전달 함수 : 전기자 제어의 경우, 계자 전압과 전류는 일정.

$$v_a = R_a i_a + L_a \frac{di_a}{dt} + K_a \cdot \omega_m$$

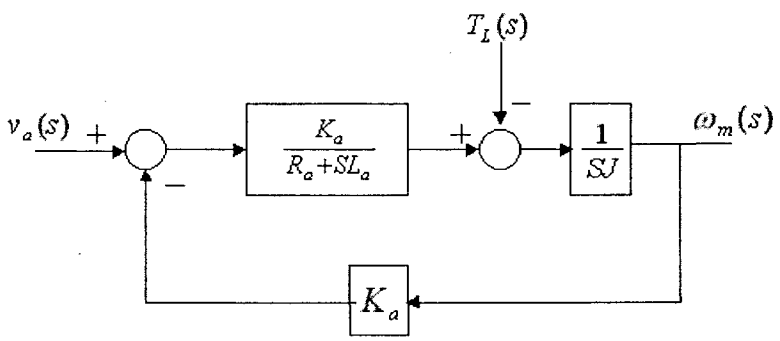
$$J \frac{d\omega_m}{dt} = K_a i_a - T_L$$

여기서, J : 모터와 부하의 합성 관성 모멘트

T_L : 부하의 반항 토크

$K_a = K \cdot I_f$

- 블록선도



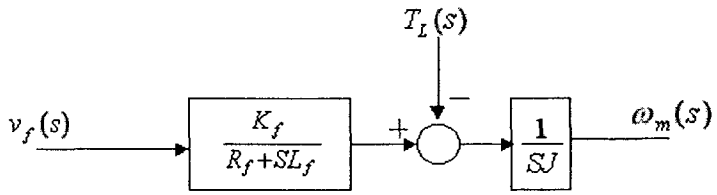
(4) 계자 제어법

전기자 회로가 정전류 급전 되어서 $i_a = I_a$가 일정하다고 가정함.

$$R_f \cdot i_f + L_f \frac{di_f}{dt} = v_f$$

$$J \frac{d\omega_m}{dt} = K_f \cdot i_f - T_L$$

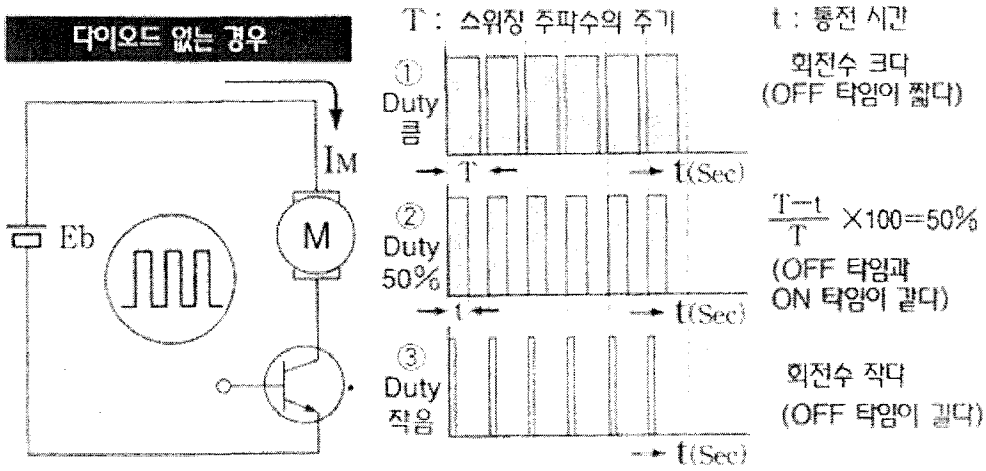
- 블록선도.



(5) PWM Control (펄스폭 변조 제어법)

On 펄스의 통전 폭을 임의로 변화시켜서 모터로의 입력 전력을 제어하는 것이다.

(가) 다이오드가 없는 경우.



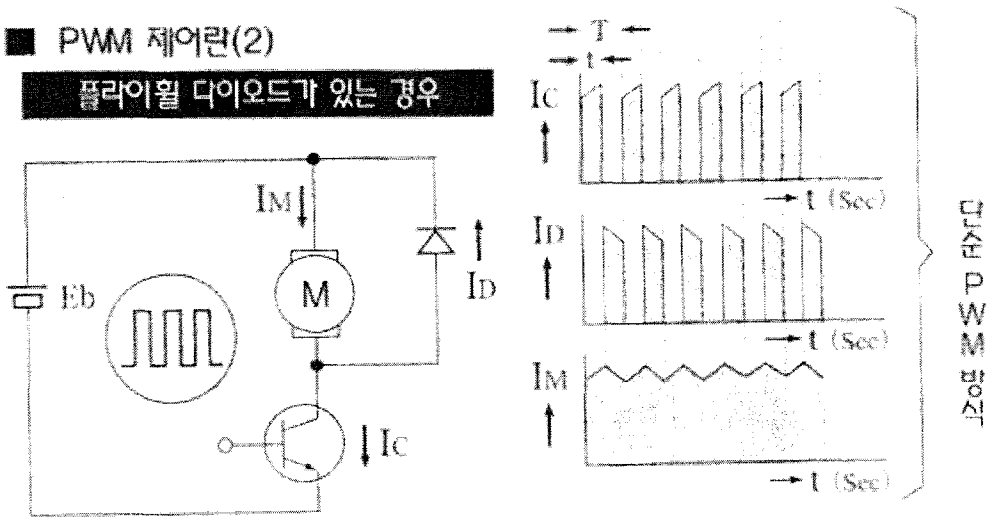
- 위 그림에서 1)은 듀티가 크며 그 에너지가 가장 크고 회전수도 이것에 비례하여 높게 되어 있다. 3)은 Duty가 작으며 회전수도 그에 대응하여 된다.
- PWM 제어법을 포함한 모터의 펄스 제어법은 전력 펄스가 ON일 때만 모터 전류가 흐르고 그 이외일 때는 흐르지 않기 때문에 트랜지스터의 부담이나 전원의 부하

가 매우 가볍게 된다.

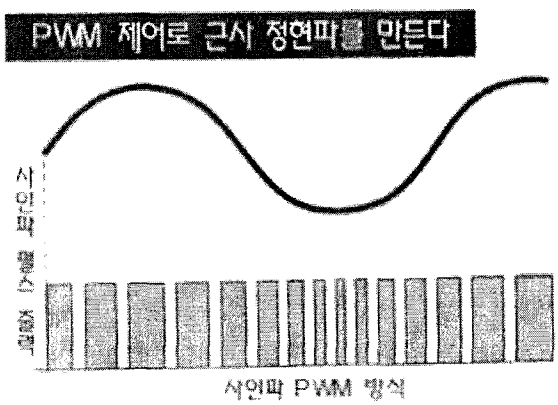
모터의 권선은 인덕턴스를 포함. 이것에 의해 자기 유도 작용이 발생. 큰 스파이크 모양의 전압이 발생. ⇒ 트랜지스터 파괴, 큰 전기 노이즈 발생.

대책은 모터의 권선에 플라이휠 다이오드를 삽입. 모터 Off 시에 유발하는 스파이크 모양의 전력을 모터로 회생시켜주는 방법 채용. 이것에 의해 높은 레벨의 전기 노이즈가 소멸될 뿐만 아니라 그 에너지를 모터에 흐르게 할 수 있으므로 모터에 흐르는 전류는 연속적으로 되어 모터의 움직임도 매우 원활하게 된다.

(나) 플라이휠 다이오드가 있는 경우.



(다) PWM 제어로 근사 정현파를 만든 경우.



라. 시스템 통합 모델링

- 관련식 Integration.

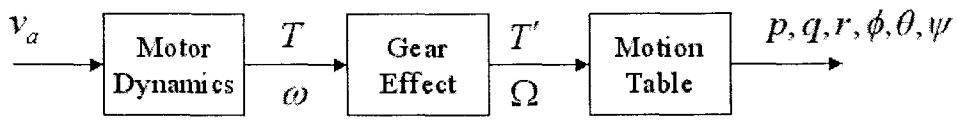
위 정리된 식에서 알 수 있듯이 Motor의 운동과 관련해서 입력 값인 전압, 출력 값인 각속도와 토크, 그리고 전류가 독립변수로써 작용한다. 그러나 일반적인 식과 달리 본 관계식에서는 외부 부하토크에 의한 영향으로 결정되는 모터의 회전 각속도가 전류에 영향을 주고, 이 전류가 다시 토크를 생성하므로, 선형적인 관계를 유지하지 않게 된다.

또한 Motor 관련 논문에서 다루고 있는 주제는 고정 부하에 대한 모터의 회전수나 각도를 제어하는 내용이다. 이는 본 과제에서 고려하는 모터 인가전압을 조정하여 Motion-Table의 자세를 조종하는 방법과는 다르다.

따라서 본 과제에서는 다음 식들의 관계를 고려해서 전체 시스템을 Integration 하여야 할 것이다.

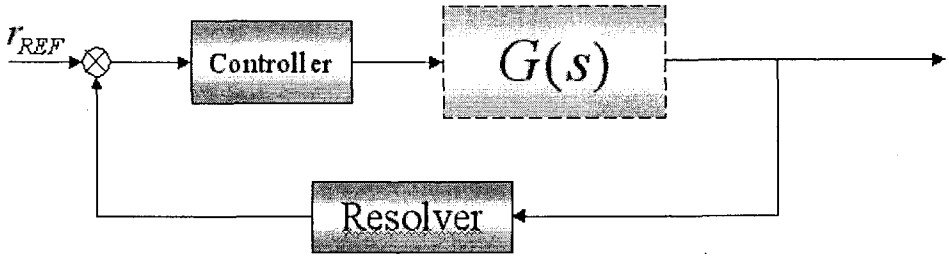
- Motor 관계식.
- Gear Effect 관계식.
- 운동판 관계식.

본 과제에서 시행되는 시스템은 다음과 같은 구성을 가진다.



- Linear system.

위에서 논의된 방법으로 시스템 관련 식을 구축할 경우, Nonlinear Equation이 유도될 것이다. 이를 제어기 설계에 사용하기 위해서는 임의의 조건에 대해 선형화 하는 작업이 필요하다. 이에 대해 추후 조사가 수행되어야 할 것이다.
이러한 과정을 거쳐 구한 제어기를 포함한 시스템은 다음과 같을 것이다.

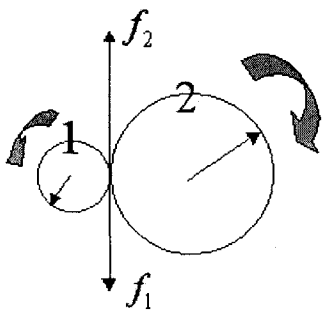


위 그림에서 G(s)는 위에서 구한 Motor, Gear 그리고 Motion-table을 Integration한 결과로 얻어진 비선형 관계식을 선형화함으로써 얻어진 전달함수이다.

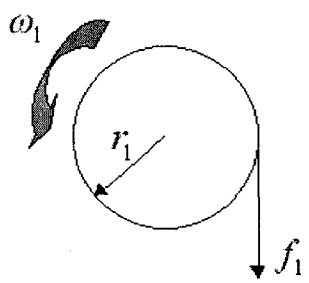
마. 기어

(1) 두 개의 기어가 맞물려 있는 경우.

마찰, Backlash, Slip이 없는 경우에 대해 다음과 같이 연결되어있는 기어에 대해 생각해 보자.



1번 기어에 토크가 적용된다고 가정하고 식을 유도해 보도록 한다. 우선 1번 기어에 모터가 연결되어 있는 경우, 1번 기어만을 독립적으로 생각해보자.



1번 기어에 연결된 모터에 토크 T_1이 작용할 경우, 위 그림에서 1번 기어의 각속도 ω_1은 다음 식을 만족한다.

$$J_1 \frac{d\omega_1}{dt} = T_1 - r_1 f_1$$

여기서, J_1은 모터의 회전자와 1번 기어를 합한 관성 모멘트이고, r_1은 1번 기어의 반경, f_1은 1번 기어에 작용하는 반작용 힘이다.

1번 기어의 회전에 의해 2번 기어가 회전할 경우, 이의 회전 각속도 ω_2는 다음 식을 만족한다.

$$J_2 \frac{d\omega_2}{dt} = T_2 + r_2 f_2$$

만약 2번 기어에 운동판이 연결되어 있다고 가정할 경우, 위 식에서 J_2는 2번 기어와 함께 운동판의 2번 기어 회전축 방향의 관성 모멘트의 합으로 이루어진다.

그리고 T_2는 무게중심이 2번 기어 회전축 상에서 벗어날 경우 존재하는 불균일 토크이고, r_2는 2번 기어의 반경, f_2는 1번 기어의 회전에 의해 2번 기어에 작용하는 힘이다.

본 과제에선 기어사이의 마찰과 Backlash를 무시하였으므로, 두 기어는 다음 식을 만족하면서 운동한다.

$$r_1 \omega_1 = r_2 \omega_2$$
$$f_1 = f_2$$

위 네 식을 이용하여 정리하면 다음과 같이 1번 기어에 대한 회전수의 관계식을 얻을 수 있다.

우선 반지름과 회전수의 관계식을 이용하면,

$$T_2 + r_2 f_2 = J_2 \frac{d}{dt}\left(\frac{r_1}{r_2}\omega_1\right)$$

이고,

위 식과 1,2번 기어의 기본 방정식을 아래와 같이 변환한다.

$$r_2 T_1 - r_1 r_2 f_1 = r_2 J_1 \frac{d\omega_1}{dt}$$
$$r_1 T_2 + r_1 r_2 f_1 = J_2 \frac{r_1^2}{r_2} \frac{d\omega_1}{dt}$$

위 두식을 더하면 다음과 같은 관계식을 얻을 수 있다.

$$r_2 T_1 + r_1 T_2 = \left(r_2 T_1 + J_2 \frac{r_1^2}{r_2}\right) \frac{d\omega_1}{dt}$$

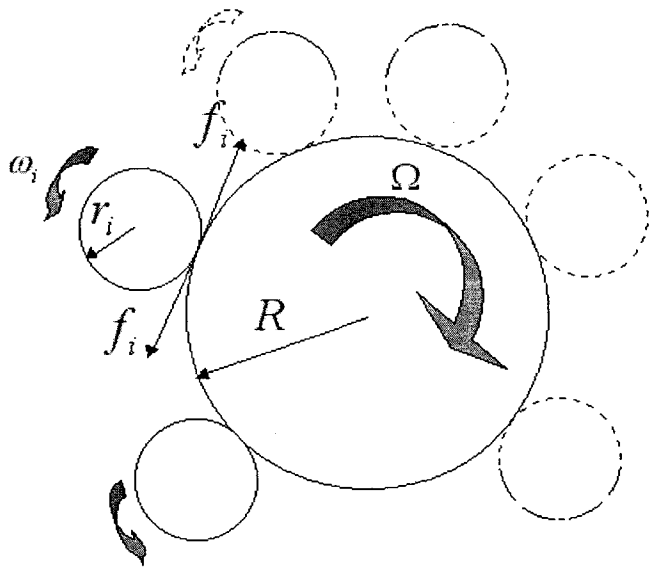
이 식을 정리하면 1번 기어에 작용하는 토크에 의한 1번 기어 각속도의 변화량 관계식을 구할 수 있다.

$$\frac{d\omega_1}{dt} = \frac{r_2 T_1 + r_1 T_2}{r_2 J_1 + J_2 \frac{r_1^2}{r_2}}$$

그리고 위 관계식에서 구한 1번 기어의 각속도와 반지름비 관계식을 이용하여 2번 기어의 회전각속도를 구할 수 있다.

(2) 하나의 Main Gear 주위에 소형 기어가 다수 연결되어 있는 경우.

소형기어의 반지름과 관성 모멘트는 모두 동일하다고 가정하고, Backlash 및 slip은 무시한다.



i-번째 모터에 의해 기어에 작용하는 토크를 T_i라 할 경우, 다음과 같은 관계식을 만족한다.

$$J_i \frac{d\omega_i}{dt} = T_i - r_i f_i$$

그리고 운동판에 연결된 Main Gear에 대한 관계식을 구하면 다음과 같다.

Main Gear의 경우 소형 모터측 기어와 연결되어 있으므로 각각에 의한 반작용 힘이 합해져 영향을 끼친다.

$$J_M \frac{d\Omega}{dt} = \sum_{i=1}^{N} R \cdot f_i + T_M$$

여기서, J_M은 운동판을 포함한 Main Gear 축상의 관성 모멘트이고, R은 Main Gear의 반지름 그리고 T_M은 Main Gear 축 방향에 존재하는 불균일 토크의 크기이다.

위 식에서 각 소형 모터에 의해 작용하는 토크가 일정하고, 각 소형 기어의 반지름이 동일 할 경우, 소형 기어와 Main Gear 사이에 존재하는 반작용 힘은 일정하다.

즉,

$T_1 = T_2 = \cdots = T_N$ 이고, $r_1 = r_2 = \cdots = r_N = r$ 이면,

$f_1 = f_2 = \cdots = f_N = f$

이다.

그리고 마찰 및 Backlash 등의 영향을 무시할 경우, 각 소형 기어의 회전 각속도 또한 동일하다.

따라서

$\omega_1 = \omega_2 = \cdots = \omega_N = \omega$

$r \cdot \omega = R \cdot \Omega$

이다.

위 관계를 이용해 정리하면 다음과 같은 세 개의 관계식을 구할 수 있다.

$$J \frac{d\omega}{dt} = T - r \cdot f$$

$$J_M \cdot \frac{d\Omega}{dt} = R \cdot f \cdot N + T_M$$

$$r \cdot \omega = R \cdot \Omega$$

여기서, J는 소형 모터측의 관성 모멘트이고, T는 소형모터에 의한 토크, f는 반작용 힘이다. 그리고 J_M은 Main Gear 측의 관성 모멘트로 Gear와 운동판의 합으로 구해진다. 그리고 Ω는 Main Gear의 회전각속도, N은 소형 모터의 개수이다.

T_M은 Main Gear 측의 불균형 토크이다.

위 식을 정리하면 다음과 같은 관계식을 구할 수 있다.

$$\left(J_M \frac{r}{R} + R \cdot N \cdot \frac{I}{r}\right) \frac{d\omega}{dt} = R \cdot N \cdot \frac{T}{r} + T_M$$

제 4 장 연구개발 목표달성도 및 대외기여도

제 1 절 계획대비 달성도

당초 계획 대비 전반적으로 상당한 성과를 얻었으나 세부 몇 분야에서는 약간 미진하게 연구가 수행 되었다. 그러나 이번 개발연구를 통하여 목표한 성능의 김발개발을 한층 앞당기는 성과를 거두었다고 볼 수 있다. 요소기술별로 달성 내용을 정리하면 다음과 같다.

세부연구개발 목표		달성도 (%)	달성내용
Gimbal 기구설계/제작		98	- 상용 모델 자료 조사 및 분석 - 김벌 기구설계 및 재료 선정 - 구조물 동특성해석 FEM모델 개발 - 기구물 시제제작 및 조립
센서/구동기 시스템 구성		98	- 구동기 선정 및 시험 - 센서 요구 조건 설정 및 특성시험 - 오차 모델링 및 보정기법 연구 - 자이로/가속도계 혼합 알고리즘 연구 - 가속도계를 이용한 변위측정
Gimbal 제어기	하드웨어 시험모델제작	100	- 제어기 프로세서 설계 및 제작 - 제어기 파워 분배기 설계 및 제작 - 김발 제어기 및 모터 드라이버 하우징 설계 및 제작
	탑재 소프트웨어 개발	100	- 제어기 탑재 소프트웨어 제작 - 제어기 FPGA 코딩 완료 - 제어용 PC 소프트웨어 제작
	성능시험 및 검증	80	- 김발 각 명령 추종 제어 성능 검증
성능해석 및 시험평가		100	- 100μrad급 안정정확도 측정기법 연구 - 마찰 시상수 및 관성 모멘트 측정법 연구 - 3축 운동시험대를 이용한 gyro 모델변수 측정기법 연구 - 열환경시험 및 진동시험 방법 정립
성능평가 기법 및 시뮬레이션 (위탁연구)		98	- Gimbal운동 모델링 및 알고리즘 연구 - 무인항공기 운동 모델링 및 6DOF 시뮬레이션 프로그래밍 - Gimbal 성능평가 절차 및 기법 조사연구

제 2 절 대외 기여도

이번 수행 연구결과는 총4년 기간 중 2년의 응용연구 단계 결과로 추후 시험개발 연구에 많은 기여를 하게 되겠다. 그리고 관련분야 기술발전에 많은 기여가 예상되나 현재 기술완성의 중간단계이므로 직접적결과 기여도는 크지 않다. 그러나 앞으로 완성되는 저가 경량의 시선안정화 김발 시스템 개발기술은 산업전반으로 영향을 미칠 것이다. 이러한 기여도를 예상해보면 다음과 같다.

1. 기술적 측면

항공정찰 및 사격통제 시스템에 필수적인 안정화 장치에 대한 연구를 수행함으로써 제어시스템, 구동기, 센서 및 소프트웨어 통합기술을 확보하여 자동화 로봇과 정밀계측센서 등 정밀산업 분야에 기여함.

2. 군사적 측면

비접근 지역의 공간 정보, 인공지물, 전장 현황 등의 탐지, 목표물 감시 및 요격 등을 위한 정밀하고 안정된 영상정보를 얻기 위해 필수적인 기술을 확보함으로써 선진 기술국의 기술이전 회피 정책에 대응하여 무기체계의 독자개발 기술에 기여함.

3. 경제적 측면

현재 전량 수입에 의존하고 있으며 고가인 영상장치 Gimbal 시스템의 국산화로 외화 유출을 최소화 할 수 있을 뿐 아니라, 광학계, 전자회로, 정밀기계, 계측/제어 및 신뢰도 기술 등의 부가 가치가 큰 기술의 수입 대체 기술로 기여함.

제 5 장 연구개발결과의 활용계획

제 1 절 추가연구의 필요성

저가의 소형 김발의 개발을 목표하여 4축의 고가 김발이 아닌 2축의 모터 구동기만을 적용하여 연구를 수행하였으나, 당초 계획대로 모터만을 적용하여 100마이크로라디안의 성능을 얻기는 약간 힘이 든다는 결과를 얻어서 추후 시험개발단계에서는 마찰이 거의 존재하지 않고 동작 반응이 보다 신속한 보이스 코일 액튜에이터(Voice coil actuator)를 피치축의 Inner김발에 적용하여 3축의 김발로 계획하고 있다. 본 응용연구단계의 연구에 추가하여 보이스 코일 액튜에이터의 동작 특성 및 전반에 관하여 추가 조사와 연구가 필요하겠다.

제 2 절 타 연구에의 응용

본 연구를 통해 산출된 김발 시스템의 요소기술 및 제품기술은 활용분야가 넓다. 직접적으로는 다음 연구단계인 시험개발단계의 기본적인 밑거름으로 목표하는 성능의 제품 개발에 큰 영향을 미칠 것이다. 요목별 결과의 활용 계획을 살펴보게 되면 다음과 같다.

첫째, 김발 기구설계 및 제작기술은 다른 종류의 정밀기계 개발사업과 더불어 2차 시제품 설계/제작 시 기구적으로 요구되는 경량, 소형, 신뢰도, 구조적 최적화, 강도, 재료 선정, 적용공차선정, 제작방법, 종합적인 통합설계 등에 큰 초석이 될 것이다. 특히 이번 1차 시제품의 설계 시 축적된 상용 CAD/CAM/CAE 프로그램의 활용기술 및 DMU 적용기술은 설계하고자 하는 제품에 대하여 3차원 공간상에서 최적의 형상설계 및 다양한 형상 적용으로 최적의 설계를 이룰 뿐만 아니라 설계 시 수시로 설계 검증 및 파라메트릭 디자인 기번을 적용하여 설계기간을 단축시킬 수 있게 되었다. 게다가 1차 시제품 및 실험용 기구의 제작 경험은 보다 완성도가 높은 2차 김발기구 제작에 상당한 영향을 미칠 것으로 예상된다.

둘째, 센서/구동기 시스템 구성은 자동제어제품 개발 시 중요한 요소기술이 된다. 이번 개발연구에서 쌓인 센서/구동기 선정 및 시스템 구성 방법은 이후 다른 개발 프로젝트도 아주 주요한 기술이 될 것이다. 센서/구동기의 선정은 즉 개발하고자 하는 제품의 가격 경쟁력에 많은 영향을 미친다. 실제로 고가의 부품을 적용하게 되면 보나 나은 성능의 제품을 얻는 것은 당연한 일이다. 그러나 현 세계는 글로벌화 경쟁 사회로 제품개발에 있어서 무작정 고성능 고가의 제품을 생산하는 것에는 무리가 따른다. 이번 김발 개발의 주요한 목표도 저가 경량의 소형 김발의 개발이다. 그러므로 이왕이면 저가의 부품으로 목표성능을 낼 수 있는 시스템 구성 기술은 아주 요긴하게 쓰일 것이다. 이번 센서/구동기 시스템 구성 연구에 동반하

여 산출된 결과 중 하나인 "저가 관성센서 기반의 시선안정화 제어 시스템 설계" (제어.자동화.시스템공학 논문지 제 9권 제3호 2003.3)연구는 성능만족의 저가 부품의 활용성을 입증하여 추후 개발 과제 등에서도 아주 요긴한 경험적 배경지식으로 사용 될 것이다.

셋째, 제어기 설계 및 제작은 이번 연구과제의 핵심 연구로서 앞으로 타 분야 및 다음 연구 기간동안 가장 많은 영향을 끼칠 것으로 예상된다. 이번 제어기 설계에서 확보된 김발 제어를 위한 DSP 프로세서 기반의 제어기 보드 및 파워 공급부 H/W 및 S/W의 기반기술은 앞으로도 타 제어분야에 많이 활용 될 것이다. 더블어 센서와 구동기 제어기술도 범용적으로 활용될 것으로 기대된다.

넷째, 시험 및 평가 방법은 현재 항우연이 보유한 각종 시험 장비들을 이용하여 본 연구와 같은 특정 제품개발 시 성능 검증을 할 수 있는 기틀을 마련하였고, 앞으로 이와 유사한 프로젝트에 시험 규정 및 기법 연구, 측정 시스템 준비와 센서에 대한 데이터 측정 등에 유용하가 사용될 것이다. 특히 국제 기준에 맞는 성능 검증 방법의 모색으로 추후 정밀 산업제품의 개발 시 보다 정확한 평가를 제시 할 것으로 기대된다.

제 3 절 기업화 추진방안

앞으로 이어지는 시험개발단계에서 2차 시제품을 이용하여 기술시험과 군 적용시험을 통하여 개발 성능을 검증한 후, 군수 및 민수 분야(항공 촬영, 산불관측, 교통방송, 영상 엔터테인먼트, 스포츠 산업) 등을 위주로 생산 판매할 계획이다. 이를 위해 시장 변화와 다양한 요구에 맞추어 고부가가치의 다품종 소량생산을 계획하고 있다. 한편, 개발된 김발 시스템의 주요 사업 분 대상이 될 군수분야의 실용화계획은 아래와 같다.

군적용분야	기존 무기체계 적용(국산화)계획 시제품내역		신규 무기체계적용계획 시제품 내역	
	시제품명	적용장비명	시제품명	적용장비명
항공 (소형무인기용 영상 송수신 및 지상조종장치)	• 영상안정화 김벌시스템 • 지상조종장치	• 정찰용 무인 항공기 • 탱크 포수 조준장치	• 영상 안정화 김벌시스템 • 지상조종장치	• 고공 정찰용 무인항공기 • 무인 정찰용 지상장비

제 6 장 참고문헌

1. Paul G. Fahlstrom and Thomas J. Gleason, Introduction to UAV Systems, 2nd Ed., UAV Systems Inc., June 1998.
2. George M. Siouris, Aerospace Avionics Systems: A Modern Synthesis, Academic Press, Inc., 1993.
3. William L. Brogan, Modern Control Theory, 3rd Ed., Prentice Hall, 1991.
4. T.H.Lee, K. K. Tan, A.Manmum, M.W.Lee, and C.J. Khoh, "Composite control of a gyro mirror line-of-sight stabilization platform design and auto tuning", Intelligent Control and Automation, vol. 5, pp.3150-3155, 2000.
5. B. Li, D. A. Hullender, and M. DiRenzo, "Nonlinear Induced Disturbance Rejection in Inertial Stabilization Systems", IEEE Trans. of Automatic Control, vol. 6, no. 3, pp. 421-427 May. 1998.
6. B. Li and D. A. Hullender, "A self-tuning controller for nonlinear inertial stabilization systems", IEEE Control Systems. Technology, vol. 6, no 3, 1998.
7. H. Ambrose, Z. Qu, and R. Johnson, "Nonlinear Roust Control For A Passive Line-of-Sight Stabilization System", IEEE Inter Conference pp. 942-947, 2001.
8 W. Vukovich, G. Zywiel, J. scherzinger, B. M. Russell, and H. Burke, "The Honeywell/DND helicopter integrated navigation system(HINS)", IEEE Aerospace and Electronics system Magazine, vol. 4, pp. 18-28, 1989.
9 M. C. Algrain and J. Quinn, "Accelerometer based line-of-sight stabilization approach for pointing and tracking systems", Second IEEE Conference, vol. 1, pp. 159-163, 1993.
10 박찬국, 김광진, 박흥원, 이장규, "스트랩다운 관성항법 시스템의 초기 개략정렬 알고리즘 개발", 제어· 자동· 시스템공학 논문지, 제 4국, 제 5호, pp. 647-679, October. 1998.
11 D. H. Titterton and J. L. Weston, Starpdown inertial navigation technology, Peter Pergrinus Ltd, 1997.
12 김종혁, 문승욱, 이시호, 김세환, 황동환, 이상정, 나성웅, "스트랩다운 관성항법시스템의 초기정렬 알고리즘구현," 제어· 자동화·시스템공학 논문지, 제6권, 제 2호, pp. 138-145, February. 2000.
13 F. Gene, J. Franklin, D. Powell, and A. Emami-Naeini, Feedback Control of Dynamic Systems, Addison-Wesley Publishing Company, 1995.
14 J. Winkelhake, U, Konigorski and L. Wiebking, "Modelbased multirate cascade control of direct drives in automotive systems", World Automation Congress, 2001. Proceedings of the 5th Biannual, vol. 14, pp. 121-126 2002.
15 U. Halldorsson, M. Fikar, and, H. Unbehauen, "Multirate nonlinear predictive control", Proc. American Control Conference, 2002, vol. 6, pp. 4922-4927, 2002.
16. Inertial Navigation Systems, Broxmeyer, McGRAW-HILL Electronic Sciences Series, 1964
17. 조기대, 전원석, 김정주, 박좌근,"헬기용 FLIR 장착을 위한 진동저감 시스템," 한국항공우주학회 추계학술발표회, 제1권,PP. 230-234, December. 2002.

첨부1 : 운동판 제어 프로그램 소스 코드

프로그램언어 : MFC (Visual C++ 6.0)
프로그램 작성자 : 박무혁
프로그램 구성파일 :

CPP 파일
- FDRDemo.cpp
- FDRDemoDlg.cpp
- PortSet1.cpp
- FDRDemo.rc
- stdafx.cpp

Header 파일
- FDRDemo.h
- FDRDemoDlg.h
- PortSet1.h
- ComLsh.h
- stdafx.h

주요 CPP 파일 그로그램 소스

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////
FDRDemoDlg.cpp
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////
#include "stdafx.h"
#include "FDRDemo.h"
#include "FDRDemoDlg.h"
#include "ComLsh.h"
#include "PortSet1.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define IP_ADDR_SAT             "150.197.46.20"
#define PORT_ADDR_SAT       7100

#define WM_ADD_LIST  WM_USER +1
#define WM_ADD_LIST2 WM_USER +2
#define WM_ON_BUTTON2 WM_USER +3
//////////////////////////////////////////////////////////////
#define SCALE_POS       (4294967296.0/360.0)
#define SCALE_RAT       (4294967296.0/22500.0)
#define SCALE_ACCEL     (4294967296.0/1406250.0)
#define SCALE_FREQ  (4294967296.0/4000.0)
#define SCALE_AMPL      (4294967296.0/360.0)
#define SCALE_PHASE     (4294967296.0/360.0)

char position_cmd[] = {"d:P 1,#14\0\0\0\0:P 2,#14\0\0\0\0:P 3,#14\0\0\0\0\n"};
char rate_cmd[]     = {"d:R 1,#14\0\0\0\0:R 2,#14\0\0\0\0:R 3,#14\0\0\0\0\n"};
char accel_cmd[]    = {"d:A 1,#14\0\0\0\0:A 2,#14\0\0\0\0:A 3,#14\0\0\0\0\n"};
char  osc_cmd[]             =  {"d:o  1,#212\0\0\0\0\0\0\0\0\0\0:o  2,#212\0\0\0\0\0\0\0\0\0\0:o
3,#212\0\0\0\0\0\0\0\0\0\0\0\0\n"};

char ratel_cmd[]    = {"d:r 2,#14\0\0\0\0\n"};
char svectorl_cmd[]    = {"d:s 1,#18\0\0\0\0\0\0\0\0\n"};
double x_pos, y_pos, z_pos;
char mode_rate[] = {"m:r 1:r 2:r 3\n"};
char mode_position[] = {"m:p 1:p 2:p 3\n"};

void write_position(double x,double y,double z)
{

        *(long *) &position_cmd[9]  = (long)( x * SCALE_POS);
        *(long *) &position_cmd[21] = (long)( y * SCALE_POS);
        *(long *) &position_cmd[33] = (long)( z * SCALE_POS);

}

void write_svector(double x,double y,double z,double p,double q,double r)
{


        *(long *) &svectorl_cmd[9]  = (long)( x * SCALE_POS);
//      *(long *) &svectorl_cmd[21] = (long)( y * SCALE_POS);
//      *(long *) &svectorl_cmd[33] = (long)( z * SCALE_POS);
        *(long *) &svectorl_cmd[13]  = (long)( p * SCALE_RAT);
```

```
}

void write_rate(double x,double y,double z)
{
        long scaled_rate[3];

        *(long *) &rate_cmd[9]  = (long)( x * SCALE_RAT);
        *(long *) &rate_cmd[21] = (long)( y * SCALE_RAT);
        *(long *) &rate_cmd[33] = (long)( z * SCALE_RAT);


}

void write_osc(double mag1,double freq1,double phase1,
        double mag2,double freq2,double phase2,
        double mag3,double freq3,double phase3)
{


        *(long *) &osc_cmd[10] = (long)(mag1   * SCALE_AMPL);
        *(long *) &osc_cmd[14] = (long)(freq1  * SCALE_FREQ);
        *(long *) &osc_cmd[18] = (long)(phase1 * SCALE_PHASE);

        *(long *) &osc_cmd[31] = (long)(mag2   * SCALE_AMPL);
        *(long *) &osc_cmd[35] = (long)(freq2  * SCALE_FREQ);
        *(long *) &osc_cmd[39] = (long)(phase2 * SCALE_PHASE);

        *(long *) &osc_cmd[52] = (long)(mag3   * SCALE_AMPL);
        *(long *) &osc_cmd[56] = (long)(freq3  * SCALE_FREQ);
        *(long *) &osc_cmd[60] = (long)(phase3 * SCALE_PHASE);
}
void Delay(int sec)
{
        int InitialTic,FinalTic,CompDuration,cnt=0;
        InitialTic = GetTickCount();
        FinalTic = GetTickCount();
        CompDuration= FinalTic-InitialTic;
        while(CompDuration/1000 < sec)
        {
                FinalTic = GetTickCount();
                CompDuration= FinalTic-InitialTic;

        }
}

////////////////////////////////////////////////////////////////////////////

HWND MainWndHandle;
BOOL THREADRUN = TRUE;
BOOL MTOPERATION = FALSE;
BOOL LAST = FALSE;
BOOL GPSTHREADRUN = TRUE;
int IntervalTime;
CSerialComm MySerialCom,m_pComm;
CSerialComm MyGPSSerialCom;
CUdpCliCom TxPortToSATmodem ;
float Xp,Yp,Zp,Xr,Yr,Zr;
float           RollAngle,
                        PitchAngle ,
                        YawAngle ,
                        RollRate ,
                        PitchRate,
                        YawRate,Ax,Ay,Az,AccTemp,SysTemp;
unsigned int AcqTime ;

CListBox *List1,*List2, *List3;
CString ListS;
CString AHRS,COMMAND,GPSDATA;
static struct SaterllitePacket
{
        unsigned char header;
        float RollAngle;
        float PitchAngle;
        float YawAngle;
        float RollRate;
        float PitchRate;
        float YawRate;
        float   Latitude;
        unsigned char N;
        float   Longitude;
        unsigned char E;
        float   Altitude;
```

```
        unsigned char CheckSum;
} SPacketForm;
SaterllitePacket SPacket;


float     X_Rate,    Y_Rate,          Z_Rate          ,X_Vel          ,Y_Vel,Z_Vel,X_Acc,Y_Acc,Z_Acc,X_Angle,
Y_Angle,Z_Angle,GPS_X_Vel,GPS_Y_Vel,GPS_Z_Vel;

int                GPS_X_Pos,GPS_Y_Pos,GPS_Z_Pos,X_Pos,Y_Pos,Z_Pos,GPS_TOW;


UINT ThreadRoutine(LPVOID pParam)
{
        unsigned char MsgData[500];
        DWORD MsgNo;
        int i,cnt=0;int InitialTic,FinalTic,CompDuration;
        unsigned int Sum = 0;
        FILE* fpMTCommand, * fpDataSave;
        char Command_filename[30];
        BOOL FIRST=TRUE;
        CFDRDemoDlg Mydlg;
        unsigned char CommandMsg[10];
        unsigned char MainMsg[]="test\r";

        InitialTic = GetTickCount();
        fpMTCommand = fopen("MTCommand.txt","r");
        fpDataSave = fopen("DUMMY_WithoutMotion.txt","w");

        while(THREADRUN)
        {

                FinalTic = GetTickCount();
                CompDuration= FinalTic-InitialTic;

            if(MTOPERATION)
                {
                        if(CompDuration > IntervalTime*1000 || FIRST)
                        {
                                FIRST =FALSE;
                                if(!feof(fpMTCommand))
                                {
                                fclose(fpDataSave);
                                fscanf(fpMTCommand,"%s       %f       %f       %f       %f       %f
%f\n",Command_filename,&Xp,&Yp,&Zp,&Xr,&Yr,&Zr);
                                ::SendMessage(MainWndHandle,WM_ADD_LIST2,0,0);
                                fpDataSave = fopen((LPCTSTR)Command_filename,"w");

                                m_pComm.Write((unsigned          char          *)mode_position,
sizeof(mode_position)-1);
                                write_position(Xp,Yp,Zp);
                                m_pComm.Write((unsigned          char          *)position_cmd,
sizeof(position_cmd)-1);
                                Delay(5);
                                m_pComm.Write((unsigned char *)mode_rate, sizeof(mode_rate)-1);
                                write_rate(Xr,Yr,Zr);
                                m_pComm.Write((unsigned char *)rate_cmd, sizeof(rate_cmd)-1);

                                InitialTic = GetTickCount();
                                }
                                else
                                {
                                        fclose(fpMTCommand);
                                        MTOPERATION=FALSE;
                                        LAST = TRUE;
                                        ::SendMessage(MainWndHandle,WM_ON_BUTTON2,0,0);

                                }
                        }
                        else;
                }
                else
                {
                //              write_rate(0,0,0);
                //              m_pComm.Write((unsigned char *)rate_cmd, sizeof(rate_cmd)-1);


                }


                        AcqTime = GetTickCount();
                ::SendMessage(MainWndHandle,WM_ADD_LIST,0,0);
```

```
                                        AHRS.Format("%d    %12.4f    %12.4f    %12.4f    %12.4f    %12.4f
%12.4f",AcqTime,X_Rate,Y_Rate,Z_Rate,X_Vel,Y_Vel,Z_Vel);
                                        fprintf(fpDataSave,"%d %12.4f %12.4f %12.4f %12.4f %12.4f %12.4f
%12.4f %12.4f \n",AcqTime,  X_Rate,Y_Rate,Z_Rate,X_Vel,Y_Vel,Z_Vel,X_Acc
                                                                                        ,Y_Acc);


                }

        if(CompDuration > IntervalTime*1000 && LAST)
                fclose(fpDataSave);


        return 0;
}


class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// Dialog Data
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
                // No message handlers
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFDRDemoDlg dialog

CFDRDemoDlg::CFDRDemoDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CFDRDemoDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CFDRDemoDlg)
        m_IntervalTime = 5;
        m_Xp = 0.0f;
        m_Yp = 0.0f;
        m_Zp = 0.0f;
        m_Xr = 0.0f;
        m_Yr = 0.0f;
        m_Zr = 0.0f;
        m_Mode = _T("LOCAL");
        m_editvalue = _T("");
        //}}AFX_DATA_INIT
        // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
        m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CFDRDemoDlg::DoDataExchange(CDataExchange* pDX)
{
```

```
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CFDRDemoDlg)
	DDX_Text(pDX, IDC_EDIT1, m_IntervalTime);
	DDX_Text(pDX, IDC_EDIT2, m_Xp);
	DDX_Text(pDX, IDC_EDIT3, m_Yp);
	DDX_Text(pDX, IDC_EDIT4, m_Zp);
	DDX_Text(pDX, IDC_EDIT5, m_Xr);
	DDX_Text(pDX, IDC_EDIT6, m_Yr);
	DDX_Text(pDX, IDC_EDIT7, m_Zr);
	DDX_LBString(pDX, IDC_LIST2, m_Mode);
	DDX_Text(pDX, IDC_EDIT8, m_editvalue);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CFDRDemoDlg, CDialog)
	//{{AFX_MSG_MAP(CFDRDemoDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
	ON_EN_CHANGE(IDC_EDIT1, OnChangeEdit1)
	ON_EN_CHANGE(IDC_EDIT2, OnChangeEdit2)
	ON_EN_CHANGE(IDC_EDIT3, OnChangeEdit3)
	ON_EN_CHANGE(IDC_EDIT4, OnChangeEdit4)
	ON_EN_CHANGE(IDC_EDIT5, OnChangeEdit5)
	ON_EN_CHANGE(IDC_EDIT6, OnChangeEdit6)
	ON_EN_CHANGE(IDC_EDIT7, OnChangeEdit7)
	ON_BN_CLICKED(IDC_BUTTON2, OnButton2)
	ON_EN_CHANGE(IDC_EDIT8, OnChangeEdit8)
	//}}AFX_MSG_MAP
	ON_MESSAGE(WM_ADD_LIST,OnDisplayList)
	ON_MESSAGE(WM_ADD_LIST2,OnCommandList)
	ON_MESSAGE(WM_ON_BUTTON2,OnButton2)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFDRDemoDlg message handlers

BOOL CFDRDemoDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu)
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here
	MainWndHandle = m_hWnd;
	IntervalTime=m_IntervalTime;
//TxPortToSATmodem.Open (IP_ADDR_SAT,PORT_ADDR_SAT,FALSE, 160) ;
	Setcomm();
//	GPSSetcomm();


	List1 = (CListBox*)GetDlgItem(IDC_LIST1);
	List2 = (CListBox*)GetDlgItem(IDC_LIST2);
//	List3 = (CListBox*)GetDlgItem(IDC_LIST3);
//	SetTimer(0,1000,NULL);
	m_Mode = "LOCAL";
	List2->DeleteString(0);
	List2->InsertString(0,(LPCTSTR)m_Mode);
	return TRUE;  // return TRUE  unless you set the focus to a control
}
```

```
void CFDRDemoDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CFDRDemoDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CFDRDemoDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}


//pmh code start
void CFDRDemoDlg::Setcomm()
{
	// TODO: Add your control notification handler code here

	CPortSet MyPort;



/*	if(MyPort.DoModal() == IDOK)
	{
		if(THREADRUN)
		{
			THREADRUN = FALSE;
			Sleep(1000);
		}
		MyPort.sCommPort = "COM1";
		MyPort.dwBaudRate = 38400;
		MyPort.byByteSize = 8;
		MyPort.byParity = 0;
		MyPort.byStopBits = 1;*/
		MySerialCom.Open("COM1",38400,8,NOPARITY,ONESTOPBIT);

		THREADRUN = TRUE;
		AfxBeginThread(ThreadRoutine,m_hWnd,THREAD_PRIORITY_NORMAL,0,0,NULL)
//	}


}

void CFDRDemoDlg::GPSSetcomm()
```

```
{
        // TODO: Add your control notification handler code here

        CPortSet MyPort;

/*      MyPort.sCommPort = "COM2";
        MyPort.dwBaudRate = 4800;
        MyPort.byByteSize = 8;
        MyPort.byParity = 0;
        MyPort.byStopBits = 1;*/
        MyGPSSerialCom.Open("COM2",4800,8,NOPARITY,ONESTOPBIT);

        GPSTHREADRUN = TRUE;
//      AfxBeginThread(GPSThreadRoutine,m_hWnd,THREAD_PRIORITY_NORMAL,0,0,NULL)

//      SetTimer(0,100,NULL);
}

BOOL CFDRDemoDlg::CheckSum(unsigned char *Data, int no)
{
        int i;
        int rtn;

//      rtn = Data[2];
        for(i = 1; i <= 28; i++)
        {
                rtn += Data[i];
        }
        if (rtn % 256 == Data[29])
          return TRUE;
        else
                return FALSE;
}

long CFDRDemoDlg::OnDisplayList(WPARAM wParam, LPARAM lParam)
{
        List1->InsertString(0,(LPCTSTR)AHRS);
        UpdateData(FALSE);
        List1->DeleteString(17);
        return 0;
}
long CFDRDemoDlg::OnCommandList(WPARAM wParam, LPARAM lParam)
{
//      List2->InsertString(0,(LPCTSTR)COMMAND);
//      UpdateData(FALSE);
//      List1->DeleteString(10);
        m_Xp = Xp;
        UpdateData(FALSE);
        m_Yp = Yp;
        UpdateData(FALSE);
        m_Zp = Zp;
        UpdateData(FALSE);
        m_Xr = Xr;
        UpdateData(FALSE);
        m_Yr = Yr;
        UpdateData(FALSE);
        m_Zr = Zr;
        UpdateData(FALSE);
        return 0;
}

long CFDRDemoDlg::OnGPSList(WPARAM wParam, LPARAM lParam)
{
//      List3->InsertString(0,(LPCTSTR)GPSDATA);
//      UpdateData(FALSE);
//      List1->DeleteString(10);
        return 0;
}
void CFDRDemoDlg::OnOK()
{
        // TODO: Add extra validation here
        MyGPSSerialCom.Close();
        MySerialCom.Close();
        m_pComm.Close();
        CDialog::OnOK();
}


void CFDRDemoDlg::OnButton1()
{
        // TODO: Add your control notification handler code here
```

```
        m_pComm.Open("COM2",9600,8,2,2);
        unsigned char remote[] = "R\r";
        m_pComm.Write(remote, 2);
        Delay(1);
        m_Mode = "REMOTE";
        List2->DeleteString(0);
        List2->InsertString(0,(LPCTSTR)m_Mode);
        UpdateData(FALSE);
        write_position(0.0,0.0,0.0);

        MTOPERATION = TRUE;

}
void CFDRDemoDlg::OnButton2()
{
        // TODO: Add your control notification handler code here
        MTOPERATION = FALSE;
        unsigned char local[] = "L\r";

        m_pComm.Write(local, 2);
        m_Mode = "LOCAL";
        List2->DeleteString(0);
        List2->InsertString(0,(LPCTSTR)m_Mode);


        UpdateData(FALSE);
}
void CFDRDemoDlg::OnChangeEdit1()
{
        // TODO: If this is a RICHEDIT control, the control will not
        // send this notification unless you override the CDialog::OnInitDialog()
        // function and call CRichEditCtrl().SetEventMask()
        // with the ENM_CHANGE flag ORed into the mask.

        // TODO: Add your control notification handler code here
        IntervalTime=       m_IntervalTime;
        UpdateData(TRUE);
}

void CFDRDemoDlg::OnChangeEdit2()
{
        // TODO: If this is a RICHEDIT control, the control will not
        // send this notification unless you override the CDialog::OnInitDialog()
        // function and call CRichEditCtrl().SetEventMask()
        // with the ENM_CHANGE flag ORed into the mask.

        // TODO: Add your control notification handler code here
                UpdateData(FALSE);
}

void CFDRDemoDlg::OnChangeEdit3()
{
        // TODO: If this is a RICHEDIT control, the control will not
        // send this notification unless you override the CDialog::OnInitDialog()
        // function and call CRichEditCtrl().SetEventMask()
        // with the ENM_CHANGE flag ORed into the mask.

        // TODO: Add your control notification handler code here
                UpdateData(FALSE);
}

void CFDRDemoDlg::OnChangeEdit4()
{
        // TODO: If this is a RICHEDIT control, the control will not
        // send this notification unless you override the CDialog::OnInitDialog()
        // function and call CRichEditCtrl().SetEventMask()
        // with the ENM_CHANGE flag ORed into the mask.

        // TODO: Add your control notification handler code here
        UpdateData(FALSE);
}

void CFDRDemoDlg::OnChangeEdit5()
{
        // TODO: If this is a RICHEDIT control, the control will not
        // send this notification unless you override the CDialog::OnInitDialog()
        // function and call CRichEditCtrl().SetEventMask()
        // with the ENM_CHANGE flag ORed into the mask.

        // TODO: Add your control notification handler code here
```

```
        UpdateData(FALSE);
}

void CFDRDemoDlg::OnChangeEdit6()
{
        // TODO: If this is a RICHEDIT control, the control will not
        // send this notification unless you override the CDialog::OnInitDialog()
        // function and call CRichEditCtrl().SetEventMask()
        // with the ENM_CHANGE flag ORed into the mask.

        // TODO: Add your control notification handler code here
        UpdateData(FALSE);
}

void CFDRDemoDlg::OnChangeEdit7()
{
        // TODO: If this is a RICHEDIT control, the control will not
        // send this notification unless you override the CDialog::OnInitDialog()
        // function and call CRichEditCtrl().SetEventMask()
        // with the ENM_CHANGE flag ORed into the mask.

        // TODO: Add your control notification handler code here
        UpdateData(FALSE);
}



void CFDRDemoDlg::OnChangeEdit8()
{
        // TODO: If this is a RICHEDIT control, the control will not
        // send this notification unless you override the CDialog::OnInitDialog()
        // function and call CRichEditCtrl().SetEventMask()
        // with the ENM_CHANGE flag ORed into the mask.

        // TODO: Add your control notification handler code here
        int cnt=0;

//      if (m_Connected == FALSE) {
//              MessageBox("Can't open Comport. Comport Setup First Please!");
//              return;
//      }
        UpdateData(TRUE);

        if (*((LPCTSTR)(m_editvalue.Right(1))) == 0xa) {

                cnt = m_editvalue.GetLength();
                MySerialCom.Write((unsigned char *)((LPCTSTR)m_editvalue), cnt-1)
                m_editvalue.Empty();
        }
        UpdateData(FALSE);
}
```

첨부2 : 데이터 획득 프로그램 소스코드

프로그램언어 : MFC (Visual C++ 6.0)
프로그램 작성자 : 박무혁
프로그램 구성파일 :

CPP 파일
- rcmprevDlg.cpp
- Comm.cpp
- CommSet.cpp
- rcmprev.cpp
- rcmprev.rc
- stdafx.cpp

Header 파일
- rcmprevDlg.h
- Comm.h
- CommSet.h
- rcmprev.h
- stdafx.h

주요 CPP 파일 그로그램 소스

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////
```

rcmprevDlg.cpp

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////
#include "stdafx.h"
#include "rcmprev.h"
#include "rcmprevDlg.h"
#include "Comm.h"
#include "CommSet.h"
#include "MulTimer.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define WM_TEST_TIMER WM_USER+10
char LF = 0x0a;
////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

CComm               m_pComm;
CCommSet  dlg;
FILE                *fpt;
BOOL                m_Connected=FALSE, hexa=TRUE;
CString             m_rcvdatar;
int                         tcnt;

char tempblock[100];

CMulTimer MyTimer;

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// Dialog Data
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

```
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CRcmprevDlg dialog

CRcmprevDlg::CRcmprevDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CRcmprevDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CRcmprevDlg)
	m_editvalue = _T("");
	m_rcvdata = _T("");
	m_timeinterval = 100;
	m_duration = 100;
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CRcmprevDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CRcmprevDlg)
	DDX_Control(pDX, IDC_LIST1, m_comctrl);
	DDX_Control(pDX, IDC_LISTData2, m_gndctrl);
	DDX_Text(pDX, IDC_EDIT1, m_editvalue);
	DDX_Text(pDX, IDC_EDIT2, m_rcvdata);
	DDX_Text(pDX, IDC_EDIT3, m_timeinterval);
	DDV_MinMaxInt(pDX, m_timeinterval, 0, 10000);
	DDX_Text(pDX, IDC_EDIT4, m_duration);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CRcmprevDlg, CDialog)
	//{{AFX_MSG_MAP(CRcmprevDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDCommSet, OnCommSet)
	ON_BN_CLICKED(IDTelStart, OnTelStart)
	ON_BN_CLICKED(IDTelStop, OnTelStop)
	ON_BN_CLICKED(IDC_LOG, OnLog)
	ON_BN_CLICKED(IDExit, OnExit)
	ON_BN_CLICKED(IDC_CHECK1, OnCheck1)
	ON_BN_CLICKED(IDRCStart, OnRCStart)
	ON_BN_CLICKED(IDRCStop, OnRCStop)
	ON_EN_CHANGE(IDC_EDIT1, OnChangeEdit1)
	ON_BN_CLICKED(IDC_RADIO1, OnRadio1)
	ON_BN_CLICKED(IDC_RADIO2, OnRadio2)
	ON_WM_TIMER()
	ON_EN_CHANGE(IDC_EDIT3, OnChangeInterval)
	ON_EN_CHANGE(IDC_EDIT4, OnChangeDuration)
	//}}AFX_MSG_MAP
	ON_MESSAGE(WM_RECEIVEDATA, OnReceiveData)
	ON_MESSAGE(WM_TEST_TIMER, OnTestRoutine)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CRcmprevDlg message handlers

BOOL CRcmprevDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.
```

```
	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu)
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here


	m_pComm.SetComPort(1, 9600,8, 0, 1);
	m_pComm.CreateCommInfo();
	if(!m_pComm.OpenComPort()){
		MessageBox("Can't open Comport.");
		m_Connected = FALSE;
	}
	else{
		m_pComm.SetHwnd(this->m_hWnd);
		m_Connected = TRUE;
		m_comctrl.AddString("COM 1");
	}
	TimeInterval = m_timeinterval;
	Duration = m_duration*1000;
	FirstTic = TRUE;

	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CRcmprevDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CRcmprevDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
```

```
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CRcmprevDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

void CRcmprevDlg::OnCommSet()
{
	// TODO: Add your control notification handler code here
	CString portno="1";

	if (m_Connected == TRUE) {
		m_pComm.DestroyComm();
	}
	if(dlg.DoModal() == IDOK){
		Port = dlg.m_Port+1;
		m_pComm.SetComPort(dlg.m_Port+1, dlg.m_Rate, dlg.m_Bit, dlg.m_Parity, dlg.m_Stop)
		m_pComm.CreateCommInfo();
		if(!m_pComm.OpenComPort()){
			MessageBox("Can't open Comport.");
			m_Connected = FALSE;
		}
		else{
			m_pComm.SetHwnd(this->m_hWnd);
			m_Connected = TRUE;
			portno.Empty();
			portno.Format("COM %d 선택되었음.",dlg.m_Port+1);
			m_comctrl.ResetContent();
			m_comctrl.AddString(portno);
			return ;
		}
	}
}


void CRcmprevDlg::OnTelStart()
{
	// TODO: Add your control notification handler code here
	FirstTic =TRUE;
	MyTimer.SetMulTimer(m_hWnd,WM_TEST_TIMER,TimeInterval);

}

void CRcmprevDlg::OnTelStop()
{
	// TODO: Add your control notification handler code here

	if (m_Connected == FALSE) {
			MessageBox("Can't open Comport. Comport Setup First Please!");
			return;
	}


	MyTimer.KillMulTimer();

	m_gndctrl.ResetContent();
	m_gndctrl.AddString("자동 종료명령 처리 완료");

}



void CRcmprevDlg::SendData(BYTE Com)
{
	BYTE ahn[6];
	int i=0;


	ahn[0] = 0x3e;
	ahn[1] = 0x48;
	ahn[2] = 0x31;
	ahn[3] = 0x41;
	ahn[4] = 0x30;
	ahn[5] = 0x0d;
	m_pComm.WriteCommBlock((char *)ahn, 3);
}

void CRcmprevDlg::Modem_Init()
{
```

```
        int i=0 ;

    BYTE tempblock[14] ;
        tempblock[0] = 0x72;
        tempblock[1] = 0x72;
        tempblock[2] = 0x72;
        tempblock[3] = 0x72;
        tempblock[4] = 0xb9;
        tempblock[5] = 0x65;
        tempblock[6] = 0x98;
        tempblock[7] = 0x4c;
        tempblock[8] = 0x26;
        tempblock[9] = 0x82;
        tempblock[10] = 0xc1;
        tempblock[11] = 0xfe;
        tempblock[12] = 0xd9;


         m_pComm.WriteCommBlock((char *)tempblock, 13);

                .
}


LONG CRcmprevDlg::OnReceiveData(UINT WParam, LONG LParam)
{
        CString tmp;
        int cnt=0, i;
        m_gndctrl.ResetContent();

        // 컴포트로부터 데이타를 받으면, 버퍼에 저장한다.

        if(hexa == FALSE) {
                cnt = m_rcvdata.GetLength();
                m_rcvdata+=m_pComm.abIn;
                cnt = m_rcvdata.GetLength()-cnt;

                for(i = 0; i < cnt; i++)
                {
                        if (logok == TRUE)
                                fprintf(fpt,"%c", m_pComm.abIn[i]);
                }
                fprintf(fpt,"\n");
                m_gndctrl.ResetContent();
                m_rcvdatar = m_rcvdata;

        }
        else {

                m_rcvdata=m_pComm.abIn;
                cnt = m_rcvdata.GetLength();
                m_rcvdata.Empty();
                m_rcvdata += m_rcvdatar;
                for(i = 0; i < cnt; i++)
                {
                        if (m_pComm.abIn[i] == '<')
                        {
                                m_rcvdata = m_rcvdata + "\r\n";
                                tcnt++;
                                if (logok == TRUE)
                                {
                                        fprintf(fpt," \n");
                                        fprintf(fpt,"%d  ", GetTickCount())
                                }

                        }
                        m_rcvdatar.Format("%c", m_pComm.abIn[i]);
                        if (logok == TRUE)
                                fprintf(fpt,"%c", m_pComm.abIn[i]);

                        m_rcvdata += m_rcvdatar;

                m_rcvdatar = m_rcvdata;
                }

        }

                cnt = m_rcvdata.GetLength();
                m_rcvdata.Delete(0,cnt-124);

        UpdateData(FALSE);
        m_gndctrl.ResetContent();
```

```
        m_gndctrl.AddString("텔리메트리 데이타 수신 성공");
        return TRUE;
}


void CRcmprevDlg::OnLog()
{
        CString fn;
        CFileDialog m_ldFile(TRUE);
        if (m_ldFile.DoModal()==IDOK)
        {
                fn = m_ldFile.GetFileName();
                UpdateData(FALSE);
        }

                fpt = fopen(fn, "w");
                if(fpt == NULL) {
                        TRACE("File Open Error");
                        logok = FALSE;
                }
                else {
                        logok = TRUE;
                        m_gndctrl.ResetContent();
                        m_gndctrl.AddString("데이타 로깅 선택되었음");
                }

}

void CRcmprevDlg::OnExit()
{
        // TODO: Add your control notification handler code here

        if(logok == TRUE) {
        fclose(fpt);
        logok = FALSE;
        }
        if (m_Connected == TRUE) {
                m_pComm.DestroyComm();
        }

        MyTimer.KillMulTimer();
        CDialog::OnOK();
}


void CRcmprevDlg::OnCheck1()
{
        // TODO: Add your control notification handler code here
        fpt = fopen("DFTData.txt", "w");
        if(fpt == NULL) {
                TRACE("File Open Error");
                logok = FALSE;
        }
        else {
                logok = TRUE;
                m_gndctrl.ResetContent();
                m_gndctrl.AddString("디폴트 데이타 로깅 선택되었음");
        }

}

void CRcmprevDlg::OnRCStart()
{
        // TODO: Add your control notification handler code here

        if (m_Connected == FALSE) {
                        MessageBox("Can't open Comport. Comport Setup First Please!")
                        return;
        }


        char c[] = ">1A0..3\r";

        m_pComm.WriteCommBlock((char *)c, 8);
        m_gndctrl.ResetContent();
        m_gndctrl.AddString(">1A0..3\r 송신완료");

}

void CRcmprevDlg::OnRCStop()
{
        // TODO: Add your control notification handler code here
```

```
        if (m_Connected == FALSE) {
                        MessageBox("Can't open Comport. Comport Setup First Please!")
                        return;
        }
        char c[] = "d:P 1,15.0\r";


        m_gndctrl.ResetContent();
        m_gndctrl.AddString("d:P 1,15.0\r 송신완료");

}

void CRcmprevDlg::OnChangeEdit1()
{
        // TODO: If this is a RICHEDIT control, the control will not
        // send this notification unless you override the CDialog::OnInitDialog()
        // function and call CRichEditCtrl().SetEventMask()
        // with the ENM_CHANGE flag ORed into the mask.

        // TODO: Add your control notification handler code here
        int cnt=0;

        if (m_Connected == FALSE) {
                MessageBox("Can't open Comport. Comport Setup First Please!");
                return;
        }
        UpdateData(TRUE);

        if (*((LPCTSTR)(m_editvalue.Right(1))) == 0xa) {

                cnt = m_editvalue.GetLength();
                m_pComm.WriteCommBlock((char *)((LPCTSTR)m_editvalue), cnt-1);
                m_editvalue.Empty();
        }
        UpdateData(FALSE);
}

void CRcmprevDlg::OnOK()
{
        // TODO: Add extra validation here

//      CDialog::OnOK();
}

void CRcmprevDlg::OnRadio1()
{
        // TODO: Add your control notification handler code here
                        hexa = TRUE;
}

void CRcmprevDlg::OnRadio2()
{
        // TODO: Add your control notification handler code here
                        hexa = FALSE;
}


void CRcmprevDlg::OnTimer(UINT nIDEvent)
{
        // TODO: Add your message handler code here and/or call default
        if(nIDEvent == 0) {

        Modem_Init();
        }

        if(nIDEvent == 1){ }
        CDialog::OnTimer(nIDEvent);
}

LONG CRcmprevDlg::OnTestRoutine(WPARAM wParam, LPARAM lParam)
{
        int FinalTic,CompDuration,cnt=0;
        FinalTic = GetTickCount();
        if (FirstTic)
        {
                InitialTic = FinalTic;
                FirstTic =FALSE;
        }
        CompDuration = FinalTic - InitialTic;
        if(CompDuration > Duration)   MyTimer.KillMulTimer();
```

```
//        TRACE("%d\n", tttt);
        if (m_Connected == FALSE) {
                        MessageBox("Can't open Comport. Comport Setup First Please!");
                        return 0;
        }

        char c[] = ">1A0..3\r";

        m_pComm.WriteCommBlock((char *)c, 8);

        m_gndctrl.ResetContent();
        m_gndctrl.AddString(">1A0..3\r 송신완료");
        return 0;
}

void CRcmprevDlg::OnChangeInterval()
{
        // TODO: If this is a RICHEDIT control, the control will not
        // send this notification unless you override the CDialog::OnInitDialog()
        // function and call CRichEditCtrl().SetEventMask()
        // with the ENM_CHANGE flag ORed into the mask.

        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        TimeInterval = m_timeinterval;
}

void CRcmprevDlg::OnChangeDuration()
{
        // TODO: If this is a RICHEDIT control, the control will not
        // send this notification unless you override the CDialog::OnInitDialog()
        // function and call CRichEditCtrl().SetEventMask()
        // with the ENM_CHANGE flag ORed into the mask.

        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        Duration = m_duration*1000;
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////
Comm.cpp
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////
//Comm.cpp Rs232c통신을 하기 위한 클래스
#include "stdafx.h"
#include "comm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

IMPLEMENT_DYNCREATE(CComm, CObject)


CComm::CComm( )
{
   idComDev=NULL;
   bFlowCtrl= FC_XONXOFF ;
   fConnected      = FALSE ;
}

CComm::~CComm( )
{
    DestroyComm();
}

//BEGIN_MESSAGE_MAP(CComm, CObject)
        //{{AFX_MSG_MAP(CComm)
                // NOTE - the ClassWizard will add and remove mapping macros here.
        //}}AFX_MSG_MAP
//END_MESSAGE_MAP()


////////////////////////////////////////////////////////////////////////////
// CComm message handlers
//CommWatchProc()
```

```
//통신을 하는 프로세저 즉 데이타가 들어왔을대 감시하는
//루틴 본루틴은 OpenComPort 함수 실행시 프로시저로 연결됨
//OpenComPort 함수 참조
DWORD CommWatchProc(LPVOID lpData)
{
   DWORD       dwEvtMask ;
   OVERLAPPED  os ;
   CComm*      npComm = (CComm*) lpData ;
   char InData[MAXBLOCK + 1];
         int         nLength ;
   //idCommDev 라는 핸들에 아무런 com 포트가 안붙어 잇으면
   // 에라 리턴
   if ( npComm == NULL ||
      !npComm->IsKindOf( RUNTIME_CLASS( CComm ) ) )
      return (DWORD)(-1);

   memset( &os, 0, sizeof( OVERLAPPED ) ) ;


   os.hEvent = CreateEvent( NULL,    // no security
                            TRUE,    // explicit reset req
                            FALSE,   // initial event reset
                            NULL ) ; // no name
   if ( os.hEvent == NULL )
   {
      MessageBox( NULL, "Failed to create event for thread!", "comm Error!",
                  MB_ICONEXCLAMATION | MB_OK ) ;
      return ( FALSE ) ;
   }

   if (!SetCommMask(npComm->idComDev, EV_RXCHAR ))
      return ( FALSE ) ;

   while (npComm->fConnected )
   {
                  dwEvtMask = 0 ;

                  WaitCommEvent(npComm->idComDev, &dwEvtMask, NULL );

                  if ((dwEvtMask & EV_RXCHAR) == EV_RXCHAR)
                  {

                           do
                     {
                              memset(InData,0,80);
                                    if (nLength = npComm->ReadCommBlock((LPSTR) InData, MAXBLOCK ))
                                    {
                        npComm->SetReadData(InData);
                  //이곳에서 데이타를 받는다.
                                    }
                     }
                     while ( nLength > 0 ) ;
                  }
   }


   CloseHandle( os.hEvent ) ;


   return( TRUE ) ;

}
//데이타를 일고 데이타를 읽었다는
//메세지를 리턴한다.
void CComm::SetReadData(LPSTR data)
{
   lstrcpy((LPSTR)abIn,(LPSTR)data);
   //ConverData
   //설정된 윈도우에 WM_RECEIVEDATA메세지를
   //날려주어 현제 데이타가 들어왔다는것을
   //알려준다.
   SendMessage(m_hwnd,WM_RECEIVEDATA,0,0);
}
//메세지를 전달할 hwnd설정
void CComm::SetHwnd(HWND hwnd)
{
   m_hwnd=hwnd;
}
//컴포트를 설정한다.
void CComm::SetComPort(int port,DWORD rate,BYTE bytesize,BYTE stop,BYTE parity)
{
   bPort=port;
```

```
   dwBaudRate=rate;
   bByteSize=bytesize;
   bStopBits=stop;
   bParity=parity;
}
//XonOff 즉 리턴값 더블 설정
void CComm::SetXonOff(BOOL chk)
{
   fXonXoff=chk;
}
void CComm::SetDtrRts(BYTE chk)
{
   bFlowCtrl=chk;
}

//컴포트 정보를 만든다.
//이것을 만들때 이전에 할일이
// SetComPort(); -> SetXonOff() ->SetDtrRts() 한다음 설정한다.

BOOL CComm::CreateCommInfo()
{

   osWrite.Offset = 0 ;
   osWrite.OffsetHigh = 0 ;
   osRead.Offset = 0 ;
   osRead.OffsetHigh = 0 ;

//이벤트 창구 설정
   osRead.hEvent = CreateEvent( NULL, TRUE, FALSE, NULL ) ;
   if (osRead.hEvent == NULL)
   {
      return FALSE ;
   }
   osWrite.hEvent = CreateEvent( NULL,   TRUE,   FALSE,   NULL ) ;
   if (NULL == osWrite.hEvent)
   {
      CloseHandle( osRead.hEvent ) ;
      return FALSE;
   }


   return TRUE ;

}

//com 포트를 열고 연결을 시도한다.
//OpenComport()
BOOL CComm::OpenComPort( )
{
   char       szPort[ 15 ] ;
   BOOL       fRetVal ;
   COMMTIMEOUTS  CommTimeOuts ;


   if (bPort > MAXPORTS)
      lstrcpy( szPort, "\\\\.\\TELNET" ) ;
   else
      wsprintf( szPort, "COM%d", bPort ) ;

   // COMM device를 화일형식으로 연결한다.

   if ((idComDev =
      CreateFile( szPort, GENERIC_READ | GENERIC_WRITE,
                  0,                    // exclusive access
                  NULL,                 // no security attrs
                  OPEN_EXISTING,
                   FILE_ATTRIBUTE_NORMAL |
                  FILE_FLAG_OVERLAPPED, // overlapped I/O
                  NULL )) == (HANDLE) -1 )
      return ( FALSE ) ;
   else
   {
      //컴포트에서 데이타를 교환하는 방법을 char단위를 기본으로 설정
      //하자
      SetCommMask( idComDev, EV_RXCHAR ) ;
      SetupComm( idComDev, 4096, 4096 ) ;
      //디바이스에 쓰레기가 있을지 모르니까 깨끗이 청소를 하자!
      PurgeComm( idComDev, PURGE_TXABORT | PURGE_RXABORT |
                                    PURGE_TXCLEAR | PURGE_RXCLEAR )


      CommTimeOuts.ReadIntervalTimeout = 0xFFFFFFFF ;
```

```
      CommTimeOuts.ReadTotalTimeoutMultiplier = 0 ;
      CommTimeOuts.ReadTotalTimeoutConstant = 1000 ;
      CommTimeOuts.WriteTotalTimeoutMultiplier = 0 ;
      CommTimeOuts.WriteTotalTimeoutConstant = 1000 ;
      SetCommTimeouts( idComDev, &CommTimeOuts ) ;
   }

   fRetVal = SetupConnection() ;

   if (fRetVal)//연결이 되었다면 fRetVal TRUE이므로
   {
      fConnected = TRUE ;//연결되었다고 말해줌
      //프로시전를 CommWatchProc에 연결하니까 나중에 데이타가 왔다갔다
      //하면 모든 내용은 CommWatchProc가 담당한다.
      AfxBeginThread((AFX_THREADPROC)CommWatchProc,(LPVOID)this);
   }
   else
   {
      fConnected = FALSE ;
      CloseHandle( idComDev) ;
   }

   return ( fRetVal ) ;

}

//화일로 설정된 컴포트와 실질 포트와 연결을 시킨다.
//SetupConnection 이전에 CreateComPort를 해주어야 한다.
BOOL CComm::SetupConnection()
{
   BOOL        fRetVal ;
   BYTE        bSet ;
   DCB         dcb ;

   dcb.DCBlength = sizeof( DCB ) ;

   GetCommState( idComDev, &dcb ) ;//dcb의 기본값을 받는다.


   //이부분을 수정해야 합니다.
   dcb.BaudRate = dwBaudRate;//전송속도
   dcb.ByteSize = bByteSize ;//데이타비트
   dcb.Parity = bParity;//패리티 체크
   dcb.StopBits = bStopBits;//스톱비튼

   dcb.fOutxDsrFlow =0 ;//Dsr Flow
   dcb.fDtrControl = DTR_CONTROL_ENABLE ;//Dtr Control
   dcb.fOutxCtsFlow = 0 ;//Cts Flow
   dcb.fRtsControl = RTS_CONTROL_ENABLE ; //Ctr Control
   dcb.fInX = dcb.fOutX = 1 ; //XON/XOFF 관한것
   dcb.XonChar = ASCII_XON ;
   dcb.XoffChar = ASCII_XOFF ;
   dcb.XonLim = 100 ;
   dcb.XoffLim = 100 ;
   dcb.fBinary = TRUE ;
   dcb.fParity = TRUE ;



   dcb.fBinary = TRUE ;
   dcb.fParity = TRUE ;

   fRetVal = SetCommState( idComDev, &dcb ) ; //변경된 Dcb 설정

   return ( fRetVal ) ;

}

//컴포트로 부터 데이타를 읽는다.
int CComm::ReadCommBlock(LPSTR lpszBlock, int nMaxLength )
{
         BOOL       fReadStat ;
         COMSTAT    ComStat ;
         DWORD      dwErrorFlags;
         DWORD      dwLength;

         // only try to read number of bytes in queue
         ClearCommError( idComDev, &dwErrorFlags, &ComStat ) ;
         dwLength = min( (DWORD) nMaxLength, ComStat.cbInQue ) ;

         if (dwLength > 0)
         {
```

```
                    fReadStat = ReadFile( idComDev, lpszBlock,
                                          dwLength, &dwLength, &osRead ) ;
                    if (!fReadStat)
                    {
          //이곳에 에라를 넣는것이다.
          //즉 ReadFile 했을때 데이타가 제대로 안나오면 fReadState에 여러
          //에라 코드를 리턴한다. 이때 복구할수있으면 좋지만 실질적인
          //복구가 불가능하다 따라서 재송출을 해달라는 메세지를 해주는것이
          //좋다.
                    }
   }

   return ( dwLength ) ;

}
//컴포트를 완전히 해제한다.

BOOL CComm::DestroyComm()
{


   if (fConnected)
      CloseConnection( ) ;


   CloseHandle( osRead.hEvent ) ;
   CloseHandle( osWrite.hEvent ) ;

   return ( TRUE ) ;

}

//연결을 닫는다.
BOOL CComm::CloseConnection()
{

   // set connected flag to FALSE

   fConnected = FALSE ;

   // disable event notification and wait for thread
   // to halt

   SetCommMask( idComDev, 0 ) ;


   EscapeCommFunction( idComDev, CLRDTR ) ;


   PurgeComm( idComDev, PURGE_TXABORT | PURGE_RXABORT |
                                PURGE_TXCLEAR | PURGE_RXCLEAR ) ;
   CloseHandle( idComDev ) ;
   return ( TRUE ) ;

}



BOOL CComm::WriteCommBlock( LPSTR lpByte , DWORD dwBytesToWrite)
{

          BOOL        fWriteStat ;
          DWORD       dwBytesWritten ;


          fWriteStat = WriteFile( idComDev, lpByte, dwBytesToWrite,
                                  &dwBytesWritten, &osWrite ) ;


          if (!fWriteStat)
          {
      //컴포트에 데이타를 제대로 써넣지 못했을경우이다.
      //이때는 어떻게 할까 그것은 사용자 마음이겠다.
      //다시 보내고 싶으면 재귀송출을 하면 된다.
      //그러나 주의점 무한 루프를 돌수 있다는점을 생각하라
          }
          return ( TRUE ) ;

}
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////
```

CommSet.cpp

```
////////////////////////////////////////////////////////////////////////////////////////////////////////
//////
// CommSet.cpp : implementation file
//

#include "stdafx.h"
#include "RCMPrev.h"
#include "CommSet.h"
#include "rcmprevDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CCommSet dialog


CCommSet::CCommSet(CWnd* pParent /*=NULL*/)
        : CDialog(CCommSet::IDD, pParent)
{
        //{{AFX_DATA_INIT(CCommSet)
        m_Parity = 0;
        m_Bit = 0;
        m_Rate = 0;
        m_Stop = 0;
        m_Port = -1;
        //}}AFX_DATA_INIT
}


void CCommSet::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CCommSet)
        DDX_Text(pDX, IDC_PARITY, m_Parity);
        DDX_Text(pDX, IDC_BIT, m_Bit);
        DDX_Text(pDX, IDC_RATE, m_Rate);
        DDX_Text(pDX, IDC_STOP, m_Stop);
        DDX_CBIndex(pDX, IDD_PORTCB, m_Port);
        //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CCommSet, CDialog)
        //{{AFX_MSG_MAP(CCommSet)
        ON_CBN_SELCHANGE(IDD_PORTCB, OnSelchangePortcb)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCommSet message handlers

BOOL CCommSet::OnInitDialog()
{
        CDialog::OnInitDialog();

        // TODO: Add extra initialization here

        m_Port = 0;
        m_Parity = 0;
        m_Rate = 9600;
        m_Stop = 1;
        m_Bit = 8;

        // 그리고 다이얼로그에 출력한다.

        UpdateData(FALSE);

        return TRUE;  // return TRUE unless you set the focus to a control
                      // EXCEPTION: OCX Property Pages should return FALSE
}

void CCommSet::OnSelchangePortcb()
{
        // TODO: Add your control notification handler code here

        // 컴포트 번호 콤보박스가 변경되면,
        // 값을 얻어오고,
```

```
        UpdateData(TRUE);

        CButton *tButton = (CButton *)GetDlgItem(IDOK)

        // 현재 바뀐 값이 기존것이면,
        // 버튼을 비활성화
        // 그렇지 않으면, 활성화 한다.
        if(m_Port != (Port - 1))
                tButton->EnableWindow(TRUE);
        else
                tButton->EnableWindow(FALSE);

}

void CCommSet::OnOK()
{
        // TODO: Add extra validation here

        UpdateData(TRUE);
        CDialog::OnOK();
}
```