한국항행학회 논문지 제14권 제3호 2010년 6월

CANopen 프로토콜을 이용한 전동밸브 제어시스템 설계 및 구현

The Design and Implementation of the Motorized Valve Control System using CANopen Protocol

이명의*, 신근수*, 양성현**

Myung-Eui Lee*, Keun-Soo shin* and Sung-Hyun Yang**

요 약

본 논문은 CANopen 프로토콜을 이용하여 전동 구동기 제어시스템을 설계하고 개발하는데 관련된 논문이다. 이 논문에서는 네트워크의 물리계층(1계층) 및 데이터 링크(2계층)은 CAN네트워크 프로토콜을 이용하였으며, 그 위의 상위 프로토콜은 CANopen 프로토콜을 이용하였다. 전동밸브 제어기는 PIC 마이크로프로세서를 이용하여 구현하였으며, 제어시스템 사용자를 위한 서버 응용프로그램은 C#언어로 작성하였다. 실시간 실험을 통하여 본 논문에서 구현된 전동밸브 제어 시스템을 평가하여 설계된 바와 같이 동작하였다.

Abstract

This paper deals with the design and development of the motorized valve control system using CANopen protocol. The CAN network protocol is used in the physical layer(layer 1) and data link layer(layer 2), and other upper network layer above that layer 1 and 2 utilize the CANopen protocol in this paper. The motorized valve controller is implemented by a PIC microprocessor, and the server application software for the control system user is written in C# language. In particular the CANopen protocol is widely used int the area such as ship automation systems and marine transportation systems. The experimental result of the proposed control system implemented in this paper is evaluated via real-time experiments, which works well as designed

Key words : CAN, CANopen, Field Bus, Actuator, Motorized valve controller

I. 서 론

디지털 신호처리의 발전으로 인해 최근 제어시스템에서 디지털신호를 사용하는 장치(Device)의 개수가 증가하고, 서로 통신하는 데이터양이 증가하면서, 과거의 point to point 네트워크 시스템으로 모든 제어 시스템과 통신하기가 어렵게 되었다. 이러한 단점을 극복하고자, 산업현장에서는 토큰링 방식으로 노드를 설계하는 필드버스(FieldBus)시스템의 사용이 증가하게 되었다. 필드버스 시스템의 장점은 배선을 단순화시켜 비용을 절감하고 개발시간을 단축하며, 서로 공용된 프로토콜을 이용하여, 추후 각종 장치의

* 한국기술교육대학교 정보기술공학부(Dept. of Information Technology Eng., Korea University of Technology and Education.)
** 광운대학교 전자정보공과대학(College of Electronics and information, kwangwoon university)
· 교신저자 (Corresponding Author) : 신근수
· 투고일자 : 2010년 4월 13일
· 심사(수정)일자 : 2010년 4월 14일 (수정일자 : 2010년 6월 21일)
· 게재일자 : 2010년 6월 30일

추가 및 교체함에 드는 비용을 절감할 수 있다. 본 논문은 현재 주목받고 있는 필드버스 시스템인 CAN과 CAN의 상위 프로토콜인 CANopen을 사용하여, 구동기(Actuator)를 제어하는 시스템을 개발하여, 현장에서 필드버스 시스템을 도입하는 데 대한 장점을 연구하고자 한다.[1]

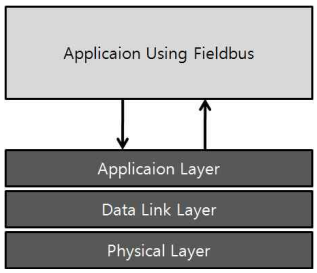

그림 1. 필드버스 시스템의 구조
Fig. 1. Architecture of field bus system

Ⅱ. CANopen 프로토콜

2-1 CAN의 물리적 계층

CAN 통신은 OSI 7계층 중 하위 2개의 계층(물리적 계층, 데이터링크 계층)에 대해 정의를 내린다. LAN보다 적은 범위에서 사용되고, 다른 네트워크와 연계할 필요성이 없으므로, OSI 모델 중 3,4,5계층을 구현할 필요가 없으며, 바로 상위 계층과 통신한다(CANopen, DeviceNET 등).

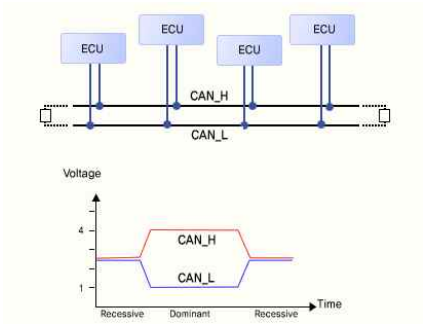

그림 2. CAN 노드의 구조
Fig. 2. Architecture of CAN node system

CAN의 매체 접근방식은 기본적으로 CSMA/CD(Carrier Sense Multiple Access With Collusion Detection)방식을 사용한다.

이더넷과 달리, 메시지끼리 충돌이 발생하면, 메시지에 포함된 우선순위에 따라, 데이터를 보내는 노드가 결정된다. 높은 우선순위가 있는 노드가 네트워크에 접근하는 권한을 가지게 되며, 이 우선순위가 그 노드의 식별자(ID)가 된다. 이러한 방식으로 충돌을 피하는 기능을 CSMA/CD-AMP(Arbitration by Message Priority) 또는 CSMA-NDA(None-destructive Arbitration)이라 한다.[2][3]

2-2 CANopen

CANopen은 CAN의 상위프로토콜로서, 이벤트 동작 및 제어, 긴급 명령 전송, 시스템 처리와 같은 동작 중심적인 네트워크를 위해 설계되었다. CiA(Car in Automation)에 의해 제정되었으며, 북미와 아시아 지역보다 유럽지역에서 널리 사용된다.[4]

CANopen은 각 프로토콜을 비슷한 성질별로 분류하여 객체로 묶는다. 대표적인 객체로는 Network Management , Process Data Object, Service Data Object, Error Control등이 있다.[5]

● Object Dictionary(OD)

CANopen은 비슷한 역할을 하는 통신을 객체로 묶어서 분류하고, 통신 객체는 네트워크 설계 전에 미리 정의한다. 이러한 객체를 Object Dictionary(OD)라고 한다.

OD는 16비트 인덱스 주소를 가지고 있고, 각각의 하위 객체 원소를 구분하기 위해 추가로 8비트의 하위 인덱스 주소를 사용한다. OD는 Electronic Data Sheet(EDS)안에 정의되어 있다. EDS는 네트워크 설계에 대한 내용을 담고 있는 일종의 데이터베이스이다. EDS를 노드에서 사용하기 위해 ASCII포맷을 이용하여 기술하는데, 이러한 파일을 Device Configuration File(DCF)이라 한다.. EDS와 DCF는 data carrier의 형태로 제공되며, 이것은 서로 다른 장치에서 다운로드 하거나 인터넷을 통해 전송되기도 한다.

● Network Management(NMT)

NMT 객체는 네트워크를 관리하는 여러 객체로 구성되어 있다. Boot-up 메시지, Heartbeat 프로토콜, 그 외의 MNT 프로토콜을 포함한다. 단일 CAN 프레임에 1바이트의 데이터필드, Boot-up 메시지, Heartbeat 프로토콜을 포함한다.

● Process Data Object(PDO)

PDO는 실시간으로 데이터를 전송하며, 데이터의 오버헤드를 방지하기 위해 최대 8바이트로 제한한다. PDO는 이벤트의 발생이나 원격에서 데이터를 요청하였을 때 사용한다.

PDO는 사전에 각 노드에서 사용되는 PDO를 정의한다. PDO는 한 노드에 최대 4개를 설정할 수 있다.

PDO는 송신 측과 수신 측으로 나뉜다. 송신 측을 producer, 수신 측을 consumer라고 하며 하나의 producer에서 다수의 consumer로 데이터 전송이 가능하다. 메시지 패킷은 producer에서만 생성이 가능하고, consumer에서 데이터를 보내야 할 때는 producer에게 패킷을 요청한다. PDO 전송은 실시간으로 빠른 데이터를 전송하지만 데이터의 신뢰도를 확신하지 못한다. [6]

COB-ID	Byte 0	Byte 1	Byte2
0x180 + Node ID	8-bits digital in	LSB 16-bit analog in	MSB 16-bit analog in

그림 3. PDO 패킷
Fig. 3. PDO packet

● Service Data Object(SDO)

SDO의 전송은 두 개의 네트워크 노드 사이에서 일대일 연결 형태로 전송된다.

SDO는 일반적으로 서버/클라이언트로 통신한다. 클라이언트에서 서버의 OD를 요청하면, 서버는 OD에 대한 응답을 클라이언트로 전송한다. 데이터를 요청하고 응답하는 데는 CAN 메시지를 사용하며, 그러므로 오버헤드가 발생한다.

Byte 0	Byte 1- 7
SDO command Specifier	Up to 7 bytes of data

그림 4. SDO 패킷
Fig. 4. SDO packet

● Error Control

Error Control 메시지는 장치 자체 내에서의 심각한 오류나, 장치 간의 통신 오류가 발생하였을 때 발생한다. 그 어떤 메시지보다 높은 우선순위를 가진다.

Ⅲ. CANopen 노드설계 및 실험 결과

기본적으로 하위 프로토콜은 CAN 통신을, 상위 프로토콜을 CANopen을 사용하여 설계한다.

Control Server : PC에 BECKOFF사의 CAN제어 보드를 사용하였다. CANopen 프로토콜 제어를 위해 TwinCAT V2.10을 사용하고, 서버 응용프로그램은 visual studio 2005의 C#을 사용하여 작성하였다.

구동기 : MicroChip사의 dsPIC 30F5011과, 컴파일러로 MicroChop사의 MPLAB 3.0을 사용하였다.[7]

3-1 Producer에서 전송

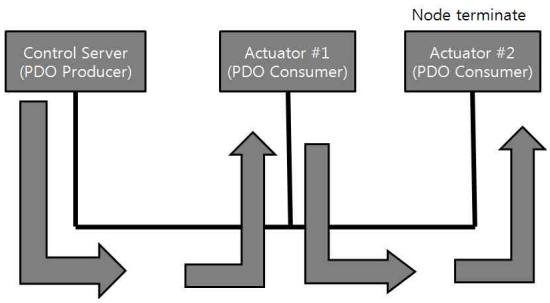

그림 5. PDO Producer에서 Consumer로 전송
Fig. 5. transmitting PDO from Producer to Consumer

PDO Producer에서 발생한 패킷은 구동기 #1을 지나 구동기 #2까지 전송된다. 각 구동기는 패킷의 COB-ID를 확인하여 데이터의 목적지를 확인하여. 자신에게 발송된 데이터면 그에 맞는 동작을 한다. 구

동기#2는 terminate노드이므로, 패킷은 구동기#2에서 소멸된다.[4]

3-2 Producer로 귀환

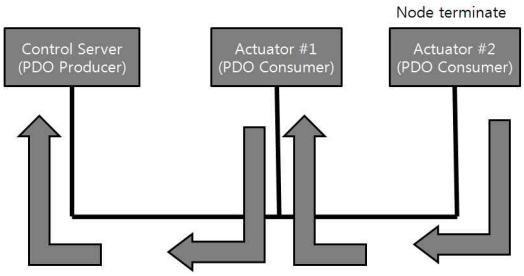

그림 6. PDO Consumer에서 Producer로 전송
Fig. 6. transmitting PDO from Consumer to Producer

terminate 노드인 Actuator#2에서 Producer로 패킷을 전송한다. 패킷은 Actuator#1을 거친 다음 PDO Producer로 전송된다. 각 구동기에서는 패킷이 지나갈 때 자신의 COB-ID와 상태를 실어 보낸다. PDO Producer는 수신된 패킷의 COB-ID를 통하여 구동기의 현재 상태를 파악한다..[4]

3-3. Control Server

다음은 Control Server의 데이터 흐름도이다.

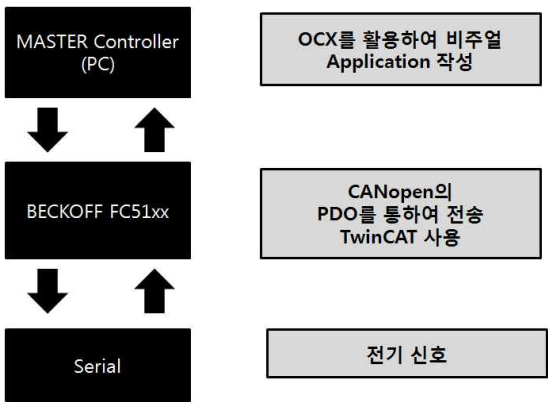

그림 7. Control server 구조
Fig. 7. Architecture of Control server

Control Server의 응용프로그램은 TwinCAT에서 제공하는 OCX를 사용하여 작성하였다. 응용프로그램에서 제어 명령을 내리면, OCX를 통하여 TwinCAT 제어프로그램의 데이터 전송 큐에 메시지가 쌓이게 된다. 또한 응용프로그램에서 일정한 시간이 흐르면 TwinCAT에 속해 있는 수신 큐의 메시지를 가져와 화면에 표시해준다.

TwinCAT은 일정한 시간마다 토큰을 노드로 전송한다. 이 토큰에는 전송 큐에 있는 데이터를 읽어 각 노드에 전송하며 또한 노드에서 전송받은 데이터를 수신 큐에 쌓아 놓는다.

3-4 데이터 송신 패킷 구조

데이터 송신은 16비트 아날로그 신호를 사용한다.

그림 8. 송신 패킷 구조
Fig. 8. Structure of sender packet

송신패킷은 데이터 출력으로 하위 2개의 비트를 사용하는데, 비트가 00(b)이면 STOP 제어 신호, 10(b)이면 CLOSE 제어 신호 11(b)이면 OPEN 제어 신호를 전송한다.

3-5 데이터 수신 패킷 구조

데이터 수신은 8비트 디지털 신호를 사용한다.

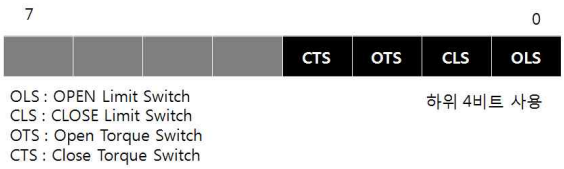

그림 9. 수신 패킷 구조
Fig. 9. Structure of receiver packet

OLS, CLS, OTS, CTS는 구동기의 상태를 나타내는 스위치들이다. 각 스위치가 눌리면 더 이상 유량 제어 모터가 동작하지 않는다.

OLS : 구동기가 최대로 개방되어 있을 때 발생
CLS : 구동기가 완전히 닫혔을 때 발생

OTS : 구동기가 OPEN동작 상태일 때, 외부의 문제로 더는 진행하지 못할 때(이물질의 유입 등) 모터의 과열을 막기 위해 모터를 정지시키고 OTS 메시지를 발생

CTS : 구동기가 CLOSE 동작 상태일 때, 외부의 문제로 더는 진행하지 못할 때(이물질의 유입 등) 모터의 과열을 막기 위해 모터를 정지시키고 CTS메시지를 발생

하위 비트가 모두 0이면 OPEN 상태이다.

3-5 RxPDO 데이터의 처리

Controller Sever에서 각 구동기 제어 데이터를 전송하면, 전송된 데이터를 수신하여 처리하게 된다. 수신한 데이터는 OPEN, CLOSE, STOP 3개이다.

구동기에서 모터 제어는 2개의 디지털 신호로 이루어져 있다. 이들은 정방향(Open) 회전 On, Off, 역방향(Close) 회전 On, Off 신호로 구성된다. 동시에 두 방향으로 동작하라는 데이터가 들어올 때에는 역방향 회전을 우선하여 처리한다.

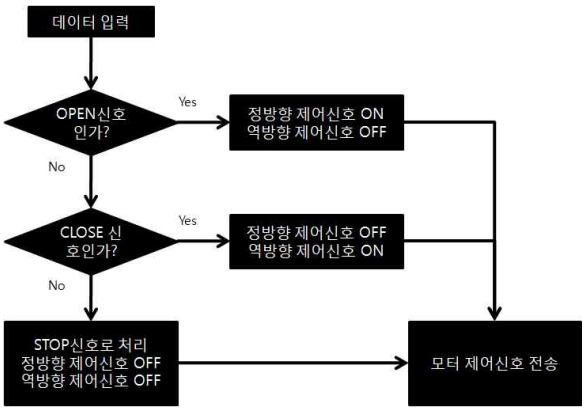

그림 10. RxPDO 수신 데이터 처리
Fig. 10. Flowchart of RxPDO Data

데이터를 확인하여 정지 명령이면 모터를 모두 RELAY_OFF로 설정한다. 그렇지 않을 때에는 데이터 신호에 따라 정방향, 역방향으로 모터를 제어한다.

3-6 TxPDO 데이터 처리

그림 11. TxPDO 수신 데이터 처리
Fig. 11. Flowchart of TxPDO Data

Control Server에서 데이터 전송 요청이 들어오면 구동기 Contoller는 GPIO에 연결된 센서를 검색한다. 센서에 ActiveHigh 신호가 들어오면 그 신호를 unsigned char 하위비트부터 차례로 입력하여 전송한다.

OLS, CLS, OTS, CLS 이외의 상태는 구동기가 개방된 상태이므로 0x00을 전송한다. Control Server에서는 Ox00이 수신이 되면 OPEN 상태로 간주한다.

3-7 실험 결과

노드에 전원이 인가되면, boot-up 과정을 거친 후 Ready 상태가 된다. Controller server는 패킷을 보내 현재 노드의 상태를 확인한 후 화면에 표시한다. 표시화면에서 원하는 동작을 클릭하게 되면 구동기가 동작을 하는데, 구동기 밸브의 스위치를 통하여 구동기가 제어되고 있음을 확인할 수 있다.

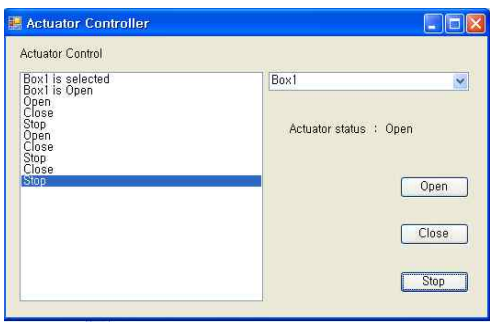

그림 12. 구동기 OPEN 상태(Control Server)
Fig. 12. Actuator is OPEN(Control Server)

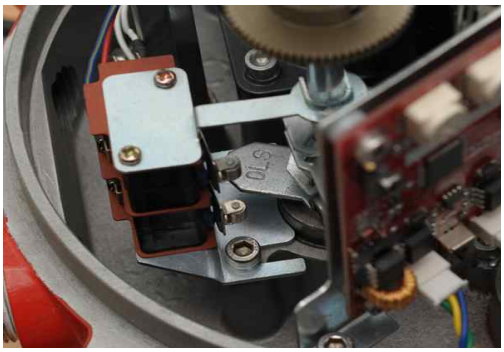

그림 13. 구동기 OPEN 상태(Actuator)
Fig. 13. Actuator is OPEN(Actuator)

위 그림에서 확인되는 바와 같이 OPEN 상태에서는 OLS와 CLS의 스위치가 인가되지 않는다.

구동기가 OPEN상태에서 지속적으로 OPEN이 되면, 최대 개방상태가 되고, 구동기의 OLS 스위치가 눌리게 된다. 그러면 구동기 Contoller는 OLS메시지를 전송하게 되고 Control Server에서는 이 메시지를 확인할 수 있다.

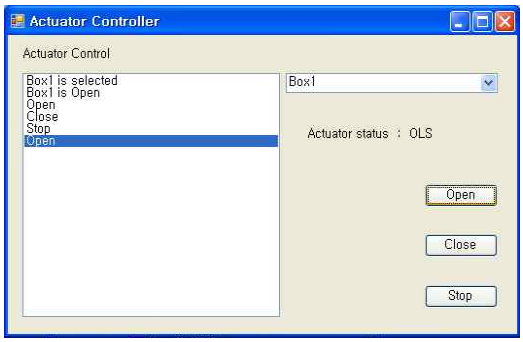

그림 14. 구동기 OLS 상태(Control Server)
Fig. 14. Actuator is OLS(Control Server)

그림 15. 구동기 OLS 상태(Actuator)
Fig. 15. Actuator is OLS(Actuator)

반대로 OPEN상태에서 지속적으로 CLOSE를 하게 되면, 폐쇄 상태가 되며 CLS 스위치가 눌리게 된다. 그러면 구동기 Controller는 CLS 메시지를 전송하게 되며, Control Server에서는 CLS 메시지를 확인할 수 있다.

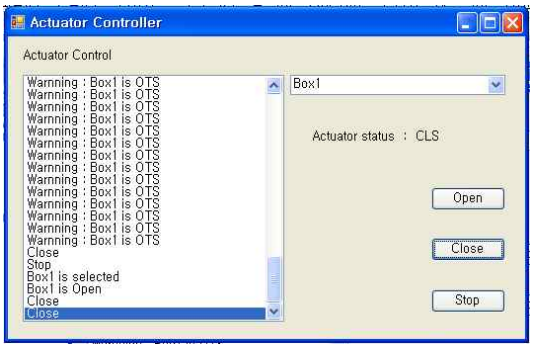

그림 16. 구동기 CLS 상태(Control Server)
Fig. 16. Actuator is CLS(Control Server)

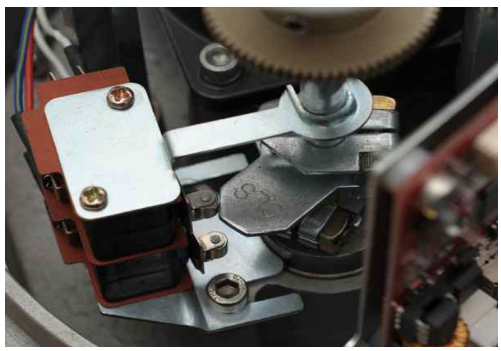

그림 17. 구동기 CLS 상태(Actuator)
Fig. 17. Actuator is CLS(Actuator)

만약 OPEN이나 CLOSE 도중 외부의 방해로 더는 구동기의 모터가 동작하지 않을 때에는 구동기의 OTS 혹은 CTS 스위치가 눌리게 되고, 이럴 때에는 각 상황에 맞는 메시지가 Control Server로 전송된다.

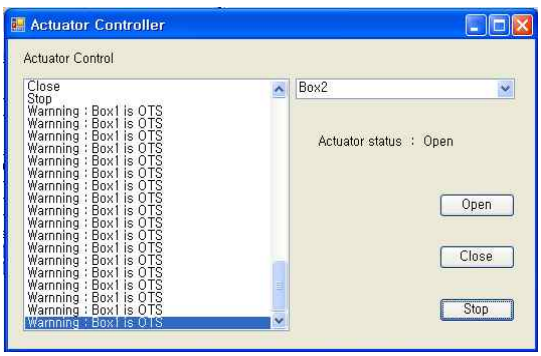

그림 18. 구동기 OTS 상태(Control Server)
Fig. 18. Actuator is OTS(Control Server)

Ⅴ. 결 론

지금까지 현재 필드버스시스템에서 주목받고 있는 CAN과 CANopen 프로토콜을 이용하여 구동기를 제어하는 기술을 구현하였다. Control Server는 BECKOFF사에서 제공하는 보드를 사용하고, 구동기 Controller 부분은 dsPIC30F5011을 사용하였다. 구동기의 동작을 제어하는 간단한 동작을 사용하였지만, 최대 64개의 서로 다른 신호를 전송할 수 있는 CANopen의 기능을 살린다면 공장 자동화나, 선박과 차량 제어, 로봇제어까지 광범위한 영역에서 활용 가능하다.

이 네트워크 시스템의 장점으로는, 각 구동기에서 발생하는 에러신호를 Control Server에서 실시간으로 확인할 수 있고, 구동기가 정상동작을 하지 않아도 네트워크에 오류가 발생하지 않는다. 그러므로 유지/보수를 하면서도 다른 구동기에 영향을 미치지 않아 피해를 최소화할 수 있다. 또한, 새로운 구동기를 추가할 때, 새로운 노드의 설계가 필요치 않아, 확장성 또한 쉽다. [8]

반면에 OD의 내용이 상당히 방대하여 초기 구성에 많은 어려움이 있었으며, 비교적 최근에 발표된 프로토콜이라 많은 레퍼런스가 존재하지 않아 개발에 어려움이 많은 것은 단점으로 뽑힌다. 많은 장점을 가지고 있는 프로토콜인 만큼 국내에서도 많은 연구가 필요하다.

지금까지 간단한 동작을 하는 CANopen 프로토콜에 대해 알아보았다. 프로토콜의 성능을 모두 활용한다면, 각종 필드버스 현장에서 유용하게 이용할 수 있을 전망이다.

참 고 문 헌

[1] 박진원, “통합 필드버스 게이트웨이 설계”, *한국정보과학회 가을 학술발표논문집,* 2004

[2] 김주찬, “엘리베이터 제어를 위한 CAN-기반 실시간 통신 시스템의 개발 ”, *건국대학교 대학원 석사학위논문,* 2006

[3] Robert Bosch Gmbh., "CAN Specification Ver. 2.0", *Robert Bosch Gmbh.*, 1991

[4] H. Boterenbrood, CANopen high-level protocol for CAN-bus NIKHEF, Amsterdam March 20, 2000

[5] CAN-in-Automation, CANopen, CAL-based Communication Profile for Industrial System.

[6] 강민구, “CANopen PDO 서비스를 이용한 프레임 패킹 매커니즘”, *한국 자동차 공학회 창립 30주년 기념 학술대회 논문집,* 2008

[7] 최웅, “DeviceNet과 CAN의 게이트웨이를 이용한 전동기 구동 시스템에 관한 연구”, *울산대학교 대학원 석사학위 논문,* 2006

[8] 홍승호, “생산자동화를 위한 통신망 : 필드버스의 개요” *한국정밀공학회지* 1994. 10.

이 명 의 (李明儀)

1985년 2월 : 인하대학교 전기공학(공학사)
1987년 2월 : 인하대학원 기기 및 제어 (공학석사)
1991년 8월 : 인하대학원 기기 및 제어 (공학박사)
1995년 8월 : 현대전자 선임연구원
2004년 1월 ~ 2005년 1월 : U.C.Berkeley 객원교수
1995년 9월 ~ 현재 : 한국기술교육대학교 교수
관심분야 : 제어 계측, 시스템 소프트웨어, 위성통신 시스템

신 근 수 (申劤秀)

2005년 8월: 한국기술교육대학교 정보통신공학과 (공학사)

2007년 8월~현재 : 한국기술교육대학교 전기전자공학과 (공학석사과정)
관심분야 : Embedded System.

양 성 현 (梁 城 鉉)

1985년 2월 : 광운대학교 전기공학과 (공학사)
1988년 2월 : 광운대학교 대학원 제어공학 (공학석사)
1993년 2월 : 광운대학교 대학원 제어공학 (공학박사)
1991년 3월 ~ 현재 : 광운대학교 전자정보공과대학 교수
1996년 8월 ~ 1998년 2월 : Boston Univ. Research Scientist
2005년 8월 ~ 현재 : 광운대학교 유비쿼터스 홈 네트워크 센터 센터장
1990년 1월 ~ 현재 : 한국통신학회 정회원
2007년 5월 ~ 2008년 4월 : 홈 네트워크 산업협회 홈네트워크 시장 활성화 분과 위원장
관심분야 : 홈 네트워크, 정보통신