

# 실시간 플러그 앤 플레이 가능한 CAN 통신 시스템의 설계 및 구현

김승희\*, 황광일\*

\*인천대학교 임베디드시스템공학과

e-mail : hkwangil@inu.ac.kr

## Design and Implementation of CAN communication System capable of Supporting Real-Time Plug and Play

Sungheo Kim\*, Kwang-il Hwang\*<sup>1</sup>

\*Dept. of Embedded Systems Engineering, Incheon National University

### 요약

오늘날 차량은 고성능화 및 이용자의 편의를 증진 시키기 위하여 다양한 전자장비가 탑재 되고 있다. 이러한 전자장비들을 효율적으로 제어하기 위하여 신뢰성이 입증된 CAN(controller Area Network)이 보편적으로 적용된다. 또한 수신만 가능한 노드(node)는 기존시스템에 추가가 가능하지만 송·수신을 할 수 있는 노드를 추가 하는 경우에는 전체 시스템을 변경 시스템을 변경해야 하는 경우가 발생 할 수 있다. 이러한 특징을 해결하기 위하여 본 논문에서는 CAN 코디네이터(Coordinator)라는 새로운 컴포넌트를 제안하고 이를 기반으로 긴급메시지에 대한 전송지연 개선 및 플러그 앤 플레이 기능을 가능케하는 새로운 CAN 프레임워크를 제안한다.

### 1. 서론

편리한 운송을 목적으로 등장한 자동차는 최신의 전기, 전자, 정보 통신 기술이 융합되어 운전자에게 보다 높은 안전과 편의를 제공하며, 운송수단 만이 목적이 아닌 사람이 생활하는 공간으로 변화되고 있으며, 차량관련 기술 수준이 높아짐에 따라 전자제어장비들의 종류도 급속도로 증가하게 되었다. 특히 완성차 업체에서 자동차 제작 비용 중에 전자제어장비가 차지하는 비중은 점점 늘어나는 추세이고, 전문가들 또한 향후 혁신적인 기술의 80%는 전자기술에 관련 될 것이라고 예측한다[1]. 이렇게 다양한 전자제어장비들을 제어하기 위해서는 배선이 증가하게 되는데, 이러한 배선의 증가는 차량 설계 시 어려움이 있고, 차량의 중량을 증가시켜 성능 및 연비가 저하되며, 제조 비용 또한 증가하게 된다[2]. 이에 전자제어장비들 간의 유기적인 제어를 가능하게 해주며, 여러 문제를 해결해 줄 수 있는 네트워크 기반의 통신시스템이 필요하게 되었고, 이러한 요청에 의해 1980년 대 독일 보쉬(Bosch)사에 의해 CAN(Controller Area Network) 시스템이 개발되었다[3]. 이는 보다 정밀한 제어가 가능하며, 차량의 배선 감소로 차량의 경량화를 실현하여 차량의 성능 향상 및 연비 개선 등의 여러 가지 이점을 갖게 된다. 하지만 이러한 네트워크

시스템은 무엇보다 신뢰성 보장에 관한 것이 중요한 이슈가 되는데, 특히 네트워크에서의 신뢰성은 연결되어 있는 디바이스들이 오류 없이 제어 될 수 있도록 적절한 타이밍에 데이터를 지연 없이 송신 및 수신 할 수 있도록 하는 것이 중요하다.

CAN 의 메시지 송신에 대한 타이밍 오류가 발생 할 수 있는 부분은 버스상에서 메시지가 충돌하게 되고, 그 충돌에 대한 중재과정에서 보내고자 하는 메시지의 식별자 보다 우선순위가 높은 메시지가 많을 경우 전송지연이 발생, 원하는 시간 내에 메시지를 전송하지 못하거나 전송기회를 놓치게 되는 문제점이 발생 할 수 있다[4].

이에 본 논문에서는 CAN 에 연결되어 있는 각 노드(Node)들의 정보를 통합 관리하는 신규 모듈인 CAN 코디네이터(Coordinator)를 추가하여, 전송지연 부분에 대해 이중 ID 배치를 통한 개선방안과 추가적으로 기존에 구성되어 있는 CAN 시스템에 신규 노드를 추가할 수 있는 플러그 앤 플레이(Plug and Play)기능을 제안하고자 한다.

### 2. CAN(Controller Area Network)

CAN 프로토콜은 ISO11898(고속전송)과 저속 전송을 위한 ISO11519-2(저속전송)에 정의되어 있는 국제 표준으로서 초기에는 자동차에 적용하기 위해 고안된

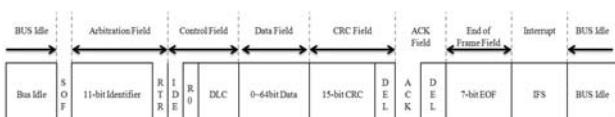
<sup>1</sup>본 연구는 2014 산학연공동기술개발지원사업에 의해 지원되었음.(과제번호: C02496350100425923, 과제명: Multi-Functional 초저전력 IoT Connectivity (Plug and Connect) 시스템 개발)

통신방식이며, 다른 통신 프로토콜에 비해 가격대비 성능이 우수할 뿐 아니라, 외부 전자파나 노이즈에 강해 최근에는 다양한 산업분야에 꽤 넓게 적용되고 있다[5].

CAN의 특징은 ISO 규격에 따라 ISO11898-2(High-speed : 50kbps에서 1Mbps)와 ISO 11898-9-3(Low-speed : 50kbps에서 125kbps)로 구분이 되고, 비파괴 비트별 중재(Non-Destructive Bitwise Arbitration), 브로드캐스팅(Broadcasting) 등이 있다. CAN 프로토콜은 OSI 7 Layer를 기반으로 구동이 되며, OSI 7 Layer 중 하위 두 계층인 데이터 링크 계층과 물리 계층을 사용하고 있다.

본 논문에서는 제안하는 방식의 구현을 위해서는 (그림 2-1)에서 확인할 수 있듯이 CAN의 프레임의 7개의 필드로 구성되어 있고, 그 중에서도 중재필드(Arbitration)가 중요한데, CAN2.0A와 CAN2.0B의 가장 큰 차이점이기도 하다. CAN2.0A는 11비트로 구성되어 있고, CAN2.0B는 기본 ID 11비트와 확장형 ID 18비트를 포함한 총 29비트로 구성이 되어 있다. 여기에서 CAN2.0B는 기본 ID 11비트를 갖고 있어 CAN 2.0A의 중재 필드 11비트와 호환이 가능하다.

CAN 2.0A(Standard Format)



CAN2.0B(Extended Format)



(그림 2-1) CAN의 데이터 프레임 구조

또한 CAN 코디네이터와 노드의 통신에서 데이터의 내용을 전달하게 될 데이터 필드는 최대 64비트의 데이터를 전송할 수 있다[6].

### 3. 관련 연구

#### 3.1. CAN의 전송지연 개선에 대한 기존 연구

CAN에서 여러 이벤트가 동시에 발생하여 각각의 전자제어모듈이 한 번에 메시지를 CAN BUS로 전송하게 되면, 충돌하게 되고, 이 충돌을 중재하는 과정에서 메시지 식별자 우선순위 전송 특성에 의하여 전송되어야 하는 메시지가 있음에도 다른 메시지의 식별자에 우선순위에서 밀려, 전송주기 내에 전송을 완료하지 못하는 문제점이 발생 할 수 있다. 이러한 문제들을 개선하기 위해 CAN에서 발생하는 전송지연과 관련된 여러 가지 연구가 진행되었고, 기존의 연구들을 보면 메시지들이 전송되는 순서를 적절하게 스케줄링하여 전송지연이 되는 문제를 개선하고자 하였다. 전송지연이 되는 문제를 개선하고자 하는 스케줄링 방법에는 정적(Static) 또는 동적(Dynamic) 스케줄링, 선점(Preemptive) 또는 비선점(Non-preemptive) 스케줄링으로 구분된다[7].

정적 스케줄링이라는 것은 기존 CAN 시스템에 많이 적용되어 있는 스케줄링 방법으로, 네트워크를 구

축하는 초기에 메시지의 중요도에 따라 우선 순위를 부여하고 그에 따른 ID를 배치한다. 이렇게 배치된 ID에 따라 메시지가 전송되는 것을 의미한다. 이에 비하여 동적 스케줄링은 시스템이 정상동작을 하고 있는 가운데 상황 또는 필요에 따라 전송되는 메시지의 ID가 달라지는 것 즉 이벤트가 발생하는 것에 따라 우선순위의 변화가 있다는 것을 의미한다.

선점 스케줄링은 그 의미 그대로 버스 상에 임의의 메시지가 버스를 선점하여 데이터 메시지를 전송하고 있다면 그 데이터 메시지가 전송이 완료 될 때까지 대기 하는 것을 의미하고, 비선점 스케줄링은 임의의 메시지가 버스 상에 있어도 우선순위에 의해 메시지가 전송 될 수 있게 하는 것을 말한다.

#### 3.2. 스케줄링(Scheduling) 기법

정적인 스케줄링 기법의 대표적인 DMS는 RMS(Rate Monotonic Scheduling)의 기법을 CAN 시스템에 적합한 형태로 변형 시킨 것으로서 RMS는 주기의 길이로 우선순위를 결정하는데, 짧은 주기를 갖는 메시지가 긴 주기를 갖는 메시지보다 우선순위에서 앞서게 된다. 이러한 방식에 의하여 부여된 우선순위대로 메시지들은 동기화된 시간 선상에 차례대로 배치되어 전송된다. RMS에서는 일반적으로 전송주기를 데드라인(Deadline)과 동일하다고 가정하지만 항상 그런 것은 아니다. 이에 주기가 아닌 메시지의 전송이 완료되었을 때까지의 데드라인 시간이 짧은 메시지에 우선순위를 높게 부여하도록 변화 시킨 것이 DMS이다. 이러한 고정된 스케줄링 기법들은 구현이 간단하지만 이것을 위해서는 노드들 간의 동기화가 필요한 문제가 있고, 다량의 메시지가 충돌하여 높은 트래픽이 발생하는 경우 우선순위가 상대적으로 낮은 데이터 메시지들은 전송지연 또는 전송이 불가능할 수도 있다.

동적인 스케줄링 기법은 EDF(Earliest Deadline First)이 대표적인데, 이 기법에서는 데드라인이 가까운 메시지를 먼저 전송하게 된다. 이러한 방법은 데이터 메시지의 우선순위를 정하는 것이 아니라 한 번의 데이터 메시지 전송이 끝나게 되면 바로 모든 노드들은 자신이 가지고 있는 메시지들의 데드라인을 확인하여 시간이 적개 남은 메시지를 먼저 전송하게 되는 방법이다. 동적인 EDF 기법은 정적인 DMS에서 발생할 수 있는 문제를 개선할 수 있지만 통신을 하기 위해 버스를 차지하는 과정에서 매번 과도한 오버헤드가 발생하고, EDF를 수행하기 위해서는 모든 노드들은 동기화되어야 한다. 또한 네트워크에 연결되어 있는 모든 노드들은 자신이 전송하고자 하는 메시지가 데드라인에 얼마나 가까이 있는 지와 다른 메시지들과 얼마나 차이가 나는지를 계속해서 모니터링 해야 하기 때문에 실질적으로 구현하기도 힘들고 그것을 처리하는 프로세서에도 많은 부하를 주게 된다[8].

### 4. 제안시스템

#### 4.1. 제안시스템의 개요

본 장에서는 CAN 코디네이터라는 신규 디바이스

타입을 새롭게 정의하고 이를 기반으로 하여 기존의 CAN 시스템의 취약점인 전송지연에 대한 개선책과 송, 수신이 가능한 신규 노드를 적용 시 기존 노드들의 업그레이드 없이 바로 시스템에서 구동 가능한 플러그 앤 플레이 기능을 제안 한다. 또한 기존의 CAN 시스템에 영향을 미치지 않도록 하기 위하여 등록모드라는 개념을 새롭게 도입하였다. 제안된 시스템은 등록모드에서만 동작을 하며 등록모드 종료 후에는 CAN 코디네이터는 기능이 정지 되고, 다른 CAN 디바이스들은 기존의 CAN 시스템과 동일하게 유지시켜 안정성까지 확보되는 시스템을 제안하고자 한다.

#### 4.2. 제안시스템의 정의

CAN 코디네이터는 CAN BUS 에 연결되어 있는 다양한 제어 모듈의 정보를 갖고 있으며 그러한 정보를 바탕으로 CAN 시스템에 변화(신규 모듈의 진입 등) 가 생겨도 다른 제어 모듈로 정보를 송,수신하며 변화에 대해 유연하게 대응해 나가게 된다.

CAN 코디네이터가 제어 모듈과 메시지를 송,수신하기 위해서는 먼저 ID 영역을 구분해 줄 필요가 있다. 본 논문은 CAN2.0A 를 기준으로 하기 때문에 11비트의 식별자를 보유하고, 이로 인하여 식별자의 범위는 000-7FF 까지가 된다. 이러한 식별자를 <표 4-1>에서 볼 수 있듯이 긴급메시지(전송지연을 받으면 안되는 이벤트 성 메시지)를 배치하기 위한 구간, 일반적인 통신을 하는 일반메시지 구간, 등록모드에서 CAN 코디네이터와 노드들 간의 데이터 메시지를 송,수신하기 위한 구간으로 구분 한다.

<표 4-1>데이터 구분에 따른 ID 영역

ID	사용구간
000-0FF	긴급메시지 데이터 구간
100-6FF	일반메시지 데이터 구간
700-7FF	등록모드 데이터 구간

보다 많은 CAN ID 가 필요한 경우 확장 형인 CAN 2.0B 를 사용하면 29 비트의 식별자를 갖고 있기 때문에 더 큰 범위로 시스템을 구성할 수 있다.

본 논문이 제안하는 시스템의 핵심적인 역할을 하는 CAN 코디네이터가 동작을 하는 타이밍은 등록모드로서 전체 시스템이 처음 파워-온(Power-On)이 되면 등록모드로 진입할 준비를 하게 된다. CAN 코디네이터는 등록모드 데이터 구간 ID 700 번을 갖고 각각의 노드가 연결되어 있는 CAN BUS에 등록모드 설정 시작 메시지를 보낸다. 설정 시작 메시지를 수신한 노드들은 노드 번호와 그 노드가 갖고 있는 기능의 수를 701-7FF 사이의 ID 를 갖고 첫번째 응답 메시지에 담아 CAN 코디네이터에게 전송한다.

노드의 첫번째 메시지를 수신한 CAN 코디네이터는 현 CAN BUS 에 연결되어 있는 노드의 수와 해당 노드가 가지고 있는 기능이 몇 개 인지를 연산 한다.

CAN 코디네이터는 연산을 완료하고, 각각의 노드들이 갖고 있는 기능에 대한 제어 및 상태 정보를 알려 주는 데이터를 수신 받을 수 있도록 각각의 노드에게 전송권한 메시지를 순차적으로 보낸다. 전송

권한 메시지는 1 번 노드가 전송권한을 받아 다 전송 하면 CAN 코디네이터는 다시 2 번 노드에 전송권한을 부여 하는 방식으로 각각의 노드에 순차적으로 전송 권한을 부여한다. 각각의 노드 들로부터 기능별 데이터 정보가 들어 있는 두 번째 메시지를 받은 CAN 코디네이터는 정보를 분석 하여 저장한다. 이렇게 분석 된 정보 중 데이터에 있는 기능별 ID 배치 요청에 맞게 배치 하게 된다. 기능별 ID 배치가 완료되면 긴급 메시지가 있는지 확인하고 긴급메시지 요청이 있으면 000~OFF 사이의 비어 있는 ID 를 CAN 코디네이터가 긴급 정도에 따라 선택하여 100~6FF 사이의 기본 ID 와는 별도로 추가 배치한다. 즉, 평상시에는 100~6FF 사이의 기본 ID 로 통신을 하다가 이벤트가 발생하여 긴급하게 보내야 하는 경우 000~OFF 사이에 배치된 ID 를 갖고 통신을 하게 된다.

이렇게 데이터의 ID 배치 정의가 끝나면 메시지 기능에 대한 동작 정보 및 방법, 기본 ID, 긴급 ID 등을 포함하여 노드가 설정을 할 수 있도록 설정메시지를 보내게 된다. 이때 설정메시지는 CAN 의 특징인 브로드캐스팅으로 인하여 CAN BUS 에 연결되어 있는 모든 노드 들이 설정 메시지를 순차적으로 들을 수 있게 된다. 마지막으로 설정 메시지를 모두 전송하면 CAN 코디네이터는 등록모드 종료 메시지를 보내고, 설정을 완료한 노드들도 설정 완료 및 등록 모드 종료 메시지를 CAN 코디네이터에게 보내게 되면 등록 모드는 종료 되고 시스템은 CAN 정상 운행 모드로 들어 간다.

#### 5. 실험 및 성능평가

본 논문에서 제안하는 CAN 시스템의 전송지연 개선에 대한 성능 평가 및 플러그 앤 플레이 기능을 구현하기 위해 (그림 5-1)처럼 CAN 코디네이터 및 CAN 노드로 사용될 모듈 7 개를 제작하였다.



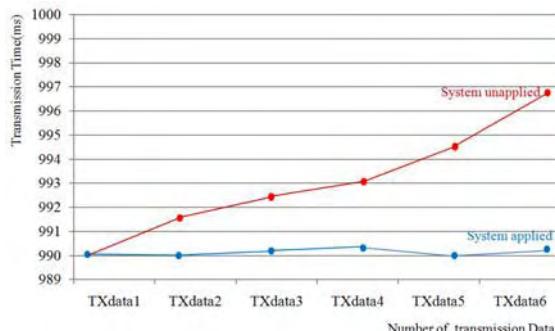
(그림 5-1)CAN 테스트 모듈

실험에 대한 결과를 확인하기 위해서는 Intrepid Control System 사의 CAN 계측 장비를 사용하였다. CAN 계측장비는 CAN BUS 상에 메시지가 통신을 하고 있다면 그 메시지를 모니터링 할 수 있다.

먼저 전송지연 개선 부분을 실험하기 위해 각각 노드에 CAN2.0A 를 사용하였고, 식별자를 데이터 구분에 의한 ID 영역으로 나누어 100-6FF 사이의 ID 를 부여하였으며, 부여된 ID 를 갖고 100kbps 의 속도로 1000ms 에 한번씩 동시에 CAN 메시지를 전송하였다. 이로 인하여 여러 개의 메시지가 충돌하게 되고 이 충돌을 중재하는 과정에서 우선순위가 밀리는 메시지는 바로 앞 우선순위 메시지에 비하여 약 1.2-1.7ms 의

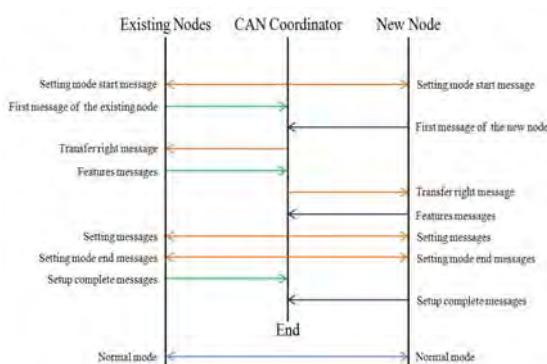
전송지연이 발생하였다. 실험상 6 개의 메시지가 충돌이 발생 했을 때 최고 후 순위 메시지는 약 6ms 이상의 전송지연이 발생하는 결과를 얻을 수 있었다.

이에 긴급메시지에 대한 전송지연 개선 방법인 000-0FF 사이의 ID 추가로 일반메시지 데이터 구간에서 우선 순위가 가장 낮은 메시지에 적용하면, 긴급 메시지 발생시 최우선 순위로 올라가기 때문에 긴급 메시지에 대한 부분에서는 전송지연이 발생하지 않는 것을 (그림 5-1)에서 확인 할 수 있었다.



(그림 5-1) 긴급메시지 적용과 미적용 비교

시스템이 적용되지 않은 경우를 보면 Y축의 TX데이터 수가 증가함에 따라 우선순위가 낮은 데이터는 전송지연 증가로 인하여 전송시간이 증가됨을 볼 수 있다. 이것과 비교하여 긴급메시지 전송지연 개선 시스템이 적용된 경우에는 버스 상에 한 번에 전송되는 TX데이터 수가 증가를 해도 긴급하게 전송되어야 하는 메시지는 우선권이 확보되어 전송지연 없이 데이터가 전송되는 것을 확인 할 수 있었다.



(그림 5-2) Plug and Play를 위한 통신과정

다음으로 플러그 앤 플레이 기능 구현에 대한 실험은 CAN코디네이터를 제외한 노드 4개를 사용하였으며, 기존 시스템에 신규모듈이 추가 되었을 때를 가정하여 실험하였고 알고리즘에 맞게 동작하는가는 CAN계측 장비를 이용하였다. CAN코디네이터가 제안된 시스템에 맞게 각각의 노드들을 관리하여 플러그 앤 플레이 기능이 적용되는 것에 대하여 측정 장비를 통해 코디네이터와 노드 간의 실질적인 통신 내용을 확인하였고 그 과정을 (그림 5-2)과 같이 표현하였다.

## 6. 결론

먼저 전송지연 개선 부분은 동시에 여러 개의 노드에서 데이터 메시지를 전송하게 되면, 메시지 우선순위에 따라 전송 지연 발생하는 것을 볼 수 있었고, 이에 이벤트 발생 시 긴급하게 전송되어야 하는 데이터 메시지가 있는 경우 CAN 코디네이터에서 추가적으로 ID 를 배치하여 다른 메시지 보다 우선권을 주어 긴급한 데이터 메시지를 전송지연 없이 전달 할 수 있도록 하였다. 하지만 모든 노드들이 동일한 시간에 데이터 메시지를 계속 전송하고 있는 실험환경에서는 긴급메시지 전송으로 인하여 다른 메시지들은 우선권에서 밀려나는 현상 즉, 전송지연이 다른 메시지로 전가되는 현상이 일어나는 것을 확인 할 수 있었다.

물론 실제 적용된 CAN 시스템의 경우에는 모든 데이터 송신이 같은 시간에 일어나는 것이 아니라 필요에 의하여 시간차를 두기 때문에 긴급 메시지에 대해서 우선권을 부여해도 모든 메시지가 충돌되어 전송지연이 전가되는 상황이 나타날 확률은 매우 낮다.

다음으로 플러그 앤 플레이(Plug and Play) 기능에 관해서는 데이터의 기능, 동작방법, 동작을 위한 데이터 메시지 위치 등의 정보를 CAN 코디네이터가 모두 관리하기 때문에 기존의 시스템에 새로운 송, 수신을 하는 모듈이 진입을 해도 CAN 코디네이터가 가지고 있는 정보 등을 알려 줌으로써 신규 모듈은 기존에 존재하는 데이터에 관한 정보를 인지하고, 기존의 노드들과 유기적으로 운영되는 것을 실제로 구현하여 결과를 도출하였다.

이렇듯 기존의 CAN 시스템에 CAN 코디네이터를 추가하여 CAN 시스템을 보완 사항을 개선하여, 다방면으로 활용할 수 있는 여건이 마련되었으며, 향후 다양한 연구를 통하여 CAN 코디네이터를 가지고 있는 CAN 시스템이 더욱 진화 될 수 있을 것으로 기대된다.

## 참고문헌

- [1] D. Marsh, "Network Protocols Compete for highway supremacy," EDN Europe, June 2003
- [2] 이석, 김만호, 이경창, "차량용 네트워크 기술 연구 동향," 한국정밀공학회 논문집, 제 23 권, 제 9 호, 2006
- [3] Bosch. CAN specification version 2.0. Published by Bosch GmbH, September 1991
- [4] K. N. Ha, M. H. Kim, K. C. Lee, S. Lee "Performance Evaluation of Network Protocol for Automated Transfer Crane System," Journal of Control, Automation, and Systems Engineering Vol. 11, No. 8, August, 2005
- [5] 박장식·윤병우, 『AT90CAN128 을 이용한 CAN통신 실무』, 흥룡과학출판사, 2009
- [6] William E. Seitz, "Controller Area Network in Embedded Machine Control," Controlin Automation North America, 2004
- [7] J. P. Lehoczky and L. Sha, "Performance of Real-Time Bus Scheduling Algorithm," ACM Performance evaluation Review, 1986
- [8] G. C. Buttazzo, "Rate Monotonic As, EDF:Judgment Day," Journal of Real-Time System, 29, 2005