논문 14-39B-02-08

한국통신학회논문지(J-KICS) '14-02 Vol.39B No.02
http://dx.doi.org/10.7840/kics.2014.39B.2.123

CAN 네트워크의 시간동기를 위한 IEEE1588 구현

박 성 원◆, 김 인 성*, 이 동 익°

Implementation of IEEE1588 for Clock Synchronization

Sung-won Park◆, In-sung Kim*, Dongik Lee°

요 약

본 논문에서는 CAN(Controller Area Network)의 시간동기를 위한 IEEE1588 알고리즘의 구현에 관한 연구결과를 제시한다. 시간동기는 네트워크 기반 임베디드 시스템의 안정성, 효율, 신뢰성 개선 측면에서 매우 중요하다. 최근 전용 칩을 이용하는 IEEE1588 표준이 Ethernet 기반 임베디드 시스템의 시간동기에 폭넓게 적용되고 있다. IEEE1588과 같은 표준화된 시간동기 기법은 기존의 'in-house' 시간동기 기법에 비해 많은 장점들을 제공하지만, CAN을 위한 IEEE1588 전용 칩은 현재까지 상용화된 제품을 찾아보기 어렵다. 본 논문에서는 전용 칩을 사용하지 않고 소프트웨어와 CAN 메시지만을 이용하여 IEEE1588 알고리즘을 구현한다. 제안된 방법의 효용성을 확인하기 위해 간단한 모델을 이용하여 추정한 동기정밀도와 실험용 네트워크를 통해 측정한 동기정밀도를 비교분석 한다.

Key Words : Controller Area Network, Clock synchronization, IEEE1588

ABSTRACT

In this paper, an IEEE1588 based clock synchronization technique for CAN (Controller Area Network) is presented. Clock synchronization plays a key role to the success of a networked embedded system. Recently, the IEEE1588 algorithm making use of dedicated chipsets has been widely adopted for the synchronization of various industrial applications using Ethernet. However, there is no chipset available for CAN. This paper presents the implementation of IEEE1588 for CAN, which is implemented using only software and CAN packets without any dedicated chipset. The proposed approach is verified by the comparison between the estimated synchronization precision with a simple model and the measured precision with experimental setup.

Ⅰ. 서 론

1980년대 이후 마이크로프로세스 기술과 이를 이용한 산업용 네트워크 기술의 발전에 힘입어 다양한 분야에서 네트워크 기반 임베디드 시스템의 적용이 이루어지고 있다. 필드버스(fieldbus) 또는 데이터버스(databus)로 불리는 산업용 네트워크는 단순히 복잡한 배선 문제를 개선하는 것 뿐 아니라, 모듈화 설계, 유지보수 효율성, 시스템 설치 및 수정보완 편리성 등의 장점을 제공하므로, 최근에는 고도의 안전성과 신뢰도가 요구되는 경성실시간(hard real-time) 시스템에서도 많이 이용되고 있다. 1980년대 중반 독일의 Bosch 사에 의해 차량내 통신 용도로 개발된 CAN (Controller Area Network)[1]은 짧고 주기적인 메시지

※ 본 연구는 미래창조과학부 및 정보통신산업진흥원의 IT융합 고급인력과정 지원사업의 연구결과로 수행되었음 (NIPA-2013-H0401-13-1005)

◆ First Author : 경북대학교 전자공학부 고신뢰성 임베디드 제어 시스템 연구실, itisuptous@nate.com, 학생회원
° Corresponding Author : 경북대학교 전자공학부 고신뢰성 임베디드 제어 시스템 연구실, dilee@ee.knu.ac.kr, 정회원
* 경상공업고등학교 지능형로봇과, solitarykim@hanmail.net

논문번호 : KICS2014-02-035, 접수일자 : 2014년 2월 4일, 심사일자 : 2014년 2월 17일, 최종논문접수일자 : 2014년 2월 21일

의 전송에 적합한 대표적인 필드버스이다. CAN은 차량에 성공적으로 적용되어 안정성이 검증되었을 뿐 아니라, 사용이 간편하고, 전용 칩과 개발장비를 비롯해 다양한 지원 제품들이 개발됨에 따라 최근에는 자동차 산업 외에도 제트엔진, 풍력발전, 잠수함 등에 이르기까지 CAN의 적용분야가 꾸준히 확대되는 추세이다[2-5]. 하지만 최대 1Mbps의 데이터 전송률, 우선순위가 낮게 지정된 메시지의 과도한 전송지연, 메시지 충돌 발생시 전송지연이 랜덤하게 변하는 특성 등으로 인해 CAN을 제동장치나 조향장치 등 경성실시간 시스템에 직접 적용하기에는 어려움이 있다. 이러한 문제점을 극복하기 위한 대안으로써 시스템을 시간기반(time-triggered)으로 설계 및 구현하는 방안이 자주 이용된다[6-8]. 즉 네트워크에 연결된 노드들이 수행하는 태스크에 대해 그림 1과 같은 round-robin 방식의 스케줄을 적용함으로서 네트워크의 대역폭 사용효율을 높이고, 메시지 충돌이 발생하지 않도록 하여 우선순위와 무관하게 일정한 전송 특성을 얻을 수 있다.

그러나 시간기반 시스템 구현을 위해서는 높은 신뢰도와 정밀도를 가지는 시간동기(clock synchronization)의 확보가 반드시 필요하며[9], CAN 기반 시스템의 시간동기에 대한 연구도 다수 발표된 바 있다[10-13]. 하지만 기존 연구에서 제시한 방법들은 'in-house' 구현, 즉 해당 시스템을 위해 자체 개발한 소프트웨어에 의존하므로 여러 가지 문제점이 따른다. 무엇보다 동기 알고리즘의 신뢰성을 검증하기 어렵고, 소프트웨어 엔지니어의 역량에 따라 시간동기 성능의 편차가 발생할 수 있으며, 새로운 시스템 또는 네트워크 적용시 시간동기 소프트웨어를 재작성할 필요가 있다. 이러한 문제점은 표준화된 시간동기 기법을 적용하여 극복 가능하다. 대표적인 임베디드 네트워크용 시간동기 표준으로 IEEE1588[14]을 들 수 있다. PTP(Precision Time Protocol)로 불리우는 IEEE1588은 마스터-슬레이브 방식의 시간동기 기법이며, 마스터가 전송하는 Sync 메시지와 FollowUp 메시지를 이용하여 전송지연에 의한 동기오차를 보상한다. 멀티캐스트(multicast)

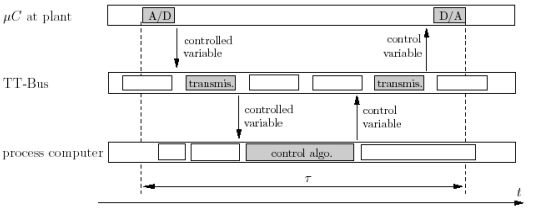


그림 1. Round-robin 스케줄을 적용한 시간기반 시스템
Fig. 1. Round-robin scheduling of time-triggered system.

방식의 네트워크에 모두 적용가능 하지만, 아직까지는 EtherCAT, TTEthernet 등 산업용 Ethernet에만 이용되고 있다. 일반적으로 IEEE1588의 적용을 위해서 전용 칩을 사용하는데 현재까지 상용화된 칩은 모두 Ethernet 용도로 개발되었기 때문이다. 최근에는 IEEE1588의 동기성능 개선 또는 마스터 고장 상황 인지 및 복구 기법 등에 관한 연구결과가 제시되고 있으나 이들 역시 Ethernet에 적용한 연구들이 대부분이다[15-21].

본 논문에서는 CAN 기반 시스템의 시간동기를 위해 IEEE1588을 소프트웨어로 구현하고, 실험용 네트워크를 구성하여 동기정밀도를 정량적으로 분석한다. 아울러 시간동기 정밀도 추정을 위한 모델을 도출하여, 실험을 통해 얻은 정밀도와 모델의 결과를 비교한다. 일반적으로 IEEE1588 전용 칩을 이용하여 Ethernet에 적용시 수 us 수준의 동기정밀도를 얻을 수 있으나, 본 논문에서는 소프트웨어 방식의 구현과 저사양 클럭 및 CAN의 낮은 전송속도를 고려하여 수백 us 수준의 동기정밀도를 얻는 것을 목표로 한다.

Ⅱ. 배 경

2.1 CAN 개요

CAN은 자동차에 탑재된 ECU(Electronic Control Unit) 사이의 데이터 공유를 위해 Bosch사에 의해 개발되고 ISO11898로 등록된 필드버스이다[1]. 매년 수천만 대의 자동차가 생산되어 매우 열악한 환경에서 장기간 운용될 뿐 아니라 가격 경쟁도 치열한 자동차 산업의 특성이 반영된 CAN은 높은 전송신뢰도와 낮은 가격의 장점을 동시에 제공한다. 그 결과 최근에는 자동차 산업 뿐 아니라 공장자동화, 항공기, 철도, 로봇, 선박해양, 수중무기체계 등 여러 분야에서 폭넓게 적용되고 있다. CAN은 이벤트 기반(event triggered) 네트워크의 하나로서, 다중마스터(multi master) 방식과 함께 11비트 또는 29비트의 식별자(identifier, ID)를 이용한 메시지 충돌 중재(arbitration) 기법을 적용하고 있다. 따라서 노드 또는 메시지의 추가/제거 등 시스템의 수정변경이 용이하다. 또한 CSMA/CD+AMP(carrier sense multiple access/collision detection with arbitration on message priority)로 불리우는 ID 기반의 메시지 충돌 중재 기법은 메시지의 중요도에 따라 전송 우선순위를 부여함으로서, 우선순위가 높은 메시지의 전송지연을 최소화할 수 있다.

2.2 CAN 네트워크 시간동기의 필요성

앞서 언급한 것처럼 CAN은 CSMA/CD+AMP를 이용하는 이벤트 기반 네트워크로서 많은 장점을 제공한다. 하지만, 최근 자동차의 기능과 편이성이 복잡하고 다양해짐에 따라 여러 가지 문제점에 직면하고 있다. 첫 째, 네트워크의 부하가 높아지면 우선순위가 낮은 메시지의 전송지연이 급속히 증가하므로(그림 2), 높은 전송신뢰도와 확정성이 요구되는 시스템에서는 대역폭의 30% 이상을 초과하여 사용하기가 어려운 실정이다[22]. CAN을 통한 메시지 전송 요구량이 증가하는 추세를 고려하면 이는 매우 큰 제약사항이다. 두 번째, 이벤트 기반 네트워크의 특성상 전송지연의 불규칙한 변화 즉 지터(jitter)가 발생하는데, 네트워크 부하가 커지고 메시지의 우선순위가 낮을수록 큰 값을 나타낸다. 지터는 시스템이 시변(time-varying) 특성을 갖게 함으로서 설계 및 구현에 큰 어려움으로 작용한다[23]. 이를 극복하기 위한 연구들이 많이 진행되었으나, 대부분 버퍼나 관측기 등을 이용하여 지연시간의 불규칙성을 최소화하는 방법에 주안점을 두고 있다[23-25]. 그런데 이러한 접근법은 과도한 계산량, 소프트웨어 복잡성, 모델링 오차 등으로 인해 CAN이 주로 적용되는 저비용 임베디드 시스템에는 적합하지 않다. 그 밖에 엄격한 데이터 일치성(data consistency) 및 동기화된 태스크 수행이 요구되는 경성실시간 시스템과 고장허용시스템(fault tolerant system) 구현에도 어려움이 따른다[26]. 이러한 제약조건들을 극복하기 위한 많은 노력들이 있었으나, 대부분은 클럭의 동기화를 전제로 하고 있다[26]. 요약하면, CAN 네트워크에 연결된 노드의 동기화된 클럭이 시스템과 네트워크의 신뢰성, 확정성, 효율성 개선에 공통적으로 요구된다고 볼 수 있다. 동기화된 클럭 기반의 차량용 네트워크로서 이미 TTP/C[27], FlexRay[28] 등이 개발되어 있으나, 이들은 전송속도, 비용, 전송 데이터 크기 등의 측면에서 CAN의 적용 분야와 상이하므로 본 논문에서는 고려하지 않는다.

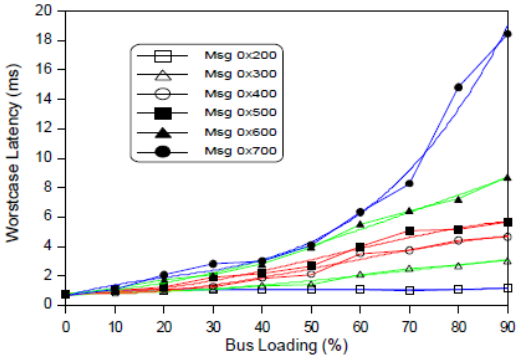


그림 2. 버스부하 및 우선순위에 따른 메시지 전송지연[22]
Fig. 2. Latency with respect to bus load and message priorities[22].

2.3 CAN을 위한 시간동기 기법

네트워크 기반 시스템을 구성하는 각 노드에 내장된 클럭의 특성은 온도, 습도, 압력 등 다양한 외부 환경에 의해 비선형적으로 변하며, 일반적으로 10-4~10-6sec/sec의 드리프트율(drift rate)을 갖는다[29]. 그 결과 매일 수 초씩 클럭오차가 누적될 수 있으며, 이러한 클럭오차는 다양하고 심각한 시스템 오류와 사고를 유발한다[30]. 시간동기(clock synchronization)란 네트워크에 연결된 각 노드의 클럭(local clock)을 이용하여 전체 시스템에서 공통으로 인식할 수 있는 전역 기준시간을 추정함으로서, 시스템에 포함된 임의의 두 클럭 사이의 동기 오차를 허용범위 이내로 유지하는 기술을 가리킨다[9]. 대규모 컴퓨터 시스템을 위한 시간동기 관련 연구는 20년 이상 진행되어 다양한 방법들이 개발되었다. 그러나 이들은 대부분 Ethernet으로 연결된 고성능 워크스테이션에 적용하기 위해 개발되었으므로 복잡한 연산과 알고리즘 구조, 대량의 메시지 전송 등을 사용한다. 따라서 이러한 방법들을 메모리, 연산속도, 클럭 사양, 통신 대역폭,동작환경 등에 있어서 큰 제약이 따르는 임베디드 시스템에 직접 적용하기는 어렵다. 반면 참고문헌[11-12]이 제시한 CAN 기반 임베디드 시스템을 위한 시간동기 기법은 단순한 마스터-슬레이브 구조를 가지는 소프트웨어 방식을 적용하여 상기 제약사항을 극복하였으며, 메시지 전송지연으로 인한 동기정밀도 저하를 보상하기 위해 후처리(a posteriori) 기법을 이용하고 있다. 하지만 기존 연구에서 제시한 결과들은 표준화되지 않은 'in-house' 방법들로서, 이를 실제 시스템에 적용 시 여러 가지 문제점을 유발한다. 특히 정형화 기법(formal method) 등을 이용하여 알고리즘의 신뢰성을 검증하는 것이 어렵거나 많은 비용이 소요된다[30]. 또한 자체적으로 작성한 소프트웨어에 의존하므로 소프트웨어 엔지니어의 역량에 따라 오류가 발생하거나 시간동기 성능이 예상과 다른 결과를 나타낼 가능성이 있다. 아울러 새로운 시스템이나 네트워크에 적용 시 소프트웨어를 재작성하는 번거로움도 발생한다. 이러한 문제점은 표준화된 시간동기 기법을 적용함으로써 극복 가능하다.

Ⅲ. CAN을 위한 IEEE1588 알고리즘 구현

3.1 IEEE1588 개요

시간동기는 네트워크 기반 시스템에서 공통적으로 요구되는 기술이므로 시간동기 기법의 표준화 관련 연구가 진행된 바 있다. 표준화된 기술을 적용할 경우, 기존의 'in-house' 방식에 비해 개발기간 및 비용 절감, 성능 및 신뢰성 검증 편이성 등 여러 가지 장점을 기대할 수 있다. 대표적인 시간동기 표준은 Network Time Protocol(NTP)[31]과 Precision Time Protocol (PTP)[14]이 있다. NTP는 주로 LAN과 인터넷 환경에서 적용되며 msec 수준의 동기정밀도를 제공한다. 반면 IEEE1588로 알려진 PTP는 네트워크 기반 임베디드 시스템의 고정밀/고신뢰성 시간동기 기능을 구현하기 위한 국제표준으로서 전용 칩을 사용하여 usec 수준의 동기정밀도를 제공한다. 2008년에 발표된 버전2는 Ethernet을 비롯하여 멀티캐스트 방식의 다양한 네트워크 환경에 적용할 수 있다. 하지만 현재까지 상용화된 전용 칩은 모두 Ethernet을 위한 것이며, 본 논문에서 다루는 CAN에 적용 가능한 전용 칩은 찾아보기 어렵다. 따라서 본 논문에서는 IEEE1588 알고리즘을 CAN 네트워크에 적용할 수 있도록 소프트웨어로 구현한 결과를 소개한다.

3.2 IEEE1588의 동작 원리

IEEE1588은 로컬클럭을 'Boundary Clock'에 동기화시키는 마스터-슬레이브 구조를 가진다. 마스터와 슬레이브 노드의 결정은 BMC(Best Master Clock) 알고리즘을 통하여 진행되는데, Boundary Clock 가운데 최적의 클럭이 마스터가 되고 다른 모든 클럭은 슬레이브로 동작한다. 마스터 클럭의 고장에 대비하여 다수의 Boundary Clock을 예비 마스터로 지정할 수 있다. IEEE1588의 실행에 사용되는 메시지는 이벤트 메시지와 일반 메시지로 구분된다. 이벤트 메시지는 실제 동기화에 사용되는 메시지이고, 일반 메시지는 초기화 및 지연 정보 갱신 등의 보조적인 처리를 위해 사용된다. 또한 현재의 마스터 시간을 추정하여 슬레이브의 로컬시간을 보정하는 과정은 타임스탬프(timestamp)라는 클럭 정보의 교환을 통해 이루어진다. 그림 3은 메시지 교환을 통해 IEEE1588이 시간동기를 수행하는 과정을 나타내며, 시간차(offset)를 추정하는 과정과 전송지연을 추정하는 과정으로 나눌 수 있다. 시간차는 마스터와 슬레이브 사이의 오차를 가리키며, 전송지연 추정은 동기 메시지의 송수신 과정에 포함된 지연값을 추정하여 보상함으로써 동기정

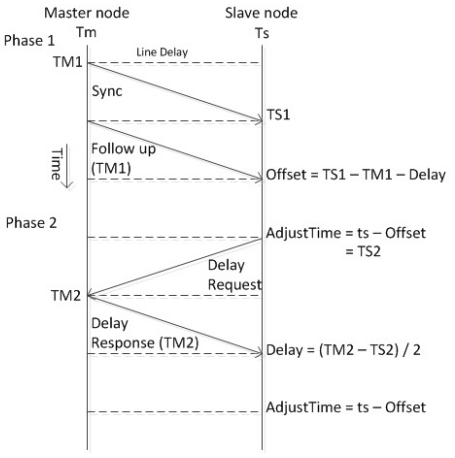


그림 3. IEEE1588의 메시지 교환[14]
Fig. 3. Message exchange in IEEE1588[14].

밀도를 개선하기 위한 과정이다.

먼저, 시간차를 추정하는 과정은 마스터 노드에서 슬레이브 노드로 'Sync' 메시지를 전송함으로서 시작된다.

마스터와 슬레이브는 각각 자신의 로컬클럭을 기준으로 마스터가 'Sync' 메시지를 전송하는 순간(TM1)과 슬레이브가 'Sync' 메시지를 수신하는 순간(TS1)에 타임스탬프를 측정한다. 이어서 마스터 노드가 자신의 타임스탬프 값(TM1)을 포함하는 'FollowUp' 메시지를 슬레이브로 전달한다. 슬레이브는 전송 받은 TM1값과 TS1값의 타임스탬프 차이를 이용하여 두 노드 간의 시간차를 계산하고, 이 값을 슬레이브 클럭에 반영하고 마스터와의 클럭오차를 보정한다. 시간차(offset) 측정을 위한 식은 다음과 같다[14].

$$Offset = TS1 - TM1 - Delay \quad (1)$$

전송지연을 추정하는 과정은 앞서 설명한 시간차 추정과정을 통해 클럭오차를 보정한 직후에 실행된다. 슬레이브가 'DelayRequest' 메시지를 마스터에게 전송하는 순간(TS2)과 마스터가 이 메시지를 수신하는 순간(TM2)에 각각의 로컬클럭을 기준으로 타임스탬프를 측정한다. 슬레이브가 TS2를 포함한 'DelayRequest' 메시지를 전송하면 마스터가 TM2를 포함한 'DelayResponse' 메시지를 슬레이브로 전달하여 전송지연을 구할 수 있다. 슬레이브가 양방향 지연의 평균값을 이용하여 클럭을 보정함으로서 마스터와 슬레이브의 동기가 이루어진다. 이 과정은 양방향의 전송지연이 동일하다는 가정하에서 이루어지며, 아래

식으로부터 전송지연을 추정할 수 있다.

$$Delay = \frac{TS2 - TM2}{2} \quad (2)$$

IEEE1588에서는 위의 두 과정을 주기적으로 반복하여 전체 노드의 시간을 동기화하는데, 이 과정을 수행하는 전용 칩을 적용할 경우 usec 또는 그 이상의 동기정밀도를 확보할 수 있다. 반면 전용 칩을 사용하지 않을 경우 시간정보를 획득하고 연산을 수행하기 위한 소요시간과 정확도에 따라 동기성능의 차이를 보일 수 있다.

3.3 CAN 노드에서의 IEEE1588 메시지 구현

앞 절에서 소개한 것처럼 IEEE1588 알고리즘의 구현을 위해 Sync, FollowUp, DelayRequest, DelayResponse 메시지가 필요하다. Sync와 DelayRequest는 재동기 주기에 따라 전송된 후, 수신노드에서 메시지를 받은 시점에 해당 노드의 로컬클럭 정보를 저장하는 기능이 필요하다. 반면, FollowUp과 DelayResponse는 저장된 시간정보를 수신노드에 전달하게 된다. 소프트웨어 기반으로 IEEE1588 시간동기를 구현하기 위해서 그림 4와 그림 5의 순서도에 따라 CAN 메시지를 생성한다. 이때 메시지 수신에 즉각적인 반응을 유도하기 위해서 하드웨어 인터럽트를 이용하여 동기에 필요한 메시지를 구현한다. IEEE1588의 구현에 사용된 CAN 메시지의 ID는 표 1과 같다.

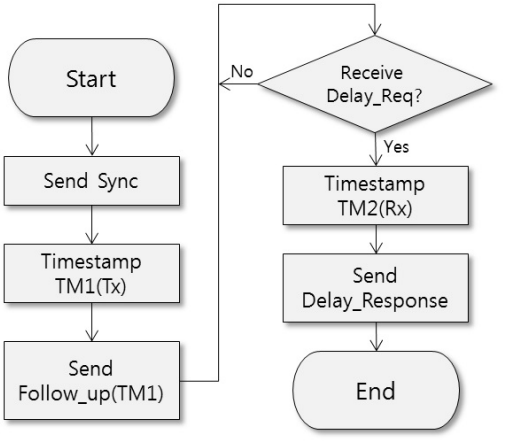


그림 4. Master 노드의 순서도
Fig. 4. Flowchart of master node.

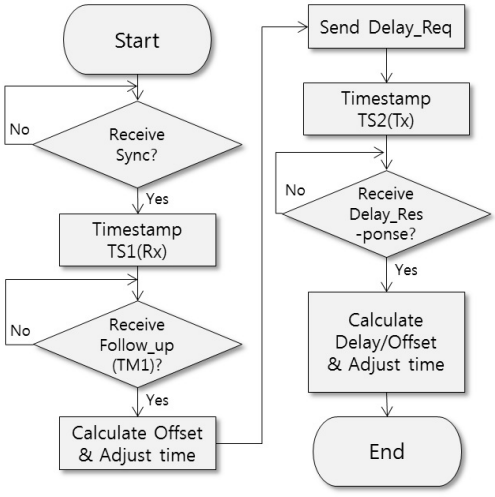


그림 5. Slave 노드의 순서도
Fig. 5. Flowchart of slave node.

표 1. IEEE1588 구현에 사용된 CAN 메시지
Table 1. CAN messages used for IEEE1588.

Message Name	ID	Data	Remark
Sync	0x01	-	
FollowUp	0x02	TM1	
DelayReq	0x03	-	
DelayResponse	0x04	TM2	
DelayOffset	0x05	Delay, Offset	For CANalyzer

3.4 시간동기 정밀도 모델링

CAN 네트워크에 연결된 임의의 노드 i의 로컬클럭 값을 $C_i(t)$라고 하자. 여기서 t는 실시간을 가리킨다. 실시간 t를 완벽하게 측정할 수 있는 클럭은 사실상 불가능하므로 로컬클럭은 실시간과 차이가 날 수 밖에 없다. 온도와 습도 등 외부환경의 변화를 무시할 수 있다면, 이 오차는 시간의 흐름에 따라 일정한 비율로 누적이 되는데, 이 값을 드리프트율(drift rate, ρ)이라고 부른다. 드리프트율은 시간동기 오차를 유발하는 여러 요인 가운데서 가장 큰 비중을 차지한다. 임베디드 시스템에 적용되는 클럭의 드리프트율은 다소 차이가 있지만 일반적으로 1~100μs/s 정도의 값을 가진다.

그림 6에 나타낸 것처럼, 드리프트율 ρ을 가지는 임의의 두 로컬클럭 $C_i(t)$ 및 $C_j(t)$의 오차를 시간차(offset) $\Phi_{ij}(t)$라고 하자. 두 클럭의 시간차가 너무 커

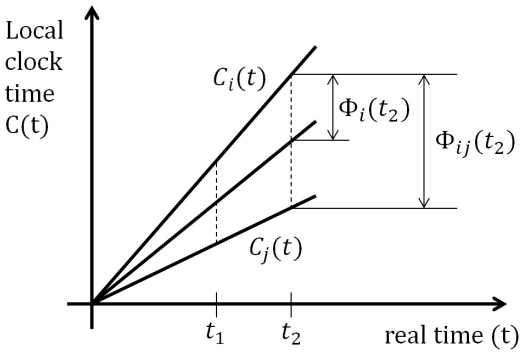


그림 6. 로컬 클럭과 마스터 클럭의 시간차
Fig. 6. Offset between local clocks and master clock.

지지 않도록 주기 R마다 재동기를 실행한다면 시간차 $\Phi_{ij}(t)$는 다음과 같이 표현될 수 있다.

$$\Phi_{ij}(t) = |C_i(t) - C_j(t)| \le \delta, \ \forall t \tag{3}$$

즉 주기적인 재동기로 인해 두 클럭의 시간차는 최대 δ보다 크지 않게 되며, 여기서 δ를 시간동기 정밀도(precision)라고 정의한다. 마스터-슬레이브 방식의 시간동기 기법 적용시 동기오차는 아래 식과 같이 재동기 주기와 클럭의 드리프트율에 비례하는 관계로 표현할 수 있다.

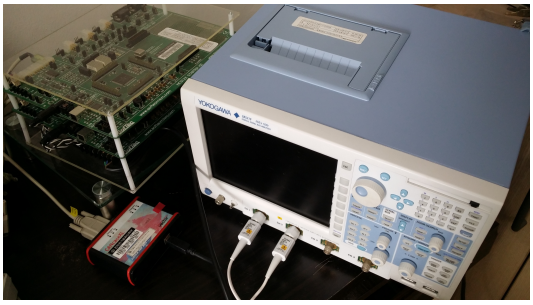
$$2\rho R + \xi \le \delta \tag{4}$$

여기서 2를 곱해 준 것은 동일한 절대값의 드리프트율을 가지는 두 개의 클럭이 서로 반대 방향으로 드리프트하는 경우를 반영한 것이며, ξ는 클럭측정오차(clock reading error)를 가리킨다. 클럭측정오차는 네트워크 전송시간, 타임스탬프를 획득하는 과정에서 발생하는 지연, 양자화 및 비트수 제약에 따른 오차 등을 포함하며, 일반적으로 동일한 시스템에 대해서는 비교적 일정하다. 따라서 드리프트율을 측정함으로써 마스터-슬레이브 기반의 시간동기 정밀도를 추정할 수 있다.

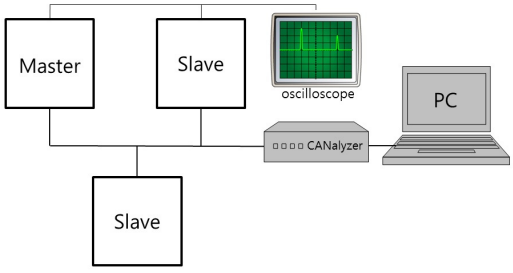
Ⅳ. 실험 및 고찰

4.1 실험용 CAN 네트워크 구성

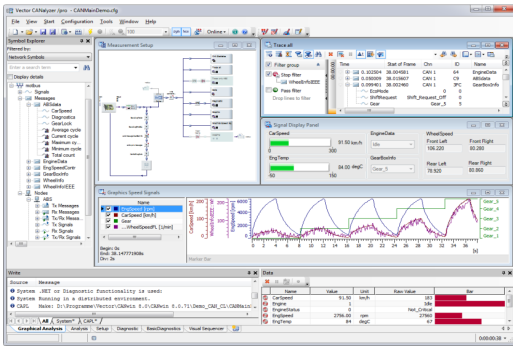
CAN 네트워크의 시간동기를 위해 IEEE1588 알고리즘을 소프트웨어 방식으로 구현하고 그에 따른 동기정밀도를 측정하고자 3개의 HCS12X Starter Kit 보드를 이용하여 그림 7(a)-(c)와 같은 실험환경을 구



(a) Equipment of experimental CAN network



(b) Block diagram of experimental setup.



(c) Screenshot of CANalyzer.

그림 7. 실험용 CAN 네트워크 구성
Fig. 7. Experimental CAN setup

성하였다. Freescale사에서 제작한 HCS12X Starter Kit 보드는 MC9S12XDP512 칩을 기반으로 하고 있으며, 모두 5개의 MSCAN 인터페이스를 제공한다. 각 인터페이스는 'Fault-Tolerant CAN,' 'High-Speed CAN' 등 서로 다른 특성과 트랜시버를 지원하는데, 실험에서는 High-Speed CAN을 지원하는 MSCAN2와 PCA82C250 트랜시버를 사용하였다. 메시지 전송에 관련된 모든 동작은 Freescale사의 운영체제인 Code Warrier Studio에서 동작하며, 이로 인한 오버헤드가 포함되었다.

IEEE1588 알고리즘 구현은 앞선 설명에서처럼 소프트웨어 기반으로 구현하였으며, 마스터 1개와 슬레

이브 2개로 구성하였다. 시간동기 성능을 측정하기 위해서 각 노드의 로컬클럭을 기준으로 발생시킨 신호를 오실로스코프를 이용하여 비교하였다. 아울러 시간동기 관련 메시지의 송수신 상태, 버스의 부하 상태 등을 실시간으로 분석하고자 Vector사의 CANalyzer[32]를 사용하였다. 특히 슬레이브 노드에서 계산된 지연 및 시간차를 CANalyzer로 전송하여 실시간으로 그 값을 모니터링 하였다.

실험은 다음과 같이 두 가지로 나누어 실행하였다: (i) 실험에 사용된 클럭의 드리프트율을 측정하고 이를 식(4)에 적용하여 시간동기 정밀도를 예측; (ii) 구현된 IEEE1588 시간동기 소프트웨어를 적용하여 동기정밀도 측정.

4.2 클럭 드리프트율 측정 결과

그림 8은 각 노드의 로컬클럭을 기준으로 328ms마다 발생하는 신호의 상승에지를 16.4초 동안 오실로스코프로 관찰한 결과를 나타낸다. 이 때 시간동기 알고리즘은 적용하지 않은 상태이다. 오실로스코프 화면의 위쪽은 Ch1로 측정한 마스터 노드의 신호이며, 아래쪽은 같은 시간동안 Ch2를 통해 반복적으로 측정한 슬레이브 노드의 신호를 나타낸다. 이 실험결과를 통해서 16.4초 후의 두 클럭 시간차는 약 2.5ms가 발생하였으며, 이는 슬레이브의 클럭이 마스터 클럭을 기준으로 약 152μs/s의 드리프트율을 가지는 것을 의미한다. 재동기 주기 R=1초, 그리고 클럭측정오차 ξ를 무시할 수 있다고 가정했을 때, 측정된 드리프트율을 식(4)에 대입하면 동기정밀도는 약 304μs로 예상할 수 있다. 측정한 ρ의 값이 152μs/s로서 다소 큰 이유는 마스터 클럭과 슬레이브 클럭이 서로 반대 방향, 즉 두 클럭이 각각 빨라지는 특성과 느려지는 특성을 가지기 때문인 것으로 판단된다.

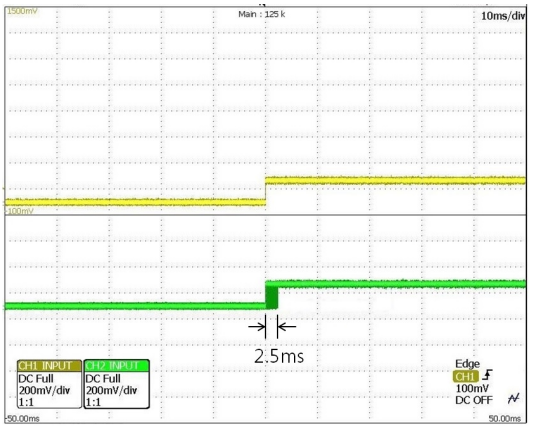


그림 8. 마스터를 기준으로 측정된 슬레이브의 drift rate
Fig. 8. Measured drift rate of slave with respect to master

4.3 동기 정밀도 측정 결과

그림 9는 본 논문에서 구현한 IEEE1588 시간동기 소프트웨어를 적용한 상태에서 측정한 마스터와 슬레이브의 출력신호를 비교 제시한 결과이다. 그림 5와 마찬가지로, 오실로스코프의 위쪽은 마스터 클럭의 출력신호, 그리고 아래쪽은 슬레이브 클럭의 출력신호를 나타낸다. 즉 328ms 간격으로 출력되는 슬레이브 신호는 마스터 클럭을 기준으로 최대 160us 이하의 시간차를 유지함을 알 수 있다. 그런데 본 실험에서 측정한 시간차는 마스터와 1개의 슬레이브 사이에 발생한 값이다. 반면, 2개의 슬레이브 사이의 시간차를 측정할 경우, 두 개의 슬레이브 노드가 서로 반대 방향으로 시간차가 발생하는 경우를 예상할 수 있으므로 최대 시간차는 160us의 두 배인 320us라고 추정할 수 있다. 앞선 실험에서 예상한 동기 정밀도 304μs와 다소 차이가 있음을 알 수 있는데, 이는 동기정밀도 추정시 고려하지 않은 클럭측정오차에 기인하는 것으로 판단된다. 클럭측정오차는 일반적으로 일정한 값을 나타내므로, 식(4)를 통해 추정한 동기정밀도와 실제 측정값은 비교적 일치하는 것을 확인할 수 있다.

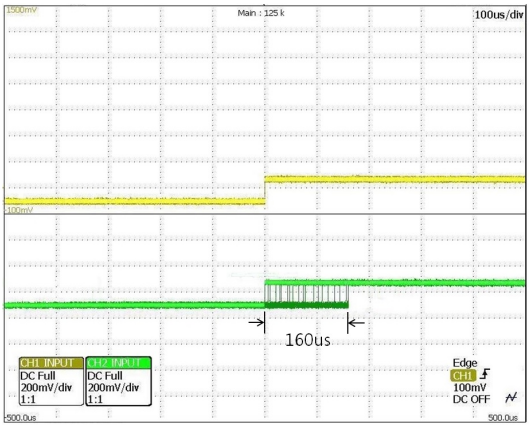


그림 9. 동기정밀도 측정 결과
Fig. 9. Measured synchronization precision.

V. 결 론

네트워크 기반 임베디드 시스템의 구현 시 각 노드의 시간동기를 유지하는 것은 매우 중요하다. 지금까지 다양한 시간동기 기법들이 연구되었으나,

IEEE1588과 같이 전용 칩을 이용하는 표준화된 동기 기법을 적용할 경우 많은 장점들을 기대할 수 있다. 하지만 현재까지 CAN 버스를 위한 IEEE1588 전용 칩은 알려진 것이 없다. 본 논문에서는 CAN을 위한 IEEE1588 전용 칩이 제공되지 않는 점을 고려하여 CAN 패킷과 소프트웨어만을 이용하여 IEEE1588 알고리즘을 구현하고, 동기정밀도를 계산하기 위한 모델을 제시하였다. 그리고 모델을 통해 추정한 동기정밀도와 3개의 노드로 이루어진 실험용 네트워크를 이용해 측정한 동기정밀도를 비교분석함으로써 제안된 방법의 타당성을 확인하였다. 후속 연구로서, 본 논문에서 구현한 시간동기 소프트웨어를 실제 시스템과 유사한 환경에서 검증할 필요가 있다. 노드 수 및 버스 부하가 증가할 경우, 규모가 작은 실험용 네트워크와 달리 시간동기 메시지 전송에 불규칙적인 장애가 발생할 가능성이 높아지므로 이에 대한 분석이 필요하다. 아울러, FPGA 기반으로 CAN을 위한 IEEE1588 알고리즘을 구현하고 그 결과를 본 논문에서 제시한 소프트웨어 기반 결과와 비교분석하는 연구도 진행할 예정이다.

References

[1] ISO 11898, *Road vehicles – interchange of digital information – Controller Area Network(CAN) for high-speed communication*, ISO, Nov, 1993.

[2] H.A. Thompson, et al., "A CANbus-based safety-critical distributed aeroengine control systems architecture demonstrator," *J. MICPRO*, vol. 23, pp. 345-355, Nov. 1999.

[3] M. A. Parker, L. Ran, and S. J. Finney, "Distributed control of a fault-tolerant modular multilevel inverter for direct-drive wind turbine grid interfacing," *IEEE Trans. Ind. Electron.*, vol. 60, no. 2, pp. 509-522, Feb. 2013.

[4] M. Song, E. Kim, and D. Lee, "Reliability analysis of dual-channel CAN bus for submarine combat system," *J. KICS*, vol. 28, no. 12, Dec. 2013.

[5] D. Kim, S. Yang, H. Kim, Y. Son, and S. Han, "Implement of VLC System Based on CAN Communication," *J. KICS,* vol. 36, no. 11, Nov. 2011.

[6] T. Führer, B. Müller, W. Dieterle, F. Hartwich, R. Hugel, M. Walther, and R. B. GmbH, "Time triggered communication on CAN," in *Proc. Int. CAN Conf.,* Amsterdam, The Netherlands, 2000.

[7] L. Almeida, P. Pedreiras, and J. A. Fonseca, "The FTT-CAN protocol: Why and how," *IEEE Trans. Ind. Electron.,* vol. 49, no. 6, pp. 1189-1201, Dec. 2002.

[8] M. Short and M. Pont, "Fault-tolerant time-triggered communication using CAN," *IEEE Trans. Ind. Informat.,* vol. 3, no. 2, pp. 131-142, May 2007.

[9] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communication of the ACM*, vol. 21, no. 7, pp. 558-565, July 1978.

[10] G. Rodríguez-Navas and J. Proenza, "Clock synchronization in CAN distributed embedded systems," in *Proc. RTN,* Italy, June 2004.

[11] M. Gergeleit and H. Streich, "Implementing a distributed high-resolution real-time clock using the CAN-bus," in *Proc. 1st CiA Int. CAN Conf.*, Sept. 1994.

[12] D. Lee, "Fault-tolerant clock synchronization for low-cost networked embedded systems," *J. Korean Sensors Soc.*, vol. 16, no. 1, pp. 52-61, 2007.

[13] G. Rodriguez-Navas, S. Roca, and J. Proenza, "Orthogonal, fault-tolerant, and high-precision clock synchronization for the controller area network," *IEEE Trans. Ind. Informat*. vol. 4, no. 2, May 2008.

[14] IEEE Instrumentation and Measurement Society, "*IEEE standard for a precision clock synchronization protocol for networked measurement and control system*," July 2008.

[15] M. Anyaegbu, C. Wang, and W. Berrie, "Dealing with packet delay variation in IEEE 1588 synchronization using a sample-mode filter," *IEEE ITS Mag.*, vol. 5, no. 4, pp. 20-27, 2013.

[16] G. Giorgi and C. Narduzzi, "Performance analysis of Kalman-filter-based clock synchronization in IEEE 1588 networks,"

IEEE Trans. Instrumentation and Measurement, Vol. 60, No. 8, pp. 2902-2909, 2011.

[17] J. H. Han and D. K. Jeong, "A practical implementation of IEEE 1588-2008 transparent clock for distributed measurement and control system," *IEEE Trans. Instrumentation and Measurement*, vol. 59, no. 2, pp. 433-439, 2010.

[18] Z. Du, Y. Lu, and Y. Ji, "An enhanced end-to-end transparent clock mechanism with a fixed delay ratio," *IEEE Commun. Lett.*, vol. 15, no. 8, pp. 872-874, 2011.

[19] A. Bondavalli, F. Brancati, A. Flammini, and S. Rinaldi, "Master failure detection protocol in internal synchronization environment," *IEEE Trans. Instrumentation and Measurement*, vol. 62, no. 1, pp. 4-12, 2013.

[20] G. Gaderer, P. Loschmidt, and T. Sauter, "Improving fault tolerance in high-precision clock synchronization," *IEEE Trans. Ind. Informat.*, vol. 6, no. 2, pp. 206- 215, 2010.

[21] D. Kim, Y. Jo, D. Lee, "Analysis of Transmission Delay and Fault Recovery Performance with EtherCAT for In-Vehicle Network," *J. KICS,* vol. 37C, no. 11, Nov. 2012.

[22] D. Lee, J. Allan, and S. Bennett, *Distributed real-time control systems using CAN*, in Fieldbus Technol. (Ed. N. P. Mahalik), Springer, 2003.

[23] T.C. Yang, "Networked control system: a brief survey," *IEE Proc. Control Theory Appl.*, vol. 153, no. 4, pp. 403-412, 2006.

[24] R. Luck and A. Ray, "An observer-based compensator for distributed delays," *Automatica,* vol. 26, no. 5, pp. 903-908, 1990.

[25] H. Chan and U. Ozguner, "Closed-loop control of systems over a communications network with queues," *Int. J. Control*, vol. 62, no. 3, pp. 493-510, 1995.

[26] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, Kluwer, 1997.

[27] TTTech Computertechnik GmbH, *Time-triggered protocol TTP/C, High-Level Specification Document*, Available: http://www.tttech.com

[28] *FlexRay Requirement Specification*, ver 2.0.2, April 2002, Available: http://www.flexray.com

[29] A.V. Schedl, "Design and simulation of clock synchronization in distributed systems," Ph.D. dissertation, Technical University of Vienna, Austria, 1996.

[30] J. M. Rushby and F. Henke, "Formal verification of algorithms for critical systems", *IEEE Trans. Software Engineering*, vol. 19, no. 1, pp. 13-23, 1993.

[31] D. L. Mills, A. Thyagarjan, B. C. Huffman, "Internet timekeeping around the globe," *Tech. Report*, University of Delaware, 2002.

[32] Vector GmbH, CANalyzer, Available: http://vector.com/vk_canalyzer_ko.html

박 성 원 (Sung-won Park)

2012년 8월 : 경북대학교 전자공학부 졸업

2012년 9월~현재 : 경북대학교 전자공학부 석사과정

<관심분야> 네트워크 기반 제어, 고장대처 제어

김 인 성 (In-sung Kim)

2005년 2월 : 동아대학교 금속공학과 졸업

2014년 2월 : 경북대학교 산업대학원 제어 및 계측공학전공 석사 졸업

2006년 3월~현재 : 경상공업고등학교 지능형로봇과 재직중

<관심분야> 네트워크 기반 제어

이 동 익 (Dongik Lee)

1987년 8월 : 경북대학교 전자공학과 공학사
1990년 2월 : 경북대학교 전자공학과 공학석사
1990년 2월~1997년 8월 : 국방과학연구소 연구원
1997년 9월~2002년 4월 : 영국 셰필드대학교 자동제어시스템공학과 공학박사
2002년 1월~2005년 3월 : 영국 DRTS Ltd 공동설립
2005년~현재 : 경북대학교 전자공학부 부교수
<관심분야> 네트워크 기반 제어, 시스템 안전, 지능형 자동차, 수중로봇