논문 2012-07-29

가상물리시스템 개념을 이용한 임베디드 제어 네트워크 시스템 설계에 관한 연구

(Study on Design of Embedded Control Network System using Cyber Physical System Concept)

박 지 훈, 이 석, 이 경 창*
(Jee-Hun Park, Suk Lee, Kyung-Chang Lee)

Abstract : Recent advances in electronics have enabled various conventional products to incorporate with numerous powerful microcontroller. Generally, an embedded system is a computer system designed for specific control functions within a larger system, often with real-time computing constraints. The growing performance and reliability of hardware components and the possibilities brought by various design method enabled implementing complex functions that improve the comport of the system's occupant as well as their safety. A cyber physical system (CPS) is a system featuring a tight combination of, and coordination between, the system's computational and physical elements. The concept of cyber physical system, including physical elements, cyber elements, and shared networks, has been introduced due to two general reasons: design flexibility and reliability. This paper presents a cyber physical system where system components are connected to a shared network, and control functions are divided into small tasks that are distributed over a number of embedded controllers with limited computing capacity. In order to demonstrate the effectiveness of cyber physical system, an unmanned forklift with autonomous obstacle avoidance ability is implemented and its performance is experimentally evaluated.

Keywords : Cyber physical system (CPS), Embedded control system, Unmanned autonomous forklift, Controller area network (CAN)

Ⅰ. 서론

임베디드 시스템은 다른 시스템의 일부로 내장되어 전체 시스템의 내·외부에 대한 센싱, 모니터링, 제어를 수행하는 마이크로프로세서 기반 디지털 시스템으로 정의된다 [1-3]. 임베디드 시스템은 1950년대 통신장비를 제어하기 위한 컴퓨팅 시스템을 전자기기에 내장하면서 등장하기 시작했으며, 이후 주로 군사용기기와 특정 산업용기기 등에서 제한적으로 사용되었다. 그 당시에는 임베디드 시스템을 사용하기 위해서는 디지털 로직으로 구성된 FPGA(Field-Programmable Gate Array)나 SoC (System on Chip)를 사용자가 직접 설계해야 하는 등 개발 비용이 상대적으로 높았으며, 일반화되었던 PC에 비해서 개발 환경도 충분하지 않았기 때문이다 [4].

그러나 1980년대 이후 반도체 설계 기술의 획기적인 발전으로 인하여 마이크로프로세서의 성능이 향상되고 가격 경쟁력이 높아짐에 따라 임베디드 시스템의 적용이 확대되기 시작했다. 이와 동시에, 임베디드 시스템 개발을 위한 다양한 크로스 컴파일 환경이 개발되었고, 소프트웨어(펌웨어)의 설계를 통해서 일정 부분 범용성을 제공하는 마이크로컨트롤러가 등장하면서 개발 비용도 범용 PC에 비

* 교신저자(Corresponding Author)
논문접수 : 2012. 07. 30., 채택확정 : 2012. 09. 06.
박지훈 : 자동차부품연구원 스마트자동차기술연구센터
이 석 : 부산대학교 기계공학부
이경창 : 부경대학교 제어계측공학과

ISSN : 1975-5066

해서 낮아지게 되었다 [5]. 이런 이유로 임베디드 시스템은 산업용 제어시스템을 비롯해서 휴대폰, 가전제품, 조선산업, 지능형 로봇, 지능형 자동차와 같은 다양한 분야에서 광범위하게 적용되고 있다.

임베디드 시스템은 오랜 기간 동안 오류 없이 안정적으로 작동하도록 설계되어야 하기 때문에 시스템에 내장되는 소프트웨어의 구조와 시스템 설계법은 더욱 신중한 개발과 테스트가 요구된다. 더욱이 최근에는 전체 시스템의 성능 및 정밀도를 향상시키기 위해서 더욱 더 다양하고 복잡한 알고리즘이 도입되고 있으며, 점차 복잡한 대형 임베디드 시스템(complex large-scale embedded system)으로 발전하고 있다 [6]. 따라서 이전에 비해 더욱 강력한 연산 능력을 제공하는 마이크로컨트롤러가 필요하며, 소프트웨어 복잡도 증대에 따라 내장된 알고리즘의 원활한 동작을 위한 고도의 프로그래밍 기법이 요구되고 있다. 더욱이 하나의 임베디드 시스템을 통해서 시스템을 제어하고 운영하던 구조에서 벗어나서 유·무선 네트워크를 통해 정보를 서로 교환하면서 더욱 대형화되고 복잡해진 시스템을 제어하고 있다 [7].

현재 임베디드 시스템 개발자는 제품의 성능이나 기능을 향상시키기 위해 고 성능의 하드웨어를 사용하면서 소프트웨어 최적화에도 많은 노력을 기울이고 있다. 그러나 임베디드 시스템이 다양한 현실 세계에 적용되고, 고 신뢰성의 서비스를 요구하는 시스템이 증가하면서 기존의 임베디드 시스템의 개발 방법으로는 새로운 분야에서의 요구를 만족시키기 못하고 있다. 특히 안전에 관련된 중요시스템들(자동차, 항공기, 로봇 및 자동화, 무기/전투체계 등)은 고성능화, 복합기능화, 자동화, 지능화, 상호연동성, 실시간성 및 고신뢰성 등의 다양한 요구사항을 동시에 만족시켜야만 하는 복합 시스템이다. 따라서 시스템 구성 컴포넌트들 간이나 외부 환경과의 상호 작용으로 인한 효과를 동시에 이해하고 해석해서 제어하기 위한 시스템을 구성해야 한다. 하지만 갈수록 증가하고 있는 구성 컴포넌트는 시스템 설계의 복잡도를 증가시키게 되며, 심지어 전체 시스템의 신뢰성에 치명적인 영향을 미치게 된다. 이런 이유로, 최근에는 시스템 복잡도와 신뢰성 문제를 해결하고 임베디드 시스템을 지속적으로 발전시키기 위한 목적의 가상물리시스템(cyber physical system)이라는 개념이 등장하였다.

가상물리시스템은 적용분야가 다양한 만큼, 다양한 정의가 있지만 임베디드 시스템공학 분야에서는 “센서와 액추에이터를 가지는 물리시스템과 이를 제어하는 컴퓨팅 시스템들이 밀집되어 연동된 복합 시스템(system of system)”으로 정의하고 있다 [8]. 가상물리시스템에서는 전체 기능의 분산에 의한 시스템의 모듈화(modularization), 네트워크를 통한 모듈간의 정보 공유(networking), 분산된 모듈들간의 협조제어(coordination)가 강조된다. 이러한 개념을 이용하여 임베디드 시스템을 구현하면, 단일 모듈에 구현되어야 하는 기능들을 줄여서 설계의 난이도 및 시스템의 복잡도를 감소시키는 것이 가능하다. 또한, 서로 다른 분야의 설계 인력이 단일 모듈을 구성함에 있어, 표준화된 네트워크 입력과 출력을 이용하기 때문에 서로의 개발 내용을 완전히 이해하지 않더라도 전체 시스템의 구현에는 영향을 주지 않는다. 즉, 완전히 새로운 기능이 추가된다고 하더라도 각각의 설계자는 표준화된 네트워크 입력과 출력만을 반영하여 구현할 수 있기 때문에 설계의 유연성도 향상시킬 수 있다.

전체 시스템에서 기능을 분산시켜 모듈화하고, 모듈간의 협조제어를 수행해서 전체 시스템을 제어하기 위해서는 모듈에서 생성된 모든 정보를 공유할 수 있는 네트워크가 필수적이다. 특히, 가상물리시스템에서 사용되는 네트워크는 물리적 요소가 가상 요소에 의해서 제어되며, 이러한 정보는 대부분 실시간성이 요구된다. 기존의 네트워크 기반 제어 시스템 관련 연구에서 알려진 바와 같이 네트워크 프로토콜에 의하여 발생되는 네트워크 지연은 전체 시스템의 성능에 큰 영향을 미칠 수 있다. 따라서, 가상물리시스템을 구성할 때에는 네트워크 지연이 시스템에 미치는 영향을 분석하고 이를 최소화 할 수 있는 임베디드 제어 네트워크 설계 방법이 필요하다.

본 논문은 서론을 포함하여 5장으로 구성되어 있다. 2장에서는 가상물리시스템의 개념에 대해서 기술하였으며, 3장에서는 가상물리시스템을 이용한 임베디드 제어 네트워크 시스템의 설계 방법에 대해서 기술하였다. 다음으로 4장에서는 본 논문에서 제안하고 있는 설계 방법의 유용성을 확인하기 위하여 테스트베드를 구성하고 성능을 평가하였다. 마지막으로 5장에서는 결론 및 향후 연구 방향에 대해서 기술하였다.

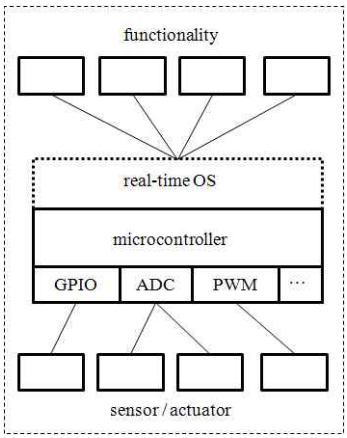
Ⅱ. 가상물리시스템 개요

가상물리시스템은 2006년 말 미국국립과학재단(National Science Foundation)에 의하여 주요 연

구 분야로 지정된 이후, 현재 다수의 정부 연구기관에서 연구가 진행되고 있는 공학의 한 분야이다 [8]. 가상물리시스템은 물리적으로 구성된 실제의 요소가 유·무선 네트워크를 통해 가상의 요소와 통합된 시스템으로 정의된다. 즉, 자동화, 로봇과 같은 물리적인 형태의 시스템은 가상의 개체로 정의되는 컴퓨팅 시스템(임베디드 컴퓨터, 네트워크 감시, 제어기능 등)과 피드백을 하면서 전체 시스템의 신뢰성과 유연성을 확보할 수 있다. 이런 이유로 현재 가상물리시스템은 항공기, 자동차, 로봇, 화학공정, 에너지 시스템, 생산 및 제조, 운송 등과 같은 복잡한 대형 임베디드 시스템을 구현하기 위한 설계 방법에 응용되고 있다. 지난 수 십 년간 연구된 네트워크 기반 제어시스템(Networked Control System, NCS) [9-11]이나 다수의 로봇이 센서 네트워크를 통해 연결되는 MIT의 distributed robotics garden 등이 대표적인 가상물리시스템의 예가 된다 [12].

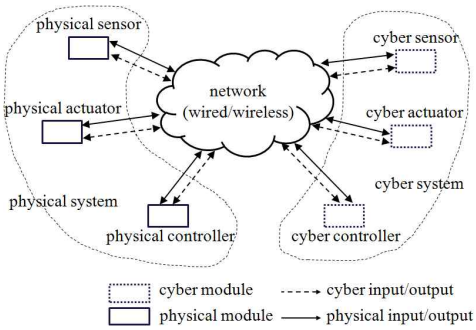
그림 1은 전통적인 임베디드 시스템과 가상물리시스템의 개념을 비교하여 도식적으로 나타낸 것이다. 먼저 그림 1(a)에 나타낸 전통적인 임베디드 시스템에서는 마이크로컨트롤러의 연산능력과 GPIO(general purpose input/output), ADC (analog to digital converter), PWM(pulse width modulation) 등과 같은 기능들이 단일 시스템에 통합되어 있다. 즉, 임베디드 시스템의 요구 조건을 만족시키기 위하여 다수의 센서(sensor)와 액추에이터(actuator), 기능(functionality)들이 하나의 마이크로 컨트롤러에 연결된다.

반면, 그림 1(b)에 나타낸 것과 같이 가상물리시스템은 모터와 같은 하드웨어에 의해 기능을 수행하는 물리적인 시스템(physical system), 연산과 같이 소프트웨어에 의해 기능을 수행하는 가상 시스템(cyber system), 가상 시스템과 물리적인 시스템 간의 정보 공유를 위한 네트워크(network)로 구성되어 있다. 물리적인 시스템에는 PWM 제어기나 서보 제어기 등과 같은 제어기(physical controller), 물리적인 신호를 전기적인 형태로 변환하는 물리적인 입력(physical input)을 가지는 전통적인 형태의 센서(physical sensor), 모터나 유압실린더와 같이 물리적인 출력(physical output)을 가지는 구동기(physical actuator) 등이 있다. 반면 가상 시스템에는 임베디드 시스템에서 구동되는 시스템 다이나믹스 모델(system dynamics model)과 같은 가상 제어기(cyber controller), 다수의 물리적인 센서 또는 가상 센서들의 센서 융합(sensor fusion)에 의한



(a) 기존의 임베디드 시스템

(a) Traditional embedded system



(b) 사이버 물리 시스템

(b) Cyber physical system

그림 1. 기존의 임베디드 시스템과 가상물리시스템의 비교

Fig. 1. Comparison of traditional embedded system and cyber physical system

가상 입력(cyber input)을 만들어내는 가상 센서(cyber sensor), 물리적인 구동기의 출력을 보정하기 위하여 사용되는 가상 출력(cyber output)을 만들어내는 가상 구동기(cyber actuator) 등이 있다. 여기에서, 물리적인 모듈이나 가상 모듈은 네트워크를 통하여 공유되는 입출력 정보를 이용하여 단위 모듈의 성능을 만족시킬 뿐만 아니라 전체 시스템의 요구 조건을 만족시킨다. 또한, 물리적인 시스템에 영향을 주지 않고 다양한 형태의 가상 제어기나 센서, 구동기를 추가함으로써, 시스템의 성능을 향상시킬 수 있다.

이와 같이 가상물리시스템을 이용하여 임베디드 시스템을 구현하게 되면, 단일의 모듈 내에 구현되

어야 하는 기능들을 다수의 마이크로컨트롤러로 나누어서 구현할 수 있다. 또한, 이러한 모듈들은 표준화된 네트워크를 통하여 연결되며, 정보의 상호 교환을 통해 전체 제어시스템을 구성한다. 이러한 네트워크에 연결되어 분산된 모듈간의 정보 교환을 기반으로 한 설계 방법을 이용하는 경우, 복잡한 대형 임베디드 시스템을 단순한 임베디드 시스템들의 조합을 통해 구현할 수 있기 때문에, 기존의 임베디드 시스템에 비해 설계의 난이도를 감소시키고 유연성을 향상시킬 수 있다. 특히, 시스템 다이나믹스 모델을 추가하게 되는 경우, 모델로부터 생성된 정보를 이용하여 임베디드 시스템의 성능을 향상시킬 수도 있다.

Ⅲ. 가상물리시스템 개념을 이용한 임베디드 제어 네트워크 시스템의 설계

가상의 개체와 물리적인 개체가 혼재된 가상물리시스템이 안정적으로 동작하기 위해서는 각 개체들간의 원활한 정보 교환이 이루어질 수 있도록 적합한 유형의 네트워크 프로토콜을 선정하여야 한다. 특히, 가상물리시스템에서 사용되는 네트워크는 물리적 요소가 가상 요소에 의해서 제어되며, 이러한 정보는 실시간성이 요구된다. 네트워크 기반 제어시스템 관련 연구에서 알려진 바와 같이 네트워크 프로토콜에 의하여 발생되는 네트워크 지연은 시스템의 성능에 큰 영향을 미칠 수 있다. 따라서, 가상물리시스템을 구성할 때에는 네트워크 지연이 시스템에 미치는 영향이 최소화되도록 네트워크 프로토콜을 선정하여야 하며, 데이터의 공유를 위해서 메시지 규격을 표준화해서 설계해야 한다. 이를 위해서는 네트워크를 선정하기 전에 개별 프로토콜의 매체 접근제어 방식에 대한 상세한 분석이 선행되어야 하며, 시스템에서 사용되는 모든 메시지에 대한 정보(전송주기, 데이터크기 등)를 분석해야 한다.

그림 2는 가상물리시스템에서 고려해야 하는 지연요소를 타이밍차트를 통해서 나타낸 것이다. 그림 2에서 나타낸 것과 같이 일반적인 가상물리시스템은 센서, 제어기, 액추에이터를 하나의 제어시스템 구성요소로 하며, 기능의 추가나 변경에 따라 센서, 제어기, 액추에이터가 추가적으로 사용된다. 먼저 센서에서 제어기로의 전송지연(τ_k^{sc})은 네트워크로 데이터를 전송하기 위해서 송신 버퍼에서 대기하는 큐 지연(d_{queue}^{sc})과 전송속도에 따른 송신지연(d_{trans}^{sc}), 그리고 매체에서의 전파지연(d_{propa}^{sc})으로 구분할 수 있다. 다음으로 제어기에서의 연산지연(τ_k^c)은 제어기에서 알고리즘을 수행하기 위해서 필요한 연산시간(d_k^c)과 네트워크 태스크를 실행하기 위한 대기 지연($d_k^{hold-on}$)으로 구분된다. 마지막으로 제어기에서 구동기로의 전송지연(τ_k^{ca})은 센서에서 제어기로의 전송지연과 마찬가지로 송신 버퍼의 큐 지연(d_{queue}^{ca})과 송신지연(d_{trans}^{ca}), 전파지연(d_{propa}^{ca})으로 구분된다.

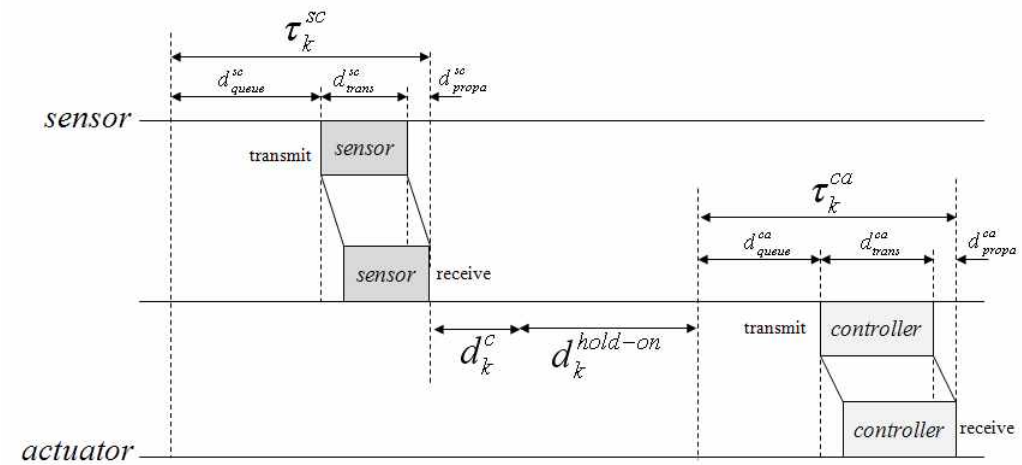


그림 2. 네트워크 제어 시스템에서 전송지연 타이밍 차트
Fig. 2. Timing chart of network delay in networked control system

그림 2에서 나타낸 지연의 속성을 다시 분석해 보면, 우선 전송속도에 따른 송신지연(d^{sc}_{trans})과 매체에서의 전파지연(d^{sc}_{propa})은 전송을 해야 하는 데이터의 비트 수와 통신속도를 통해서 식 (1)과 같이 계산할 수 있으며 일정한 값으로 계산된다. 또한 제어기에서의 연산지연(d^{c}_{k})의 경우에도, 제어기에서 실행해야 하는 제어알고리즘의 명령어 수와 제어기의 연산능력에 따라서 식 (2)와 같이 일정한 값으로 도출 할 수 있다.

$$d^{sc}_{trans}(sec) = \frac{data\ length\ [bit]}{transmission\ speed\ [bit\ per\ second]}$$
$$d^{sc}_{propa}(sec) = \frac{distance\ of\ a\ node[m]}{3\times10^{8}\times0.7\ [m/sec]} \quad (1)$$

$$d^{c}_{k}(sec) = \frac{instruction\ of\ an\ algorithm\ [instruction]}{MIPS\ of\ MCU\times10^{6}\ [instruction\ per\ second]} \quad (2)$$

반면에 태스크 실행을 위한 대기 지연($d^{hold-on}_{k}$)은 운영체제에서 수행중인 태스크의 우선순위에, 송신 버퍼에서의 큐 지연(d^{sc}_{queue}, d^{ca}_{queue})은 네트워크의 현재 트래픽 등에 종속적이며, 이러한 요소들이 시스템의 제어성능에 크게 영향을 미치게 된다.

지금까지는 마이크로컨트롤러에서는 실시간 운영체제와 같은 복잡한 형태의 소프트웨어의 사용이 일반적이지 않았기 때문에 네트워크 설계 시에 제어기의 태스크 대기 지연에 대한 고려가 부족하였다. 하지만 현재와 같이 대형화되고 복잡해진 임베디드 시스템에서는 태스크의 대기 지연이 시스템의 성능에 상당 부분 영향을 미치게 된다. 예를 들어, 1msec마다 모터의 회전 속도를 제어하는 단순한 형태의 시스템(교환되는 데이터의 크기는 오버헤드를 포함하여 센서와 제어기 모두 64bit로 가정)에서 기존의 네트워크에서는 아래 식 (3)과 같이 간단하게 트래픽이 계산된다. 이 경우 250kbps의 대역폭을 제공하는 네트워크를 사용한다면 시스템의 제어성능에 영향을 미치지 않으므로, 실제로 이러한 형태의 네트워크 구성을 현장에서 많이 사용하고 있었다 [12].

$$Traffic = \frac{data\ length\times number\ of\ a\ node}{period\ of\ data}$$
$$= \frac{64bit\times2node}{1msec}\times1{,}000msec = 128kbps \quad (3)$$

그러나 운영체제의 우선순위에 따라서 1msec이상의 태스크 대기 지연이 발생하면 트래픽과는 별개로 1msec마다 데이터를 송신 할 수 없게 되며, 이런 경우 시스템의 정상 동작여부를 판단할 수 없게 된다. 이러한 문제는 현장에서 서로 다른 모터 제어 알고리즘 설계자와 OS의 설계자가 시스템을 구성하는 경우에 종종 발생하며, 설계가 완료된 모터 제어 알고리즘과 OS의 태스크 주기를 변경하는 것은 매우 어려운 작업이 된다. 따라서 네트워크 설계 시에 데이터 발생주기에 대해서 분석하고 태스크 동기화를 수행하는 등 태스크 실행을 위한 대기 지연($d^{hold-on}_{k}$)을 일정하게 유지하기 위한 노력이 모듈의 설계 시에 필요하다.

다음으로 송신 버퍼에서의 큐 지연을 일정하게 유지하기 위해서는 네트워크 트래픽을 최소화하기 위한 전송 절차를 설계해야 한다. 일반적으로 센서, 액추에이터, 제어기가 모두 네트워크에 연결되어 각각의 정보를 모두 네트워크에 송신하면 네트워크 트래픽에 상당한 영향을 주게 된다. 반면에 가상물리시스템과 같이 단위 기능이 모듈화 되어 있는 시스템의 형태로 구성된 모듈을 사용하는 경우, 모듈 내부에서 폐루프 제어시스템을 구성해서 자체적으로 데이터를 관리하도록 구성 되어야 한다. 예를 들면, 일정한 주기마다 제어값을 전송하는 대신에, 제어지령이 변경되는 경우에만 메시지를 전송하는 전송절차를 설계하는 방식을 통해서 트래픽에 대한 영향과 실시간성에 대한 성능을 향상 시켜야 한다.

그림 3은 가상물리시스템을 위해서 설계한 메시지 전송절차와 이 때의 지연요소를 모터의 속도선도와 함께 타이밍차트로 나타낸 것이다. 속도선도는 주 제어기에서 모터속도의 제어를 위해서 사용하는 제어값을 임의로 표현한 것이며, ECU #1은 가상의 제어기, ECU #2는 폐루프로 구성된 물리적인 모터 드라이버를 나타낸 것이며, 그림 3에서 나타낸 것과 같이 시스템에서는 두 번의 속도 제어량을 네트워크를 통해서 전송한다. 속도 제어량이 전달되는 첫 번째 지점(point #1)에서는 모터의 속도 지령 데이터(1st rpm) 를 ECU #2로 전송한다. 이 때 네트워크의 상태에 따라서 제어기와 제어기간의 전송지연($\tau^{cc}_{k} = d^{cc}_{queue} + d^{cc}_{trans} + d^{cc}_{porpa}$)이 발생하게 되며, 메시지를 수신 받은 ECU #2에서는 연산지연과 함께 태스크에 의한 대기 지연(τ^{sc}_{k},$\tau^{hold-on}_{k}$)이 발생한다. 태스크 대기 지연은 주로 ECU #2에서 수행하고 있는 높은 우선순위의 태스크의 실행이 완료된 후, 제어 알고리즘을 수행하는데 필요한 시간이다. 이후 제어기의 정보를 이용해서 액추에이터를 제어한다. 현재의 액추에이터 상태는 주 제어기에서 다음 제어값을 산출할 때 사용되며, 이때 액추에이터의 상태를 측정하기 위한 샘플링지연($\tau^{sampling}_{k}$)도 함께 발생한다.

다음으로 ECU #2에서 ECU #1으로 액추에이터의 상태 정보를 전송하기 위한 지연($\tau_k^{sc} = d_{queue}^{sc} + d_{trans}^{sc} + d_{porpa}^{sc}$)은 전술한 지연과 동일하게 전송 큐 대기지연, 전파지연, 전송지연을 포함하고 있다. 이와 같은 과정에서도 액추에이터의 상태 정보는 액추에이터의 상태가 변경되는 경우에만 전송한다. 따라서 시스템의 상태를 관측하는 시스템 제어기나 액추에이터의 정보를 필요로 하는 경우에는, 자체적으로 이전의 정보가 갱신되지 않는 한 이를 지속적으로 유지하고 있어야 한다. 이상에서 언급한 모든 지연요소를 정리하면 식(4)와 같이 나타낼 수 있다.

식 (4)에서 나타낸 것과 같이, 가상물리시스템을 위한 네트워크에서 발생할 수 있는 모든 지연요소를 요약하면 가상의 제어기와 물리적인 제어기간의 전송지연(τ_k^{cc}), 제어기의 연산지연(τ_k^{c}), 제어기의 태스크 대기지연($\tau_k^{hold-on}$), 액추에이터의 샘플링지연($\tau_k^{sampling}$), 액추에이터에서 제어기로의 전송지연(τ_k^{sc})이 있다. 여기서 네트워크의 대역폭이나 제어기의 연산능력을 고려하면 송신지연(d_{trans}^{cc}, d_{trans}^{sc}), 전파지연(d_{propa}^{cc}, d_{propa}^{sc}), 연산지연(τ_k^{c}), 샘플링지연($\tau_k^{sampling}$)은 일정한 값을 가진다. 반면에 전송 큐 대기지연 (d_{queue}^{sc}, d_{queue}^{ca})과 태스크 대기지연($d_k^{hold-on}$)은 네트워크 트래픽의 상태나 ECU에서의 태스크 우선순위에 따라서 불확정적인 전송지연을 가진다. 따라서 설계 전 시뮬레이션을 통해서 전체 네트워크의 트래픽에 따른 확률적인 전송 큐 대기지연을 분석하는 것이 요구되며, 이를 통해서 시스템의 허용 전송지연($\tau_{delay}^{allowable}$)을 산출해야 한다. 그러나 시간에 따라 트래픽이 가변적인 상황에서 메시지의 전송지연이 허용 전송지연을 초과하지 않도록 네트워크를 설계하는 것은 대부분의 경우에서 간단하지 않다. 따라서 해당 프로토콜에 대해서 트래픽과 전송지연간의 관계를 분석한 기존의 연구결과를 인용하여, 트래픽이 전체 대역폭의 약 50%를 초과하지 않도록 네트워크를 설계해야 한다 [13].

$$\tau_{delay} = \tau_k^{cc} + \tau_k^{c} + \tau_k^{hold-on} + \tau_k^{sampling} + \tau_k^{sc}$$
$$\left\{ \begin{array}{l} \tau_k^{cc} = d_{queue}^{cc} + d_{trans}^{cc} + d_{propa}^{cc} \\ \tau_k^{sc} = d_{queue}^{sc} + d_{trans}^{sc} + d_{propa}^{sc} \end{array} \right\} \quad (4)$$

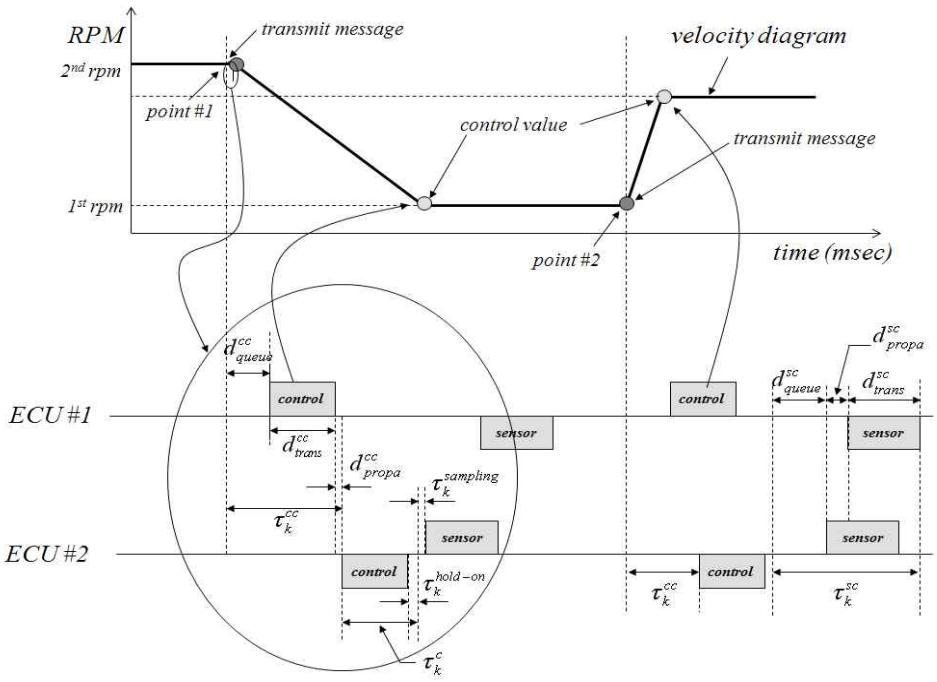


그림 3. 모터 제어 시스템의 메시지 운영 절차

Fig. 3. Message sequence chart for motor control system

그림 4는 이상을 정리하여 본 논문에서 제안하고 있는 가상물리시스템 개념을 이용한 임베디드 제어 네트워크 시스템 설계 방법을 도식적으로 표현한 것이다. 그림 4에서 나타낸 것과 같이 시스템에서 생성되는 모든 데이터를 분석해서 제어와 관련된(control-related) 메시지와 상태 측정과 관련된(monitoring-related) 메시지를 구분해야 한다. 다음으로 제어와 관련된 메시지는 생성되는 개체에서 태스크의 동기기법을 적용하여 태스크 대기지연을 최소화하고, 상태 측정과 관련된 메시지는 필요에 따라서 이를 적용할 수도 있다. 다음으로 메시지가 생성되는 개별 개체에서의 태스크 대기지연에 대한 분석이 완료되면, 네트워크에서 발생할 수 있는 모든 지연요소를 고려하여 소요되는 트래픽을 산출해야 한다. 그리고 산출된 트래픽을 만족시킬 수 있는 프로토콜을 선정해서, 해당 프로토콜에 알맞은 형태의 메시지규격을 설계해야 한다. 모든 메시지의 표준 규격을 설계하고 나면, 각각의 메시지에 대한 전송절차를 설계하고, 시뮬레이션을 통해서 허용 전송지연을 도출해야 한다. 이때, 시스템에서 요구하는 허용 전송지연을 초과하는 메시지지연이 나타날 수 있으며, 이런 경우 트래픽의 영향을 낮출 수 있도록 전송절차를 다시 수정해야 할 수도 있다. 마지막으로 이를 정리하여 시스템에 적용해서 성능을 평가한 후, 추가적인 정보가 요구되거나 실시간 성능을 개선해야 하는 경우가 생기면 앞의 단계에서 이를 반영하고 다시 각각의 절차를 수행해야 한다.

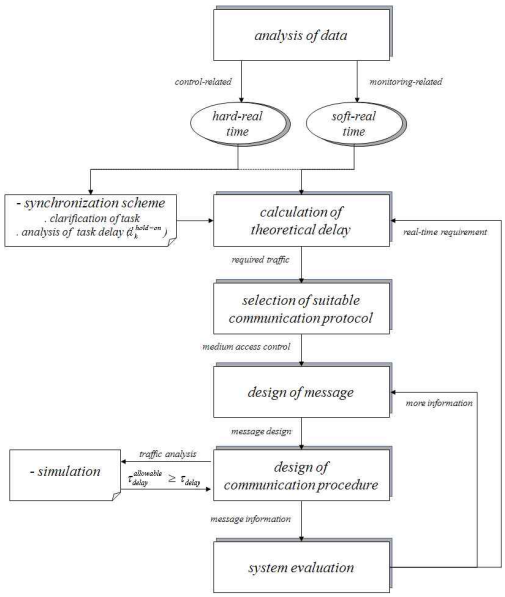


그림 4. 가상물리시스템 개념을 이용한 임베디드 제어 네트워크 시스템 설계 방법

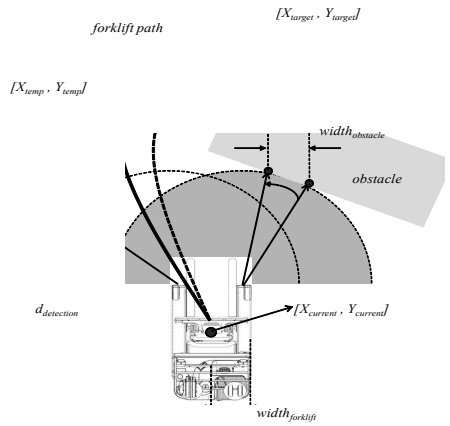
Fig. 4. Design methodology of embedded control network system using cyber physical system concept

IV. 무인지게차에서의 성능 평가

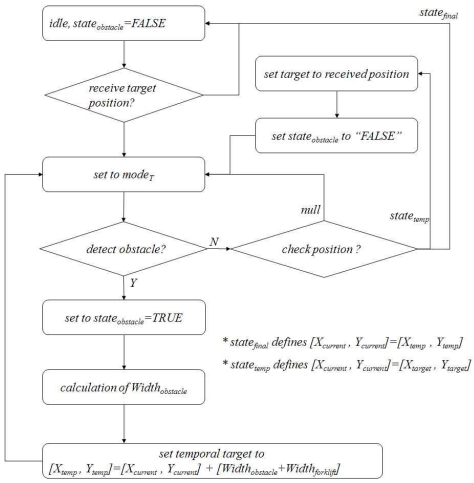
본 논문에서 제안하고 있는 가상물리시스템을 이용한 임베디드 제어 네트워크 시스템 설계 방법의 유용성을 평가하기 위하여, 저자가 이전 연구를 통해서 개발한 그림 5와 같은 무인지게차 시스템을 대상으로 간단한 테스트베드를 구축하고 장애물 회피 알고리즘을 적용하여 평가하였다. 사용된 전동식 지게차는 CLARK사의 1.0톤급 입식 전동식 지게차인 CRX-10 모델이다. 지게차 하부에는 전방 양쪽으로 고정된 휠 2개와 후방 오른쪽 캐스터 휠 1개, 후방 왼쪽의 주행 및 조향 휠(main wheel) 1개로 구성된 4개의 바퀴가 설치되어 있으며 무 부하 시 지게차의 최대 주행 속도는 9.8Km/h이다. 그러나, 유인 운전의 경우에 평균 주행 속도는 안정성 등의 이유로 최대 5.0Km/h가 권장 사항이다 [14].

무인지게차의 구동을 위해서 구성한 H/W에 대해서 간략히 요약하면, 메인 제어 ECU 모듈에는 Freescale사의 16비트급 마이크로컨트롤러인 MC9S12XF512가 사용되었으며, 지게차의 전역 위치를 측정하기 위하여 SICK사의 레이저 기반 위치인식 시스템인 NAV200가 사용되었다. NAV200은 작업 영역에 미리 설치되어 있는 리플렉터(reflector)와의 거리 값을 삼각 측정하여 현재 위치와 방향 정보를 계산할 수 있으며, 20~30m의 영역에서 최대 4mm 및 0.1° 이내의 측정 오차를 가진다. 주행 및 조향 제어 ECU 모듈에는 ATmel사의 8비트급 마이크로컨트롤러인 AT90CAN128이 사용되었으며 주행 속도의 피드백을 위하여 기존 지게차에 설치 되어 있는 엔코더가 사용되었다. 또한, 조향 각도의 피드백을 위한 가변 저항이 조향 모터에 설치 되어있다.

이와 함께, 장애물 감지 및 회피기능의 구현을 위해서 7m 범위 내에 존재하는 장애물의 위치를 측정하기 위하여 SICK사의 레이저 거리 측정 센서인 LMS200이 사용되었다. LMS200은 평면상의 180° 범위를 스캔(scan)하여 장애물의 위치 및 상



(a) 장애물 감지를 통한 회피경로 생성의 도식
(a) Diagram schematic of obstacle detection and avoidance scheme



(b) 장애물 감지 및 회피과정의 흐름도
(b) Flowchart of obstacle detection and avoidance

그림 5. 장애물 감지와 회피 설계
Fig. 5. Obstacle detection and avoidance scheme

대 각도를 측정할 수 있으며, 지게차에는 전방 좌우 각각 1개씩, 후방 중앙 1개를 설치하였다. 그리고 2.5m 이내에 존재하는 근접 장애물을 감지하여 긴급 정지를 하기 위해 SensorTech사의 STMA-506ND 초음파 센서가 사용되었다. 초음파 센서는 지게차 좌・우측면에 각각 50cm간격으로 4개씩 설치하였다. 마지막으로 각 장애물 감지 ECU 별로 센서 신호 처리를 위하여 앞서 설명한 AT90CAN-128이 사용되었다.

그림 5(a)는 주행 중 장애물을 감지하여, 경유지점을 생성하는 과정을 도식적으로 표현한 것이다. 그림 5(a)에서 나타낸 장애물 감지 및 회피를 위한 기법은 네트워크로 정보를 수신하여 연산을 수행하는 가상의 개체를 통해서 구현하였다. 먼저, 지게차가 주행을 하면서, 물리적인 센서인 LMS200을 통해 장애물을 인식하면 장애물의 양 끝의 좌표(X, Y, Θ)를 네트워크로 전송한다. 네트워크를 통해서 장애물의 정보를 수신한 가상의 개체에서는 지게차의 진행방향과 수직으로 장애물의 폭(Width-obstacle)을 계산한다. 그리고 현재 지게차의 위치, 장애물의 폭, 지게차의 기구학적인 폭을 고려하여 임시의 경유지점을 생성한다. 이렇게 생성된 경유지점은 장애물의 크기나 위치가 변경된 정보를 수신 할 때마다 지속적으로 생성해서 네트워크로 송신하게 된다.

그림 5(b)은 지게차의 장애물 감지 및 회피기능을 구현하기 위해서 설계한 간단한 형태의 알고리즘을 나타낸 것이다. 그림 5(b)에서 나타낸 것과 같이, 초기상태에 있던 지게차는 외부에서부터 목적지 좌표(Xtarget, Ytarget)를 수신하면 주행모드(modeT)로 변경하여 주행을 시작한다. 만약 주행 중 지게차 주변의 센서로부터 장애물에 대한 정보가 네트워크를 통해서 수신되면 장애물 상태 비트를 “TRUE”로 설정하고, 네트워크를 통해서 경유지점을 수신 할 때까지 대기한다. 가상의 개체로부터 생성된 경유지점을 네트워크로 수신하면, 지게차는 주행모드로 천이하여 경유지점으로 다시 주행을 시작하게 된다. 주행 중 경유지점에 도착하면 다시 장애물 상태 비트를 “FALSE”로 설정하고, 목표지점을 최초에 수신한 좌표로 변경하여 주행을 시작하고, 목표지점에 도착하면 다음 지령이 수신될 때까지 대기(idle)한다.

위에서 기술한 모든 개체간의 정보를 교환하기 위해 설계한 CAN메시지의 ID와 데이터 길이, 신호는 메시지의 실 시간성에 따라서 표 1과 같이 설계하였다. 모든 메시지는 향후 확장성을 고려하여 CAN 2.0B 프로토콜 규격에서 정의하고 있는 29bit 길이의 extended ID를 사용하였으며, CAN 통신속도는 250Kbps로 설정하였다. 표 1에서는 장애물 감지 및 회피 기능의 수행을 위해서 필수적인 메시지만 나타내었으며, 기존의 방식으로 네트워크를 설계하였을 경우, 트래픽은 76.09Kbps(CAN 오버헤드를 포함함)으로 계산된다. 하지만, 이 외에 지게

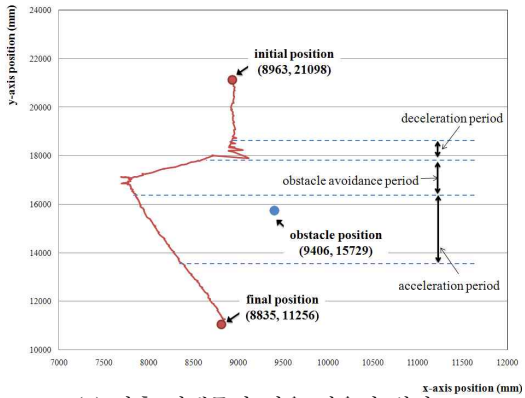
차의 기능을 수행하기 위해서 사용되는 다른 기능을 구현하기 위해서 사용되는 트래픽이 약 90Kbps 정도 있으므로 전체의 트래픽은 약 166.09Kbps이며, 전체 대역폭의 약 66.5%로 과부하 상태인 것을 알 수 있다.

그림 6은 본 논문에서 제안하고 있는 임베디드 제어 네트워크 시스템 설계 방법을 이용하여 네트워크를 설계한 후, 장애물 감지 및 회피기능을 수행한 경우의 실험결과를 나타낸 것이다. 먼저 그림 6(a)와 6(c)는 자율 주행하는 지게차의 좌측과 우측에 각각 장애물이 존재하는 경우, 이를 인식하고 회피하여 최종 목적지까지 이동하는 경로를 그래프로 나타낸 것이다. 다음으로 6(b)와 6(d)는 위에서 설명한 주행을 수행하는 동안 지게차의 속도선도를 나타낸 것이다. 본 논문에서 제안한 설계방법에 따르면, 실시간성이 중요한 메시지는 제어량 또는 상태측정값이 변경되는 경우에만 네트워크로 전송되기 때문에 속도선도를 이용하여 전송되는 메시지를 파악하였다.

먼저 그림 6(a)와 6(c)에서는 목표지점으로 주행하고 있는 지게차의 주행방향을 기준으로 좌측과 우측에 장애물이 감지되면 장애물 회피기능을 수행하기 위해서 지게차의 주행속도를 조절한다. 통합제어기에서는 네트워크로 수신된 장애물의 정보를 이용해서 장애물의 폭(Widthobstacle)을 계산하고, 좌측으로 선회하는 것과 우측으로 선회하는 경우의 최단거리를 판단하는 과정을 거친다. 장애물을 선회하는 동안 지속적으로 장애물의 유무를 판단하며, 장애물이 더 이상 주행경로에 존재하지 않게 되면 다시 주행속도를 조절하여 최종 목적지에 도착한다. 좌측 선회와 우측 선회 실험에서 장애물의 위치는 (x, y)=(9406, 15729)로 고정된 상태로 실험하였으며, 좌측 선회의 경우 초기 위치와 목적지에서의 위치는 각각 (8963, 21098)과 (8835, 11256)으로 측정되었다. 다음으로 우측 선회의 경우 초기 위치와 목적지에서의 위치는 각각 (9567, 21323), (9820, 11732)로 측정되었다. 2가지 실험모두 장애물의 감지 및 회피 기능이 정상적으로 동작하는 것을 확인

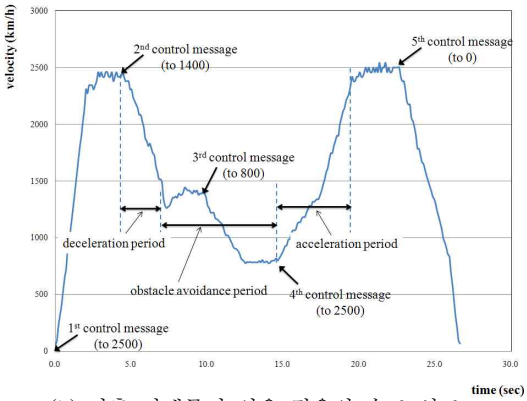
표 1. 장애물 감지와 회피 시 메시지 테이블
Table 1. Message table for the obstacle detection and avoidance

CAN ID	period (msec)	length [bit]	signal
0x01h	-	8	cmd_emergency_stop
0x02h	-	32	node_error_check
0x0Ah	10	16	target velocity
		32	target steer_angle
		16	target_fork_height
0x14h	10	8	current_forklift_position_type
		32	current_forklift_position_data
0x19h	10	16	current_forklift_velocity
0x1Eh	10	16	current_steer_angle
0x32h	20	8	left_obstacle_warning
		48	left_obstacle_position (X, Y, Θ)
0x3Ch	20	8	right_obstacle_warning
		48	right_obstacle_position (X, Y, Θ)
0x46h	20	8	rear_obstacle_warning
		48	rear_obstacle_position (X, Y, Θ)
0x64h ~ 0x6Bh	20	8	sensor_ID
		24	obstacle_distance
0x96h	-	32	cmd_target_x_position
		32	cmd_target_y_position
0xC8h	100	16	current_forklift_status



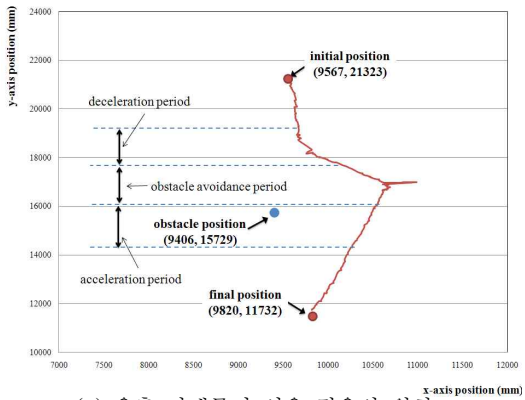
(a) 좌측 장애물이 있을 경우의 위치

(a) Position in case of left side obstacle



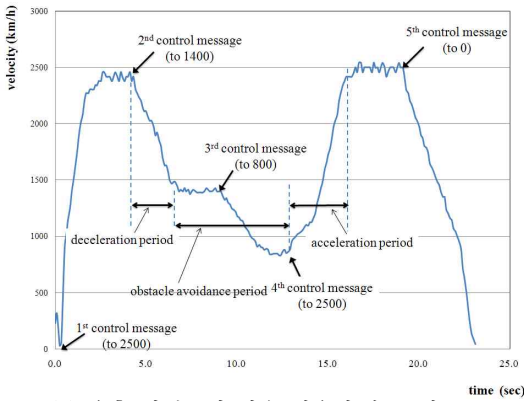
(b) 좌측 장애물이 있을 경우의 속도 선도

(b) Velocity diagram in case of left side obstacle



(c) 우측 장애물이 있을 경우의 위치

(c) Position in case of right side obstacle



(d) 우측 장애물이 있을 경우의 속도 선도

(d) Velocity diagram in case of right side obstacle

그림 6. 지게차의 장애물 회피 시 지게차의 위치와 속도선도 결과

Fig. 6. Result of forklift's position and velocity diagram for obstacle avoidance

할 수 있었다 [15].

다음으로 그림 6(b)와 6(d)는 위의 장애물 감지 및 회피 기능을 수행하는 동안의 지게차의 속도선도를 각각 그래프로 나타낸 것이다. 앞서 설명한 것과 같이, 지게차는 초기 위치에서 주행을 시작할 때, 주행속도를 2.5km/h로 설정하는 명령을 주행제어기로 송신한다. 네트워크로 수신된 속도값은 주행제어기의 내부에 구성된 페루프 제어기를 통해서 다른 명령이 수신될 때까지 제어된다. 주행 중 장애물이 감지되면 통합제어기는 지게차의 속도를 1.4km/h로 변경하는 명령을 송신하고 장애물의 정보를 바탕으로 좌측 선회와 우측 선회의 거리를 계산한다. 장애물을 회피하는 과정에서는 지게차의 속도를 0.8km/h로 변경하는 메시지를 한 번 더 전송했으며, 장애물이 경로상에서 완전히 사라진 다음, 초기의 속도인 2.5km/h로 변경하는 메시지를 전송하였다. 좌측 선회와 우측 선회에서 전송된 속도 변경 명령은 동일하게 평가되었다.

위의 실험에서 표 1에서 정의된 것과 같이 10msec마다 메시지를 전송한다면, 실험이 완료될 때까지(좌측선회 시 26.6초, 우측선회 시 23.1초) 지게차의 속도제어를 위해서 각각 2,660개와 2,310개의 메시지가 전송된다. 반면에 본 논문에서 사용한 제어 네트워크 설계방법을 적용하면 각각 5개의 메시지만을 전송하여 해당 기능을 수행할 수 있으면서도 트래픽을 최소화 할 수 있음을 실험적으로 평가하였다.

V. 요약 및 결론

최근의 임베디드 시스템은 전체 시스템의 성능 및 정밀도를 향상시키기 위해서 더욱 더 다양하고 복잡한 알고리즘이 도입되고 있으며, 점차 복잡한 대형 임베디드 시스템으로 발전하고 있다. 이런 이유로 이전에 비해 더욱 강력한 연산능력을 제공하는 마이크로 컨트롤러가 요구되고, 소프트웨어 복잡도 증대에 따라 내장된 알고리즘의 원활한 동작을 위해 고도의 프로그래밍 기법이 요구되고 있다. 그러나 강력한 성능을 가지는 마이크로컨트롤러나 복잡한 소프트웨어의 사용은 오히려 시스템의 신뢰성과 설계의 유연성을 떨어뜨리는 원인이 되고 있다. 이러한 임베디드 시스템 설계의 문제점을 해결하기 위해서 다양한 형태의 설계방법이 개발 되어 왔으며, 최근에는 전체 시스템을 가상의 개체와 물리적인 개체로 분리하여 시스템을 설계하는 가상물리시스템에 대한 연구가 증가하고 있다.

본 논문에서는 다양한 형태로 설계된 가상의 개체와 물리적인 개체가 혼재해 있는 가상물리시스템에서 개체간의 정보공유를 원활하게 하기 위한 목적으로 가상물리시스템 개념을 이용한 임베디드 제어 네트워크 시스템 설계 방법을 제안하였다. 또한, 제안하고 있는 설계 방법의 유용성을 평가하기 위하여 실제 테스트베드를 구축하여 성능을 평가하였다. 이상을 통해서 다음과 같은 결론을 도출할 수 있었다.

1) 가상물리시스템의 설계 방법을 적용한 임베디드 제어 네트워크 시스템은 시스템에서 요구되는 개별적인 기능들이 ECU로 모듈화되어 독립적으로 동작되고, 각 모듈들은 네트워크를 통하여 전송되는 규격화된 정보 규격을 이용하여 해당 기능을 동작시키게 된다. 따라서 사용자의 요구에 따라 자유롭게 가상의 개체 또는 물리적인 개체를 시스템에 추가하여 구성할 수 있으며, 특정 모듈의 기능을 개선할 때 해당되는 모듈에 탑재된 프로그램만 수정함으로써 다른 모듈에 영향을 미치지 않고 기능을 개선할 수 있다는 장점이 있다. 이러한 특징들은 전체 시스템의 유연성을 증가시켜주고, 시스템 복잡도를 줄여준다.

2) 가상물리시스템에서는 물리적인 개체가 가상의 개체에 의해서 제어되기 때문에 시스템에서 사용되는 모든 정보들은 네트워크를 통해서 공유되고 있으며, 이러한 정보는 높은 실시간성이 요구된다. 또한 가상물리시스템에서는 기존의 네트워크 기반 제어 시스템에 비해서 네트워크에 접속되는 단말(센서, 액추에이터, 제어기)의 수가 급격히 증가하고 있기 때문에 트래픽에 의한 지연관리가 중요하다. 본 논문에서 제안한 네트워크 설계방법을 이용하면 동일한 기능을 수행하면서 트래픽을 줄일 수 있음을 실험적으로 평가하였다.

3) 본 논문에서는 물리적인 개체를 사용하여 기본적인 기능을 설계하고, 가상의 개체를 이용하여 장애물 감지 및 회피기법을 설계하였다. 이와 같은 구조를 통해서 시스템에 대한 사전 지식이 없는 설계자가 새로운 알고리즘을 도입 할 때, 알고리즘의 개발에만 집중할 수 있다. 즉, 표준화된 네트워크의 정보만 설계자에게 주어진다면 설계자는 시스템의 작동방식이나 구조에 대한 분석을 최소화 할 수 있다. 따라서 전체 시스템에서 필요한 기능을 설계하기 위한 다수의 해당분야 전문 설계자가 공동개발을 진행 하는 경우, 상대적으로 시스템의 통합작업이나 설계변경이 용이하다.

이상의 결론을 통해서 가상물리시스템은 기존의 임베디드 시스템에 비해서 설계의 변경이 유용하고 시스템 통합에 유리함을 확인하였다. 특히 시스템에서 사용하는 메시지에 대한 지연요소를 분석하고 전송절차를 개발하는 일련의 과정을 통해서 기존의 시스템과 유사한 성능을 가지면서도 추가적인 성능향상을 기대할 수 있었다. 그러나, 본 논문에서 사용하고 있는 가상물리시스템은 세계적으로 설계의 타당성을 확보하기 위한 이론적인 해석과 같은 연구가 진행되고 있는 초기 연구단계에 있다. 따라서 제안한 설계방법에 대한 이론적인 해석보다는 실험적인 시스템의 성능평가와 정성적인 트래픽 분석과 같은 방법만을 이용하여 유용성을 평가하였다. 향후에는 제안한 설계방법의 이론적인 해석을 비롯하여 네트워크 트래픽의 평가지수 개발과 같은 부분에 대한 연구가 추가적으로 필요하다.

참고문헌

[1] P. Liggesmeyer, M. Trapp, "Trends in embedded software engineering," IEEE Software, Vol. 26, No. 3, pp. 19-25, 2009.

[2] 이소진, 안영호, 한현희, 황영시, 정기석, "임베디드 시스템의 가상 ARM 머신의 개발," 임베디드공학회논문집, Vol. 3, No. 1, pp. 19-24, 2008.

[3] P. Martí, M. Velasco, J.N. Fuertes, A. Camacho, G. Buttazzo, "Design of an embedded control system laboratory experiment," IEEE Transactions on Industrial Electronics, Vol. 57, No. 10, pp. 3297-3307, 2010.

[4] E. Madnick, Y. Stuart, R. Wang, "Performance analysis of multi-microprocessor systems," IEEE Transactions on Industrial Electronics, Vol. 34, No.3, pp. 345-350, 1987.

[5] E.A. Lee, "Absolutely positively on time: what would it take?," Computer, Vol. 38, No. 7, pp. 85-87, 2005.

[6] E. Nasr, J. McDermid, G. Bernat, "A technique for managing complexity of use cases for large complex embedded systems," Proceedings on ISORC'02, pp. 225-232, 2002.

[7] T. Furuyama, "Trends and challenges of large scale embedded memories," Proceedings of the IEEE 2004 Custom Integrated Circuits Conference, pp. 449-456, 2004.

[8] E.A. Lee, "Cyber physical systems are computing foundation adequate?," Proceedings on NSF Workshop on Cyber Physical Systems, 2006.

[9] M.Y. Chow, Y. Tipsuwan, "Network-based control systems: a tutorial," Proceedings on IECON'01, Vol. 3, pp. 1593-1602, 2001.

[10] K.C. Lee, S. Lee, M.H. Lee, "Worst case communication delay of real-time industrial switched Ethernet with multiple levels," IEEE Transactions on Industrial Electronics, Vol. 53, No. 5, pp. 1669-1676, 2006.

[11] 양인석, 강선영, 이동익, "무인잠수정 제어시스템을 위한 네트워크 전송지연 및 패킷분실 보상기법," 임베디드공학회논문집, Vol. 6, No. 3, pp. 149-156, 2011.

[12] N. Correll,, N. Arechiga, A. Bolger, M. Bollini, B. Charrow, A. Clayton, F. Dominguez, K. Donahue, S. Dyar, L. Johnson, H. Liu, A. Patrikalakis, T. Robertson, J. Smith, D. Soltero, M. Tanner, L. White, D. Rus, "Building a distributed robot garden," Proceedings on IROS'09, pp. 1509-1516, 2009.

[13] K. Tindell, A. Burns, A.J. Wellings, "Calculation controller area network (CAN) message response times," Control Engineering Practice, Vol. 3, No. 8, pp. 1163-1169, 1995.

[14] CLARK Material Handling Asia, CRX 10-25 User manual, CLARK Material Handling Asia, Korea, 2006.

[15] 지규인, 임준혁, 유승환, 이달호, "무인자동차의 경로점 주행 시 장애물 회피를 위한 경로생성 알고리즘," 제어.자동화.시스템공학 논문지, Vol. 17, No. 8, pp. 843-850, 2011.

저 자 소 개

박 지 훈 (Jeehun Park)

2004년 부산대 기계공학부 학사.
2012년 부산대 기계공학부 박사.
현재, 자동차부품연구원 연구원.
관심분야: 지능형 자동차, 차량용 네트워크, Cyber-Physical System
Email: parkjh@katech.re.kr

이 석 (Suk Lee)

1984년 서울대 기계공학과 학사.
1985년 펜실바니아주립대 석사.
1990년 펜실바니아주립대 박사.
현재, 부산대학교 기계공학부 교수.
관심분야: 산업용 네트워크, 차량용 네트워크, 센서 네트워크.
Email: slee@pusan.ac.kr

이 경 창 (Kyung-Chang Lee)

1996년 부산대 생산기계공학과 학사.
1998년 부산대 생산기계공학과 석사.
2003년 부산대 지능기계공학과 박사.
현재, 부경대학교 제어계측공학과 교수.
관심분야: Cyber-Physical System, 차량용 네트워크, 로봇용 네트워크.
Email: gclee@pknu.ac.kr