차량 내 네트워크 선정에 따른 ESC 성능 분석을 위한 ECU-In-the-Loop 시뮬레이션

양 승 문¹⁾ · 김 성 엽²⁾ · 기 영 훈¹⁾ · 안 현 식^{*1)}

국민대학교 전자공학과¹⁾ · 캐시시스템 시스템사업부²⁾

ECU-In-the-Loop Simulation for ESC Performance Analysis on the Selection of In-vehicle Networks

Seung-Moon Yang¹⁾ · Seong-Yeop Kim²⁾ · Young-Hun Ki¹⁾ · Hyun-Sik Ahn^{*1)}

¹⁾Department of Electronics Engineering, Kookmin University, Seoul 136-702, Korea

²⁾R&D, CANsystem, 307-2 Sangdaewon-dong, Jungwon-gu, Seongnam-si, Gyeonggi 462-736, Korea

(Received 31 August 2012 / Revised 20 November 2012 / Accepted 4 March 2013)

Abstract : This paper shows how the performance of an ESC(Electronic Stability Control) system can be affected by the selection of in-vehicle network protocols such as CAN or FlexRay. The vehicle control performance under ESC operation is analyzed by EILS(ECU-In-the-Loop Simulation). The experimental set-up for the EILS of the ESC system consists of two 32-bit microcontroller boards communicated with CAN or FlexRay protocols. A 7-DOF vehicle model and an ESC algorithm with 2-DOF reference vehicle model are implemented on each microcontroller respectively. It is shown by experimental results that the ESC system using the FlexRay protocol can achieve better performance than that using the CAN protocol for a fast and accurate lane changing.

Key words : Electronic stability control(전자 안전성 제어), Vehicle dynamic modeling(차량 동력학 모델링), Controller Area Network(CAN, 캔), FlexRay(플렉스레이)

Nomenclature

$F_{x_{ti}}$: vehicle longitudinal force on wheel i , N
 $F_{y_{ti}}$: vehicle lateral force on wheel i , N
 I_z : moment of yaw inertia, Nm
 $I_{y_{ti}}$: moment of inertia of wheel i rotation, Nm
 K_f : cornering power on front wheel, N
 K_r : cornering power on rear wheel, N
 T_{bi} : breaking torque on wheel i , Nm
 V : velocity, m/s
 M : vehicle total mass, kg
 m : vehicle body mass, kg

l_f : distance from front axle to CoG, m
 l_r : distance from rear axle to CoG, m
 l : wheel base, m
 γ : yaw rate, rad/sec
 $\dot{\gamma}$: yaw acceleration, rad/sec²
 δ : steering angle, rad
 β : slip angle, rad
 $\dot{x}_c$: x-direction velocity, m/s
 $\ddot{x}_c$: x-direction acceleration, m/s²
 $\dot{y}_c$: y-direction velocity, m/s
 $\ddot{y}_c$: y-direction acceleration, m/s²
 ω_i : wheel angular velocity on wheel i , rad/s
 r_{ti} : wheel radius on wheel i , m

*Corresponding author, E-mail: ahs@kookmin.ac.kr

Subscripts

f	: front
r	: rear
b	: brake
i	: wheel number
z	: vertical factor
t	: tire

1. 서 론

지난 20년 이상 상업용 차량을 위한 제동 장치 기술에 관한 많은 연구가 진행되어 왔다. ABS(Anti-lock Braking System)가 휠 슬립을 제어함으로써, 제동 중에 휠이 잠기는 것을 방지하여 제동 및 조향 성능을 향상하였고, ESC(Electronic Stability Control)는 횡 방향 제어를 통해 운전자가 의도한 차량의 궤도를 유지하게 하여 미끄러짐이나 탈선 사고를 방지할 수 있게 하였다.^{1,2)}

한편, 차량의 안전과 연료 효율, 다양한 편의를 위해 차량 내 ECU(Electronic Control Unit)가 증가하게 되었다. 현대의 차량에서는 다수 ECU들의 정보 공유 및 효율적인 제어를 위해 실시간 분산 제어 시스템이 구성 되어 있다. 이에 따라 차량 내 다수의 ECU간 통신을 위하여 다양한 종류의 차량 내 네트워크(In-Vehicle Network)를 구현하고 차량의 첨단 장치들을 분산형으로 제어하는 기법에 대한 연구가 활발히 진행되고 있다.³⁾

현재 차량 내 네트워크로 가장 많이 사용되고 있는 CAN(Controller Area Network) 프로토콜은 차량 내 ECU가 증가함에 따라, 상대적으로 낮은 속도와 이벤트 기반 방식으로 인한 불확정적인 시간 지연 때문에 실시간 분산제어 시스템을 위한 네트워크에서 요구되는 특성을 만족시키기 어렵다. 이러한 CAN이 가지고 있는 통신방식의 한계를 극복하기 위해 TTP(Time Triggered Protocol), Byteflight, TTCAN(Time Triggered CAN) 및 FlexRay등 새로운 통신방식의 프로토콜들이 등장하게 되었다.^{4,5)} FlexRay는 10Mbps의 통신 속도와 넓은 대역폭, 확정적인 시간 지연 등의 장점을 가지고 있어 실시간 분산 제어 시스템에 적합한 통신 프로토콜로 인정받고 있다. FlexRay는 2006년에 BMW X5 시리즈의 Electronic

Damper Control에 최초로 적용된 이후 그 사용이 증가하고 있는 추세이다.^{6,7)}

본 논문에서는 차량 내 네트워크 환경이 ESC 시스템 성능에 어떤 영향을 미치는지 확인하기 위하여 32비트 마이크로 컨트롤러 중심의 EILS(ECU-In-the-Loop Simulation)를 이용하기로 한다. EILS는 제어기 알고리즘을 ECU 상에서 동작시키고 또 다른 마이크로 컨트롤러 보드에 차량 모델링을 포팅한 후 ECU와 이 보드를 차량 내 네트워크로 연결시켜 시뮬레이션 하는 것이다. 이러한 EILS의 적용으로 실차 적용 전 저비용으로 차량 제어기 성능을 예측할 수 있게 된다. 실제 실험을 통하여 CAN 프로토콜 및 FlexRay 프로토콜이 차량 내 네트워크에 사용된 경우 각각에 대한 ESC 시스템의 성능을 비교 검토하기로 한다.

2. 차량 모델 및 제어기 설계

본 장에서는 이 논문에서 사용된 차량 모델과 ESC 제어기의 차량 제어 알고리즘을 살펴본다.

2.1 차량 모델

본 논문에서는 ESC제어기의 알고리즘을 테스트하기 위해 7-자유도 차량 모델을 사용한다. 7-자유도 차량 모델은 Fig. 1 및 Fig. 2와 같이 차체에서 중·횡·요 운동에 대한 3-자유도와 각 휠의 회전 운동에 대한 4-자유도를 고려한다.^{8,9)}

차량의 중·횡·요 방향의 3자유도 모델은 각각 식 (1) ~ (3)과 같다.⁹⁾

$$M(\ddot{x}_c - \gamma \dot{y}_c) = \cos \delta \left(\sum F_{x1,2} \right) - \sin \delta \left(\sum F_{x1,2} \right) + \sum F_{x3,4} \quad (1)$$

$$M(\ddot{y}_c + \gamma \dot{x}_c) = \cos \delta \left(\sum F_{y1,2} \right) + \sin \delta \left(\sum F_{x1,2} \right) + \sum F_{y3,4} \quad (2)$$

$$I_z \dot{\gamma} = l_f \left(\cos \delta \left(\sum F_{y1,2} \right) - \sin \delta \left(\sum F_{x1,2} \right) \right) - l_r \sum F_{y3,4} \quad (3)$$

식 (1) ~ (3)의 운동 방정식을 식 (4) ~ (6)으로 정리할 수 있다. 타이어 모델로부터 계산된 중·횡 방향의 힘과 조향각을 통해 중·횡 방향 가속도와 요 속도를 구하여 차량의 수평 방향의 특성을 알 수 있다. 여기서 타이어 모델은 Dugoff 타이어 모델을 사용한다.

Fig. 1 7-DOF vehicle model

Fig. 2 Wheel motion of a vehicle

$$\ddot{x}_c = \left(\cos \delta \left(\sum F_{x1,2} \right) - \sin \delta \left(\sum F_{y1,2} \right) + \sum F_{x3,4} \right) / M + \gamma \dot{y}_c \quad (4)$$

$$\ddot{y}_c = \left(\cos \delta \left(\sum F_{y1,2} \right) + \sin \delta \left(\sum F_{x1,2} \right) + \sum F_{y3,4} \right) / M - \gamma \dot{x}_c \quad (5)$$

$$\dot{\gamma} = \left(l_f \left(\cos \delta \left(\sum F_{y1,2} \right) + \sin \delta \left(\sum F_{x1,2} \right) \right) - l_r \sum F_{y3,4} \right) / I_z \quad (6)$$

또한 각 휠의 운동 방정식은 식 (7) ~ (10)과 같이 표현된다. 이로부터 각 휠의 제동토크를 계산할 수 있다.

$$I_{y1} \dot{\omega}_1 = -F_{x1} \times r_{t1} - T_{b1} \quad (7)$$

$$I_{y2} \dot{\omega}_2 = -F_{x2} \times r_{t2} - T_{b2} \quad (8)$$

$$I_{y3} \dot{\omega}_3 = -F_{x3} \times r_{t3} - T_{b3} \quad (9)$$

$$I_{y4} \dot{\omega}_4 = -F_{x4} \times r_{t4} - T_{b4} \quad (10)$$

2.2 ESC 제어기 설계

2.2.1 기준 차량 모델

차량을 제어하기 위해서 기준이 되는 모델이 필요하다. ESC 제어기를 설계하기 위해 Fig. 3과 같은

Fig. 3 2-DOF bicycle model

2-자유도 자전거 모델로 참조 차량 모델을 구성한다.

2-자유도 참조 차량 모델의 기본적인 운동 방정식은 식 (11), (12)와 같이 나타낼 수 있다. 차량이 정상 원 선회를 하는 상황이라고 가정하면 차량 중심점의 슬립각과 요 속도는 일정하므로 $d\beta/dt$, $d\gamma/dt$ 는 모두 0이 된다. 이를 식 (11), (12)에 적용하면 식 (13), (14)를 구할 수 있고 최종적으로 ESC 제어를 위한 기준 요 속도를 식 (15)로 나타내었다.

$$mV \frac{d\beta}{dt} + 2(K_f + K_r)\beta + \left\{ mV + \frac{2}{V}(l_f K_f - l_r K_r) \right\} \gamma = 2K_f \delta \quad (11)$$

$$2(l_f K_f - l_r K_r)\beta + I \frac{d\gamma}{dt} + \frac{2(l_f^2 K_f + l_r K_r)}{V} \gamma = 2l_r K_f \delta \quad (12)$$

$$2(K_f + K_r)\beta + \left\{ mV + \frac{2}{V}(l_f K_f - l_r K_r) \right\} \gamma = 2K_f \delta \quad (13)$$

$$2(l_f K_f - l_r K_r)\beta + \frac{2}{V}(l_f^2 K_f + l_r K_r) \gamma = 2l_r K_f \delta \quad (14)$$

$$\gamma_{ref} = \left(\frac{1}{1 - \frac{m(l_f K_f - l_r K_r)}{2l^2 K_f K_r}} \right) \frac{V}{l} \delta \quad (15)$$

2.2.2 차량 제어 알고리즘

ESC 제어기의 로직 구조는 Fig. 4와 같다. ESC 제어기는 사용자로부터 입력 받은 조향각과 7-자유도 차량 모델로부터 출력되는 차량의 속도를 입력으로 받아, 2-자유도 참조 차량 모델로부터 참조 요 속도를 계산한다. 또한 Fig. 5와 같이 2-자유도 참조 모델

Fig. 4 Control scheme of ESC

Fig. 5 Braking position on oversteer and understeer

과 7-자유도 차량 모델의 요 속도 사이의 차이를 기준으로 각 바퀴의 제동력을 제어하도록 하였다. 7-자유도 차량 모델이 2-자유도 참조 차량 모델보다 적게 회전하는 언더스티어인 경우 선회의 안쪽 바퀴를 제동한다. 반대로 7-자유도 차량 모델이 2-자유도 참조 차량 모델보다 더 회전하는 오버스티어인 경우 선회의 바깥쪽 바퀴를 제동한다.

3. 차량 내 네트워크 프로토콜

본 장에서는 네트워크에 따른 ESC의 성능 차이를 살펴보기 위해 현재 자동차 분야에서 널리 사용되는 CAN 프로토콜과 차세대 실시간 분산 제어용 프로토콜로 주목 받고 있는 FlexRay 프로토콜의 특성을 검토한다.

3.1 Controller Area Network (CAN)

현재 차량에서 가장 많이 사용되고 있는 CAN 프

로토콜은 1980년대에 Bosch사에 의해 개발되었다. CAN 프로토콜의 전송 속도는 최대 1Mbps까지 지원하고 데이터는 메시지를 통해 전송되며, 데이터의 크기는 최대 8 바이트까지 허용된다.

CAN 프로토콜은 다중-마스터/다중-슬레이브(multi-master/multi-slave) 방식으로 각 ECU를 공통 직렬 버스로 연결한다. 또한 CSMA/CD(Carrier Sense Multiple-Access/Collision Detection) 방식으로 여러 노드에서 동시에 메시지를 전송하려 할 때, 메시지의 충돌이 발생한다면, 메시지 ID의 우선순위에 따라 메시지를 전송한다. CAN 프로토콜에는 표준 메시지 프레임(CAN 2.0A)과 확장 메시지 프레임(CAN 2.0B) 형식이 있다. 본 논문에서는 Fig. 6과 같은 표준 메시지 프레임을 사용한다.¹⁰⁾

3.2 FlexRay

FlexRay 프로토콜은 CAN 프로토콜의 상대적으로 낮은 속도와 이벤트 기반 프로토콜의 전형적인 전송 지연 문제를 해결하기 위해 FlexRay Consortium을 통하여 개발되었다.

FlexRay 프로토콜은 채널 당 최대 10Mbps의 고속 전송 속도를 지원하며, 고장 허용 기능을 위해 이중 채널을 가지고 있다. 또한 시분할 방식을 이용하여 프레임을 전송하므로 확정적인 전송 지연을 가지고 있어 실시간 분산 제어 시스템에 적합하다. FlexRay 프레임은 Fig. 7과 같이 헤더, 페이로드, 트레일러로 구성된다. 헤더는 프레임 ID, 페이로드 길이, 헤더 CRC(Cyclic Redundancy Check) 등 프레임에 관한 정보를 가진다. 페이로드에는 송신할 데이터가 기록되어 있으며, 트레일러에는 헤더와 페이로드를 통하여 계산된 CRC값이 있다. CRC 값을 이용하여 통신 에러의 위치를 추정할 수도 있다. FlexRay 프로토콜에서 사용되는 통신 사이클은 Fig. 8과 같이 정적

Fig. 6 Standard CAN message frame structure (CAN 2.0A)

Fig. 7 FlexRay protocol frame format

Fig. 8 Communication cycle structure of the FlexRay protocol

구간(Static Segment), 동적 구간(Dynamic Segment), 심볼 윈도우(Symbol Window), 네트워크 유휴 시간(Network Idle Time)으로 이루어진다.

정적 구간은 TDMA(Time Division Multiple Access) 기반의 접근 방식이 적용되며, 동일한 대역폭을 가지는 정적 슬롯(Static Slot)들로 구성되어 있다. 정적 슬롯은 매크로틱(Macrotick)으로 구성되어 있으며, 각 슬롯의 번호는 프레임 ID로 대응된다. 동적 구간은 FTDMA(Flexible Time Division Multiple Access) 기반의 접근 방식을 적용하며, 미니 슬롯(Mini slot)으로 구성된다. 미니 슬롯은 정적 슬롯과는 달리 대역폭이 가변이고 우선순위가 높은 프레임에 넓은 대역을 할당한다. 심볼 윈도우는 프로토콜에 정의된 심볼을 전송하는 구간이며, 네트워크 유휴 시간은 네트워크에서 데이터 전송이 없는 부분으로 각 노드 사이의 동기화를 위해 사용되는 구간이다.¹¹⁾

4. ESC EILS 시스템 구현 및 결과

본 장에서는 2장에서 설명한 차량 모델과 제어기를 EILS 시스템에 구현하는 과정과 그 시뮬레이션 결과에 대해 기술한다.

4.1 EILS 시스템의 구성

7-자유도 차량 모델과 2-자유도 참조 차량 모델을 이용한 ESC 제어를 각각 구현하기 위해 32비트 마이크로 컨트롤러인 Infineon사의 TC1782 Easy Board 2장을 사용한다. Infineon TC1782 Easy Board에는 Fig. 9와 같이 CAN 및 FlexRay 통신 컨트롤러가 포함되어 있어 별도의 외부 컨트롤러를 추가할 필요가 없다.

본 논문에서 구현한 EILS 시스템의 구성은 Fig. 10과 같다. N2 노드는 조향각 센서(Steering Angle Sensor: SAS)의 조향 입력을 7-자유도 차량 모델과

Fig. 9 Infineon TC1782 Easy Board

Fig. 10 Configuration of EILS system

네트워크를 통해 N1 노드의 ESC 제어기에 전송한다. 여기서, 7-자유도 차량 모델은 1ms 마다 시뮬레이션이 수행된다. ESC 제어기는 10ms 주기로 동작하며 네트워크를 통해 전달받은 조향 입력과 2-자유도 참조 차량 모델을 이용하여 기준 요 속도를 구한다. 기준 요 속도와 7-자유도 차량 모델로부터 전송받은 실제 차량의 요 속도를 비교하여 네 바퀴에 필요한 제동 토크를 계산한다. 이와 같이 N1노드의 ESC 제어기에서 계산된 제동 토크는 네트워크를 통해 N2 노드의 액추에이터(ACT)에 전달된 후 7-자유도 차량 모델에 제동명령을 인가한다. 또한, N1과 N2 이외의 노드에서 발생하는 메시지들에 의한 영향을 분석하기 위하여 동일한 통신 버스를 공유하는 N3와 N4의 더미노드를 추가한다.

4.2 EILS의 네트워크 구성

EILS에는 CAN과 FlexRay 프로토콜을 각각 적용하여, 네트워크에 따른 성능을 비교하고자 한다. ESC 제어기와 7-자유도 차량 모델 간 네트워크로 전송되는 데이터의 전송 주기는 ESC 제어 주기와 같은 10ms 로 설정한다.

4.2.1 네트워크 전송 데이터

ESC 제어기와 7-자유도 차량 모델 사이의 네트워크로 전송되는 데이터는 Table 1과 같다. 이 때 실차에서는 센서 데이터가 센서 노드에서 네트워크로 전송되지만 본 논문에서는 7-자유도 차량 모델에서 계산되어 전송된다. 전송노드에 따라 메시지를 구분하면 Table 1의 모든 데이터는 6개의 메시지로 전송할 수 있다. 또한 추가된 더미노드에서 전송되는 메시지는 Table 2와 같다.

Table 1 Data list transferred via in-vehicle network

이름	송신	수신	데이터	크기 (byte)
ESC_01	ESC	ACT	Ref. brake torque (FL)	2
	ESC	ACT	Ref. brake torque (FR)	2
	ESC	ACT	Ref. brake torque (RL)	2
	ESC	ACT	Ref. brake torque (RR)	2
SAS_01	Sensor	ESC	Steering angle	2
SPD_01	Sensor	ESC	Longitudinal velocity	2
YAW_01	Sensor	ESC	Yaw velocity	2
WHL_01	Sensor	ESC	Wheel speed (FL)	2
	Sensor	ESC	Wheel speed (FR)	2
	Sensor	ESC	Wheel speed (RL)	2
	Sensor	ESC	Wheel speed (RR)	2
ACT_01	ACT	ESC	Brake torque (FL)	2
	ACT	ESC	Brake torque (FR)	2
	ACT	ESC	Brake torque (RL)	2
	ACT	ESC	Brake torque (RR)	2

Table 2 Data list from dummy node

이름	송신	수신	데이터	크기 (byte)
DUMMY_1	N3	N4	Dummy Data1	8
DUMMY_2	N4	N3	Dummy Data2	8

4.2.2 네트워크 통신 대역폭

EILS 시스템에서는 500kbps의 CAN 프로토콜과 10Mbps의 FlexRay 프로토콜을 이용하여 분산 제어 시스템을 구축한다.

CAN 프로토콜의 경우, 8바이트 데이터를 포함한 메시지 1개가 전송되는데 필요한 최대 비트수는 bit stuffing을 고려하여 식 (16)과 같이 계산할 수 있다.¹²⁾

$$N_{CAN_MAX} = \left(\frac{34 + 8d_l}{5} + 47 + 8d_i \right) [bit] \quad (16)$$

여기서 d_t 은 데이터의 크기(바이트)이므로, CAN 프로토콜로 보내는 메시지의 크기는 131 비트가 된다. 따라서 CAN 메시지 1개가 전송되는 시간은 262us로 계산되며, Table 1의 모든 데이터가 전송되는 데 약 1.6ms가 소요 된다. 또한 최대 38개의 메시지를 10ms안에 전송 가능하다.

그러나 네트워크를 공유하는 다른 노드에서 우선 순위가 높은 38개 이상의 CAN 메시지를 10ms 주기 전송한다면, 네트워크 점유에 따른 ESC 제어기와 7-자유도 차량 모델 간 메시지 전송 지연이 발생하여, ESC 제어 성능이 예측가능하지 않게 된다. 이러한 상황을 구현하기 위하여 실제로 더미노드로부터의 메시지 개수가 10ms에 38개가 되도록 함으로써, ESC 제어를 위한 CAN 메시지에 전송지연이 발생하도록 구성한다.

CAN을 적용한 EILS와 객관적인 성능 평가가 이루어 질 수 있도록 FlexRay의 메시지 전송 주기도 CAN과 동일하게 10ms로 설정한다.

본 실험에서 구현된 FlexRay의 통신 사이클은 정적 구간, 동적 구간 및 휴지 시간으로 구성되며, 10ms의 길이를 가진다. 정적 구간에서는 메시지가 전송되는 시간이 확정적이기 때문에, ESC 제어를 위한 메시지 프레임의 정적 슬롯에 할당한다. 프레임의 길이(Frame length)는 식 (17)과 같이 계산된다.

$$\begin{aligned} FrameLength[bit] = & TSS[bit] + FSS[bit] + 80[bit] \\ & + PayloadLength \times 20[bit] + FES[bit] \end{aligned} \quad (17)$$

여기서, TSS (Transmission Start Sequence)는 6[bit], FSS (Frame Start Sequence)는 1[bit], FES (Frame End Sequence)는 2[bit]를 가진다. $PayloadLength$ 의 경우, CAN과 동일한 데이터를 사용하므로, 4 words로 설정한다. 따라서, 프레임 길이는 169[bit]이다.

$$\begin{aligned} D_T[MT] = & 2 \times gdActionPointOffset[MT] + \text{ceil}(((aFrameLengthStatic[gdBit] \\ & + cChannelIdleDelimiter[gdBit]) \times gdBitMax[\mu s / gdBit] \\ & + gdMinPropagationDelay[\mu s] + gdMaxPropagationDelay[\mu s]) \\ & / (gdMacroTick[\mu s / MT] \times (1 - cClockDeviationMax))) \end{aligned} \quad (18)$$

정적 슬롯 한 개의 길이 D_T 는 식 (18)로 나타낼 수 있다.¹¹⁾ 여기서, $gdBitMax$ 는 0.10015[μs/gdBit], $gdMacroTick$ 은 2[μs/MT], $gdActionPointoffset$ 은 4[MT]

로 설정한다. $gdMinPropagationDelay$ 와 $gdMaxPropagationDelay$ 는 0[μs]과 0.930[μs]로 설정하며, $cChannelIdleDelimiter$ 와 $cClockDeviationMax$ 는 각각 11[gdBit]와 0.0015로 FlexRay protocol specification 2.1 rev. A 표준에 정의되어 있다. 따라서, 프레임 길이가 169[bit]일 때, 정적 슬롯 한 개가 전송되는 시간은 36us가 된다. 이 때 네트워크의 물리적 전송지연은 무시하였다. 정적 슬롯의 개수가 N_S 만큼 주어졌을 때, 정적 구간에서 소요되는 시간 T_{Static} 은 식 (19)으로 계산될 수 있다.

$$T_{Static} = D_T \times N_S [\mu s] \quad (19)$$

정적 구간에서는 정적 슬롯을 통해 메시지가 전송되는 시점이 확정적이므로, ESC 제어 알고리즘과 7-자유도 차량 모델의 시뮬레이션이 처리되는 시간을 고려하여 프레임을 슬롯에 할당하여야 한다. 따라서 정적 슬롯을 25개로 구성하고 Table 1의 데이터를 전송하기 위한 프레임을 슬롯 6개에 할당하여, 나머지 19개의 슬롯에 해당되는 시간에 ESC 제어 알고리즘과 7-자유도 차량 모델의 시뮬레이션이 수행될 수 있게 구성한다.

FlexRay 경우에도 더미노드는 불특정한 시점에서 10ms 주기로 38개의 메시지를 전송한다. 따라서 더미노드가 메시지를 전송할 경우를 대비하여 프레임을 동적구간의 미니 슬롯에 할당한다. 동적 구간의 길이는 전체 10ms의 통신 사이클 중에서 정적 구간과 휴지 시간을 제외한 9ms로 설정된다. 미니 슬롯의 길이는 프레임의 페이로드 데이터 크기에 따라 가변적이기 때문에, 더미노드의 데이터 크기를 적용하여 페이로드 데이터의 크기를 최대 4word로 설정한다.

정적 슬롯과 미니슬롯은 고유한 슬롯 카운터를 가지고 있어 사전에 스케줄링한 프레임 ID와 슬롯 카운터가 일치하면, 프레임의 충돌이 없이 메시지 전송이 가능하다. 따라서, 네트워크의 대역폭이 충분하다면, FlexRay는 더미노드에서 전송되는 메시지를 포함한 44개의 메시지를 10ms에 전송할 수 있다.

4.3 CAN 및 FlexRay 기반 ESC의 성능 비교

4.3.1 EILS 환경 구성

EILS 시스템의 하드웨어 구성은 Fig. 11과 같다.

Fig. 11 The hardware configuration of EILS

Table 3 Conditions of experiments

파라미터	값
Initial vehicle velocity	80 [km/h]
Coefficient of road friction	0.4
Steering input	Fig.12
Front wheel base	0.998[m]
Rear wheel base	1.572 [m]
Vehicle Body mass	1395 [kg]
Vehicle Total mass	1635 [kg]
Coefficient of front cornering	36500 [m/s]
Coefficient of rear cornering	63000 [m/s]
Body inertia	2600 [kg·m ²]

Fig. 12 Test profile of steering input for the EILS

EILS에서는 Table 3과 같이 시뮬레이션 조건을 설정하고, 차선을 변경하는 시나리오를 수행한다. 이때 노면 마찰 계수는 0.4로 눈길과 같이 매우 미끄러운 상황을 가정하였고, 차선을 변경하기 위한 조향 입력은 Fig. 12와 같다.

4.3.2 CAN과 FlexRay의 ESC 성능 비교

CAN과 FlexRay 프로토콜에 따른 ESC 성능을 평

(a) Without dummy nodes

(b) With dummy nodes

Fig. 13 Yaw rates in ESC EILS

가하기 위하여 두 가지의 EILS를 수행한다. 먼저 ESC 제어기와 7-자유도 차량 모델만을 CAN 또는 FlexRay로 연결하여 ESC 성능 분석을 위해 요 속도, 요 각도 및 횡 변위를 비교한다. 그 다음, 더미노드를 1.5s부터 2.5s까지 네트워크에 연결하고 ESC 성능을 분석한다. EILS에서는 요 속도, 요 각도 및 횡 변위의 오차를 줄이기 위해 실제 운전자가 조향 입력을 재조정하는 상황은 고려하지 않는다.

더미노드에서 메시지 전송이 없는 경우의 기준 요 속도와 실제 요 속도를 Fig. 13(a)에 나타내었다. EILS 결과를 통해 CAN과 FlexRay의 ESC 성능 차이가 없음을 알 수 있다. Fig. 13(b)는 더미노드가 네트워크에 연결되었을 때 요 속도를 나타낸 것이다. CAN의 경우 ESC제어기와 7-자유도 모델 사이에서 메시지 전송 지연이 발생되어 요 속도가 기준과 큰 차이를 보이고 있다. 반면, FlexRay에서는 Fig. 13(a) 결과와 동일한 과형을 보이며 더미노드가 ESC 성능에 영향을 미치지 않는 것을 알 수 있다.

또한 이 경우의 요 각도와 횡 변위를 각각 Fig. 14

(a) Without dummy nodes

(b) With dummy nodes

Fig. 14 Yaw angles in ESC EILS

(a) Without dummy nodes

(b) With dummy nodes

Fig. 15 Lateral displacements in ESC EILS

와 Fig. 15에 나타내었다. 더미노드가 네트워크에 연결된 (b)의 상황에서, CAN의 경우 요 각도가 기준과 차이를 보이고 있으며, 횡 변위는 3.5s 이후로 기준으로부터 벗어나는 경향을 보인다. 반면, FlexRay의 경우, 요 각도와 횡 변위가 더미노드의 메시지 전송 여부와 관계없이 기준을 잘 추종하고 있음을 알 수 있다.

5. 결 론

본 논문에서는 분산 제어 환경에서 네트워크 성능 및 특성에 따른 ESC의 성능의 차이를 확인하기 위해 EILS를 구성하고 차선 변경 시뮬레이션을 수행하였다. ESC가 동작되는 새시 분산 제어 시스템을 구현하기 위해 ESC 제어기와 7-자유도의 차량 동역학 모델을 각각 32비트 마이크로컨트롤러 보드에 포팅하고 CAN과 FlexRay 통신을 이용해 각각 연결하였다.

또한, 동일한 통신 버스에 연결되어 있는 더미노드에서 일시적으로 데이터 전송이 발생하는 경우를 고려한 시뮬레이션을 수행하였다. CAN을 적용한 EILS의 경우 더미노드에서 전송되는 메시지로 인하여 전송 지연이 발생하여, ESC 제어 성능이 저하됨을 확인 하였다. 그러나 FlexRay의 경우 CAN 보다 대역폭이 높기 때문에 ESC 제어에 필요한 메시지를 제어 주기 안에 모두 전송하여 ESC 제어 성능에 영향이 미치지 않음을 확인하였다.

후 기

본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학IT연구센터육성 지원사업(NIPA-2013-H0301-13-2007), IT융합 고급인력과정 지원사업(NIPA-2013-H0401-13-1009) 및 미래창조과학부 산업원천기술개발사업(No.10033751-2012-13, 그린카 EMB액츄에이터 핵심 부품개발)의 지원을 받아 수행되었음.

References

- 1) E. K. Liebmenn and T. Fuehrer, "More Safety with Vehicle Stability Control," SAE 2007-01-2759, 2007.

- 2) D. Neuhaus and J. Willms, "Vehicle Dynamics-Continuous Improvements in Vehicle Safety from ABS to Electronic Stability Control," SAE 2005-26-065, 2005.
- 3) S. M. Hong, M. H. Kim, S. Lee and K. C. Lee, "Development of Time-triggered Network for the Distributed Real-time Control System," KSAE Spring Conference Proceedings, pp.1492-1497, 2006.
- 4) N. Navet, Y. Song, F. Simonot-Lion and C. Wilwert, "Trends in Automotive Communication Systems," Proceedings of the IEEE, Vol.93, No.6, pp.1204-1223, 2005.
- 5) E. G. Schmidt and K. Schmidt, "Message Scheduling for the FlexRay Protocol: The Dynamic Segment," IEEE Transactions on Vehicular Technology, Vol.58, No.5, pp.2160-2169, 2009.
- 6) I. S. Park and M. H. Sunwoo, "FlexRay Network Parameter Optimization Method for Automotive Applications," IEEE Transactions on Industrial Applications, Vol.58, No.4, pp.1449-1459, 2011.
- 7) J. Berwanger, W. Kuffner, M. Peteratzinger, G. Reichart and A. Schedl, "FlexRay - Exploitation of a Standard and Future Prospects," SAE 2006-21-0039, 2006.
- 8) C. Zhao, W. Xiang and P. Richardson, "Vehicle Lateral Control and Yaw Stability Control Through Differential Braking," IEEE International Symposium on Industrial Electronics, Vol.1, pp.384-389, 2006.
- 9) R. Rajamani, Vehicle Dynamics and Control, Springer, pp.16-256, 2006.
- 10) Robert Bosch GmbH, CAN specification Ver. 2.0, 1991.
- 11) FlexRay Consortium, FlexRay protocol specification 2.1 revision A, 2005.
- 12) K. W. Tindell, H. Hansson and A. J. Wellings, "Analysing Real-time Communications: Controller Area Network(CAN)," Proceedings of IEEE Real-time Systems Symposium, pp.259-263, 1994.