논문 2011-48SD-4-7

비트 재배열을 이용한 가변길이 CAN 메시지 압축

(Variable Length CAN Message Compression Using Bit Rearrangement)

조 경 주*

(Kyung-Ju Cho)

요 약

본 논문에서는 CAN 버스로드를 줄여 CAN 네트워크 비용을 줄이고 오류 확률을 낮추기 위해서 메시지를 압축하는 방법을 제안한다. 제안하는 비트 재배열 방법은 기존의 메시지 변화량만 전송하는 압축방식과 달리 각 변화량의 하위 비트들로부터 시작하여 전송데이터를 재구성하도록 하여 결과적으로 데이터가 압축되는 방법이다. 차량 CAN 로깅 파일을 이용하여 제안한 방법으로 압축한 결과 전송 데이터량이 기존 압축알고리즘에 비해 26% 더 감소함을 보였다.

Abstract

In this paper, we propose a CAN message compression method using bit rearrangement to reduce the CAN bus load and the error probability during the transmission of CAN messages. In conventional CAN message compression methods, message compression is accomplished by sending only the differences between the previous data and the current data. In the proposed method, the difference bits are rearranged to further increase the compression efficiency. By simulations in car applications, it is shown that the CAN transmission data is further reduced up to 26% by the proposed method, compared with the conventional method.

Keywords : CAN, 가변길이 메시지 압축, 메시지 변화량, 비트 재배열

Ⅰ. 서 론

오늘날 편의주행, 안전주행, 쾌적주행을 추구하는 소비자의 수요를 만족시키기 위해 차량에는 과거보다 훨씬 많은 전자제어장치가 장착되고 있으며, 이에 따라 효율적인 차량내 네트워크의 구현은 중요한 이슈가 되고 있다.

1980년대 후반 독일의 Bosch사에서 차량 내 전자장치들을 단일 네트워크로 연결하기 위해 고안된 CAN (Controller Area Network)은 최근에는 자동차 산업뿐만 아니라 비행기, 공장자동화, 선박 전자장치, 의료장비 등 많은 산업 분야에서 폭 넓게 사용되고 있다. 두 가닥의 꼬임선으로 구성된 CAN 버스는 1Mbps의 전송속도를 제공하며, 최대 110개의 노드를 단일 네트워크로 연결할 수 있다[1~4].

차량내 전자제어장치가 증가함에 따라 기존 CAN의 통신네트워크 대역폭으로 이를 수용하는 데 한계에 부딪히자, BMW, 모토롤라 등이 컨소시엄을 구성하여 10Mbps의 전송속도를 제공하는 FlexRay를 개발하여 2006년에 Version 2.1이 발표되었으며, 국내에서도 FlexRay 프로토콜이 구현되었다[5~6]. 또한, 차량내 대용량의 멀티미디어 정보를 전달하기 위해 광케이블을 사용하는 MOST(Media Oriented System Transport)가 개발이 진행되고 있다[7]. 그러나 FlexRay와 MOST는 복잡한 인터페이스 및 로열티 문제 등으로 값비싼 대가를 치러야 하지만 CAN은 저가하여 여전히 자동차뿐만 아니라 다양한 산업분야에서 널리 쓰이고 있다.

최근 차량에 연결된 센서의 개수가 증가하면서 CAN 통신 버스로드가 크게 증가하고 있다. 버스 로드에 과부하가 걸리면 CAN 프로세서가 중단되거나 메시지 에러가 발생할 수 있다. 특히, 우선순위가 낮은 데이터는

* 정회원, 항로표지기술협회
(Korea Association of Aids to Navigation)
접수일자: 2011년3월2일, 수정완료일: 2011년4월15일

전송되기 어렵고 전송되지 못한 메시지가 버퍼에 쌓이다가 송신 에러를 발생시킬 수 있다. 또한 버스로드가 클 때 CAN 데이터 비트 오류 발생 확률도 증가 한다[3].

CAN 통신량이 많은 경우 이러한 위험성을 방지하기 위해 CAN 채널을 더 개설하여 버스 사용량을 줄여야 한다. CAN 채널을 더 개설하는 대신 메시지의 변화량만 전송함으로서 CAN 데이터를 압축하여 버스로드를 감소시킬 수도 있다[8]. 하지만, 이 경우 신호의 최대 변화량을 정확히 예측하여야 하며 만약 예측이 틀리면 오류가 발생한다. 또한 매번 정해진 변화량 비트를 전송하므로 압축효율이 떨어진다.

본 논문에서는 CAN 메시지 비트 재배열 압축 방식을 이용하여 최대 변화량을 고려하지 않고 압축률을 향상시키는 방법을 제안한다. Ⅱ 장에서는 기존의 CAN 데이터 압축 알고리즘을 살펴보고, Ⅲ 장에서는 새로운 압축방법을 제안한다. Ⅳ 장에서는 제안한 알고리즘의 성능을 분석하며, Ⅴ 장에서 결론을 맺는다.

Ⅱ. 기존의 CAN 통신 데이터 압축 알고리즘

1. CAN 소개

단일 직렬버스에 연결된 여러 전자장치들간에 데이터를 전송하기 위해서는 버스사용권을 획득하여야 하는데 CAN은 CSMA/CR(carrier sensing multiple access /collision resolution)을 통해 다중충돌을 조정한다. 2개 이상의 노드가 메시지가 전송하려 할 경우 식별자(Indentifier)를 통해 메시지 충돌을 중재한다.

CAN은 CAN 트랜시버와 CAN 컨트롤러로 구성된다. CAN 트랜시버는 버스에 직접 연결되는 물리계층 소자로 수신된 신호를 CAN 컨트롤러에 전달하고, CAN 컨트롤러에서 전달받은 신호를 물리계층의 조건에 맞게 변환한 후 버스에 실어 노드에 전송한다. CAN 컨트롤러는 CAN 트랜시버에서 받은 신호를 이용하여 동기를 맞춘후 데이터 포맷에 따른 프로토콜 분석절차에 의해 버스사용권 획득여부를 결정하고 host 프로세서와 연동하여 데이터를 전송한다.

11비트 식별자를 갖는 표준 CAN 데이터 프레임은 그림 1과 같이 구성된다[4]. 그림 1에서 Data 필드는 64비트로 가장 큰 부분을 차지하며 Stuffing 규칙에 의해 같은 비트를 5번 이상 사용할 수 없도록 메시지를 재구성하면 Data 필드는 더욱 확장된다. Data 필드는 최대 8바이트로 구성되며 실제 전송 데이터 바이트 수는 그

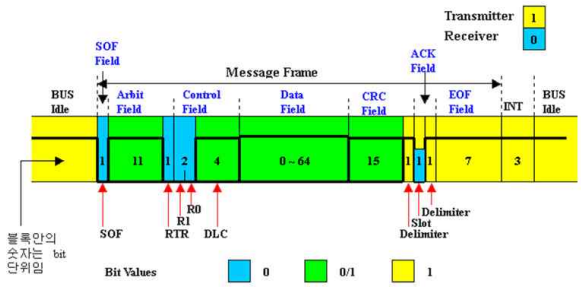


그림 1. 표준 CAN 메시지 구조
Fig. 1. Structure of standard CAN message.

표 1. CAN 데이터 메모리 맵
Table 1. Memory map of CAN data.

구 분	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
byte 0	7	6	5	4	3	2	1	0
byte 1	15	14	13	12	11	10	9	8
byte 2	23	22	21	20	19	18	17	16
byte 3	31	30	29	28	27	26	25	24
byte 4	39	38	37	36	35	34	33	32
byte 5	47	46	45	44	43	42	41	40
byte 6	55	54	53	52	51	50	49	48
byte 7	63	62	61	60	59	58	57	56

림 1의 DLC에서 정의된다. 만약 DLC가 0이면 Data 필드는 전송되지 않는다.

CAN 메시지를 2차원 메모리로 간주할 때 메모리를 byte 0~7과 각각의 비트로 구분하여 표 1과 같이 총 64비트로 표현가능하다.

2. 변화량을 이용한 CAN 데이터 압축

차량에서 사용하는 CAN 통신 데이터 전송은 사람의 차량조작 속도나 주위환경 변화에 비해 훨씬 빠른 속도로 수행된다. 예를 들어, 차량 속도관련 데이터는 10ms마다 매번 전송되는데 비하여 실제 차량 속도는 이보다 느리게 변한다. 따라서 바로 이전 프레임과 현재 프레임의 데이터변화량은 크게 차이가 나지 않으며 이러한 특징을 이용하여 전송메시지를 압축할 수 있다[3].

64비트의 Data 필트가 각각 16비트를 가지는 4개의 신호 {sig_A, sig_B, sig_C, sig_D}로 구성된다고 하자. 한 신호의 변화량을 8비트로 표현하며 상위 7비트는 변화량의 크기, 최하위 비트는 부호 비트로 정의하면 변화량의 표현 범위는 ±127이다.

만약 이전 프레임의 {sig_A, sig_B, sig_C, sig_D}가 {12, 23, 34, 56}이었고 현재 프레임에는 {11, 25, 30, 58}로 변하였다면 차이 값은 표 2와 같이 표현 가능하다.

표 1의 CAN 데이터 메모리 맵을 이용하여 표 2의

표 2. 전송할 데이터의 차이 값
(부호 비트는 LSB에 표시)
Table 2. Difference values to send (LSB: sign bit).

신호	sig_A[7-0]	sig_B[7-0]	sig_C[7-0]	sig_D[7-0]
이전 신호 값	12	23	34	56
현재 신호 값	11	25	30	58
차이 값	-1	2	-4	2
차이 값 (2진수)	00000011	00000100	00001001	00000100

표 3. 변화량을 이용한 CAN 데이터 압축 메모리 맵
Table 3. CAN data memory map of difference-based compression.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
byte 0	sig_A[7]	sig_A[6]	sig_A[5]	sig_A[4]	sig_A[3]	sig_A[2]	sig_A[1]	sig_A[0]
byte 1	sig_B[7]	sig_B[6]	sig_B[5]	sig_B[4]	sig_B[3]	sig_B[2]	sig_B[1]	sig_B[0]
byte 2	sig_C[7]	sig_C[6]	sig_C[5]	sig_C[4]	sig_C[3]	sig_C[2]	sig_C[1]	sig_C[0]
byte 3	sig_D[7]	sig_D[6]	sig_D[5]	sig_D[4]	sig_D[3]	sig_A[2]	sig_D[1]	sig_D[0]

차이 값(2진수표현)을 전송하려면 표 3과 같이 byte 0에서부터 byte 3까지 4 byte의 데이터만 전송하면 되므로 전송 데이터는 기존의 8바이트에서 4바이트로 50% 감소되었음을 알 수 있다.

변화량을 이용한 CAN 데이터 압축알고리즘은 최대 차이 값을 예측하여 변화량 비트를 설정하고 압축하는 방식으로서 최대 변화량을 잘못 예측하여 최대 변화량이 예측한 값보다 크면 데이터의 오류가 발생한다. 반면, 최대 변화량을 너무 크게 설정하면 압축효율이 감소되므로 모든 환경을 고려하여 최대 변화량 비트가 할당되어야한다.

3. 능동형 CAN 데이터 압축 방법

변화량을 이용한 데이터 압축방법에서는 이전신호와 현재신호가 같은 경우 0을 전송한다. 데이터의 변화량이 0인 경우의 발생빈도가 높다면 보다 효율적인 압축을 위해 압축 여부 헤더 1바이트를 사용하여 변화량이 0인 경우 압축데이터 자체를 보내지 않도록 메시지를 구성할 수 있다[8].

만약 {sig_A, sig_B, sig_C, sig_D}의 신호 변화량이 {-1, 1, 0, 4}이면 헤더는 1101을 전송하고 실제 변화된 값 {-1, 1, 4}만 전송한다. 데이터가 전송되면 수신부에서는 헤더의 1101을 분석하여 이전 값 sig_C는 그대로 두고 sig_A, sig_B, sig_D에 수신된 값 {-1, 1, 4}을 더하여 각각의 신호값을 복원한다.

이러한 방식은 차이 값이 변하지 않는 경우 데이터가 전송되지 않도록 하므로 경우에 따라 버스로드가 감소하지만 헤더를 1바이트 사용하는 오버헤드가 발생하므로 최악의 상황에서는 효율이 감소할 수도 있으며 Data 필트 8바이트를 모두 사용하는 경우 헤더 바이트를 사용할 수 없게 되는 경우도 발생한다. 표 4는 헤더 1바이트와 차이 값이 각각 8비트로 설정된 4개의 신호에 대한 데이터 메모리 맵을 보여준다.

표 4. 헤더를 이용한 압축방식의 데이터 메모리 맵
Table 4. Memory map of header-based compression.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
byte 0	Header							
byte 1	sig_A[7]	sig_A[6]	sig_A[5]	sig_A[4]	sig_A[3]	sig_A[2]	sig_A[1]	sig_A[0]
byte 2	sig_B[7]	sig_B[6]	sig_B[5]	sig_B[4]	sig_B[3]	sig_B[2]	sig_B[1]	sig_B[0]
byte 3	sig_C[7]	sig_C[6]	sig_C[5]	sig_C[4]	sig_C[3]	sig_C[2]	sig_C[1]	sig_C[0]
byte 4	sig_D[7]	sig_D[6]	sig_D[5]	sig_D[4]	sig_D[3]	sig_D[2]	sig_D[1]	sig_D[0]

기존의 CAN 압축 알고리즘들은 최대 변화량 예측이 난해한 경우 데이터 오류가 발생할 수 있으며, 이를 피하기 위해서는 최대 변화량을 확장하여야하므로 압축효율이 감소하게 되는 단점이 있다. 또한, 작은 차이 값 전송에도 이미 할당된 변화량 비트를 모두 사용해야 하는 비효율성이 발생하게 되므로 이러한 문제를 개선하기 위해 다음 장에서 메시지 재배열을 이용한 압축방법을 제안한다.

Ⅲ. 제안한 가변길이 CAN 압축/복원

1. 비트 재배열을 이용한 가변길이 데이터 압축

4개의 신호 {sig_A, sig_B, sig_C, sig_D}가 각각 16비트로 구성되어 있고 표 5와 같이 차이 값이 {-1, 2, -4, 2}라고 가정하자. 압축을 위해서는 먼저 다음과 같은 절차를 통해 압축데이터 구성영역을 결정한다.

압축데이터 구성영역 결정절차

1. 각 데이터 차이 값을 원 신호와 같은 비트 수를 할당하여 표시한다.
2. MSB 열부터 시작하여 한 열의 해당 비트들이 모두 0 인 경우 해당 열을 제거한다.
3. 처음으로 제거되지 않고 남은 열부터 시작하여 LSB까지의 열들을 압축데이터 구성영역으로 결정한다.

표 6은 표 5의 차이 값에 대한 압축데이터 구성영역 결정 결과를 보인다. 열 15에서 열 4는 모든 비트들이 0이므로 제거되고 열 3에서 열 0이 압축데이터 구성영역으로 결정된다.

표 5. 전송할 데이터의 차이 값(부호 비트는 LSB에 표시)
Table 5. Difference values to send (LSB: sign bit).

신호	이전 신호 값	현재 신호 값	차이 값 10진수표현	차이 값 2진수표현
sig_A	12	11	-1	0000000000000011
sig_B	23	25	2	0000000000000100
sig_C	34	30	-4	0000000000001001
sig_D	56	58	2	0000000000000100

표 6. 표 5의 데이터에 대한 압축데이터 구성영역 결정
Table 6. Compression region for the data in Table 5.

비트 / 차이값	15	…	5	4	3	2	1	0
sig_A	0	…	0	0	0	0	1	1
sig_B	0	…	0	0	0	1	0	0
sig_C	0	…	0	0	1	0	0	1
sig_D	0	…	0	0	0	1	0	0

표 7. 제안한 압축 방식의 데이터 메모리 맵 구조
Table 7. Data memory map structure of proposed compression method.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
byte 0	sig_D[1]	sig_C[1]	sig_B[1]	sig_A[1]	sig_D[0]	sig_C[0]	sig_B[0]	sig_A[0]
byte 1	sig_D[3]	sig_C[3]	sig_B[3]	sig_A[3]	sig_D[2]	sig_C[2]	sig_B[2]	sig_A[2]

표 8. 표 6의 데이터에 대한 메모리 맵 구성
Table 8. Memory map for the data in Table 6.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
byte0	0	0	0	1	0	1	0	1
byte1	0	1	0	0	1	0	1	0

표 7은 제안한 방식을 적용한 데이터 메모리맵을 나타낸다. 표 6의 압축데이터 구성영역에서 sig_A~D의 LSB 열부터 세로 방향으로 데이터의 값을 표 1의 메모리맵의 인덱스를 0에서부터 증가시키며 값을 채워 넣는다. 표 6의 압축데이터 구성영역이 16비트로 표시되므로 표 7에서 데이터 메모리 맵도 2바이트, 즉 16비트로 구성하였으며 압축데이터 구성영역의 비트 수에 따라 데이터 메모리맵의 크기도 변화하게 된다.

표 8은 표 6의 데이터에 대한 메모리맵 구성 결과이다. 압축하기 전의 8바이트 신호와 비교해 볼 때 데이터 양이 75% 감소되었음을 알 수 있다. 변화량의 최대값을 표현하기 위해 8비트가 필요하다고 가정하면 변화량을 이용한 압축방법에서는 4바이트를 전송해야하므로

표 9. 신호의 워드길이가 다를 때 표 1의 메모리 맵에 값을 채워 넣는 순서
Table 9. Memory map mapping order for signals with different word sizes.

비트 / 차이값	15~8	7	6	5	4	3	2	1	0
sig_A	**00000000**	0	0	0	0	0	0	1	1
sig_B	**00000000**	0	0	0	0	0	1	0	0
sig_C	**skip**	0	0	0	0	1	0	0	1
sig_D	**skip**	0	0	0	0	0	1	0	0

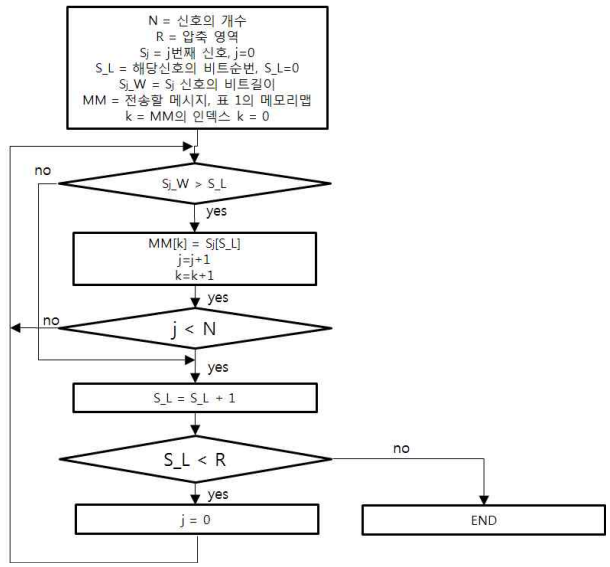


그림 2. 제안한 CAN 메시지 압축 순서도
Fig. 2. The flowchart of the proposed compression method.

압축효율이 50%인데 비해 헤더를 이용한 방식에서는 변화량이 0인 경우가 발생하지 않았다면 5바이트(헤더 1바이트, 데이터 4바이트)를 전송하여야 하므로 압축효율이 37.5%이다. 따라서, 기존의 방법에 비해 25%의 추가적인 압축효율을 얻을 수 있다.

제안한 방식에서는 변화량의 최대값과 무관하게 압축을 수행하므로 기존의 방법에서 발생하는 최대변화량의 불환전한 예측으로 인해 발생하는 오류나 압축효율 감소 현상을 피할 수 있다.

만약 sig_A와 sig_B는 2바이트, sig_C와 sig_D는 1바이트와 같이 서로 다른 워드길이의 신호들을 압축하는 경우 표 9와 같이 신호들을 배열하고 상위 비트 부분에 존재하지 않는 비트는 skip하면서 LSB의 신호들부터 표 1에 채워 넣으면 보낼 메시지가 구성된다. 전송될 구성영역이 확정된 후 메모리 맵에 데이터가 재구성 되는 순서를 정리하면 그림 2와 같다.

2. 가변길이 압축 데이터의 복원

수신부에서 수신한 ID로부터 데이터의 개수와 종류를 알고 있다면 {sig_A, sig_B, sig_C, sig_D} 4 신호에 해당하는 비트들을 순차적으로 sig_A~D의 LSB부터 채워 넣는다. 각 신호에 해당하는 차이 값을 구한 후 이전 프레임의 신호 값에 더하여 메시지를 복구한다.

그림 3은 수신한 메시지를 복원하기 위해 차이 값을 구하는 과정을 나타낸다.

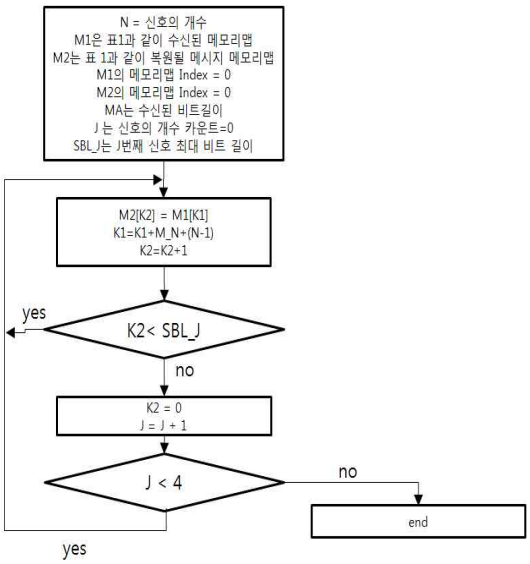


그림 3. 제안한 CAN 메시지 복원 순서도
Fig. 3. The flowchart of proposed recovery method.

Ⅳ. 가변길이 압축/복원 알고리즘 성능분석

표 10은 차량에서 사용되는 고속 CAN 신호 중 EMS (Engine Management System)에 관련된 신호의 예이다. 64비트 중 56비트는 1바이트 또는 2바이트 신호이고 나머지 8비트는 1비트 또는 2비트인 신호이다. 1비트 또는 2비트는 그룹지어 8비트 신호로 간주한다. Matlab을 이용하여 차량 CAN 신호 통신환경 및 EMS 신호에 대한 압축 알고리즘을 구현하고 압축효율을 계산하였다.

표 11은 제안한 방법과 변화량을 이용한 기존 방법에 의한 압축효율 비교 결과를 나타낸다. 제안한 방법에 의해 26%의 추가적인 압축효율을 얻을 수 있음을 보인다.

참고문헌 [8]의 능동형 데이터 압축 알고리즘은 6개의 전자제어장치로 구성된 네트워크에 적용하여 시뮬레이션한 결과, 18.9%의 압축률을 얻었다. 그러나 EMS 응용은 데이터 필드 8바이트를 모두 사용하는 경우로, 헤더 바이트에 대한 여유가 없어 능동형 데이터 압축방법을 적용할 수 없으므로. 비교대상에서 제외하였다.

그림 4는 표 10에서 사용된 EMS 신호를 제안한 방식으로 압축하고 복원한 결과의 예이며 압축처리 후 신호가 동일함을 보여준다.

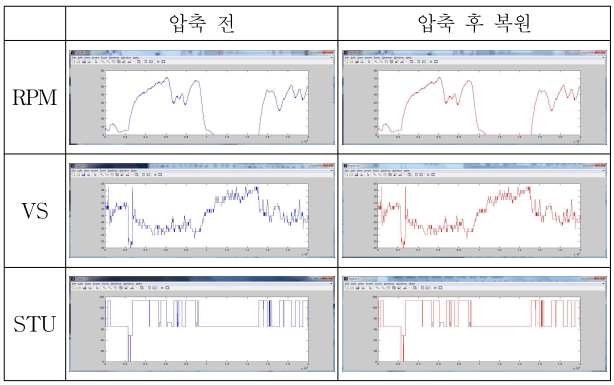
표 10 EMS의 CAN 신호
Table 10. EMS CAN signals

신호	신호 설명	비트길이
SWI_IGK	key on 상태	1
F_N_ENG	engine speed signal이 error상태	1
ACK_TCS	TCS(Traction Control System) 상태	1
PUC_STAT	연료 차단 상태	1
TQ_COR_STAT	토크 조정 상태인지	1
RLY_AC	에이콘 압력기 동작 상태	1
F_SUB_TQI	MEF(Mass Air Flow) 이상상태	1
CT	현재 토크 값	8
RPM	RPM 속도 값	16
IET	엔진 토크 지시 값	8
FT	토크 저항값	8
VS	차량 속도 값	8
R_T_S	표준 토크 비율 값	8

표 11. 전송 효율 비교
Table 11. Comparison of the compression efficiency.

구 분	전송비트	압축률
원본 메세지	1280000	1.00
변화량 이용 압축방식	720000	0.44
제안한 압축방식	378332	0.70

그림 4. CAN 신호를 제안한 압축방식으로 압축 후 복원한 결과 예
Fig. 4. The recovered CAN signals by the proposed method.



V. 결 론

본 논문에서는 메시지의 변화량을 나타내는 비트들을 재구성하여 최대변화량을 지정하지 않는 가변길이 CAN 메시지 압축 알고리즘을 제안하였다. 제안한 방법에서는 기존의 알고리즘과 달리 차이 값을 저장하기 위한 최대압축 데이터 길이를 설정하지 않아도 되기 때문에 부정확한 설정에서 발생하는 오류나 지나친 설정에서 발생하는 압축효율 저하를 피할 수 있다. 차량에 적용한 시뮬레이션 결과 기존 알고리즘에 비하여 26%의 추가적인 압축효율을 얻을 수 있음을 보였다.

본 논문에서 제안한 방식을 자동차의 전자제어장치 간 메시지 송수신에 적용하면 메시지 전송률, 버스로드, 메시지 평균길이를 상당히 감소시킬 수 있으며, 통신트래픽을 최소화하여 기존 CAN 통신 네트워크의 사용량을 극대화시켜 네트워크 증설문제를 최소화할 수 있다.

압축효율을 더욱 향상시키기 위한 신호의 배열 등 신호의 특성에 기반한 압축알고리즘의 개발은 추후 연구해야 될 과제이다.

참 고 문 헌

[1] 임명섭, “차량 통신 네트워크 기술,” 한국통신학회지, 제24권 제9호, 86-95쪽, 2007년 9월.

[2] S. Channon and P. Miller, “The requirement of future in-vehicle networks and an example implementation, ” SAE paper 2004-01-0206, Mar. 2004.

[3] Wolfhard Lawrenz, “CAN system Engineering: From theory to practical applications,” Sringer

[4] Bosch, “CAN specification 2.0,” Robert Bosch GmbH, 1991.

[5] FlexRay Communications System Electrical Physical Layer Specification Version 2.1 Revision B. 2006.

[6] 강현수, 허일남, 김용은, 정진균, “FlexRay 프로토콜 설계 및 로봇 시스템 응용,” 전자공학회논문지 제45권 TC편 제6호, 440-446쪽, 2008년 6월.

[7] MOST Homepage, http://www.mostcooperation.com

[8] 손영욱, 문희석, 정재일, 이수영, “능동형 CAN 통신데이터 압축 알고리즘,” 한국 자동차 공학회 2006년도 춘계 학술대회 논문집, pp. 1427-1477, 2006.

저 자 소 개

조 경 주(정회원)

2000년 원광대학교 전자공학과 학사 졸업

2002년 전북대학교 정보통신학과 석사 졸업

2006년 전북대학교 정보통신공학과 박사 졸업

2006년～2009년 2월 전북대학교 Post-Doc.

2009년～현재 항로표지기술협회 연구개발팀 과장

<주관심 분야 : LSI 신호처리, 저전력 회로설계, SoC 설계, 임베디드시스템 설계>