효율적인 리프로그래밍을 위한 데이터 전송 방식 제안

이정주[1], 김종훈[1], 도영수[1], 전재욱[1]
[1]성균관대학교 정보통신대학
cooooooomet@skku.edu, jhkim2022@g.skku.edu, cok2529@g.skku.edu,
jwjeon@yurim.skku.ac.kr

Proposal of data transmission method for efficient reprogramming

Jung-Ju Lee[1], Jong-Hun Kim[1], Young-Soo Do[1] Jae-Wook Jeon[1]
[1]Dept. of Electronic and Electrical Engineering, Sungkyunkwan University

요 약

자동차의 첨단화·다기능화에 따라 내장되는 프로그램이 점차 복잡해지고 ECU 개수가 증가하고 있다. 이로 인해 리프로그래밍 횟수가 늘어나고, 시간과 비용 또한 증가하였다. 현재 리프로그래밍 시간을 줄이는 다양한 연구가 진행되고 있으며, 앞으로 더 중요해질 것으로 보인다. 사람의 목숨과 직접 연결되는 자동차의 특성을 고려하면, 데이터의 변형 여부 파악도 빼놓을 수 없는 중요한 문제이다. 이 논문에서는 데이터의 빠른 전송에 초점을 맞춘 데이터 포맷과 빠른 전송 및 기존 데이터 전송 포맷의 장점을 살린 새로운 데이터 전송 포맷을 제안하였다.

1. 서론

자동차는 Advanced Driver Assistance System (ADAS) 등의 자율주행 기술 적용 및 인포테인먼트의 다양화를 통해 과거 기계 중심의 단순 이동수단에서 전자, 소프트웨어 중심의 스마트카로 변화해나가고 있다. 자동차의 기능이 다양해짐에 따라 카메라, 라이다(LiDAR), 레이더(RADAR) 등의 많은 종류의 센서가 사용되었고, 프로그램의 복잡성과 크기 및 내장되는 Electronic Control Units (ECUs) 개수가 급격하게 증가하였다. 이에 따라, 차량 소프트웨어 업데이트의 횟수가 늘어나고 더불어 시간과 비용 또한 증가하였다.[1] 현재 자동차 산업에서는 차량 소프트웨어의 업데이트 시간을 단축하여 효율성을 올리는 다양한 연구가 진행 중이며, 자율주행 등 고차원의 복잡한 통신 기술을 요구하는 미래형 자동차에서는 이를 줄이는 것이 더욱 중요해질 것으로 보인다.[2] 단순히 데이터를 빠르게 보내는 것도 중요하지만, 사람의 목숨과 직접 연결되는 자동차의 특성을 생각하면 데이터의 훼손 및 변형에 대해 보수적으로 생각해야 한다. 본 논문에서는 자동차 ECU 소프트웨어 업데이트 방법 및 데이터 전송 포맷에 관하여 설명하고, 차량 업데이트 시간을 줄일 수 있는 새로운 데이터 전송 포맷을 제안하고자 한다.

2. 배경

2.1 In-Vehicle Network(IVN)

IVN이란. 차량 내에서 제어, 정보 및 엔터테인먼트를 제공하기 위해 다양한 ECU들을 서로 연결하고 제어할 수 있는 차량 내부 네트워크를 의미한다. 차량 내 ECU는 서로 다른 도메인 간의 원활한 정보 교환을 위한 중앙 게이트웨이를 기준으로 파워트레인, 샤시, 바디, 멀티미디어 및 ADAS 도메인이 연결되어 있다. 각 도메인을 연결하기 위해, Controller Area Network(CAN), Local Interconnection Network(LIN), Ethernet과 같은 네트워크가 사용된다. 중앙 게이트웨이 구조에서는 서로 다른 IVN 사이의 통신을 위해서 게이트웨이의 사용이 필수적이다. 게이트웨이란, 서로 다른 네트워크 간의 통신을 가능하게 하고 네트워크 효율을 올려주는 하드웨어나 소프트웨어를 의미한다.

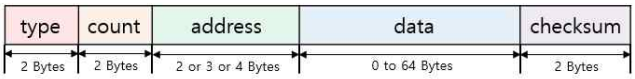
2.2 Re-Programming

리프로그래밍(Re-programming)이란, 제어기의 펌웨어를 직접 수정하지 않고 통신을 통해 소프트웨어를 간접적으로 업데이트하는 방법이다. 자동차의 제어기가 차량 내부에 존재하고, 보안 및 방수 등의 이유로 ECU 플래시 메모리의 프로그램을 업데이트

하기 위해서는 리프로그래밍이 필수적으로 요구된다. 차량 리프로그램시, 외부 진단기는 CAN 통신을 기반으로 On Board Diagnostic(OBD) 단자를 이용하여 업데이트할 소프트웨어를 중앙 게이트웨이의 진단 채널로 전송하고, 게이트웨이는 업데이트할 ECU가 있는 도메인 네트워크로 메시지를 전달한다.

2.3 Motorola S19 Format

Motorola에서 개발한 파일 포맷인 Motorola S19는 플래시 메모리, EPROM, EEPROM 등의 프로그래밍이 가능한 논리 장치에 흔히 사용한다. S19 Motorola의 형태는 그림 1과 같다.

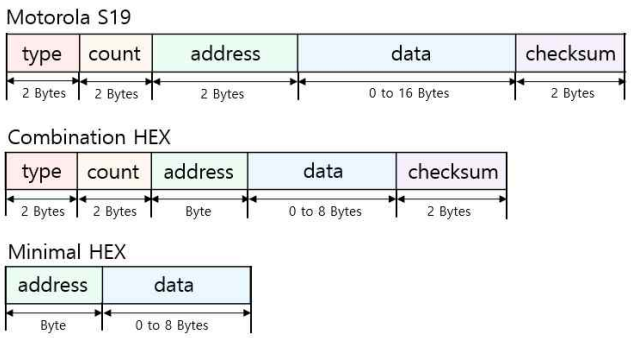


(그림 1) Motorola S19 포맷

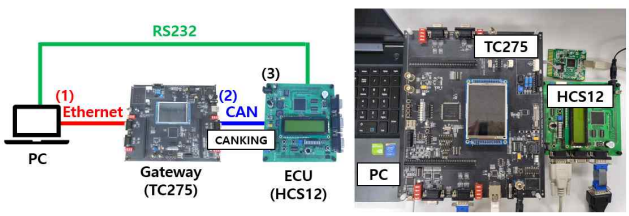
S19 Motorola 포맷의 모든 Byte는 16진수의 ASCII로 이루어져 있다. 2개의 Byte로 이루어진 타입 영역(type)은 프레임의 시작을 알리는 'S'(0x53)와 프레임의 역할 및 주소 길이를 알리는 Record type으로 이루어져 있다. 이때, Record type이 '1', '2', '3'일 때는 데이터를 전송하는 프레임을 '7', '8', '9'일 때는 종료 프레임을 '0'일 때는 시작 프레임을 의미한다. 2개의 Byte로 이루어진 카운트 영역(count)은 주소(address)부터 checksum까지의 16진수 쌍의 수를 나타낸다. 주소 영역(address)은 데이터를 저장하고자 하는 메모리 주소를 나타낸다. Record type이 '1', '9'일 때는 16bit, '2', '8'일 때는 24bit, '3', '7' 일 때는 32bit 주소를 나타낸다. 데이터 영역(data)은 메모리에 저장하고자 하는 데이터를 나타낸다. 마지막으로, 2개의 byte로 이루어진 checksum 영역을 통해 데이터가 올바르게 전송되었는지 확인할 수 있다. S19 파일 형식은 프레임의 시작 위치와 역할, 전체 byte 수 등에 대한 정보를 제공하고 checksum을 통해 전송된 데이터의 변형 여부를 확인할 수 있어 데이터의 가독성이 좋고 변형 여부를 쉽게 파악할 수 있다는 장점이 있다.

3. 데이터 전송 방법

Motorola S19 포맷에서 주소 영역과 데이터 영역은 ASCII로 나타내는데, 이를 ASCII 값 그대로 각각의 bit라고 생각하여 데이터를 전송하면 데이터를 절반가량으로 압축할 수 있다. 예를 들어, 14라는 데이터를 전송하고 싶을 때, 기존 S19 포맷을 이용하면 1(0x31), 4(0x34)의 2개의 byte를 사용하였지만, 이를 0x14로 전송하면 1개의 byte를 사용하여 데이터 송수신이 가능하다. 추가로, 데이터를 저장할 때 주소 영역이 HEX로 되어있으면, 주소 정보에 대한 별도의 변환 과정 없이 플래시 메모리에 데이터를 바로 저장할 수 있어 ASCII로 전송했을 때와 비교하면 처리시간을 줄일 수 있다는 장점이 있다.[3] Motorola S19 포맷은 프레임에 관한 정보를 추가로 전송하기 때문에 주소와 데이터만을 전송했을 때보다 더 많은 시간이 걸린다는 단점이 있지만, 데이터의 가독성이 좋고 데이터 변형 여부 파악이 쉽다는 장점이 있다. 따라서, 이 논문에서는 플래시 메모리에 데이터를 저장하기 위한 최소한의 정보인 주소와 데이터만을 HEX 형태로 전송하는 Minimal HEX 포맷과 HEX와 기존 S19 포맷의 장점을 융합한 Combination HEX 포맷을 제안하고자 한다. 기존의 데이터 송수신 파일 포맷(Motorola S19)과 새로 제안하는 파일 포맷(Combination HEX, Minimal HEX)의 프레임을 다음과 같이 그림 2로 나타내었다.



(그림 2) Motorola S19, Combination HEX, Minimal HEX 포맷



(그림 3) 실험 구성 (좌) 네트워크 구성도, (우) 실험을 위한 실제 네트워크 환경

네트워크 구성과 실제 네트워크 환경을 그림 3과 같이 나타내었다. 리프로그래밍을 구현하기 위해 외부 진단기는 PC, 중앙 게이트웨이는 TC275, ECU는 HCS12를 사용하였고, PC와 TC275는 Ethernet,

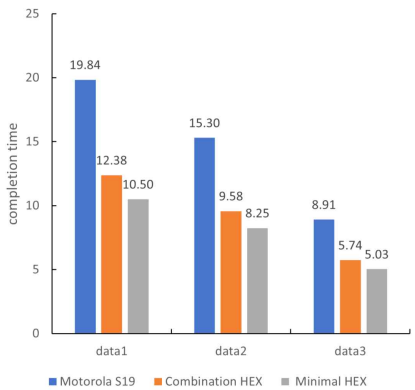
TC275와 HCS12는 CAN을 통해 연결하였다. 서로 다른 세 종류의 프로그램을 전송하여 CAN 프레임 개수와 걸린 시간을 비교하여 Motorola S19, Combination HEX, Minimal HEX 포맷의 성능을 비교하였다. 실험은 다음과 같은 순서로 진행하였다. (1) PC에서 Ethernet을 통해 TC275로 프로그램을 전송한다. (2) TC275에서는 들어온 프로그램을 CAN 데이터 전송 형식에 맞춰 8byte씩 나눈 후, 이를 CAN을 통해 HCS12로 전송한다. (3) 모든 프로그램이 HCS12에 들어왔을 때, HCS12의 내장된 플래시 메모리에 프로그램을 저장한 후, 알맞게 저장되었으면 TC275로 CAN 메시지를 전송한다. CANking을 통해 CAN 데이터 송수신 시 CAN 프레임 개수를 확인하였고, TC275에서 HCS12로 CAN 메시지를 보내기 시작한 시간과 HCS12에 데이터가 저장된 시간을 확인하여 전송 시간을 계산하였다.

4. 실험 결과

서로 다른 세 종류의 프로그램을 Motorola S19, Combination HEX, Minimal HEX 포맷으로 보냈을 때의 CAN 프레임의 개수와 걸린 시간을 비교하였다. CAN 프레임의 개수는 표 1에, 걸린 시간은 그림 3에 막대그래프 형태로 기록하였다. 실험 결과에 따르면, 동일 프로그램에 대한 전송 시간은 Minimal HEX 포맷이 가장 적게 걸렸고, Combination HEX, Motorola S19 포맷 순으로 오래 걸렸다.

<표 1> Motorola S19, Combination HEX, Minimal HEX의 CAN 프레임 개수

	data1	data2	data3
Motorola S19	184	78	140
Combination HEX	111	47	84
Minimal HEX	78	26	57



(그림 4) Motorola S19, Combination HEX, Minimal HEX의 전송 시간

각각의 프로그램을 전송했을 때, CAN 프레임의 개수는 Motorola S19 포맷이 가장 많았고, Combination HEX, Minimal HEX 포맷 순으로 적은 값을 가지는 것을 확인하였다.

5. 결론

본 논문에서는 차량 리프로그래밍 분야에서 데이터의 빠른 전송 요구에 따라 주소와 데이터만을 전송하는 Minimal HEX 포맷과 Motorola S19와 HEX의 장점을 융합한 Combination HEX 포맷을 제안하였다. 분석 결과를 통해 새로 제안한 데이터 전송 포맷들은 Motorola S19와 비교하면 약 40%의 CAN 프레임 개수 압축과 전송 시간 단축을 얻을 수 있었다. 이 논문에서는 단순히 전송 속도에만 초점을 맞춘 실험을 진행하였기 때문에, 추후 연구에서 데이터 전송 포맷의 신뢰성을 입증하는 테스트가 필요할 것으로 예상된다.

6. Acknowledgement

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program(IITP-2021-2015-0-00742) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation)

참고문헌

[1] Onuma, Yutaka et al. “ECU Software Updating in Future Vehicle Networks.” 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), Taipei, 2017, pp. 35-40.

[2] J.-H. Kim and K.-J. Ha, “Method of In-Vehicle Gateway to Reduce the Reprogramming Time,” Journal of Convergence for Information Technology, vol. 9, no. 7, pp. 25 - 32, Jul. 2019.

[3] Iwiński, Marcin and Sosnowski, Janusz. “Remote software reprogramming in embedded systems.” Pomiary Automatyka Kontrola, vol. 59, no. 8, pp. 769-771. 2013.