논문 21-46-03-17 The Journal of Korean Institute of Communications and Information Sciences '21-03 Vol.46 No.03
https://doi.org/10.7840/kics.2021.46.3.549

LED Matrix를 사용한 Smart Headlight 제어 시스템의 구현

김 봉 효•, 이 성 원°

Implementation of the Smart Headlight Control System Using an LED Matrix

Bong-hyo Kim•, Seong-won Lee°

요 약

최근 차량용 헤드라이트는 LED Matrix를 적용하여 ADAS 정보를 받아 주변 환경에 적합한 주사가 가능하도록 지능적으로 제어하는 방향으로 발전하고 있다. 하지만 LED의 개수가 증가함에 따라 제어하기 위해 필요한 통신 데이터 수가 증가하게 되고, 데이터 수 증가로 통신오류 및 EMI 문제가 발생하게 된다. 본 연구에서는 이를 개선하기 위해 LED Matrix 간에 MESH network을 구성하여 개별적으로 LED를 제어하는 것뿐만 아니라 그룹으로 shift 제어가 가능 하도록 시스템을 설계하였고, 통신 데이터를 줄이기 위해 헤드라이트 패턴을 분석하여 통해 그룹제어와 개별제어의 순서를 조정하여 기존보다 더 적은 명령어로 헤드라이트를 제어할 수 있도록 하였다. 또한 CRC 오류검출을 통하여 통신의 신뢰성을 향상하였다.

Key Words : automotive headlight, LED Matrix, ADAS, smart headlight

ABSTRACT

Recently vehicle headlights have been developed to control the LED Matrix intellectually with the help of ADAS information. However, as the number of LED increases, the amount of communication data to control increases, which leads to communication errors and EMI issues. In this study, we build a headlight system where the LED Matrix forms a MESH network and group shift operations are implemented. Furthermore, by analyzing headlight patterns, the issue order of group operation commands and individual control commands are rearranged to reduce communication data. The communication reliability is also improved using CRC error detection scheme.

Ⅰ. 서 론

차량의 헤드라이트는 주행상황 속에서 운전자 및 보행자에게 차량의 존재를 알리거나 안전한 운행을 위해 시야 확보를 하는데 있어 중요한 역할을 한다. 최근 차량용 헤드라이트에 주변 상황에 따라 지능적으로 헤드라이트 밝기를 조절하기 위하여 고해상도의 LED Matrix를 사용하는 연구가 진행되고 있다[1,2]. LED 헤드라이트는 기존에 사용되던 다른 램프들에 비해 매우 긴 수명을 가지고 있으며, 전력소모 역시

※ 본 연구는 산업통상자원부의 재원으로 한국에너지기술평가원(KETEP)의 지원을 받아 수행한 연구과제(20190100000050)와 산업통상자원부와 KSRC 지원 사업인 미래반도체소자 원천기술개발사업(10080649)의 지원으로 수행되었음.
• First Author : Department of Computer Engineering, Kwangwoon University, terry5621@kw.ac.kr, 학생회원
° Corresponding Author : Department of Computer Engineering, Kwangwoon University, swlee@kw.ac.kr, 종신회원
논문번호 : 202011-287-D-RE, Received November 18, 2020; Revised January 4, 2021; Accepted January 5, 2021

작다. 또한 ADAS(Advanced Driver Assistance System)의 정보를 받아 Matrix내의 개별 LED를 제어함으로써 주변 환경에 보다 적합한 조명을 주사할 수 있다[3],[4]. LED 헤드라이트는 flickering을 방지하기 위하여 60 ~ 100hz의 주기로 전체 Matrix를 업데이트 하므로 Matrix내의 LED 개수가 증가하면 데이터 통신량이 더욱 많아지므로, 차량내의 전기적 잡음으로 인한 오류에 취약하게 되고, LED 통신으로 인한 EMI가 발생 등 다양한 문제를 야기 시킨다[5].

본 논문에서는 LED Matrix Headlight의 제어 신호를 오류에 강건하게 하고, EMI발생을 줄이기 위하여 제어 통신량을 줄이는 시스템을 제안한다. LED Matrix에 전송하는 데이터를 효율적으로 줄이고자 LED Matrix내 LED를 MESH network형태로 구성하여 shift 명령어를 추가하여 그룹 제어를 가능하게 하였으며, 명령어를 사용하여 제어할 수 있도록 통신을 위한 명령어 포맷을 정의하였다. 또한 주행 상황 속에서 헤드라이트의 변화 패턴을 분석하여 그룹 제어와 개별 제어의 순서를 조정하여 더 적은 명령어로 통신하여 동일한 결과를 나오게 하는 알고리즘을 제안하였다. 통신 오류를 줄이기 위해 명령어에 AUTOSAR CRC-8(Cyclic Redundancy Check)를 적용하였으며, LED Matrix의 왼쪽과 오른쪽을 함께 또는 독립적으로 통신할 수 있도록 하였다. 제안한 시스템은 마이크로프로세서와 FPGA를 사용하여 구현하였다.

본 논문의 2장에서는 현재 상용화된 차량의 헤드라이트의 발전과 제어에 대한 배경지식을 소개하고 3장에서는 전체적인 시스템 구조와 통신에 사용된 명령어 포맷, 헤드라이트 패턴을 분석하여 명령어를 통신하도록 하는 알고리즘을 소개한다. 4장에서는 실험 환경 및 다양한 주행상황 데이터, 그리고 실험 결과를 보여준다. 마지막으로 5장에서 본 논문에 대한 결론을 맺는다.

Ⅱ. 배경지식

최근의 헤드라이트는 단순히 어두운 곳을 비추는 역할을 할 뿐만이 아니라 ADAS의 일부로 기능이 확장되고 있다. 여러 개의 LED를 정교하게 배치하여, 배치한 LED의 일부 또는 전체의 밝기를 개별적으로 조절할 수 있게 만든 것이 LED Matrix headlight이다[7].

현재 Benz 사의 "Multi beam headlight"와 Audi 사의 "Matrix LED headlight"와 같이 LED Matrix headlight에 ADAS를 적용하여 주행상황 중 안전을 위해 상황에 맞추어 헤드라이트를 제어하도록 개발하고 있다[8,9]. Benz 차량의 Multi beam headlight의 경우 좌, 우 각 3열로 배열 된 84개씩 168개로 구성되어 있으며 4개의 컨트롤 유닛은 주행 상황에 맞는 최적의 조명을 1초에 100번 계산을 한다. Audi 차량의 Matrix LED headlight도 5개의 리플렉터 안에 5개의 LED를 묶어서 좌, 우 각 25개씩 50개로 구성되어 있다. 위 2개의 헤드라이트는 ADAS에 쓰이는 카메라나 레이더가 파악한 주변 상황, 휠과 페달 작동 상태, 주행 속도와 도로 관련 정보를 이용하여 현재 주행 상황에 알맞게 조절한다. 또한 전방 차량, 보행자 그리고 표지판을 인식하여 해당 방향의 밝기를 고려하여 제어한다[6].

향후 더욱 많은 LED를 사용하는 스마트 헤드라이트(smart headlight)를 사용하는 방향으로 개발이 진행되고 있어, LED 데이터의 전송량을 줄이는 문제는 스마트 헤드라이트 개발의 중요한 이슈가 되고 있다.

Ⅲ. 본 론

전체 시스템은 크게 마이크로프로세서와 오류 검출 기능을 포함한 SPI Interface, command decoder, 그리고 LED driver로 나누어진다. Fig. 1은 본 논문에서 제시하는 전체 시스템 구조이다.

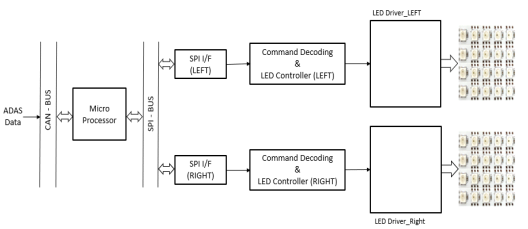


Fig. 1. System architecture

3.1 마이크로프로세서와 SPI Interface

마이크로프로세서에서는 CAN bus를 통해 전달받은 ADAS의 헤드라이트 제어 패턴을 분석하여 최소의 LED Matrix 제어 명령어를 SPI Interface를 통해 헤드라이트로 전송한다. 이 때 헤드라이트는 좌, 우에 각각 있으므로 좌, 우 2개의 SPI slave를 각각 또는 동시에 제어할 수 있도록 명령어 포맷을 정하였다. Command는 총 32bit로 control 8bit, address 8bit, data 8bit, 마지막으로 오류 검증을 위한 CRC 8bit 순으로 구성되어 있다.

Fig. 2는 LED Matrix를 제어하기 위해 마이크로프로세서에서 전송하는 명령어 포맷이다. 명령어 포맷의

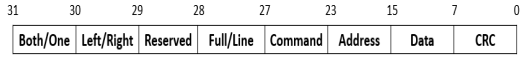


Fig. 2. Data format

최상위 8bit는 control이며 control의 8bit 중 최상위 bit는 LED Matrix 좌, 우를 동시에 제어할 것인지 한 방향을 제어할 것인지 결정하고, 다음 1 bit는 최 상위 bit가 0일 경우, 1이면 좌측, 0이면 우측을 선택하여 제어한다. 상위 4번째 bit은 LED Matrix의 모든 라인을 제어할지 아니면 한 라인만을 제어할 지를 선택하는 bit이다. Full을 가리키게 되면 address의 8bit는 제어할 위치 값이 들어가며, Line을 가리키게 되면 address의 값은 라인 값을 가리킨다. 다음 4bit는 control opcode를 나타내며, LED driver에 저장 및 초기화, 상, 하, 좌, 우 shift를 전체 및 line별로 제어하는 opcode들이 있다. 또한 status register를 읽어 LED 오류를 확인하는 opcode도 포함하고 있다. 그 다음 8bit는 LED 주소를 가리키며, 다음 8bit는 밝기 값, 마지막 8bit는 AUTOSAR CRC-8 오류 검증 바이트이다.

3.2 Command decoder

Command Decoder는 SPI interface를 통해 마이크로프로세서로부터 명령어를 받아 해석하고, LED Matrix의 값들의 이동을 제어하거나, 개별 LED에 값을 지정한다. SPI slave에는 CRC를 이용한 error detection module이 있어 AUTOSAR CRC-8을 이용하여 계산한 CRC 값을 검증하여 오류를 검출한다. 그리고 전송된 명령어의 무결성이 확인되면 MISO(Master In Slave Out)를 통해 master로 ACK 신호를 보낸다. 마이크로프로세서는 SPI master를 통하여 ACK 신호를 받았을 경우, 오류가 없는 것으로 판단하여 다음 LED Matrix 상태에 필요한 명령어를 보내주고, 못 받았을 경우 전송 오류로 판단하여 기존의 보냈던 명령어를 재전송한다. 제안된 시스템에서는 헤드라이트의 실시간성을 고려하여 마이크로프로세서가 ACK 신호를 2번 못 받았을 경우 해당 LED를 무시하고 다음 명령어를 전송하도록 하였다. 해당 LED에 대해서는 전체 LED Matrix의 data 전송이 끝난 후 LED driver의 LED 오류정보를 확인하여 전송 오류인지, LED 오류인지를 파악한다.

3.3 Command decoder

LED driver는 LED 상태를 저장하는 MESH network 형태로 구성되어 있는 LED Dat와 직렬 연결된 32개의 LED를 제어하는 PWM generator가 좌, 우 헤드라이트의 4 line을 제어하도록 한다. LED Dat는 command decoder에서 보낸 신호를 받아 개별 LED의 밝기 값의 저장 및 그룹 단위의 shift 작업을 진행한다.

Fig. 3은 MESH network로 이루어진 LED Driver의 블록도이다. PWM generator는 LED Dat에 저장된 데이터들을 LED Matrix를 제어하기 위한 PWM 신호로 변환하여 전송하며 좌우 4개 씩, 총 8개를 사용하였다. LED는 256단계로 밝기 조절이 가능하도록 하였으며, 제어회로의 간소화를 위하여 라인 별로 32개가 직렬로 연결되어 데이터의 전달이 daisy chain 형태로 전달되는 부품으로 구현하였다.

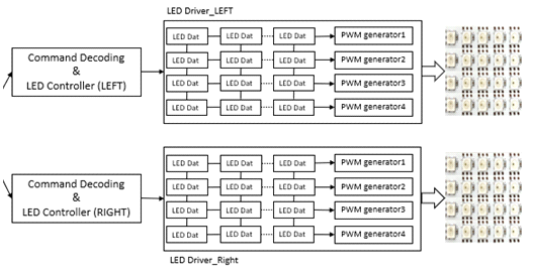


Fig. 3. LED driver

3.4 통신량 감축 알고리즘

마이크로프로세서는 LED driver와 별개로 LED Matrix의 데이터를 자체적으로 저장하고 있으며, ADAS에서 새로운 LED Matrix의 값들을 보내오면 가능한 적은 수의 명령어로 LED Matrix의 값들을 최신의 값들로 업데이트하기 위하여 전송할 명령어의 값들과 순서를 결정한다.

Fig. 4는 마이크로프로세서에서 LED Matrix를 제어하기 위해 최소의 명령어를 내보내기 위해 제안한 알고리즘의 flowchart이다. 처음에는 256개의 LED가 각각 밝기 값을 가지고 있다고 가정한다. 그 다음 새로운 256개의 LED address에 밝기 값으로 초기화한다. 처음에 켜져 있는 LED 상태와 새로 들어온 LED 상태를 비교하여 다른 부분의 개수를 찾는다. 이때 다른 부분의 개수를 diff_cnt라고 정의하였다. 알고리즘은 diff_cnt 개수에 따라 4가지 형태로 분기한다. diff_cnt가 0일 때는 다른 LED 상태가 없으므로 명령어 없이 LED driver에 저장된 값을 유지하고 다음 256개의 LED 정보 값을 받는다. diff_cnt가 1일 때는 1개의 명령어를 이용하여 다른 하나의 LED address에 data값을 전송한다. diff_cnt가 2일 경우, 하나의 명령어인 shift 기능을 이용하여 다음 LED 상태와 같으

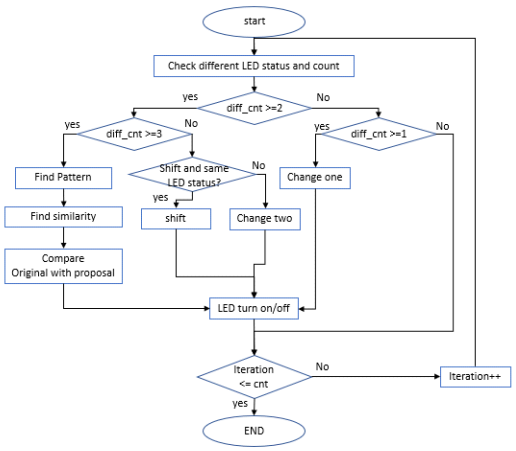


Fig. 4. Program flowchart

면 1개의 명령어를 전송하고 아닐 경우 2개의 명령어를 이용하여 다른 LED address에 data 값을 전송한다. 마지막으로 diff_cnt가 3이상일 경우 패턴을 분석하고 명령어로 해결 못하게 되면 유사도 값을 측정하여 찾아가는 방식을 사용했다. 유사도 측정은 현재 LED 상태에서 shift를 상, 하, 좌, 우 하였을 때의 LED 상태와 다음 상황의 LED 상태가 얼마나 비슷한지를 수치를 나타내는 과정이다. 패턴 분석을 이용해 찾은 개수, 유사도 측정을 하여 찾은 개수와 diff_cnt의 개수를 비교하여 최소의 개수를 선택하여 LED Matrix를 제어한다. 위의 2가지 방법으로 못 찾은 경우 diff_cnt만큼의 명령어를 이용하여 LED를 제어한다.

Fig. 5는 Find pattern 알고리즘 부분에 대한 과정을 설명하기 위한 플로우차트(flowchart)이다. 첫 번째로 전체 shift로 패턴 확인 과정은 이전 LED Matrix 상태에서 전체적인 상, 하, 좌, 우 shift 기능만을 이용하여 현재 LED Matrix 상태와 동일한지 찾아본다. 그 다음 패턴 분석은 전체적인 상, 화, 좌, 우 shift 기능을 사용 후 line 별로 diff_cnt 개수를 다시 확인하여 diff_cnt의 개수대로 패턴을 다시 찾아본다. Line shift로 패턴 확인 과정은 전체적인 shift가 아닌 line 별로 각각 shift 기능을 이용하여 전송량을 줄일 수 있는지 찾아본다. 마지막으로 유사도를 찾아 어느 방향으로 shift를 할 것인지를 결정한다.

Fig. 6은 Find similarity 알고리즘 부분에 대한 과정을 설명하기 위한 플로우차트이다. 유사도 측정은 line 별로 진행이 되며 이전 LED 상황에서 상, 하, 좌, 우 shift를 진행했을 경우 현재 LED 상황과 얼마나 비슷한 지를 구한다. 유사도를 구하는 과정에서 같은 유

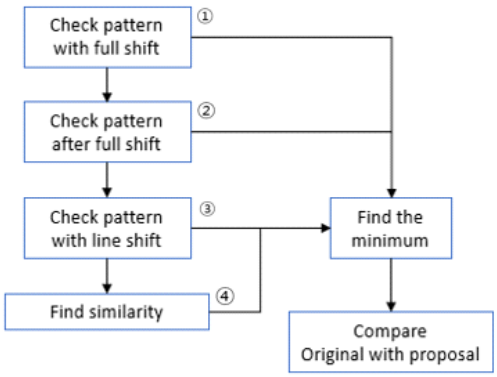


Fig. 5. Find Pattern flowchart

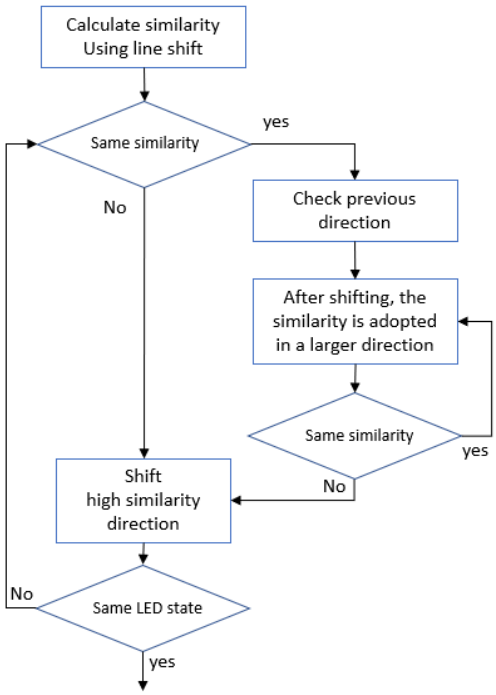


Fig. 6. Find Similarity flowchart

사도가 나올 수가 있으므로 Figure 6의 알고리즘처럼 유사도를 추가적인 shift 기능을 이용하여 다시 유사도를 구한다. 추가적인 shift를 하기 전에 이전에 shift를 어느 방향으로 했는지 체크한다. 예를 들어 좌로 shift한 결과에서 추가적으로 우로 shift하게 될 경우 제자리로 돌아가게 되므로 그 부분을 방지하기 위해 체크한다. 유사도 찾기 과정까지 마치면 모든 경우의 수를 체크하여 최솟값을 찾은 후 현재 LED Matrix 상황에 맞게 명령어를 전송한다. 마지막으로 패턴 분석을 이용하여 찾은 개수와 유사도를 측정하여 찾은

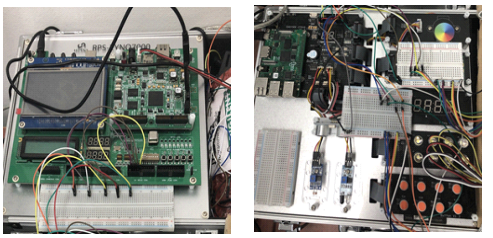
개수를 비교하여 diff_cnt 보다 더 많은 패턴 분석을 하게 되면 바로 다음 shift 패턴으로 넘어가도록 하였고, 선택된 최소의 명령어의 개수를 선택하여 LED Matrix를 제어한다.

Ⅳ. 실 험

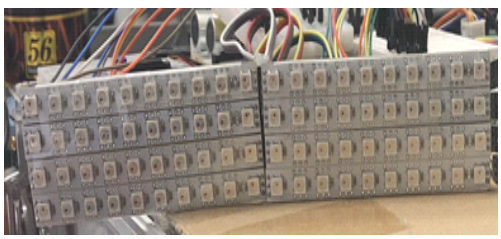
Zynq 7020을 사용하여 마이크로프로세서와 SPI master부분을 구현하였으며, Zynq 7010를 사용하여 SPI slave와 LED driver를 구현하였다. LED Matrix로는 WS2812B를 사용하였다. Zynq 7020의 마이크로프로세서로부터 최소의 명령어를 출력하여 SPI master로 전송하고, Zynq 7010의 FPGA에서 구현된 SPI slave로 받아 command decoding을 거쳐 LED 상태를 driver에 저장 후 8개의 PWM generator로 총 256개로 좌, 우 각 128개의 LED를 동시에 또는 독립적으로 제어하도록 구현했다. Fig. 7은 실험환경의 사진이다.

Fig. 8은 LED Matrix 출력 결과를 보여준다.

실험은 Fig. 9의 사진들처럼 교차로 시골길, 언덕길, 터널 그리고 여러 상황이 섞인 복잡한 주행상황



(a) Zynq7020 (b) Zynq7010



(c) WS2812B

Fig. 7. Experimental environment

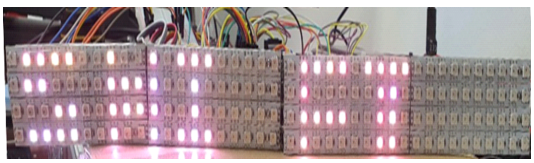
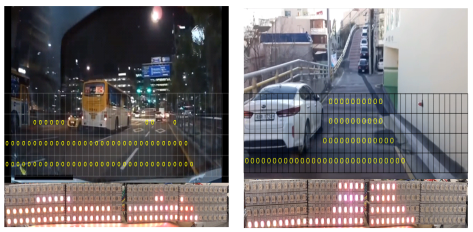
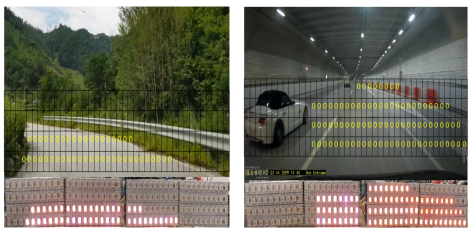


Fig. 8. Display LED Matrix



(a) 교차로 (b) 언덕길



(c) 시골길 (d) 터널길

(e) 복합 상황

Fig. 9. Experimental image

속에서 차량이 이동함에 따라 LED Matrix를 제어하는데 있어 LED driver에 LED 상태를 저장하는 메모리의 유무에 따라 제어하는데 필요한 byte 수와 개별적으로 제어하던 기존의 방식과 달리 알고리즘 적용 후 전송할 byte가 얼마나 줄었는지에 대해 실험하였다. 100hz로 LED Matrix headlight를 업데이트를 목표로 해야 하지만 실험에서는 검증의 편의상 5hz의 속도로 진행하였다.

표. 1은 LED Matrix를 제어하는데 필요한 frame당 통신 데이터를 보여주는 결과표이다. 실험은 총 3가지로 나타내어 비교를 하였다. 첫째로 LED driver에 LED 상태를 저장하는 메모리가 없이 매 프레임 전체 LED 값만을 전송하는 경우 frame당 필요한 byte 수, 둘째로 LED 상태를 저장할 수 있는 메모리가 있는 경우 명령어에 의해서 필요한 정보만 전송하는 경우, 마지막으로 LED 상태를 저장하면서 헤드라이트 패턴 분석을 통한 최소의 명령어를 통하여 LED를 제어하여 출력하는 알고리즘을 적용한 경우이다. 총 frame 수는 각 41, 118, 88, 95, 100장으로 시간은 약 8초,

Table 1. Experimental Results

		교차로	시골길	언덕길	터널	복합상황
총 frame		41	118	88	95	100
frame당 byte		512	512	512	512	512
With LED Data buf (bytes/frame)	min	0	0	0	0	0
	avg	52	20	40	32	12
	max	328	60	100	404	28
With command Reduction algorithm (bytes/frame)	min	0	0	0	0	0
	avg	20	16	20	12	8
	max	56	60	100	288	20

22초, 18초, 17초, 20초의 영상으로 실험하였다. frame당 byte수는 기존의 LED Matrix를 켜기 위해 LED address와 데이터를 각 256개씩 보내어 512 byte로 동일하다. LED driver에서 LED 상태를 저장한 메모리가 있을 경우 frame당 byte 수를 최소와 평균 그리고 최대로 나타내었다. 최소의 경우 한 곳도 안 변하는 경우가 있어 0byte다. 그러나 평균으로 나타내게 되면 교차로는 52byte, 시골길은 20byte, 언덕길은 40byte, 터널은 32byte 마지막으로 복합주행 상황은 12byte로 줄어든 것을 확인할 수 있다. 최대로 많이 변하는 경우의 frame은 교차로는 328byte, 시골길은 60byte, 언덕길은 100byte, 터널은 404byte 그리고 복합주행 상황은 28byte가 필요로 한다. LED 상태를 저장하는 상태에서 헤드라이트 패턴 분석을 통한 알고리즘을 적용하게 되면 frame당 평균 교차로는 20byte, 시골길은 16byte, 언덕길을 20byte, 터널 길은 12byte 마지막으로 복합주행 상황은 8byte로 알고리즘 적용 전보다 감소한 것을 확인하였다. frame당 최대 byte도 교차로는 56byte, 시골길은 60byte, 언덕길은 100byte, 터널 길은 288byte 그리고 복합주행 상황은 20byte가 필요로 한다.

V. 결 론

본 논문에서는 LED Matrix 제어를 위한 구조 설계와 명령어 통신량을 줄이기 위한 알고리즘을 제안하였다. 기존의 개별적으로 제어하는 명령어 이외에 그룹으로 제어할 수 있는 shift 명령어를 추가하였으며, 이를 시스템에 적용하기 위해 LED 정보를 MESH network로 구성하였다. 또한 LED Matrix headlight 패턴 분석을 통해 기존의 통신량보다 줄일 수 있었다. 그러나 모든 주행상황 속에서 통신량을 감소시킬 수 있는 것이 아니라 다양한 주행상황에 따라 평균값이 다르다. 앞으로 추가적인 연구에서 켜고 끄는 것만이 아닌 밝기 제어를 추가하여 알고리즘을 구현할 것이며, shift 이외에 다른 명령어를 추가하여 LED Matrix를 제어하기 위해 필요한 통신 data를 줄이는 연구를 진행할 계획이다.

References

[1] B. Kang, "New technology of automobile front lighting," *AUTO J.*, pp. 16-20, Jul. 2011.

[2] C.-K. Cho and S.-H. Park, "Buck converter driving matrix LED with constant current for vehicle's high-beam," *J. IEIE*, pp. 32-35, Nov. 2018.

[3] P. Song, Y. Zhang, X. Wu, and Y. Lan, "Design and implementation of the adaptive control system for automotive headlights based on CAN/LIN network," *3rd ICIMCCC*, pp. 1598-1602, 2013.

[4] W. Choi and J. Shon, "A study on reducing can data traffic for improving vehicle safety," *KIIT*, pp. 87-92, Nov. 2016.

[5] T. Maeda, T. Hirayama, Y. Kawanishi, D. Deguchi, I. Ide, and H. Murase, "Analyzing headlight flicker patterns for improving the pedestrian detectability from a driver," *Int. Conf. Intell. Transp. Syst.*, pp. 3113-3118, Nov. 2018.

[6] H.-W. Song, H.-J. Do, and S.-H. Park, "An matrix manager IC with LED fault detection and low current consumption for automotive headlight systems," *J. IEIE*, pp. 36-39, Nov. 2018.

[7] B. Kang, H. Lee, and J. Kim, "A study on the possibility of applications of LED headlamp," *The Korean Soc. Automotive Eng.*, pp. 177-183, Aug. 2006.

[8] Audi AG Pressetext, *A8 erstrahlt in neuem Licht*, 2013.

[9] Mercedes-Benz Pressetext, *MULTIBEAM LED bringt Licht ins Dunkel.*

김 봉 효 (Bong-hyo Kim)

2019년 2월 : 광운대학교 컴퓨터 공학과 학사
2021년 2월 : 광운대학교 컴퓨터 공학과 석사
<관심분야> 임베디드 시스템, 신호처리

이 성 원 (Seong-won Lee)

1988년 2월 : 서울대학교 제어계측공학과 학사
1990년 2월 : 서울대학교 제어계측공학과 석사
2003년 5월 : University of Southern California 박사
2005년 3월~현재 : 광운대학교 교수
<관심분야> 프로세서구조, 신호처리, 영상처리