

보행 교통사고 감소를 위한 시스템 구축

(System construction for pedestrian traffic accidents)

본 보고서를 융복합 캡스톤 디자인 과제의 최종보고서로 제출합니다.

〈 요약 문 〉

설계과제 목표	<ul style="list-style-type: none"> ➤ 사용자 및 환경 분석 <ul style="list-style-type: none"> ○ 페르소나 선정 <ul style="list-style-type: none"> ▪ 보행자 / 운전자 관점에서 페르소나 선정 ➤ 아두이노 기반 보행 시스템 구축 <ul style="list-style-type: none"> ○ 압전센서, 열 감지 센서, 도트 LED를 활용한 시스템 구축 		
내용	<ul style="list-style-type: none"> ➤ 설문조사 <ul style="list-style-type: none"> ○ 운전자 관점과 보행자 관점, 관찰자 관점으로 구분하여 설문 ○ 신호등 잔여 시간 시스템에 관한 문항을 중심으로 설문 ○ 조사 기간 : 19.06.30 ~ 19.07.05 ○ 응답자 수 : 162명(남 : 59, 여 103) ➤ 아두이노 기반 시스템 개발 <ul style="list-style-type: none"> ○ Use-Case 다이어그램 Flow Chart 작성 ○ CASE별 관리 <ul style="list-style-type: none"> ▪ 차량 우회전 시 횡단보도에 보행자가 존재하는지 알려주는 서비스 ▪ (Dual-Tasking) 스마트폰 사용하는 보행자 관점에서 보조 알림 서비스 ▪ 인적이 드문 도로 환경에서 보조 알림 서비스 ➤ 테스트 및 오류 수정 <ul style="list-style-type: none"> ○ 기능 테스트 및 오류 수정 		
기대효과	<ul style="list-style-type: none"> ➤ 실제 행위자 데이터 기반으로 시스템 개발 <ul style="list-style-type: none"> ○ 보행자 / 운전자 관점을 구분하여 개발함으로 보다 안전하게 교통 가능 <ul style="list-style-type: none"> ▪ 야간 / 주간에 따른 안전한 행위 가능 ▪ Dual-Tasking 환경에서 안전한 행위 가능 ○ 보조 알림 시스템을 추가함으로 의료 보험비, 교통처리 비용 등 절감 될 것 		
Keywords	아두이노	보행 안전 사고	인지 사고 능력
	Dual-Tasking	3D 프린팅	사용성 평가

목 차

I. 서 론	1
1. 설계 배경	1
2. 기술적 이론적 배경 및 최종 목표	2
II. 설계 수행 내용	6
1. 보행 실태 조사	6
2. 보행 안전 시스템 설계	8
2.1 하드웨어 설계	10
2.2 소프트웨어 설계	11
3. 보행 안전 시스템 구현	13
3.1 하드웨어 구현	13
3.2 소프트웨어 구현	15
4. 보행 안전 시스템 결과	17
5. 설계 구성 요소 및 제한 요소	19
III. 결 론	21
1. 과제 결과 요약	21
2. 개선 방안 및 향후 과제 방향	21
3. 설계 목표의 중요도 및 달성도	22
IV. 참고문헌	23
[부록]	24

I. 서론

1. 설계 배경

도로교통공단의 자료에 따르면 2016년 기준으로 우리나라 교통사고 사망자중 보행사망자 점유율은 OECD 평균인 19.5%보다 %p가 2배 높은 39.9%라고 한다. 그 중에서도 일반 성인 과 다른 행태적 특성을 가진 어린이와 노약자에 관한 사고, 스마트 기가 사용으로 인한 사고 율이 높은 것을 알 수 있다[1]. 보행 교통사고의 높은 발생원인으로 Dual-Tasking을 들 수 있다. Dual-Tasking이란 사람이 운동 작업을 하는 동시에 인지 작업을 하는 것을 말한다. 보행을 하는 동시에 스마트폰을 사용하는 동작 또한 이에 속한다. Dual-Tasking은 양쪽 작업 수행에 모두 영향을 미칠 수 있다. 이와 관련하여 한국교통연구원 자료에 의하면 보행교통사고 발생 시 70%가 접근 차 량을 응시 하지 않는다고 나타났다. 또한, 보행자가 출현한 시점부터 사고발생 순간까지 불 과 2.6초로 분석되었으며, 주간보다는 야간, 자동차 속도가 빠를수록 사고소요시간이 감소 한 것으로 나타났다[2]. 이는 노약자, 어린이뿐만 아니라 일반 성인도 70%가 넘는다는 뜻 이며, 사고 발생 시간이 찰나에 일어난다는 것을 뜻한다. 현재까지 어린이, 노약자, 스마트 기기 사용자를 대상으로 하는 방안이 많이 제시되어 왔 다. 그러나 일반 성인들을 포괄하는 방안은 부족한 경우가 많다. 따라서 본 과제에서는 포괄적인 대상으로서 일반 보행자에 관점에서 보행 사고를 줄이기 위해 Dual-Tasking 환경으로 행위자 관점에서 보행자와 운전자로 나누고, 교통 시스템 관점에서 신호등 유무로 나누어 신호등과 횡단보도를 비롯한 주변 환경에 시각과 청각적 인 보조 알림을 제공하여 위험을 인식할 수 있도록 하고자 한다. 최종적으로 인지신경학적인 인문학과 소프트웨어적인 공학적 소양을 융합하여 보행자/운 전자 관점에서 청각, 시각적 자극에 기반 교통 시스템을 제안하고자 한다.

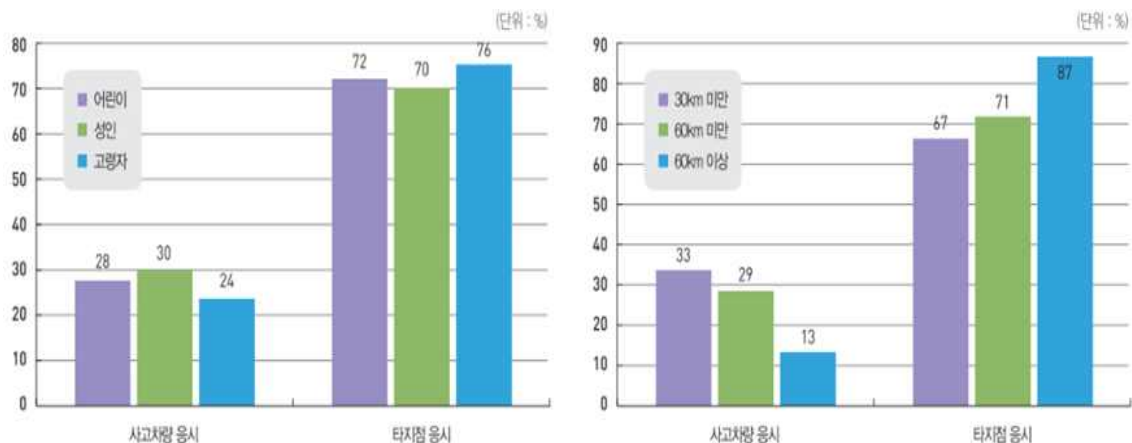


그림 1. 사고발생 순간의 보행자 시선방향(좌)과 자동차 속도별 보행자 시선방향(우)

2. 기술적 배경 및 최종 목표

2.1. 기술적 배경

행위자 관점에서 운전자, 보행자로 구분하여 CASE를 구분한다. 첫 번째 CASE로 보행 신호 등에 황색 LED를 추가한다. 점자블록에 압전센서를 설치하여 추가한 LED 포함 총 3단계로 나누어 점진적인 도트 알리를 준다. 두 번째 CASE로 인적이 드문 도로에서 LED를 이용한 보조 알리를 준다. 보행자가 점자 블록에 압력을 가하면 임계점이 넘었을 시 LED 알리를 주 어 운전자가 멀리서도 확인할 수 있도록 한다. 세 번째 CASE인 차량 우회전 시 보행자가 보행하고 있다는 것을 거리 가지 센서, 열 감지 센서를 통해 인식하여 LED 알리를 준다.

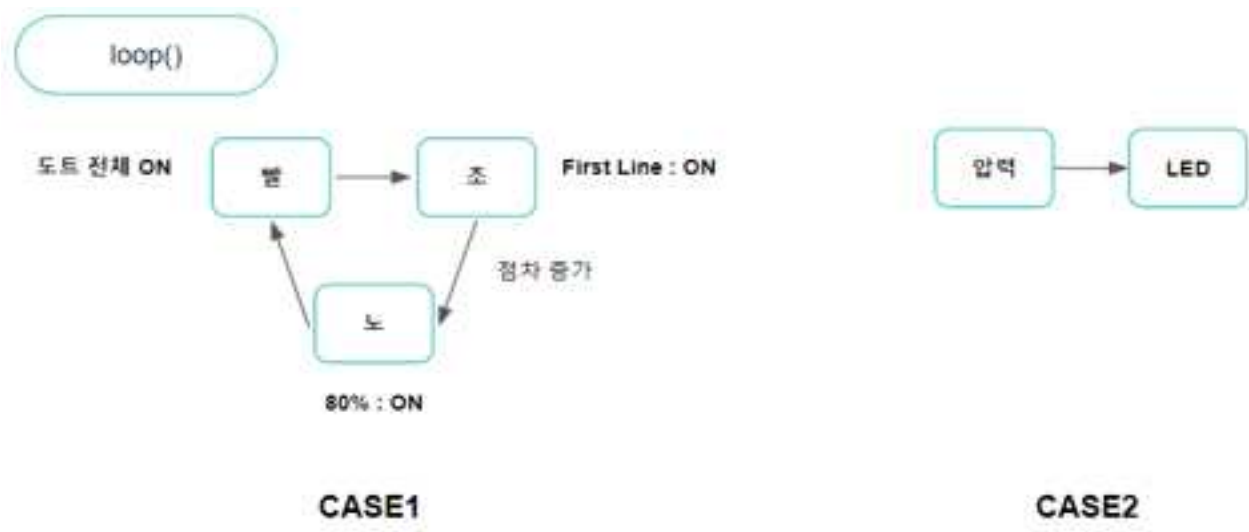


그림 2. 시스템 흐름도

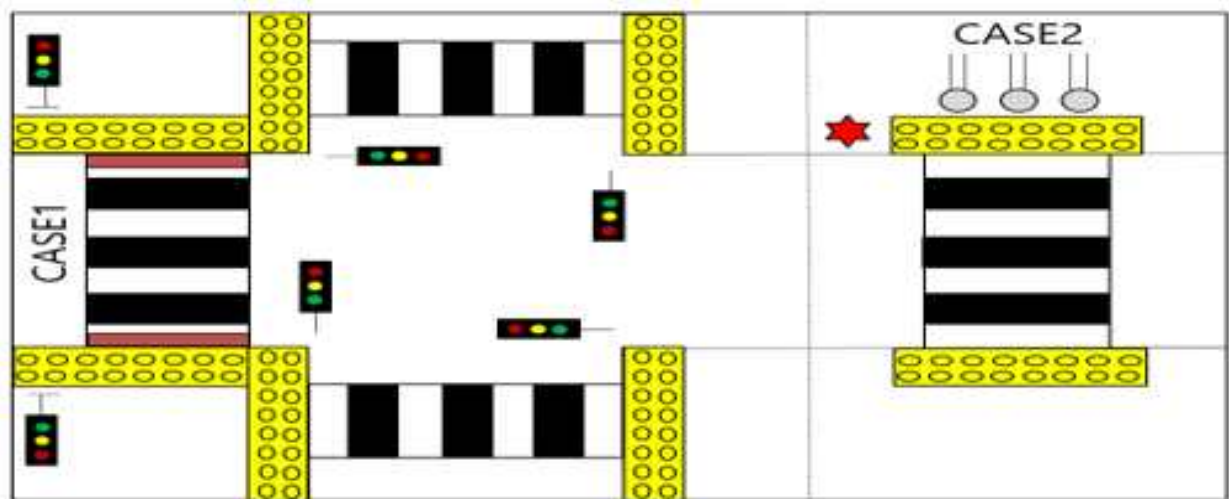


그림 3. 시스템 구성

2.2. 이론적 배경

2.2.1 보행 신호등 운영 현황



그림 4. 우리나라의 보행신호등

우리나라의 보행신호등에 대한 운영방법은 도로교통법시행규칙 '신호기가 표시하는 신호의 종류 및 신호의 뜻'에 명시되어 있다. 이에 따르면 우리나라의 보행 신호는 [그림 번호 넣을 것]에서 볼 수 있듯이 녹색 및 적색의 2색등으로 구분되며, 적색등은 점등상태와 소등상태, 녹색등은 점등상태, 소등상태, 점멸상태로 운영된다. 이 때 각 상태별 보행 신호의 의미는 다음과 같다.

- 적색등화 : 보행자는 횡단을 하여서는 안 된다.
- 녹색등화 : 보행자는 횡단보도를 횡단할 수 있다.
- 녹색등화의 점멸
: 보행자는 횡단을 시작하여서는 안 되고 횡단하고 있는 보행자는 신속하게 횡단을 완료하거나 중지하고 보도로 되돌아와야 한다.

2.2.1 보행 신호등 보조장치 운영 현황

우리나라 보행신호등 보조 장치에 대한 설치 근거는 도로 교통법시행 규칙 '신호등의 종류, 만드는 방식 및 설치기준'의 주석에 명시되어 있으며 의미는 다음과 같다.

- 보행등을 설치하는 경우 보행자의 안전한 보행을 위하여 다음 각 목의 보조 장치를 설치 할 수 있다
- (가) 보행자에게 남은 보행 시간을 알려주는 보조 장치
- (나) 보행자에게 보행신호 등을 음성으로 알려주는 보조 장치

경찰청 “보행신호등 보조장치 표준지침” 주요 내용은 다음과 같다.

○ 종류

보조장치의 종류는 두 가지로 구분된다. 하나는 숫자로 잔여시간을 알려주는 ‘숫자형 보조장치’이고, 가른 하나는 도형(역삼각형, 역사다리형, 일자형, 기타도형 등)을 사용하여 잔여시간을 알려는 도형형 보조 장치이다.

○ 설치가능 지점

보조장치는 왕복 6차로 이상인 도로 중에서 보행자 통행이 빈번하고 보행자 횡단사고가 잦은 횡단보도에 설치한다. 단, 왕복 6차로 미만의 도로라도 교통안전상 부득이 설치할 필요가 있을 경우 관할 경찰기관 관계 위원회의 결정에 따라 설치할 수 있다.

○ 운영방법

- (1) 숫자형 보조장치의 잔여시간을 나타내는 숫자는 녹색점멸과 동시에 시작되고 녹색 점멸과 동시에 종료되며 적색등화 동안은 꺼져 있어야 한다.
- (2) 도형형 보조장치는 보행등 녹색등화와 동시에 등화되어야 하고, 녹색점멸 신호시간을 모듈의 개수로 동일하게 나누어 순차 소등시키도록 설계되어야 하며, 적색등화 동안은 꺼져 있어야 한다.
- (3) 신호제어기가 수동으로 조작되고 있을 때에는 수동조작 시점을 기준으로 다음 주기부터 보조장치가 소등된 상태를 유지하여야 한다.
- (4) 보조장치를 최초로 작동 할 때(최초 또는 정전 후 전원 공급 시) 또는 신호제어기를 수동 조작한 후 자동제어로 복귀할 때에는 2주기 내에 신호시간을 자동 감지하여 그 다음 주기부터는 정상적으로 작동되어야 하며, 신호시간을 감지하는 동안은 보조장치가 작동(등화)되어 보행자에게 혼란을 초래하여서는 안된다.

2.3. 설계 최종 목표

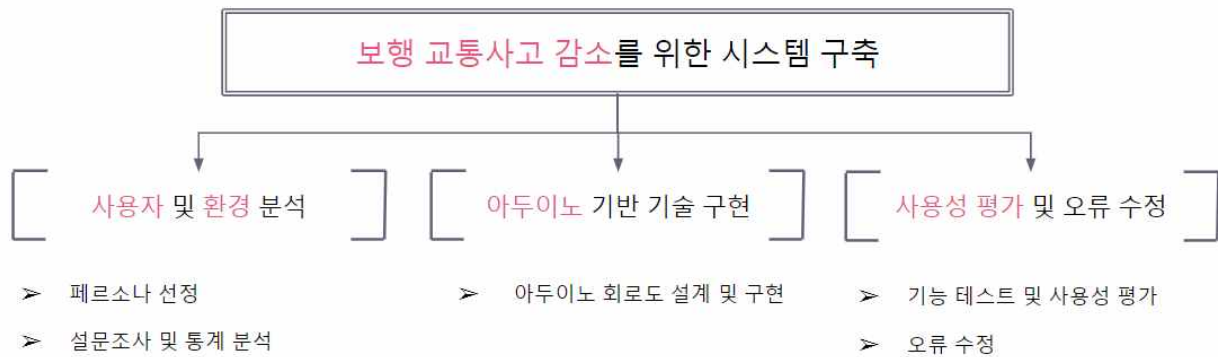


그림 5. 설계 목표

보행 교통사고 감소를 위한 시스템 구축을 목적으로 두고 세부 목표로 사용자 및 환경 분석, 아두이노 기반 기술 구현, 사용성 평가 및 오류 수정이다. 사용자 및 환경 분석은 실제 행위 자 대상으로 설문을 통한 페르소나를 선정하고 결과를 통한 CASE를 구분한다. 구분한 CASE 별로 아두이노 기반 기술 구현을 하며 사용성 평가 및 오류 수정을 한다.

II. 설계 수행 내용

1. 보행 실태 조사



1.1. 조사 개요

본 실태 조사는 실제 행위자 관점에서 본인의 행위에 대한 인식과 잔여 시간 표시장치에 대한 인식을 확인하기 위하여 일주일(19.06.30 ~ 19.07.05)동안 조사를 실시하였다.

조사 방법은 인터넷 설문조사를 하였으며, 문항은 다음과 같은 도로 환경으로 가정하고 조사하였다.

도로 환경
<ul style="list-style-type: none">- 도로폭 35m- 편도 3차선도로- 차량의 비보호- 잔여표시 시작 시간 15초

행위자 관점에서 보행자, 운전자, 관찰자로 유형을 구분하였으며, 잔여 시간 표시 장치는 시간형, 도형형으로 구분하여 조사하였다.

시간형	도형형
	

1.2. 조사 결과

전체 조사 응답자수는 162명(남자 : 59명(36.4%), 여자 : 103명(63.6%))이었고, 그 중에서 10대는 10명(6.2%), 20대는 71명(43.8%), 30대는 11명(6.8%), 40대는 33명(20.4%), 50대는 31명(19.1%), 60대 이상은 8명(4.9%) 이었다.

잔여 시간 표시 장치 존재 유무 문항 조사 결과 118명이 응답자 본인이 많이 다니는 도로에 보편적으로 존재한다 하였으며, 16명이 보편적으로 존재하지 않는다고 나타났다. 잔여 시간 표시 장치 인식에 대한 조사 결과 156명이 의식하고 건넌다고 응답하였으며, 149명이 장치에 대해 긍정적인 효과를 보였다.

이것은 도로에 잔여 시간 표시 장치가 보편화 되어 있지만 사고율을 높다는 것을 파악할 수 있으며, 의존도가 높다는 것을 알 수 있다.

잔여 시간 표시 방법에 대한 문항 조사 결과 135명이 시간으로 표시하는 것이 적당하다 하였으며, 긍정적인 순서대로 도형, 단어, 문장으로 나타났다. 또한, 선호하는 표시 장치 색은 123명이 초록색을 선호하였으며, 파랑색(21명), 빨간색(18명), 주황색(9명)으로 나타났다. 이 같은 결과는 대한민국 기준 잔여 시간 표시 장치가 초록색으로 표현되며 시간표시형이 원칙으로 운행되기 때문인 것으로 파악된다.

보행자가 건널 수 있는 최소 잔여 시간에 대한 조사 결과 13~15초(50명(30.9%)), 10~12초(53명(32.7%)), 7~9초(41명(25.3%)), 4~6초(18명(11.1%))로 나타났다. 이와 같은 결과는 보행자들이 다양한 인식을 하고 있으므로, 짧은 시간도 간과해서는 안되는 시간으로 판단할 수 있다.

보행자 관점에서 보행 행위 설문 조사 결과 신호를 기다리는 동안 Dual-Tasking을 한다는 응답이 66명(37.8%)으로 나타났으며, Dual-Tasking을 하다가 신호를 착각했다는 응답이 29명(17.9%)으로 나타났다. 또한, 보행하던 도중 신호가 변경되었을 시 빠르게 지나간다는 응답이 112명(75.3%), 지나가는 차량 운전자와 커뮤니케이션 한다는 응답이 29명(17.9%), 되돌아간다는 응답이 5명(3.1%), 중앙차선에서 기다린다는 응답이 6명(3.7%)로 나타났다. 이 같은 결과는 중간에 신호가 변경됨에 따라 다른 알림을 주는 것이 타당하다고 판단할 수 있다.

운전자 관점에서 황색 불에 대한 인식 조사 결과 차량이 정지선에 있거나 횡단보도에 있을 때 정지한다는 응답이 113명(79.6%), 꼬리 물기 하여 빠르게 지나간다는 응답이 27명(19%)로 나타났다. 또한 야간 운행 시 사람이나 물체를 인지할 수 있는 보조 시스템이 설치되는 것이 합리적이라는 응답이 111명(79.8%)로 나타났다.

운전자 관점에서 황색 보조 알림은 긍정적이라 판단 가능하며, 야간 시간에 보조 시스템이 존재하는 것이 타당하다 할 수 있다.

2. 보행 안전 시스템 설계

본 설계에서 제안하는 보행 안전 시스템 환경은 다음과 같다.

[표 1] 보행 안전 시스템 환경

CASE1	➤ Dual-Tasking 환경
CASE2	➤ 인적이 드문 도로 환경

CASE1인 Dual-Tasking 환경은 신호를 기다리는 보행자가 신호가 변하는 것에만 집중하지 않고 휴대전화를 사용하는 환경으로 설정하였다. 실제 설문조사 결과 보행자는 Dual-Tasking 상태에서 휴대전화를 보다 다른 사람이 건너는 모습만 보고 보행한다는 응답이 존재하였다. 이와 같이 Dual-Tasking 환경은 보행 안전과 밀접한 연관을 가지며, 중요한 요소이다.

CASE2는 인적이 드문 도로 환경으로 야간 시간에 신호등이 존재하더라도 점멸등으로 변경된다는 환경으로 설정하였다. 설문 조사 결과 인적이 드문 도로, 학교 앞과 같은 특수한 도로에 보조 알림 시스템이 존재해야 한다고 나타났다. 특히, 인적이 드문 도로는 보행자와 운전자 모두 인지하기 어렵기 때문에 과제의 CASE로 설정하였다.

본 설계에서 설정한 변인은 다음과 같다.

가. 통제 변인

○ 도로 환경 관점

■ 편도 3차선, 4거리

■ 도로 폭 35m, 횡단보도 폭 30m

○ 신호등 관점

[표 2] 신호등 시간 간격

(단위 : 초)

신호등	운전자	보행자
시간 간격(43)	- 빨간색 : 30 - 주황색 : 3 - 초록색 : 10	- 빨간색 : 30 - 주황색 : 5 - 초록색 : 8

나. 종속 변인

- 일반적 환경
- Dual-Tasking 환경(스마트폰 사용으로 한정)

다. 독립 변인

- 시각 자극
 - 색, 시각 알림 빈도
- 횡단보도와 행위자 사이의 거리
- 시간(야간, 주간)

라. 조절 변인

- 채도, 명도
- 신호등 깜빡임 정도

마. 매개 변인

- 행위자의 인지 능력

[그림 6]은 제안한 보행 안전 시스템 Use-Case Diagram이다. Dual-Tasking, 차량 우회전 시, 인적이 드문 상황 총 3개의 Use- Case로 구성되어 있고, 각 상황별로 DB에 데이터로 저장되어 교통 상황에 따라 유동적으로 변하는 상황으로 설정하였다.

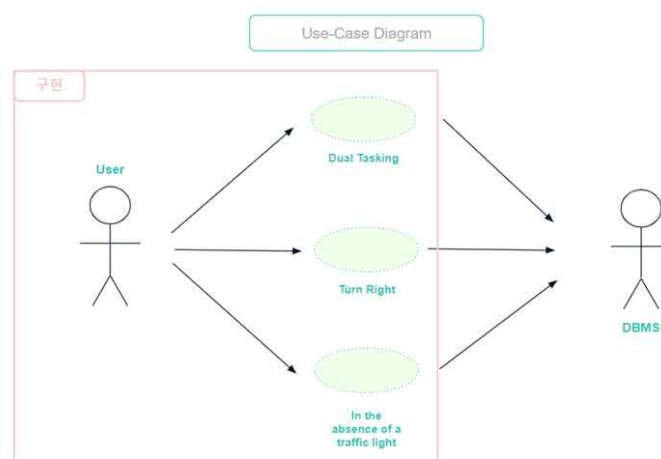


그림 6. Use-Case Diagram

본 과제에서는 아두이노 기반 구현을 목표로 유동적인 연결보다는 각 상황별 구현에 초점을 맞추어 설계하였다.

2.1. 하드웨어 설계

2.1.1. 전체 시스템 설계

[표 3] 도로 선정 후보



[표3]은 실제 도로 환경을 기반으로 시스템을 설계하기 위한 후보군이다. 후보군 모두 교통사고가 많이 발생하며, 후보군 중 부전동 사거리를 제외하고 나머지는 부산 도로의 특징을 담고 있어 전국적으로 대표할 수 있는 도로 환경을 가진 부전동 사거리를 과제 환경으로 하였다.

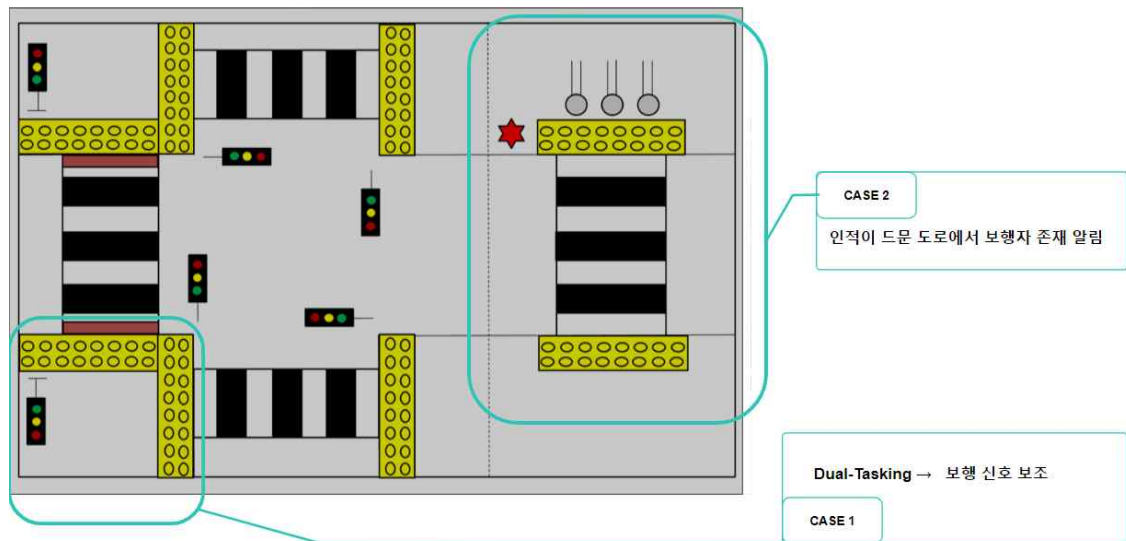


그림 7. 전체 시스템 설계

[그림 7]은 실험 구상 시 설계한 전체 시스템이다. 좌측 하단 CASE 1은 인적이 드문 도로에서 보행자 존재 알림을 설계한 것이고, 우측 CASE 2는 Dual-Tasking 상태를 설계한 것이다.

[그림 7]의 전체 시스템을 통해 다음과 같은 정보를 얻을 수 있다.

- CASE 1 : Dual-Tasking
 - 보행자 신호등 존재
 - 점자 블록 부근에 LED로 보행 알림 표현
- CASE 2 : 인적이 드문 도로
 - 보행자 신호등 없음
 - 점자 블록 기준으로 FSR 센서 존재

2.2. 소프트웨어 설계

CASE 1. Dual-Tasking UML Diagram은 [그림 6]에 명시된 바와 같으며 Dot LED 주기에 대해 명시되어 있다. 입출력 제약이나 부가적인 정보를 기술하기 위해 OCL(Object Constraint Language)을 이용하여 추가사항을 명시하였다.

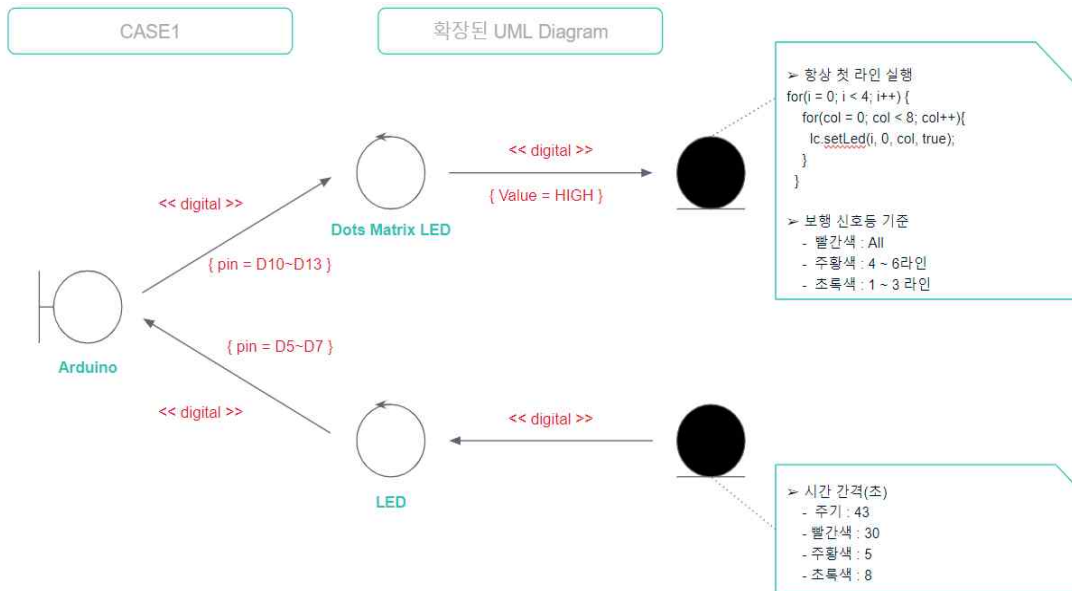


그림 8. CASE 1 확장된 UML Diagram

[그림 8] Diagram을 통해 다음과 같은 정보를 얻을 수 있다.

- 첫 라인은 항상 ON
- 보행 신호등이 빨간색이면 모든 Dot는 ON
- 보행 신호등이 주황색이면 : 1 ~ 6라인 ON
- 보행 신호등이 초록색이면 : 1 ~ 3라인 ON
- 시간 간격은 주기 43초
 : 빨간색 30초, 주황색 5초, 초록색 8초
- 보드에 보행자 신호가 입력장치로 연결되고, Dots Matrix LED가 출력장치로 연결됨
- Dots Matrix LED는 D10 ~ D13에 연결되어 있고, 디지털을 출력(digitalWrite)함
- LED는 D5 ~ D7에 연결되어 있고, 디지털 값을 입력(digitalRead)받음

CASE 2. 인적이 드문 도로 UML Diagram은 [그림 8]에 명시된 바와 같다. 입출력 제약이나 부가적인 정보를 기술하기 위해 OCL(Object Constraint Language)을 이용하여 추가사항을 명시하였으며, 입출력 제약이나 부가적인 정보를 기술하기 위해 OCL(Object Constraint Language)을 이용하여 추가사항을 명시하였다.

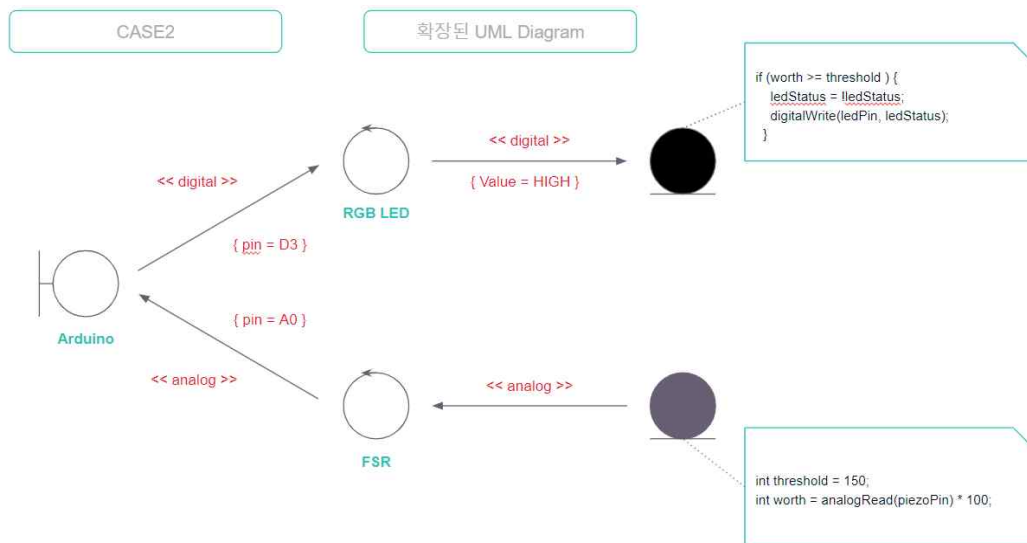


그림 9. CASE 2 확장된 UML Diagram

[그림 9] Diagram을 통해 다음과 같은 정보를 얻을 수 있다.

- 보드에 압전센서가 입력장치로 연결되고, RGB LED가 출력장치로 연결됨
- 압전센서는 A0에 연결되어 있고, 아날로그를 입력(analogRead)함
- LED는 D3에 연결되어 있고, 디지털 값을 출력(digitalWrite)받음

3. 보행 안전 시스템 구현

3.1. 하드웨어 구현

3.1.1. 전체 시스템 구현

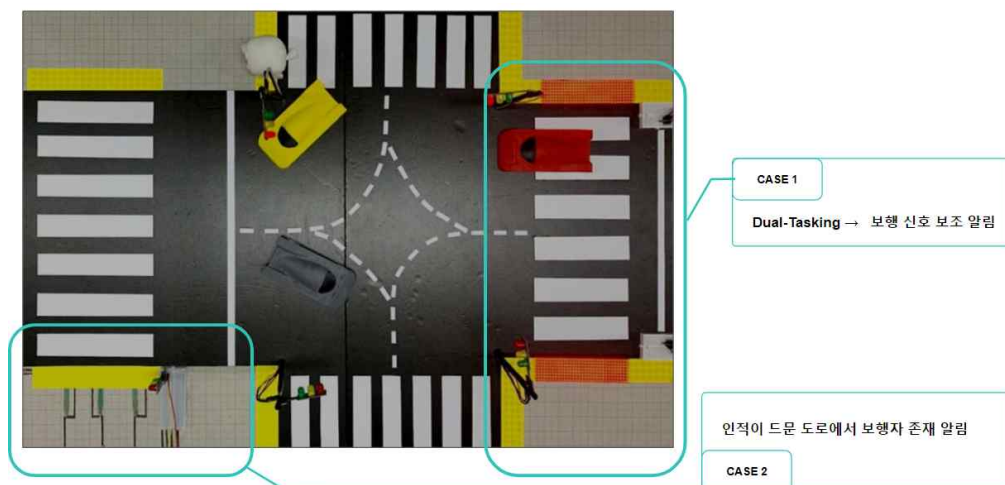


그림 10. 전체 시스템 구현

[그림 10]은 보행 안전 시스템 전체 시스템을 구현한 것이다. 우측에 CASE 1을 구현하였으며, 좌측 하단에 CASE2를 구현하였다. [그림 10]이 상황은 우측 보행 신호등이 빨간불일 때 도로 상황을 나타낸 것으로, Dot Matrix LED에 전부 ON 되어 있는 모습을 볼 수 있다.

3.1.2. CASE 1 : Dual-Tasking 구현

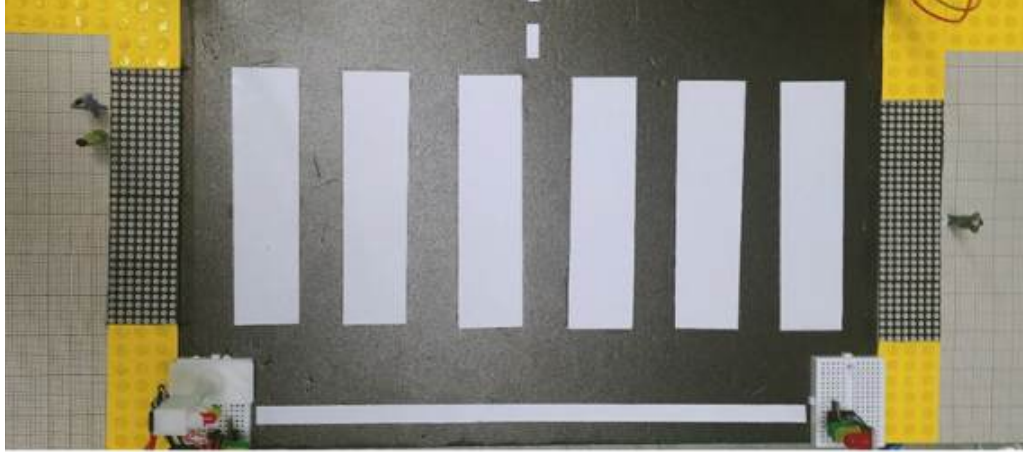


그림 11. Dual-Tasking 구현

[그림 11]은 CASE 1을 구현한 것이다. 오른쪽은 단일 Tasking을 표현하였으며, 왼쪽은 대화를 하는 Dual-Tasking을 표현하였다. 점자 블록 사이에 Dot LED를 설치하여 Dual-Tasking 상황에서도 명확히 확인할 수 있도록 바닥에 설치하였다.

3.1.3. CASE 2 : 인적이 드문 도로 구현

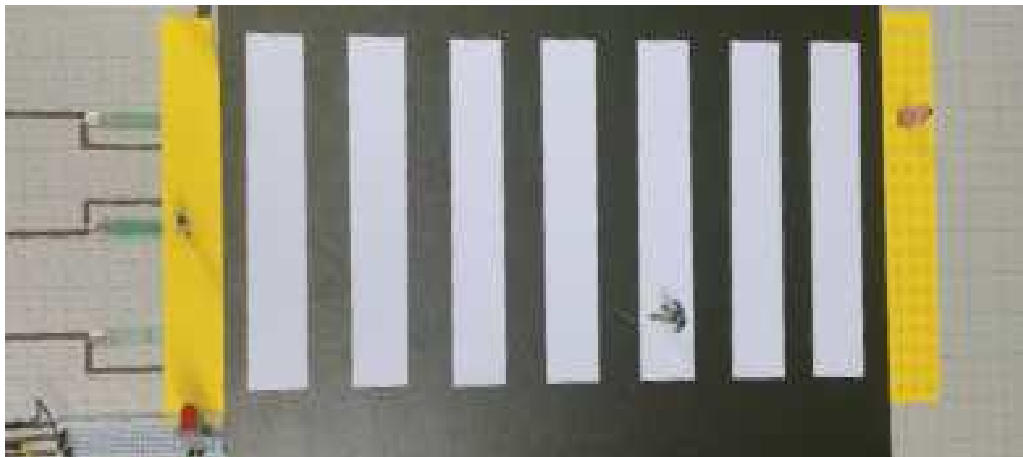


그림 12.인적이 드문 도로 구현

[그림 12]은 CASE 2를 구현한 것이다. 오른쪽은 제안한 방법으로 점자 블록에 압전센서를 설치한 것을 표현하였으며, 왼쪽은 오른쪽과 대비를 주어 기존 시스템인 점자 블록만 있는 상태를 표현한 것이다.

3.2. 소프트웨어 구현

3.2.1. CASE 1 동작 코드

```
void loop(){

    if(value > 0) {
        for(i = 0; i < 4; i++) { // 첫줄 출력
            for(col = 0; col < 8; col++){
                lc.setLed(i, 0, col, true);
            }
        }

        Grean();
        // 보행자 신호등이 초록색일 때 코드
        /*
    void Grean(){
        digitalWrite(Gled, HIGH);
        for (row = 1; row < 3; row++){ // col 출력
            for(col = 0; col < 8; col++){
                for (i = 0; i < 4; i++) { // row 출력
                    lc.setLed(i, row, col, true);
                }
            }
        }
        delay(2000); // 딜레이 2000ms
    }
    digitalWrite(Gled, LOW);
    */

        Orange();

        Red();

        clean();
    }
}
```

그림 13. CASE 1 loop 함수와 초록색일 때 코드 주석

[그림 13]은 CASE 1 loop 함수와 보행 신호가 초록색일 때 코드이다. 첫 if문은 전력이 들어오면 항상 첫줄이 ON 되도록 구현한 것이다. 보행 신호가 초록색, 주황색, 빨간색 순서로 켜지기 때문에 이와 같은 순서로 실행되도록 하였다. 모든 신호 코드는 색만 다를 뿐이기 때문에 초록색 코드를 대표적으로 나타내었다. 중첩 for문을 통해 시간이 증가함에 따라 LED도 동시에 증가하도록 하였다. 설정한 시간 간격에 맞도록 딜레이를 주어 색이 변하는 주기를 설정하였다.

3.2.2. CASE 2 동작 코드

```
#define piezoPin A0 // 압전센서 의 아날로그 단자
#define ledPin 3

int worth = 0; // 현재 측정 된 값
int ledStatus = LOW; // LED 현재 상태 (OFF) (충격 이벤트)

int threshold = 150;

void setup() {
  pinMode(ledPin, OUTPUT); // 핀 3번 LED 출력핀 설정
  Serial.begin(9600); // PC로 전송
}

void loop() { // 셋팅후 연속적 실행

  worth = analogRead(piezoPin) * 100; // A0 핀 값을 판독
  Serial.println(worth);
  // 임계 값 초과시 이벤트 발생
  if (worth >= threshold ) {
    ledStatus = !ledStatus; // LED 상태 (1/0 =시작/종료)
    digitalWrite(ledPin, ledStatus); // 핀 3번 으로 실행
    Serial.println(worth); // PC로 값 보내기
    delay(2000);
  }
  ledStatus = LOW;
  digitalWrite(ledPin, ledStatus);
  delay(200); // 시리얼 모니터 프린팅 시간
}
```

그림 14. CASE 2 loop 함수와 압전 센서 코드 주석

[그림 14]은 CASE 2 전체 동작 코드이다. 압전 센서에서 값을 받아 곱하기 100을 해준 값이 150이 넘을 시 LED에 불빛이 들어오도록 구현하였다. 150이라는 숫자는 임의의 임계 값으로 테스트 결과 작은 물건을 제외하고 고양이와 같은 동물까지 감지할 수 있는 무게로 판단하여 150으로 정하였다. CASE 2는 인적이 드문 도로 상황이므로 인적이 드문 도로는 야생 동물들이 활동할 가능성이 높다 판단하여 동물까지 값을 받는 변수로 설정하였다.

4. 보행 안전 시스템 결과

4.1 전체 시스템

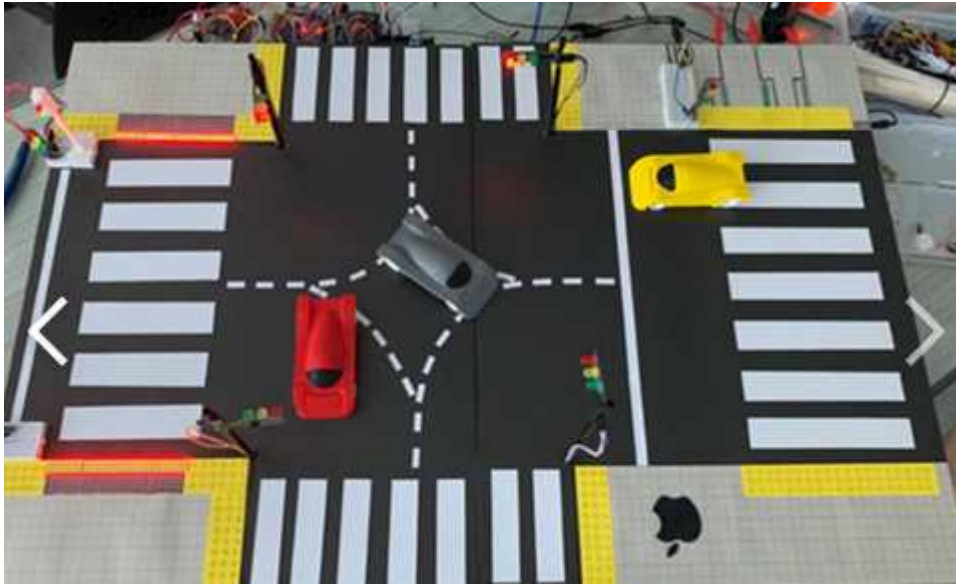


그림 15. 보행 안전 전체 시스템

[그림 15]는 보행 안전 시스템의 전체적인 흐름을 알 수 있다. 전체적으로 보면 case1의 신호의 길이가 긴 사거리에서 스마트폰을 이용하는 보행자를 위한 시스템과 case2는 인적이 드물어 신호등이 존재하지 않아 압전센서를 이용하여 임계치로 사람 혹은 동물까지도 운전자가 볼 수 있도록 설치한 것이다.

4.2. CASE 1

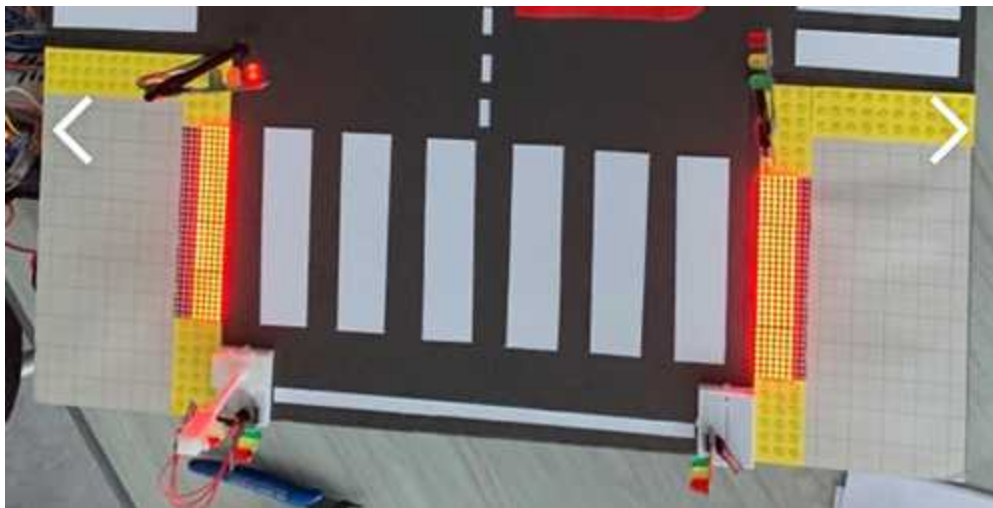


그림 16. CASE 1의 도트 LED를 이용한 신호등

[그림 16]는 CASE 1의 상황을 나타낸 것이다. 도트 LED를 도로가 아닌 점자블럭에 설치하여 도로와 인도의 경계선을 확실히 운전자에게 알려줄 수 있고, 보행자에게는 3단계의 점진적인 시각적 알림으로 Dual-Tasking 보행자에게도 확실한 신호를 보여줄 수 있다. 또한 정지 예고의 알림인 노란색 불을 추가하여 보행자에게 신호의 주의를 알려준다.

4.3. CASE 2

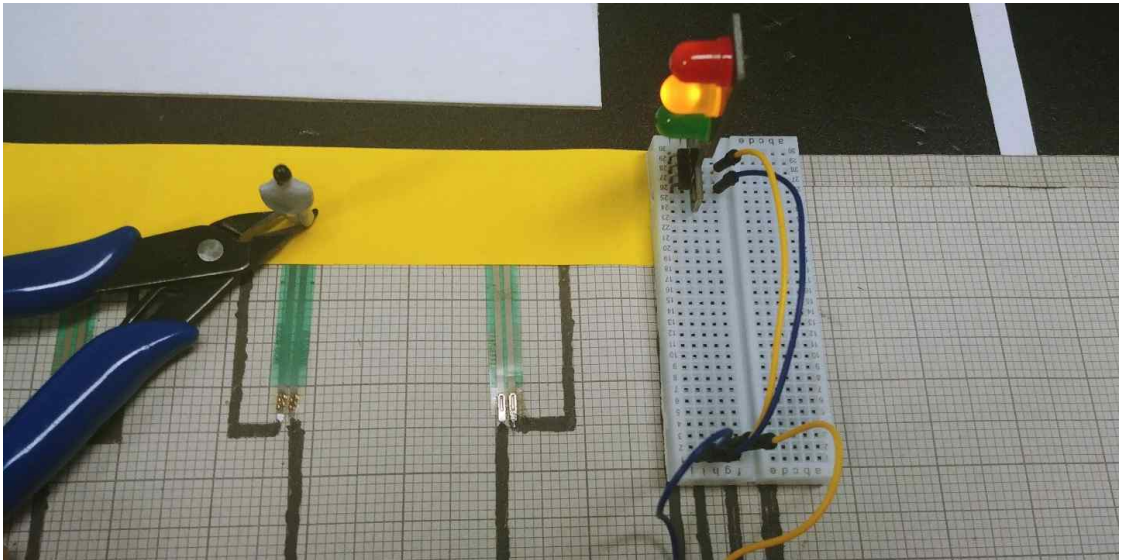


그림 17. CASE 2의 인적이 드문 도로의 압전센서 이용한 LED 알림

[그림 17]은 CASE 2의 상황을 나타낸 것이다. 인적이 드물어 신호등이 없는 도로에 압전 센서를 설치하여 신호등을 건널 가능성이 있는 보행자와 뿐만 아니라 야생 동물까지도 볼 수 있도록 운전자에게 LED로 알려주는 것이다.

5. 설계 구성요소 및 제한요소

[표 4] 설계 구성요소

설계 구성요소	내용
합성	<ul style="list-style-type: none"> ➤ 신호등과 연계된 도트 LED 알림 ➤ 압전센서를 이용한 LED 알림
분석	<ul style="list-style-type: none"> ➤ 하드웨어적 관점 → 도트 LED, 압전센서 이용 ➤ 소프트웨어적 관점 → 신호시간과 임계치 분석
제작	<ul style="list-style-type: none"> ➤ CASE 1 제작 -> CASE 2 제작 -> 신호등 & 압전센서 → 전체적인 신호 흐름 제어
시험	<ul style="list-style-type: none"> ➤ 전도성 펜을 이용한 압전센서와 LED 알림 제어 가능 유무 ➤ 전체적인 흐름 제어 가능 유무
평가	<ul style="list-style-type: none"> ➤ 하나의 아두이노를 이용하지 못한 시간 흐름 제어 부족 ➤ 하드웨어적인 관점과 소프트웨어적인 관점 융합 필요
기타	<ul style="list-style-type: none"> ➤ CASE 3인 열감지 센서 이용한 LED 분석 필요

[표 5] 설계 제한요소

설계 제한요소	내용
원가	<ul style="list-style-type: none"> ➤ 예산 장비는 50만 원인 것에 비해 실제 환경에 사용하려면 추가적인 예산 필요
안정성	<ul style="list-style-type: none"> ➤ 하드웨어 관점에서 아두이노 기반 시스템 설계
신뢰성	<ul style="list-style-type: none"> ➤ 신호의 데이터를 받아 도트 LED의 알림 추가
미학	<ul style="list-style-type: none"> ➤ 실제 도로 환경 기반한 도로 구축 ➤ 도트 LED를 이용한 신호 시간 알림과 압전센서를 이용한 LED 알림을 주도록 설계
내구성	<ul style="list-style-type: none"> ➤ 3D 프린팅 출력 시 필라멘트 내구성과 설정에 따라 결정됨
기타	<ul style="list-style-type: none"> ➤ 신호 알림 추가 기능과 압전 센서를 이용한 LED 알림으로 사람들의 인식 개선

[표 4], [표 5]는 설계 구성요소 및 제한요소를 나타낸 것이다. [표 4]는 합성, 분석, 제작, 시험, 평가로 구성요소를 구분하여 분석 및 정리하였고, [표 5]는 원가, 안정성, 신뢰성, 미학, 내구성 요소로 구분하여 분석 및 정리하였다.

III. 결론

1. 과제 결과 요약

보행자/운전자 관점에서 시각적 자극 기반 시스템이 구축되면 보행자들의 안전함이 확보되어 교통사고의 보행 사망자 점유율이 줄어들 것으로 예상된다. 보행사망자가 가장 많이 일어나는 지역에 시스템을 시범적으로 설치를 하여 시스템의 결과를 예측해보고 전국적으로 시행될 수 있을 것이다. 기술적 측면에서는 숫자, 도형 뿐만 아니라 단어나 문장을 이용하여 다양한 관점에서 복합적인 보조 알림 서비스를 제공할 수 있다. 경제적 측면으로는 보조 알림을 추가함으로써 의료보험비, 교통사고 처리 비용 등 절감될 것으로 추정할 수 있다. 행위자 측면에서는 보행자 입장에서는 Dual-tasking 환경에서 안전하게 보행을 할 수 있고 운전자 입장에서는 야간 환경에서 보행자를 인식하기가 수월해짐으로써 사고율이 줄어들게 된다.

2. 개선 방안 및 향후 과제 방향

보행자/운전자 관점에서 시각적 자극 기반 시스템이 구축되면 보행자들의 안전함이 확보되어 교통사고의 보행 사망자 점유율이 줄어들 것으로 예상된다. 보행사망자가 가장 많이 일어나는 지역에 시스템을 시범적으로 설치를 하여 시스템의 결과를 예측해보고 전국적으로 시행될 수 있을 것이다. 기술적 측면에서는 숫자, 도형 뿐만 아니라 단어나 문장을 이용하여 다양한 관점에서 복합적인 보조 알림 서비스를 제공할 수 있다. 경제적 측면으로는 보조 알림을 추가함으로써 의료보험비, 교통사고 처리 비용 등 절감될 것으로 추정할 수 있다. 행위자 측면에서는 보행자 입장에서는 Dual-tasking 환경에서 안전하게 보행을 할 수 있고 운전자 입장에서는 야간 환경에서 보행자를 인식하기가 수월해짐으로써 사고율이 줄어들게 된다.

3. 설계 목표의 중요도 및 달성도

[표 7] 설계 목표의 중요도 및 달성도

목표	중요도(%)	달성도(%)	수행내용
기반 시스템 설계	50	50	아두이노 회로도, 압전 센서, 적외선 열감지 센서, LED 도트 설계
기반 기술 구현	35	30	전체적인 신호 흐름 제어 부족
테스트 및 오류 수정	10	5	신호등의 세기 약함
외관 제작	15	15	실제 도로 환경을 기반으로 구축
합계	100		

IV. 참고문헌

- [1] 도로교통공단. 교통사고통계보고서, "taas.koroad.or.kr/web/bdm/srs/selectStaticReportsList.do?menuId=WEB_KMP_IDA_SRS_TAD, 06.25, 2019
- [2] 김동준. (2012). 노인, 보행교통사고 발생 시 76%가 접근차량 응시 안 해. 월간교통, (), 107-107.
- [3] 대법원 2001. 10. 9. 선고 2001도2939 판결








[부 록 1]

➤ UML을 이용한 아두이노 어플리케이션 설계

아두이노 어플리케이션은 C언어를 기반으로 개발되기 때문에 프로그램의 효과적인 설계 양식을 제공하지 못하는 문제점이 있다. 선행 연구에서는 기존의 이해하기 힘든 구조도 (Structure Chart)를 UML의 확장 매커니즘을 이용한 아두이노 어플리케이션 설계방법을 제안하였다. 여기서 UML은 객체 기술에 대한 표준화 기구에서 인정한 객체지향 분석, 설계를 위한 모델링 언어(모델을 표현하는 언어)이다.

UML에서 스테레오타입은 기존의 메타 클래스에서 추가적인 의미를 부여한 시각적인 표기법을 정의한 하나의 모델요소이다. 스테레오타입은 클래스, 관계, Use-Case, Actor 등 UML의 모든 구성요소에 적용할 수 있으며, 스테레오타입을 표기할 때에는 모델요소에 "<<스테레오타입 명 >>"형식을 붙여 확장된 의미를 부여한다.

아두이노 어플리케이션 설계를 위한 스테레오타입과 아이콘, 태그는 아래의 그림과 같이 정의한다.

스테레오타입	이미지
<<Arduino>>	 아두이노 보드를 표시
<<Digital Input>>	 디지털/아날로그 입출력 표시
<<Digital Output>>	
<<Analog Input>>	
<<Analog Output>>	
<<Value>>	 디지털값 : HIGH  디지털값 : Low  아날로그값 : 0 ~ 255
<<digital>>	
<<analog>>	

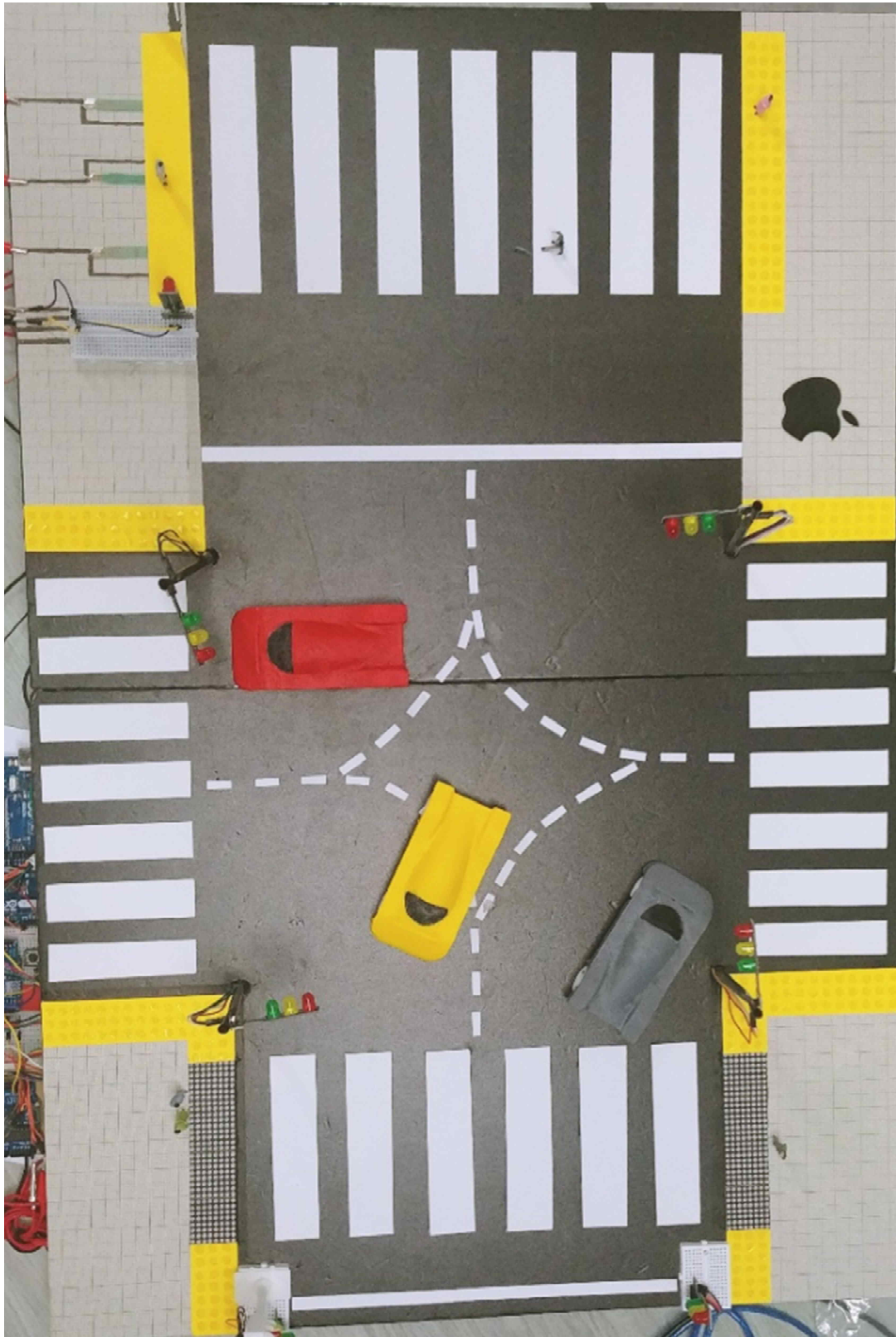
확장된 UML의 스테레오타입과 아이콘(a)

태그	적용대상	설명
pin	아두이노와 입출력 장치 사이의 연관 (Association)	<ul style="list-style-type: none"> pinMode(), digitalWrite(), analogRead/Write()에 사용되는 파라미터 pin 번호를 표시 아두이노와 입출력 장치와 물리적으로 연결되는 핀 번호를 의미
value	출력장치와 출력값 사이의 연관 (Association)	<ul style="list-style-type: none"> 출력 값을 나타내기 위한 조건 표시

확장된 UML의 태그 정의(b)

[부 록 2]

➤ 전체 구현 모습



[부 록 3]

➤ 전체 제어 코드

```
#define Traffic_1R 1
#define Traffic_1G 2
#define Traffic_1B 3
#define Traffic_2R 4
#define Traffic_2G 5
#define Traffic_2B 6
#define Traffic_3R 7
#define Traffic_3G 8
#define Traffic_3B 9
#define Traffic_4R 10
#define Traffic_4G 11
#define Traffic_4B 12

#define Walking_light_1R 13
#define Walking_light_1G 14
#define Walking_light_1B 15
#define Walking_light_2R 16
#define Walking_light_2G 17
#define Walking_light_2B 18

#define case_1 21

int i = 0;
unsigned long pr_time = 0;

void setup() {

    pinMode(Traffic_1R, OUTPUT);
    pinMode(Traffic_1G, OUTPUT);
    pinMode(Traffic_1B, OUTPUT);
    pinMode(Traffic_2R, OUTPUT);
    pinMode(Traffic_2G, OUTPUT);
    pinMode(Traffic_2B, OUTPUT);
    pinMode(Traffic_3R, OUTPUT);
    pinMode(Traffic_3G, OUTPUT);
    pinMode(Traffic_3B, OUTPUT);
    pinMode(Traffic_4R, OUTPUT);
    pinMode(Traffic_4G, OUTPUT);
    pinMode(Traffic_4B, OUTPUT);

    pinMode(Walking_light_1R, OUTPUT);
    pinMode(Walking_light_1G, OUTPUT);
    pinMode(Walking_light_1B, OUTPUT);
    pinMode(Walking_light_2R, OUTPUT);
    pinMode(Walking_light_2G, OUTPUT);
    pinMode(Walking_light_2B, OUTPUT);
    pinMode(case_1, OUTPUT);

    digitalWrite(Traffic_1R, HIGH);
    digitalWrite(Traffic_2R, HIGH);
    digitalWrite(Traffic_3R, HIGH);
    digitalWrite(Traffic_4R, HIGH);
    digitalWrite(Walking_light_1R, HIGH);
    digitalWrite(Walking_light_2R, HIGH);
    digitalWrite(case_1, HIGH);

}

void low() {
    for(i = 1; i < 19; i++) {
        if(i % 3 == 1){
            continue;
        } else {
            digitalWrite(i, LOW);
        }
    }
}
```

```

void loop() {

    unsigned long current_time = millis() - pr_time;
    low();

    if(current_time <= 10000) {
        digitalWrite(Traffic_2R, LOW);
        digitalWrite(Traffic_2G, HIGH);
        digitalWrite(Walking_light_1R, LOW);
        digitalWrite(Walking_light_1G, HIGH);
    } else if(current_time <= 13000) {
        digitalWrite(Traffic_2B, HIGH);
        digitalWrite(Walking_light_1B, HIGH);
    } else {
        low();
        digitalWrite(Traffic_2R, HIGH);
    }

    if(current_time > 13000 && current_time <= 23000) {
        digitalWrite(Traffic_3R, LOW);
        digitalWrite(Traffic_3G, HIGH);
    } else if(current_time > 23000 && current_time <= 26000 ) {
        digitalWrite(Traffic_3B, HIGH);
    } else {
        low();
        digitalWrite(Traffic_3R, HIGH);
    }

    if(current_time > 26000 && current_time <= 36000) {
        digitalWrite(Traffic_4R, LOW);
        digitalWrite(Traffic_4G, HIGH);
    } else if(current_time > 36000 && current_time <= 39000 ) {
        digitalWrite(Traffic_4B, HIGH);
    } else {
        low();
        digitalWrite(Traffic_4R, HIGH);
    }

    if(current_time > 39000 && current_time <= 49000) {
        digitalWrite(Traffic_1R, LOW);
        digitalWrite(Traffic_1G, HIGH);
    } else if(current_time > 49000 && current_time <= 52000 ) {
        digitalWrite(Traffic_1B, HIGH);
    } else {
        low();
        digitalWrite(Traffic_1R, HIGH);
    }
    pr_time = current_time;
}

```


[부 록 4]

➤ CASE 1 전체 Code

```
#include "LedControl.h"

// DIN 핀을 12번에 CS 핀을 10번에 CLK핀을 11번에 연결해줌
// (DIN, CLK, CS, 연결할 도트 매트릭스의 개수)
LedControl lc=LedControl(12,11,10,4);

#define Rled 5
#define Yled 6
#define Gled 7

#define start 3

int row, col, i;
int value = 0;

void setup()
{
    pinMode(start, INPUT);
    for(i=0; i<4; i++){ // 도트 매트릭스 0~3번
        lc.shutdown(i,false); // 디스플레이 초기화
        lc.setIntensity(i,5); // 도트 매트릭스 밝기 (매트릭스 번호, 밝기) 1~15
        lc.clearDisplay(i); // led 를 전체 꺼주는 함수
    }
    value = digitalRead(start);
}

// 전체led를 꺼주는 함수
void clean(){
    for(i = 0; i < 4; i++)
        lc.clearDisplay(i);// clear screen
}

// 보행자 신호등이 빨간색일 때
void Red(){
    digitalWrite(Rled, HIGH);
    for(i = 0; i < 8; i++)
        for (col = 0; col < 8; col++){ // col 출력
            for(i = 0; i < 8; i++){
                for (row = 0; row < 8; row++) { // row 출력
                    lc.setLed(i, col, row, true);
                }
            }
        }
    delay(30000);
    digitalWrite(Rled, LOW);
}
```

```

// 보행자 신호등이 초록색일 때
void Grean(){
    digitalWrite(Gled, HIGH);
    for (row = 1; row < 3; row++){      // col 출력
        for(col = 0; col < 8; col++){
            for (i = 0; i < 4; i++) {    // row 출력
                lc.setLed(i, row, col, true);
            }
        }
    }
    delay(2000); // 딜레이 2000ms
}
digitalWrite(Gled, LOW);
}

```

```

// 보행자 신호등이 주황색일 때
void Orange(){
    digitalWrite(Yled, HIGH);
    for (row = 3; row < 6; row++){      // col 출력
        for(col = 0; col < 8; col++){
            for (i = 0; i < 4; i++) {    // row 출력
                lc.setLed(i, row, col, true);
            }
        }
    }
    delay(1550); // 딜레이 1600ms
}
delay(100); // 딜레이 100ms
digitalWrite(Yled, LOW);
}

```

```

void loop(){

    if(value > 0) {
        for(i = 0; i < 4; i++) { // 첫줄 출력
            for(col = 0; col < 8; col++){
                lc.setLed(i, 0, col, true);
            }
        }

        Grean();
        Orange();
        Red();

        clean();
    }
}

```


[부 록 5]

➤ 사용 핵심 부품

부품	기능
	Arduino MEGA • 시스템 컨트롤러
	FSR-420 • 압전 센서
	32 x 8 Dots Matrix LED • 보행자 보조 알림
	IR Sense 2 click • 적외선 열감지, 인체 감지