		<h2 style="text-align: center;">Capstone Design 과제 결과보고서</h2>		
과 목 명	종합설계프로젝트			
과 제 명	시각 장애인을 위한 보행 보조 장치			
팀 명	A급 인재들			
지도교수 (과제책임자)	학과(부)	전자전기	성명	조준동
팀장 (대표학생)	학과(부)/학년	전자전기공학부	성명	정재관
	E-mail	<a href="mailto:nesalang6@naver.com">nesalang6@naver.com</a>	휴대전화	01098705078
팀원	학과(부)/학년	전자전기공학부	성명	금혜란
	학과(부)/학년	전자전기공학부	성명	배기문
	학과(부)/학년	전자전기공학부	성명	서승현
	학과(부)/학년	전자전기공학부	성명	김영민
	학과(부)/학년	전자전기공학부	성명	전우성
	학과(부)/학년	전자전기공학부	성명	김석진
	학과(부)/학년	전자전기공학부	성명	최명준
프로젝트 지원기업	기업명	-	업종	-
	주소	-		
과 제 수 행 비	구 분	현 금	현 물	
	국 비(LINC 3.0)	500,000원		
	기 업 체	-		
	합 계	500,000원		
과제기간	2023 년 8 월 26 일 - 2023년 12 월 15 일 ( 4개월)			
<p>위와 같이, 성균관대학교 산학협력 선도대학 육성사업(LINC 3.0) Capstone Design 지원사업 과제결과보고서를 제출합니다.</p> <p style="text-align: right;">2023 . 12 . 10 .</p> <p style="text-align: right;">팀 장 : 정 재 관 <span style="border: 1px solid black; padding: 2px;">정재관</span></p> <p style="text-align: right;">지도교수 : 조 준 동 (인)</p> <p style="text-align: center; font-size: 1.2em; font-weight: bold;">LINC 3.0 사업단장 귀하</p>				

## Capstone Design 과제 결과보고서 요약

과 제 명	아두이노 보드와 AI 학습 모델 기반의 보행 보조 장치
팀 구성원 및 역할	<div>정재관 - 디자인</div> <div>서승현 - 기획</div> <div>배기문 - 디자인</div> <div>금혜란 - 기획</div> <div>김석진 - 기획</div> <div>김영민 - 기획, 개발</div> <div>최명준 - 기획</div> <div>전우성 - 개발</div>
개발동기 목적 및 필요성	<p><b>1. Discovery</b></p> <p>장애가 있는 분들은 삶을 살아가는 데에 있어 많은 어려움을 겪는다. 따라서 ‘A급 인재들’ 팀은 해당 프로젝트를 진행하며 장애우들의 향상된 삶을 도모하고자 한다.</p> <p>우선, Discovery 과정을 통해 시각 장애우들이 일상생활에서 겪고 있는 구체적인 불편한 점을 발견하고 선행 물품을 조사하였다.</p> <p><b>1-2. 시각 장애인</b></p> <p>시각 장애인들은 일상생활에서 안전상의 문제와 다양한 불편함을 겪고 있었다. 보행 시, 지팡이가 있더라도 장애물 혹은 주변 사람들과 부딪힐 수 있고 공중에 있는 장애물과 충돌할 가능성이 언제나 존재했다. 또한, 최근 많은 가게에 키오스크가 도입되고 있어 이를 이용하는 데에도 불편하였다.</p> <p>이를 해결하기 위한 선행 물품으로는 자율 주행 시스템을 탑재한 ‘길 안내 및 장애물 센서 장치’, 장애우들을 위해 키오스크의 높낮이를 조절해주는 ‘배리어프리 키오스크’ 등이 있다.</p> <p>팀 논의를 통해 시각 장애인들의 보행에 도움을 주는 장치를 설계하기로 결정하였다.</p> <p><b>2. Concept Design</b></p> <p>‘시각 장애인의 보행 중 사고위험’을 주제로 더욱 구체적인 상황, 자료조사, 디자인의 Concept을 정하였다.</p> <p>‘계단이나 턱이 있을 때 걸려 넘어지기 쉬움’, ‘공중에 떠 있는 장애물 회피가 어려움’, ‘길 안내에 대한 정보가 없으면 목적지로 이동하는 것이 어려움’, ‘점자 안내표기와 선형유도 블록의 부재’ 등의 불편한 상황을 발견하였다.</p> <p>이의 자료조사 결과, 시각 장애인의 경우 이동수단이 특별교통수단인 ‘장애인 택시’와 ‘보행’으로 국한되었다. 더불어, 국토교통부의 연구에 따르면 시각 장애인의 횡단보도 보행 환경이 열악한 것으로 나타났다. 시각 장애인의 보행을 도와주는 점자블록이 잘못된 방향으로 설치되어 있거나 파손되어 기능을 못 하는 경우도 많았다.</p> <p>따라서, 본 프로젝트를 진행하여 시각 장애 보행자의 점자블록이 제공하는 정보를 파손될 수 있는 공공도로가 아닌 개인의 신발을 통해 제공할 수 있도록 장치를 개발하기로 하였다. 그리고 보행자 앞에 장애물이 있는지 확인하여 장애물과의 거리를 알려줄 수 있도록 하였다.</p> <p>이를 활용하여 시각 장애 보행자의 안전을 도모하고자 한다.</p>

과제 해결 방안  
및 과정

## 1. 구현 방법

### 1-1. AI 모델

AI 모델은 총 4가지 단계를 통해 학습된다.

첫 번째는 “Data Collection Code 작성” 단계이다. 와이파이로 PC와 핸드폰을 연결한 후 핸드폰 카메라상의 영상을 실시간으로 PC 화면에 띄우고 화면을 클릭하는 순간, 사진이 찍혀 Data를 수집할 수 있는 코드를 작성하였다.

두 번째는 “Data Collection” 단계이다. AI 모델을 학습시킬 Data Set을 구축하는 단계이다. 우리 팀은 보행자가 더욱 다양한 길을 거닐 수 있게 하도록 실내와 실외 두 곳에서 Data를 수집하였다. 카메라가 신발에 부착되면 낮은 시야에서 도로를 촬영할 것이기 때문에 그에 맞추어 촬영하였다. 또한, 우리의 제품이 보행자가 도로의 중앙에 맞추어 걸을 수 있도록 도움을 줄 것이므로 길의 여러 방향에 치우친 곳에서 촬영하여 다양한 Data를 수집하였다.

세 번째는 “Model Train” 단계이다. 수집한 데이터를 이미지 전처리, 데이터 분할, model transfer를 거쳤다. 그 후 optimizer Adam을 사용해서 epoch 50번을 진행하며 성능이 제일 좋은 모델을 저장하며 학습시켰다.

네 번째는 “Model Evaluation” 단계이다. Test Data를 사용해 모델 결과를 평가하고 성능 개선을 위해 최적화를 진행하였다. 그 후, 실시간으로 아두이노와 Bluetooth 연결하여 카메라를 통해 들어오는 사진을 평가한 결과값을 아두이노로 송신하였다.

### 1-2. 카메라 / 초음파 센서

카메라는 신발에 부착되어 시각 장애인의 눈이 되어 준다. 실시간으로 도로를 촬영하여 Bluetooth 모듈을 통해 Google Co-Lab의 AI Model로 촬영한 사진을 보내준다. 초음파 센서는 실시간으로 장애물을 감지한다. 50cm 이내에 장애물이 있다면 Arduino board에 신호를 전송한다.

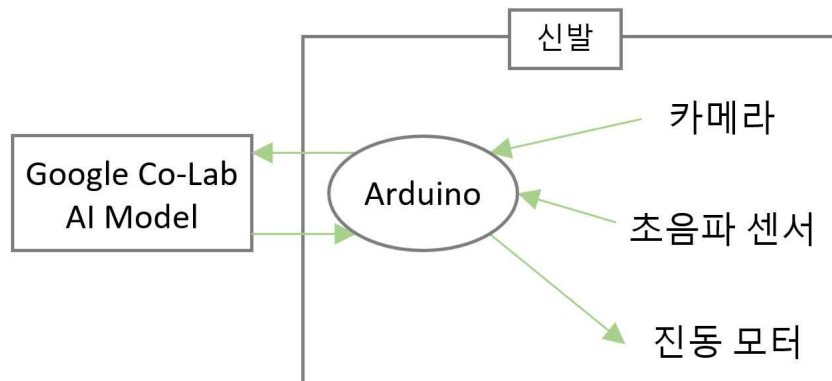
### 1-3. Arduino

Arduino nano(Arduino board)와 HC-06 Bluetooth 모듈을 사용하였다. Bluetooth 모듈은 카메라의 영상과 AI 모델을 통한 결과값의 송수신을 담당한다. Arduino board는 AI 모델 결과값과 초음파 센서값을 종합하여 신발에 부착되는 진동 모터를 제어한다.

### 1-4. 신발

최종적으로 신발에 모든 부품을 장착하였다. 초음파 센서와 카메라는 신발의 앞코 부분에 들어가 보행자가 걸어야 할 도로를 촬영하거나 장애물을 감지하였다. Arduino와 송수신을 담당할 Bluetooth 모듈 그리고 9V 배터리는 신발의 외부 측면에 부착하여 추후 신발이 개발되었을 때 눈에 보이는 부분을 꾸밀 수 있게 해주었다.

## 2. 시스템 구성



	<p>전체적인 “Built-in 점자블록” 신발의 시스템은 위의 사진과 같다. 신발에 카메라, 초음파 센서, 진동 모터와 해당 모듈들을 연결하고 신호 송수신을 담당하는 Arduino board가 있다.</p> <p>카메라와 초음파 센서는 실시간으로 작동된다. 카메라는 실시간으로 보행자 앞의 도로를 촬영하여 Bluetooth 모듈을 통해 Google Co-Lab으로 전송한다. 초음파 센서도 실시간으로 장애물과의 거리를 측정하여 Arduino board로 전송한다. Arduino board는 많은 모듈을 연결하는 동시에 블루투스를 통해 데이터의 송수신을 담당한다.</p> <p>Google Co-Lab에 작성된 AI Model은 교내·외 도로의 사진을 촬영한 뒤 약 400장의 데이터로 학습되었다. 학습된 AI Model은 Arduino board로부터 전송된 사진을 받는다. 촬영된 사진과 학습된 여러 데이터를 비교하여 현재 촬영된 사진이 도로의 중앙으로부터 얼마나 벗어났는지를 계산한다. AI Model이 도로의 중앙을 인식할 수 있게끔 학습시키기 때문에 이러한 계산이 가능하다. 현재 촬영된 사진이 좌 혹은 우로 얼마만큼 움직여야 하는지 계산하여 “강하게 왼쪽으로 움직이기”, “약하게 왼쪽으로 움직이기”, “직진하기”, “약하게 오른쪽으로 움직이기”, “강하게 오른쪽으로 움직이기” 총 5가지 경우의 수를 제공한다. 해당 경우를 Bluetooth 모듈에 제공한다.</p> <p>Arduino board는 AI Model에서 계산된 경우의 수 그리고 초음파 센서에서 장애물 유무를 확인한 값을 제공받는다. 이 값들을 종합하여 진동 모터를 제어하게 된다. “강하게 왼쪽으로 움직이기”의 경우 좌측 진동 모터 2개가 진동한다. “약하게 왼쪽으로 움직이기”의 경우 좌측 진동 모터 1개가 진동한다. “직진하기”의 경우 보행에 문제가 없으므로 진동 모터들이 진동하지 않는다. “약하게 오른쪽으로 움직이기”의 경우 우측 진동 모터 1개가 진동한다. “강하게 오른쪽으로 움직이기”의 경우 우측 진동 모터 2개가 진동한다. 마지막으로 초음파 센서에서 50cm 이내에 장애물이 있다고 판단한 경우인 “멈추기”의 경우 모든 진동 모터가 강하게 진동한다.</p> <p>일련의 과정을 통해 점자블록이 제공하는 정보인 도로 중앙길 알림과 직진, 멈춤 등을 제공할 수 있고 추가로 장애물을 감지하여 보행자에게 알려준다.</p>						
<p>★대표적 결과물 (도면, 사진 등)</p>							
<p>산업체 연계 활동 내역 요약</p>	<input type="checkbox"/> 프로젝트 주제발굴	<input type="checkbox"/> 멘토링	<input type="checkbox"/> 프로젝트 수행	<input type="checkbox"/> 샘플(부품) 제공	<input type="checkbox"/> 프로젝트 실습비 지원	<input type="checkbox"/> 현장학습	<input type="checkbox"/> 기타
<p>해당 없음</p>							

## [서식13-3]

### 1. 과제 개요

과제명 : Built-in 점자블록 (아두이노 보드와 AI 학습 모델 기반의 보행 보조 장치)

참여자 : 금혜란, 김석진, 김영민, 배기문, 서승현, 전우성, 정재관, 최명준

지도교수 : 조준동 교수님

역할 담당 : 디자인 (정재관, 배기문), 기획(김석진, 최명준, 서승현, 금혜란), 개발(김영민, 전우성)

참여기업 : 해당 없음

시각 장애인의 경우 이동수단이 '특별교통수단'과 '보행'으로 비교적 한정되어 있다. 국토교통부의 연구에 따르면 시각 장애인의 횡단보도 보행 환경이 열악한 것으로 나타났다. 시각 장애인의 보행을 도와주는 점자블록이 잘못된 방향으로 설치되어 있거나 파손되어 기능을 못 하는 경우도 많았다. 따라서 본 연구는 사용자의 신발에 카메라, 센서, 모터를 아두이노와 함께 연결하고 AI 모델 기반의 연산을 통해 보행에 필요한 정보를 제공하는 장치를 구현할 것이다.

### 2. 과제 추진배경(개발동기) 및 목적, 필요성

#### 2-1. Discovery

장애가 있는 분들은 삶을 살아가는 데에 있어 많은 어려움을 겪는다. 따라서 'A급 인재들' 팀은 해당 프로젝트를 진행하며 장애우들의 향상된 삶을 도모하고자 한다.

우선, Discovery 과정을 통해 시각 장애우들이 일상생활에서 겪고 있는 구체적인 불편한 점을 발견하고 선행 물품을 조사하였다.

##### 2-1-1. 청각·언어 장애인

청각 장애인은 소리를 듣지 못해 안전상의 문제가 발생할 수 있다. 도로에서 주변 소리를 듣지 못하기 때문에 안전상의 문제가 존재한다. 또한, 운전할 때 주변 음성 환경을 알 수 없기 때문에 전방주시예 방해를 받고, 대중교통을 이용할 때에도 정류장 도착 정보를 파악하기가 어려웠다.

이를 해결하기 위한 선행 물품으로는 음성을 자막을 바꾸어 안경에 띄워주는 '스마트 자막 안경', 횡단보도에서 보행 신호를 알려주는 '횡단보도 LED' 등이 있다.

언어 장애인은 정보 전달에 어려움을 겪는다. 발음의 명료도가 떨어져 시끄러운 곳에서 의사소통이 어렵다. 병원에서 진료하거나 위급 상황이 발생하더라도 상태를 명확하게 말할 수 없어 도움을 받기 어렵다.

이를 해결하기 위한 선행 물품으로는 텍스트를 음성으로 변환해주는 어플인 '마이토키'와 눈동자 인식을 통해 글을 생성해주는 'AAC 제품' 등이 있다.

##### 2-1-2. 시각 장애인

시각 장애인들은 일상생활에서 안전상의 문제와 다양한 불편함을 겪고 있었다. 보행 시, 지팡이가 있더라도 장애물 혹은 주변 사람들과 부딪힐 수 있고 공중에 있는 장애물과 충돌할 가능성이 언제나 존재했다. 또한, 최근 많은 가게에 키오스크가 도입되고 있어 이를 이용하는 데에도 불편하였다.

이를 해결하기 위한 선행 물품으로는 자율 주행 시스템을 탑재한 '길 안내 및 장애물 센서 장치', 장애우들을 위해 키오스크의 높낮이를 조절해주는 '배리어프리 키오스크' 등이 있다.

팀 논의를 통해 시각 장애인들의 보행에 도움을 주는 장치를 설계하기로 결정하였다.

#### 2-2. Concept Design

'시각 장애인의 보행 중 사고위험'을 주제로 더욱 구체적인 상황, 자료조사, 디자인의 컨셉을

정하였다.

‘계단이나 턱이 있을 때 걸려 넘어지기 쉬움’, ‘공중에 떠 있는 장애물 회피가 어려움’, ‘길 안내에 대한 정보가 없으면 목적지로 이동하는 것이 어려움’, ‘점자 안내표기와 선형유도 블록의 부재’ 등의 불편한 상황을 발견하였다.

이의 자료조사 결과, 시각 장애인의 경우 이동수단이 특별교통수단인 ‘장애인 택시’와 ‘보행’으로 국한되었다. 더불어, 국토교통부의 연구에 따르면 시각 장애인의 횡단보도 보행 환경이 열악한 것으로 나타났다. 시각 장애인의 보행을 도와주는 점자블록이 잘못된 방향으로 설치되어 있거나 파손되어 기능을 못 하는 경우도 많았다.

따라서, 본 프로젝트를 진행하여 시각 장애 보행자의 점자블록이 제공하는 정보를 파손될 수 있는 공공도로가 아닌 개인의 신발을 통해 제공할 수 있도록 장치를 개발하기로 하였다. 그리고 보행자 앞에 장애물이 있는지 확인하여 장애물과의 거리를 알려줄 수 있도록 하였다.

이를 활용하여 시각 장애 보행자의 안전을 도모하고자 한다.

### 3. 과제 해결 방안 및 과정

#### 3-1. AI 모델

AI 모델은 총 4가지 단계를 통해 학습된다.

첫 번째는 “Data Collection Code 작성” 단계이다. 와이파이로 PC와 핸드폰을 연결한 후 핸드폰 카메라상의 영상을 실시간으로 PC 화면에 띄우고 화면을 클릭하는 순간, 사진이 찍혀 Data를 수집할 수 있는 코드를 작성하였다.

두 번째는 “Data Collection” 단계이다. AI 모델을 학습시킬 Data Set을 구축하는 단계이다. 우리 팀은 보행자가 더욱 다양한 길을 거닐 수 있게 하도록 실내와 실외 두 곳에서 Data를 수집하였다. 카메라가 신발에 부착되면 낮은 시야에서 도로를 촬영할 것이기 때문에 그에 맞추어 촬영하였다. 또한, 우리의 제품이 보행자가 도로의 중앙에 맞추어 걸을 수 있도록 도움을 줄 것이므로 길의 여러 방향에 치우친 곳에서 촬영하여 다양한 Data를 수집하였다.

세 번째는 “Model Train” 단계이다. 수집한 데이터를 이미지 전처리, 데이터 분할, model transfer를 거쳤다. 그 후 optimizer Adam을 사용해서 epoch 50번을 진행하며 성능이 제일 좋은 모델을 저장하며 학습시켰다.

네 번째는 “Model Evaluation” 단계이다. Test Data를 사용해 모델 결과를 평가하고 성능 개선을 위해 최적화를 진행하였다. 그 후, 실시간으로 아두이노와 Bluetooth 연결하여 카메라를 통해 들어오는 사진을 평가한 결과값을 아두이노로 송신하였다.

#### 3-2. 카메라 / 초음파 센서

카메라는 신발에 부착되어 시각 장애인의 눈이 되어 준다. 실시간으로 도로를 촬영하여 Bluetooth 모듈을 통해 Google Co-Lab의 AI Model로 촬영한 사진을 보내준다. 초음파 센서는 실시간으로 장애물을 감지한다. 50cm 이내에 장애물이 있다면 Arduino board에 신호를 전송한다.

#### 3-3. Arduino

Arduino nano(Arduino board)와 HC-06 Bluetooth 모듈을 사용하였다. Bluetooth 모듈은 카메라의 영상과 AI 모델을 통한 결과값의 송수신을 담당한다. Arduino board는 AI 모델 결과값과 초음파 센서값을 종합하여 신발에 부착되는 진동 모터를 제어한다.

#### 3-4. 신발

초기에 보행 보조 장치로 팔찌, 허리띠, 발찌와 신발과 같은 많은 아이디어가 있었다. 그러나 점자블록의 경우 시각 장애인이 그 위를 걸음으로써 제공되는 정보이기 때문에 신발이 프로젝트의 취지에 가장 부합한다는 의견이 있었다. 하지만 장치가 깔창에 들어갔을 경우 보행자의 체중이 가해져 장치가 고장

날 우려가 있으므로 보드를 신발의 외 측면으로 빼고 센서를 신발 앞에 부착하였다. 또한, 진동 모터를 발목 스트랩에 부착하여 보행자가 신호를 알아차릴 수 있게끔 하였다.

따라서, 초음파 센서와 카메라는 신발 앞부분에 부착하고, Arduino board와 Bluetooth 모듈, 9V 배터리는 신발의 외 측면, 진동 센서는 발목 스트랩에 부착된다.



[ 신발 Concept Design]

#### 4. 과제 세부내용 (표, 설계도면, 그림 등 포함)

##### 4-1. AI 모델

##### 4-1-1. Data Collection Code 작성

Wifi를 통해 PC와 핸드폰을 연결한 후 핸드폰 카메라상의 영상을 실시간으로 PC 화면에 띄우고 화면을 클릭하는 순간, 사진이 찍혀 Data를 수집할 수 있는 코드를 작성하였다. 해당 코드를 Google Co-Lab을 통해 실행한다.

[Data Collection Code]

##### DATA COLLECTION

add web cam to capture and save iamge

These code is for collecting image and x coordinate for side walk guide network.

make a directory with name "training\_dataset"

```
[ ] !mkdir training_dataset
```

take\_phone function enables a connection between google colab and local web cam.

java script code is for showing the preview of the webcam and enable a button to stop image capture.

It returns image and x coordinates(0~255).

[Code 설명]



```

[ ] from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode
import os

def take_photo(cnt):
    quality = 80
    stop = False

    js = Javascript("""
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const video = document.createElement('video');
      const button = document.createElement('button');
      button.textContent = 'Stop';
      div.appendChild(button);
      video.style.display = 'block';

      const devices = await navigator.mediaDevices.enumerateDevices();
      const videoDevices = devices.filter(device => device.kind === 'videoinput');

      // const stream = await navigator.mediaDevices.getUserMedia({video: true});
      const stream = await navigator.mediaDevices.getUserMedia({video: { deviceId: { exact: videoDevices[0].deviceId } }});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      let cursorX;

      // Add a click event listener to the video element.
      video.addEventListener('click', (e) => {
        // Capture the x-coordinate of the cursor when clicked.
        cursorX = e.clientX;
      });

      let stop = false;
    """)

    Preview of the web camera will be show by this code.

    In order to capture the current view shown by web camera, click a point on the preview. The point where you click will be checked and the x
    coordinate will be used to save the image.

    Data collection will continue until stop button is clicked.

[ ] try:
    stop = False
    cnt = 1;
    while not stop:
        capture_image, cursorX, stop = take_photo(cnt)
        cnt = cnt + 1
        print('Saved to {}'.format(str(cursorX)))

except Exception as err:
    # Errors will be thrown if the user does not have a webcam or if they do not
    # grant the page permission to access it.
    print(str(err))

```

[ Code ]

#### 4-1-2. Data Collection

AI 모델을 학습시킬 Data Set을 구축하는 단계이다. IRIUN WEBCAM 어플을 사용하였다. 우리 팀은 보행자가 더욱 다양한 길을 거닐 수 있게 하도록 실내와 실외 두 곳에서 Data를 수집하였다. 실내는 교내 기초학문관 1층에서 약 200장의 사진을, 실외는 SAINT 뒤쪽 커브 길에서 약 200장의 사진을 촬영하였다. 카메라가 신발에 부착되면 낮은 시야에서 도로를 촬영할 것이기 때문에 그에 맞추어 촬영하였다. 또한, 우리의 제품이 보행자가 도로의 중앙에 맞추어 걸을 수 있도록 도움을 줄 것이므로 길의 여러 방향에 치우친 곳에서 촬영하여 다양한 Data를 수집하였다.

[실내 Data Set 예시]



[ 실내 좌측 촬영 Data ]



[ 실내 중앙 촬영 Data ]



[ 실내 우측 촬영 Data ]

[실외 Data Set 예시]





[ 실외 좌측 촬영 Data ]



[ 실외 중앙 촬영 Data ]



[ 실외 우측 촬영 Data ]

작성했던 Data Collection Code를 통해 해당 사진을 촬영하며 AI 모델을 학습시킬 사진을 촬영하였다. 커서를 통해 해당 도로의 중앙을 클릭하면 AI 모델은 어떤 기준으로 도로의 중앙을 인식할 것인지 그리고 촬영된 사진이 중앙에서 좌 혹은 우로 얼마만큼 치우쳐졌는지 인식하게 될 것이다.

#### 4-1-3. Model Train

실질적으로 AI 모델을 학습시키는 단계이다. 우선 수집한 약 400장의 Data Set을 불러와 이미지 전처리를 진행한다. 여기서 이미지 전처리는 2D 사진을 1D로 바꾸고 사진의 색을 분리해 학습시키는 과정을 의미한다.

#### [이미지 전처리 과정]

```

Create Dataset Instance

Here we create a custom torch.utils.data.Dataset implementation, which implements the __len__ and __getitem__ methods. This class is responsible for loading images and parsing the x, y values from the image filenames. Because we implement torch.utils.data.Dataset class, we can use all of the torch data utilities

def get_x(file_name):
    """Gets the x value from the image filename"""
    token = file_name.split("-")
    return (float(int(token[1].split(".")[0]) - 300.0) * 5 / 300.0)

class XLabelAugmentImageTrainData(torch.utils.data.Dataset):
    def __init__(self, directory, random_hflips=False):
        self.directory = directory
        self.random_hflips = random_hflips
        self.image_paths = glob.glob(os.path.join(self.directory, '*.jpg'))
        self.augmentation = transforms.Compose([
            transforms.ColorJitter(0.3, 0.3, 0.3, 0.3),
            transforms.RandomRotation(degrees=30),
            transforms.RandomVerticalFlip(p=0.5),
            transforms.RandomApply([transforms.ColorJitter(0.5, 0.5, 0.5, 0.5)], p=0.5),
            transforms.RandomApply([transforms.Grayscale(num_output_channels=3)], p=0.2),
        ])

    def __len__(self):
        return len(self.image_paths)

    def __getitem__(self, idx):
        image_path = self.image_paths[idx]
        image = PIL.Image.open(image_path)
        x = float(get_x(os.path.basename(image_path)))

```

```

image = PIL.Image.open(image_path)
x = float(get_x(os.path.basename(image_path)))

if self.random_hflips and float(np.random.rand(1)) > 0.5:
    image = transforms.functional.hflip(image)
    x = -x

# Apply augmentations
image = self.augmentation(image)

image = transforms.functional.resize(image, (224, 224))
image = transforms.functional.to_tensor(image)
image = image.numpy()[::-1].copy()
image = torch.from_numpy(image)
image = transforms.functional.normalize(image, [0.485, 0.456, 0.406], [0.229, 0.224, 0.229])

return image, torch.tensor([x]).float()

# Example usage
dataset = XLabelAugmentImageTrainData('training_dataset', random_hflips=True)

for i in range(5):
    image, x = dataset[i]
    print(f"Sample {i+1} - Image Shape: {image.shape}, x: {x.item()}")
total_samples = len(dataset)
print(f"Total number of samples in the dataset: {total_samples}")

Sample 1 - Image Shape: torch.Size([3, 224, 224]), x: -1.649999976158142
Sample 2 - Image Shape: torch.Size([3, 224, 224]), x: -0.23333333432674408
Sample 3 - Image Shape: torch.Size([3, 224, 224]), x: -4.3000000190734863
Sample 4 - Image Shape: torch.Size([3, 224, 224]), x: 1.416666626982368
Sample 5 - Image Shape: torch.Size([3, 224, 224]), x: -2.65000000953674316
Total number of samples in the dataset: 201

```

[ 이미지 전처리 Code ]

다음으로 Data Set을 분할시킨다. Data의 80%는 AI 모델을 학습시키는 데 사용하고 20%는 성능의 과적합을 방지하는 데 사용된다. 즉, Data의 80%를 Train data로, 20%를 Validation data로 사용한다.

#### [Data 분할 과정]

```

Split dataset into train and test sets

Once we read dataset, we will split data set in train and test sets. In this example we split train and test a 80%-20%. The test set will be used to verify the accuracy of the model we train.

[ ] test_percent = 0.2
num_test = int(test_percent * len(dataset))
train_dataset, test_dataset = torch.utils.data.random_split(dataset, [len(dataset) - num_test, num_test])

Create data loaders to load data in batches

```

[ Data 분할 Code ]

분할된 Train Data를 model transfer 방식으로 학습시킨다. Model Transfer 방식으로 학습시킬 수 있는

뛰어난 모델인 resnet18을 사용하였다. Resnet18은 18개의 layer로 이루어진 ResNet을 의미한다. ResNet은 Image Recognition에 뛰어난 CNN Network이다. ResNet의 Layer 수가 많아질수록 더욱 복잡한 패턴을 학습시킬 수 있지만, Residual 학습 특성상 성능이 떨어지기 때문에 18개의 Layer를 가진 Resnet18 모델을 사용하기로 하였다. 기존에 구성된 Resnet18 모델에 마지막 프로젝트 상황에 맞추어 마지막 Layer를 추가하였다.

#### [Model Transfer 과정]

```

v Define Neural Network Model

We use ResNet-18 model available on PyTorch TorchVision.

In a process called transfer learning, we can repurpose a pre-trained model (trained on millions of images) for a new task that has possibly much less data available.

More details on ResNet-18 : https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py

More Details on Transfer Learning: https://www.youtube.com/watch?v=yofjEOddwHE

[ ] model = models.resnet18(pretrained=True)

Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/.cache/torch/hub/checkpoints/resnet18
100%|██████████| 44.7M/44.7M [00:00<00:00, 123MB/s]

ResNet model has fully connect (fc) final layer with 512 as in_features and we will be training for regression thus out_features as 1

Finally, we transfer our model for execution on the GPU

[ ] model.fc = torch.nn.Linear(512, 1)
device = torch.device('cuda')
model = model.to(device)

```

[ ResNet18 + Layer 추가 Code ]

구축된 Train Data Set과 학습 모델을 Optimizer Adam을 사용해 epoch 50번을 진행하며 학습시킨다. Epoch를 1번부터 50번을 반복적으로 진행하면서 성능이 가장 좋은 모델의 Parameter 정보를 저장하였다.

#### [Optimizer Adam을 통한 학습 과정]

```

NUM_EPOCHS = 50
CHECKPOINT_PATH = 'checkpoint.pth'
best_loss = 3e9

optimizer = optim.Adam(model.parameters())

for epoch in range(NUM_EPOCHS):
    model.train()
    train_loss = 0.0

    # Initialize counters for debugging
    total_train_batches = len(train_loader)
    train_batch_count = 0

    for images, labels in iter(train_loader):
        images = images.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = F.mse_loss(outputs, labels)
        train_loss += float(loss)
        loss.backward()
        optimizer.step()

    # Print detailed training information for each batch
    train_batch_count += 1
    print(f'Epoch [{epoch + 1}/{NUM_EPOCHS}] - Batch [{train_batch_count}/{total_train_batches}] - Train Loss: {loss:.4f}')

    train_loss /= len(train_loader)

    model.eval()
    test_loss = 0.0

    train_batch_count = 1
    print(f'Epoch [{epoch + 1}/{NUM_EPOCHS}] - Batch [{train_batch_count}/{total_train_batches}] - Train Loss: {loss:.4f}')

    # Initialize counters for debugging
    total_test_batches = len(test_loader)
    test_batch_count = 0

    for images, labels in iter(test_loader):
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        loss = F.mse_loss(outputs, labels)
        test_loss += float(loss)

    # Print detailed testing information for each batch
    test_batch_count += 1
    print(f'Epoch [{epoch + 1}/{NUM_EPOCHS}] - Batch [{test_batch_count}/{total_test_batches}] - Test Loss: {loss:.4f}')

    test_loss /= len(test_loader)

    print(f'Epoch [{epoch + 1}/{NUM_EPOCHS}] - Train Loss: {train_loss:.4f}, Test Loss: {test_loss:.4f}')

    if test_loss < best_loss:
        torch.save(model.state_dict(), CHECKPOINT_PATH)
        best_loss = test_loss
        print(f'Checkpoint saved: {best_loss}')

Epoch 1/50 - Batch 1/111 - Train Loss: 7.9882

```

[ 학습(Epoch=50) Code ]

#### 4-1-4. Model Evaluation

분할된 Validation data를 사용해서 모델의 결과를 평가하고, 성능 개선 그리고 실시간으로 모델 결과를 평가하는 단계이다.

먼저 Model Train 단계에서 Data Set의 20%로 분할했던 Validation Data Set을 통해 모델 결과를 평가한다. 학습에 사용했던 ResNet18 + Layer를 만든 후 우리가 학습시킨 모델을 업로드한다. 그 후 사용한 Test Data의 정보를 확인한다.

[Validation Data Set을 통한 평가 준비]


```
# Initialize the model architecture
model = models.resnet18(pretrained=False)
model.fc = nn.Linear(512, 1)

# Load the trained model from the checkpoint file and map it to the CPU
checkpoint = torch.load('checkpoint.pth', map_location=torch.device('cpu'))
model.load_state_dict(checkpoint)

# Use GPU if available
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = model.to(device)
```

[ ResNet18 +Layer + 학습모델 Code ]

```
Sample 1 - Image Shape: torch.Size([3, 224, 224]), x1: 0.9833333492279053
Sample 2 - Image Shape: torch.Size([3, 224, 224]), x1: -3.066666603088379
Sample 3 - Image Shape: torch.Size([3, 224, 224]), x1: 1.4833333492279053
Sample 4 - Image Shape: torch.Size([3, 224, 224]), x1: 0.5
Sample 5 - Image Shape: torch.Size([3, 224, 224]), x1: 2.066666603088379
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
x: tensor([0.9833]) x: tensor([-3.0667]) x: tensor([1.4833]) x: tensor([0.5000]) x: tensor([2.0667])
```



[ Test Data 정보 확인 Code ]

그 후 모델을 평가 상태로 바꾸고 모델에 Validation Data를 넣어서 성능을 확인한다.

[모델 성능 확인 과정]

```
import matplotlib.pyplot as plt

# Ensure matplotlib is installed: pip install matplotlib
# Set the model to evaluation mode
model.eval()

# Initialize counters for debugging
total_test_batches = len(test_loader)
test_batch_count = 0
test_loss = 0.0

# Loop to store predictions and ground truth
all_predictions = []
all_labels = []

# Lists to store images and predicted x values for visualization
selected_images = []
predicted_x_values = []

# Evaluate the model on the test data
with torch.no_grad():
    for images, labels in iter(test_loader):
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        loss = F.mse_loss(outputs, labels)
        test_loss += loss

        # Store predictions and ground truth
        all_predictions.extend(outputs.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

# Print detailed testing information for each batch
test_batch_count += 1
print(f"Batch: {test_batch_count}/{total_test_batches} - Test Loss: {loss:.4f}")
print(f"Test Loss: {test_loss:.4f}")

# Visualize predictions vs. ground truth
plt.figure(figsize=(15, 5))
plt.title('Predictions vs. Ground Truth')
plt.xlabel('Predictions')
plt.ylabel('Ground Truth')
plt.imshow(selected_images)
plt.show()

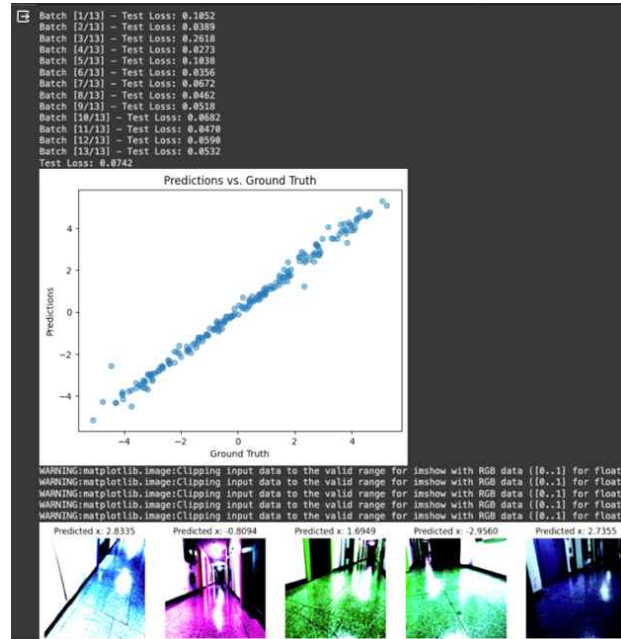
# Visualize selected images with their predicted x values
fig, axes = plt.subplots(1, 5, figsize=(15, 3))
for i in range(5):
    image = selected_images[i].permute(1, 2, 0).numpy()
    predicted_x = predicted_x_values[i]
    axes[i].imshow(image)
    axes[i].set_title(f'Predicted x: {predicted_x:.4f}')
    axes[i].axis('off')
plt.show()
```

[ 모델 성능 확인 Code ]

모델의 성능을 확인하였으니 성능을 개선하기 위해 최적화를 진행하고 성능이 향상하였는지 확인한다.



## [모델 성능 개선 과정]



[ 모델 성능 결과 ]

모델의 성능 그래프를 보면 분산되었던 그래프가 모여며 훨씬 개선된 성능을 보이는 것을 확인할 수 있다.

AI 모델을 구축하였기 때문에 Bluetooth 송수신 환경을 구축한다. 실시간으로 카메라와 연결하고 Arduino board와 Bluetooth serial 연결을 한다. 이를 통해 카메라로 들어오는 Data를 가져와서 학습시킨 AI 모델에 입력시키고 나오는 결과값을 Arduino Board로 보내준다.

## [Data 송수신 과정]

```
import cv2
import os
from PIL import Image

def take_photo(save_path, cnt):
    # Open the webcam
    cap = cv2.VideoCapture(0) # 0 is the default camera

    # Capture a single frame
    ret, frame = cap.read()

    # Save the image
    filename = os.path.join(save_path, f"{cnt}.jpg")
    cv2.imwrite(filename, frame)

    # Release the webcam
    cap.release()

    # Convert the frame to a PIL Image
    pil_image = Image.fromarray(frame)

    return pil_image
```

[ 카메라 연결 Code ]

```

import serial
import time

def send(raw):
    data = None
    if raw <= -5.0 :
        data = "1"
    elif raw <= -2.5:
        data = "2"
    elif raw <= -0.25:
        data = "3"
    elif raw <= 0.25:
        data = "4"
    elif raw <= 5.0:
        data = "5"

    # Windows에서는 'COMx' 형식으로 포트를 지정 (x는 사용 중인 COM 포트 번호)
    ser = serial.Serial('COM8', 9600)

    ser.write(data.encode())
    # 시리얼 포트 닫기
    ser.close()

```

[ Arduino와 Bluetooth 연결 Code ]

```

import time
from PIL import Image

cnt = 1
try:
    while True:
        time.sleep(1) # Sleep for 1 second
        image = take_photo(save_path="live", cnt=cnt)

        # image_path = glob.glob(os.path.join("live", str(cnt)+'.jpg'))
        color_jitter = transforms.ColorJitter(0.3, 0.3, 0.3, 0.3)

        time.sleep(5) # Sleep for 1 second

        # image = PIL.Image.open(image_path)

        image = color_jitter(image)
        image = transforms.functional.resize(image, (224, 224))
        image = transforms.functional.to_tensor(image)
        image = image.numpy()[::-1].copy()
        image = torch.from_numpy(image)
        image = transforms.functional.normalize(image, [0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
        image = image.unsqueeze(0)
        # Set the model to evaluation mode
        model.eval()

        # Evaluate the model on the test data
        with torch.no_grad():
            output = model(image)
            send(output)
            print(output.cpu().numpy())

        # print(capture_image)

        cnt = cnt + 1

except Exception as err:
    # Errors will be thrown if the user does not have a webcam or if they do not
    # grant the page permission to access it.
    print(str(err))

```

[ 카메라 Data 수신, Arduino에 결과값 송신 Code ]

위 과정까지 학습된 모델을 통해 주기적으로 사진을 찍어 사용자가 가야 하는 방향의 좌표를 -5~5 범위의 숫자 데이터로 얻은 뒤 1부터 5까지 정수로 바꾸어 아두이노로 전달하였다.

## 4-2. 카메라 / 초음파 센서

카메라는 신발의 앞부분에 부착되어 도로를 촬영하는 역할을 한다. 카메라가 도로를 실시간으로 촬영하여 Bluetooth를 통해 실시간으로 AI 모델로 전송한다. AI 모델이 실시간으로 사진을 촬영하여 계산하기 때문에 카메라는 전송하는 역할만 하게 된다.

초음파 센서 또한 신발의 앞부분에 부착되어 보행자 앞의 장애물을 감지한다. 50cm 이내에 장애물을 감지하여 진동 모터에 알람을 주는 역할을 한다. 초음파 센서는 Trig 핀에 연결된 압전소자에 전압을 가해서 높은 주파수 펄스를 발사한 뒤 echo pin에서 echo pin으로 신호가 되돌아오기까지의 시간을 계산해서 거리를 측정한다. 50cm 이내 장애물을 감지하면 부착된 4개의 진동 센서가 100ms 동안의 진동을 200ms 간격으로 울리도록 설정하였다.

## 4-3. Arduino

Arduino board가 신발에 부착되어야 하므로 크기가 작아야 한다. 따라서 Arduino nano와 Bluetooth 모듈인 HC-06을 사용하여 PC와 serial 통신할 수 있도록 하였다. HC-06의 Rx, Tx를 아두이노 나노에 각각 Tx(5), Rx(4)로 사용할 핀에 연결하였다. Arduino 코드로 Arduino와 HC-06에 업로드한 뒤 HC-06의 이름과 password, Baud rate (9600)으로 변경하였다. 해당 과정을 통해 카메라로 촬영한 사진을 학습된 AI 모델로 송신하고 AI 모델에서 계산한 값을 수신한다.

Arduino board는 AI Model에서 계산된 경우의 수 그리고 초음파 센서에서 장애물 유무를 확인한 값을 제공받는다. 이 값들을 종합하여 진동 모터를 제어하게 된다. 보행자가 도로의 우측으로 많이 치우쳐 있으면 “강하게 왼쪽으로 움직이기”로 판단되어 좌측 진동 모터 2개가 진동한다. 보행자가 도로의 우측으로 조금 치우쳐 있으면 “약하게 왼쪽으로 움직이기”로 판단되어 좌측 진동 모터 1개가 진동한다. 보행자가 도로의 중앙에 맞추어 걷고 있으면 “직진하기”로 판단되어 보행에 문제가 없으므로 진동 모터들이 진동하지 않는다. 보행자가 도로의 좌측으로 조금 치우쳐 있으면 “약하게 오른쪽으로 움직이기”로 판단되어 우측 진동 모터 1개가 진동한다. 보행자가 도로의 좌측으로 많이 치우쳐 있으면 “강하게 오른쪽으로 움직이기”로 판단되어 우측 진동 모터 2개가 진동한다. 마지막으로 초음파 센서에서 50cm 이내에 장애물이 있다고 판단한 경우인 “멈추기”의 경우 모든 진동 모터가 강하게 진동한다.

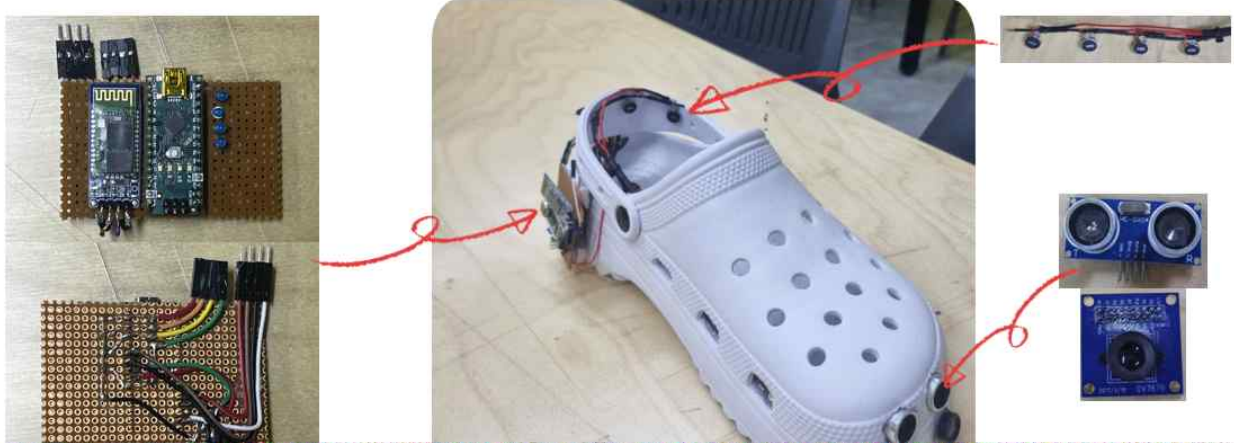
### [Arduino 진동 모터 제어 과정]

```
1 #include <SoftwareSerial.h>
2
3 SoftwareSerial softwareSerial(4, 5);
4
5 int trigPin = 6; // 초음파 거리 센서 trigPin
6 int echoPin = 7; // 초음파 거리 센서 echoPin
7 int VibPin1 = 3; // 진동 모터
8 int VibPin2 = 9;
9 int VibPin3 = 10;
10 int VibPin4 = 11;
11
12 void setup() {
13   Serial.begin(9600); // 시리얼 통신을 위해 시리얼 통신속도를 9600으로 설정
14   softwareSerial.begin(9600);
15
16   pinMode(VibPin1, OUTPUT); // 진동 모터
17   pinMode(VibPin2, OUTPUT); // 진동 모터
18   pinMode(VibPin3, OUTPUT); // 진동 모터
19   pinMode(VibPin4, OUTPUT); // 진동 모터
20   pinMode(echoPin, INPUT); // 초음파 거리 센서 echoPin 입력
21   pinMode(trigPin, OUTPUT); // 초음파 거리 센서 trigPin 출력
22 }
23
24 void loop() {
25   char direction = softwareSerial.read();
26   // 초음파 거리 센서
27   long duration, distance; // duration, distance 변수 설정
28   digitalWrite(trigPin, HIGH); // trigPin에서 초음파 발령(echoPin도 HIGH)
29   delayMicroseconds(10);
30   digitalWrite(trigPin, LOW);
31   duration = pulseIn(echoPin, HIGH); // echoPin 이 HIGH를 유지한 시간을 저장
32   distance = ((float)(340 * duration) / 10000) / 2; // 거리 계산
33
34   // 초음파 거리 센서 시리얼 모니터에 결과 출력
35   Serial.print("Distance:"); // 물체와 초음파 거리 센서간 거리 표시
36   Serial.print(distance);
37   Serial.println("cm");
38   Serial.print("direction:");
39   Serial.println(direction);
40
41   if (distance <= 50) // 물체가 특정 거리 내 진입함
42   {
43     analogWrite(VibPin1, 100);
44     analogWrite(VibPin2, 100);
45     analogWrite(VibPin3, 100);
46     analogWrite(VibPin4, 100);
47     delay(200);
48     analogWrite(VibPin1, 0);
49     analogWrite(VibPin2, 0);
50     analogWrite(VibPin3, 0);
51     analogWrite(VibPin4, 0);
52     delay(100);
53     analogWrite(VibPin1, 100);
54     analogWrite(VibPin2, 100);
55     analogWrite(VibPin3, 100);
56     analogWrite(VibPin4, 100);
57     delay(200);
58     analogWrite(VibPin1, 0);
59     analogWrite(VibPin2, 0);
60     analogWrite(VibPin3, 0);
61     analogWrite(VibPin4, 0);
62
63     switch (direction) {
64       case '1':
65         analogWrite(VibPin1, 100);
66         analogWrite(VibPin2, 100);
67         delay(500);
68         analogWrite(VibPin1, 0);
69         analogWrite(VibPin2, 0);
70         break;
71       case '2':
72         analogWrite(VibPin3, 100);
73         analogWrite(VibPin4, 100);
74         delay(500);
75         analogWrite(VibPin3, 0);
76         analogWrite(VibPin4, 0);
77         break;
78       case '3':
79         analogWrite(VibPin1, 100);
80         analogWrite(VibPin2, 100);
81         delay(500);
82         analogWrite(VibPin1, 0);
83         analogWrite(VibPin2, 0);
84         break;
85       case '4':
86         analogWrite(VibPin3, 100);
87         analogWrite(VibPin4, 100);
88         delay(500);
89         analogWrite(VibPin3, 0);
90         analogWrite(VibPin4, 0);
91         break;
92       default:
93         break;
94     }
95   }
96 }
```



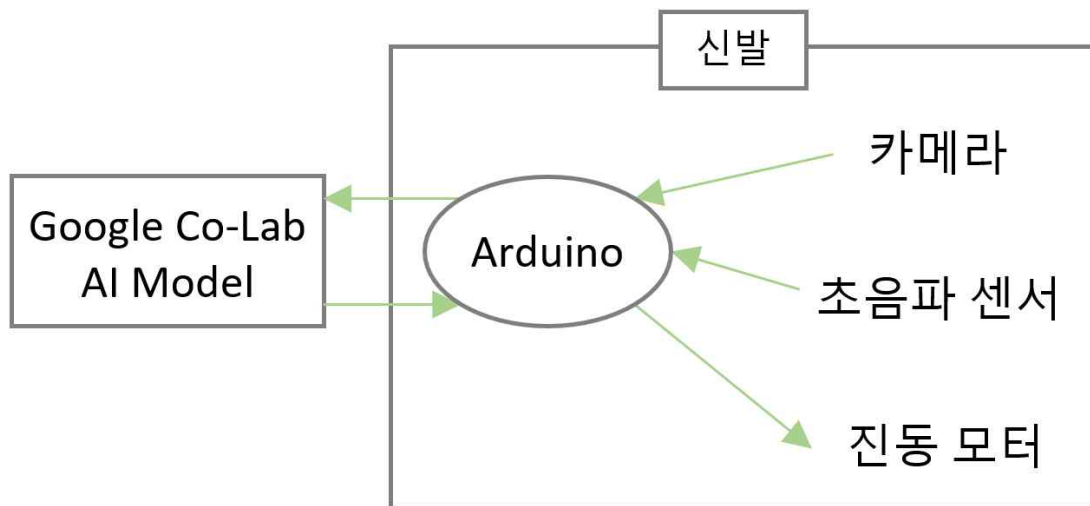
#### 4-4. 신발

최종적으로 Built-in 점자블록 신발을 구현하였다. 초음파 센서와 카메라는 신발의 앞부분, Arduino board, bluetooth 모듈, 9V 배터리는 신발의 외 측면, 진동 모터는 발목의 스트랩에 부착되어 있는 것을 확인할 수 있다.



[ Built-in 점자블록 Prototype 신발 및 HW ]

#### 4-5. 시연



구축된 전체적인 시스템은 위의 사진과 같다.



전방에 놓인 장애물로 인해 진동 센서가 작동하며 보행자에게 이를 알림



보행자가 길의 우측을 향해 보행하자 왼쪽의 진동센서가 작동하고 보행자가 이를 인지하면 다시 보행루트를 중앙으로 수정하여 걸을 수 있음

## 5. 기대효과

“Built-in 점자블록”은 아두이노 보드와 AI 모델 기반의 시각 장애인 보행 보조 장치이다. “Built-in 점자블록” 제품이 상용화된다면 시각 장애인들에게 뿐만 아니라 많은 기대효과를 예상할 수 있다.

### 5-1. 시각 장애인 보행 안전 효과

프로젝트의 목적과 같이 “Built-in 점자블록”은 도로의 중앙과 장애물을 감지하여 보행자에게 알려준다. 현재 시각 장애인들은 지팡이를 통해 길을 찾거나 장애물을 감지한다. 하지만 지팡이를 통해 본인이 길의 중앙에 있는지, 그리고 앞에 있는 모든 장애물을 탐지할 수는 없다. 그러나 센서를 통해 감지한다면 지팡이보다 더욱 잘 감지할 수 있을 것이다. 해당 제품을 이용하여 마음 놓고 어디든 걸어 다닐 수 있을 것이다.

### 5-2. 시각 장애인의 성공적인 길 찾기

Built-in 점자블록 Prototype에는 기능이 들어가지 못했지만, AI 모델과 동시에 Bluetooth로 네비게이션 기능을 제공할 수 있을 것이다. 만약 네비게이션 기능이 들어가서 진동 센서를 통해 시각 장애 보행자에게 목적지까지의 길을 안내해줄 수 있다면 매우 큰 효과를 볼 수 있을 것이다. 시각 장애인은 길 안

내를 음성을 통해 듣는다고 하더라도 본인이 길을 잘 걸어가고 있는지에 대한 의문이 들 것이다. 눈에 보이지 않기 때문이다. 하지만, Built-in 점자블록을 통해 자신이 목적지를 걸어가는 길 중 도로의 중앙에 따라 잘 걸어가고 있는지 혹은 이곳에서 좌·우회전을 해야 하는 상황인지 알 수 있다면 우리의 제품을 신뢰하여 어디든 걸어갈 수 있을 것이다.

### 5-3. 시각적 효과

시각 장애인의 경우 올바른 길의 방향으로 보행하거나 장애물을 감지하기 위해 지팡이 혹은 손을 통해 도로 주변의 지형지물을 건드려보며 걸어야 한다. 하지만 이러한 행위가 시각적으로 좋아 보이지 않고 사람들의 편견을 불러올 수 있다는 두려움 때문에 많은 시각 장애인이 꺼린다. 이는 결국 본인을 위협에 빠트리는 악순환을 이끌 수 있다. 하지만 “Built-in 점자블록” 제품이 길을 안내해주고 장애물을 감지해준다면 지팡이로 짚을 필요도 손을 휘적거릴 필요도 없다. 또한, 이를 통해 타인의 시선을 끌지 않아도 되기 때문에 외부활동에 대한 두려움을 없앨 수 있다. 눈이 보이지 않아도 “Built-in 점자블록”을 믿고 발의 촉각에만 집중하고 걸으면 자신 있게 보행할 수 있다.

### 5-4. 경제적 효과

만약 “Built-in 점자블록” 제품이 상용화된다면 경제적으로도 큰 이득을 볼 수 있다. 모든 시각 장애인들이 해당 제품을 믿고 사용한다면 더 이상 도로에 점자블록을 설치할 필요가 없다. 앞으로 많은 도로가 새로 설계될 것이지만 점자블록에 대한 필요가 없어진다면 그만큼 이득이 생길 것이다. 또한, 지자체에서는 매해 점자블록의 유지보수를 위해 큰 비용을 지불하고 있어서 그만큼의 비용 절감을 이뤄낼 수 있다. 혹은 그만큼의 비용을 시각 장애인들에게 “Built-in 점자블록” 제품을 나눠주는 데에 사용한다면 경제적 이익에 더불어 더욱 부가적인 가치를 창출할 수 있을 것으로 예상된다.

## 6. 기타(한계점 및 개선사항, 업무분장, 소요비용 등)

### 6-1. 한계점

#### 6-1-1. 깔창으로의 발전 한계

초기에 “Built-in 점자블록” 제품을 기획했을 당시, 깔창을 통해 보행자에게 점자블록 정보를 제공하려 했다. 그러나, 크게 두 가지 이유로 한계에 부딪혔다. 첫 번째는 보행자의 체중이다. 우리 조가 설계하려는 보행 보조 장치 특성상, Arduino board, Bluetooth 모듈, 진동 센서 등이 깔창에 들어가야 한다. 크기가 큰 부품은 밖에 배치한다고 하더라도, 정보를 알려주는 센서들은 깔창 내부에 들어가야 하는데 이러한 기기들에 보행자의 체중이 가해진다면 고장 날 가능성이 있다. 두 번째로, 불편한 감각이다. 본래 깔창은 폭신한 쿠션으로 부드럽게 보행자의 발을 감싸주어야 한다. 하지만 진동 센서가 깔창에 들어가고 실시간으로 발에 진동을 주게 된다면 간지럽거나 심하면 불쾌함을 제공할 수 있다고 판단하였다. 아쉬웠지만 발목 스트랩에 진동 센서를 부착하는 방식으로 변경하였다.

#### 6-1-2. Navigation 기능의 한계

기술의 한계로 길 안내 기능을 구현하지 못했지만, 만약 제품에 해당 기능까지 들어간다면 더욱 유용한 제품으로 발전할 수 있을 수 있을 것이다. 제품에는 보행자가 좌측, 우측, 직진, 정지 등의 기능이 구현되어 있기 때문에, GPS를 이용한 실시간 길 안내 기능과 좋은 시너지를 발휘할 수 있을 것이다.

#### 6-1-3. 칩 소형화와 디자인의 한계

Arduino board와 Bluetooth 모듈, 초음파 센서 등 부품들의 크기가 신발에 부착되기에는 매우 크다. Chip과 모듈을 초소형화할 수 있다면 디자인상 예쁘게 만들 수 있을 것이다. 또한, 초음파 센서의 크기가 커 신발의 앞부분에 배치되었을 때 보행자의 발끝에 닿는다. Arduino board의 크기가 크기 때문에 외관상 보이는 신발의 외 측면에 배치되었다. 보행자는 다양한 디자인의 신발을 신기 때문에 다양한 종

류의 신발로 설계해보지 못한 것이 아쉽다.

#### 6-1-4. 보행자 감각의 한계

진동 센서가 발목 스트랩에 부착되었지만, 보행자가 걸을 때 발이 움직일 수 있는 다양한 경우의 수가 존재하기 때문에 진동 센서가 잘 느껴지지 않을 때도 있다. 만약 깔창으로 설계했다면 더욱 발바닥과 맞닿아있어 정보 전달이 쉬웠을 것이다.

### 6-2. 개선사항

#### 6-2-1. 제품의 부재

"Built-in 점자블록"과 같은 제품이 시중에 없었다는 점에서 제품의 기획이 유의미하다고 생각한다. 자료조사 결과 시중의 제품 중 점자블록의 정보 제공에 집중한 것은 없었다. 시각 정보가 제한 되는 시각 장애인의 특성상 최대한 많은 정보가 제공되어야 한다. 많은 정보 중 점자블록의 정보 또한 포함되어야 한다는 사실을 인지한 것이 핵심적인 포인트이다.

#### 6-2-2. 일상에 밀접한 제품

시각 장애인이 일상생활을 하면서 많은 부분에서 타인을 도움을 받을 수 있다. 그러나 보행은 타인을 도움을 받기 쉽지 않다. 그렇기에 현재는 지팡이의 도움을 받아 보행하고 있다. 보행자가 "Built-in 점자블록" 제품을 신뢰한다면 보행의 자유를 느낄 수 있을 것이다.

### 6-3. 업무분장

#### 6-3-1. 공통 (조원 전원 참여)

A급 인재들 조원 모두 회의 과정에 참여하여 Discovery를 통한 문제 자료조사, Empathy map, use case 작성, Define, concept design을 기획하였다.

#### 6-3-2. 디자인 (정재관, 배기문)

A급 인재들의 디자인팀은 제품에 부착될 부품들의 규격을 측정하고 제품 외관 디자인을 진행하였다. 또한, PPT, 포스터, 보고서를 제작하여 제품을 디자인하는 데에 이바지하였다.

#### 6-3-3. 기획 (최명준, 서승현, 금혜란)

A급 인재들의 기획팀은 제품에 필요한 부품을 구매하고 보행자가 잘 인지할 수 있는 정확한 정보를 제공하기 위한 과정을 설계한 뒤 납땜을 통해 부품을 배치하였다. 또한, 도로 Data를 수집하고 UCC를 촬영하는 등 제품 설계를 위한 전반의 과정을 기획하는 데에 이바지하였다.

#### 6-3-4. 개발 (김석진, 김영민, 전우성)

A급 인재들의 개발팀은 제품을 동작시키기 위한 Code를 작성하였다. AI 모델학습에 필요한 Data Collection code, Bluetooth module code, 진동 센서 제어 Arduino board code 등을 작성하여 제품을 성공적으로 설계하는 데에 이바지하였다.

## 7. 참고문헌

[1] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

[2] Jo, S. (2022, October 24). "Seoul Han River North Area, Danger in the Pedestrian Environment for Visually Impaired People, Including Crosswalks!" Indigo. <https://theindigo.co.kr/archives/41027>