

AI분리수거 프로젝트

최종보고서

2021. 10. 22.

팀	명	Fossil
팀	장	20140938 이원희
팀	원	20140937 이승준 20140904 김영후
지도교수		김상근 교수님

목차

I. 종합설계 제안서

1. 종합설계 개요

1.1. 주제명	1
1.2. 목적	1
1.3. 필요성/배경	1

2. 개발범위

2.1. 구성도	2
2.1.1 H/W 구성도	2
2.1.2 S/W 구성도	2
2.1.3 업무 구성도	
2.1.3.1. 현재 업무 구성도	3
2.1.3.2. 개선된 업무 구성도	4
2.2. 대상 업무	
2.2.1 H/W 구축범위	5
2.2.2 S/W 개발범위	5
2.2.3 업무 개선범위	5
2.3. 개발환경	
2.4. H/W환경	5
2.5. S/W 환경	5

3. 개발 추진체계

3.1. 팀 소개	6
3.2. 총괄 추진체계	6
3.3. 개발 수행 추진체계	7
3.4. 추진 방법	7

4. 개발 추진 절차

5. 산출물 계획

6. 개발 추진 일정

7. 보고계획

8. 설계 제한 요소 반영

8.1. 경제성	9
8.2. 편리성	10
8.3. 윤리성	10
8.4. 안정성	10
8.5. 유지관리 용이성	10

9. 기대효과

9.1. 결과물의 기대효과 또는 혜택	10
9.2. 참고고서 및 모델, 유사 S/W 비교	11

10. 요구사항

II. 요구사항 분석서

1. 개요

1.1. 시스템 개요	13
1.2. 목표	13
1.3. 시스템 제공 기능	13

2. 요구사항

2.1. 요구사항 목록	14
2.2. 장비 요구사항	15
2.3. 기능 요구사항	18
2.4. 보안사항	21
2.5. 제약사항	22

III. 시스템 설계서

1. 개요

1.1. 시스템 개요	23
1.2. S/W의 주요기능	23

2. 시스템 구조

2.1. 시스템 구조개요	24
2.2. 시스템 구조도	24

3. 소프트웨어 설계

3.1. 유스케이스 다이어그램	25
3.2. 유스케이스 명세서	25
3.3. 클래스 다이어그램	28
3.4. 시퀀스 다이어그램	28
3.5. 활동 다이어그램	30

4. 사용자 인터페이스

4.1. UI 설계서	31
-------------------	----

IV. 소스코드

1. 하드웨어

1.1. 모듈제어	38
-----------------	----

2. 이미지 인식

2.1. Object Detection	42
-----------------------------	----

3. 통신

3.1. 소켓 통신	75
3.2. 시리얼 통신	86

I. 종합설계 제안서

1. 종합설계 개요

1.1. 주제명

- AI 분리수거 쓰레기통

1.2. 목적

- 사용자가 분리수거를 할 때 쓰레기통의 AI와 센서가 재활용 쓰레기 종류를 인식하여 사용자에게 분리방법 안내, 쓰레기 수거, 쓰레기 용량확인을 할 수 있다.

1.3. 필요성/배경

- 최근 일회용품 사용이 급증하면서 쓰레기 처리 및 분류비용이 증가하였고 매립 쓰레기장이 수용한계에 도달하면서 쓰레기를 줄이기 위한 재활용이 중요한 이슈가 되고 있다. 반면 재활용 쓰레기 분리를 하고 있는 가정 86%중 대다수가 올바른 분리배출 방법을 모르는 경우가 많아 실제 재활용률은 40%밖에 되지 않고 있다. 따라서 이 제품을 통해 정확한 분리수거를 하고 2차 분리 비용을 절감하여 결과적으로 재활용률을 올리는 것이 목표이다.

플라스틱 분리수거 한다고 전부 다시 쓰는거 아니다...재활용률 고작 30%

머니투데이 | 정한결 기자

VIEW 10,057 | 2020.09.14 16:00

인쇄 | 저장 | 가 | 의견 남기기

I [MT리포트]배달의 시대, 쓰레기의 습격③

[편집자주] '쓰레기 대란'이 일박했다. 코로나19 확산, 언택트 소비 확대 등으로 폐기물이 쏟아지면서다. 유가 하락에 따른 폐기물 재활용 수요 감소까지 맞물리면서 불에 기름을 부었다. 폐기물 재활용 업체들은 "더이상 버티기" 어렵다고 하소연한다. 이들이 손을 놓으면 동네엔 쓰레기가 쌓일 수 밖에 없다. '발등의 불'이 된 '쓰레기' 문제를 긴급 점검했다.



코로나19 여파로 배달업과 비대면 산업이 성장하면서 플라스틱 배출량이 끝없이 치솟고 있다. 문제는 해마다 1000만 톤 가량 쏟아지는 플라스틱 폐기물 중 실제 재활용되는 것은 3분의 1뿐이라는 점이다. 전문가들은 절대 소비량을 줄이지 않으면 '플라스틱 대란'을 피할 수 없다고 경고한다.



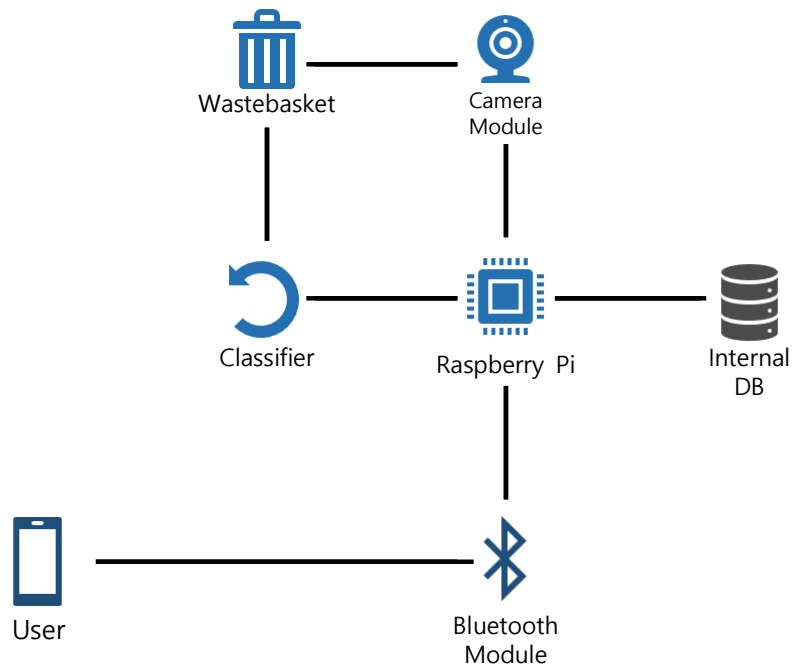
[그림 1] 분리수거에 대한 문제점 기사 일부

출처 : <https://www.nocutnews.co.kr/news/5159798>

2. 개발범위

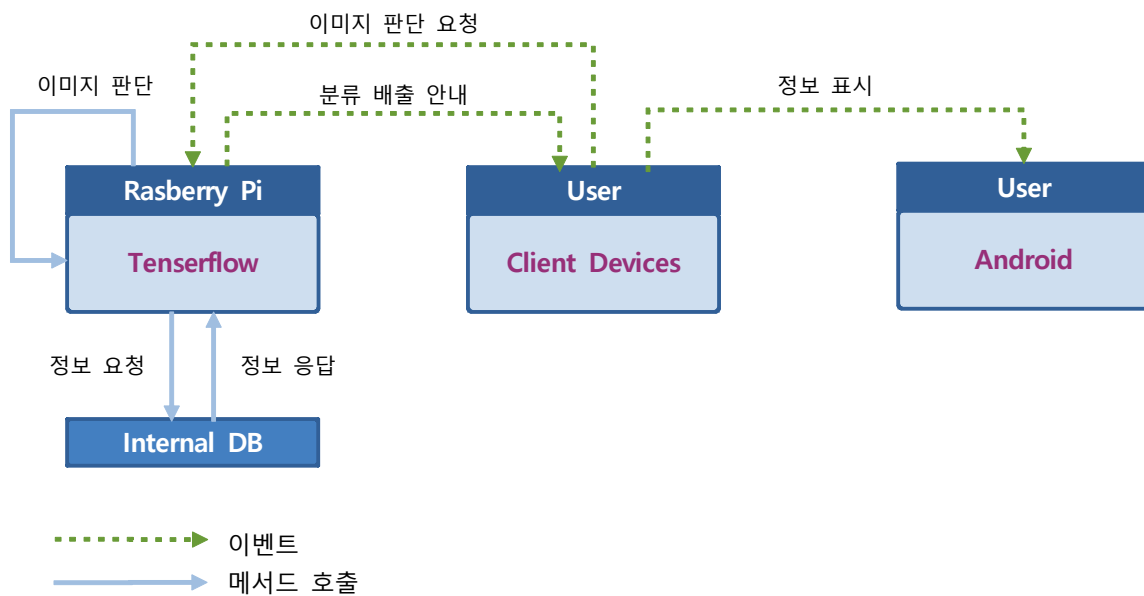
2.1. 구성도

2.1.1. H/W 구성도



[그림 2] H/W 구성도

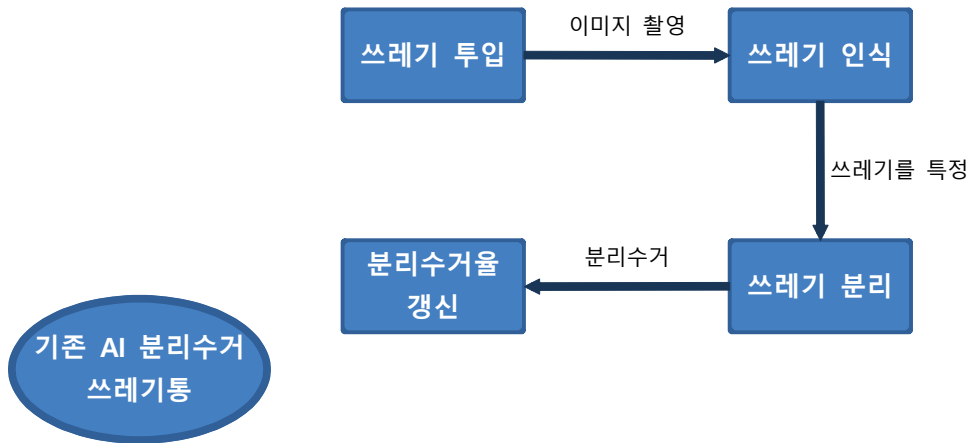
2.1.2. S/W 구성도



[그림 3] S/W 구성도

2.1.3. 업무 구성도

2.1.3.1. 현재 업무 구성도



[그림 3] 현 업무구성도

기존 제품



[그림 4] 슈퍼빈의 네프론 제품
슈퍼빈의 네프론

출처 : <https://www.startuptoday.kr/news/articleView.html?idxno=30167>

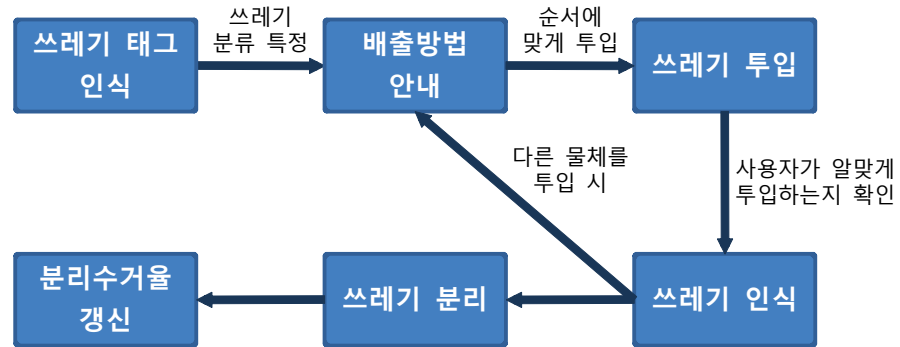


[그림 5] 파라마운트의 RoboBin 제품

파라마운트 AI의 RoboBin

출처 : <http://www.aitimes.com/news/articleView.html?idxno=120863>

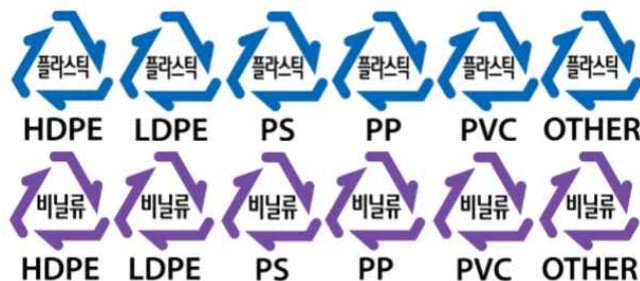
2.1.3.2. 개선된 업무 구성도



개선된 AI 분리수거
쓰레기통

[그림 6] 개선된 업무구성도

- 기존 AI 분리수거는 방식은 학습된 AI가 쓰레기를 인식하여 분류하는 방식이다. 학습 데이터에 없거나 인식률이 낮은 경우 정확하게 분류되기 힘들다.
- 또한 인식률을 높이기 위해 데이터가 너무 커지면 처리 속도가 느려지고 고성능 프로세서를 필요로 한다.
- 속도와 성능문제로 기존 제품은 캔 또는 페트병만 인식하거나 특정 조건(색깔, 형태)을 만들어 시연하는 정도이다. 투명한 플라스틱류와 유리병은 구분하기 어려워 분류를 못하고 있다.
- 개선된 제품은 분리수거마크를 인식하고 분리방법을 안내 후 쓰레기가 오투입 되는 경우를 다시 인식하기 때문에 정확한 분리수거를 가능하게 한다.



[그림 7] 각종 분리배출 마크들

현재 국내에서 사용 중인 재활용가능자원분리배출표시제도

2.2. 대상 업무

2.2.1. H/W 구축범위

<개발용>

- PC 또는 노트북 1대
- 라즈베리 파이 1대
- 이미지 촬영용 카메라 모듈 2대
- 음성 안내용 스피커 모듈 1대
- 화면표시용 LCD 모듈 1대
- 블루투스 통신 모듈 1대
- 쓰레기 분리 시 사용할 모터 1대

<운영용>

- 블루투스 통신이 가능한 안드로이드 스마트폰 1대

2.2.2. S/W 개발범위

- MySQL, Android Studio, Python, Tensorflow CNN, Android

2.2.3. 업무 개선범위

- Tensorflow 이미지 인식 기반으로 한다.

2.3. 개발환경

2.3.1. H/W 환경

- DB 및 제어용 라즈베리파이 1대
- UML 설계 용 및 코딩용 PC 1대
- 유저 인터페이스 디자인용 PC 1대
- 테스트용 스마트폰 1대 이상

2.3.2. S/W 환경

- Visual Studio 2017
- Android Studio 3.0
- Android JDK
- Java SDK
- Python 3.9.3
- Pycharm Community 2020.3.5
- MySQL

3. 개발 추진 체계

3.1. 팀 소개

- 팀 형성배경

이전 군복무를 마친 후, 현재 다시 복학한 후 이전부터 함께했었던 현재 팀원들과 같이 합을 마쳐 부족할 수 있지만 서로의 단점을 보완할 수 있는 좋은 팀이 될 수 있을 것으로 예상되어, 현재의 팀을 구성하게 됐다.

- 팀 기술력

이수한 전공 커리큘럼 : Java 프로그래밍(1) -> 모바일프로그래밍

설계프로젝트 경험 : 간단한 지하철 앱, 간단한 차량대여 앱, 간단한 웹 게시판, 데이터베이스 구축 등

- 팀 성공요소

각자의 취약점이나 강점을 알고 있기 때문에 서로서로 보완해주기 쉽다. 팀 구조가 수평적이기 때문에 의견전달이 자유롭다.

팀원 모두가 안드로이드 앱 개발을 경험한 적이 있다.

팀원 모두가 DB 시스템을 설계 및 구축해본 경험이 있다.

- 취약점 분석

파이썬의 머신러닝 대한 지식이 부족하다.

팀원 모두가 친한 관계라 회의 중 주제에서 벗어난 대화를 할 수 있다.

- 극복방안

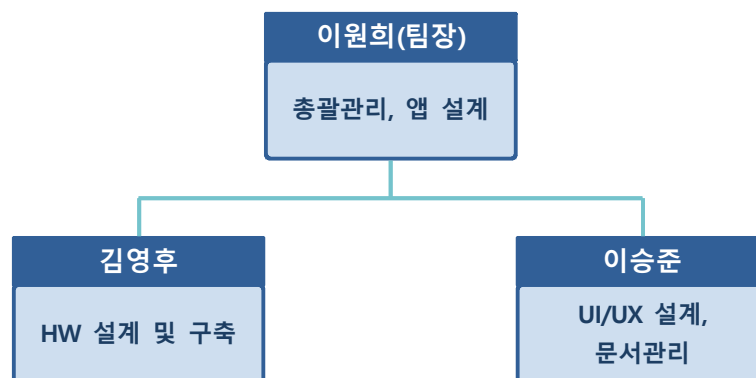
팀원 모두가 시간을 투자하여 머신러닝에 대해 학습한다.

회의 도중 주제를 과도하게 벗어난 얘기가 계속 진행되면 팀장이 중재하여 회의를 속행한다.

3.2. 총괄 추진 체계

- 조직도

[표 1] 업무 조직도



- 정기 모임

시간 : 월, 금요일 19:00~20:00

장소 : Zoom, Discord 또는 오프라인

3.3. 개발수행 추진체계

- 요구분석 : 이원희, 이승준
- DB모델링 : 이원희
- UML : 이승준, 김영후
- UI 디자인 : 이승준
- 문서작업 및 관리 : 이승준, 이원희
- 총괄관리 : 이원희
- 구현관리 : 김영후
- 일정관리 : 이승준

본 프로그램 설계 시 위 체계로 담당자를 정한 것이며 기본적으로 모든 단계 시 필요한 업무는 공동으로 한다.

3.4. 추진 방법

- 현재 시장에 출시되어 있는 제품들을 조사해 기존의 문제점을 발견하고 문제점을 찾고 보완한다.
- 사용자 관점에서 편의적인 기능인지, 필요한 기능인지 검토한다.
- 제대로 된 분리수거를 위해 분리수거가 가능한 분류 규정에 준수한다.

4. 개발 추진절차

- 애자일 방법론에 따른 개발주기

[표 2] 애자일 방법론의 개발주기

요구사항 관리	지속적인 요구사항 개발 및 변경 수용
계획 수립	두 단계 계획 (찾은 계획수립&갱신), 경험 기반 프로세스
설계	적시(Just-in-time) 설계
문서화	경량(Lightweight) 프로세스 및 문서화보다 코드를 강조
역할	전체 팀(Whole team) 워크를 중요시



[그림 8] 애자일 개발방법론 모형

5. 산출물 계획

[표 3] 산출물 계획

요구사항	계획서
분석	요구사항 정의서, 업무 흐름도 UI 정의서
설계	프로그램 설계서, 테이블 설계서, 프로그램 리스트, 시스템구조 설계서
구현	프로그램 소스
테스트	프로그램 테스트 계획서, 배치 계획서

6. 개발 추진 일정

- 목표와 기준의 설정, 요구분석, 설계 구현, 시험, 평가

[표 4] 개발 추진일정

제안서 제출	03/15
요구사항 수집 및 분석	03/15 ~ 04/11
설계(기본설계/상세설계)	04/12 ~ 05/02
구현(코딩)	05/03 ~ 07/04
테스트	07/05 ~ 07/25
평가(개선 및 완성)	07/26 ~ 10/18
최종보고서 제출	10/22

[표 5] 추진일정표

구성 \ 일정	3월	4월	5월	6월	7월	8월	9월	10월
요구사항 수집 및 분석	■							
기본 설계		■						
상세설계		■						
구현			■	■	■			
테스트					■			
개선 및 완성						■	■	■
최종보고서								■

7. 보고 계획

- 팀원 간 보고는 1주일에 2번(월, 금) 정기회의를 진행한다.
- 한 달에 한 번 담당지도 교수와의 면담을 통해 프로젝트의 전반적인 상황을 보고한다.
- 지도 교수 면담 보고는 팀 회의를 하며 나오는 결과물(회의록, 참고자료 등)로 보고한다.
- 모든 보고서는 한글 파일로 팀원 모두가 보관하며 최종보고서 이전의 보고서를 모두 종합하여 보고한다.

8. 현실적 제한조건 반영

8.1. 경제성

- 제품 제작에 필요한 부품은 종류에 따라 가격이 높기 때문에 설계 요구조건에 맞는 반드시 필요한 것만 구매하도록 한다. 부품 비용은 최대 10만원 이내에서 해결 할 수 있도록 제한하여 소모비용을 최소화 할 것이다.

8.2. 편리성

- 사용자가 분리수거 중 실수하는 문제를 최소화하기 위해 재활용 마크와 쓰레기를 인식하여 오투입이 없도록 하고 자동으로 쓰레기를 분리하도록 한다.

8.3. 윤리성

- 코딩 및 구현은 각 팀원들이 직접 하지만 책이나 인터넷 또는 다른 개발자의 오픈소스를 사용할 경우에는 라이선스 및 출처를 반드시 명시 할 것이다.

8.4. 안정성

- 센서 모듈을 마더보드와 결합 할 때 쇼트나 단선이 되지 않도록 배선에 신경 쓰고 프로젝트가 유실되는 것을 막기 위해 정기적으로 클라우드에 업로드 하여 보관할 것이다.

8.5. 유지관리 용이성

- 비슷한 기능은 최대한 기존 코드를 재활용하여 코드의 거대화를 줄이고 메소드와 변수명은 그 기능에 알맞은 이름을 사용하여 가독성을 높임과 동시에 코멘트 자세하게 적어서 다른 인원이 작업할 때 유지관리가 용이하도록 개선한다.
- 테스트를 충분히 수행하여 버그를 최소화 하고 문제가 발생했던 부분에 수정 코멘트를 작성한다.
- 상세설계 요구사항을 모두 만족하고 충분한 테스트를 거친 후에 업데이트를 진행하여 유지보수를 용이하게 한다.

9. 기대효과

9.1. 결과물의 기대효과 또는 혜택

- 결과물은 인공지능의 이미지 인식을 통해 사용자에게 분리배출 방법을 안내하고 쓰레기를 자동으로 수거하여 재활용 및 사용자 편의성 증진과 환경오염 문제 개선에 도움이 될 것이라 예상된다.

9.2. 참고 도서 및 모델, 유사 S/W 비교

- 참고자료

① 분리수거 분류 기준

별표 1 분리수거 대상 재활용가능자원의 종류 및 분리배출요령

1. 재활용가능자원 종류별배출 시 분리수거 종류 및 배출요령

종류	세부종류	배출요령
가. 플라스틱류	플라스틱상자 등	<ul style="list-style-type: none"> - 지닐코팅, 파이프, 칼자루 등이 들어있는 트레이, 칼판, 알루미늄막 등을 제거하고 잘라서 배출 - 안의 별도 보관 장소이런 등 다른 종이류와 섞이지 않게 배출
	종이팩 (살균팩, 멸균팩)	<ul style="list-style-type: none"> - 내용물을 비우고, 물과 헹구는 등 이물질물 제거하고 말린 후 배출 - 빨대, 비닐 등 종이팩과 다른 재질은 제거한 후 배출 - 다른 종이류와 혼합되지 않게 종이팩 전 용수거함에 배출 - 종이팩 전용수거함이 없는 경우에는 종이팩과 구분할 수 있도록 가림막 칸 등으로 묶어 종이류 수거함으로 배출
나. 플라스틱 외	신문지	<ul style="list-style-type: none"> - 색지(광고 등) : 우유팩, 우유캔, 소스캔, 콘스캔 등 - 불기에 못지 않도록 하고 반듯하게 펴서 - 차곡차곡 쌓은 후 묶어서 배출 - 비닐로 된 광고지, 비닐류, 기타 요물이

[그림 9] 분리수거 분류 기준

② 재활용가능자원의 분리수거에 관한 지침

[별표 1]

분리수거대상 재활용가능자원의 종류 및 분리배출요령

1. 재활용가능자원 종류별배출 시 분리수거 종류 및 배출요령

종류	세부종류	배출요령
가. 플라스틱류	플라스틱상자 등	<ul style="list-style-type: none"> - 지닐코팅, 파이프, 칼자루 등이 들어있는 트레이, 칼판, 알루미늄막 등을 제거하고 잘라서 배출 - 안의 별도 보관 장소이런 등 다른 종이류와 섞이지 않게 배출
	종이팩 (살균팩, 멸균팩)	<ul style="list-style-type: none"> - 내용물을 비우고, 물과 헹구는 등 이물질물 제거하고 말린 후 배출 - 빨대, 비닐 등 종이팩과 다른 재질은 제거한 후 배출 - 다른 종이류와 혼합되지 않게 종이팩 전 용수거함에 배출 - 종이팩 전용수거함이 없는 경우에는 종이팩과 구분할 수 있도록 가림막 칸 등으로 묶어 종이류 수거함으로 배출
나. 플라스틱 외	신문지	<ul style="list-style-type: none"> - 색지(광고 등) : 우유팩, 우유캔, 소스캔, 콘스캔 등 - 불기에 못지 않도록 하고 반듯하게 펴서 - 차곡차곡 쌓은 후 묶어서 배출 - 비닐로 된 광고지, 비닐류, 기타 요물이

[그림 10] 재활용가능 자원



출처

<https://namu.wiki/w/%EB%B6%84%EB%A6%AC%EC%88%98%E%A%B1%B0#s-6.2.1>

출처 :

<https://www.law.go.kr/%ED%96%89%EC%A0%95%EA%B7%9C%EC%B9%99/%EC%9E%AC%ED%99%9C%EC%9A%A9%EA%B0%80%EB%8A%A5%EC%9E%90%EC%9B%90%EC%9D%98%20%EB%B6%84%EB%A6%AC%EC%88%98%EA%B1%B0%20%EB%B9%93%B1%EC%97%90%20%EA%B4%80%ED%95%9C%20%EC%A7%80%EC%B9%A8>

■ 유사한 제품

① 수퍼빈의 네프론	② 파라마운트 AI의 RoboBin
 <p data-bbox="403 622 627 689">[그림 11] 수퍼빈의 네프론</p>	 <p data-bbox="866 633 1233 667">[그림 12] 파라마운트의 RoboBin</p>
<p>현재 국내 각지에 약 90대 가량 설치되어 있으며 투입할 수 있는 종류는 패트와 캔 두 종류로 쓰레기를 투입된 쓰레기를 인식해서 알맞게 분류하고 그에 맞는 포인트를 적립시킬 수 있다.</p> <p>적립방식은 투입을 완료하고 전화번호를 입력해 홈페이지에서 회원가입을 진행해 해당 회원의 포인트를 적립한다.</p> <p>하지만 1인당 최대 30개의 쓰레기를 투입할 수 있는 제한이 있으며, 투입이 가능한 종류의 개수가 2 종류이기 때문에 새새한 분류가 불가능하다는 점이 단점으로 꼽을 수 있다.</p> <p>또한 사용자가 의도하지 않은 방식 즉, 일반 쓰레기나 항목에 없는 쓰레기를 투입 시 분류 불가, 입구 막힘 등의 문제도 발생한 것으로 보인다.</p>	<p>캐나다에 있는 토론토 대학교 학생들이 개발한 제품으로 재활용품, 폐기물, 음식물 쓰레기를 구분해 배출할 수 있게 한 제품이다.</p> <p>작동방식은 폐기물을 넣고 버튼을 누르면 이미지를 이용해 쓰레기를 구분한 다음에 자동으로 배출해준다.</p> <p>앞서 제시한 네프론과 기능적인 부분도 비슷하고 문제점 또한 비슷하다.</p> <p>사용자가 의도에 맞지 않게 작동시킬 경우 비슷한 문제가 발생한다.</p>

출처 : <https://www.startuptoday.kr/news/articleView.html?idxno=30167>
<http://superbin.co.kr/new/contents/product.php>

출처 : <http://www.aitimes.com/news/articleView.html?idxno=120863>

10. 요구사항

- IoT제품 또는 인공지능 개발 경력 있는 멘토
- 제작비

II. 요구사항 분석서

1. 개요

1.1. 시스템 개요

- 실내에 있는 공기 측정기를 통해 실내의 초미세먼지, 미세먼지, 유해가스, 온도, 습도, 일산화탄소 등의 실내 공기질 정보를 측정하고 이 정보를 서버에 보내고 서버는 실내 공기질에 대한 정보를 수집한다.
- 기상청 또는 WHO에서 제공하는 API를 통해서 실외의 초미세먼지, 미세먼지, 온도, 습도 등의 실외 공기 정보를 서버에서 수집한다.
- 이렇게 수집한 정보들을 토대로 실내 환경 유지 및 관리 솔루션을 제공 또는 실내 공기질을 개선하기 위해 사용자의 조작 없이 개선하기 위한 장치가 작동한다.

1.2. 목표

- 사용자가 이용할 모바일 앱 개발
- 사용자 및 창문의 정보를 관리하기 위한 DB 구축
- 실내 공기정보와 실외 공기정보를 바탕으로 어떻게 동작할 것에 대한 대책
- 모든 팀원이 이해할 수 있도록 가독성 높게 설계

1.3. 시스템 제공기능

- 개선 장치 조회/추가/제거/정보변경
- 실외 환경정보 수집/조회
- 실내 환경정보 수집/조회
- 자동제어를 위한 설정
- 스마트 필름을 통한 블라인드
- 창문 자동 개폐
- 창문 원격 개폐
- 침입 경고 알림
- 창문 원격 잠금
- 기기 정보 확인

2. 요구사항

2.1. 요구사항 목록

[표 6] 요구사항 목록

요구사항 분류	번호	요구사항
장비 구성 요구사항 (Equipment Composition Requirement)	ECR-001	DBMS 구축
	ECR-002	센서 제어 보드 구축
	ECR-003	Android 단말기
	ECR-004	쓰레기 인식 단말기 구축
	ECR-005	분리수거 처리 단말기 구축
기능 요구사항 (System Function Requirement)	SFR-001	사용자 이용 식별
	SFR-002	쓰레기 투입구 제어
	SFR-003	투입된 쓰레기 인식
	SFR-004	투입된 쓰레기 수거
	SFR-005	쓰레기 배출 음성 안내
	SFR-006	쓰레기 용량 갱신
	SFR-007	쓰레기 배출량 표시
	SFR-008	투입된 쓰레기 반출
	SFR-009	배출량 DB 구축
	SFR-010	AI 구축
보안 요구사항 (Security Requirement)	SCR-001	시스템 정보 보안 관리 요구사항
제약사항 (Constraint Requirement)	CNR-001	시스템 구축 시 준수사항
	CNR-002	H/W 구현 시 제약사항

2.2. 장비요구사항

[표 7] 시스템 장비 구성 요구사항 - DBMS 구축

요구사항	분류	시스템 장비 구성 요구사항		
	번호	ECR-001		
	명칭	DBMS 구축		
상세설명	정의	각 제품 및 앱을 위한 DBMS 구축		
	세부내용	■ DBMS 구축		
		장치	세부규격	수량
		CPU	- Broadcom BCM2711 SoC 1.5GHz	1개
		Main Memory	- 4GB 이상	
Disk	- 16GB 이상			
산출정보				

[표 8] 시스템 장비 구성 요구사항 - 센서 제어보드 구축

요구사항	분류	시스템 장비 구성 요구사항		
	번호	ECR-002		
	명칭	센서 제어 보드 구축		
상세설명	정의	각 센서를 위한 제어 보드 구축		
	세부내용	■ 센서 제어 보드 구축		
		장치	세부규격	수량
		컨트롤러	- ATmega328P	1개
		작동 전압	- 5V	
		플래시 메모리	- 32KB	
클럭 속도	- 16MHz			
산출정보				

[표 9] 시스템 장비 구성 요구사항 - 안드로이드 단말

요구사항	분류	시스템 장비 구성 요구사항		
	번호	ECR-003		
	명칭	안드로이드 단말기 도입		
상세설명	정의	안드로이드 단말기		
	세부내용	■ 안드로이드 단말기		
		장치	세부규격	수량
		OS	- Android 5.0 이상	1개
		Communication Module	- WiFi or Bluetooth	
Storage	- 100MB 이상			
산출정보				

[표 10] 시스템 장비 구성 요구사항 - 쓰레기 인식 단말기

요구사항	분류	시스템 장비 구성 요구사항		
	번호	ECR-004		
	명칭	쓰레기 인식 단말기 구축		
상세설명	정의	분리수거 실시간 투입 쓰레기 인식을 위한 단말기 구축		
	세부내용	■ 검출기 단말기		
		장치	세부규격	수량
		Camera	- V2, 8MP	1개
		AI	- Tensorflow	
Sensor Module	- LED			
산출정보				

[표 11] 시스템 장비 구성 요구사항 - 분리수거 처리 단말

요 구 사 항	분류	시스템 장비 구성 요구사항		
	번호	ECR-005		
	명칭	분리수거 처리 단말기 구축		
상 세 설 명	정의	분리수거 투입 및 용량측정을 위한 단말기 구축		
	세부 내용	■ 제품 단말기		
		장치	세부규격	수량
		Board	- Uno R3 보드	1개
		Communication Module	- Bluetooth or WIFI	
Module	- 모터 - 서보모터 - 컨베이어 벨트 - 적외선 측정센서(QB1105)			
산출정보				

2.3. 기능 요구사항

[표 12] 기능 요구사항 - 사용자 이용 식별

요 구 사 항	분류	기능 요구사항
	번호	SFR-001
	명칭	사용자 이용 식별
상 세 설 명	정의	사용자가 분리수거기 이용하는 것을 식별
	세부 내용	<ul style="list-style-type: none"> ■ 사용자 이용 식별 - 사용자는 터치스크린에 시작 버튼을 눌러 제품 이용의 시작을 알림
산출정보		

[표 13] 기능 요구사항 - 쓰레기 투입구 제어

요 구 사 항	분류	기능 요구사항
	번호	SFR-002
	명칭	쓰레기 투입구 제어
상 세 설 명	정의	쓰레기가 투입될 투입구를 개폐 또는 폐쇄
	세부 내용	<ul style="list-style-type: none"> ■ 투입구 개폐 - 사용자의 시작을 알림 받고, 투입구를 개폐한다. ■ 투입구 폐쇄 - 사용자가 사용을 마친 것을 인식 후, 투입구를 폐쇄한다.
산출정보		

[표 14] 기능 요구사항 - 투입된 쓰레기 인식

요 구 사 항	분류	기능 요구사항
	번호	SFR-003
	명칭	투입된 쓰레기 인식
상 세 설 명	정의	투입된 쓰레기의 유형을 분석 후 특정 분류군으로 구분
	세부 내용	<ul style="list-style-type: none"> ■ 쓰레기 이미지 촬영 - 투입된 쓰레기를 내부 카메라로 촬영 후, 해당 이미지를 이용해 분석 시작 ■ 특정 유형 구분 - 촬영된 이미지를 이용해 특정 분류군으로 구분
산출정보		

[표 15] 기능 요구사항 - 투입된 쓰레기 수거

요구사항	분류	기능 요구사항
	번호	SFR-004
	명칭	투입된 쓰레기 수거
상세설명	정의	투입구에서 쓰레기의 인식이 완료되면 분류 통에 적재한다.
	세부내용	<ul style="list-style-type: none"> ■ 분류 통 이동 <ul style="list-style-type: none"> - 투입된 쓰레기의 인식이 완료되면 컨베이어 벨트가 동작하여 투입구에서 분류 통 입구로 쓰레기를 이동 ■ 분류기 작동 <ul style="list-style-type: none"> - 이동된 쓰레기가 알맞은 분류 통에 적재 될 수 있도록 이동 경로를 지정해준다.
산출정보		

[표 16] 기능 요구사항 - 쓰레기 배출 음성안내

요구사항	분류	기능 요구사항
	번호	SFR-005
	명칭	쓰레기 배출 음성안내
상세설명	정의	해당하는 유형의 쓰레기 배출 방법을 음성으로 안내
	세부내용	<ul style="list-style-type: none"> ■ 음성 안내 <ul style="list-style-type: none"> - 사용자가 시작을 요구한 경우, "쓰레기를 하나씩 투입해주세요." 라는 음성 메시지를 출력한다. - 사용자가 이용을 마쳤다고 알린 경우, "이용해 주셔서 감사합니다." 라는 음성 메시지를 출력한다. - 분류에 맞지 않는 쓰레기를 투입한 경우, "잘못된 쓰레기를 투입 하셨습니다. 수거해 주시기 바랍니다." 라는 음성 메시지를 출력한다.
산출정보		

[표 17] 기능 요구사항 - 쓰레기 용량 갱신

요구사항	분류	기능 요구사항
	번호	SFR-006
	명칭	쓰레기 용량 갱신
상세설명	정의	쓰레기 적재가 완료되면 분류 통 용량을 측정하여 갱신한다.
	세부내용	<ul style="list-style-type: none"> ■ 용량 측정 <ul style="list-style-type: none"> - 쓰레기 적재가 완료되면 적외선 센서가 분류 통 용량을 측정한다. ■ DB 갱신 <ul style="list-style-type: none"> - 측정한 데이터를 DB에 저장한다.
산출정보		

[표 18] 기능 요구사항 - 쓰레기 배출량 표시

요구사항	분류	기능 요구사항
	번호	SFR-007
	명칭	쓰레기 배출량 표시
상세설명	정의	제품 패널에 계산된 쓰레기 배출량을 표시한다.
	세부내용	<ul style="list-style-type: none"> ■ 쓰레기 잔량 표시 <ul style="list-style-type: none"> - DB에 저장된 쓰레기 배출량을 제품 패널에 표시한다. - 쓰레기통의 잔량이 모두 찬 경우, "현재 쓰레기통이 모두 찼습니다." 라는 메시지를 표시 후 투입구를 폐쇄한다.
산출정보		

[표 19] 기능 요구사항 - 투입된 쓰레기 반출

요구사항	분류	기능 요구사항
	번호	SFR-008
	명칭	투입된 쓰레기 반출
상세설명	정의	유형에 맞지 않는 쓰레기가 투입될 시 해당 쓰레기를 반출한다.
	세부내용	<ul style="list-style-type: none"> ■ 쓰레기 반출 <ul style="list-style-type: none"> - 사용자가 투입한 쓰레기가 '페트', '캔', '유리'에 대항되지 않으면 컨베이어벨트를 역회전 시켜 반출시킨다.
산출정보		

[표 20] 기능 요구사항 - 배출량 DB 구축

요구사항	분류	기능 요구사항
	번호	SFR-009
	명칭	배출량 DB 구축
상세설명	정의	저장된 분류 통 측정값을 %계산하여 배출량을 저장한다.
	세부내용	<ul style="list-style-type: none"> ■ 측정값 테이블 <ul style="list-style-type: none"> - 센서가 측정한 값을 저장할 수 있는 테이블이다. ■ 배출량 계산 <ul style="list-style-type: none"> - 센서 측정값을 %로 계산하여 배출량 테이블에 저장한다. ■ 배출량 테이블 <ul style="list-style-type: none"> - 종류별 배출량, 총 배출량 값을 저장할 수 있는 테이블이다.
산출정보		

[표 21] 기능 요구사항 - AI 구축

요구사항	분류	기능 요구사항
	번호	SFR-010
	명칭	AI 구축
상세설명	정의	사용자가 투입한 쓰레기의 이미지를 실시간으로 인식하는 인공지능 구축
	세부내용	<ul style="list-style-type: none"> ■ Object Detection - 투입구 내부에 촬영된 쓰레기 유형을 실시간으로 인식한다.
산출정보		

2.4. 보안 요구사항

[표 22] 보안 요구사항 - 시스템 정보 보안 관리

요구사항	분류	보안 요구사항
	번호	SCR-001
	명칭	시스템 정보 보안 관리 요구사항
상세설명	정의	시스템 및 데이터 침해 예방과 관련된 보안 관리
	세부내용	<ul style="list-style-type: none"> ■ 외부 접근 제한 - 온라인 접근등을 통한 시스템 변경, 수정, 삭제 제한
산출정보		

2.5. 제약사항

[표 23] 제약사항 - 시스템 구축시 준수사항

요구사항	분류	제약사항
	번호	CNR-001
	명칭	시스템 구축 시 준수사항
상세설명	정의	시스템 설계 및 구현과 관련된 기술적 제약 준수사항
	세부내용	<ul style="list-style-type: none"> ■ H/W 제약사항 <ul style="list-style-type: none"> - 제품 전원 오류, 고장, 측정 오류, 합선과 같은 문제점이 발생하지 않도록 관리 필요 ■ S/W 제약사항 <ul style="list-style-type: none"> - 개인 저작권을 침해하지 않도록 저작권 준수 ■ 표준 제약사항 <ul style="list-style-type: none"> - 국가 표준 및 정보화 기술 지원 기관에서 확정한 표준 준수 ■ 법적 제약사항 <ul style="list-style-type: none"> - 개인 정보 보호를 위해 개인 정보 보호법 준수 - 전자정부 서비스 호환성 준수지침(안전행정부 고시)준수
산출정보		

[표 24] 제약사항 - H/W 구현 시 제약사항

요구사항	분류	제약사항
	번호	CNR-002
	명칭	H/W 구현 시 제약사항
상세설명	정의	하드웨어 성능한계로 인한 처리속도 지연
	세부내용	<ul style="list-style-type: none"> ■ 하드웨어 성능한계 <ul style="list-style-type: none"> - AI의 인식률 및 전체적인 제품의 반응속도를 높이려면 고성능 부품을 사용해야 하지만 단가가 높다는 점이 있다.
산출정보		

III. 시스템 설계

1. 개요

1.1. 시스템 개요

- 쓰레기의 정확한 분류를 위해 쓰레기의 분류배출 마크를 인식하고, 사용자에게 음성 또는 화면을 이용해 분류 배출을 유도한다. 투입된 쓰레기는 내부에서는 분류에 맞게 분류한다.

1.2. 소프트웨어의 주요기능

[표 25] 소프트웨어의 주요기능

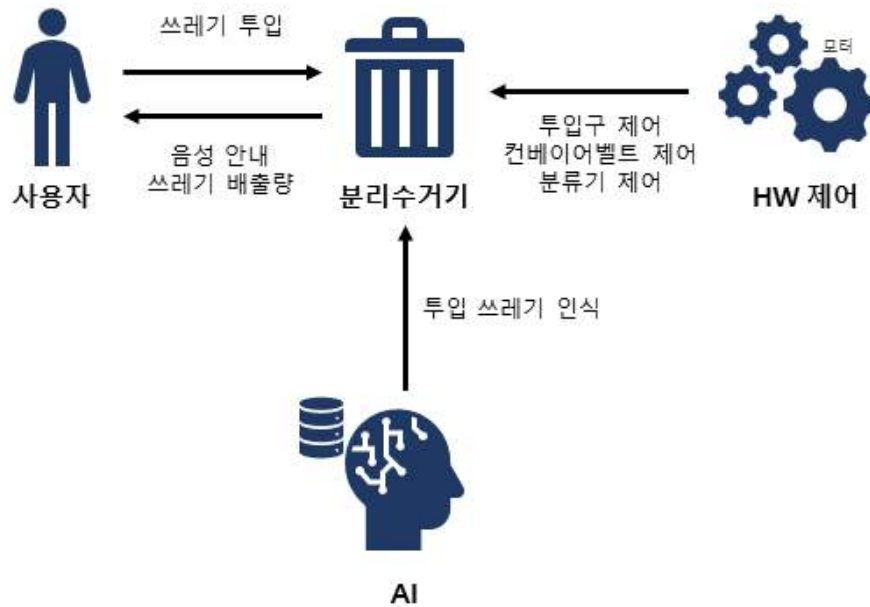
주요 기능	
쓰레기 인식	AI를 이용해 촬영된 쓰레기를 분리수거 가능한지 여부를 판단한다.
HW 제어	단계에 따라 작동되어야 하는 모듈들을 자동적으로 제어한다.
음성 안내	사용자에게 사용 방법을 안내해주고 잘못된 쓰레기를 투입 시 이를 알려주고 수거를 유도한다.

2. 시스템 구조

2.1. 시스템 구조개요

- 사용자가 투입한 쓰레기를 인식하고 '캔', '페트', '유리병'의 유형 중 하나로 분류해 해당 유형에 알맞은 분리수거 통으로 이동시켜주고 사용자가 투입했던 쓰레기들의 종류와 개수를 알려준다.

2.2. 시스템 구조도

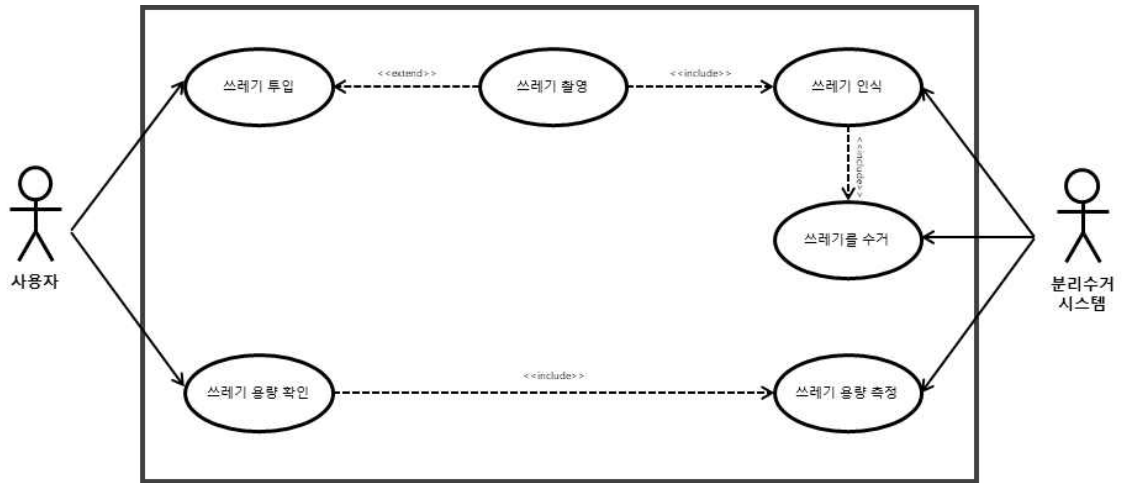


[그림 13] 시스템 구조도

- 사용자 : 별도의 절차 없이 사용자는 '시작' 버튼을 누른 후, 소지하고 있는 쓰레기를 투입한다. 정상적으로 처리될 시 배출한 쓰레기에 대한 정보가 제공된다.
- 분리수거기 : 본 제품의 상태를 알려주고, 터치스크린을 통해 조작할 수 있다. 분리수거할 쓰레기를 투입하고 잘못된 쓰레기를 투입할 시 반출된다.
- HW 제어 : 단계에 따라 투입구, 컨베이어벨트, 분류기 등을 제어한다.
- AI : 투입된 쓰레기의 이미지와 기존에 있는 이미지를 이용해서 투입된 쓰레기의 유형을 분류한다. 만약 분류할 수 없는 쓰레기이면 반출 신호를 보내준다.

3. 소프트웨어 설계

3.1. 유스케이스 다이어그램



[그림 14] 유스케이스 다이어그램

3.2. 유스케이스 명세서

[표 26] 유스케이스 명세서

유스케이스명	쓰레기 투입	액터명	사용자
개요 및 설명	사용자가 쓰레기를 투입한다.		
사전조건	쓰레기를 투입 전 시작 버튼을 누른다.		
이벤트 흐름	1. 정상흐름 가. (1) 사용자는 '시작' 버튼을 누른다. 나. (2) 투입구가 개폐된다. 다. (2) 사용자는 소지하고 있는 쓰레기를 차례대로 투입한다. (E1) 라. (3) 투입을 마치면 '마침' 버튼을 눌러 사용을 종료한다. 2. 예외흐름 가. (E1) 투입하지 않는 경우 1) a. 일정 시간이 지나면 쓰레기통은 유힘상태에 들어간다. 나. (E2) 유형에 맞지 않는 쓰레기가 투입된 경우 1) a. 투입된 쓰레기를 반출하고 컨베이어벨트의 동작을 중지한다. 2)		
후행조건	사용자에게 배출한 쓰레기의 배출량을 표시해준다.		

[표 27] 유스케이스 명세서

유스케이스명	쓰레기 촬영	액터명	분리수거 시스템
개요 및 설명	투입된 쓰레기를 촬영한다.		
사전조건	사용자가 시작버튼을 눌러 사용을 시작한 상태여야 한다.		
이벤트 흐름	3. 정상흐름 가. (1) 사용자가 투입한 쓰레기를 카메라로 촬영한다. (E1) 나. (2) 촬영된 이미지를 이용해 AI를 실행한다. 4. 예외흐름 가. (E1) 카메라가 고장난 경우 1) a. 벨트 작동을 중단하고 고장 음성안내를 한다. 2) b. 투입된 쓰레기를 반출시킨다.		
후행조건	촬영된 이미지를 이용해 쓰레기를 인식한다.		

[표 28] 유스케이스 명세서

유스케이스명	쓰레기 인식	액터명	분리수거 시스템
개요 및 설명	촬영된 쓰레기를 인식한다.		
사전조건	쓰레기가 정상적으로 촬영된 상태여야 한다.		
이벤트 흐름	5. 정상흐름 가. (1) 촬영된 쓰레기의 이미지와 DB에 있는 이미지를 비교한다. 나. (2) 가장 유사한 유형으로 분류한다. (E1) 6. 예외흐름 가. (E1) 분리 불가능 쓰레기가 촬영된 경우 1) a. 쓰레기 오투입 음성안내를 한다. 2) b. 투입된 쓰레기를 반출시킨다.		
후행조건	투입된 쓰레기는 쓰레기통 내부에서 분류한다.		

[표 29] 유스케이스 명세서

유스케이스명	쓰레기 수거	액터명	분리수거 시스템
개요 및 설명	투입된 쓰레기를 내부에서 분리수거 한다.		
사전조건	알맞게 분류된 쓰레기가 투입되어 있어야 한다.		
이벤트 흐름	<p>7. 정상흐름</p> <p>가. (1) 투입구의 개폐기가 열린다.</p> <p>나. (2) 사용자가 투입구에 쓰레기를 넣는다.</p> <p>다. (3) 투입된 쓰레기는 컨베이어 벨트를 타고 이동한다.</p> <p>라. (4) 내부에 있는 분류기가 쓰레기를 분류함으로 넣는다. (E1)</p> <p>마. (5) 투입을 정지하면 투입 여부를 확인하여 벨트를 정지하고 바. 투입구를 폐쇄한다.</p> <p>8. 예외흐름</p> <p>가. (E1) 쓰레기 용량이 꽉 찬 경우</p> <p>1) a. 쓰레기 용량이 초과했다는 안내를 표시 후, 투입구를</p> <p>2) 폐쇄한다.</p>		
후행조건	쓰레기를 수거 후 잔여 용량을 확인한다.		

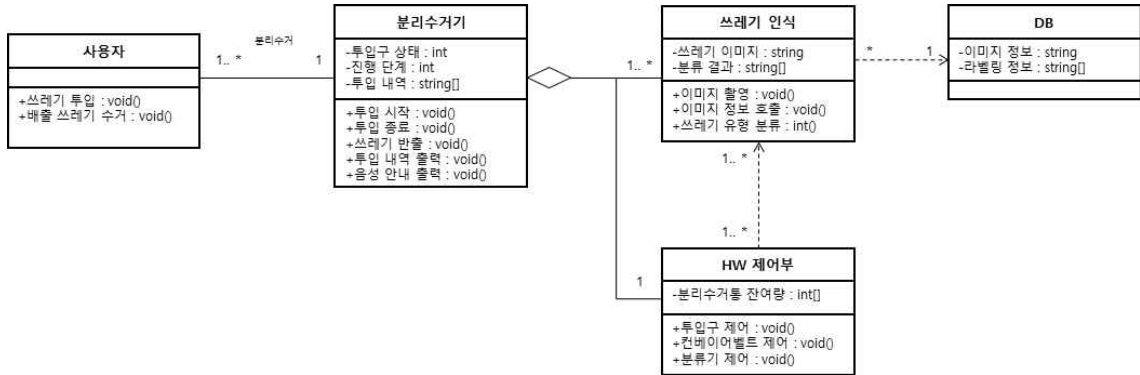
[표 30] 유스케이스 명세서

유스케이스명	쓰레기 용량 측정	액터명	분리수거 시스템
개요 및 설명	분리수거 통의 잔여 용량을 측정한다.		
사전조건	사용 전 후로 쓰레기통의 용량을 측정한다.		
이벤트 흐름	<p>9. 정상흐름</p> <p>가. (1) 분리수거기의 전원이 켜진다.</p> <p>나. (2) 용량을 측정 후 저장한다.</p> <p>다. (3) 사용자가 사용 후 용량을 측정하여 갱신한다.</p>		
후행조건			

[표 31] 유스케이스 명세서

유스케이스명	쓰레기 용량 확인	액터명	사용자
개요 및 설명	사용자가 투입한 쓰레기들의 종류와 양을 알려준다.		
사전조건	사용자가 쓰레기를 투입하고 분류에 맞게 쓰레기를 수거한 상태이다.		
이벤트 흐름	<p>10. 정상흐름</p> <p>가. (1) 쓰레기통은 DB에서 사용자가 투입한 쓰레기에 대한 정보를 가져온다.</p> <p>나. (2) 쓰레기 수거를 마친 쓰레기통은 패널과 음성으로 투입된 쓰레기들의 종류와 양을 알려준다.</p>		
후행조건	쓰레기통은 유휴상태에 들어간다.		

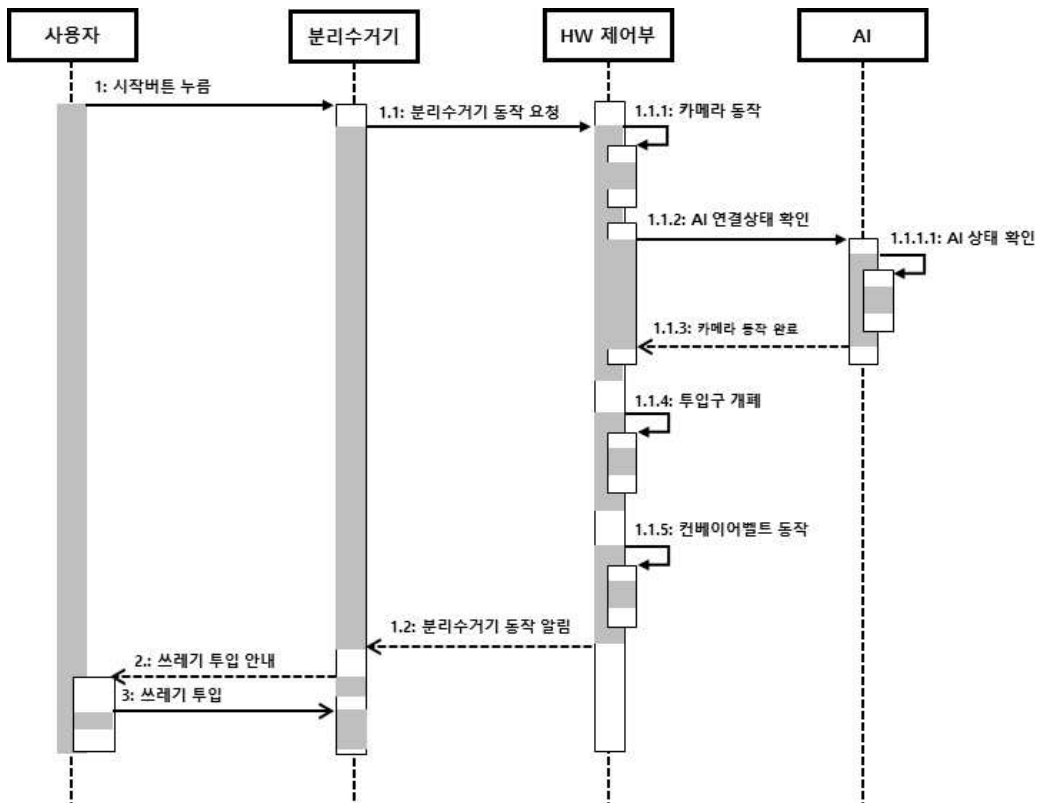
3.3. 클래스 다이어그램



[그림 15] 클래스 다이어그램

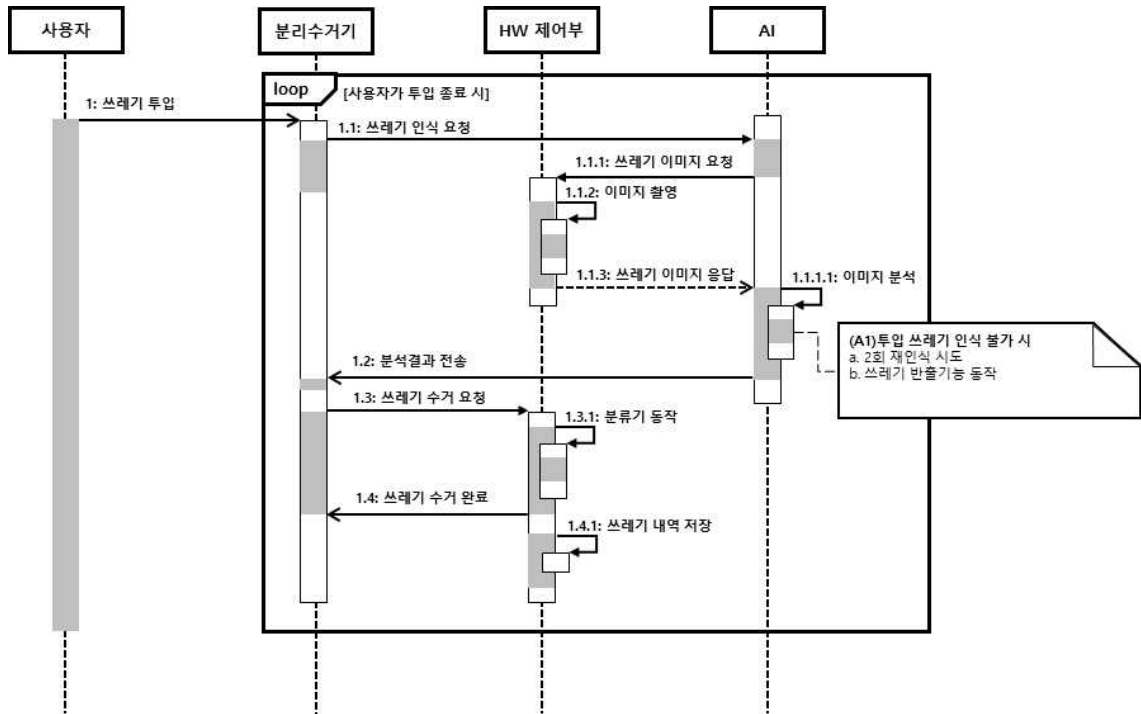
3.4. 시퀀스 다이어그램

3.4.1. 사용자 쓰레기 투입



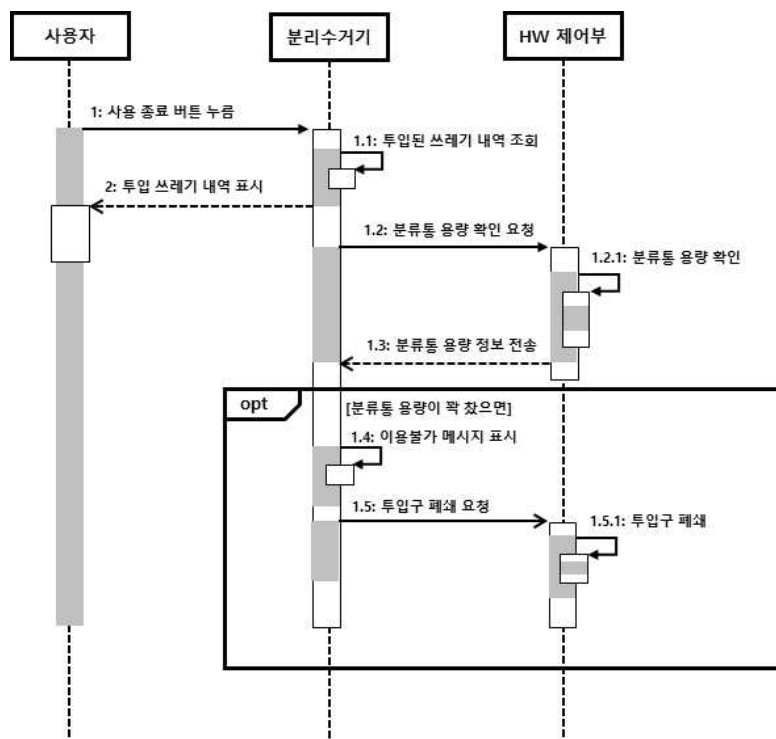
[그림 16] 사용자 쓰레기 투입 시퀀스 다이어그램

3.4.2. 쓰레기 인식



[그림 17] 쓰레기 인식 시퀀스 다이어그램

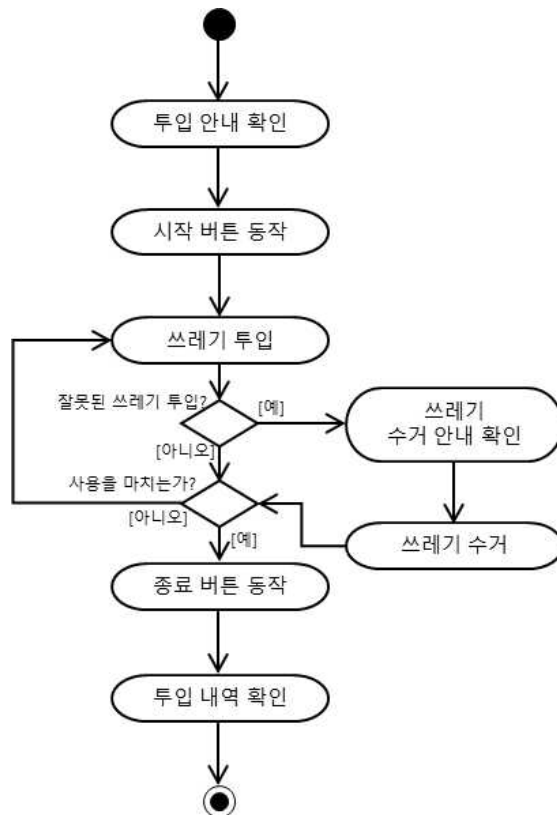
3.4.3. 사용 종료



[그림 18] 사용 종료 시퀀스 다이어그램

3.5. 활동 다이어그램

3.5.1. 사용자 쓰레기 투입




[그림 19] 사용자 쓰레기 투입 활동다이어그램

4. 사용자 인터페이스

4.1. UI 설계서

[표 32] 초기실행 화면 UI

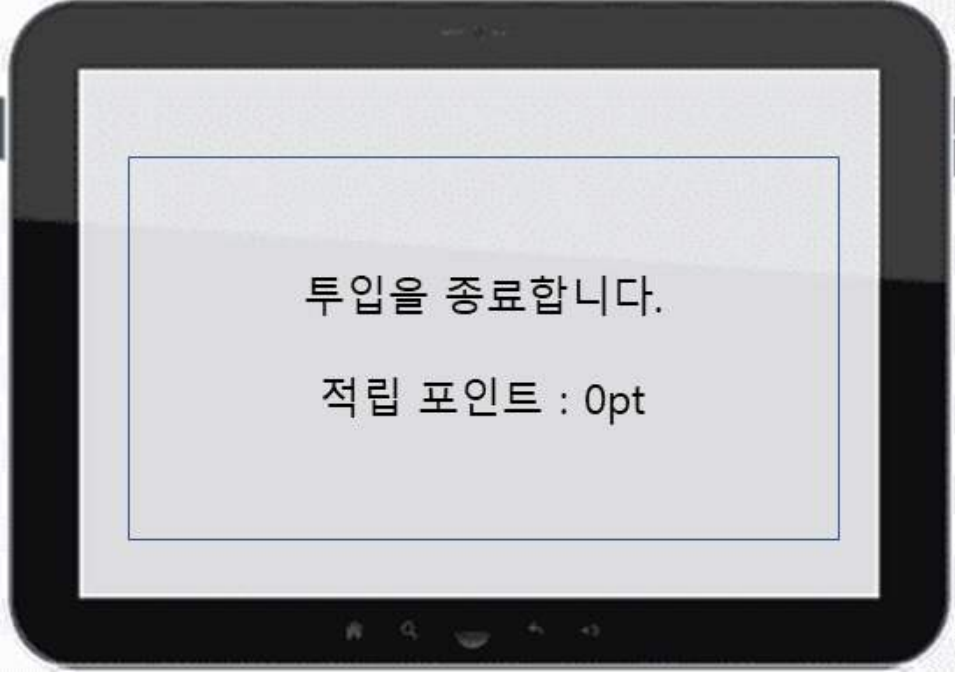
UI 명	초기화면
현재 위치	초기실행
	
■ 제품의 초기화면으로 시작버튼을 누르면 투입하는 단계로 이동한다.	

[표 33] 투입중 화면 UI

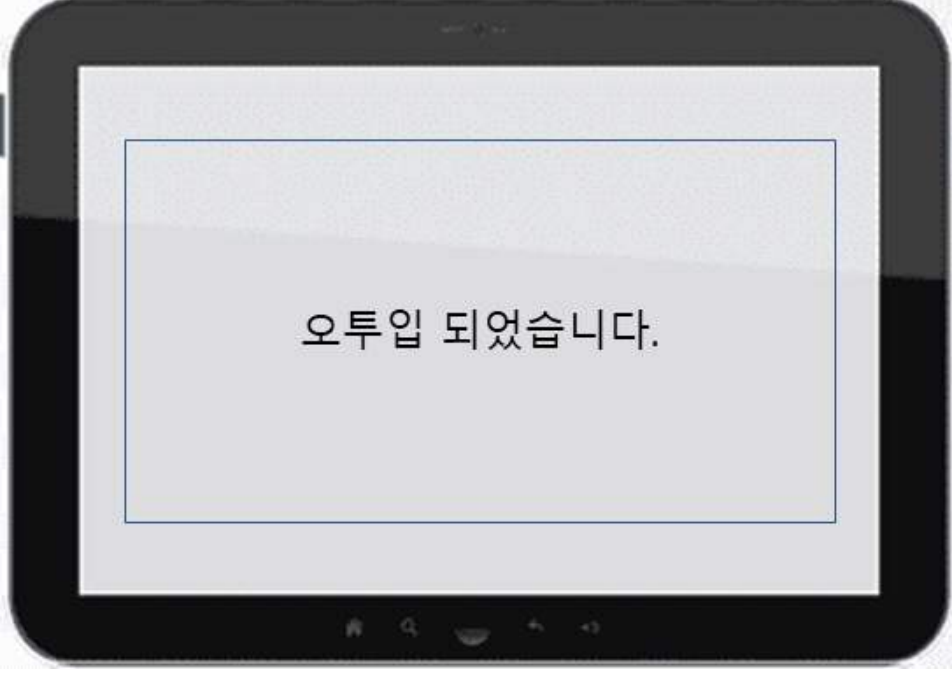
UI 명	투입중 화면
현재 위치	초기화면 → 투입중 화면

- 시작버튼을 누르면 나오는 화면이다.
- 투입한 쓰레기의 종류와 개수를 표시한다.
- 투입 시 바로 계산되어 수치가 바뀐다.
- 종료 버튼을 누를 시, 종료화면으로 돌아간다.

[표 34] 종료화면 UI

UI 명	종료화면
현재 위치	초기화면 → 투입중 화면 → 종료화면
	
<ul style="list-style-type: none"> ■ 투입중 화면에서 '종료'버튼을 누르면 나오는 화면이다. ■ 사용자의 이용을 마친 것을 알려준다. ■ 사용자가 투입한 쓰레기에 따라 적립할 포인트를 표시해준다. ■ 일정 시간이 지나면 초기화면으로 돌아간다. 	

[표 35] 오투입 UI

UI 명	오투입 알림 화면
현재 위치	초기화면 → 투입중 화면 → 오투입 알림 화면
	
<ul style="list-style-type: none"> ■ 사용자가 잘못된 쓰레기를 투입 시 나오는 화면이다. ■ 사용자에게 잘못된 쓰레기가 투입됐음을 알려준다. ■ 사용자가 반출된 쓰레기를 수거하면 투입중 화면으로 돌아간다. 	

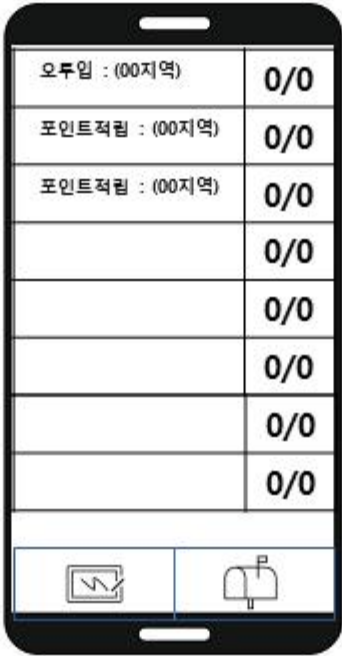
[표 36] 앱 로그인 화면 UI

UI 명	앱 로그인 화면
현재 위치	로그인 화면
	
<p>■ ID와 PW를 입력 후 로그인 버튼을 누르면 로그인이 가능하다.</p>	

[표 37] 앱 메인화면 UI

UI 명	메인화면
현재 위치	로그인 → 메인화면
<div data-bbox="619 510 965 1162" data-label="Image"> </div>	
<p>■ 사용자가 투입한 쓰레기의 종류와 양, 적립된 포인트를 표시해준다.</p>	

[표 43] 앱 메세지함 UI

UI 명	계정관리 화면
현재 위치	로그인 → 메인화면 → 메시지함
	
<ul style="list-style-type: none"> ■ 메인화면에서 메시지함 버튼을 누르면 나오는 화면이다. ■ 어떤 기기에서 이용했는지의 내역을 보여준다. ■ 어떤 기기에서 오투입을 발생시켰는지 보여준다. 	

IV. 소스 코드

1. 하드웨어

1.1. 모듈제어

[표 49] 아두이노 제어 코드

```
#include<Servo.h>
/*해야할 일 : 감지센서 주기적 측정, 투입구 개방 시 LED 활성화*/

const int LED = 13;

/* 서보모터 객체생성 */
Servo servoL; //왼쪽 분류 서보
Servo servoR; //오른쪽 분류 서보
Servo servoB; //벨트 서보
Servo servoD; //투입구 서보

/* 서보모터 Pin */
int servoLP = 11;
int servoRP = 10;
int servoBP = 7;
int servoDP = 8;

/* 서보모터 각도 초기화 */
int valL = 53; //90도 보정
int valR = 160; //90도 보정
int valB = 0;
int valD = 0;

/* 거리센서 변수 */
int Vcc = 3000;
int dist1 = 0;
int dist2 = 0;

/* 입력 선택 변수 */
String in_data = "";

void setup() {
    Serial.begin(9600);

    /* 서보모터 연결 */
```

```

servoL.attach(servoLP); //L 서보모터 연결
servoR.attach(servoRP); //R 서보모터 연결
servoB.attach(servoBP); //B 서보모터 연결
servoD.attach(servoDP); //D 서보모터 연결
servoL.write(valL);
servoR.write(valR);

/* LED 설정 */
pinMode(LED, OUTPUT);
digitalWrite(LED,LOW);

/* 기능 선택 화면 */
/*Serial.println("Arduino Started");
Serial.println(" D1 : 투입구 개방");
Serial.println(" D0 : 투입구 폐쇄");
Serial.println(" B1 : 벨트 가동");
Serial.println(" B0 : 벨트 중단");
Serial.println(" C0 : 쓰레기 중간");
Serial.println(" C1 : 쓰레기 좌측");
Serial.println(" C2 : 쓰레기 우측");
Serial.println(" L1 : LED ON");
Serial.println(" L0 : LED OFF");
Serial.println(" R1 : 거리측정");*/
}

void loop(){
  if(Serial.available()){
    /* 기능 선택 입력 */
    String in_data = Serial.readStringUntil('\n');

    /* 투입구 제어 */
    if(in_data == "D1"){
      //Serial.println("D1; 투입구 개방");
      servoD.attach(servoDP); //서보모터 재연결
      for (valD = 0; valD <140; valD++){
        servoD.write(valD);
        delay(10);
      }
      delay(1800);
      servoD.detach();
    }
  }
}

```

```

} else if(in_data == "D0"){ /*투입구 폐쇄*/
    //Serial.println("D0; 투입구 폐쇄");
    servoD.attach(servoDP); //서보모터 재연결
    for (valD = 100; valD > 0; valD--){
        servoD.write(valD);
        delay(10);
    }
    delay(1000);
    servoD.detach();
}
/* 벨트 제어 */
else if(in_data == "B1"){ /*벨트 가동*/
    servoB.attach(servoBP); //서보모터 재연결
    //Serial.println("B1; 벨트 가동");
    for (valB = 100; valB > 0; valB--){
        servoB.write(valB);
        delay(20);
    }
} else if(in_data == "B0"){ /*벨트 중단*/
    //Serial.println("B0; 벨트 중단");
    servoB.detach();
    delay(1000);
} else if(in_data == "B2"){ /*벨트 가동*/
    servoB.attach(servoBP); //서보모터 재연결
    //Serial.println("B2; 벨트 반전");
    for (valB = 0; valB < 360; valB++){
        servoB.write(valB);
        delay(10);
    }
    delay(1000);
    servoB.detach();
}
/* 쓰레기 분배기 제어 */
else if(in_data == "C0"){ /*가운데로 쓰레기 보내기*/
    //Serial.println("C0; 쓰레기 중간, 초기상태");
    servoL.write(valL);
    delay(100);
    servoR.write(valR);
    //delay(1000);
} else if(in_data == "C1"){ /*좌측으로 쓰레기 보내기*/

```

```

        //Serial.println("C1; 쓰레기 좌측");
        servoL.write(128);
        delay(100);
        servoR.write(valR);
        //delay(1000);
    } else if(in_data == "C2"){ /*우측으로 쓰레기 보내기*/
        //Serial.println("C2; 쓰레기 우측");
        servoL.write(valL);
        delay(100);
        servoR.write(110);
        //delay(1000);
    }
    /* 거리 측정 */
    else if(in_data == "R1"){ /*거리측정*/
        int volt1 = map(analogRead(A0), 0, 1023, 0, Vcc);
        int volt2 = map(analogRead(A1), 0, 1023, 0, Vcc);
        dist1 = (27.61 / (volt1 - 0.1696)) * 1000;
        dist2 = (27.61 / (volt2 - 0.1696)) * 1000;
        //Serial.println("R1; 거리 측정");
        Serial.print(dist1);
        //Serial.print(" / ");
        Serial.print(dist2);
        //Serial.println();
    }
    /* LED 제어 */
    else if(in_data == "L1"){
        turnLED(1);
    } else if(in_data == "L0"){
        turnLED(0);
    } else {
        //Serial.println("잘못된 키워드 입력됨");
    }
}

/* LED 제어 함수 */
void turnLED(int i){
    if(i == 1){
        digitalWrite(LED, HIGH);
        delay(100);
    }
}

```

```

    } else if(i== 0){
        digitalWrite(LED, LOW);
        delay(100);
    }

```

2. 이미지 인식

2.1. Object Detection

[표 50] 텐서플로우 이미지 인식 코드 1

```

import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow._api.v2.compat.v1 as tf
tf.disable_v2_behavior()
import zipfile
from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image
import cv2
cpa = cv2.VideoCapture(1)
sys.path.append("..")
from utils import label_map_util
from utils import visualization_utils as vis_util

obj = ""

cap = cv2.VideoCapture(1)
if tf.__version__ < '1.4.0':
    raise ImportError('Please upgrade your tensorflow installation to v1.4.* or later!')

MODEL_NAME = 'ssd_mobilenet_v1_coco_2018_01_28'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE =
'http://download.tensorflow.org/models/object_detection/'

```

```

PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'
PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')
NUM_CLASSES = 90
opener = urllib.request.URLopener()
opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)
tar_file = tarfile.open(MODEL_FILE)
for file in tar_file.getmembers():
    file_name = os.path.basename(file.name)
    if 'frozen_inference_graph.pb' in file_name:
        tar_file.extract(file, os.getcwd())

detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories =
label_map_util.convert_label_map_to_categories(label_map,max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

def load_image_into_numpy_array(image):
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape((im_height, im_width, 3)).astype(np.uint8)

PATH_TO_TEST_IMAGES_DIR = 'test_images'
TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR, 'image{}.jpg'.format(i)) for i in range(1, 3) ]

IMAGE_SIZE = (12, 8)
with detection_graph.as_default():
    with tf.Session() as sess:
        while True:
            ret, image_np = cap.read()
            image_np_expanded = np.expand_dims(image_np, axis=0)

```

```

        image_tensor =
detection_graph.get_tensor_by_name('image_tensor:0')
        boxes =
detection_graph.get_tensor_by_name('detection_boxes:0')
        scores =
detection_graph.get_tensor_by_name('detection_scores:0')
        classes =
detection_graph.get_tensor_by_name('detection_classes:0')
        num_detections =
detection_graph.get_tensor_by_name('num_detections:0')
        (boxes, scores, classes, num_detections) = sess.run( [boxes,
scores, classes, num_detections], feed_dict={image_tensor:
image_np_expanded})
        vis_util.visualize_boxes_and_labels_on_image_array( image_np,
np.squeeze(boxes), np.squeeze(classes).astype(np.int32), np.squeeze(scores),
category_index, use_normalized_coordinates=True, line_thickness=8)
        cv2.imshow('object detection', cv2.resize(image_np, (800,600)))
        result = [category_index.get(i) for i in classes[0]][0]['name']
        if result == "cup" or "fire hydrant":
            obj = "can"

        if result == "bottle":
            obj = "pet" # 값을 보낸다
        if cv2.waitKey(25) & 0xFF == ord('q'):
            cv2.destroyAllWindows()
            break

```

[표 52] 텐서플로우 이미지 인식 코드 2

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import copy
import functools
import os

import tensorflow.compat.v1 as tf
import tensorflow.compat.v2 as tf2
import tf_slim as slim

from object_detection import eval_util
from object_detection import exporter as exporter_lib
from object_detection import inputs
from object_detection.builders import graph_rewriter_builder
from object_detection.builders import model_builder
from object_detection.builders import optimizer_builder
from object_detection.core import standard_fields as fields
from object_detection.utils import config_util
from object_detection.utils import label_map_util
from object_detection.utils import ops
from object_detection.utils import shape_utils
from object_detection.utils import variables_helper
from object_detection.utils import visualization_utils as vis_utils

# pylint: disable=g-import-not-at-top
try:
    from tensorflow.contrib import learn as contrib_learn
except ImportError:
    # TF 2.0 doesn't ship with contrib.
    pass
# pylint: enable=g-import-not-at-top

# A map of names to methods that help build the model.
MODEL_BUILD_UTIL_MAP = {
    'get_configs_from_pipeline_file':
        config_util.get_configs_from_pipeline_file,
    'create_pipeline_proto_from_configs':
```



```

        config_util.create_pipeline_proto_from_configs,
        'merge_external_params_with_configs':
            config_util.merge_external_params_with_configs,
        'create_train_input_fn':
            inputs.create_train_input_fn,
        'create_eval_input_fn':
            inputs.create_eval_input_fn,
        'create_predict_input_fn':
            inputs.create_predict_input_fn,
        'detection_model_fn_base': model_builder.build,
    }

def _prepare_groundtruth_for_eval(detection_model, class_agnostic,
                                  max_number_of_boxes):
    """Extracts groundtruth data from detection_model and prepares it for
    eval.

    Args:
        detection_model: A `DetectionModel` object.
        class_agnostic: Whether the detections are class_agnostic.
        max_number_of_boxes: Max number of groundtruth boxes.

    Returns:
        A tuple of:
        groundtruth: Dictionary with the following fields:
            'groundtruth_boxes': [batch_size, num_boxes, 4] float32 tensor of
            boxes,
                in normalized coordinates.
            'groundtruth_classes': [batch_size, num_boxes] int64 tensor of
            1-indexed
                classes.
            'groundtruth_masks': 4D float32 tensor of instance masks (if
            provided in
                groundtruth)
            'groundtruth_is_crowd': [batch_size, num_boxes] bool tensor indicating
            is_crowd annotations (if provided in groundtruth).
            'groundtruth_area': [batch_size, num_boxes] float32 tensor indicating
            the area (in the original absolute coordinates) of annotations (if
            provided in groundtruth).

```

```

'num_groundtruth_boxes': [batch_size] tensor containing the
maximum number
    of groundtruth boxes per image..
'groundtruth_keypoints': [batch_size, num_boxes, num_keypoints, 2]
float32
    tensor of keypoints (if provided in groundtruth).
'groundtruth_dp_num_points_list': [batch_size, num_boxes] int32 tensor
    with the number of DensePose points for each instance (if
provided in
    groundtruth).
'groundtruth_dp_part_ids_list': [batch_size, num_boxes,
    max_sampled_points] int32 tensor with the part ids for each
DensePose
    sampled point (if provided in groundtruth).
'groundtruth_dp_surface_coords_list': [batch_size, num_boxes,
    max_sampled_points, 4] containing the DensePose surface
coordinates for
    each sampled point (if provided in groundtruth).
'groundtruth_track_ids_list': [batch_size, num_boxes] int32 tensor
    with track ID for each instance (if provided in groundtruth).
'groundtruth_group_of': [batch_size, num_boxes] bool tensor
indicating
    group_of annotations (if provided in groundtruth).
'groundtruth_labeled_classes': [batch_size, num_classes] int64
    tensor of 1-indexed classes.
'groundtruth_verified_neg_classes': [batch_size, num_classes] float32
    K-hot representation of 1-indexed classes which were verified as
not
    present in the image.
'groundtruth_not_exhaustive_classes': [batch_size, num_classes] K-hot
    representation of 1-indexed classes which don't have all of their
    instances marked exhaustively.
class_agnostic: Boolean indicating whether detections are class
agnostic.
"""
input_data_fields = fields.InputDataFields()
groundtruth_boxes = tf.stack(
    detection_model.groundtruth_lists(fields.BoxListFields.bboxes))
groundtruth_boxes_shape = tf.shape(groundtruth_boxes)
# For class-agnostic models, groundtruth one-hot encodings collapse to

```

```

all
# ones.
if class_agnostic:
    groundtruth_classes_one_hot = tf.ones(
        [groundtruth_boxes_shape[0], groundtruth_boxes_shape[1], 1])
else:
    groundtruth_classes_one_hot = tf.stack(
        detection_model.groundtruth_lists(fields.BoxListFields.classes))
label_id_offset = 1 # Applying label id offset (b/63711816)
groundtruth_classes = (
    tf.argmax(groundtruth_classes_one_hot, axis=2) + label_id_offset)
groundtruth = {
    input_data_fields.groundtruth_boxes: groundtruth_boxes,
    input_data_fields.groundtruth_classes: groundtruth_classes
}

if detection_model.groundtruth_has_field(fields.BoxListFields.masks):
    groundtruth[input_data_fields.groundtruth_instance_masks] = tf.stack(
        detection_model.groundtruth_lists(fields.BoxListFields.masks))

if detection_model.groundtruth_has_field(fields.BoxListFields.is_crowd):
    groundtruth[input_data_fields.groundtruth_is_crowd] = tf.stack(
        detection_model.groundtruth_lists(fields.BoxListFields.is_crowd))

if
detection_model.groundtruth_has_field(input_data_fields.groundtruth_area):
    groundtruth[input_data_fields.groundtruth_area] = tf.stack(
detection_model.groundtruth_lists(input_data_fields.groundtruth_area))

if detection_model.groundtruth_has_field(fields.BoxListFields.keypoints):
    groundtruth[input_data_fields.groundtruth_keypoints] = tf.stack(
        detection_model.groundtruth_lists(fields.BoxListFields.keypoints))

if detection_model.groundtruth_has_field(
    fields.BoxListFields.keypoint_depths):
    groundtruth[input_data_fields.groundtruth_keypoint_depths] = tf.stack(
detection_model.groundtruth_lists(fields.BoxListFields.keypoint_depths))
    groundtruth[

```

```

        input_data_fields.groundtruth_keypoint_depth_weights] = tf.stack(
            detection_model.groundtruth_lists(
                fields.BoxListFields.keypoint_depth_weights))

    if detection_model.groundtruth_has_field(
        fields.BoxListFields.keypoint_visibilities):
        groundtruth[input_data_fields.groundtruth_keypoint_visibilities] = tf.stack(
            detection_model.groundtruth_lists(
                fields.BoxListFields.keypoint_visibilities))

    if detection_model.groundtruth_has_field(fields.BoxListFields.group_of):
        groundtruth[input_data_fields.groundtruth_group_of] = tf.stack(
            detection_model.groundtruth_lists(fields.BoxListFields.group_of))

    label_id_offset_paddings = tf.constant([[0, 0], [1, 0]])
    if detection_model.groundtruth_has_field(
        input_data_fields.groundtruth_verified_neg_classes):
        groundtruth[input_data_fields.groundtruth_verified_neg_classes] = tf.pad(
            tf.stack(detection_model.groundtruth_lists(
                input_data_fields.groundtruth_verified_neg_classes)),
            label_id_offset_paddings)

    if detection_model.groundtruth_has_field(
        input_data_fields.groundtruth_not_exhaustive_classes):
        groundtruth[
            input_data_fields.groundtruth_not_exhaustive_classes] = tf.pad(
            tf.stack(detection_model.groundtruth_lists(
                input_data_fields.groundtruth_not_exhaustive_classes)),
            label_id_offset_paddings)

    if detection_model.groundtruth_has_field(
        fields.BoxListFields.densepose_num_points):
        groundtruth[input_data_fields.groundtruth_dp_num_points] = tf.stack(
            detection_model.groundtruth_lists(
                fields.BoxListFields.densepose_num_points))
    if detection_model.groundtruth_has_field(
        fields.BoxListFields.densepose_part_ids):
        groundtruth[input_data_fields.groundtruth_dp_part_ids] = tf.stack(
            detection_model.groundtruth_lists(
                fields.BoxListFields.densepose_part_ids))

```

```

if detection_model.groundtruth_has_field(
    fields.BoxListFields.densepose_surface_coords):
    groundtruth[input_data_fields.groundtruth_dp_surface_coords] = tf.stack(
        detection_model.groundtruth_lists(
            fields.BoxListFields.densepose_surface_coords))

if detection_model.groundtruth_has_field(fields.BoxListFields.track_ids):
    groundtruth[input_data_fields.groundtruth_track_ids] = tf.stack(
        detection_model.groundtruth_lists(fields.BoxListFields.track_ids))

if detection_model.groundtruth_has_field(
    input_data_fields.groundtruth_labeled_classes):
    groundtruth[input_data_fields.groundtruth_labeled_classes] = tf.pad(
        tf.stack(
            detection_model.groundtruth_lists(
                input_data_fields.groundtruth_labeled_classes)),
        label_id_offset_paddings)

groundtruth[input_data_fields.num_groundtruth_boxes] = (
    tf.tile([max_number_of_boxes],
    multiples=[groundtruth_boxes_shape[0]]))
return groundtruth

def unstack_batch(tensor_dict, unpad_groundtruth_tensors=True):
    """Unstacks all tensors in `tensor_dict` along 0th dimension.

    Unstacks tensor from the tensor dict along 0th dimension and returns a
    tensor_dict containing values that are lists of unstacked, unpadded
    tensors.

    Tensors in the `tensor_dict` are expected to be of one of the three
    shapes:
    1. [batch_size]
    2. [batch_size, height, width, channels]
    3. [batch_size, num_boxes, d1, d2, ... dn]

    When unpad_groundtruth_tensors is set to true, unstacked tensors of
    form 3
    above are sliced along the `num_boxes` dimension using the value in

```

tensor

field.InputDataFields.num_groundtruth_boxes.

Note that this function has a static list of input data fields and has to be

kept in sync with the InputDataFields defined in core/standard_fields.py

Args:

tensor_dict: A dictionary of batched groundtruth tensors.

unpad_groundtruth_tensors: Whether to remove padding along
`num_boxes`
dimension of the groundtruth tensors.

Returns:

A dictionary where the keys are from fields.InputDataFields and values are
a list of unstacked (optionally unpadded) tensors.

Raises:

ValueError: If unpad_tensors is True and `tensor_dict` does not contain
`num_groundtruth_boxes` tensor.

"""

```
unbatched_tensor_dict = {  
    key: tf.unstack(tensor) for key, tensor in tensor_dict.items()  
}
```

```
if unpad_groundtruth_tensors:
```

```
    if (fields.InputDataFields.num_groundtruth_boxes not in  
        unbatched_tensor_dict):  
        raise ValueError("`num_groundtruth_boxes` not found in tensor_dict. '  
            'Keys available: {}'.format(  
                unbatched_tensor_dict.keys()))
```

```
unbatched_unpadded_tensor_dict = {}
```

```
unpad_keys = set([
```

```
    # List of input data fields that are padded along the num_boxes  
    # dimension. This list has to be kept in sync with InputDataFields
```

```
in
```

```
    # standard_fields.py.  
    fields.InputDataFields.groundtruth_instance_masks,  
    fields.InputDataFields.groundtruth_classes,  
    fields.InputDataFields.groundtruth_boxes,
```

```

fields.InputDataFields.groundtruth_keypoints,
fields.InputDataFields.groundtruth_keypoint_depths,
fields.InputDataFields.groundtruth_keypoint_depth_weights,
fields.InputDataFields.groundtruth_keypoint_visibilities,
fields.InputDataFields.groundtruth_dp_num_points,
fields.InputDataFields.groundtruth_dp_part_ids,
fields.InputDataFields.groundtruth_dp_surface_coords,
fields.InputDataFields.groundtruth_track_ids,
fields.InputDataFields.groundtruth_group_of,
fields.InputDataFields.groundtruth_difficult,
fields.InputDataFields.groundtruth_is_crowd,
fields.InputDataFields.groundtruth_area,
fields.InputDataFields.groundtruth_weights
]).intersection(set(unbatched_tensor_dict.keys()))

for key in unpad_keys:
    unpadded_tensor_list = []
    for num_gt, padded_tensor in zip(
unbatched_tensor_dict[fields.InputDataFields.num_groundtruth_boxes],
    unbatched_tensor_dict[key]):
        tensor_shape = shape_utils.combined_static_and_dynamic_shape(
            padded_tensor)
        slice_begin = tf.zeros([len(tensor_shape)], dtype=tf.int32)
        slice_size = tf.stack(
            [num_gt] + [-1 if dim is None else dim for dim in
tensor_shape[1:]])
        unpadded_tensor = tf.slice(padded_tensor, slice_begin, slice_size)
        unpadded_tensor_list.append(unpadded_tensor)
        unbatched_unpadded_tensor_dict[key] = unpadded_tensor_list

    unbatched_tensor_dict.update(unbatched_unpadded_tensor_dict)

return unbatched_tensor_dict

def provide_groundtruth(model, labels):
    """Provides the labels to a model as groundtruth.

    This helper function extracts the corresponding boxes, classes,

```

keypoints, weights, masks, etc. from the labels, and provides it as groundtruth to the models.

Args:

model: The detection model to provide groundtruth to.

labels: The labels for the training or evaluation inputs.

"""

```
gt_boxes_list = labels[fields.InputDataFields.groundtruth_boxes]
gt_classes_list = labels[fields.InputDataFields.groundtruth_classes]
gt_masks_list = None
if fields.InputDataFields.groundtruth_instance_masks in labels:
    gt_masks_list = labels[
        fields.InputDataFields.groundtruth_instance_masks]
gt_keypoints_list = None
if fields.InputDataFields.groundtruth_keypoints in labels:
    gt_keypoints_list = labels[fields.InputDataFields.groundtruth_keypoints]
gt_keypoint_depths_list = None
gt_keypoint_depth_weights_list = None
if fields.InputDataFields.groundtruth_keypoint_depths in labels:
    gt_keypoint_depths_list = (
        labels[fields.InputDataFields.groundtruth_keypoint_depths])
    gt_keypoint_depth_weights_list = (
        labels[fields.InputDataFields.groundtruth_keypoint_depth_weights])
gt_keypoint_visibilities_list = None
if fields.InputDataFields.groundtruth_keypoint_visibilities in labels:
    gt_keypoint_visibilities_list = labels[
        fields.InputDataFields.groundtruth_keypoint_visibilities]
gt_dp_num_points_list = None
if fields.InputDataFields.groundtruth_dp_num_points in labels:
    gt_dp_num_points_list = labels[
        fields.InputDataFields.groundtruth_dp_num_points]
gt_dp_part_ids_list = None
if fields.InputDataFields.groundtruth_dp_part_ids in labels:
    gt_dp_part_ids_list = labels[
        fields.InputDataFields.groundtruth_dp_part_ids]
gt_dp_surface_coords_list = None
if fields.InputDataFields.groundtruth_dp_surface_coords in labels:
    gt_dp_surface_coords_list = labels[
        fields.InputDataFields.groundtruth_dp_surface_coords]
gt_track_ids_list = None
```



```

if fields.InputDataFields.groundtruth_track_ids in labels:
    gt_track_ids_list = labels[
        fields.InputDataFields.groundtruth_track_ids]
gt_weights_list = None
if fields.InputDataFields.groundtruth_weights in labels:
    gt_weights_list = labels[fields.InputDataFields.groundtruth_weights]
gt_confidences_list = None
if fields.InputDataFields.groundtruth_confidences in labels:
    gt_confidences_list = labels[
        fields.InputDataFields.groundtruth_confidences]
gt_is_crowd_list = None
if fields.InputDataFields.groundtruth_is_crowd in labels:
    gt_is_crowd_list = labels[fields.InputDataFields.groundtruth_is_crowd]
gt_group_of_list = None
if fields.InputDataFields.groundtruth_group_of in labels:
    gt_group_of_list = labels[fields.InputDataFields.groundtruth_group_of]
gt_area_list = None
if fields.InputDataFields.groundtruth_area in labels:
    gt_area_list = labels[fields.InputDataFields.groundtruth_area]
gt_labeled_classes = None
if fields.InputDataFields.groundtruth_labeled_classes in labels:
    gt_labeled_classes = labels[
        fields.InputDataFields.groundtruth_labeled_classes]
gt_verified_neg_classes = None
if fields.InputDataFields.groundtruth_verified_neg_classes in labels:
    gt_verified_neg_classes = labels[
        fields.InputDataFields.groundtruth_verified_neg_classes]
gt_not_exhaustive_classes = None
if fields.InputDataFields.groundtruth_not_exhaustive_classes in labels:
    gt_not_exhaustive_classes = labels[
        fields.InputDataFields.groundtruth_not_exhaustive_classes]
model.provide_groundtruth(
    groundtruth_boxes_list=gt_boxes_list,
    groundtruth_classes_list=gt_classes_list,
    groundtruth_confidences_list=gt_confidences_list,
    groundtruth_labeled_classes=gt_labeled_classes,
    groundtruth_masks_list=gt_masks_list,
    groundtruth_keypoints_list=gt_keypoints_list,
    groundtruth_keypoint_visibilities_list=gt_keypoint_visibilities_list,
    groundtruth_dp_num_points_list=gt_dp_num_points_list,

```

```

groundtruth_dp_part_ids_list=gt_dp_part_ids_list,
groundtruth_dp_surface_coords_list=gt_dp_surface_coords_list,
groundtruth_weights_list=gt_weights_list,
groundtruth_is_crowd_list=gt_is_crowd_list,
groundtruth_group_of_list=gt_group_of_list,
groundtruth_area_list=gt_area_list,
groundtruth_track_ids_list=gt_track_ids_list,
groundtruth_verified_neg_classes=gt_verified_neg_classes,
groundtruth_not_exhaustive_classes=gt_not_exhaustive_classes,
groundtruth_keypoint_depths_list=gt_keypoint_depths_list,

groundtruth_keypoint_depth_weights_list=gt_keypoint_depth_weights_list)

def create_model_fn(detection_model_fn, configs, hparams=None,
use_tpu=False,
                    postprocess_on_cpu=False):
    """Creates a model function for `Estimator`.

    Args:
        detection_model_fn: Function that returns a `DetectionModel` instance.
        configs: Dictionary of pipeline config objects.
        hparams: `HParams` object.
        use_tpu: Boolean indicating whether model should be constructed for
            use on TPU.
        postprocess_on_cpu: When use_tpu and postprocess_on_cpu is true,
            postprocess
                is scheduled on the host cpu.

    Returns:
        `model_fn` for `Estimator`.
    """
    train_config = configs['train_config']
    eval_input_config = configs['eval_input_config']
    eval_config = configs['eval_config']

    def model_fn(features, labels, mode, params=None):
        """Constructs the object detection model.

        Args:

```

```

features: Dictionary of feature tensors, returned from `input_fn`.
labels: Dictionary of groundtruth tensors if mode is TRAIN or EVAL,
        otherwise None.
mode: Mode key from tf.estimator.ModeKeys.
params: Parameter dictionary passed from the estimator.

Returns:
    An `EstimatorSpec` that encapsulates the model and its serving
    configurations.
"""
params = params or {}
total_loss, train_op, detections, export_outputs = None, None, None,
None
is_training = mode == tf.estimator.ModeKeys.TRAIN

# Make sure to set the Keras learning phase. True during training,
# False for inference.
tf.keras.backend.set_learning_phase(is_training)
# Set policy for mixed-precision training with Keras-based models.
if use_tpu and train_config.use_bfloat16:
    from tensorflow.python.keras.engine import base_layer_utils # pylint:
disable=g-import-not-at-top
    # Enable v2 behavior, as `mixed_bfloat16` is only supported in TF
2.0.
    base_layer_utils.enable_v2_dtype_behavior()
    tf2.keras.mixed_precision.experimental.set_policy(
        'mixed_bfloat16')
detection_model = detection_model_fn(
    is_training=is_training, add_summaries=(not use_tpu))
scaffold_fn = None

if mode == tf.estimator.ModeKeys.TRAIN:
    labels = unstack_batch(
        labels,

unpad_groundtruth_tensors=train_config.unpad_groundtruth_tensors)
    elif mode == tf.estimator.ModeKeys.EVAL:
        # For evaling on train data, it is necessary to check whether
groundtruth
        # must be unpadded.

```

```

boxes_shape = (
    labels[fields.InputDataFields.groundtruth_boxes].get_shape()
    .as_list())
unpad_groundtruth_tensors = boxes_shape[1] is not None and not
use_tpu
labels = unstack_batch(
    labels, unpad_groundtruth_tensors=unpad_groundtruth_tensors)

if mode in (tf.estimator.ModeKeys.TRAIN, tf.estimator.ModeKeys.EVAL):
    provide_groundtruth(detection_model, labels)

preprocessed_images = features[fields.InputDataFields.image]

side_inputs = detection_model.get_side_inputs(features)

if use_tpu and train_config.use_bfloat16:
    with tf.tpu.bfloat16_scope():
        prediction_dict = detection_model.predict(
            preprocessed_images,
            features[fields.InputDataFields.true_image_shape], **side_inputs)
        prediction_dict = ops.bfloat16_to_float32_nested(prediction_dict)
else:
    prediction_dict = detection_model.predict(
        preprocessed_images,
        features[fields.InputDataFields.true_image_shape], **side_inputs)

def postprocess_wrapper(args):
    return detection_model.postprocess(args[0], args[1])

if mode in (tf.estimator.ModeKeys.EVAL, tf.estimator.ModeKeys.PREDICT):
    if use_tpu and postprocess_on_cpu:
        detections = tf.tpu.outside_compilation(
            postprocess_wrapper,
            (prediction_dict,
             features[fields.InputDataFields.true_image_shape]))
    else:
        detections = postprocess_wrapper((
            prediction_dict,
            features[fields.InputDataFields.true_image_shape]))

```

```

if mode == tf.estimator.ModeKeys.TRAIN:
    load_pretrained = hparams.load_pretrained if hparams else False
    if train_config.fine_tune_checkpoint and load_pretrained:
        if not train_config.fine_tune_checkpoint_type:
            # train_config.from_detection_checkpoint field is deprecated. For
            # backward compatibility, set
            train_config.fine_tune_checkpoint_type
            # based on train_config.from_detection_checkpoint.
            if train_config.from_detection_checkpoint:
                train_config.fine_tune_checkpoint_type = 'detection'
            else:
                train_config.fine_tune_checkpoint_type = 'classification'
            asg_map = detection_model.restore_map(

fine_tune_checkpoint_type=train_config.fine_tune_checkpoint_type,
    load_all_detection_checkpoint_vars=(
        train_config.load_all_detection_checkpoint_vars))
    available_var_map = (
        variables_helper.get_variables_available_in_checkpoint(
            asg_map,
            train_config.fine_tune_checkpoint,
            include_global_step=False))
    if use_tpu:

        def tpu_scaffold():
            tf.train.init_from_checkpoint(train_config.fine_tune_checkpoint,
                                         available_var_map)

            return tf.train.Scaffold()

        scaffold_fn = tpu_scaffold
    else:
        tf.train.init_from_checkpoint(train_config.fine_tune_checkpoint,
                                     available_var_map)

if mode in (tf.estimator.ModeKeys.TRAIN, tf.estimator.ModeKeys.EVAL):
    if (mode == tf.estimator.ModeKeys.EVAL and
        eval_config.use_dummy_loss_in_eval):
        total_loss = tf.constant(1.0)
        losses_dict = {'Loss/total_loss': total_loss}
    else:

```

```

losses_dict = detection_model.loss(
    prediction_dict, features[fields.InputDataFields.true_image_shape])
losses = [loss_tensor for loss_tensor in losses_dict.values()]
if train_config.add_regularization_loss:
    regularization_losses = detection_model.regularization_losses()
    if use_tpu and train_config.use_bfloat16:
        regularization_losses = ops.bfloat16_to_float32_nested(
            regularization_losses)
    if regularization_losses:
        regularization_loss = tf.add_n(
            regularization_losses, name='regularization_loss')
        losses.append(regularization_loss)
        losses_dict['Loss/regularization_loss'] = regularization_loss
total_loss = tf.add_n(losses, name='total_loss')
losses_dict['Loss/total_loss'] = total_loss

if 'graph_rewriter_config' in configs:
    graph_rewriter_fn = graph_rewriter_builder.build(
        configs['graph_rewriter_config'], is_training=is_training)
    graph_rewriter_fn()

# TODO(rathodv): Stop creating optimizer summary vars in EVAL
mode once we
# can write learning rate summaries on TPU without host calls.
global_step = tf.train.get_or_create_global_step()
training_optimizer, optimizer_summary_vars = optimizer_builder.build(
    train_config.optimizer)

if mode == tf.estimator.ModeKeys.TRAIN:
    if use_tpu:
        training_optimizer = tf.tpu.CrossShardOptimizer(training_optimizer)

# Optionally freeze some layers by setting their gradients to be
zero.
trainable_variables = None
include_variables = (
    train_config.update_trainable_variables
    if train_config.update_trainable_variables else None)
exclude_variables = (
    train_config.freeze_variables

```

```

        if train_config.freeze_variables else None)
    trainable_variables = slim.filter_variables(
        tf.trainable_variables(),
        include_patterns=include_variables,
        exclude_patterns=exclude_variables)

    clip_gradients_value = None
    if train_config.gradient_clipping_by_norm > 0:
        clip_gradients_value = train_config.gradient_clipping_by_norm

    if not use_tpu:
        for var in optimizer_summary_vars:
            tf.summary.scalar(var.op.name, var)
    summaries = [] if use_tpu else None
    if train_config.summarize_gradients:
        summaries = ['gradients', 'gradient_norm', 'global_gradient_norm']
    train_op = slim.optimizers.optimize_loss(
        loss=total_loss,
        global_step=global_step,
        learning_rate=None,
        clip_gradients=clip_gradients_value,
        optimizer=training_optimizer,
        update_ops=detection_model.updates(),
        variables=trainable_variables,
        summaries=summaries,
        name='') # Preventing scope prefix on all variables.

    if mode == tf.estimator.ModeKeys.PREDICT:
        exported_output = exporter_lib.add_output_tensor_nodes(detections)
        export_outputs = {
            tf.saved_model.signature_constants.PREDICT_METHOD_NAME:
                tf.estimator.export.PredictOutput(exported_output)
        }

    eval_metric_ops = None
    scaffold = None
    if mode == tf.estimator.ModeKeys.EVAL:
        class_agnostic = (
            fields.DetectionResultFields.detection_classes not in detections)
        groundtruth = _prepare_groundtruth_for_eval(

```

```

        detection_model, class_agnostic,
        eval_input_config.max_number_of_boxes)
    use_original_images = fields.InputDataFields.original_image in features
    if use_original_images:
        eval_images = features[fields.InputDataFields.original_image]
        true_image_shapes = tf.slice(
            features[fields.InputDataFields.true_image_shape], [0, 0], [-1, 3])
        original_image_spatial_shapes = features[fields.InputDataFields
.original_image_spatial_shape]
    else:
        eval_images = features[fields.InputDataFields.image]
        true_image_shapes = None
        original_image_spatial_shapes = None

    eval_dict = eval_util.result_dict_for_batched_example(
        eval_images,
        features[inputs.HASH_KEY],
        detections,
        groundtruth,
        class_agnostic=class_agnostic,
        scale_to_absolute=True,
        original_image_spatial_shapes=original_image_spatial_shapes,
        true_image_shapes=true_image_shapes)

    if fields.InputDataFields.image_additional_channels in features:
        eval_dict[fields.InputDataFields.image_additional_channels] =
features[
        fields.InputDataFields.image_additional_channels]

    if class_agnostic:
        category_index =
label_map_util.create_class_agnostic_category_index()
    else:
        category_index =
label_map_util.create_category_index_from_labelmap(
            eval_input_config.label_map_path)
    vis_metric_ops = None
    if not use_tpu and use_original_images:
        keypoint_edges = [

```



```

(kp.start, kp.end) for kp in eval_config.keypoint_edge]

eval_metric_op_vis = vis_utils.VisualizeSingleFrameDetections(
    category_index,
    max_examples_to_draw=eval_config.num_visualizations,
    max_boxes_to_draw=eval_config.max_num_boxes_to_visualize,
    min_score_thresh=eval_config.min_score_threshold,
    use_normalized_coordinates=False,
    keypoint_edges=keypoint_edges or None)
vis_metric_ops = eval_metric_op_vis.get_estimator_eval_metric_ops(
    eval_dict)

# Eval metrics on a single example.
eval_metric_ops = eval_util.get_eval_metric_ops_for_evaluators(
    eval_config, list(category_index.values()), eval_dict)
for loss_key, loss_tensor in iter(losses_dict.items()):
    eval_metric_ops[loss_key] = tf.metrics.mean(loss_tensor)
for var in optimizer_summary_vars:
    eval_metric_ops[var.op.name] = (var, tf.no_op())
if vis_metric_ops is not None:
    eval_metric_ops.update(vis_metric_ops)
eval_metric_ops = {str(k): v for k, v in eval_metric_ops.items()}

if eval_config.use_moving_averages:
    variable_averages = tf.train.ExponentialMovingAverage(0.0)
    variables_to_restore = variable_averages.variables_to_restore()
    keep_checkpoint_every_n_hours = (
        train_config.keep_checkpoint_every_n_hours)
    saver = tf.train.Saver(
        variables_to_restore,
        keep_checkpoint_every_n_hours=keep_checkpoint_every_n_hours)
    scaffold = tf.train.Scaffold(saver=saver)

# EVAL executes on CPU, so use regular non-TPU EstimatorSpec.
if use_tpu and mode != tf.estimator.ModeKeys.EVAL:
    return tf.estimator.tpu.TPUEstimatorSpec(
        mode=mode,
        scaffold_fn=scaffold_fn,
        predictions=detections,
        loss=total_loss,

```

```

        train_op=train_op,
        eval_metrics=eval_metric_ops,
        export_outputs=export_outputs)
else:
    if scaffold is None:
        keep_checkpoint_every_n_hours = (
            train_config.keep_checkpoint_every_n_hours)
        saver = tf.train.Saver(
            sharded=True,
            keep_checkpoint_every_n_hours=keep_checkpoint_every_n_hours,
            save_relative_paths=True)
        tf.add_to_collection(tf.GraphKeys.SAVERS, saver)
        scaffold = tf.train.Scaffold(saver=saver)
    return tf.estimator.EstimatorSpec(
        mode=mode,
        predictions=detections,
        loss=total_loss,
        train_op=train_op,
        eval_metric_ops=eval_metric_ops,
        export_outputs=export_outputs,
        scaffold=scaffold)

return model_fn

def create_estimator_and_inputs(run_config,
                               hparams=None,
                               pipeline_config_path=None,
                               config_override=None,
                               train_steps=None,
                               sample_1_of_n_eval_examples=1,
                               sample_1_of_n_eval_on_train_examples=1,
                               model_fn_creator=create_model_fn,
                               use_tpu_estimator=False,
                               use_tpu=False,
                               num_shards=1,
                               params=None,
                               override_eval_num_epochs=True,
                               save_final_config=False,
                               postprocess_on_cpu=False,

```

```

export_to_tpu=None,
**kwargs):
"""Creates `Estimator`, input functions, and steps.

Args:
  run_config: A `RunConfig`.
  hparams: (optional) A `HParams`.
  pipeline_config_path: A path to a pipeline config file.
  config_override: A pipeline_pb2.TrainEvalPipelineConfig text proto to
    override the config from `pipeline_config_path`.
  train_steps: Number of training steps. If None, the number of training
steps
    is set from the `TrainConfig` proto.
  sample_1_of_n_eval_examples: Integer representing how often an eval
example
    should be sampled. If 1, will sample all examples.
  sample_1_of_n_eval_on_train_examples: Similar to
    `sample_1_of_n_eval_examples`, except controls the sampling of
training
    data for evaluation.
  model_fn_creator: A function that creates a `model_fn` for `Estimator`.
    Follows the signature:

    * Args:
      * `detection_model_fn`: Function that returns `DetectionModel`
instance.
      * `configs`: Dictionary of pipeline config objects.
      * `hparams`: `HParams` object.
    * Returns:
      `model_fn` for `Estimator`.

  use_tpu_estimator: Whether a `TPUEstimator` should be returned. If
False,
    an `Estimator` will be returned.
  use_tpu: Boolean, whether training and evaluation should run on TPU.
Only
    used if `use_tpu_estimator` is True.
  num_shards: Number of shards (TPU cores). Only used if
`use_tpu_estimator`
    is True.

```

params: Parameter dictionary passed from the estimator. Only used if
 `use_tpu_estimator` is True.
 override_eval_num_epochs: Whether to overwrite the number of epochs
 to 1 for
 eval_input.
 save_final_config: Whether to save final config (obtained after applying
 overrides) to `estimator.model_dir`.
 postprocess_on_cpu: When use_tpu and postprocess_on_cpu are true,
 postprocess is scheduled on the host cpu.
 export_to_tpu: When use_tpu and export_to_tpu are true,
 `export_savedmodel()` exports a metagraph for serving on TPU
 besides the
 one on CPU.
 **kwargs: Additional keyword arguments for configuration override.

Returns:
 A dictionary with the following fields:
 'estimator': An `Estimator` or `TPUEstimator`.
 'train_input_fn': A training input function.
 'eval_input_fns': A list of all evaluation input functions.
 'eval_input_names': A list of names for each evaluation input.
 'eval_on_train_input_fn': An evaluation-on-train input function.
 'predict_input_fn': A prediction input function.
 'train_steps': Number of training steps. Either directly from input or
 from
 configuration.

```

"""
get_configs_from_pipeline_file = MODEL_BUILD_UTIL_MAP[
    'get_configs_from_pipeline_file']
merge_external_params_with_configs = MODEL_BUILD_UTIL_MAP[
    'merge_external_params_with_configs']
create_pipeline_proto_from_configs = MODEL_BUILD_UTIL_MAP[
    'create_pipeline_proto_from_configs']
create_train_input_fn = MODEL_BUILD_UTIL_MAP['create_train_input_fn']
create_eval_input_fn = MODEL_BUILD_UTIL_MAP['create_eval_input_fn']
create_predict_input_fn =
MODEL_BUILD_UTIL_MAP['create_predict_input_fn']
detection_model_fn_base =
MODEL_BUILD_UTIL_MAP['detection_model_fn_base']
  
```

```

configs = get_configs_from_pipeline_file(
    pipeline_config_path, config_override=config_override)
kwargs.update({
    'train_steps': train_steps,
    'use_bfloat16': configs['train_config'].use_bfloat16 and use_tpu
})
if sample_1_of_n_eval_examples >= 1:
    kwargs.update({
        'sample_1_of_n_eval_examples': sample_1_of_n_eval_examples
    })
if override_eval_num_epochs:
    kwargs.update({'eval_num_epochs': 1})
tf.logging.warning(
    'Forced number of epochs for all eval validations to be 1.')
configs = merge_external_params_with_configs(
    configs, hparams, kwargs_dict=kwargs)
model_config = configs['model']
train_config = configs['train_config']
train_input_config = configs['train_input_config']
eval_config = configs['eval_config']
eval_input_configs = configs['eval_input_configs']
eval_on_train_input_config = copy.deepcopy(train_input_config)
eval_on_train_input_config.sample_1_of_n_examples = (
    sample_1_of_n_eval_on_train_examples)
if override_eval_num_epochs and eval_on_train_input_config.num_epochs
!= 1:
    tf.logging.warning('Expected number of evaluation epochs is 1, but '
        'instead encountered `eval_on_train_input_config`
        `.num_epochs` = '
        '{}. Overwriting `num_epochs` to 1.'.format(
            eval_on_train_input_config.num_epochs))
    eval_on_train_input_config.num_epochs = 1

# update train_steps from config but only when non-zero value is
provided
if train_steps is None and train_config.num_steps != 0:
    train_steps = train_config.num_steps

detection_model_fn = functools.partial(
    detection_model_fn_base, model_config=model_config)

```

```

# Create the input functions for TRAIN/EVAL/PREDICT.
train_input_fn = create_train_input_fn(
    train_config=train_config,
    train_input_config=train_input_config,
    model_config=model_config)
eval_input_fns = []
for eval_input_config in eval_input_configs:
    eval_input_fns.append(
        create_eval_input_fn(
            eval_config=eval_config,
            eval_input_config=eval_input_config,
            model_config=model_config))

eval_input_names = [
    eval_input_config.name for eval_input_config in eval_input_configs
]
eval_on_train_input_fn = create_eval_input_fn(
    eval_config=eval_config,
    eval_input_config=eval_on_train_input_config,
    model_config=model_config)
predict_input_fn = create_predict_input_fn(
    model_config=model_config,
predict_input_config=eval_input_configs[0])

# Read export_to_tpu from hparams if not passed.
if export_to_tpu is None and hparams is not None:
    export_to_tpu = hparams.get('export_to_tpu', False)
tf.logging.info('create_estimator_and_inputs: use_tpu %s, export_to_tpu %s',
                use_tpu, export_to_tpu)
model_fn = model_fn_creator(detection_model_fn, configs, hparams,
use_tpu,
                                postprocess_on_cpu)

if use_tpu_estimator:
    estimator = tf.estimator.tpu.TPUEstimator(
        model_fn=model_fn,
        train_batch_size=train_config.batch_size,
        # For each core, only batch size 1 is supported for eval.
        eval_batch_size=num_shards * 1 if use_tpu else 1,
        use_tpu=use_tpu,

```

```

        config=run_config,
        export_to_tpu=export_to_tpu,
        eval_on_tpu=False, # Eval runs on CPU, so disable eval on TPU
        params=params if params else {})
    else:
        estimator = tf.estimator.Estimator(model_fn=model_fn,
config=run_config)

    # Write the as-run pipeline config to disk.
    if run_config.is_chief and save_final_config:
        pipeline_config_final = create_pipeline_proto_from_configs(configs)
        config_util.save_pipeline_config(pipeline_config_final, estimator.model_dir)

    return dict(
        estimator=estimator,
        train_input_fn=train_input_fn,
        eval_input_fns=eval_input_fns,
        eval_input_names=eval_input_names,
        eval_on_train_input_fn=eval_on_train_input_fn,
        predict_input_fn=predict_input_fn,
        train_steps=train_steps)

def create_train_and_eval_specs(train_input_fn,
                                eval_input_fns,
                                eval_on_train_input_fn,
                                predict_input_fn,
                                train_steps,
                                eval_on_train_data=False,
                                final_exporter_name='Servo',
                                eval_spec_names=None):
    """Creates a `TrainSpec` and `EvalSpec`s.

    Args:
        train_input_fn: Function that produces features and labels on train
data.
        eval_input_fns: A list of functions that produce features and labels on
eval
data.
        eval_on_train_input_fn: Function that produces features and labels for

```

```

        evaluation on train data.
    predict_input_fn: Function that produces features for inference.
    train_steps: Number of training steps.
    eval_on_train_data: Whether to evaluate model on training data.
Default is
    False.
    final_exporter_name: String name given to `FinalExporter`.
    eval_spec_names: A list of string names for each `EvalSpec`.

Returns:
    Tuple of `TrainSpec` and list of `EvalSpecs`. If `eval_on_train_data` is
    True, the last `EvalSpec` in the list will correspond to training data.
The
    rest EvalSpecs in the list are evaluation datas.
"""
train_spec = tf.estimator.TrainSpec(
    input_fn=train_input_fn, max_steps=train_steps)

if eval_spec_names is None:
    eval_spec_names = [str(i) for i in range(len(eval_input_fns))]

eval_specs = []
for index, (eval_spec_name, eval_input_fn) in enumerate(
    zip(eval_spec_names, eval_input_fns)):
    # Uses final_exporter_name as exporter_name for the first eval spec
    for
        # backward compatibility.
        if index == 0:
            exporter_name = final_exporter_name
        else:
            exporter_name = '{}_{}'.format(final_exporter_name, eval_spec_name)
        exporter = tf.estimator.FinalExporter(
            name=exporter_name, serving_input_receiver_fn=predict_input_fn)
    eval_specs.append(
        tf.estimator.EvalSpec(
            name=eval_spec_name,
            input_fn=eval_input_fn,
            steps=None,
            exporters=exporter))

```



```

if eval_on_train_data:
    eval_specs.append(
        tf.estimator.EvalSpec(
            name='eval_on_train', input_fn=eval_on_train_input_fn,
            steps=None))

    return train_spec, eval_specs

def _evaluate_checkpoint(estimator,
                        input_fn,
                        checkpoint_path,
                        name,
                        max_retries=0):
    """Evaluates a checkpoint.

    Args:
        estimator: Estimator object to use for evaluation.
        input_fn: Input function to use for evaluation.
        checkpoint_path: Path of the checkpoint to evaluate.
        name: Namespace for eval summary.
        max_retries: Maximum number of times to retry the evaluation on
            encountering
            a tf.errors.InvalidArgumentError. If negative, will always retry the
            evaluation.

    Returns:
        Estimator evaluation results.
    """
    always_retry = True if max_retries < 0 else False
    retries = 0
    while always_retry or retries <= max_retries:
        try:
            return estimator.evaluate(
                input_fn=input_fn,
                steps=None,
                checkpoint_path=checkpoint_path,
                name=name)
        except tf.errors.InvalidArgumentError as e:
            if always_retry or retries < max_retries:

```

```

        tf.logging.info('Retrying checkpoint evaluation after exception: %s',
e)
        retries += 1
    else:
        raise e

def continuous_eval_generator(estimator,
                             model_dir,
                             input_fn,
                             train_steps,
                             name,
                             max_retries=0):
    """Perform continuous evaluation on checkpoints written to a model
    directory.

    Args:
        estimator: Estimator object to use for evaluation.
        model_dir: Model directory to read checkpoints for continuous
        evaluation.
        input_fn: Input function to use for evaluation.
        train_steps: Number of training steps. This is used to infer the last
        checkpoint and stop evaluation loop.
        name: Namespace for eval summary.
        max_retries: Maximum number of times to retry the evaluation on
        encountering
            a tf.errors.InvalidArgumentError. If negative, will always retry the
            evaluation.

    Yields:
        Pair of current step and eval_results.
    """

    def terminate_eval():
        tf.logging.info('Terminating eval after 180 seconds of no checkpoints')
        return True

    for ckpt in tf.train.checkpoints_iterator(
        model_dir, min_interval_secs=180, timeout=None,
        timeout_fn=terminate_eval):

```

```

tf.logging.info('Starting Evaluation.')
try:
    eval_results = _evaluate_checkpoint(
        estimator=estimator,
        input_fn=input_fn,
        checkpoint_path=ckpt,
        name=name,
        max_retries=max_retries)
    tf.logging.info('Eval results: %s' % eval_results)

    # Terminate eval job when final checkpoint is reached
    current_step = int(os.path.basename(ckpt).split('-')[1])
    yield (current_step, eval_results)
    if current_step >= train_steps:
        tf.logging.info(
            'Evaluation finished after training step %d' % current_step)
        break

except tf.errors.NotFoundError:
    tf.logging.info(
        'Checkpoint %s no longer exists, skipping checkpoint' % ckpt)

def continuous_eval(estimator,
                    model_dir,
                    input_fn,
                    train_steps,
                    name,
                    max_retries=0):
    """Performs continuous evaluation on checkpoints written to a model
    directory.

    Args:
        estimator: Estimator object to use for evaluation.
        model_dir: Model directory to read checkpoints for continuous
        evaluation.
        input_fn: Input function to use for evaluation.
        train_steps: Number of training steps. This is used to infer the last
        checkpoint and stop evaluation loop.

```

```

name: Namespace for eval summary.
max_retries: Maximum number of times to retry the evaluation on
encountering
    a tf.errors.InvalidArgumentError. If negative, will always retry the
    evaluation.
"""
for current_step, eval_results in continuous_eval_generator(
    estimator, model_dir, input_fn, train_steps, name, max_retries):
    tf.logging.info('Step %s, Eval results: %s', current_step, eval_results)

def populate_experiment(run_config,
                       hparams,
                       pipeline_config_path,
                       train_steps=None,
                       eval_steps=None,
                       model_fn_creator=create_model_fn,
                       **kwargs):
    """Populates an `Experiment` object.

    EXPERIMENT CLASS IS DEPRECATED. Please switch to
    tf.estimator.train_and_evaluate. As an example, see model_main.py.

    Args:
        run_config: A `RunConfig`.
        hparams: A `HParams`.
        pipeline_config_path: A path to a pipeline config file.
        train_steps: Number of training steps. If None, the number of training
        steps
            is set from the `TrainConfig` proto.
        eval_steps: Number of evaluation steps per evaluation cycle. If None,
        the
            number of evaluation steps is set from the `EvalConfig` proto.
        model_fn_creator: A function that creates a `model_fn` for `Estimator`.
        Follows the signature:

        * Args:
            * `detection_model_fn`: Function that returns `DetectionModel`
            instance.
            * `configs`: Dictionary of pipeline config objects.

```

```

    * `hparams`: `HParams` object.
    * Returns:
      `model_fn` for `Estimator`.

    **kwargs: Additional keyword arguments for configuration override.

Returns:
    An `Experiment` that defines all aspects of training, evaluation, and
    export.
    """
    tf.logging.warning('Experiment is being deprecated. Please use '
                       'tf.estimator.train_and_evaluate(). See model_main.py
for '
                       'an example.')
    train_and_eval_dict = create_estimator_and_inputs(
        run_config,
        hparams,
        pipeline_config_path,
        train_steps=train_steps,
        eval_steps=eval_steps,
        model_fn_creator=model_fn_creator,
        save_final_config=True,
        **kwargs)
    estimator = train_and_eval_dict['estimator']
    train_input_fn = train_and_eval_dict['train_input_fn']
    eval_input_fns = train_and_eval_dict['eval_input_fns']
    predict_input_fn = train_and_eval_dict['predict_input_fn']
    train_steps = train_and_eval_dict['train_steps']

    export_strategies = [
        contrib_learn.utils.saved_model_export_utils.make_export_strategy(
            serving_input_fn=predict_input_fn)
    ]

    return contrib_learn.Experiment(
        estimator=estimator,
        train_input_fn=train_input_fn,
        eval_input_fn=eval_input_fns[0],
        train_steps=train_steps,
        eval_steps=None,

```

```
export_strategies=export_strategies,  
eval_delay_secs=120,  
)
```

3. 통신

3.1. 소켓 통신

[표 53] 안드로이드 <-> 단말기 소켓 통신 코드 1

```
package com.example.myapplication2;  
  
import android.content.Intent;  
import android.os.Handler;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
  
import androidx.appcompat.app.AppCompatActivity;  
import com.example.myapplication2.StartPage;  
  
import java.io.BufferedReader;  
import java.io.DataInputStream;  
import java.io.DataOutputStream;  
import java.io.IOException;  
import java.io.PrintWriter;  
import java.net.Socket;  
  
public class MainActivity extends AppCompatActivity {  
  
    TextView can_t;  
    TextView pet_t;  
  
    int can = 0;  
    int pet = 0;  
    boolean check=true;  
    Button connect_btn;                // ip 받아오는 버튼
```

```

Button start_btn;
EditText ip_edit;           // ip 에디트
TextView show_text;         // 서버에서온거 보여주는 에디트
// 소켓통신에 필요한것
private String html = "";
private Handler mHandler;
Button btnEnd;
private Socket socket;

private BufferedReader networkReader;
private PrintWriter networkWriter;

private DataOutputStream dos;
private DataInputStream dis;

public static String ip = "";           // IP 번호
private int port = 8080;                // port 번호
int finalLine=0;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    can_t = (TextView) findViewById(R.id.can_text);
    pet_t = (TextView) findViewById(R.id.pet_text);

    // connect_btn = (Button)findViewById(R.id.connect_btn);
    show_text = (TextView)findViewById(R.id.show_text);
    btnEnd = (Button)findViewById(R.id.btnEnd);

    check=true;
    connect();

    btnEnd.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            check=false;

```

```

        Log.w("버퍼","정지정지");
        Intent intent = new Intent(getApplicationContext(),
StartPage.class);
        startActivity(intent);
    }
});

}

/*
@Override
public void onClick(View v) {
    switch(v.getId()){
        case R.id.connect_btn:// ip 받아오는 버튼
            Log.w("connect_btn", "push");
            ip = ip_edit.getText().toString(); // ip 스트링값으로 받음

    }

}

*/

// 로그인 정보 db에 넣어주고 연결시켜야 함.
void connect(){
    System.out.println(ip+"<-----ip");
    mHandler = new Handler();
    String a = "답장 : ";
    Log.w("connect","연결 하는중");
    // 받아오는거
    Thread checkUpdate = new Thread() {
        public void run() {
            // ip받기
            String newip = ip;

            // 서버 접속
            try {
                socket = new Socket(newip, port);
                Log.w("서버 접속됨", "서버 접속됨");
            } catch (IOException e1) {

```



```

        Log.w("서버접속못함", "서버접속못함");
        e1.printStackTrace();
    }

    Log.w("edit 넘어가야 할 값 : ","안드로이드에서 서버로
연결요청");

    // Buffered가 잘못된듯.
    try {
        dos = new
DataOutputStream(socket.getOutputStream()); // output에 보낼꺼 넣음
        dis = new DataInputStream(socket.getInputStream());
        // input에 받을꺼 넣어짐
        dos.writeUTF("안드로이드에서 서버로 연결요청");

    } catch (IOException e) {
        e.printStackTrace();
        Log.w("버퍼", "버퍼생성 잘못된");
    }
    Log.w("버퍼","버퍼생성 잘됨");
    check=true;
    while(true) {
        // 서버에서 받아옴
        try {
            check=true;
            String line = "";
            int line2;
            while (true) {
                //line = (String) dis.readUTF();
                line2 = (int) dis.read();
                //Log.w("서버에서 받아온 값 ", "" + line);
                //Log.w("서버에서 받아온 값 ", "" +
line2+check);

                if(line2>0) {
                    if(check==false){
                        dos.write(99);
                        Log.w("-----서버에서 받아온 값 ",""+
line2);

                        finalLine = line2;

```

```

mHandler.post(new Runnable() {
    @Override
    public void run() {

        show_text.setText("active");

        if(finalLine %6 == 0){

            can_t.setText("1");

        }
        if(finalLine %11 == 0){

            pet_t.setText("1");

        }

    }
});

socket.close();
break;
}
Log.w("-----서버에서 받아온 값 ",""+
line2);

dos.writeUTF(a + line2);
finalLine = line2;
mHandler.post(new Runnable() {
    @Override
    public void run() {

        show_text.setText("active");

        if(finalLine %6 == 0){

            can_t.setText("1");

        }
    }
});

```

```

        if(finalLine %11 == 0){

            pet_t.setText("1");

        }

    }

});
dos.flush();
}

}
} catch (Exception e) {

}

}

};

// 소켓 접속 시도, 버퍼생성
checkUpdate.start();
}

```

[표 54] 안드로이드 <-> 단말기 소켓 통신 코드 2

```
package com.example.myapplication2;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;

public class StartPage extends AppCompatActivity{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.start_page);
        EditText ip_edit = (EditText) findViewById(R.id.ip_edit);
        Button str_btn = (Button) findViewById(R.id.btn_str);
        ip_edit.setText(MainActivity.ip);
        ImageButton setBtn = (ImageButton) findViewById(R.id.setBtn);
        ip_edit.setVisibility(View.INVISIBLE);

        setBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                ip_edit.setVisibility(View.VISIBLE);

            }
        });

        str_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                MainActivity.ip=ip_edit.getText().toString();

                Intent intent = new Intent(getApplicationContext(),
                MainActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

```

    }
}

```

[표 54] 안드로이드 화면 코드 1

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="15px"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:id="@+id/linearLayout">
        <!--android:layout_centerInParent="true"-->

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
            <TextView
                android:id="@+id/show_text"
                android:text="stay"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:textSize="50dp"/>

            </LinearLayout>

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal"/>

```

```

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <TextView
        android:text="can : "
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:textSize="50dp"/>

    <TextView
        android:id="@+id/can_text"
        android:text="0"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:textSize="50dp"/>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:text="pet : "
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:textSize="50dp"/>

    <TextView
        android:id="@+id/pet_text"
        android:text="0"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:textSize="50dp"/>

</LinearLayout>

```



```

        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <ImageButton
            android:id="@+id/setBtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@android:drawable/ic_menu_preferences"
            app:srcCompat="@android:drawable/stat_notify_sdcard_prepare"
        />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:gravity="center_horizontal"
            android:text="\n버튼을 누르면 시작합니다.\n"
            android:textSize="80sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/textView3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:text="\n\n"
            android:gravity="center_horizontal"
            android:textSize="40sp"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/ip_edit"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="ip주소 입력"
            android:gravity="center"
            android:textSize="18sp">

```



```

</EditText>

<Button
    android:id="@+id/btn_str"
    android:layout_width="520dp"
    android:layout_height="240dp"
    android:layout_gravity="center"
    android:insetRight="0dp"
    android:text="시작"
    android:textSize="100sp"
    tools:layout_editor_absoluteX="16dp"
    tools:layout_editor_absoluteY="275dp" />

</LinearLayout>

</android.widget.LinearLayout>

```

3.2. 시리얼 통신

[표 57] 파이썬 <-> 아두이노 시리얼 통신 코드

```

import socket
import sys
import time
import os, subprocess
import threading
import serial
import object
global check
global var

def set():
    os.system('object.exe')

def set2():
    global check
    arduino = serial.Serial('com7', 9600)
    temp = "";
    dist1 = "";
    dist2 = "";
    dist3 = "";
    time.sleep(1)

```

```

print(check)

#while 1:
if (check == True):
    var = "D1".encode('utf-8')
    arduino.write(var)
    print("#D1; 투입구 개방")

if (check == False):
    var = "D0".encode('utf-8')
    arduino.write(var)
    print("#D0; 투입구 폐쇄")
    time.sleep(2)
    arduino.close()

if (check == True):
    var = "B1".encode('utf-8')
    arduino.write(var)
    print("#B1; 벨트 가동")

if (check == False):
    var = "B0".encode('utf-8')
    arduino.write(var)
    print("#B0; 벨트 중단")
    time.sleep(2)
    arduino.close()

# elif (var == "B2\n"):
#     var = var.encode('utf-8')
#     arduino.write(var)
#     print("#B2; 벨트 반전")

# elif (var == "C0\n"):
#     var = var.encode('utf-8')
#     arduino.write(var)
#     print("#C0; 쓰레기 중간, 초기상태")

if (object.obj == "pet"):
    var = "C1".encode('utf-8')
    arduino.write(var)

```

```

print("#C1; 쓰레기 좌측")

if (object.obj == "can"):
    var = "C2".encode('utf-8')
    arduino.write(var)
    print("#C2; 쓰레기 우측")

if (check == True):
    var = "LED ON".encode('utf-8')
    arduino.write(var)
    print("LED ON")

if (check == False):
    var = "LED OFF".encode('utf-8')
    arduino.write(var)
    print("LED OFF")
    time.sleep(2)
    arduino.close()

if check == False:
    var = "R1".encode('utf-8')
    arduino.write(var)
    """ 1번 째 센서 값 출력 """
    time.sleep(2)
    arduino.close()
    for i in range(2):
        temp = temp + str(arduino.read().decode('utf-8'))
    dist1 = int(temp)
    print(dist1)
    temp = ""
    for i in range(2):
        temp = temp + str(arduino.read().decode('utf-8'))
    dist2 = int(temp)
    print(dist2)
    time.sleep(1)

def set1(data2):
    t3 = threading.Thread(target=set)

```

```

t3.start()
while True :
    data2
    # print(data2.encode())
    # client_sock.send(data)
    # client_sock.send(data2.to_bytes(4, byteorder='little'))
    i = 2

    # 값하나 보냄(사용자가 입력한 숫자)
    client_sock.sendall(data2.to_bytes(4, byteorder='little')) # int에서
바이트 변환
    # .to_bytes(4, byteorder='little')

    # 안드로이드에서 값 받으면 "하나받았습니다 : 숫자" 보낼 것 받음
    data = client_sock.recv(1024)
    print("'" + str(data.decode("utf-8")) + "'")

    if data.decode("utf-8") == "c":
        print("stop      "+data.decode("utf-8"))
        print(os.system('tasklist')) # 프로세스 목록 출력
        # os.system('taskkill /f /pid 11172') #pid를 사용한 프로세스
종료
        os.system('taskkill /f /im object.exe') # 프로세스명을 사용한
프로세스 종료
        data2 = 1
        global check
        check = False
        break
    else:
        data2 += 1

    time.sleep(1)

host = '192.168.1.51' # Symbolic name meaning all available interfaces
port = 8080 # Arbitrary non-privileged port

while True:
    server_sock = socket.socket(socket.AF_INET)

```

```

server_sock.bind(('', port))
server_sock.listen(1)

print("기다리는 중")
client_sock, addr = server_sock.accept()

print('Connected by', addr)

# 서버에서 "안드로이드에서 서버로 연결요청" 한번 받음
data = client_sock.recv(1024)
print(data.decode("utf-8"), len(data))
if data.decode("utf-8") != "":
    global check
    check = True
    t = threading.Thread(target=set1, args=(1,))
    t.start()
    t2 = threading.Thread(target=set2, args=(True,))
    t2.start()

# 연결끊겠다는 표시 보냄
# i=99
# client_sock.send(i.to_bytes(4, byteorder='little'))

client_sock.close()
server_sock.close()

```