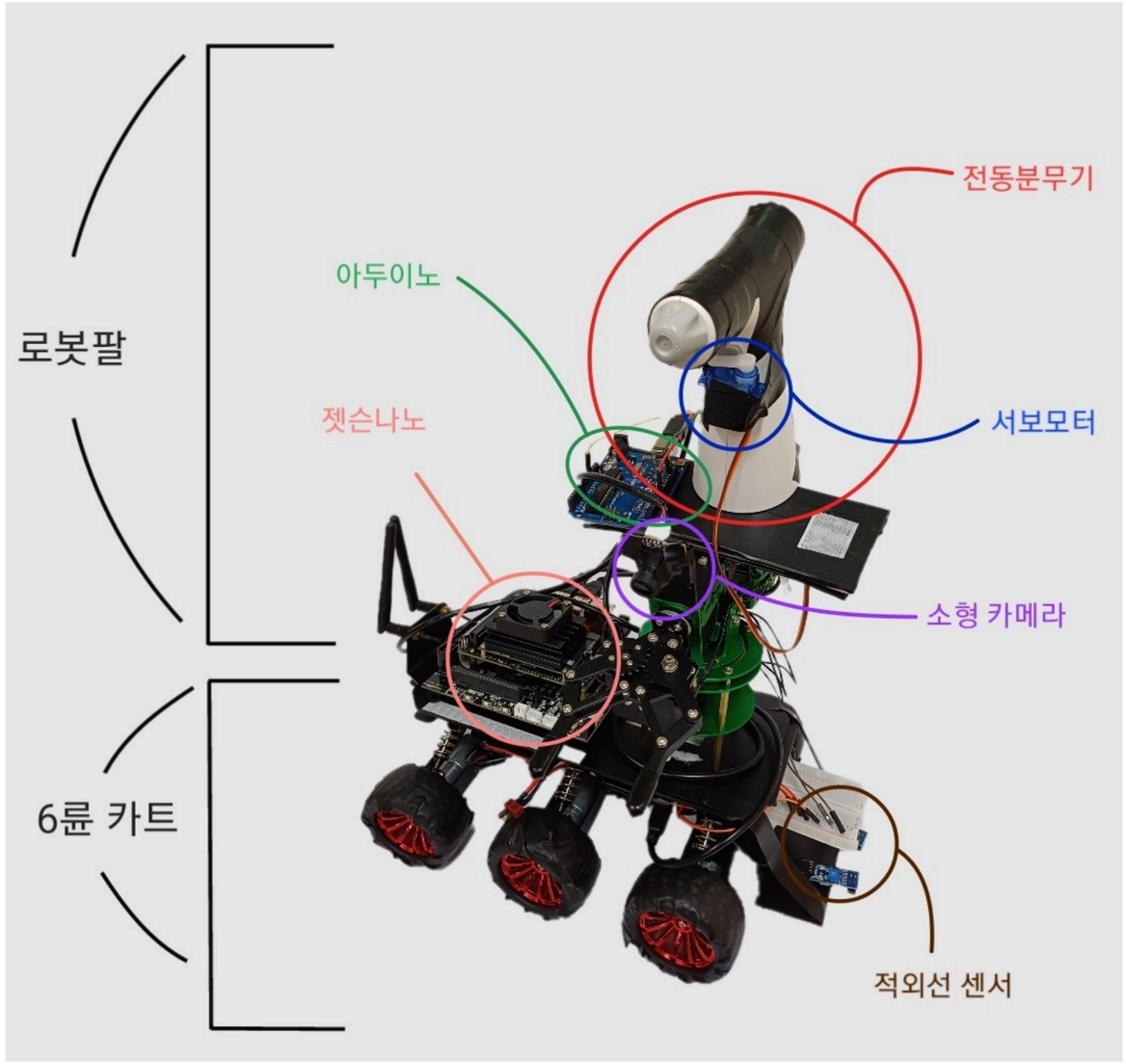
Project - 토마토 인공수분 로봇 시스템

- **토마토 인공수분(인공수정) 작업을 자동화 하는 로봇 시스템**
  a. 로봇이 경로를 주행하며 인공수분이 필요한 나무 탐색
  b. 딥러닝 모델을 통해 토마토 나무 구성요소 식별
  c. 로봇이 인공수분 필요여부를 자동으로 판단
  d. 로봇팔을 자동으로 제어하여 분무 위치 조정 후 분사
  e. a ~ d 과정을 반복

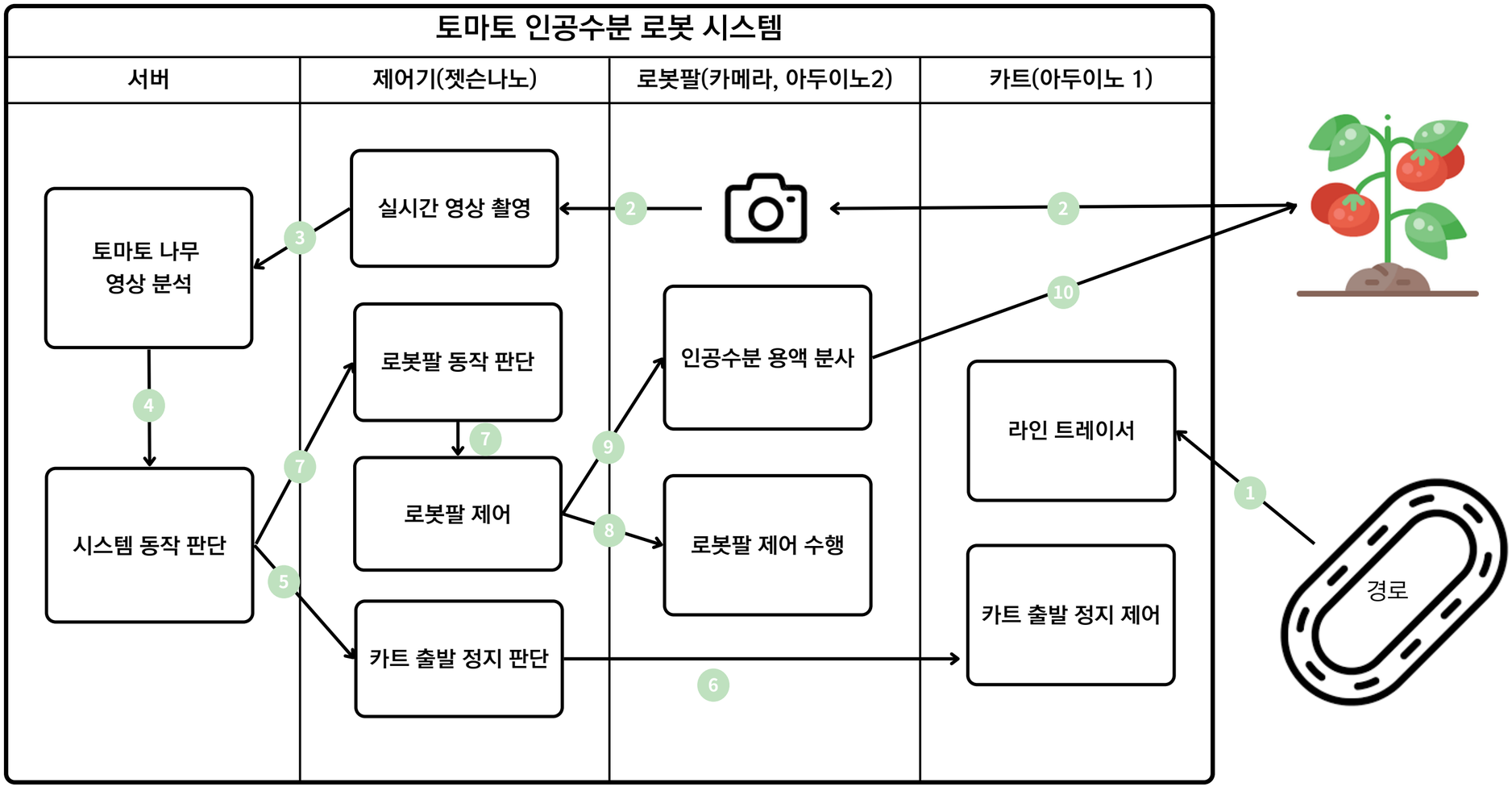
- **개발 환경**
  - ubuntu 20.04(Jetson Nano), Window
  - python 3.8, C, pytorch
  - Jupyter Notebook, Arduino IDE

- **딥러닝 모델**
  - YOLOv8s – Custom

- **하드웨어**
  - Jetson Nano, Arduino Uno, Server(개인 노트북)



Project - 시스템 아키텍처



⚠

시스템 구성을 이해를 돕기 위해 붙인 번호
시스템 흐름은 번호와 다른점이 있습니다

1. 경로에서 반사되는 빛의 차이를 통해 경로 주행
2. 로봇팔의 부착된 카메라가 실시간으로 영상을 촬영하며 토마토 나무 탐지
3. 카메라로 들어오는 영상을 젯슨 나노가 서버로 전송
4. 서버는 전송받은 영상에서 토마토 나무 구성 요소 식별 및 동작 판단
5. 토마토 나무 줄기 탐지 시 카트 정지 신호를 젯슨 나노에 전송
6. 전달 받은 제어 신호를 젯슨 나노가 카트로 전달하여 카트 주행 정지
7. 카트가 정지되면 토마토 나무 구성 요소를 식별하고 로봇팔 제어 신호 전송
8. 젯슨 나노는 로봇팔을 제어하며 카메라 위치와 인공수분 용액 분무 위치를 조정
9. 로봇팔 위치 조정이 끝나면 로봇팔 고정 후 아두이노2에 분무 신호를 보내 용액 분무
10. 전동 분무기를 작동시켜 적절한 위치에 인공 수분 용액 분사

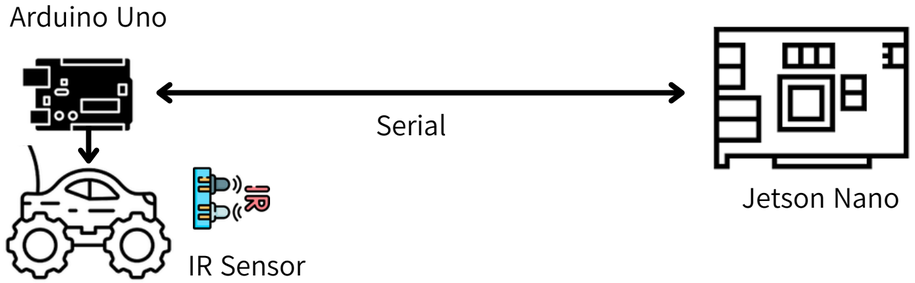
Project - 카트 주행

요구 상황

- 사람의 제어없이 스스로 토마토 재배지(스마트 팜)를 주행하며 인공수분이 필요한 나무를 찾아 인공수분 필요 여부를 판단

설계

- 경로를 라인 트레이서 기술을 활용해 로봇이 자율적으로 토마토 재배지를 주행할 수 있도록 설계
- 로봇이 토마토 나무를 인식할 경우 카트를 정지하고 인공수분 필요 여부를 판단
- 인공수분 필요 여부 판단 및 인공수분 용액 분사가 완료되면 다시 주행을 시작



핵심 구현

라인트레이서

- 카트에 아두이노와 적외선 센서 2개를 탑재
- 경로에서 반사되는 빛의 양의 차이를 통해 라인트레이서 구현

카트 정지 및 출발

- 아두이노와 젯슨 나노를 시리얼 통신을 통해 통신
- 젯슨 나노가 보내는 출발, 정지 신호에 따라 카트 출발, 정지

어려움과 해결 과정

지연 시간 문제

1. 부족한 컴퓨팅 파워로 영상 촬영과 영상 분석하는 컴퓨터를 나눠서 시스템을 구현
    a. 이로 인해 지연시간 발생
2. 지연시간이 생김으로써 토마토나무가 있음에도 계산처리를 완료하지 못해 정지 판단을 내리지 못하는 경우 발생
3. 이를 해결하기 위해 초당 30 fps로 진행하던 영상 촬영을 20 fps로 설정
4. 카트 주행 시 1초마다 정지하는 딜레이를 설정하여 정지 판단의 정확성을 개선
    - (40% -> 80%)

Project - 구성 요소 식별 모델

요구 상황

- 로봇 출발, 정지 판단을 위해 실시간으로 촬영되는 영상에서 토마토 나무를 인식
- 인공수분 판단 여부를 위해 토마토 나무 구성 요소를 탐지

설계

- 토마토 나무 구성 요소가 담긴 데이터를 Labelme를 통해 라벨링
- 라벨링이 완료되면 YOLOv8 모델을 기반으로 커스텀 학습을 진행

핵심 구현

데이터 전처리

- 원활한 모델학습을 위해 검수된 이미지들의 사이즈를 축소 후 라벨링 진행
  - 원본 -〉 0.3*원본

모델 학습

- 코랩에서 모델학습 진행 후 노트북으로 가중치 전달

이미지 크기 변환 코드 일부

```
# 이미지 크기줄이기
width, height = img.size
resized_img = img.resize((int(width*0.3), int(height*0.3)))

# 새로운 디렉토리에 이미지 저장
new_img_path = os.path.join(dst_dir, os.path.basename(img_path))
resized_img.save(new_img_path)
```

- Python의 PIL(Pillow) 라이브러리를 활용하여 데이터가 있는 폴더에 있는 모든 이미지크기를 축소

모델 가중치 저장

```
torch.save(model.state_dict(), '/content/drive/My Drive/YOLO8_tom/YOLO8_tra
```

라벨링 클래스

라벨링 클래스
1. 나무 줄기
2. 화방
3. 만개꽃

Project - 구성 요소 식별 모델

어려움과 해결 과정

버전 차이

1. 코랩에서 모델학습을 진행하고 노트북으로 가중치 파일을 옮겨 시스템을 실행하던 중 **모델에 가중치를 적용할 때 에러 발생**
2. 에러를 확인해보니 모델 레이어 층 중 출력 층에서 필터수가 다른 것을 확인
3. 원인 확인 결과 코랩과 노트북의 파이토치 및 관련 툴들의 버전들의 차이로 인해 YOLOv8 모델의 구성이 다르다는 것을 확인
4. 따라서 버전을 맞추기 위해 처음엔 노트북에 설치된 프로그램 환경 버전들을 바꾸기 시도
    - 하지만, 노트북 환경은 사양 부족으로 인해 버전들을 업그레이드 불가
5. 방법을 바꾸어 코랩에서 기존에 설치된 파이썬 버전을 낮추고 파이토치 설치 시 제 노트북과 같은 파이토치 버전으로 설정 후 설치
6. 그 후, 다시 모델 학습을 진행하고 가중치 파일을 받아 노트북에서 모델에 적용하여 사용

YOLOv8 레이어 층 코드 확인

- print(model) 코드를 사용해 모델 레이어층을 비교 진행
    - (cv3): ModuleList 층의 레이어 수가 다른것을 확인

↓

노트북 버전 변경 시도

- 노트북의 YOLOv8의 버전을 변경하려 했으나 실패
    - GTX 1060 그래픽 카드 버전의 알맞는 CUDA, 파이썬, 파이토치 버전 때문에 노트북 YOLOv8s의 버전 변경은 불가
- 따라서 코랩의 버전을 변경하기로 결정

↓

코랩 버전 변경

- 코랩에서 버전이 맞지 않는 파이 토치 관련 버전을 삭제
    - pip uninstall torch torchvision torchaudio(삭제 코드)
    - 재설치
        - torch==1.11.0+cu113
        - torchvision==0.12.0+cu113
        - torchaudio==0.11.0

- 위 과정을 통해 문제 해결

Project - 영상 송,수신 및 신호 전송

요구 상황

- 실시간 영상에서 YOLOv8이 객체를 탐지하고 제어 정보를 전송

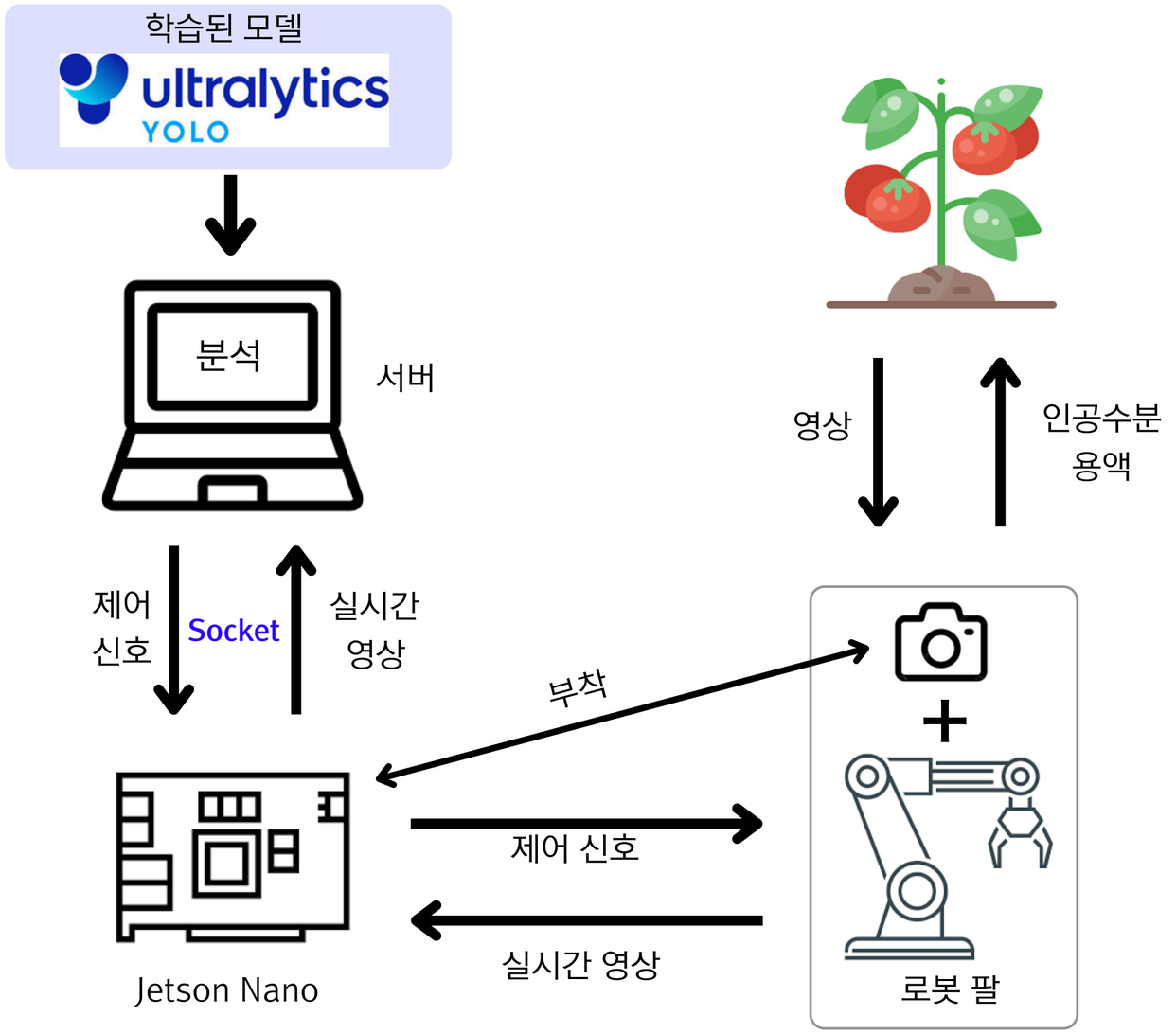
설계

- 젯슨 나노에 카메라를 연결하여 영상을 촬영
- 젯슨 나노가 촬영한 영상을 서버(노트북)로 전달
- 서버는 YOLOv8을 통해 영상을 분석
- 분석한 정보를 통해 로봇의 주행,정지, 인공수분 필요 여부 , 로봇팔 제어를 판단
- 판단 결과를 젯슨 나노로 전송

구현

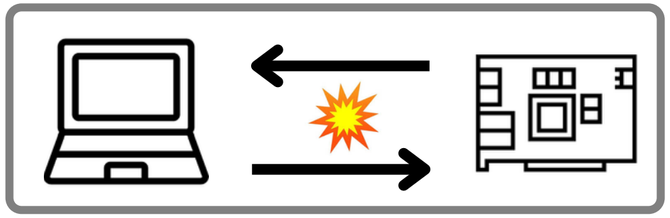
데이터 통신

- Wifi를 통해 서버와 젯슨 나노 통신
- Socket 통신을 통해 영상, 제어신호 송,수신 구현
- pickle 프레임 워크를 통해 데이터 직렬화



Project - 영상 송,수신 및 신호 전송

어려움과 해결 과정



데이터 송,수신 충돌

1. Socket을 구현하고 통신을 진행하니, **데이터 충돌이 발생하여 시스템이 멈추는 에러 발생**
    - 제어 신호를 송신하는 순간 데이터 충돌 발생
2. 문제 해결을 위해 영상을 전송하는 Socket, 제어 신호를 전송하는 Socket 2개를 설정해 시스템을 구현
3. 하지만 결국 지연시간이 누적되면서 영상과 제어의 판단이 점점 차이가 발생
    - EX) 로봇이 정확한 위치에 정지하지 못함(정지 위치 보다 더 뒤에서 정지)
4. 문제 해결을 위해 다시 소켓을 하나로 구현
5. 대신 제어 신호 전송 전, 후로 time.sleep(0.01)를 설정해 전송 간격을 조정
    - 데이터 충돌을 해결

시스템 최적화

1. 젯슨 나노가 영상 송신과 제어신호 수신, 아두이노 제어신호를 송신하는 역할을 수행하는데 컴퓨팅 파워 부족으로 인해 **지연시간과 제어신호 순서가 건너 뛰어지고 실행되는 등 시스템이 오작동 되는 경우가 발생**
2. 문제 해결을 위해 젯슨 나노에서 영상을 송신하고 제어신호를 수신하는 메인 스레드와 아두이노 제어신호를 제어하는 스레드를 설정
    - **비동시 처리 구현으로 지연시간 감소 및 데이터 충돌 방지**
3. 또한 큐를 아두이노 제어신호 순서를 저장하는 큐를 생성
    - **선입 선출 구조를 통해 제어 신호를 저장하고 나가는 순서를 고정**
    - **큐를 통해 제어신호가 건너 뛰면서 실행되는 경우를 방지**

아두이노 통신 처리 스레드 생성 및 시작

```
# 아두이노 통신 처리하는 스레드
arduino_thread = threading.Thread(target=send_to_arduino_thread)
arduino_thread.start()
```

아두이노 제어신호 순서 저장 큐 생성

```
# 명령을 저장할 큐 생성
command_queue= Queue()
```

Project - 주요 알고리즘

요구 상황

- Custom YOLOv8이 객체 탐지 후 도출하는 정보를 통해 시스템 제어

설계

- Custom한 YOLOv8s 모델을 통해 실시간 영상 분석
- YOLOv8s이 Class를 탐지할 때 나오는 Bounding box X, Y 좌표를 이용하여 객체 위치 데이터 추출
  - x1, x2, y1, y2의 중앙값 추출
  - EX) (x1+x2)/2 = x_mid

구현

인공수분 로봇 위치 조정 알고리즘

- 프레임 창 속 토마토 나무 줄기 중앙값 위치 사용

화방 정보 분석 알고리즘 구현

- 프레임 별 화방 감지 개수 및 횟수 사용

만개꽃 식별 알고리즘 구현

- 프레임 별 클래스들의 좌표 사용

용액 분사 알고리즘 구현

- 프레임 속 화방 위치 사용

구현 상세

YOLOv8 객체 탐지

1. YOLOv8은 실시간 영상에서 프레임마다 연산을 진행
   - 클래스 탐지 시 바운딩 박스 정보 추출
2. 이를 이용하기 위해 추출된 정보를 저장하는 변수 설정
   - 프레임마다 객체 확인 시 변수 값 증가
   - EX) 만개꽃 개수 저장 변수
3. 상황에 맞춰 적절한 변수값 설정
   - 시스템을 실행해보며 최적의 변수값 설정

단계별 설정

- 시스템이 시작되면 로봇은 상황에 따라 적절한 상황 대처를 해야함
  - EX) 토마토 나무 식별 후 인공수부 여부 판단
    i. 인공수분이 필요한 경우 -> 화방 정보 분석 단계로 넘어감
    ii. 인공수분이 불필요한 경우 -> 다시 로봇 주행 단계로 넘어감
- 이를 구현하기 위해 각 단계마다 일어날 경우의 수를 고려
  - 단계 별 상황에 맞는 알고리즘 구현

Project - YOLOv8 성능 개선

요구 상황

- YOLOv8의 성능의 따라 시스템 자체의 성능이 좌우되는 상황이므로 고성능의 YOLOv8 모델이 필요

설계

- 더 많은 양의 데이터 모델 학습
- 하이퍼 파라미터 값 조정

구현

데이터 수 증가

- 2000 -> 4000 -> 6000

데이터 증강 기술 활용

- 이미지 밝기, 대비 조정한 이미지 2000개 추가

조기 종료(Early Stopping)

- 과적합 방지를 위해 조기종료 기능 사용
- 손실이 개선되지 않는 에포크 수를 15로 설정(patience=10)

배치 사이즈 조정

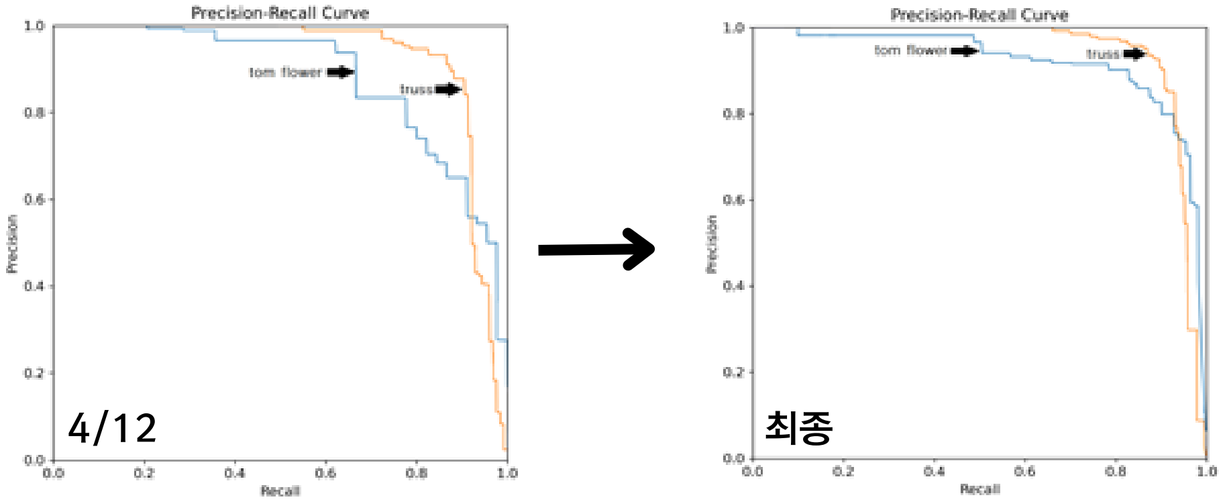
- 배치 사이즈를 8, 16, 32 조정해가며 성능 비교
- 32 일때 최고 성능

성능 변화

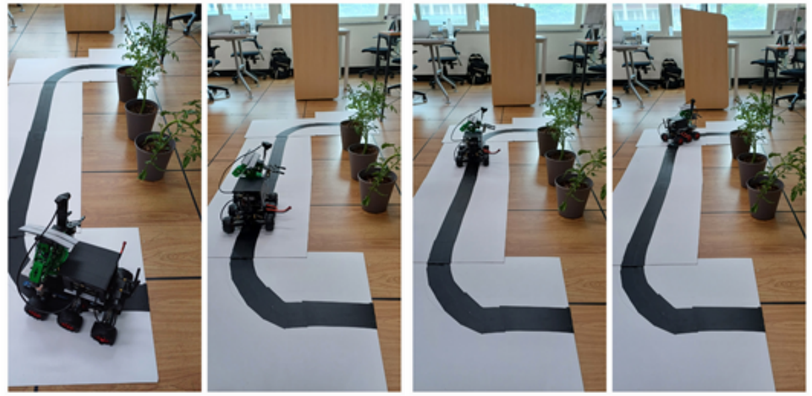
만개꽃 클래스 탐지 성능 변화

만개꽃	3/13	5/14	최종
P	0.62	0.83	0.90
R	0.53	0.65	0.83
AP50	0.71	0.82	0.92
AP50-95	0.32	0.44	0.51

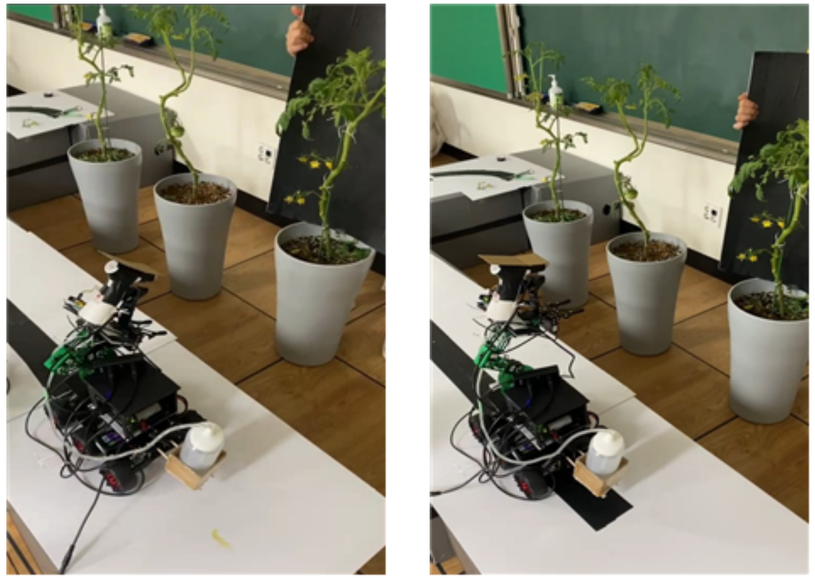
만개꽃, 화방 클래스 P-R 곡선 변화



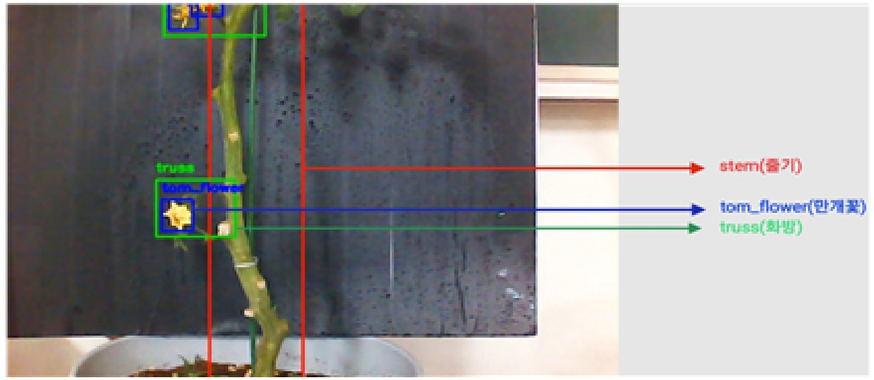
Project - 구현 모습



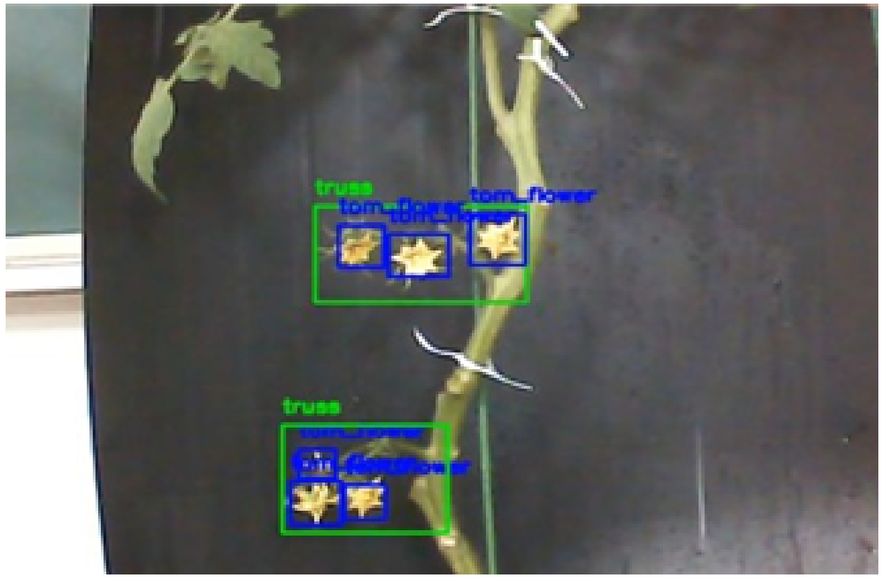
카트 주행



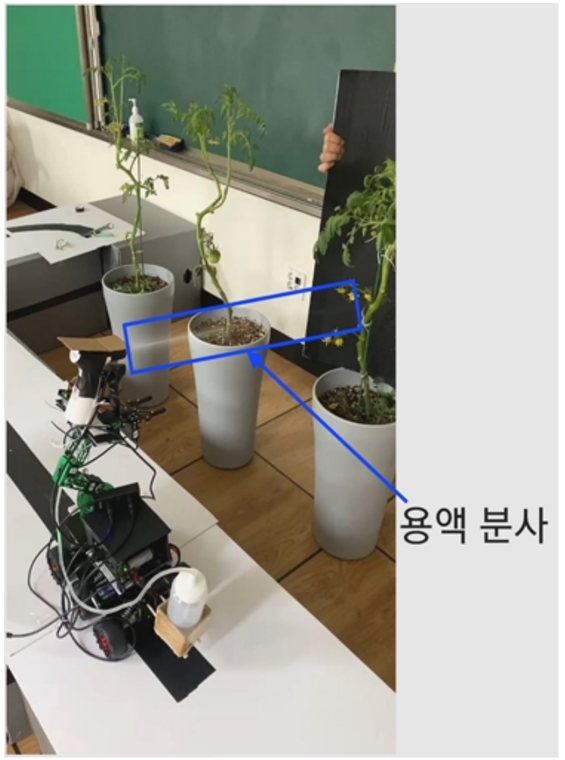
로봇팔 제어



토마토 나무 생육 상태 식별



인공수분이 필요한 화방 프레임 창 중앙 조준



용액 분사 모습