

# 에코빈 진행 보고서

## 작성 및 검토 확인란

구분	성명	년 월 일	서명
작성자	임태경	2023.05.30	

## 개정 이력

개정일자	버전	개정내용	작성자	확인자
2023.05.09	1.0	5/9 진도보고서	임태경	
2023.05.16	2.0	5/16 진도보고서	임태경	
2023.05.23	3.0	5/23 진도보고서	임태경	
2023.05.30	4.0	5/30 진도보고서	임태경	

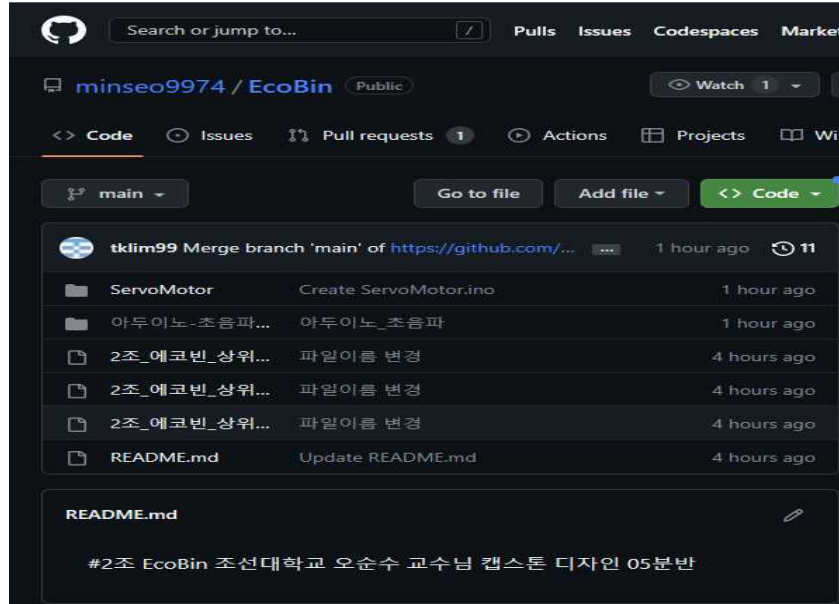
# 목 차

제 1 장. 진도보고서 .....	1
제 1 절 5월 9일 진도보고서 .....	3
제 2 절 5월 16일 진도보고서 .....	7
제 3 절 5월 23일 진도보고서 .....	12
제 4 절 5월 30일 진도보고서 .....	18

# 제 1 장. 진도보고서

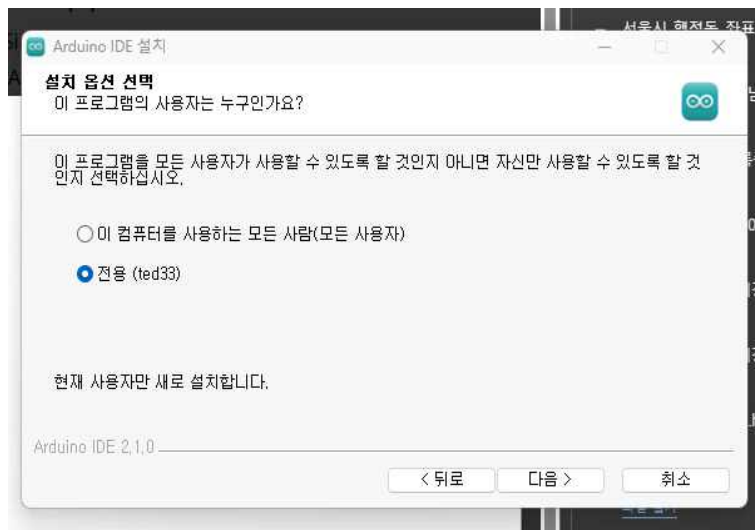
## 제 1 절 5월 9일 진도보고서

- 협업 관리 툴인 Github를 통해 프로젝트를 생성하고 팀원들과 프로젝트를 관리함.
- Github 주소 : <https://github.com/minseo9974/EcoBin>



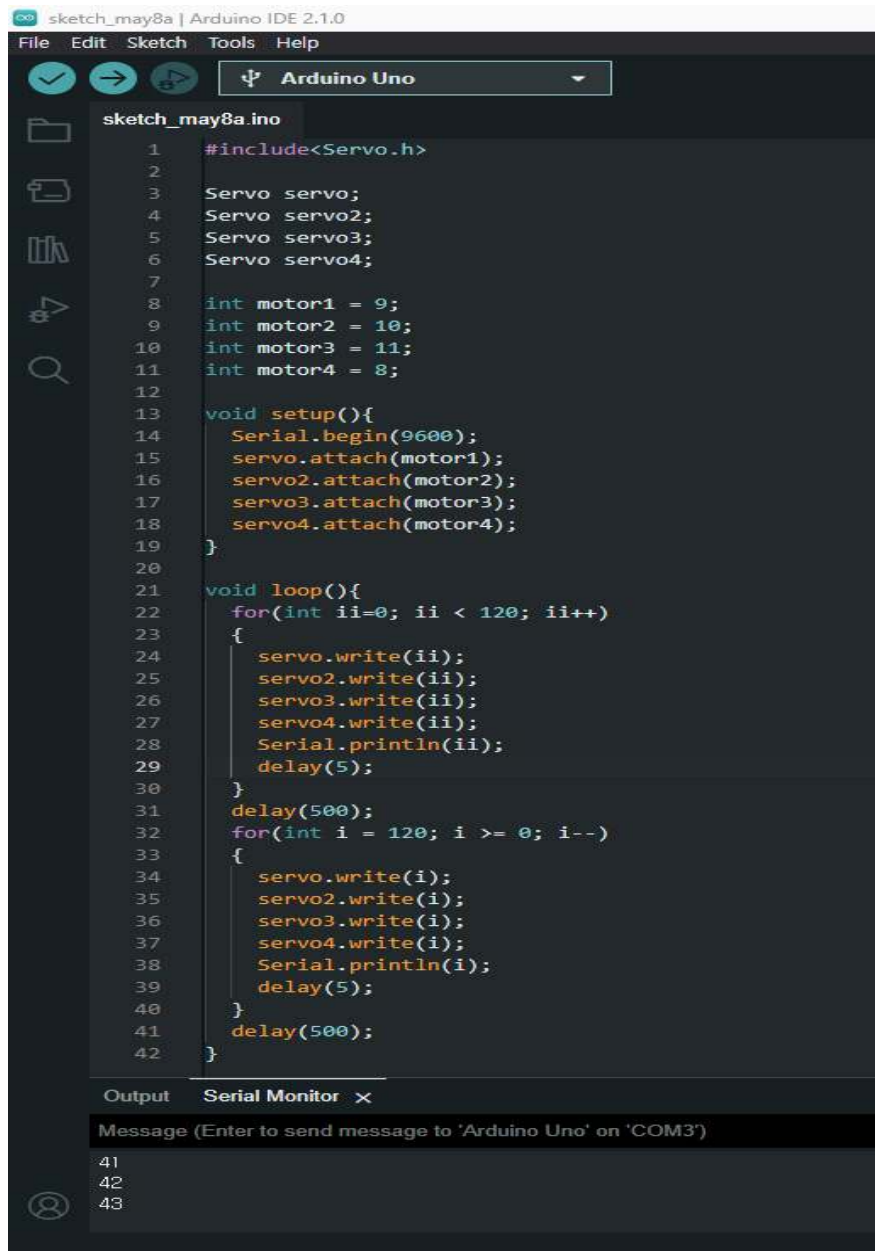
[그림 1-1] Github 프로젝트 생성

- 프로젝트 생성 후 서보모터와 스텝모터를 제어하기 위해 아두이노IDE를 설치함.



[그림 1-2] 아두이노 IDE 설치

- 분류 칸마다 2개의 서보모터가 들어가기 때문에 총 4개의 서보모터를 제어하기 위한 코드를 작성함.



```
sketch_may8a.ino
1  #include<Servo.h>
2
3  Servo servo;
4  Servo servo2;
5  Servo servo3;
6  Servo servo4;
7
8  int motor1 = 9;
9  int motor2 = 10;
10 int motor3 = 11;
11 int motor4 = 8;
12
13 void setup(){
14   Serial.begin(9600);
15   servo.attach(motor1);
16   servo2.attach(motor2);
17   servo3.attach(motor3);
18   servo4.attach(motor4);
19 }
20
21 void loop(){
22   for(int ii=0; ii < 120; ii++)
23   {
24     servo.write(ii);
25     servo2.write(ii);
26     servo3.write(ii);
27     servo4.write(ii);
28     Serial.println(ii);
29     delay(5);
30   }
31   delay(500);
32   for(int i = 120; i >= 0; i--)
33   {
34     servo.write(i);
35     servo2.write(i);
36     servo3.write(i);
37     servo4.write(i);
38     Serial.println(i);
39     delay(5);
40   }
41   delay(500);
42 }
43
```

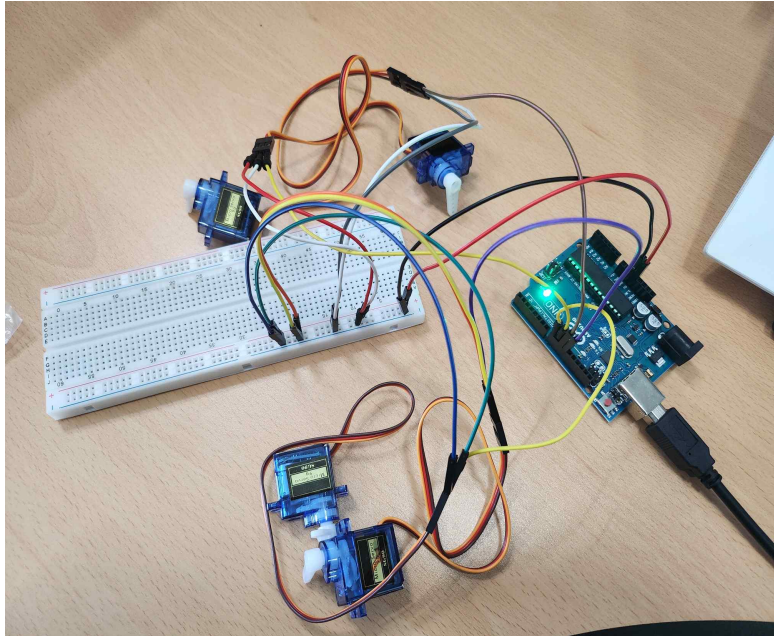
Output Serial Monitor X

Message (Enter to send message to 'Arduino Uno' on 'COM3')

41  
42  
43

[그림 1-3] 서보모터 장비 점검 코드

- 코드를 실행하였더니 4개의 서보모터 모두 이상이 없음을 확인함.



[그림 1-4] 서보모터 장비 점검 사진

[표 1] 에코빈 물품조사

물품	가격 (원)	결제방식	수량
ESP32 IOT 사물인터넷 WiFi + 블루투스 듀얼모드 아두이노 보드 모듈	5,980	선불 결제 (네이버 페이)	1
아두이노 ESP32 DevKitCV4 38p 블루투스 개발 보드	10,980(배송비 포함)		1
아두이노 SG-90 SG90서보모터	5,280		4
아두이노 스테핑 모터 키트	57,400(배송비 포함)		1
ESP32 WiFi + Bluetooth 일체형 개발보드	11,000	후불 결제 (에듀이노)	1
우노 R3 DIP 호환보드	23,000		2
초음파센서 HC-SR04	4,400		4
라즈베리파이 카메라모듈 V2	43,500		1
라즈베리파이 NOIR 적외선 카메라 모듈	35,900		1

- 현재 틀 제작을 위한 재료 제외, 기능 물품을 구매 완료 후 승인 대기 중임.
- 선불 결제 : 79,640원, 후불 결제 : 117,800원, 총 197,440원을 사용함.

상태	작성일	결제방법	신청금액	작업
교수승인중	2023-05-02	현금영수증(사업자용)	79,640원	<a href="#">보기</a> <a href="#">취소</a>
교수승인중	2023-04-28	전자(세금)계산서(청구/영수)	117,800원	<a href="#">보기</a> <a href="#">취소</a>

[그림 1-5] 제작비 지급 신청 현황

- 에듀이노는 코딩 교육을 위한 교구, 전자부품을 전문적으로 취급하는 코딩 교구 전문 쇼핑몰임.
- 후불 결제 절차가 자세히 안내되어 있어 쉽게 이용할 수 있으며, 다른 학교에서도 많이 사용하는 쇼핑몰임.



[그림 1-6] 에듀이노 후불결제

## 제 2 절 5월 16일 진도보고서

- 캔과 플라스틱을 분류하기 위한 데이터를 수집했음.
- 국내 ai허브 사이트와 해외 로보플로우 사이트로 두 가지 선택지가 있음.
- 로보플로우는 샘플링이 되어있었으나 데이터를 열어보니 쓰레기데이터가 많았음.
- 반면 ai허브는 샘플링은 되어있지 않았으나 데이터의 품질이 좋음.

**생활폐기물 데이터 활용·환류**

분류: 재난안전환경 | 용량: 이미지  
 갱신년월: 2023-05 | 구축년도: 2022 | 조희수: 419 | 다운로드: 23 | 용량: 470.61 GB

다운로드 | 샘플 데이터

관심데이터 등록 2

※ 내국인만 데이터 신청이 가능합니다.  
 ※ 23년 신규 개방되는 데이터로, 데이터 활용성 검토, 이용자 관점의 개선의견 수렴(~10월 예정) 등을 통해 수정/보완될 수 있으며 최종데이터, 샘플데이터, 산출물 등은 변경될 수 있습니다.

**데이터 개요**

**데이터 변경이력**

버전	일자	변경내용	비고
1.0	2023-05-04	데이터 개방(Beta Version)	

**roboflow** Projects Universe Documentation Forum Sign In Create Account

Search 200,000+ Open Source Computer Vision Projects ...

Roboflow Universe > recycle > recycle

☆ **recycle Computer Vision Project** Download this Dataset Try Pre-Trained Model

**TRY THIS MODEL**  
Drop an Image or browse your device

**SOURCE**  
recycle »

**LAST UPDATED**  
7 days ago

**PROJECT TYPE**  
Object Detection

**SUBJECT**  
trash

**CLASSES**  
Can, 0, 1, 2, CAN », Can », METAL », Metal », can », distorted

**VIEWS:** 28

**DOWNLOADS:** 5

**LICENSE**  
CC BY 4.0 »

Object Detection Model

**recycle**  
Object Detection

Overview  
 Images 801  
 Dataset 0  
 Model  
 API Docs  
 Health Check

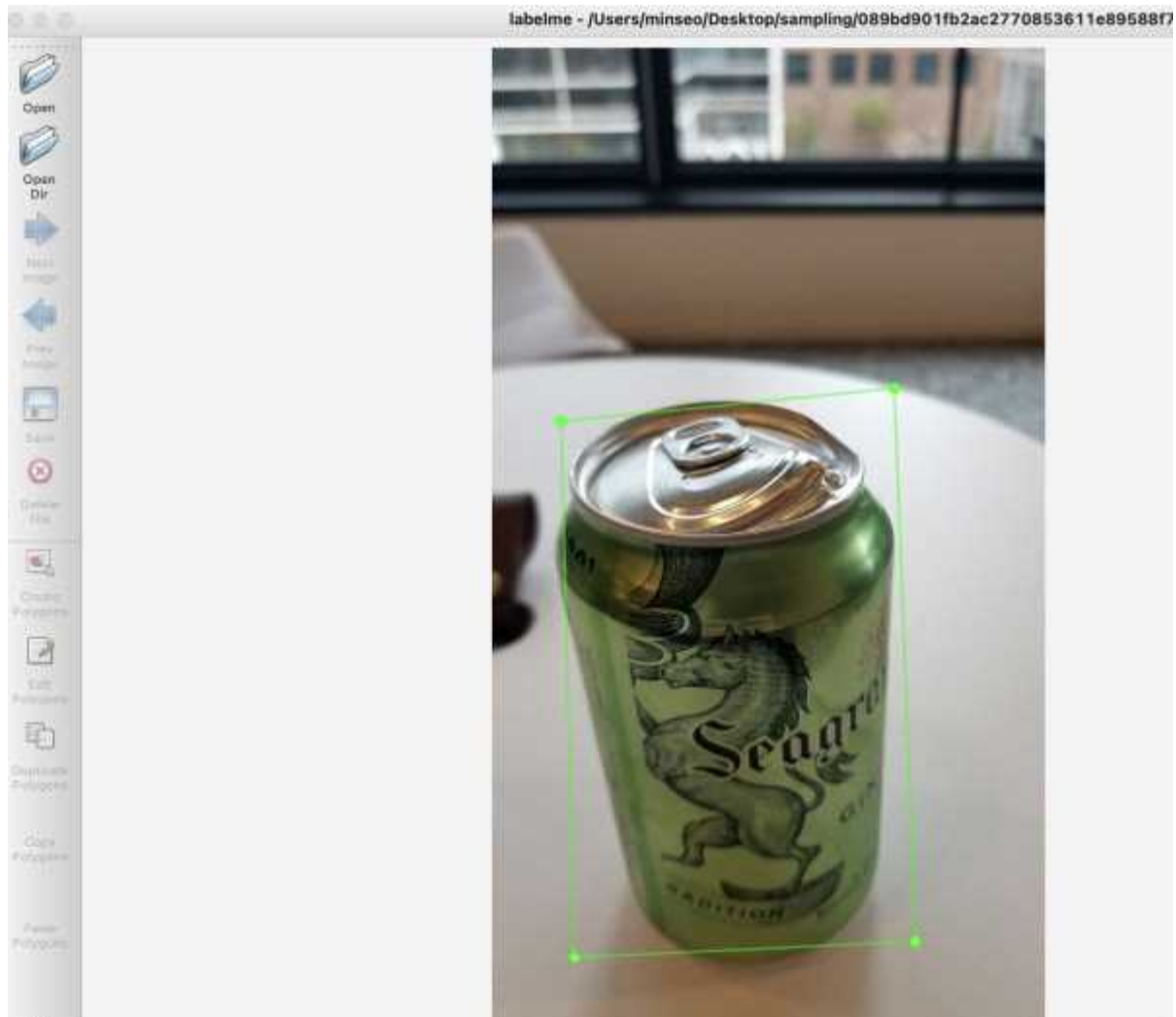
IMAGES  
 801 Images Explore Dataset »

Includes images from ...  

```
@misc( roboflow-can-data_dataset,
title = { Roboflow can data Dataset },
type = { Open Source Dataset },
author = { ecoce },
```

[그림 2-1] 위 : ai허브, 아래 : 로보플로우

- ai허브에서 가져온 데이터는 샘플링이 되어있지 않음.
- 이민서, 이지훈 학생과 함께 각각 700개의 사진을 샘플링 하기로 진행하였음.



[그림 2-2] 캔이미지 샘플링 캡처화면



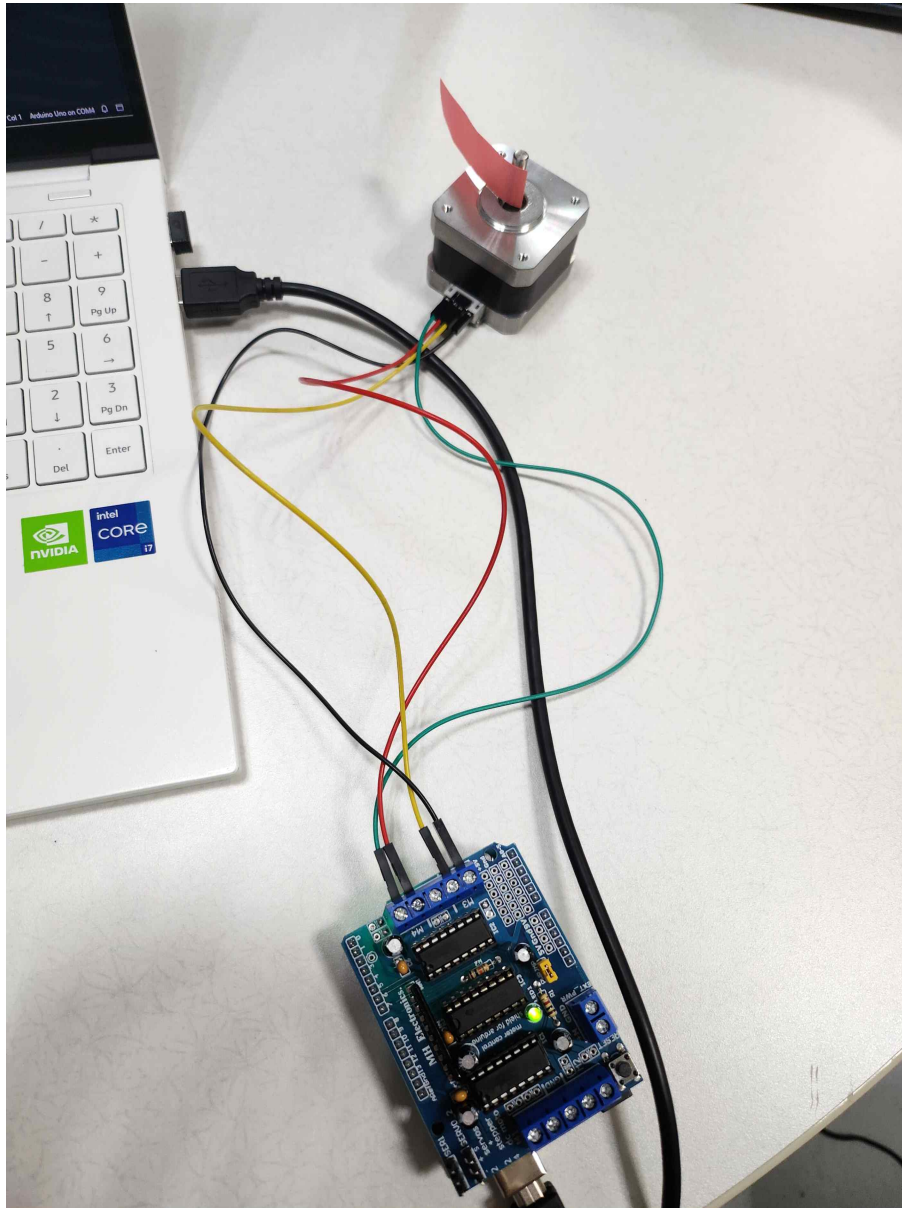
- 기존에 있던 서보모터 코드에서 대기시간 증가, 분류 칸별 서보모터 2개씩 한 묶음 작동 코드 수정 및 추가함.



```
1  #include<Servo.h>
2
3  Servo servo;
4  Servo servo2;
5  Servo servo3;
6  Servo servo4;
7
8  int motor1 = 9;
9  int motor2 = 10;
10 int motor3 = 11;
11 int motor4 = 8;
12
13 void setup(){
14   Serial.begin(9600);
15   servo.attach(motor1);
16   servo2.attach(motor2);
17   servo3.attach(motor3);
18   servo4.attach(motor4);
19 }
20
21 void loop(){
22   for(int ii=0; ii < 120; ii++)
23   {
24     servo.write(ii);
25     servo2.write(ii);
26     Serial.println(ii);
27     delay(5);
28   }
29   delay(3000);
30   for(int i = 120; i >= 0; i--)
31   {
32     servo3.write(i);
33     servo4.write(i);
34     Serial.println(i);
35     delay(5);
36   }
37   delay(3000);
38 }
39 }
```

[그림 2-3] 서보모터 수정된 코드

- 쓰레기가 해당 분류 칸까지 이동을 담당할 스테핑 모터를 구현하였음.
- 차후 상판의 틀에 부착하여 서보모터와의 연결과 현재 부착되어 있는 포스트잇 대신 이동을 도와줄 막대(판)을 설치할 예정임.



[그림 2-4] 스테핑 모터 장비 구현 사진

- 스텝핑 모터의 동작 제어 및 점검을 위한 코드를 작성함.
- 차후 스텝핑 모터의 동작 제어 시간과 횟수에 대한 코드를 수정할 예정임.

The screenshot shows the Arduino IDE 2.1.0 interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu bar, there are icons for checking, running, and uploading code, along with a dropdown menu for the board type, currently set to 'Arduino Uno'. The main workspace displays the 'SteppingMotor.ino' file with the following code:

```

1  #include <AFMotor.h>
2
3  AF_Stepper motor(20,2);
4  void setup() {
5      Serial.begin(9600);
6      Serial.println("Stepper test");
7
8      motor.setSpeed(200);
9  }
10
11 void loop() {
12     Serial.println("Single coil steps");
13     motor.step(200, FORWARD, SINGLE);
14     motor.step(200, BACKWARD, SINGLE);
15
16     Serial.println("Double coil steps");
17     motor.step(200, FORWARD, DOUBLE);
18     motor.step(200, BACKWARD, DOUBLE);
19
20     Serial.println("Interleave coil steps");
21     motor.step(200, FORWARD, INTERLEAVE);
22     motor.step(200, BACKWARD, INTERLEAVE);
23
24     Serial.println("Microstep steps");
25     motor.step(200, FORWARD, MICROSTEP);
26     motor.step(200, BACKWARD, MICROSTEP);
27
28 }
29

```

At the bottom of the IDE, the 'Serial Monitor' window is open, showing the output of the code. The output consists of several lines of text, each preceded by a 'Message' label, indicating the sequence of steps being performed by the motor.

```

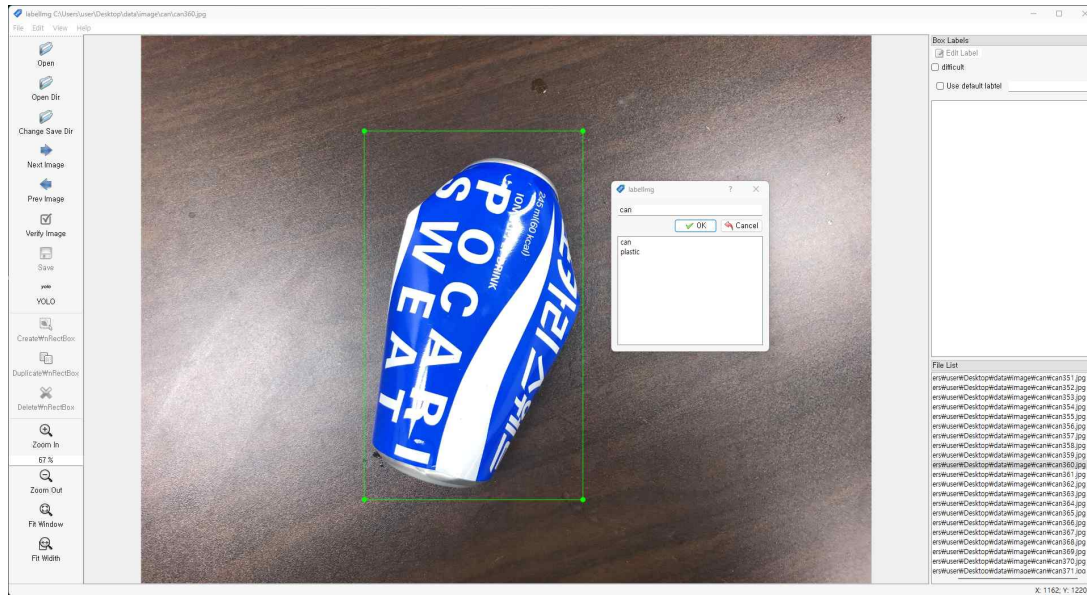
Serial Monitor x
Message (Enter to send message to 'Arduino Uno' on 'COM4')
Sinil steps
Stepper test
Single coil steps
Double coil steps
Interleave coil steps
Microstep steps

```

[그림 2-5] 스텝핑 모터 제어 코드

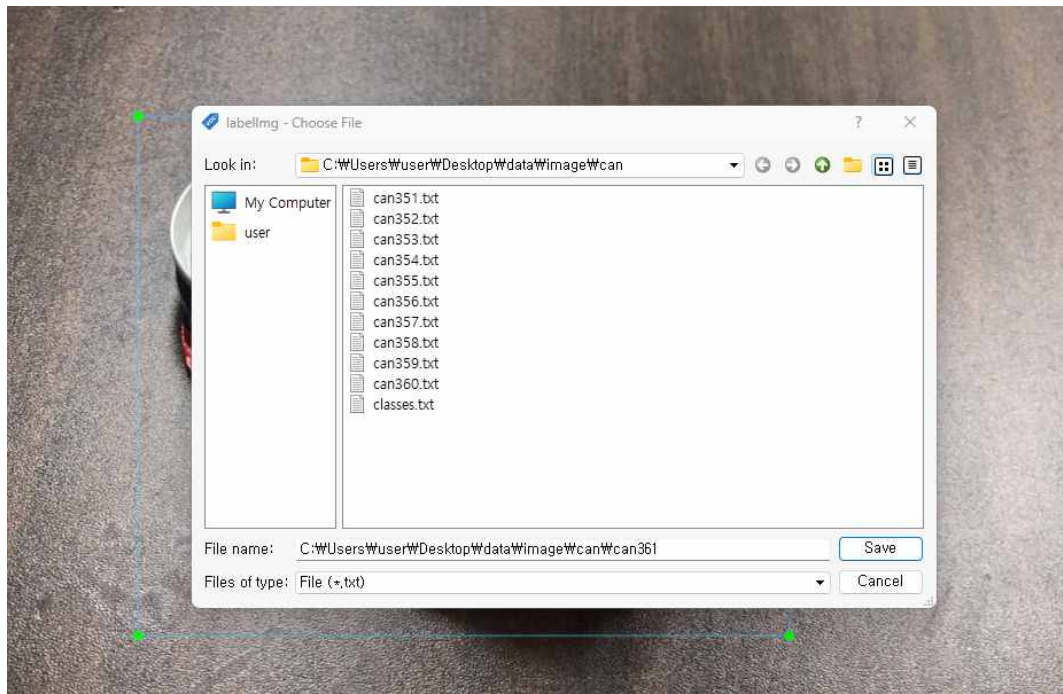
### 제 3 절 5월 23일 진도보고서

- 2주 차에 진행했던 데이터 수집을 완료한 뒤 ai허브에서 받아온 이미지 데이터는 샘플링이 되어있지 않기 때문에 각각의 이미지 데이터를 'labellmg'라는 데이터 라벨링 프로그램을 사용하여 라벨링을 진행함.



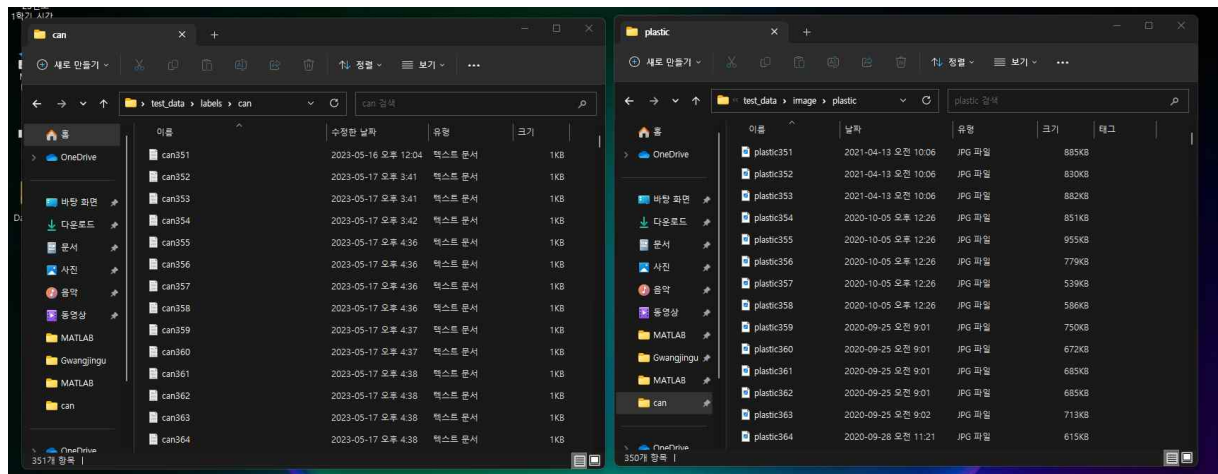
[그림 3-1] 캔 라벨링을 진행 중인 사진

- 조에서 인당 테스트 데이터 캔,플라스틱 각각 350개, 검증데이터 50개씩 라벨링을 진행함.



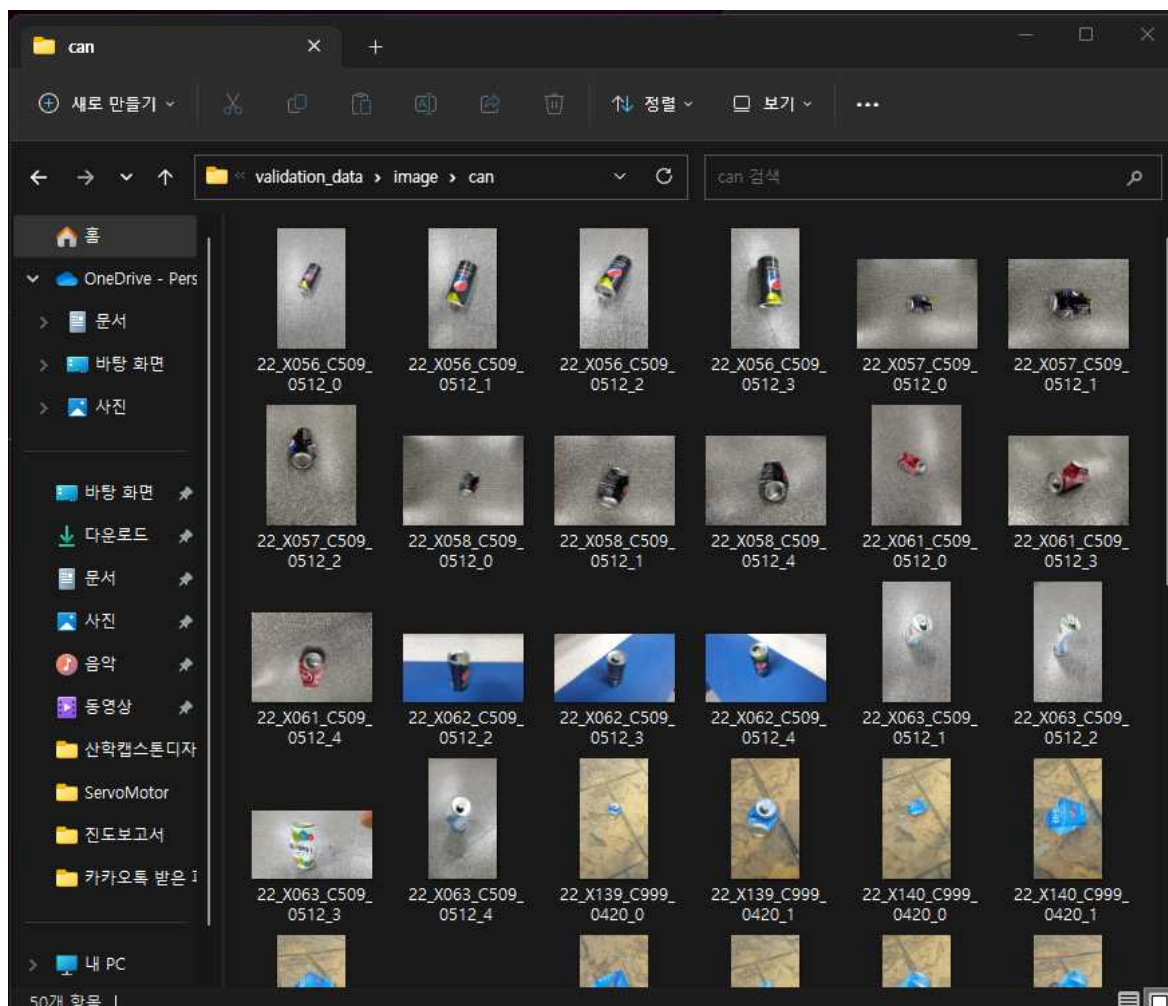
[그림 3-2] 이미지 데이터를 라벨링 후 텍스트 파일로 저장





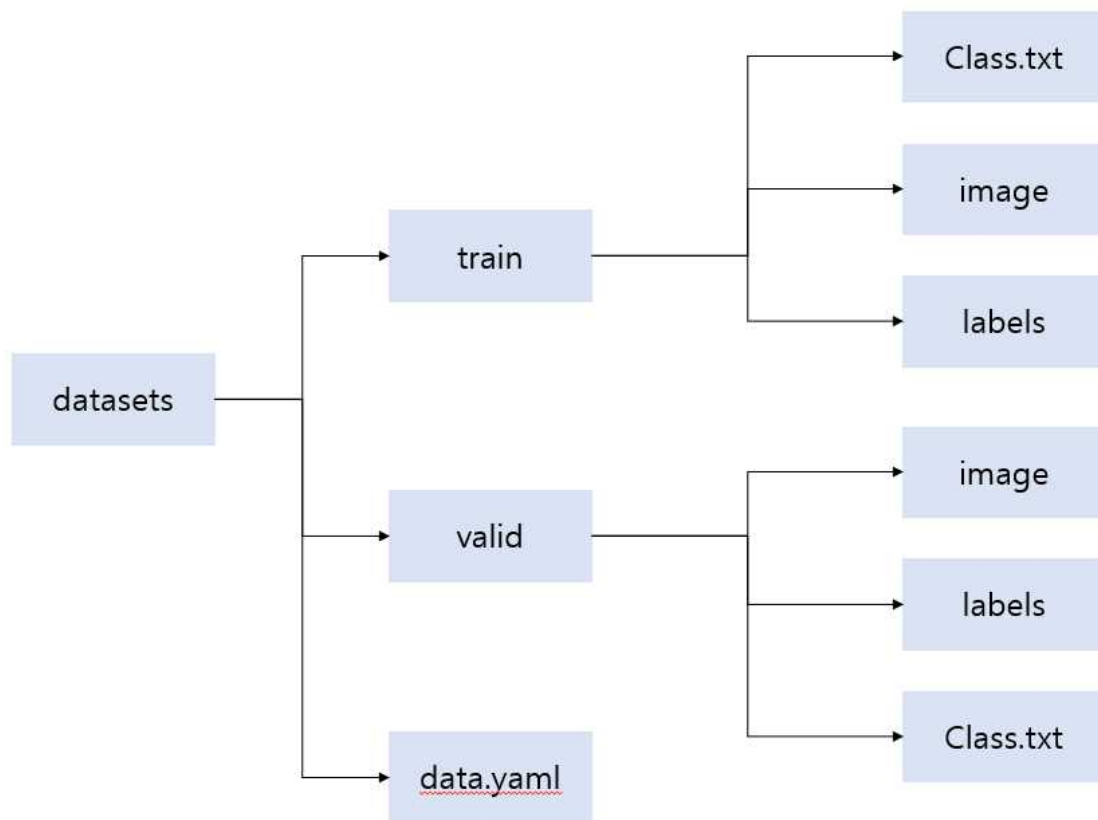
[그림 3-3] 테스트 데이터와 검증데이터 구분

- 카메라로 인식할 때 다양한 각도, 다양한 모양의 모델을 분류해야 하므로 일방적인 데이터뿐만 아니라 다양한 데이터 수집을 완료함.



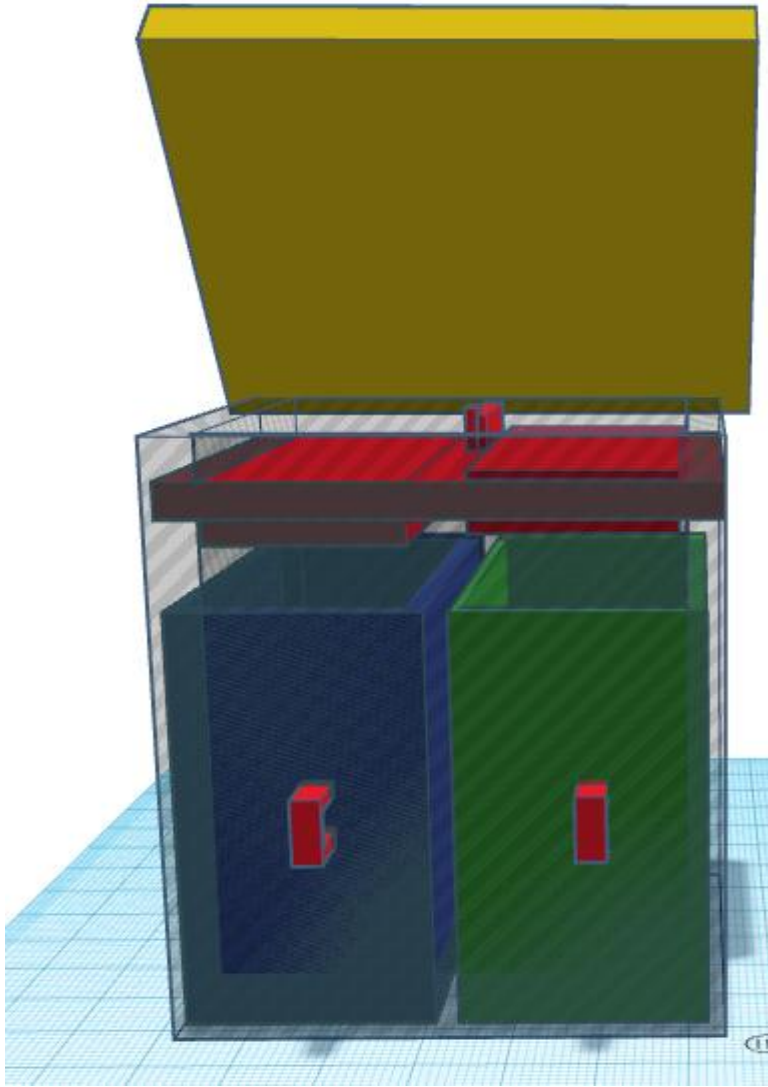
[그림 3-4] 다양한 캔 모양의 이미지 데이터

- 캔, 플라스틱의 전체적인 데이터 세트 구축에 성공함.

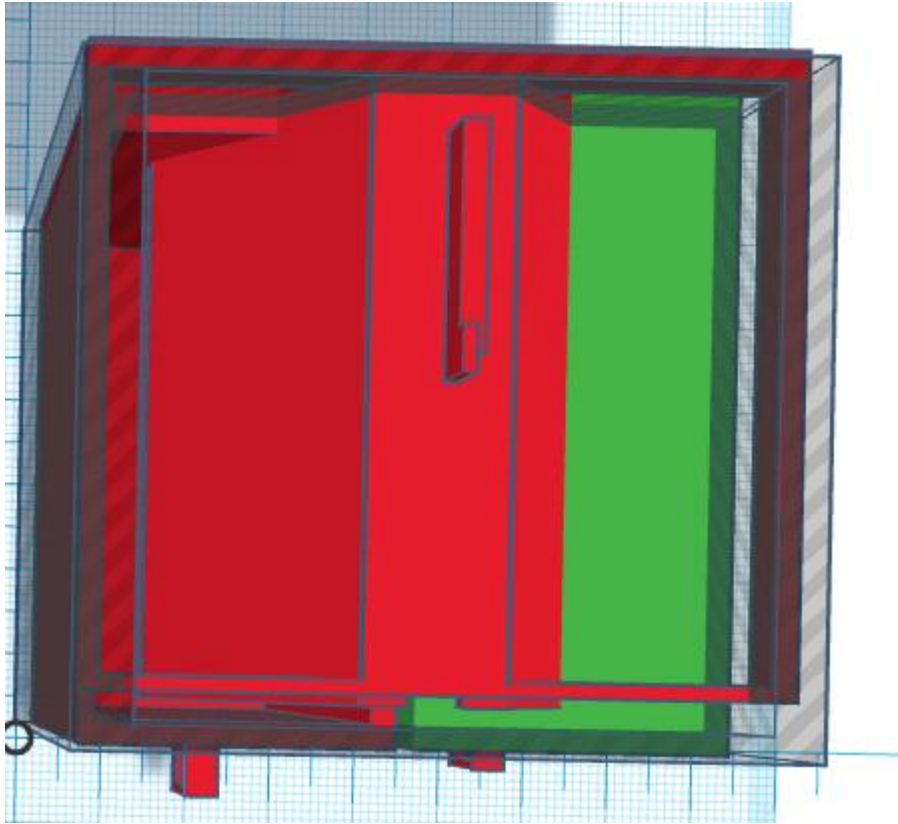


[그림 3-5] 수집한 데이터를 도식화

- ‘틴커 캐드’ 프로그램을 사용하여 전체적인 쓰레기통 구상도를 제작함.
- 쓰레기통 외부 규격, 내부 규격, 상판, 하판 규격 등 전체 틀 규격을 측정함.



[그림 3-6] 정면에서 바라본 쓰레기통



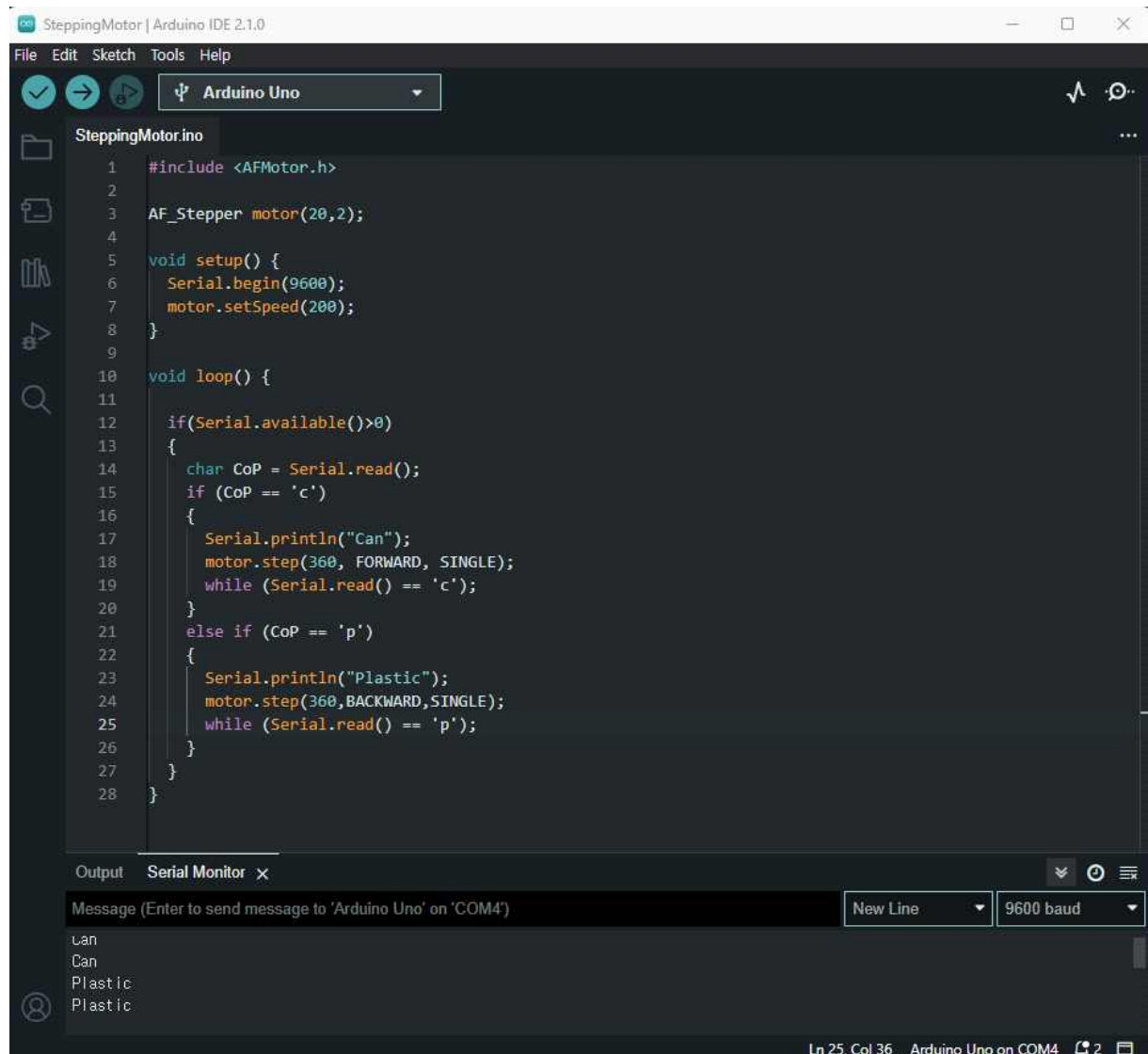
[그림 3-7] 수직으로 바라본 쓰레기통

[표 3-1] 쓰레기통 예상 규격 표

부품 \ cm	가로	세로	높이(두께)
외부 쓰레기통	40	40	60
내부 쓰레기통(2개)	16	40	40
상판 개폐 틀	18	40	3
물체 이동판	3	18	5
상판 중심부	8	40	3



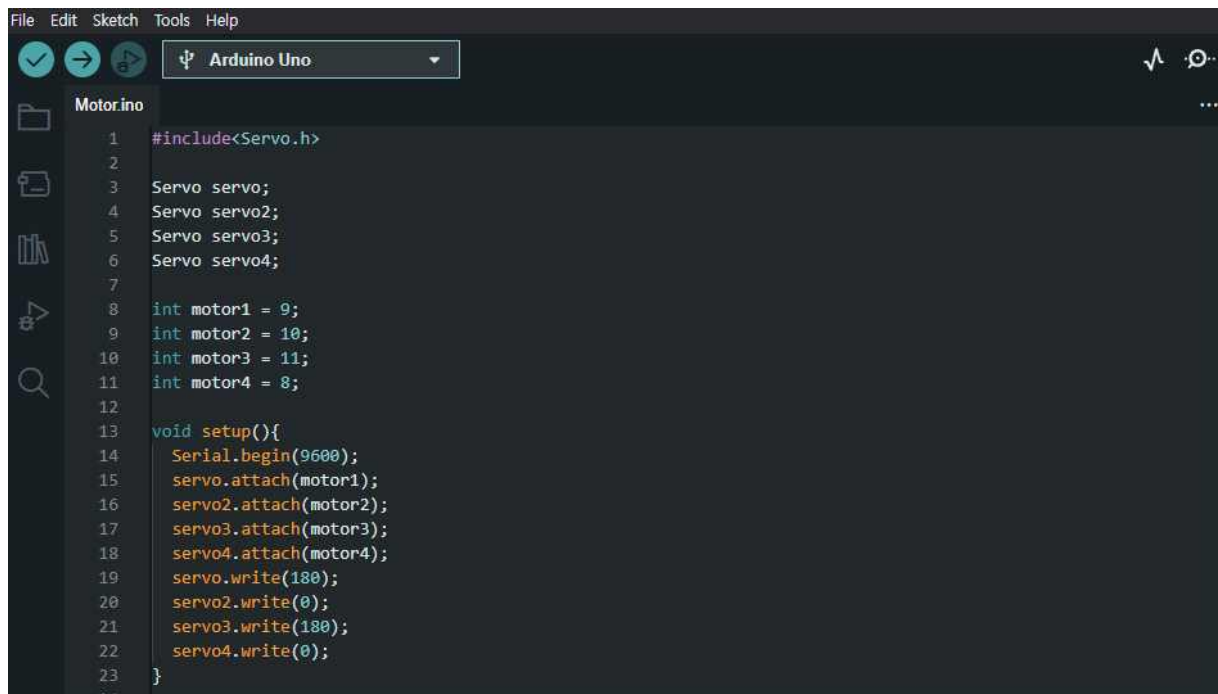
- 기존의 스텝핑 모터 코드에서 테스트 코드와 불필요한 코드를 정리 후 작성함.
- 가상의 결괏값 c와 p를 설정하여 해당 결과에 대해 특정 동작이 수행하는지 파악함.



[그림 3-8] 결괏값에 대해 스텝핑 모터가 특정 동작하는 사진

## 제 4 절 5월 30일 진도보고서

- [그림 2-3]의 기존 코드에서는 정확한 각도 조절과 모터별 개폐 각도가 불안정했지만 servo.write() 코드를 통해서 시작 각도 설정을 진행함.
- 카메라 인식 결과에 따른 결과값을 can 일 때 0, platsit일 때 1로 설정하여 해당 개폐 칸에서 서보모터가 2개씩 작동하는 코드를 완성함.



```
File Edit Sketch Tools Help
Motor.ino
1 #include<Servo.h>
2
3 Servo servo;
4 Servo servo2;
5 Servo servo3;
6 Servo servo4;
7
8 int motor1 = 9;
9 int motor2 = 10;
10 int motor3 = 11;
11 int motor4 = 8;
12
13 void setup(){
14   Serial.begin(9600);
15   servo.attach(motor1);
16   servo2.attach(motor2);
17   servo3.attach(motor3);
18   servo4.attach(motor4);
19   servo.write(180);
20   servo2.write(0);
21   servo3.write(180);
22   servo4.write(0);
23 }
24
```

[그림 4-1] 수정된 서보모터 코드 1



```
25 void loop(){
26   if(Serial.available() > 0)
27   {
28     int result = Serial.parseInt();
29     // 결과값이 0이면 can의 개폐칸 작동
30     if(result==0){
31       Serial.println("can");
32       for(int ii=0; ii <= 90; ii++)
33       {
34         servo.write(180 - ii);
35         servo2.write(ii);
36         delay(5);
37       }
38       delay(3000);
39       for(int ii=90; ii <= 180; ii++)
40       {
41         servo.write(ii);
42         servo2.write(180 - ii);
43         delay(5);
44       }
45       while (Serial.read() == 0);
46     }
47   }
48 }
```

[그림 4-2] 수정된 서보모터 코드 2

```

47 // 결과가 1이면 plastic의 개폐간 작동
48 else if(result==1){
49     Serial.println("plastic");
50     for(int ii=0; ii <= 90; ii++)
51     {
52         servo3.write(180 - ii);
53         servo4.write(ii);
54         delay(5);
55     }
56     delay(3000);
57     for(int ii=90; ii <= 180; ii++)
58     {
59         servo3.write(ii);
60         servo4.write(180 - ii);
61         delay(5);
62     }
63     while (Serial.read() == 1);
64 }
65 }
66 }

```

[그림 4-3] 수정된 서보모터 코드 3

- [그림3-8]의 스텝핑 모터 코드와 완성된 서보모터를 하나의 모터 코드로 제작함.
- 라즈베리파이의 영상인식 결과값이 byte형식으로 보내지기 때문에 지정된 버퍼 크기만큼의 공간을 설정하여 문자로 받는 buffer 코드를 추가함.
- buffer가 'a'일 때 캔으로 인식하고 'b'일 때 플라스틱으로 결과값을 인식함.
- 스텝핑모터 회로와 결합하여 작동할 때 서보모터의 개폐시간과 스텝핑모터의 작동시간의 비교가 필요하므로 delay 코드는 수정될 가능성이 있음.

```

1  #include<Servo.h>
2  #include <AFMotor.h>
3
4  Servo servo;
5  Servo servo2;
6  Servo servo3;
7  Servo servo4;
8
9  AF_Stepper motor(20,2);
10
11 int motor1 = 9;
12 int motor2 = 10;
13 int motor3 = 11;
14 int motor4 = 8;
15
16 const int buffersize = 64;
17 char buffer[buffersize];
18 int bytesRead = 0;
19
20 void setup(){
21   Serial.begin(9600);
22   motor.setSpeed(200);
23   servo.attach(motor1);
24   servo2.attach(motor2);
25   servo3.attach(motor3);
26   servo4.attach(motor4);
27   servo.write(180);
28   servo2.write(0);
29   servo3.write(180);
30   servo4.write(0);
31 }
32
33 void loop(){
34
35   if (Serial.available()){
36     bytesRead = Serial.readBytesUntil('\n', buffer, buffersize);
37     buffer[bytesRead]= '\0';
38   }
39

```

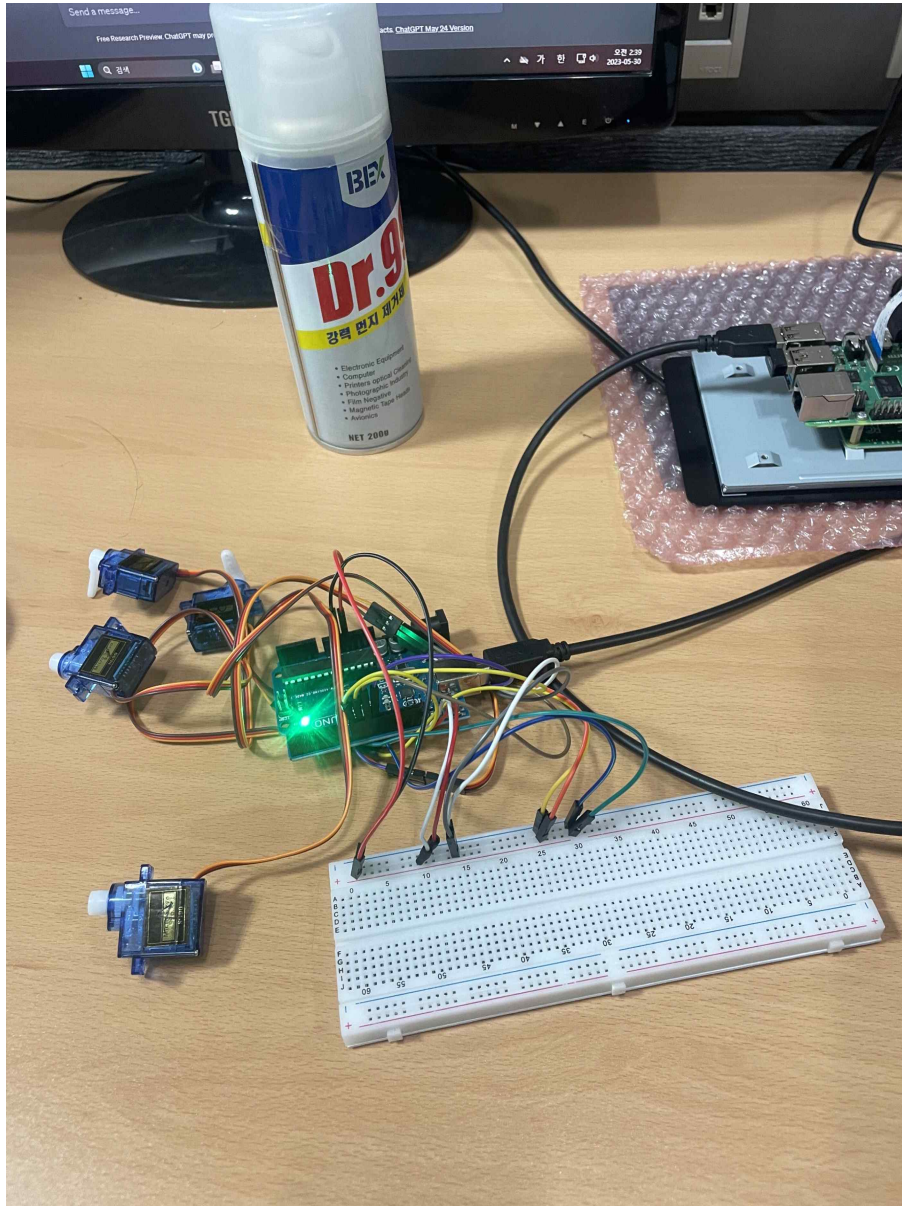
[그림 4-4] 스텝핑모터, 서보모터 코드들을 종합한 코드

```
40 if(Serial.available() > 0)
41 {
42     // 결과값이 0이면 can의 개폐칸 작동
43     if(buffer[0]=='a'){
44         Serial.println("can");
45         for(int ii=0; ii <= 90; ii++)
46         {
47             servo.write(180 - ii);
48             servo2.write(ii);
49             delay(5);
50         }
51         delay(3000);
52         motor.step(360, FORWARD, SINGLE);
53         for(int ii=90; ii <= 180; ii++)
54         {
55             servo.write(ii);
56             servo2.write(180 - ii);
57             delay(5);
58         }
59         while (Serial.read() == 'a');
60     }
61     // 결과값이 10이면 plastic의 개폐칸 작동
62     else if(buffer[0]=='b'){
63         Serial.println("plastic");
64         for(int ii=0; ii <= 90; ii++)
65         {
66             servo3.write(180 - ii);
67             servo4.write(ii);
68             delay(5);
69         }
70         delay(3000);
71         motor.step(360, BACKWARD, SINGLE);
72         for(int ii=90; ii <= 180; ii++)
73         {
74             servo3.write(ii);
75             servo4.write(180 - ii);
76             delay(5);
77         }
78         while (Serial.read() == 'b');
79     }
80 }
81 }
```

Output Serial Monitor

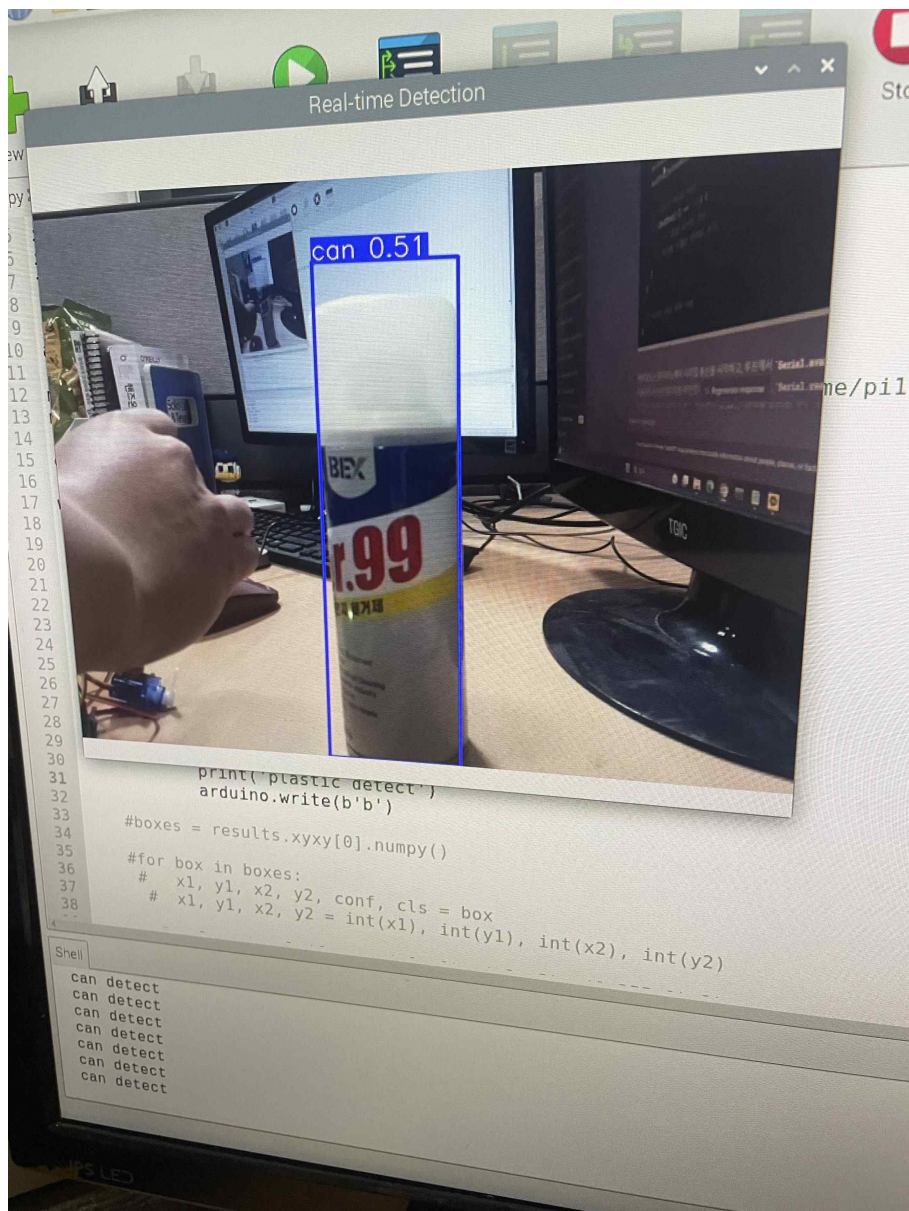
[그림 4-5] 모터 코드뿐만 아니라 buffer코드를 추가함으로써 라즈베리파이와 연동

- 라즈베리파이에서 카메라를 통해 캔과 플라스틱을 판별한 후 아두이노와 직접 연결을 통해 결과값을 전달하여 실행함.
- 캔, 플라스틱 결과값에 따라 해당 개폐 기능을 담당하는 서보모터가 작동함.



[그림 4-6] 라즈베리파이와 연결한 아두이노





[그림 4-7] 카메라를 통해 인식한 캔(고철)