

# 2024

## 종합설계 Capstone design [11반]

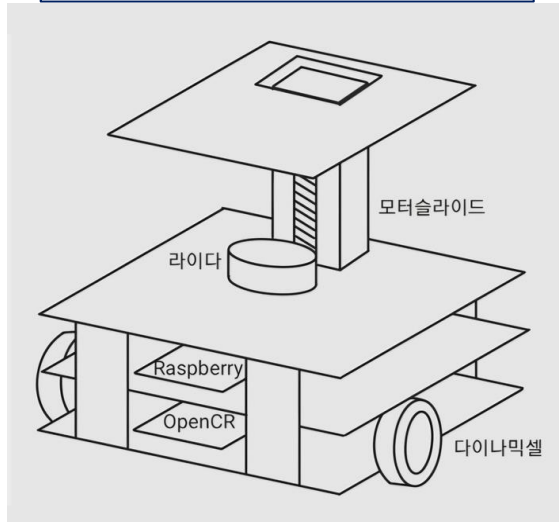
### 스마트 패키징 시스템

조원 이름 :유환성, 조수완, 현승헌

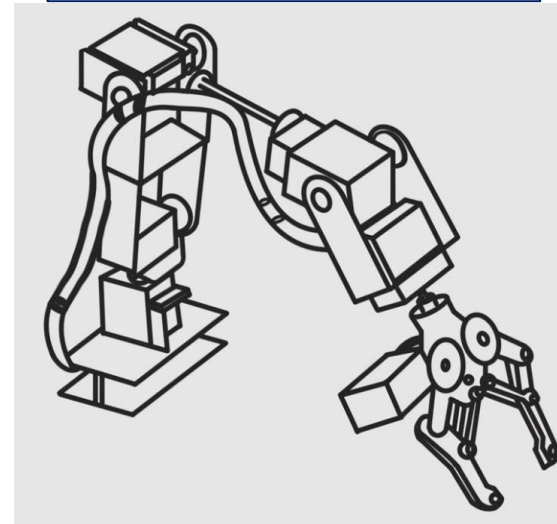
# 여러분 수행한 과제를 소개 해 주세요.

## 선정한 배경

자율주행 모빌리티



로봇팔



화상, 통신, 위치의 기술 집약체

화상 통신, 위치 기술을 결합한 스마트 포장 시스템을 과제로 선정하였습니다

## 과제의 처음설계와 최종완성 변화사항을 알려 주세요.

| 초안 (3월)  | 완성 (6월)   |
|--|---|
| <p>(제품의 전 상품을 라이다 센서가 탑재된 로봇이 lifting하여 가져온다.<br/>그 후 로봇 팔은 가져온 품목에서 주문이 들어온<br/>수량만큼 꺼내어 효율적으로 포장 박스에 포장한다.</p> <p>로봇 팔 – 삼각함수를 통해 모든 상황의 각도 알고리즘 계산</p> <p>카메라 – 일정 프레임에 객체가 계속 들어오도록 개발</p> <p>라이다 센서 – Landmark를 통해 상품 위치 A,B,C를 저장하여<br/>제품 값이 들어오면 자동으로 물품 위치로 이동하여 제품을<br/>Lifting하여 가져온다</p> | <p>카메라를 통해 제품을 인식하고 좌표 값을 추출해서 로봇 팔이<br/>객체가 있는 좌표로 이동하여 객체를 집는다.</p> <p>로봇 팔 – 역기구학(Inverse kinematics)을 이용하여<br/>좌표 값이 주어진다면 각 관절이 움직여야 할 각도를 계산하여<br/>End-effector에 도달하게 함</p> <p>카메라 – YOLO 프로그램으로 객체를 학습하고 인식하여<br/>Bounding Box를 그려서 Bounding Box의 좌표를 찾아서<br/>객체의 좌표를 구함</p> |

여러분 수행한 과제에 탑재된 **핵심 기술 3개**를 소개 해 주세요.(난이도 순서로)

|                     |  |
|---------------------|--|
| 1.<br>3D 적재<br>알고리즘 | 서로 다른 두 종류의 블록의 길이와 개수가 정해질 때 최적의 박스를 선택하여 로봇팔을 이용해 최적의 3D 적재를 실행하는 알고리즘이다.  |
| 2.<br>역기구학<br>알고리즘  | 역기구학은 로봇 팔의 관절 각도를 계산하는 알고리즘이다. 원하는 엔드 이펙터(End-effector)의 위치와 방향을 알고 있을 때, 로봇 팔을 어떻게 움직여야 그 위치와 방향을 달성할 수 있는지 계산하는 알고리즘이다. 해당 알고리즘을 통해서 카메라에 있는 좌표를 얻고 로봇팔의 End-effector가 이동할 수 있도록 하는 기술이다. |
| 3.<br>카메라 좌표        | Localization/Detection을 통해 이미지에서의 내용물이 어디 있는지 확인하고 Object를 Bounding Box를 통해서 잡아낸 후 Bounding Box의 pixel 값을 계산하여 center 좌표를 추출한다.  |

# 초안 - 운반 로봇

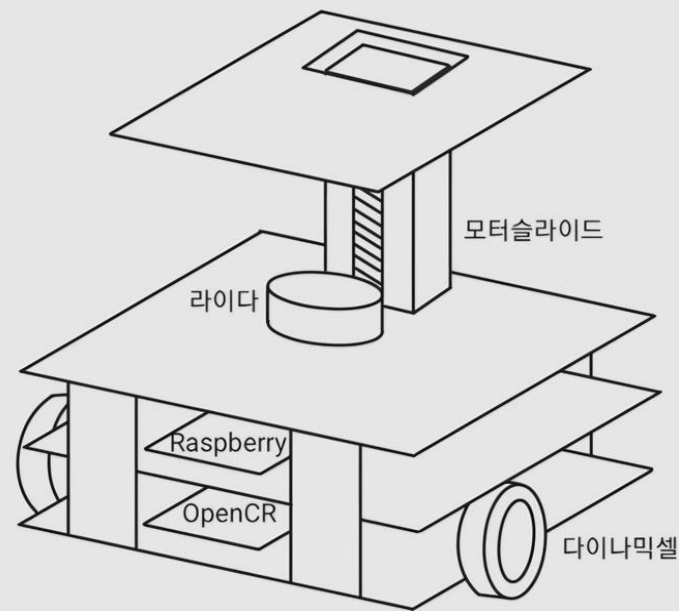
## 운반로봇 세부 사항

4층 : 모터슬라이드 + 짐 옮기는 판

3층 : 라이다 센서

2층 : OpenCR보드 + 배터리

1층 : 라즈베리파이 + 다이내믹셀

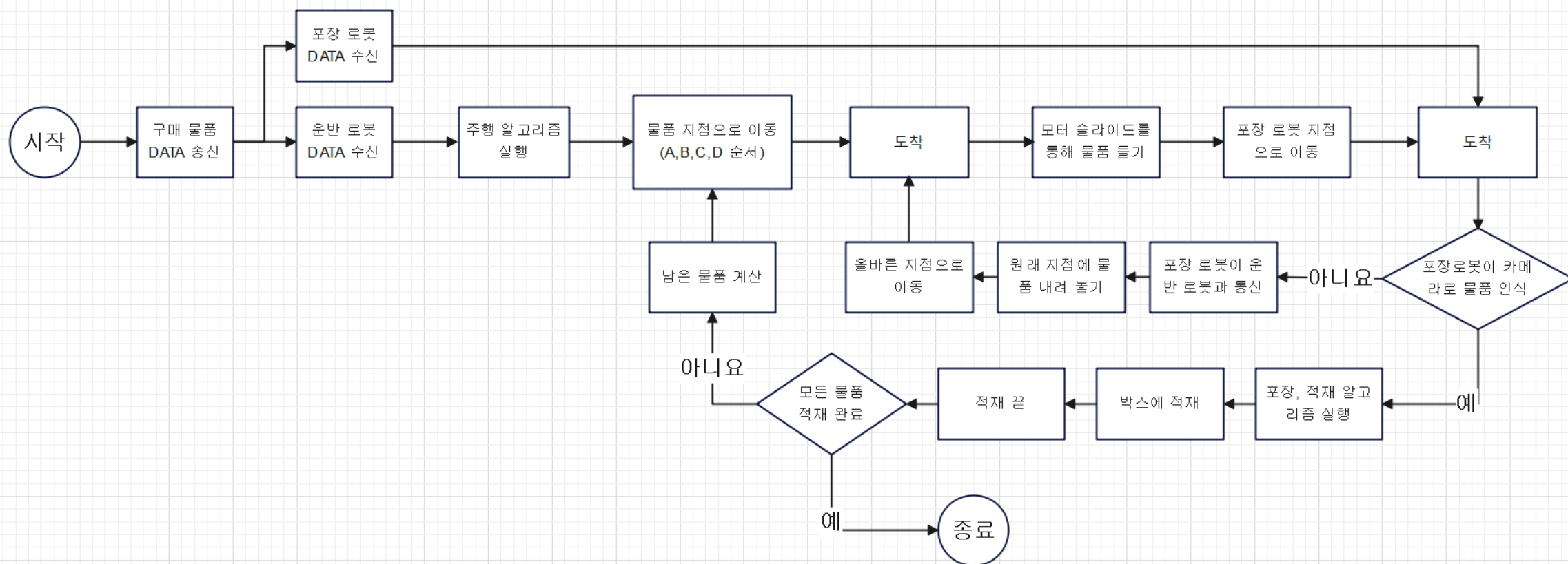


## 주요 과제

- Slam map에 landmark 설정
- Data 수신 시 자동 navigation 경로 설정
- TCP/IP 통신을 통한 ros-slave node 설정

# 초안 - 운반 로봇

Flow Chart



# 결과 - 운반로봇 HW

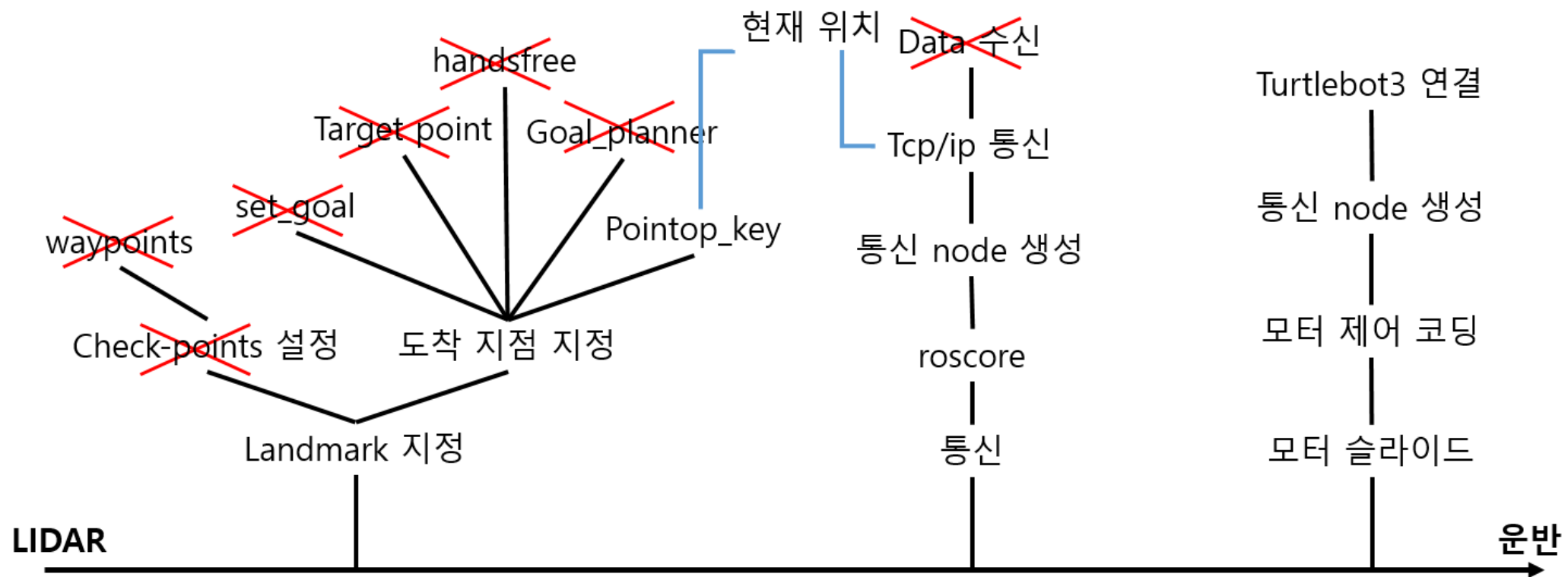


정면



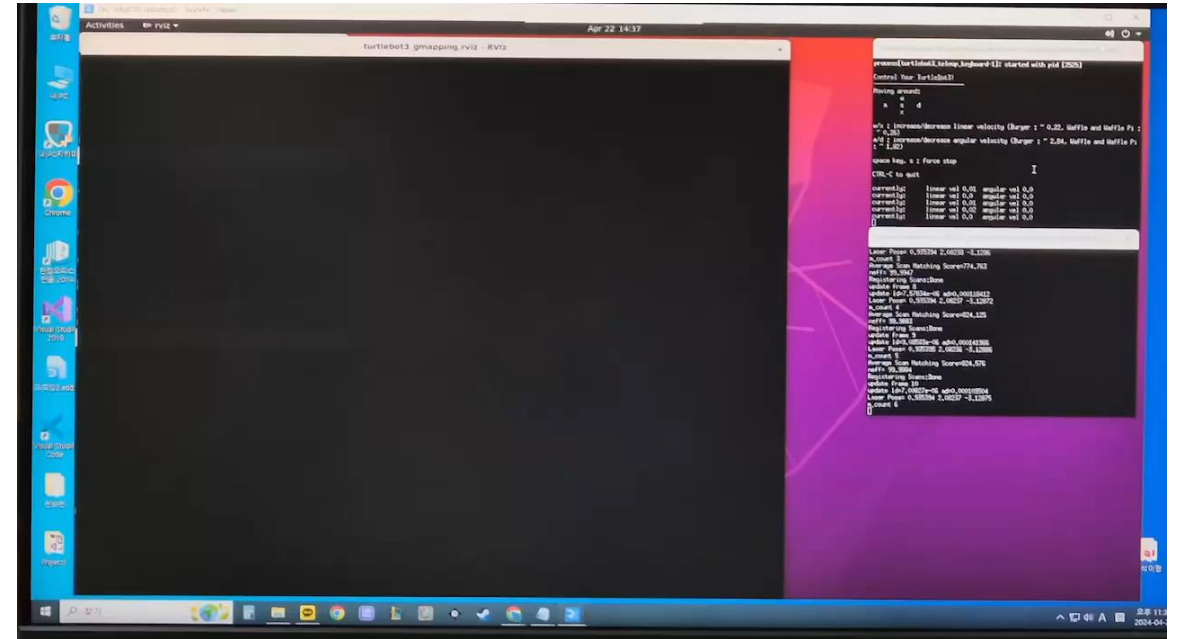
평면

## 결과 - 운반로봇 SW

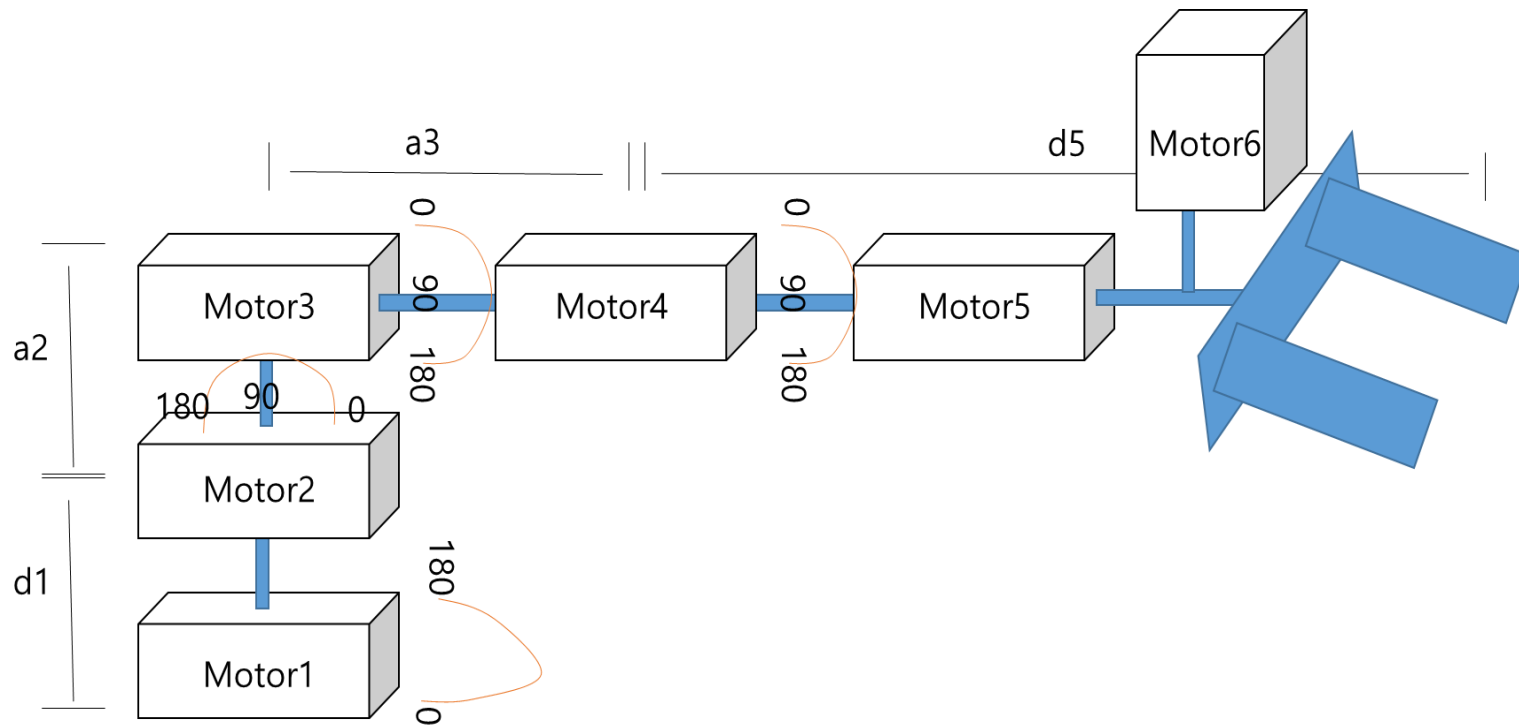




## 결과 - 운반로봇 동작



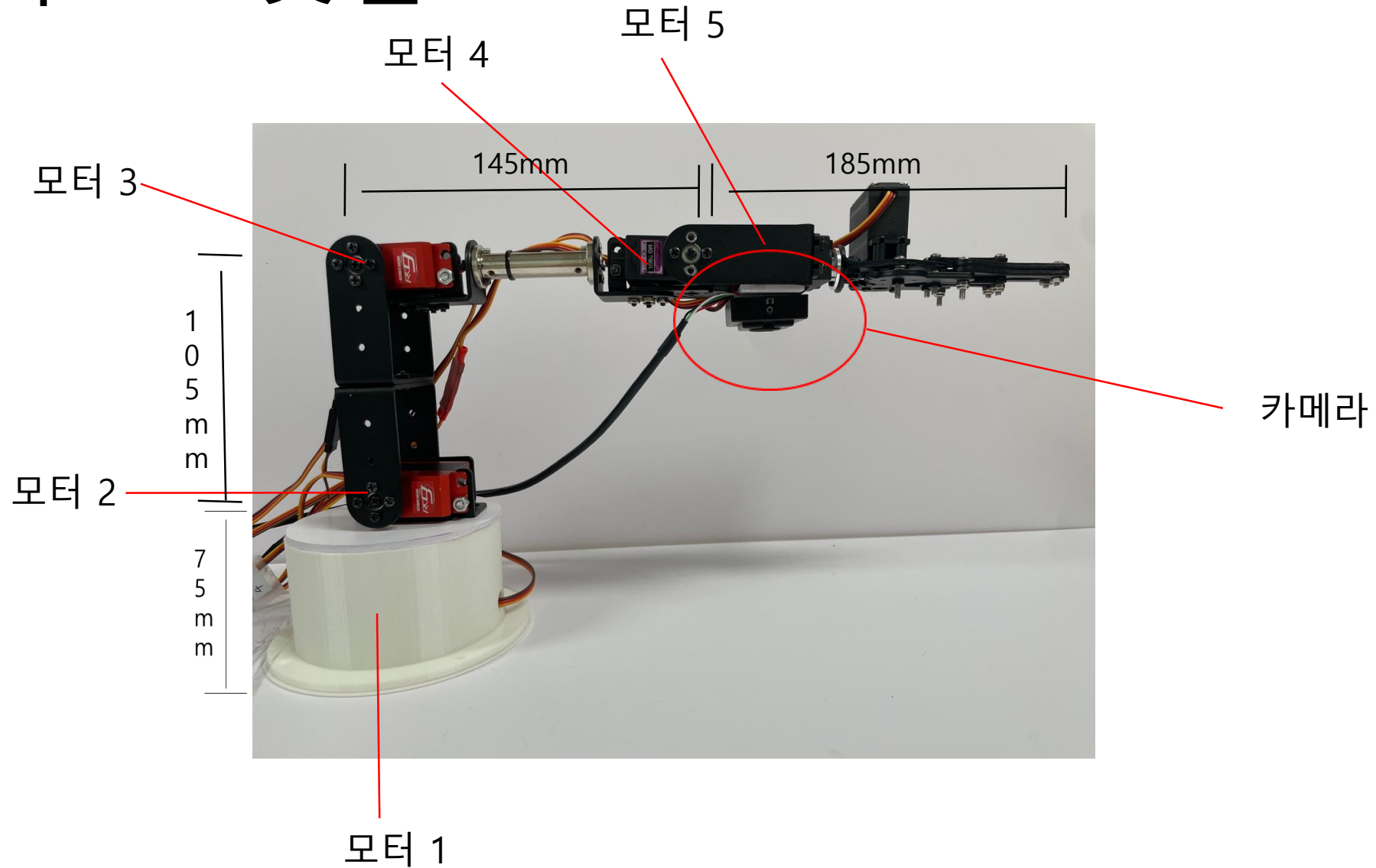
# 초안 - 로봇팔 HW



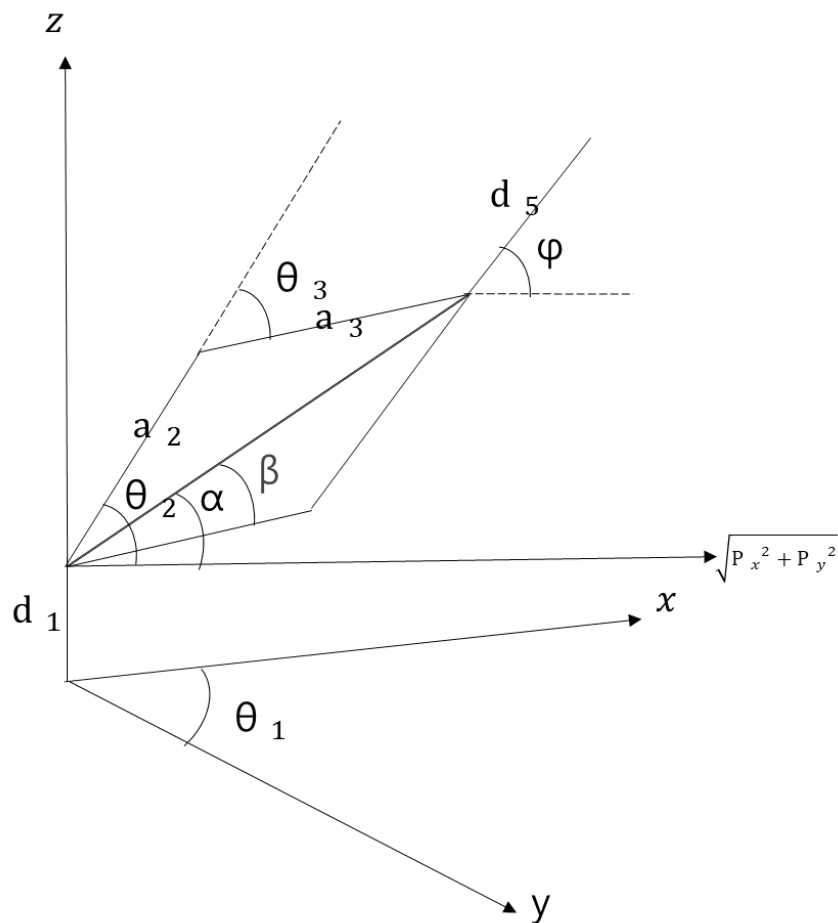
## 로봇팔 세부 사항

- 6축 자유도 로봇팔
- 카메라를 통한 객체 인식
- 역기구학 알고리즘 적용
- 적재 알고리즘 적용

# 결과 - 로봇팔 HW



# 결과 - 로봇팔 역기구학 알고리즘



$$\alpha = \text{atan2}(P_z - d_1 - d_5 s_{234}, \sqrt{P_x^2 + P_y^2} - d_5 c_{234})$$

$$\beta = \text{atan2}(a_3 s_3, a_2 + a_3 c_3)$$

$$\theta_2 = \alpha + \beta$$

$$\theta_4 = \theta_2 - \theta_3$$

$$P_x = c_1(a_2 c_2 + a_3 c_{23} + d_5 c_{234}) \cdots \textcircled{1}$$

$$P_y = s_1(a_2 c_2 + a_3 c_{23} + d_5 c_{234}) \cdots \textcircled{2}$$

$$P_z = d_1 + a_2 s_2 + a_3 s_{23} + d_5 s_{234} \cdots \textcircled{3}$$

$$\theta_1 = \text{atan2}(P_y, P_x)$$

$$\textcircled{1}^2 + \textcircled{2}^2 \rightarrow a_2 c_2 + a_3 c_{23} = \pm \sqrt{P_x^2 + P_y^2} - d_5 c_{234}$$

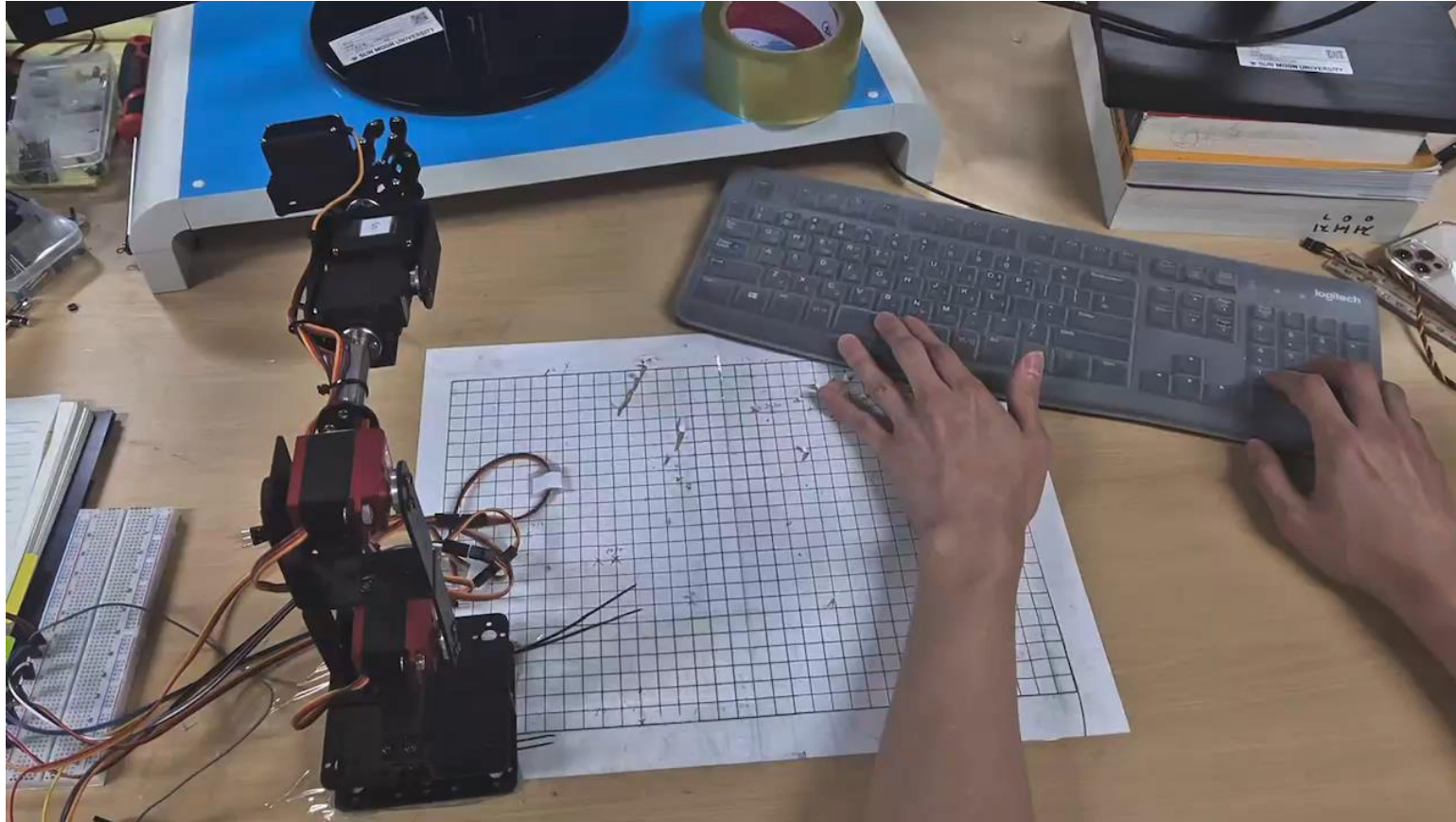
$$\textcircled{3} \rightarrow a_2 s_2 + a_3 s_{23} = P_z - d_1 - d_5 s_{234}$$

$$c_3 = \frac{(P_z - d_1 - d_5 s_{234})^2 + \sqrt{P_x^2 + P_y^2} - d_5 c_{234}^2 - a_2^2 - a_3^2}{2a_2 a_3}$$

$$s_3 = \sqrt{1 - c_3^2}$$

$$\theta_3 = \text{atan2}(s_3, c_3)$$

# 결과 - 로봇팔 시연 영상



# 카메라

1. 1pixel 당 길이 변환
2. 사물의 bounding box의 픽셀 좌표를 통해 w, h 길이 계산
3. w, h의 픽셀 개수를 cm 길이로 변환

Ex) 15cm의 픽셀 수 = 480

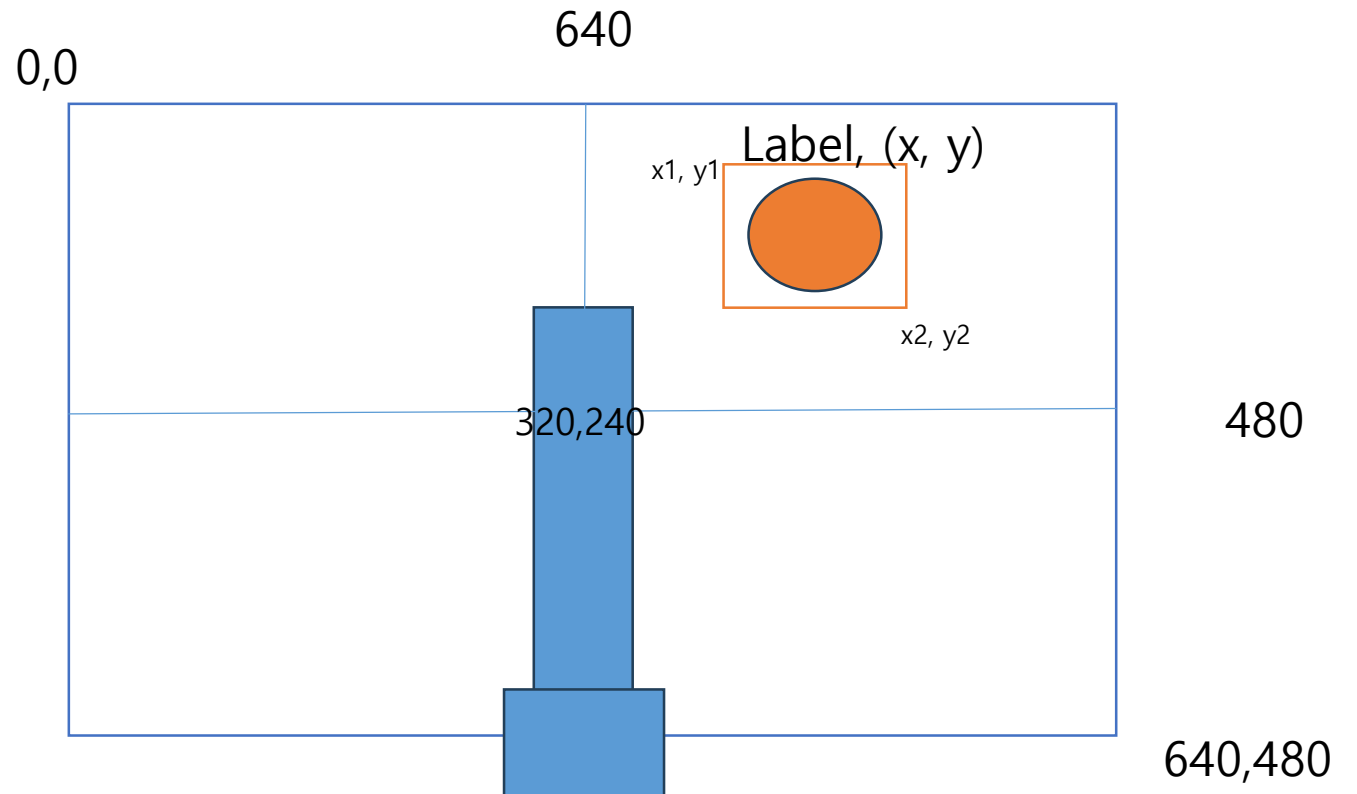
1cm 의 픽셀 수 =  $480/15 = 32$

$$x = ((x1 + x2)/2 - 320) / 32$$

$$y = ((640-y1 + 640-y2)/2 - 240) / 32$$

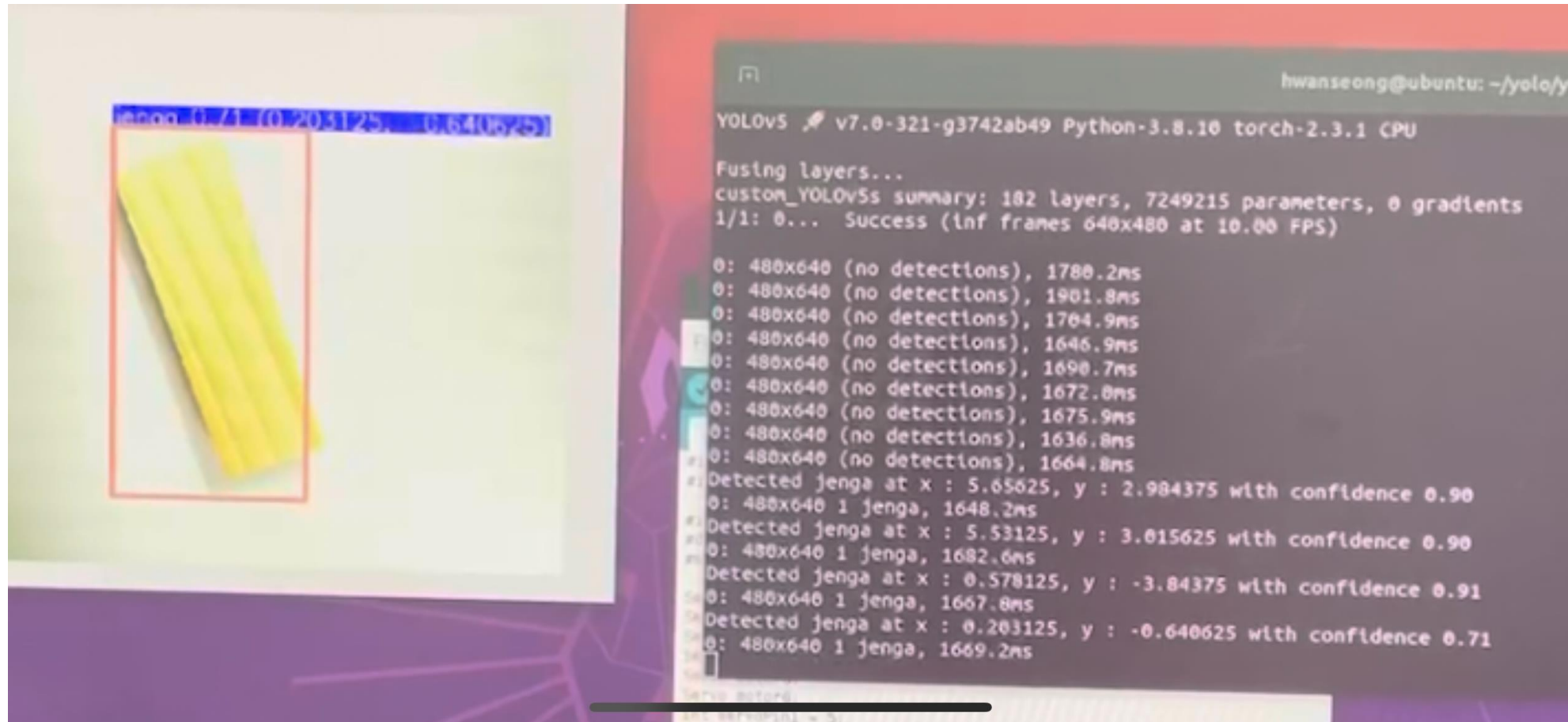
$$Px += x$$

$$Py += y$$

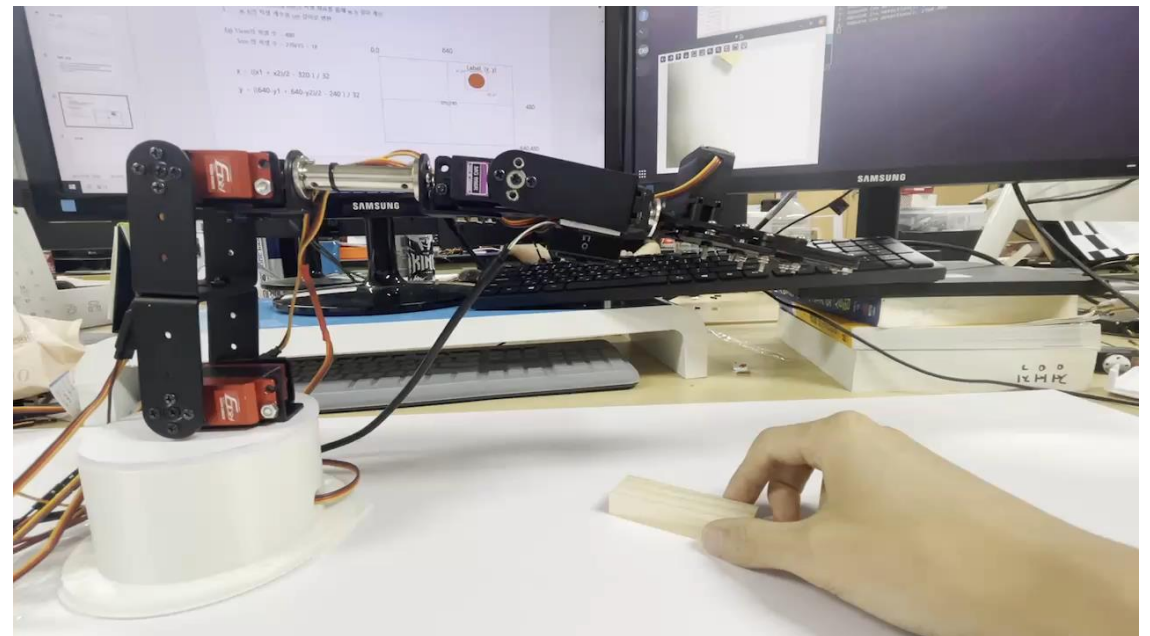
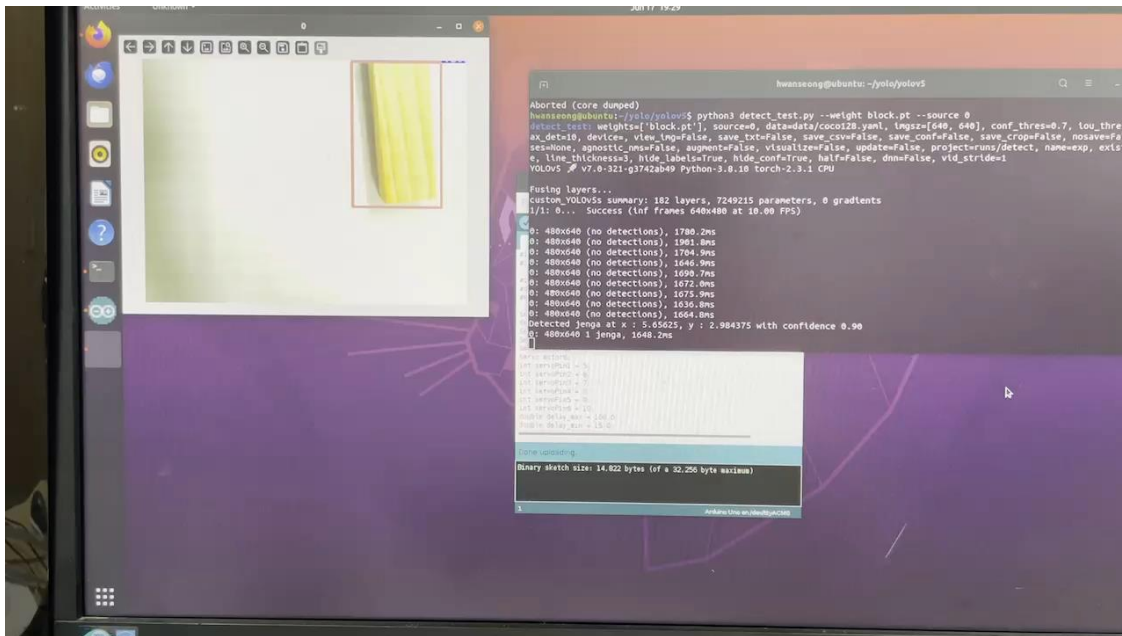




# 결과 - 카메라 object-detection



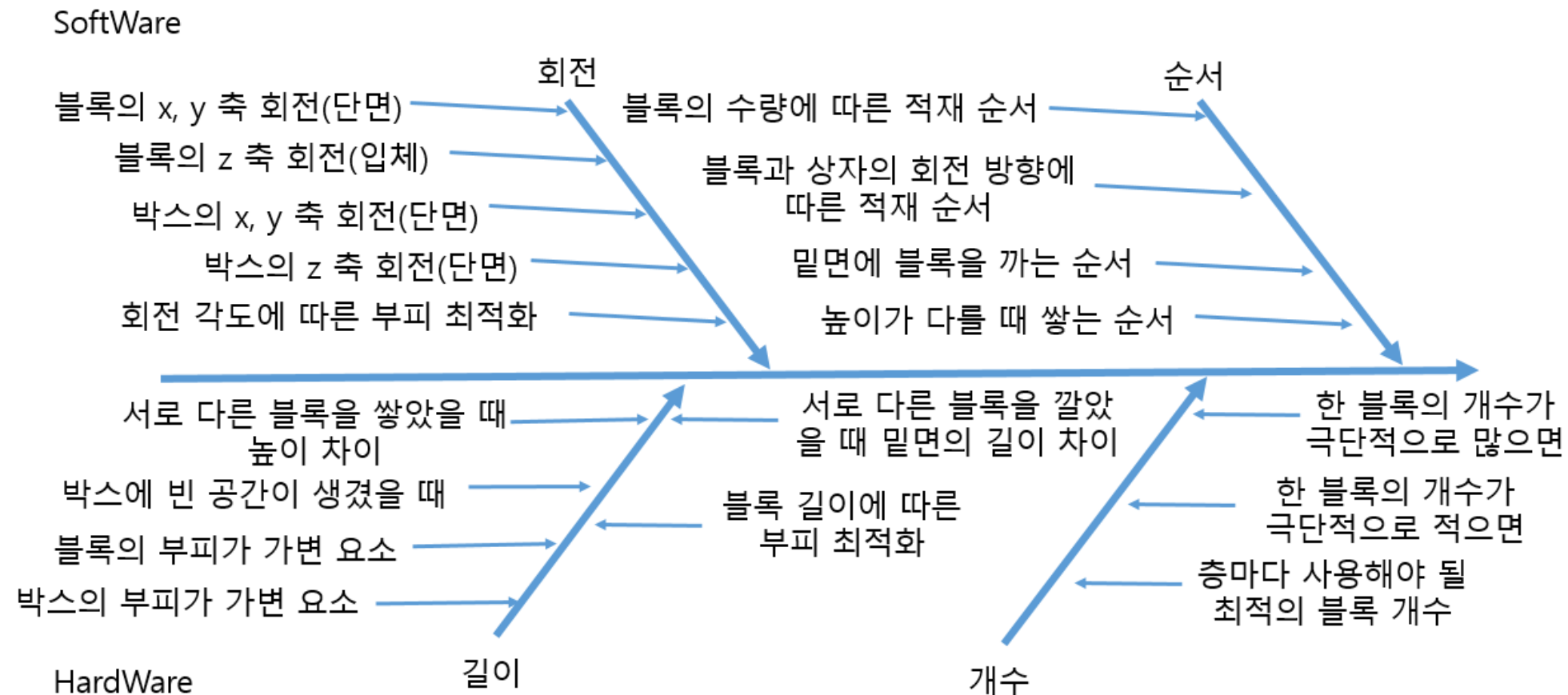
# 결과 - 로봇팔 시연 영상





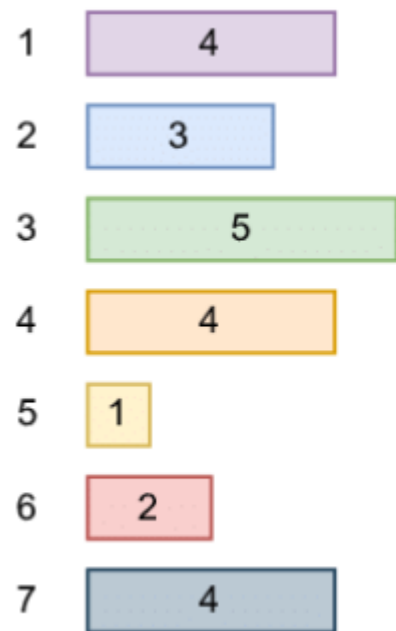
# 초안 - 3D 적재 알고리즘

발생가능한 문제점 fishbone-diagram 분석

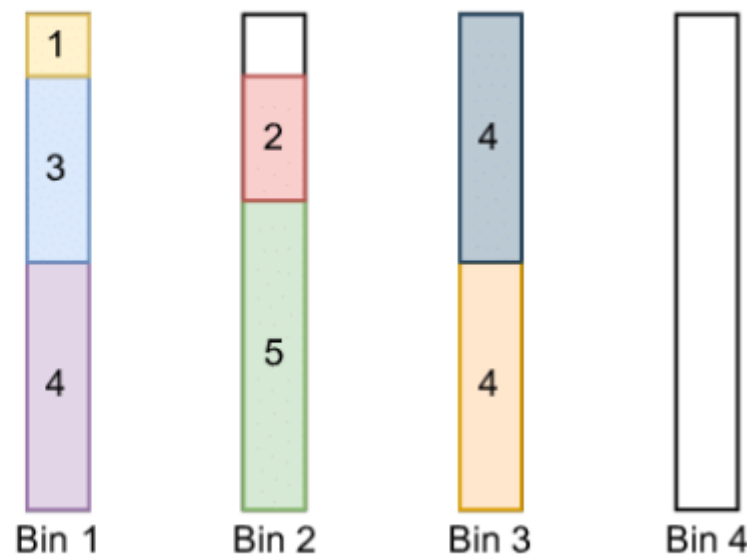


# 초안 - 3D 적재 알고리즘 SW

Initial objects (elements)



Bins (Containers, Blocks...) max capacity = 8



초기 선정 알고리즘

- Bin-packing 알고리즘
- 일반적인 packing 알고리즘
- 빈 공간에 큰 물건 순서로 적재

문제점

- 2D부터는 예외 상황 多
- 블록의 길이가 긴 경우 최적화 불가

Illustration of the bin packing problem

# 결과 - 3D 적재 알고리즘 SW

## scamper

### 대체

- 한 층을 시계방향으로 쌓기
- 상자의 대각선의 길이를 고려하여 쌓기
- 블록의 대각선의 길이를 고려하여 쌓기

### 결합

- 두 블록을 정사각형 모양(밀면)으로 만들어서 쌓기

### 적용/응용

- 블록을 상자에 쏟아넣고 흔들어서 맞추기
- 블록에 자석을 붙여 단차를 극복하기

### 변경/확대

- 작은 블록 먼저 쌓기

### 용도변경

- 블록의 길이 최적화에 맞춰 상자를 선택

### 제거/축소

- 한 종류의 블록으로만 층을 쌓기

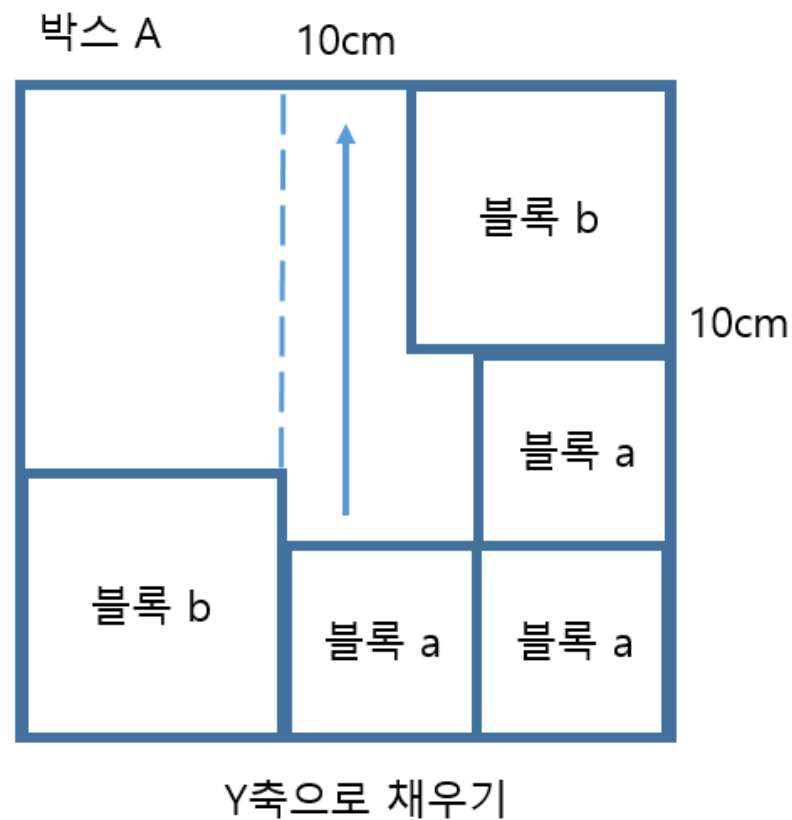
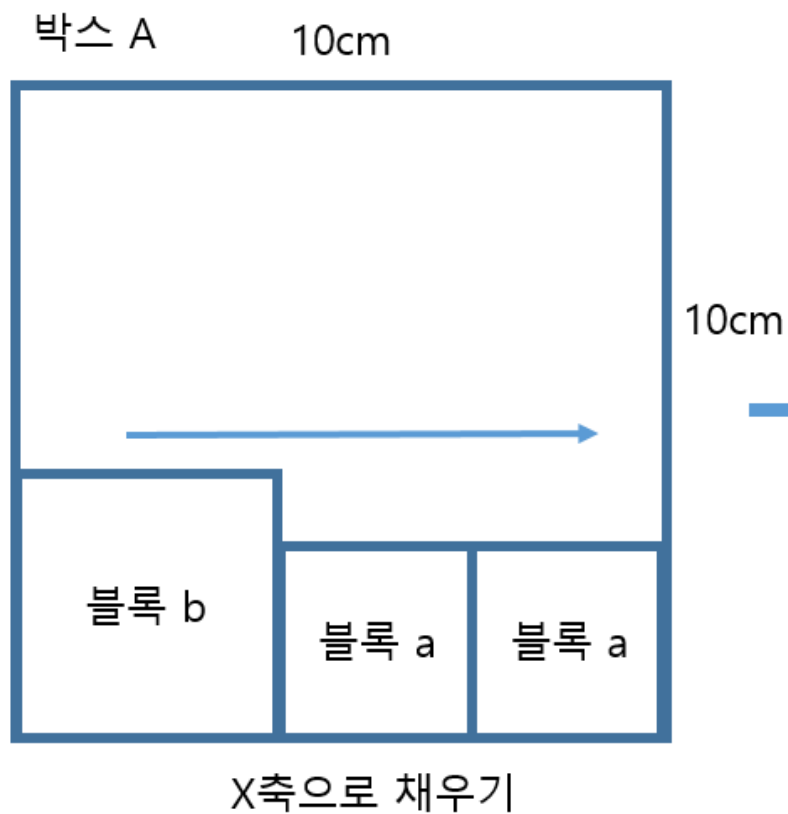
### 뒤집기/재배열

- 상자를 회전하여 쌓기
- 블록의 높이(최소 공배수)에 맞춰서 쌓기

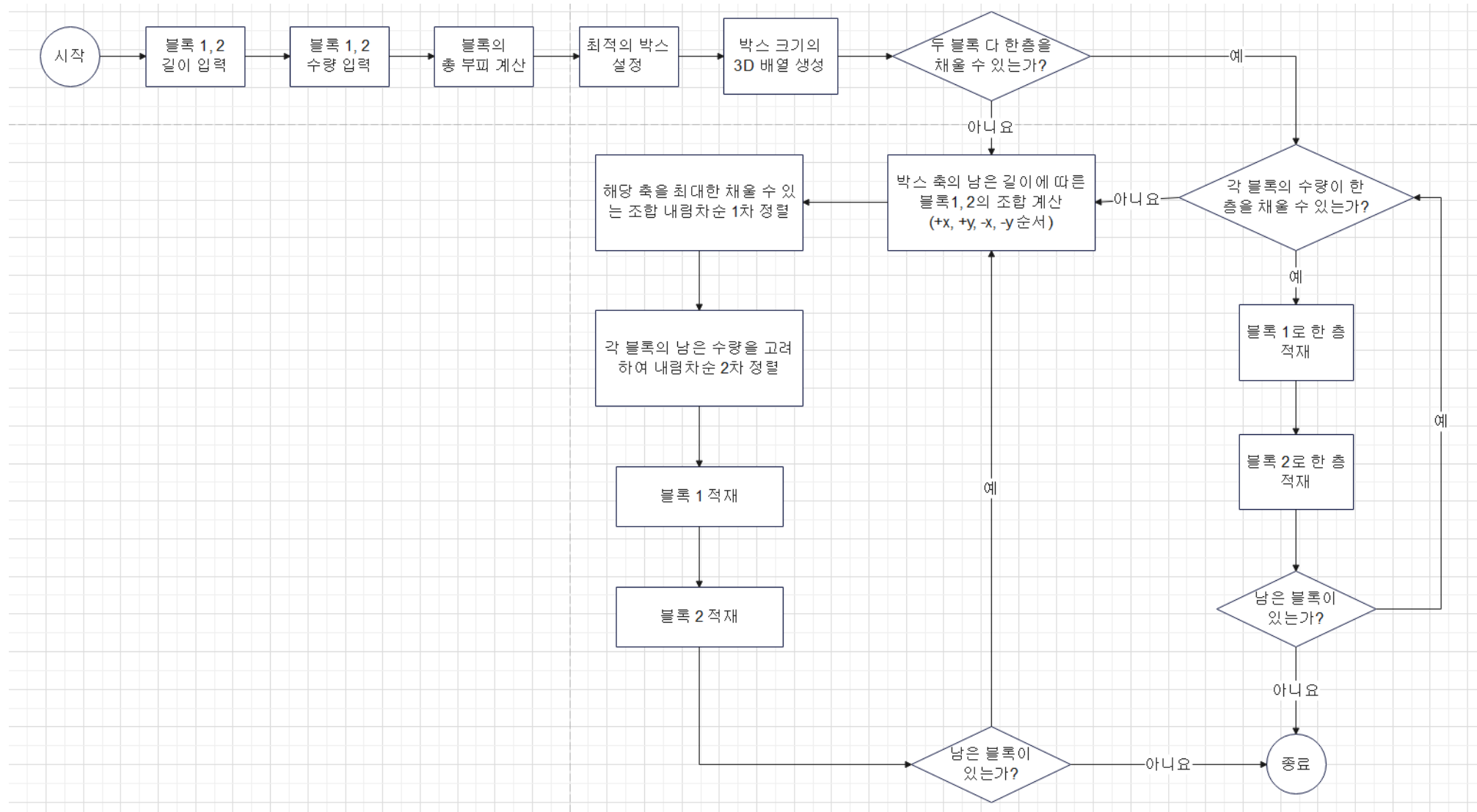
# 결과 - 3D 적재 알고리즘 SW

최종 선정 알고리즘

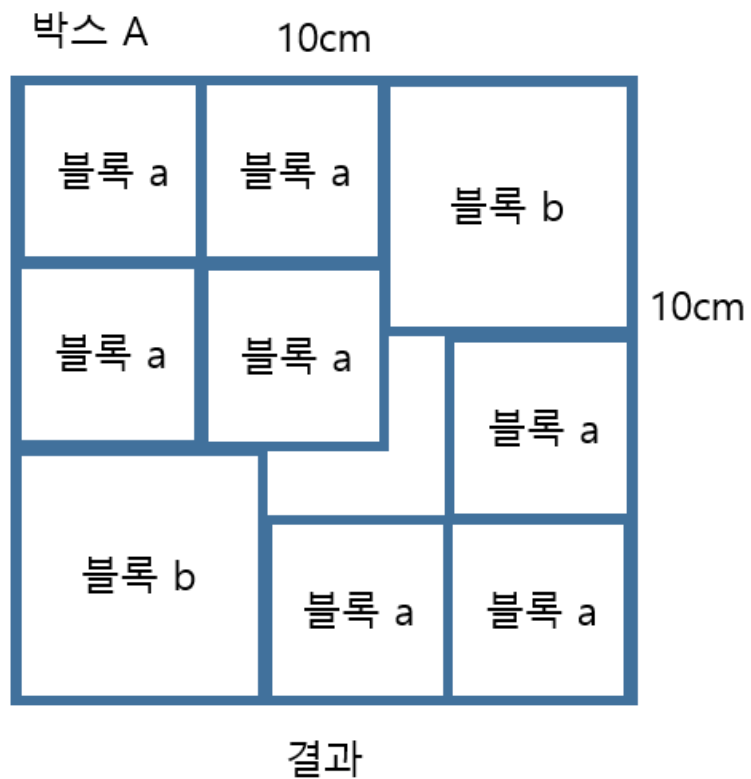
- 시계방향 • 반시계 방향으로 채우기



# 결과 - 3D 적재 알고리즘 SW



# 결과 - 3D 적재 알고리즘 SW

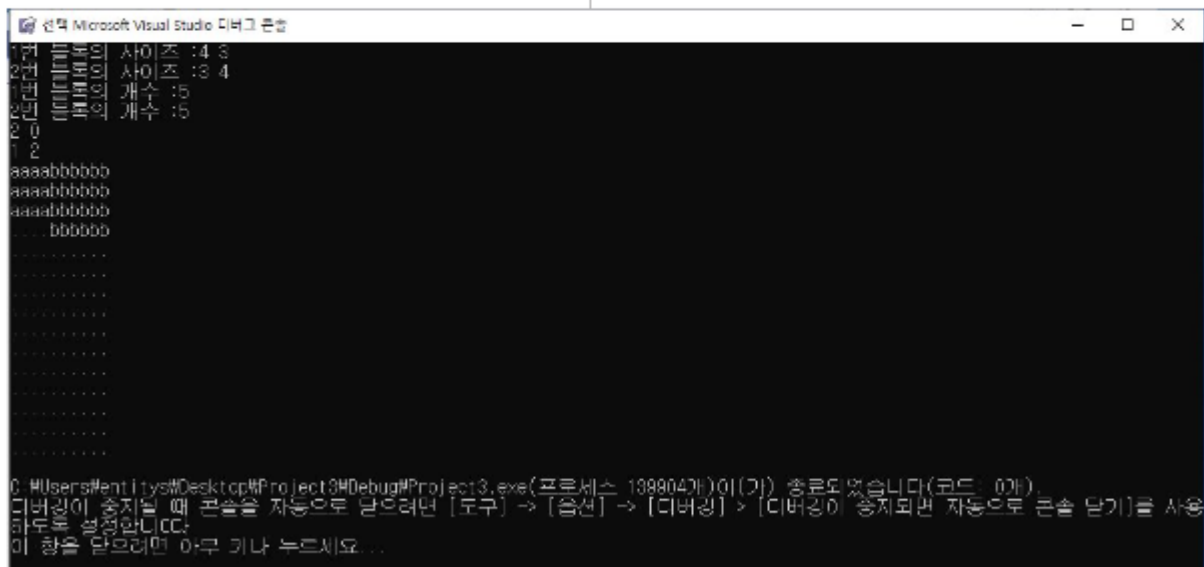


## 장점

- 각 블록의 수량에 따라 층마다 사용할 블록 수 최적화 쉬움
- 2D 적재의 예외 상황이 적어짐
- 블록의 적재 높이를 계산하기 쉬움

# 결과 - 3D 적재 알고리즘 SW

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4
5 int main()
6 {
7     char arr_block1[10][10]; // 1번 블록 설정
8     char arr_block2[10][10]; // 2번 블록 설정
9     char arr_box[15][15] = { 0 }; // 초기화를 0으로 설정하여 기본값을 빈칸으로 설정
10
11     char arr_sample[15][15]; // 막쓰는 배열
12
13     int i = 0; // 막쓰는 변수
14     int j = 0;
15     int k = 0;
16
17     int x1, y1; // 1번 블록의 길이 변수 선언
18     int x2, y2; // 2번 블록의 길이 변수 선언
19
20     int x = 0;
21     int y = 0;
22     int n, m; // 한 줄을 채울 때 필요한 블록의 수량
23     int xy[10][2]; // 한 줄을 채울 때 필요한 블록의 수량의 조합
24
25     int box_x = 10; // 박스 크기
26     int box_y = 15;
27
28     //int box_volume; // 박스 부피
29
30     int block1_x; // 1번 블록 크기
31     int block1_y;
32
33     int block2_x; // 2번 블록 크기
34     int block2_y;
35
36     int current_x = 0; // 막스 배열 변수
37     int current_y = 0;
38
39     int block1_n; // 1번 블록 개수
40     int block2_n; // 2번 블록 개수
41
42     int count_x = 0; // box_x 별찍기 counter
43     int count_y = 0; // box_y 별찍기 counter
44
45     printf("1번 블록의 사이즈 : ");
46     scanf_s("%d %d", &block1_x, &block1_y);
```



선택 Microsoft Visual Studio 디버그 콘솔

```
1 1번 블록의 사이즈 : 4 3
2 2번 블록의 사이즈 : 3 4
3 1번 블록의 개수 : 5
4 2번 블록의 개수 : 5
5
6 1 2
7 aaaaabbbbb
8 aaaaabbbbb
9 aaaaabbbbb
10 ..... bbbbbb
11 .....
12 .....
13 .....
14 .....
15 .....
16 .....
17 .....
18 .....
19 .....
20 .....
21 .....
22 .....
23 .....
24 .....
25 .....
26 .....
27 .....
28 .....
29 .....
30 .....
31 .....
32 .....
33 .....
34 .....
35 .....
36 .....
37 .....
38 .....
39 .....
40 .....
41 .....
42 .....
43 .....
44 .....
45 .....
46 .....
47 .....
48 .....
49 .....
50 .....
51 .....
52 .....
53 .....
54 .....
55 .....
56 .....
57 .....
58 .....
59 .....
60 .....
61 .....
62 .....
63 .....
64 .....
65 .....
66 .....
67 .....
68 .....
69 .....
70 .....
71 .....
72 .....
73 .....
74 .....
75 .....
76 .....
77 .....
78 .....
79 .....
80 .....
81 .....
82 .....
83 .....
84 .....
85 .....
86 .....
87 .....
88 .....
89 .....
90 .....
91 .....
92 .....
93 .....
94 .....
95 .....
96 .....
97 .....
98 .....
99 .....
100 .....
101 .....
102 .....
103 .....
104 .....
105 .....
106 .....
107 .....
108 .....
109 .....
110 .....
111 .....
112 .....
113 .....
114 .....
115 .....
116 .....
117 .....
118 .....
119 .....
120 .....
121 .....
122 .....
123 .....
124 .....
125 .....
126 .....
127 .....
128 .....
129 .....
130 .....
131 .....
132 .....
133 .....
134 .....
135 .....
136 .....
137 .....
138 .....
139 .....
140 .....
141 .....
142 .....
143 .....
144 .....
145 .....
146 .....
147 .....
148 .....
149 .....
150 .....
151 .....
152 .....
153 .....
154 .....
155 .....
156 .....
157 .....
158 .....
159 .....
160 .....
161 .....
162 .....
163 .....
164 .....
165 .....
166 .....
167 .....
168 .....
169 .....
170 .....
171 .....
172 .....
173 .....
174 .....
175 .....
176 .....
177 .....
178 .....
179 .....
180 .....
181 .....
182 .....
183 .....
184 .....
185 .....
186 .....
187 .....
188 .....
189 .....
190 .....
191 .....
192 .....
193 .....
194 .....
195 .....
196 .....
197 .....
198 .....
199 .....
200 .....
201 .....
202 .....
203 .....
204 .....
205 .....
206 .....
207 .....
208 .....
209 .....
210 .....
211 .....
212 .....
213 .....
214 .....
215 .....
216 .....
217 .....
218 .....
219 .....
220 .....
221 .....
222 .....
223 .....
224 .....
225 .....
226 .....
227 .....
228 .....
229 .....
230 .....
231 .....
232 .....
233 .....
234 .....
235 .....
236 .....
237 .....
238 .....
239 .....
240 .....
241 .....
242 .....
243 .....
244 .....
245 .....
246 .....
247 .....
248 .....
249 .....
250 .....
251 .....
252 .....
253 .....
254 .....
255 .....
256 .....
257 .....
258 .....
259 .....
260 .....
261 .....
262 .....
263 .....
264 .....
265 .....
266 .....
267 .....
268 .....
269 .....
270 .....
271 .....
272 .....
273 .....
274 .....
275 .....
276 .....
277 .....
278 .....
279 .....
280 .....
281 .....
282 .....
283 .....
284 .....
285 .....
286 .....
287 .....
288 .....
289 .....
290 .....
291 .....
292 .....
293 .....
294 .....
295 .....
296 .....
297 .....
298 .....
299 .....
300 .....
301 .....
302 .....
303 .....
304 .....
305 .....
306 .....
307 .....
308 .....
309 .....
310 .....
311 .....
312 .....
313 .....
314 .....
315 .....
316 .....
317 .....
318 .....
319 .....
320 .....
321 .....
322 .....
323 .....
324 .....
325 .....
326 .....
327 .....
328 .....
329 .....
330 .....
331 .....
332 .....
333 .....
334 .....
335 .....
336 .....
337 .....
338 .....
339 .....
340 .....
341 .....
342 .....
343 .....
344 .....
345 .....
346 .....
347 .....
348 .....
349 .....
350 .....
351 .....
352 .....
353 .....
354 .....
355 .....
356 .....
357 .....
358 .....
359 .....
360 .....
361 .....
362 .....
363 .....
364 .....
365 .....
366 .....
367 .....
368 .....
369 .....
370 .....
371 .....
372 .....
373 .....
374 .....
375 .....
376 .....
377 .....
378 .....
379 .....
380 .....
381 .....
382 .....
383 .....
384 .....
385 .....
386 .....
387 .....
388 .....
389 .....
390 .....
391 .....
392 .....
393 .....
394 .....
395 .....
396 .....
397 .....
398 .....
399 .....
400 .....
401 .....
402 .....
403 .....
404 .....
405 .....
406 .....
407 .....
408 .....
409 .....
410 .....
411 .....
412 .....
413 .....
414 .....
415 .....
416 .....
417 .....
418 .....
419 .....
420 .....
421 .....
422 .....
423 .....
424 .....
425 .....
426 .....
427 .....
428 .....
429 .....
430 .....
431 .....
432 .....
433 .....
434 .....
435 .....
436 .....
437 .....
438 .....
439 .....
440 .....
441 .....
442 .....
443 .....
444 .....
445 .....
446 .....
447 .....
448 .....
449 .....
450 .....
451 .....
452 .....
453 .....
454 .....
455 .....
456 .....
457 .....
458 .....
459 .....
460 .....
461 .....
462 .....
463 .....
464 .....
465 .....
466 .....
467 .....
468 .....
469 .....
470 .....
471 .....
472 .....
473 .....
474 .....
475 .....
476 .....
477 .....
478 .....
479 .....
480 .....
481 .....
482 .....
483 .....
484 .....
485 .....
486 .....
487 .....
488 .....
489 .....
490 .....
491 .....
492 .....
493 .....
494 .....
495 .....
496 .....
497 .....
498 .....
499 .....
500 .....
501 .....
502 .....
503 .....
504 .....
505 .....
506 .....
507 .....
508 .....
509 .....
510 .....
511 .....
512 .....
513 .....
514 .....
515 .....
516 .....
517 .....
518 .....
519 .....
520 .....
521 .....
522 .....
523 .....
524 .....
525 .....
526 .....
527 .....
528 .....
529 .....
530 .....
531 .....
532 .....
533 .....
534 .....
535 .....
536 .....
537 .....
538 .....
539 .....
540 .....
541 .....
542 .....
543 .....
544 .....
545 .....
546 .....
547 .....
548 .....
549 .....
550 .....
551 .....
552 .....
553 .....
554 .....
555 .....
556 .....
557 .....
558 .....
559 .....
560 .....
561 .....
562 .....
563 .....
564 .....
565 .....
566 .....
567 .....
568 .....
569 .....
570 .....
571 .....
572 .....
573 .....
574 .....
575 .....
576 .....
577 .....
578 .....
579 .....
580 .....
581 .....
582 .....
583 .....
584 .....
585 .....
586 .....
587 .....
588 .....
589 .....
590 .....
591 .....
592 .....
593 .....
594 .....
595 .....
596 .....
597 .....
598 .....
599 .....
600 .....
601 .....
602 .....
603 .....
604 .....
605 .....
606 .....
607 .....
608 .....
609 .....
610 .....
611 .....
612 .....
613 .....
614 .....
615 .....
616 .....
617 .....
618 .....
619 .....
620 .....
621 .....
622 .....
623 .....
624 .....
625 .....
626 .....
627 .....
628 .....
629 .....
630 .....
631 .....
632 .....
633 .....
634 .....
635 .....
636 .....
637 .....
638 .....
639 .....
640 .....
641 .....
642 .....
643 .....
644 .....
645 .....
646 .....
647 .....
648 .....
649 .....
650 .....
651 .....
652 .....
653 .....
654 .....
655 .....
656 .....
657 .....
658 .....
659 .....
660 .....
661 .....
662 .....
663 .....
664 .....
665 .....
666 .....
667 .....
668 .....
669 .....
670 .....
671 .....
672 .....
673 .....
674 .....
675 .....
676 .....
677 .....
678 .....
679 .....
680 .....
681 .....
682 .....
683 .....
684 .....
685 .....
686 .....
687 .....
688 .....
689 .....
690 .....
691 .....
692 .....
693 .....
694 .....
695 .....
696 .....
697 .....
698 .....
699 .....
700 .....
701 .....
702 .....
703 .....
704 .....
705 .....
706 .....
707 .....
708 .....
709 .....
710 .....
711 .....
712 .....
713 .....
714 .....
715 .....
716 .....
717 .....
718 .....
719 .....
720 .....
721 .....
722 .....
723 .....
724 .....
725 .....
726 .....
727 .....
728 .....
729 .....
730 .....
731 .....
732 .....
733 .....
734 .....
735 .....
736 .....
737 .....
738 .....
739 .....
740 .....
741 .....
742 .....
743 .....
744 .....
745 .....
746 .....
747 .....
748 .....
749 .....
750 .....
751 .....
752 .....
753 .....
754 .....
755 .....
756 .....
757 .....
758 .....
759 .....
760 .....
761 .....
762 .....
763 .....
764 .....
765 .....
766 .....
767 .....
768 .....
769 .....
770 .....
771 .....
772 .....
773 .....
774 .....
775 .....
776 .....
777 .....
778 .....
779 .....
780 .....
781 .....
782 .....
783 .....
784 .....
785 .....
786 .....
787 .....
788 .....
789 .....
790 .....
791 .....
792 .....
793 .....
794 .....
795 .....
796 .....
797 .....
798 .....
799 .....
800 .....
801 .....
802 .....
803 .....
804 .....
805 .....
806 .....
807 .....
808 .....
809 .....
810 .....
811 .....
812 .....
813 .....
814 .....
815 .....
816 .....
817 .....
818 .....
819 .....
820 .....
821 .....
822 .....
823 .....
824 .....
825 .....
826 .....
827 .....
828 .....
829 .....
830 .....
831 .....
832 .....
833 .....
834 .....
835 .....
836 .....
837 .....
838 .....
839 .....
840 .....
841 .....
842 .....
843 .....
844 .....
845 .....
846 .....
847 .....
848 .....
849 .....
850 .....
851 .....
852 .....
853 .....
854 .....
855 .....
856 .....
857 .....
858 .....
859 .....
860 .....
861 .....
862 .....
863 .....
864 .....
865 .....
866 .....
867 .....
868 .....
869 .....
870 .....
871 .....
872 .....
873 .....
874 .....
875 .....
876 .....
877 .....
878 .....
879 .....
880 .....
881 .....
882 .....
883 .....
884 .....
885 .....
886 .....
887 .....
888 .....
889 .....
890 .....
891 .....
892 .....
893 .....
894 .....
895 .....
896 .....
897 .....
898 .....
899 .....
900 .....
901 .....
902 .....
903 .....
904 .....
905 .....
906 .....
907 .....
908 .....
909 .....
910 .....
911 .....
912 .....
913 .....
914 .....
915 .....
916 .....
917 .....
918 .....
919 .....
920 .....
921 .....
922 .....
923 .....
924 .....
925 .....
926 .....
927 .....
928 .....
929 .....
930 .....
931 .....
932 .....
933 .....
934 .....
935 .....
936 .....
937 .....
938 .....
939 .....
940 .....
941 .....
942 .....
943 .....
944 .....
945 .....
946 .....
947 .....
948 .....
949 .....
950 .....
951 .....
952 .....
953 .....
954 .....
955 .....
956 .....
957 .....
958 .....
959 .....
960 .....
961 .....
962 .....
963 .....
964 .....
965 .....
966 .....
967 .....
968 .....
969 .....
970 .....
971 .....
972 .....
973 .....
974 .....
975 .....
976 .....
977 .....
978 .....
979 .....
980 .....
981 .....
982 .....
983 .....
984 .....
985 .....
986 .....
987 .....
988 .....
989 .....
990 .....
991 .....
992 .....
993 .....
994 .....
995 .....
996 .....
997 .....
998 .....
999 .....
1000 .....
```

0: H:\Users\entl\MyDesktop\Project3\Debug\Project3.exe(프로세스 199804가)이(가) 종료되었습니다(코드: 0x0).  
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용  
하도록 설정합니다.  
이 창을 닫으려면 아무 키나 누르세요...

## 과제 수행에서 발생한 문제점들이 뭐가 있었고, 어떻게 해결했나요 ?

|                     | 발생한 문제점  | 해결 방안   |
|---------------------|--|---|
| 1.<br>3D 적재<br>알고리즘 | <ul style="list-style-type: none"> <li>- 다양한 요소들에 의해 알고리즘에서 예외 상황이 발생함 (길이, 회전, 개수, 순서)</li> </ul>   | <ul style="list-style-type: none"> <li>- 예외 상황에 대해 fishbone-diagram를 작성함</li> <li>- Scammer를 통해 해결 가능한 방식을 도출함</li> </ul>   |
| 2.<br>로봇 팔          | <ul style="list-style-type: none"> <li>- Servo-motor를 이용하다 보니 -각도를 처리할 수가 없었음</li> </ul>             | <ul style="list-style-type: none"> <li>- Servo-motor의 각도를 90° 기준으로 <math>\pm 90^\circ</math>로 설정하여 동작하도록 설계 및 코딩</li> </ul> |
|                     | <ul style="list-style-type: none"> <li>- End-effector쪽 무게가 무거워서 팔을 지지하는 motor 1번 축이 기울어졌음</li> </ul> | <ul style="list-style-type: none"> <li>- motor 1번의 서보혼이 흔들리지 않도록 HW를 큰 원통형 판으로 구조 변경</li> </ul>                             |
|                     | <ul style="list-style-type: none"> <li>- End-effector가 앞으로 기울면 motor 3번에 부하때문에 발열이 심해짐</li> </ul>    | <ul style="list-style-type: none"> <li>- 각 motor가 필요한 토크 계산 후 적절한 토크의 모터로 교체</li> </ul>                                     |



## 과제 수행에서 발생한 문제점들이 뭐가 있었고, 어떻게 해결했나요 ?

|           | 발생한 문제점  | 해결 방안   |
|-----------|--|---|
| 3.<br>카메라 | - 영상 처리 과정에서의 정확성, 환경 변화에 대한 대응 등의 문제점이 있음   | - 객체 data를 YOLO 프로그램으로 더 많이 학습시켜 bounding box 정확도를 높임.<br>- 로봇팔 동작 delay를 증가시켜 카메라가 객체를 인식하는 시간을 늘림 |
|           | - 객체의 좌표를 찾을 수 없음  | - 카메라의 높이를 일정하게 유지하여 pixel 당 길이를 cm로 변환하여 좌표 값을 추출함   |
|           | - 2대의 카메라로 yolo 프로그램 실행 오류   |   |
|           | 추후 develop 방향  |   |
|           | 1. 카메라는 렌즈에 의해 왜곡이 생기는데, 이를 camera calibration을 통해 카메라의 내•외부 파라미터를 보정하여 카메라의 위치를 결정하고, 카메라와 객체의 거리를 계산하여 좌표를 추출<br>2. OpenCV 라이브러리를 통해 contour로 객체의 외곽 부분의 경계를 그리고 이 경계를 통해 중심 좌표를 추출<br>3. top position, side position의 2대의 카메라로 좌표 추출(보드 교체 또는 노트북에서 실행) |   |

## 체계적 문제해결 능력이란 무엇이라고 생각하나요?

|             |   |
|-------------|---|
| 조장<br>현승헌   | 체계적인 문제 해결 능력이란 빠르고 정확하게 결과에 도달하는 것이라고 생각합니다. 문제를 빠르게 해결하기 위해서는 다양한 해결 방식을 알아야 합니다. 문제를 정확하게 해결하기 위해서는 문제가 발생한 원인을 정확하게 파악해야 합니다. 이 두가지 선행 요건이 충족된다면 체계적인 문제 해결 능력을 갖췄다고 할 수 있습니다.  |
| 조원1<br>유환성  | 프로젝트를 진행할 때는 기한이 정해져 있고 그 시간을 얼마나 효율적이게 사용하냐에 따라 프로젝트의 성공률이 올라간다고 생각합니다.<br>체계적인 문제해결 능력이란 프로젝트를 진행하면서 나타나는 문제를 fishbone-diagram이나 마인드맵 등을 이용하여 한눈에 문제점을 파악하고 각각의 문제점을 팀워크를 통해 차근차근 해결해 나가는 능력이며 궁극적으로는 시간을 효율적으로 사용하는 능력이라고 생각합니다. |
| 조원2.<br>조수완 | 문제가 정의되면 다양한 해결책을 모색해서 문제를 해결해 나가는 것이 체계적인 문제해결이라고 생각합니다.<br>체계적인 문제 해결을 위해 문제를 명확하게 정의하고 구체적으로 규명하는 것이 가장 중요하다고 생각합니다. 문제를 해결하기 위해 fishbone 방법을 통해 해결 해야할 과제의 주제를 정하고 여러 해결책을 위한 정보를 수집하여 이번 과제를 해결해 나갔습니다.                        |