학번 : 20150405
이름 : 손기영
그룹 : group 8
지도교수 : 박영진
맡은 분야 : ROS

기간 : 3/19 ~ 3/25

3/19

컴퓨터의 문제로 ubuntu 와 ros 설치가 3/19 월요일날 완료되었다. 그래서 3/19 월요일 부터 ROS 과제 1을 시작하였다. 과제 1을 시작하기전 첫번째 ROS TA session 에 참석하여 전반적인 ROS 에 관한 설명을 들었다. ROS 설치는 klms 에 주어진 교과서를 보며 진행하였고 session 에서는 과제 1에 대한 큰 흐름을 들을 수 있었다.

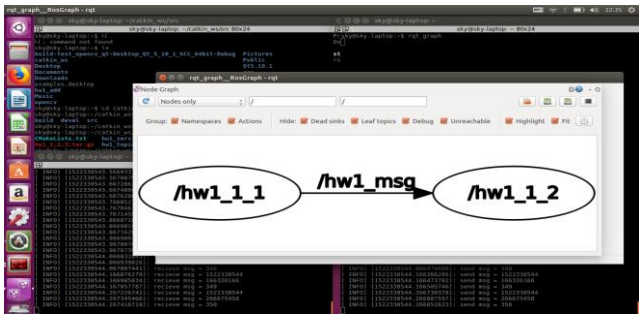
3/20

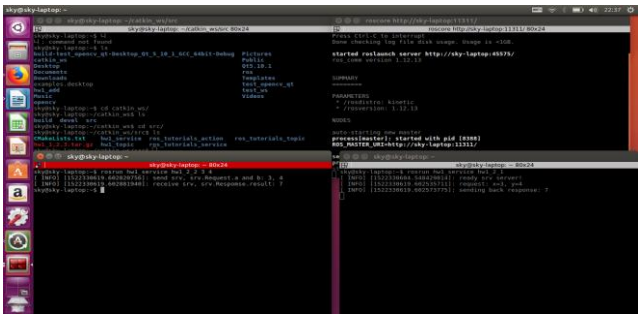
ROS 책을 보며 기본적인 예제를 따라해보았다. 과제로 나온것은 topic, service, launch 예제 해보기 였지만 전반적인 이해를 하고나서 과제를 진행하기 위해 과제보다 먼저 책 앞의 1~4 단원 까지 공부를 해보았다.

3/21

교수님과의 면담이 있었다.

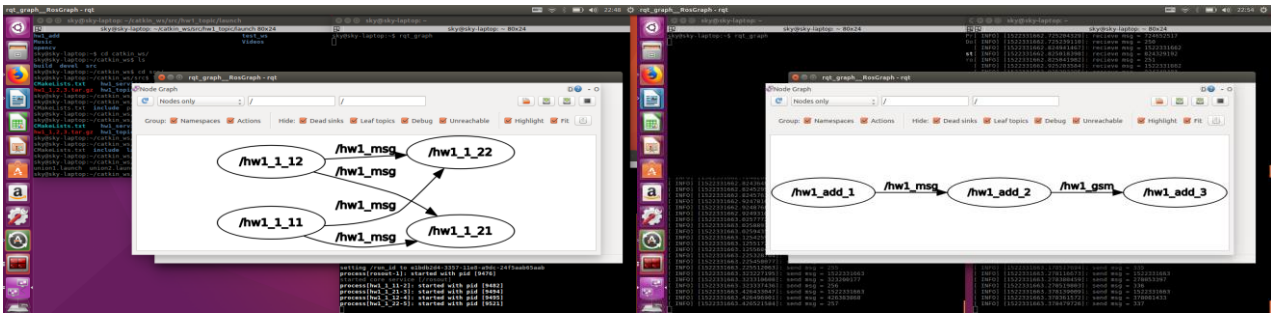
교수님과의 면담이 끝난 후 과제 1을 시작해 보았다. Cpp 라는 언어를 처음 사용하기에 처음에는 책에 그대로 나와있는 예제를 따라해보았다. 아직은 언어에 익숙치 않아 패키지명과 노드명을 변경하고 과제를





이행하는데 조금 문제가 있어 시간이 꽤 걸렸다. 이 날 과제 1 의 topic 과 service 부분을 하였다.

3/22



3/21 에 하고 남은 과제인 launch 와 additional 과제에 대해 숙제를 진행하였다.

Additional 과제의 경우 난 메시지 2개를 이용하여 코딩해주었는데 조교님께서 하나의 메시지로 해결해보라 하셔서 추가적으로 다시 진행해 보아야 할 것 같다. 저녁 밤늦게 같은 ROS 파트인 김경서를 만나 숙제 제출용 영상을 촬영하였다.

이후에 조모임을 진행하였는데 concept 를 최대한 detail 하게 하기 위해 여러 아이디어들을 제시하였다. 나의 경우 동력을 최소로 하기 위해 motor 를 1개만 사용해서 공 수집과 배출을 할 수 있는 디자인을 고려해 보았고 그 결과 주로 rack&pinion 구조를 사용한 concept 를 제시하였다.

3/23

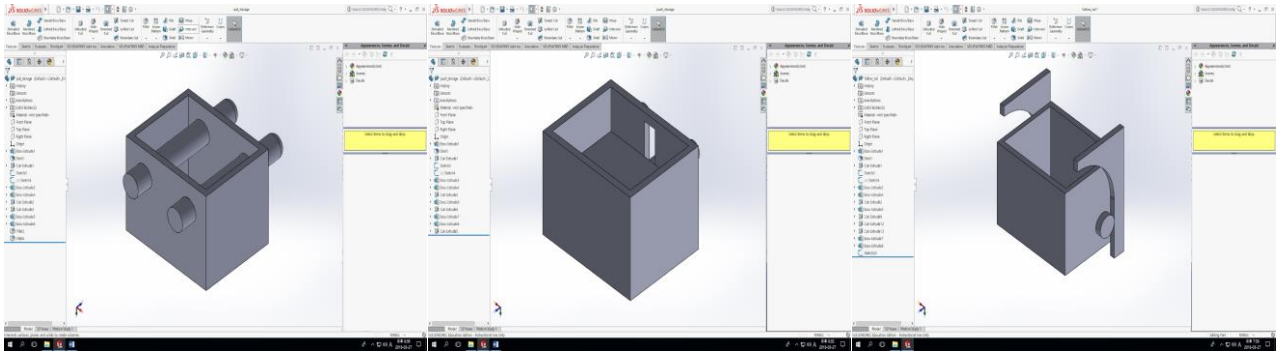
두번째 ROS TA session 이 아침 10시부터 1시까지 진행되었다. 가서 기본적으로 사용되어지는 source code 를 받고 간단한 설명을 들었다.

이후에 ROS 교과서 5챕터를 보며 혼자 공부해보았다.

기간 : 3/26 ~ 4/1

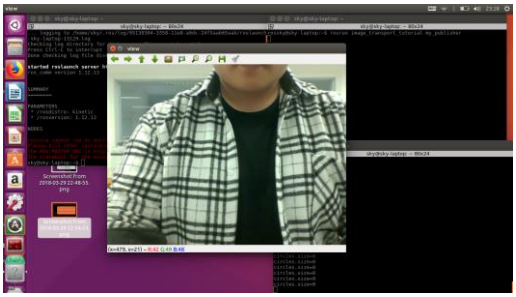
3/26

3/30 이 capstone design1 의 첫번째 발표날이기에 이에 필요한 solid works model 을 제작하였다. 나의 경우 ROS 파트 이지만 과목 특성상 학기 초반에는 solid works 파트가 할 일이 많기 때문에 solid works 파트를 도와주었다. 내가 디자인한 각각의 model 들은 각각 storge 자체를 떨어뜨리는 model, 위에서 공을 눌러서 들어온대로 다시 나가게 하는 model, 그리고 rack&pinion 을 이용한 모델로 rack 을 linear 한 파트와 curve 를 섞어서 모터 하나로 공 수집부터 배출까지 가능하게 한 model 이었다.



3/27

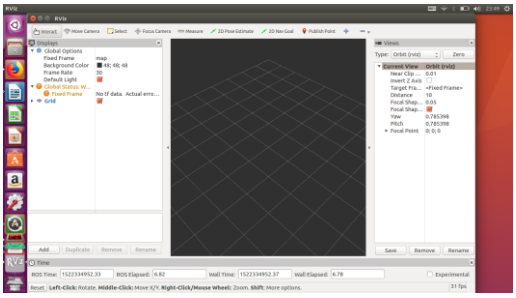
3/23 에 받은 source code 에 대해 하나씩 공부해보았다. 이 code 의 경우 내가 짠 코드가 아니기에 code 를 어떻게 짰는지 그리고 어떻게 돌아가는지에 대해 파악하고 우리 조의 모델에 잘 사용할 수 있게끔 해야 하는것이 목적이다. 이 날은 image 를 webcam 으로 어떻게 작동이 되는지에 대해 파악해보았다. 우선 이 code 를 시행시키기 위해서는 open CV 를 깔아야 했기에 open CV 부터 설치해주었다. 이 code 의 경우 코드 자체가 길지않고



주석이 잘 달려 있어서 이해하는데 큰 어려움이 있지는 않았다. 이 code 에 대한 의문점은 code 의 주석들을 보면 ball 을 detect 하고 이의 개수, 위치를 파악한다고 되어있는데 직접 실험해보니 ball 을 잘 detect 하지 못한다.

3/28

3/30 에 있을 1차 발표 리허설을 교수님과 함께 해보았다. 나는 교수님이 우리 그룹의 발표에 대해 feedback 해주시는걸 모두 다 기록하는 역할을 하였다.



3/29

rviz 에 대해 간단히 공부해보았다. 왜냐면 우선적으로 생각하는 ROS 의 공부목표가 vision 에 관련된 것인데 이는 RPlidar 와 webcam 이 연동된다. 이 두가지 결과가 rviz 로 출력이 되어지는데 지금 주어받은 code 들은 이를 사용하지 않기에 rviz 에 대한 공부가 필요하다고 판단했다. frame 을 바꾸어 보고 tool 을 추가하며 여러 시도를 해보았다. 하지만 교과서에 나와있지 않았기 때문에 저번 두번째 TA session 의 기억을 더듬으며 하여 공부의 효율이 좋지 않았다. 이에 대한 공부는 다음으로 넘기고 나머지 것들을 공부하고 추후에 조교님께 질문을 많이 드려야 할 것 같다.

3/30

오늘 1차발표가 있었다. 우리조의 경우 리허설때와 달리 발표당시 ppt 를 넘기는 과정에서 문제가 생겨 연습했던 만큼 발표하지 못해 피드백을 얻지못했다. 그래서 자체적으로 많은 고민을 해보았는데 rack&pinion 이 에너지 효율이 안좋은데 우리 조의 경우 두개를 사용하므로 power 적인 측면에서 좋지 않다고 생각이 들었다. 그래서 rack&pinion 을 두개 쓰는 메커니즘 대신 사용할 수 있을만한 메커니즘을 조사해보았다. 먼저 여러가지 종류의 6, 8 bar linkage 들의 움직임 영상들을 보며 쓸만한 구조를 생각해보았다. 그결과 8bar linkage 중 elevator linkage 라는 것이 쓸만할 것 같다 생각이 들었다. 또한 linkage system 말고도 나사산 있는 두개의 shaft 와 톱니를 써서 선형, 곡선 운동가능한 모델도 찾았다.
기간 : 4/2 ~ 4/8

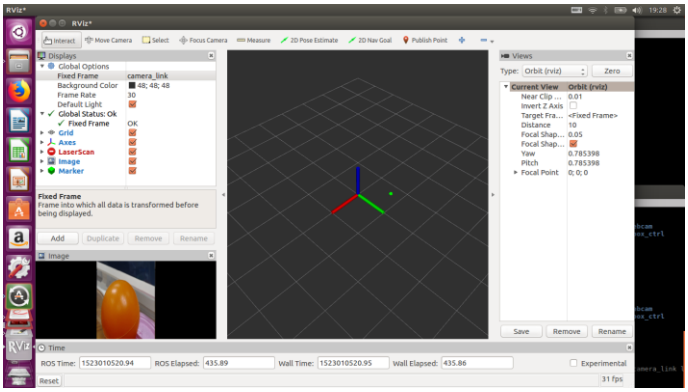
4/3

1차 발표에 대한 자체 피드백을 하였다. 우리조의 경우 교수님들에게 거의 피드백을 받지 못하였기 때문에 꼭 필요한 과정이라고 생각했다. 다른조와 비교하였을때 현재 발표는 하지 않았지만 software 적인 측면에서는 openCV 는 다른조와 진도가 비슷하지만 나머지 두 파트가 조금 뒤쳐진 것 같아 분발하기로 하였고, heat management, vibration 등의 우리가 아직까지 고려하지 못한 부분에 대해서는 다른조의 발표를 기록해둔 정보를 많이 인용하면 좋을거 같다고 판단했다. 다음 교수님과의 미팅전까지 새로운 mechanism 을 위해 bar-linkage system 등 rotation 을 linear motion 으로 바꿔주는 다양한 method 들을 찾아보았다.

4/4

조모임때 ROS 파트끼리 빠져서 github 에서 받은 코드에 대해 공부해보았다. 우선적으로 webcam 과 rplidar 를 둘 다 한군데에 띄우기 위해 공부를 진행하였다. webcam 의 경우 image_transport_tutorial package 의 코드를 돌리고, rplidar 는 그냥 돌렸는데 공이 잘 detect 되지 않았다. 그래서 package 들을 뒤적거리다가 webcam, ball_detection package 를 찾아서 그것을 image_transport_tutorial package 대신 돌리니까 ball_detect 까지는 진행이 잘되었다. 다음으로는 rviz 에서 각각을 띄우고 새로운 frame 에 두가지를 동시에 띄우는 작업을 시도하였다.

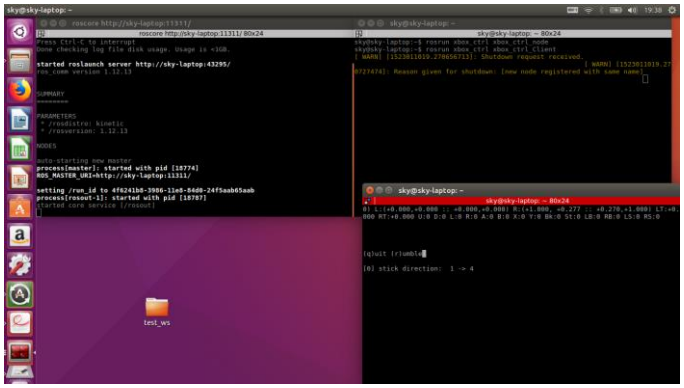
4/5



처음에는 잘 되지 않았다. 이게 장비를 추가할때 add 에서 topic 을 추가해야되는데 그걸 모르고, 또 카메라 추가할때 이미지인지, 이미지면 compressed 인지 raw 인지 몰라서 많은 시행착오 끝에 오른쪽과 같이 axes 표시하고, ball 에 marker 띄우고 까지 완성했다. rplidar 와 함께 표시하는것의 경우에는 tf 를 이용해서 camera_link 와 laser_map 의 좌표를 맞춰주고 base_link 하나에 다 내용을 전달해주어 base_link 에서 함께 돌아가는 것을 확인해 볼 수 있었다. 하지만 이걸 해보고 얻은 문제점이 있다. 첫번째는 rplidar 가 같은 layer 상에 있는 물체밖에 detect 하지 못해 현재 우리 group 의 디자인처럼 rplidar 가 위에 위치한 경우 바닥에 있는 공을 detect 하지 못할거 같아 prototype 제작전에 실험해봐야 할 것같고, 안된다면 rplidar 를 아래에 다는 디자인으로 수정해야 할 것 같다. 두번째는 openCV 코드와 ROS 코드를 아직 섞지 않아서 ball 을 잘 detect 하지 못한다. 이 문제는 추후에 openCV 코드와 합치면서 해결해야 할 것 같다.

4/6

어제 있었던 교수님과의 미팅에서 교수님이 현재 우리 group 의 collecting mechanism 인 로봇이 멈춘뒤 stamping 이라는 것의 문제점을 많이 지적해 주셔서 새로운 mechanism 을 생각해보았다. 이전의 문제점중 가장 큰 문제인 로봇이 자꾸 서야 하여 동력손실이 심하고 제어가 어렵고, 그리고 시간낭비가 심하다는 점을 고려해 멈추지 않고 계속 이동하면서 공을 줍는 방법에 대해 고려했다. 우리 조 말고 6개의 조가 현재 사용하는 mechanism 인 blade 에는 공을 줍는데 있어 불확실성이 있는 단점이 있었다. 그래서 우리 조는 이를 가로가 아닌 세로로 세우고, blade 형태가 아닌 보호막이 있는 arm 형태로 제작하여 움직이며 공을 줍는 mechanism 을 고려해보았고, 저번 mechanism 에서 칭찬받은 고무줄의 경우 collect 한 ball 이 storage 에서 벗어나려 할때 막아주고 들어올때는 거슬리지 않도록 배치하여 저번 mechanism 과의 연성을 주었다.



4/7

다음주에 myrio 와 서버를 연결해 보기 위해 xbox 에 대해 공부했다. README 파일에 있는대로 driver 를 설치해주고 xbox_ctrl_node 를 실행시켜보았다. 이후에 일단 개인용 컴퓨터와 잘 연결되있는지 확인해보기 위해 comm_tcp 로 확인해 주었다.
기간 : 4/9 ~ 4/15

4/9

조모임때 새로운 mechanism 의 dimension 들을 좀 detail 하게 잡고, 이 프로젝트에서 로봇이 어떻게 행동하는지에 대한 algorithm 을 생각해보았다. 이후에 ROS 와 openCV 의 ball_detection 코드를 합쳐보기 위해 서로의 코드를 설명하는 시간을 가졌다. 코드에 대한 설명이 끝난 뒤 openCV 의 ball_detection 코드가 좀 더 정확성을 띄기에 path generation algorithm 을 어떻게 짤지에 대해 생각해 보기 위해 코드를 실행시켜 어느정도 거리까지 측정이 되는지 확인해보았다. 그 결과 1~1.5m 까지의 공 밖에 detect 하지 못했다. openCV 쪽에서 좀 더 개선을 앞으로 해나가겠다고 하여 현재 상황에서의 path generation 과 개선되어서 멀리서도 ball 을 잘 detect 하는 경우에 path generation 두 가지 모두에 대해 생각해보았다. 첫번째로 현재상황에서는 어떻게든 공을 하나 찾을 때까지 이동한 다음에 처음의 공을 기준으로 외곽으로 돌면서 ball 을 줍는거고, 멀리서도 ball 을 잘

detect 하는 경우에는 멀리서 파란 ball 만 찾아서 직선으로 이었을때 빨간 ball 이 경로안에 안들어오면 그 길로 따라가고, 아닌경우에는 빨간 ball 에 로봇의 반경만큼 추가적인 반지름을 갖는 원을 그리고 그 접선을 타고 돌게끔 이동을 시킬 예정이다. ROS 와 openCV 의 코드를 잇는 작업은 다음주에 시험이 없는 내가 집에서 해보기로 했다. 이후 조모임이 끝난 뒤 myrio 와 ROS 를 연결해 보려는 시도를 하였다. 밤이 늦어 1시간 정도만 시도해보았는데, 처음에는 comm_tcg 코드로 연결상태를 확인하고 다음으로 넘어가려했는데 잘 되지않았다. 이후에 조교님들이 comm_tcg 코드 안써도 된대서 쓰지 않고 시도해보았다. 이날의 결론은 현재 조교님들께 받은 코드로는 계속된 에러가 나서 개선을 해야된단걸 느꼈다.

4/10

myRIO 파트의 조현근씨와 7시에 만나 어제 해결하지 못한 연결문제를 해결해보려 하였다. LAB view 코드를 계속 손을 보다 보니 어느순간부터 데이터를 받기 시작했다. 하지만 데이터를 받는데 현재 받아들이는 상황으로 봤을때는 컨트롤러 키 한번 움직일때마다 몇 메가바이트씩 정보를 받아들였다. 이를 곧이 곧대로 받아들여 생각을 해보면 몇번 꾹꾹 누르면 영화하나 보내는건데 상식적으로 틀렸다는것을 직감할 수 있었다. 그래서 LAB view 를 원래대로 돌려놓고 ROS 코드를 변경하기 시작했다. ROS 코드와 LAB view 코드를 분석해보았을때 서로 주고받는 메시지의 양식이 다른것 같다는 것을 느꼈지만 고치지는 못했다. 조교님들께 질문을 했지만 각각의 파트 조교님들이 서로 다른 파트가 잘못됐을 것이라고 그 쪽 조교한테 질문하라 그러셔서 결국에는 해결하지 못했다.그래서 우선 ROS 에서 실행한 파일이 어느정도 실행이 되어지는지를 파악하기 위해 디버깅을 하고 싶었지만, ROS 디버깅을 몰라서 각각의 실행문 들 뒤에 printf 숫자를 달아서 어디까지 실행이 되는지를 파악했다. 파악하고 코드 수정하고를 반복하다가 계속 안되서 이날도 해결하지 못한채 해산했다.

4/11

오늘도 교수님 미팅전에 myRIO 파트인 조현근씨와 만나 LAB view 와 ROS 를 연결해 보려 하였다. 어제 두 개가 서로 다른 형식의 메시지를 주고받는단걸 알게 되었고, 오늘은 그걸 맞춰주기 위해 코드를 수정해보았다. 우선 LAB view 코드를 수정했는데 잘 되지 않았다. 그래서 다시 되돌리고 ROS 에 코드를 좀 추가시켜줬다. 그랬더니 조금 딜레이가 있지만 컨트롤러의 행동을 LAB view 에서 잘 받아들였다. 하지만 딜레이 때문에 명령이 곧바로 전달이 안됐다. 그래서 코드안에 들어있는 sleep 문의 간격을 수정해서 delay 를 없앴다. 창시구에서 Xbox controller 를 이용해서 하는 작업은 다 끝낸거 같고, LAB view 와의 협업도 대충 다 끝난 거 같다. 앞으로는 서로 각자 할 일 잘하면 잘 될것 같다.

이후에 교수님과 미팅을 하였는데 path generation 쪽에서 많은 피드백을 받았다. RPlidar 와 webcam 을 합치면 path generation 이 어떻게든 될거 같았는데 교수님 말씀을 들어보니 많은 개선이 필요해 보였다. 시작하고 나서 좌우로 이동하며 ball map 을 그리던, 카메라를 회전시켜 ball map 을 그리던 다양한 방법을 모색해 봐야 할 것 같다. 어떠한 방법으로 path generation 을 채택할지에 대해 선택하기 위해 우선 webcam 이 찍을 수 있는 반경을 파악하고, 그리고 실제 데모 장소인 기계동 1층에서 실험을 해보아야 할 것 같다는 생각이 들었다.

4/12

openCV 로 기계동 1층 로비에서 ball_detection 을 할 때 어떻게 되는지를 파악해 보기 위해 openCV 파트인 부준호씨와 기계동 1층에서 3시간정도 작업을 진행했다. 우선 개인 노트북으로 진행해보려 하였는데 내 노트북에서 openCV 가 알수없는 이유로 자꾸 실행이 되지않았다. 자꾸 있는 폴더가 없다고 나오면서 실행이 되지 않아 조교님들께 질문도 하고 갠톡으로도 질문했는데 다들 답변조차 해주시지 않아서 개인 노트북으로 해보지 못했다. 다음으로는 공학설계실에서 NUC 를 가져와서 실행해보았다. NUC 쓸 때 추가적인 모니터가 필요해서 이 또한 공학설계실에서 가져와서

진행했는데 모니터가 오래되서 NUC 과 연결되지 않아 결국 진행하지 못하고 시험끝나고 다시 해보기로 하였다.
기간 : 4/16 ~ 4/22

4/19

ROS 와 OPEN CV 파트의 ball detection 코드를 합쳐 보려 하였다. 일단은 두 개의 코드 모두 살려놓고 싶었기 때문에 새로운 package 파일을 하나 추가적으로 만들어 진행하였다. 우선 OPEN CV 의 ball detection code 에서 header 들을 ROS 의 ball detection code 로 다 옮겨주었다. 이후에는 우리가 궁극적으로 하고 싶은 것은 ROS 에서 OPEN CV 파트를 돌아가게 하는것이기 때문에 ROS 의 ball detection code 에 있는 code 들 중 msg publish 하는 부분이랑 main 함수를 제외하고 다 삭제시키고 그 자리에 OPEN CV 코드를 넣어주었다. 저 두 개만 남겨놓은 이유는 다른 부가적인 marker, imageshow 등과 같은 다양한 기능들이 code 안에 있는데 이 코드를 구현함으로써 궁극적으로 얻고싶은 정보는 msg 하나이기 때문에 저것을 제외하고 다 삭제시켰다. 그러고 난 후 catkin_make 로 컴파일 해주었는데 중복되는 CMakelist 를 사용하는 파일이 존재한다고 계속 에러가 떴다.

4/20

어제 하던 작업에서 중복되는 CMakelist 를 사용하는 파일이 있다고 나왔는데 이는 아마 원래 ROS 의 ball detection code 와 겹치는 것으로 생각이 되어진다. 그래서 아쉽지만 원래있던 코드를 지우고 다시 컴파일을 하니 컴파일은 되지만 원하는대로 실행이 되지 않았다. 코드 안에서 여러개를 계속 변경해 보았지만 의도대로 되지 않았다.

4/22

우선 path generation 먼저 생각해보기로 하였다. Final demo 의 condition 은 다음과 같다. 9*6m 맵에 흰 벽으로 테두리를 감싸고 있고 half line 을 지나고 나서부터 빨간공과 파란공이 배치되어있다. 그리고 각각의 공은 최소 50cm 씩 서로 거리를 두고 있다. 최종적으로 해야 할 일은 로봇의 시작점보다 20cm 뒤에 있는 바구니에 파란공 3개를 가져다 두면 된다.

우리 조는 webcam 을 이용해서 mapping 을 한 뒤 공을 주으러 다니기로 했다. 파란공만 주으면 되기 때문에 파란공의 정보만 받아도 가능할 것이라는 판단을 했다. 왜냐면 빨간공이 파란공을 주을때 방해가 되지 않는다고 생각됐기 때문이다. 하지만 만약 파란공과 빨간공이 일직선 상에 있는 경우 파란공을 주으러 가다가 빨간공과 부딪히게 되고 그 뒤 빨간공이 굴러가서 파란공을 부딪히고, 혹은 빨간공이 벽과 부딪히고 파란공과 부딪히는 등 다양한 변수들이 생길 수 있을거라 생각되고, 뿐만 아니라 이에 따른 시간 딜레이도 있을거라 생각되어 아예 빨간공을 부딪히지 않고 이동하는 방법을 고려하기로 했다.

이동 방식에는 처음에는 직선으로만 쭉쭉 움직이려 했다. 하지만 교수님의 피드백을 받고 직선으로만 다니는건 아마 불가능 할 것이라고 하셔서 이를 어떻게 해결할 까 고민중이다. 나온 생각들 중 가장 쓸만한건 수치해석에서 배운 것을 바탕으로 4점을 이어서 곡선으로 만드는 방법중 가장 경로 낭비를 안하는 방법이 뭐가 있을 지 고민을 하고 있었다. 하지만 로봇을 옆조에서 직접 구동시켜 보았는데 멈춰서 빙글 돌아서 방향전환하는 거를 우리가 받은 바퀴들이 상당히 잘 하는 것을 볼 수 있어서 직선으로 이동하는 것을 다시 채택하게 되었다.

그래서 다음과 같은 path generation 방법을 최종적으로 선택하게 되었다. 먼저 현재 우리조 OPENCV 가 4m 거리까지 ball 을 detect 할 수 있다. 그렇기 때문에 로봇이 우선 4.5m 앞으로 직진을 한다. 그 이후 혹시나 가려져서 보이지 않는 b all 이 있을 수 있기 때문에 half line 을 따라서 좌우로 이동하며 map 을 전체적으로 scan 하고 ball map 을 그린다. 그 다음 파란볼들을 직선으로 잇고 각각의 직선에 좌우로 13cm

씩(로봇의 가로길이의 절반) 떨어진 거리에 점선을 그어 그 영역안에 빨간 공이 들어오는지 체크를 한다. 그리고 만약에 빨간 공이 그 영역 안에 없으면 바로 이동하며 공을 줍고 바구니를 향해 직선운동하여 온다.

만약에 그 영역안에 빨간 공이 있는 경우 오른쪽 그림과 같이 빨간 공 주변에 추가적인 원 두개를 그린다. 각각의 원은 빨간 공으로부터 15cm, 28cm 떨어져 있다. 이는 로봇이 빨간 공 주변을 타고 돌때 다른 빨간 공이 들어오지 않는지 체크를 하기 위한 공인데 모든 공들이 최소 50cm 씩은 떨어져 있으니 이 방법으로 하면 빨간공을 건드리지 않고 파란공만 주을수 있다. 뒷일은 앞과 같이 행동한다.

기간 : 4/23 ~ 4/29

4/23

ROS 와 OPEN CV 합치는 작업을 다시 시도하였다. 과정 자체는 저번주에 진행했던 것과 같았다. 하지만 code 중간중간에 필요한 코드를 계속 추가시키면서 진행을 하였다. code 에서 marker, image show 등 불필요한(간단한 시각화를 위한 부분들) 부분들을 제거하고 통합시켜 주었다.

catkin_make 는 성공했지만 webcam node 와 같이 실행을 시키면 이미 webcam 을 사용중이라고 나오고, webcam 을 안키면 실행이 안됐다. 그래서 코드를 다시 확인해주니 open cv 코드 안에 자체적으로 webcam 을 구동시키는 코드가 내포되어져 있어 그것을 제거해주니 실행이 되었다.

여러개의 컴퓨터로 동시에 진행하다 보니 github 에서 수정된 코드말고 이전의 코드로 작업을 한 경우가 있어 화면의 사이즈가 480*640으로 나오며 segmented error 가 떴다. 이는 빠르게 수정해주었다.

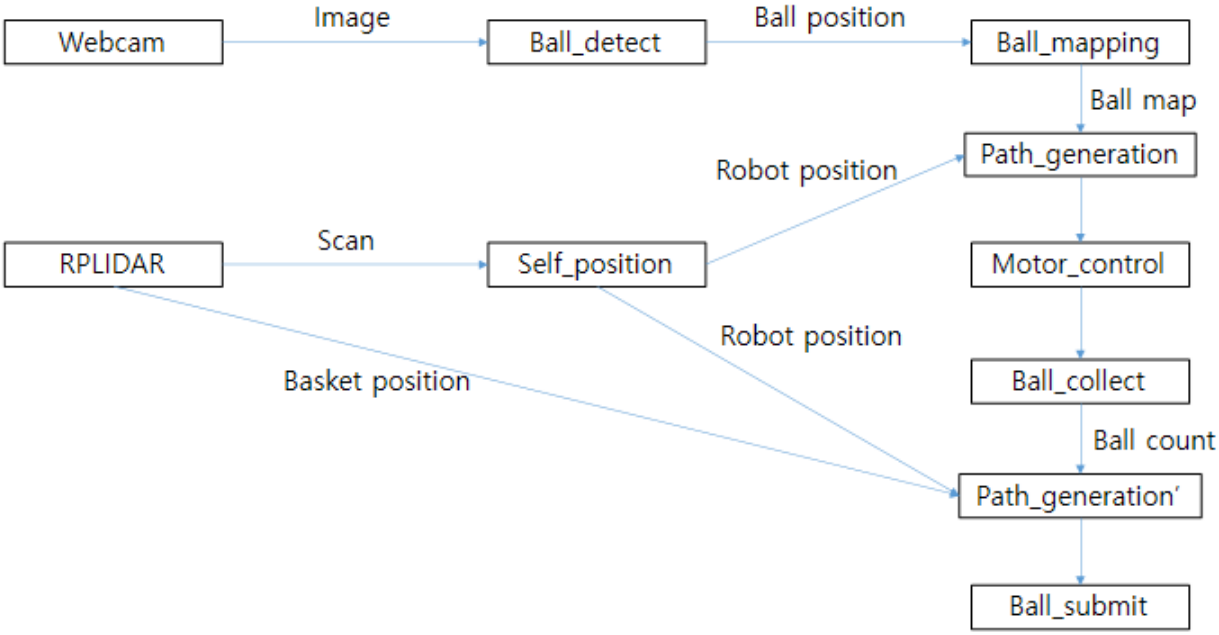
현재 publish 하는 msg 에 ball position 을 넣지 않고 아무거나 넣어놓고 컴파일하고 실행중인데 추후에 msg 에 blue ball position 과 red ball position 을 넣어줄 것이다.

4/25

오늘 교수님과 미팅을 진행하였다. 다른건 별다를 것 없었지만 교수님께서 전륜이나 후륜으로 바퀴 두개만 사용해보는것이 어떤지에 대해 제안해주셨다. 그래서 wheel kinematics 를 찾을 때 4륜으로 찾지 말고, 2륜으로 된 데이터를 찾아 대입해 보는것이 어떨지 고민하게 되었다. 이에 대한 수식은 ball detect node 의 다음 node 인 data integrate node 에 넣어서 마이리오로 데이터 전송하게 코딩을 하야 할 것 같다.

4/26

내일 있을 ROS TA SESSION 을 대비하여 전체 RQT graph 를 그려보았다.



4/27

TA SESSION 을 다녀온 뒤 많은 것이 바뀌게 되었다. 우선 SLAM 이 너무 부정확해서 RPLIDAR 를 제외하고 다시 알고리즘을 생각해봐야한다. 두번째로는 바구니를 RPLIDAR 로 detect 하려 했는데 바구니 양옆에 초록색 공을 붙여주신 다고 하셔서 opencv 에서 green ball detect 를 추가적으로 넣어주면 해결 될 것 같다. 또한 webcam 의 데이터 값들이 어느정도 부정확성을 띈다고 하셨다. 그렇기 때문에 현재 우리조의 ball mapping 이후 path generation 이 힘들게 되었다. 그리고 map 의 size 도 변화한다고 하셔서 그것이 나오면 또 webcam 위치, mechanism 등이 많이 변화할 것 같다.

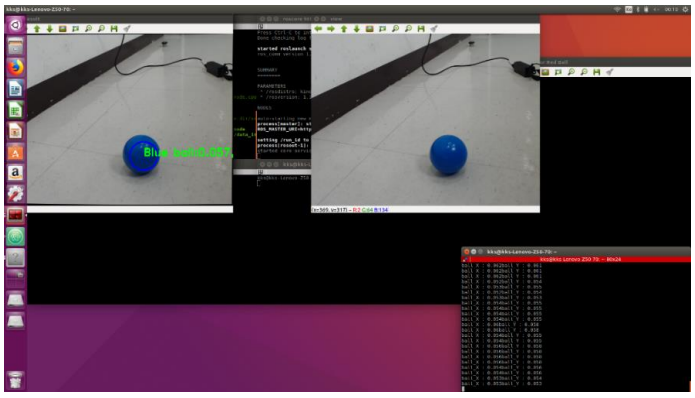
4/28

path generation 을 ball mapping 에서 ball tracking 으로 변경해주었다. 이에 대해서는 모든 조가 거의 통일 될 것 같다. 그리고 opencv 에서 green ball detect 를 추가하려 하였고, ROS 쪽에서 publish 하는 msg 에 blue ball position 이랑 green ball position 을 넣어주려 하였다.

기간 : 4/30 ~ 5/6

4/30

ROS 에서 ball_detect 코드를 바꿔주었다. 우선 open cv 쪽에서 아직 초록색 공이랑 파란색 공을 잘 구분하지 못하는 상태이기 때문에 파란공과 빨간공만 ball position 을 뽑아내는 코드로 수정을 해주었다. 그런데 현재 상황에서 내 생각에는 map 이 줄어들었기 때문에 파란공을 줍는 과정에서 빨간공을 줍는 경우가 거의 없다고 판단을 하여 전체 algorithm 의 시간 단축을 위하여 빨간 공의 ball position 을 뽑아내는 과정을 주석처리하였다. 그리고 전체 algorithm 은 이전과 같이 간단한 ball tracking 으로 하기로 하였다.



5/1

이번주 금요일이 발표이기 때문에 ROS 진도를 팍 빼기로 하였다. Ball detect 코드에 대해서는 이제 open cv 에서 도맡기로 하고 그 뒤 노드인 data integrate 를 손보기로 하였다.

Data integrate 노드에서 하는 일은 ball detect 노드에서 보내주는 ball position data 를 수용하는 노드인데 여기다가 전체 mobile platform 의 algorithm 을 추가하기로 계획했다. 랩뷰와의 socket 통신을 통해 mobile platform 이 어떻게 움직일지를 결정해주게끔 코딩을 진행하였다.

우선 labview 파트분들에게 socket 통신에 대해 설명을 듣고 오늘은 파란공이 mobile platform 의 정면 한중앙에 올때까지 mobile platform 이 회전하면서 공을 찾고, 그 뒤 공이 mobile platform 의 collector incline slope 직전까지 올때까지 움직이게끔 코딩을 하였다.

그리고 open cv 파트와 함께 webcam 의 이상적인 높이를 계산해보았다. 이전에는 webcam 의 진동과 충분한 시야, 두가지를 고려하여 미분방정식을 세우둔, 다른 수식적인 방법을 통해 높이를 찾아보려 하였다. 그런데 mobile platform 구동 결과 진동이 고려할 만큼 크지는 않았기 때문에 충분한 시야만을 고려하여 webcam 의 이상적인 높이를 찾아보았다. 조건을 worst case 에 대해 webcam 의 시야가 collector incline slope 의 바로 앞부분부터 시작되고 두개의 파란공을 detect 할때 두개가 겹쳐 보이지 않게한다로 두고 계산을 하였다. Worst case 는 mobile platform 이 map 의 중앙에 있고 파란 공 두개가 한쪽 꼭짓점 그리고 거기서 50cm 떨어져있는 상황으로 설정하였다. 그 결과 webcam 을 지상으로부터 34.5cm, 그리고 각도는 지상을 0도라고 했을때 57도로 설정해야 한다는 결론을 얻었다.

5/3

내일이 발표기 때문에 발표자료를 검토해보고 필요한 영상을 촬영하였다. 우선 발표자료를 검토하는 과정으로는 전체흐름을 파악하고 분량에 대해 10분안에 충분히 가능할지를 판단 한 뒤 평가항목에 있지만 우리 발표자료에 빠진 부분을 보충하는 방향으로 진행하였다.

전체적으로 봤을때 현재 열관리 문제에 대한 자료, 그리고 mobile platform 의 움직임과 vibration management 에 관한 영상이 부족하다고 판단이 되었다.

먼저 열관리 문제에 대해서는 현재 우리조의 converter 가 spec 부족으로 사용하지 못하고 새로운 converter 를 주문하였는데 사용하고 있다는 가정하에 진행하였다. 현재 구동했을때 가장 뜨거워지는 PMS 에 대류가 잘 되게끔 하기 위해 전체 mobile platform 의 뼈대를 이층으로 하고 층사이의 거리를 벌려두었다. converter 에는 방열판을 달고 mobile platform 의 두 층 사이 틈에 PMS 를 설치하고 뒷부분에 fan 을 달아 대류가 잘되게끔 배치하기로 하였다.

Vibration management 는 우리조의 경우 구동할때 생기는 진동이 고려할 만큼의 양이 안된다고 판단을 하여 진동이 많이 생기지 않는다는 영상을 촬영하였다.

그리고 마지막으로 mobile platform 의 움직임에 대한 영상으로는 필요한 노드 다 실행시켰을때 자동으로 돌아가는 영상을 촬영하고, 또 collector 로 공을 수거하는 영상을 찍고 싶었는데 현재 sub motor 가 오지 않아서 그 부분에 대해서는 손으로 촬영했다.

기간 : 5/7 ~ 5/13

5/8

발표가 끝난 뒤 어린이날을 포함한 긴 연휴가 있어서 좀 쉬었다.

연휴가 끝난 뒤 첫 조모임을 가졌다. 일단 저번주 금요일날 발표가 있었기 때문에 발표에 대한 feedback 을 해보았다.

ROS 와 관련된 자체 피드백으로는 이전까지는 빨간공의 존재를 크게 신경을 쓰지 않았는데 다른 조들의 경우 이를 많이 고려하고 있기 때문에 다시 한번 생각해보자, 우리조는 현재 공을 주을때 앞쪽 시야를

조금 남겨두고 공을 주을때 공이 시야에서 스르르 사라지는 구조이기 때문에 이를 이용하여 공 개수를 counting 할 수 있을거 같은지 로 크게 두가지 정도 고려해 볼만 한 것 같았다.

먼저 첫번째 피드백을 통해 전체 algorithm 을 다시 생각해보게 되었다. 현재는 빨간 공을 고려하지 않는데 혹시나 모를 경우에 대해 고려하기로 하였다. 다른 조들의 발표를 바라본 결과 현재 다른조들이 여기서 시간을 상당히 낭비하고 있는것 같아 여기서 창의성을 많이 갈아넣어서 다른조와의 격차를 벌려야 할 것 같다. 하지만 아직 마땅히 방법이 떠오르지는 않는다.

두번째 피드백을 통해 공을 주을때 카메라에 잡히는 공의 모션으로 공 개수를 카운팅하자는 피드백인데 현재는 mobile platform 이 전체 map 을 어디서든 바라볼 수 있기때문에 공 3개를 다 주웠다고 생각하는 시점에서 한바퀴 회전하면서 map 에 파란공이 남았나 파악을 하려했는데 이 두가지중 창의성과 시간을 고려해서 무엇을 할지는 향후 test drive 를 하면서 결정해야 할 것 같다.

그리고 다음으로는 test drive 를 5월 18일부터 하며 계속 수정을 하기 위해 그 전의 모든 단계를 17일까지 완료하기로 하였다.

5/9

교수님과 미팅을 하고 나서 많은 문제를 깨달았다.

오늘은 ROS 부분에 대한 커멘트 보다는 collecting system 에 대해 많은 지적을 받아 이에 대해 미팅이 끝난뒤 많이 생각해보았다.

우선 현재 바퀴를 돌리는 drive motor 로 collector 를 돌려보았을때 arm 과 slope, 그리고 ball 사이의 impact 때문에 motor 가 정상적인 기작이 되지 않았다. 그런데 현재 우리가 추가로 주문한 모터가 어떤 종류의 모터인지는 모르지만 최대 토크가 drive motor 보다 작기 때문에 현재 collector 의 외형을 바꿀 필요가 있다고 판단되었다.

그 결과 현재는 앞쪽 프로파일이 배터리 부분앞으로도 쭉 튀어나와있고 그 부분에 storage incline slope 랑 collector incline slope 가 달려있었는데 이런 구조이기 때문에 자연스레 slope 의 경사가 커져 이를 해결하기 위해 튀어나온 프로파일을 잘라내고 아래쪽으로 프로파일을 새로 연결해 지금 collector system 을 compact 시켜 달기로 하였다. 또 현재 그냥 linear 한 경사인데 이거를 타원형, 혹은 사이클로이드 곡선과 같이 변경하여 공이 처음 들어올때의 경사각을 줄이고, 또 사포로 경사를 열심히 갈아서 마찰을 줄이기로 하였다.

collector 의 경우 계속해서 RP 를 뽑아가면서 test 해보고 계속 improve 시켜나가야 할 것이다.

5/11

앞서 현재의 algorithm 에서는 빨간공을 고려하지 않기 때문에 여러 문제가 파생될 수 있다고 판단이 되어 어떻게 개선해야 될지에 대해 고려하였다.

다른 조의 algorithm 을 간단하게 말하면 크게 보면 우리조와 같다. 파란공을 하나씩 찾아다니면서 공을 3개 줍고 초록 공을 detect 해서 바구니로 와 공을 떨어트리는 시스템이다. 여기서 우리조와 다른 점은 빨간공과 부딪힐 경우를 고려하여 로봇의 시야에 빨간공이 들어오면 빨간공이 시야에서 사라질때까지 로봇이 삐뚤삐뚤 움직이면서 시야에서 빨간공을 제거하는 방식이다. 하지만 현재 삐뚤삐뚤 이동하며 빨간공을 피하는 과정에서 상당히 많은 시간이 소모되고 있다. 그래서 이를 개선하는 방식이 필요하다고 판단하였다.

첫번째로 생각한 방법은 먼저 파란공을 detect 한뒤 내 시야안에 빨간 공이 파란공보다 앞에 있는지를 판단한다. 좀더 정확하게 말하자면 시야 중에서도 mobile platform 이 이동하는 경로안에 있는지를 파악한다. 파악한 뒤 mobile platform 의 경로 안에 있다면 파란공을 중심으로 한 큰 원을 그리며 빨간 공이 mobile platform 의 이동경로에서 벗어나게끔 한 뒤 직진을 하는 방법이다. 하지만 이런 방법의 경우 원을 돌다가 mobile platform 이 다른 공을 칠 수 있을것 같았다. 그래서 생각한 방법이 두번째 방법이다. 이 방법의 앞 부분은 앞서 말한 방법과 같다. 일단 이동경로에 빨간공이 오는지 파악한다. 그 뒤 만약 그렇다면 빨간공에 로봇의 가로길이 만한 반지름을 가지는 원을 그리고 이 원의 접선을 타고 돌며 다음 파란공으로 이동하는 방식이다.

기간 : 5/14 ~ 5/20

5/14

mobile platform 이 공을 어떻게 빼낼지에 대해 고려를 해보았다.

현재 mobile platform 은 collecting 과 submitting 에 대해 각각 하나의 모터를 사용하고 있다. 다른조들의 경우 대부분이 mobile platform 의 앞부분에서 공을 collecting 하고 뒷부분으로 submitting 을 하고 있다. 반면 우리조의 mobile platform 의 경우 앞부분에서 공을 collecting 하는 것 까지는 같지만 뒷부분이 아닌 옆부분으로 공을 submitting 하기 때문에 두 개의 모터가 비교적 가까운 거리에 위치했다.

비교적 가까이에 있지만 조금 거리가 있기에 그냥 그대로 사용할 까 했는데 교수님께서 가까이 있으니까 하나로 합치라 하셔서 열쇠를 모방한 mechanism 을 구현하기로 하였다.

현재 공을 collecting 하는 모터의 윗부분에 key 를 달고 가까운 거리에 holder 를 달아서 공을 collecting 할 때는 서로 맞물리지 않다가 arm 이 반대로 회전하면 두개가 맞물려서 door 가 열리는 구조를 만들었다.

하지만 문제가 door 의 initial position 을 고정하지 못하여 이를 고정할 필요가 있었는데 이는 자석을 side 에 배치하고 platform 에 이에 맞는 다른 자석을 붙여서 고정하기로 계획하여 3D printing 을 맡겼다.

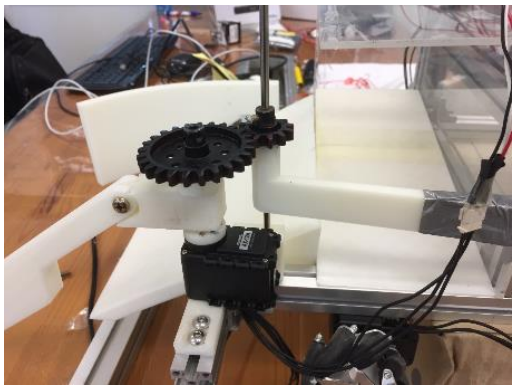
5/15

mobile platform 의 submitting system 이 3D printing 되어 나왔다.

자석과 3D printing 된 element 들을 조립해보니 생각한 대로 잘 작동이 되었다.

하지만 현재 key 와 holder 파트들을 기어로 연결해 놓았는데 이 기어가 빨간색이라서 mobile platform 의 전체 algorithm 을 돌렸을때 vision 에서 이 기어를 빨간 공으로 인식하는 문제가 생겼다.

뿐만 아니라 자석의 힘 때문에 공을 뱉어낼때 축이 흔들리는 문제도 발생했다.



5/16

교수님과의 면담이 있었다.

면담 이후 창시구 실로 돌아와서 전체 mobile platform 의 algorithm 을 돌려보았다. 현재 algorithm 은 빨간공은 피하고 파란공만 줍는 것 까지 짜져있었다. 이를 실행해보니 현재 빨간공을 피하는 걸 거리에 따라 5개로 나눠놨는데 안되는 경우가 있어서 변수 값들을 조정해주었다.

또 공을 collecting 은 확실하게 하는데 pick 을 하는 횟수가 한번만에 되지 않는 경우가 다수로 일어났다. 현재 slope, arm 그리고 공 3개의 마찰때문에 한번에 올라오지 않았다. 그래서 해결책으로 slope 를 길이를 늘리고 경사를 줄이기, slope 에 셀로판지 붙이기, arm 의 형태를 바꿔 힘의 중심이 아래로 오게 하기 등 여러 해결책을 세워놓았다.

5/17

잡화점에 들러 무광흑 스프레이, 셀로판지, 본드등을 샀다.

먼저 무광흑 스프레이로 submitting system 의 빨간색 기어들을 색칠해주었다. 그 결과 vision 쪽에서 아무 문제 없이 공들을 잘 detect 할 수 있게 되었다.

두 번째로 셀로판지로 slope 를 감싸려 했는데 이전에 사용하던 길지만 경사가 낮은 slope 르 끼워보니 잘 되서 사용하지 않기로 하였다.

마지막으로 공을 submit 할 때 door 의 축이 흔들리는 문제는 key 에 붙어있는 기어에 축을 달고 그 축과 holder 의 축을 연결하는 bar 를 하나 달아서 흔들림을 제거해주었다.

이후 바뀐 mobile platform 의 hardware 들을 가지고 전체 algorithm 이 잘돌아가는지 확인을 해보았다. 그 결과 파란 공을 3개 줍는것까지의 과정이 잘 진행되고 있음을 확인할 수 있었다.

기간 : 5/21 ~ 5/27

5/22

mobile platform 의 바퀴 중 하나가 헛 돌고 있어서 mobile platform 의 motion 이 원하는대로 잘 이루어지지 않고 어느정도 error 가 생기면서 이동을 하였다. 그래서 이를 해결하기 위해 mobile platform 의 바퀴를 떼어보았다. 보니 dynamic cell 을 고정하는 고정장치는 잘 고정이 되어있었고, 반면 mechanum wheel 과 dynamic cell 의 연결부인 플라스틱 부분이 마모되어져 있었다. 그래서 이를 대채해줄 것이 필요했다.

바퀴 하나당 하나의 허브가 주어졌다. 허브가 알루미늄 재질이라 바퀴와 연결해도 마모가 되지않기 때문에 이를 플라스틱 대신 대체하기로 했다. 하지만 허브가 dynamic cell 을 direct 하게 연결할 수 없기 때문에 이 두개를 이어줄 마운틴을 구매하여야 했다.

5/23

교수님과의 면담이 있었다. 면담 결과 교수님이 꽤 괜찮다고 말씀해주셔서 계속 데모 연습을 해보면서 생기는 문제들을 해결하기로 하였다.

5/24

mobile platform 이 basket 을 찾아가서 주차를 하고 공을 뱉어야 하기 때문에 우선 basket 을 어떻게 찾아갈지에 대해 고민해야 했다.

큰 algorithm 은 두개의 초록공을 확인하고 두개의 좌표를 확인한 뒤 두개의 중점을 향해 이동하고 어느정도 거리에 다다르면 mobile platform 이 그 자리에서 회전을 하여 두 초록공의 y 좌표가 같아질 때까지 motion 을 취하고, 이후에 90도 회전하고 옆으로 병진운동하여 basket 에 다가가는 흐름이다.

이 과정을 수행할 수 있는지를 확인하기 위해 vision 에서 얼마나 정확한 정보를 전송하는지를 파악할 필요가 있었다. 확인을 해보니 vision 에서 조금 문제가 있어서 해결을 해주었다.

창시구실에서 구매한 마운틴을 이용하여 허브와 연결을 해보려 하였는데 해외구매라서 데모 끝나고 나서 도착한다고 하여 자체제작을 solidworks 로 해주었다.

5/25

저녁 8시에 데모 리허설이 있어서 오후 2시에 창시구실에 가서 자체제작한 마운틴을 연결해보려 하였다.

시중에 파는 마운틴과 같은 규격으로 제작을 하였으나 사이즈가 맞지 않아서 다시 mechanum wheel 과 허브의 규격을 재서 만들었다.

3번 정도의 시행착오를 거쳐 마운틴을 만들었고, 바퀴와 연결을 해주었는데 시간이 많이 들레이 되고 모터에 이상이 생겨서 실질적인 데모를 하지 못했다.

5/26

mx 모터가 고장이 났어서 여분의 mx 모터를 받아서 다시 연결을 해주고, 바퀴가 흔들리는 문제들을 다 공작실 들어가서 수정해주었다.

이후 모터와 PMS 를 연결하는 몇몇 케이블에 문제가 있음을 발견하고 이를 수정해주었다.

하드웨어가 다 완성이 되고 basket 으로 주차하는 code 를 시행해봤는데 잘 진행이 되었고 주차이후에 병진과 회전운동이 섞여 basket 에 딱 들어붙게 되는 code 까지 완성했다.

이제 내일 데모에서 안되는거만 확인해서 고치기만 하면 될 것 같다.