2019 Fall

Capstone Design 2

Final report

Team name: 창시구실로 롤러가자

Team professor: 윤국진 교수님

Teaching assistant: 김동현

Member: Farhad Hasanli, Zulfiqar Nasir, 박성혁,

박채리, 오준영, 이승언, 장민석, 조현태

Submission date: 20th Dec 2019

Contents

1. Individual Expressions

Farhad Hasanli

Capstone Design II was a very tough and challenging task. However, it was an extremely valuable experience for me since I got two use the engineering principles I have learned throughout the years at KAIST. It also taught me to be a good team player and handle things in high-pressure environments

Zulfiqar Nasir

Being one of the most prestigious course of Mechanical Department, the experience lived up exactly to my expectations. The fact that the main goal of this course matched somewhat with Capstone design I made it easier to adapt with the dynamics of this course. However, despite that the course was a 'Real Challenge' as the modifications introduced in this course rendered our initial Capstone design I robot unfit for the task and the robot had to designed from scratch. In this courses, I learned how important good planning and team work can be in group projects. Instead of separate people in a group working separately on distinct parts of the robot alone, a better strategy is when there is continuous network of mutual feedback from all the teammates and in cases when one teammate is having hard time with his assigned role, other teammates should help him out as that is what being in a team is about. This the most important thing I learned in this course and think is the reason of our success.
Personally, the biggest hurdle in this course was designing the suspension of our robot. Suspension design was the most important concept I got to learn and we were quite successful in that regards.

박성혁

예비군 훈련이나 건강상의 문제로 소홀히 했던 부분도 있었는데 책망보다는 걱정해주는 팀원들이 너무 고마웠다. 특히 조장을 맡아주었던 민석이에게 가장 미안하고 고마운 감정을 느낀다. 조장이 작업으로 불렀을 때 어영부영 하다가도 막상 가서 작업을 할 때마다 즐거움을 느꼈다. 항상 창시구실에서 고생하는 모습도 많이 보았지만 직접 내 눈으로 보지 못했던 팀원들도 각자 보이지 않는 곳에서 노력했음을 잘 알고 고맙다. 끝으로 마지막 시연 때 성공/실패의 유무와 상관없이 각 로봇이 움직이는 모습을 보았을 때 같이 수업을 들은 학생들의 한 학기동안 고생과 간절함을 공감할 수 있었고 뿌듯했다.

박채리

한 학기동안 같이 열심히 해준 팀원들에게 고맙고 덕분에 좋은 결과로 마무리할 수 있었던 것 같습니다.

오준영

이번 창시구 2 에서는 소프트웨어를 담당하였었는데 크게 3 가지, CNN, SLAM, DQN 에 대해 다뤘었다. 평소에 궁금하였던 deep learning 에 대해 조금이나마 알아볼 수 있었고 창시구 1 에 비해 task 가 매우 어려웠고 투자한

시간도 컸지만 이를 성공시킴으로써 얻는 성취감도 엄청났다. 창시구 1 이 끝나고 많이 고민했었는데 창시구 2 를 수강한건 후회 없는 선택이 된 것 같다.

이승언

카이스트에 입학하고 이렇게 열심히 산적이 있나 싶을 정도로 창시구 2 의 일은 정말 많고 힘들었습니다. 하지만 좋은 성적으로 끝내고 나니 이보다 더 뿌듯할 수 없었습니다. 같이 열심히 해 준 조원들과 교수님 그리고 조교님께 정말 감사합니다.

장민석

창시구 1 을 하면서 고생을 너무 해서 창시구 2 를 들을지 말지 고민을 많이 했었다. 고민 끝에 창시구 2 를 수강하게 되었는데 창시구 1 에서 배운 노하우들이 있어 설계를 하면서 발생하는 문제들을 신속하게 해결할 수 있어 효율적인 설계를 할 수 있었다. 그리고 혼자 해결할 수 없는 문제들을 조원들과 상의하며 발전해 나갈 수 있어 조원들의 역할이 매우 중요했다고 생각한다. 한 학기동안 고생한 팀원들과 모든 조교님들, 그리고 교수님들께 감사의 말을 전하고 싶다.

조현태

창시구 2 는 정말 재밌습니다.
다들 창시구 하세요

2. System Design

Week4:

1) 박성혁

지난 기간 동안 설계팀에서는 픽업 모듈과 릴리즈 모듈의 구체적인 모습을 디자인하였고, 창시구1에서 사용했던 롤러를 다시 사용하기 결정하였다. 창시구1과 달리 빨간 공과 파란 공을 각각 집어 분류해야 하는 과정이 추가되었기 때문에 픽업 모듈도 그런 부분에서 변화가 생겼다. 기본적으로는 빨간 공과 파란 공을 집는 구역을 나누었고 그 이후에 sorting bar로 한번 더 분류하는 방식을 채택하였기에 기존에 만들었던 롤러보다 약 2배 긴 디자인을 채택하게 되었다. 현재 pick up module의 파트는 3d 프린터로 제작되어있는 상태이다.

기어박스와 서스펜션 등 actuator module의 경우 선택한 기존 로봇의 것을 거의 그대로 사용하기로 정하였다. (로봇 선택의 기준이 이런 actuator module이 잘 된 것을 선택) 서스펜션은 응력의 문제도 존재하지만, 적어도 바퀴 4개가 고르게 지면에 접하기 용이하다는 점에서 충분히 도입된다는 판단을 하였다. 설계팀 내부에선 각자 파트를 나누어서 진행하기로 하였는데 내가 수행하기로 한 역할은 배터리, 라이더, 카메라등 다른 제품들을 로봇에 설치하는 역할이다.

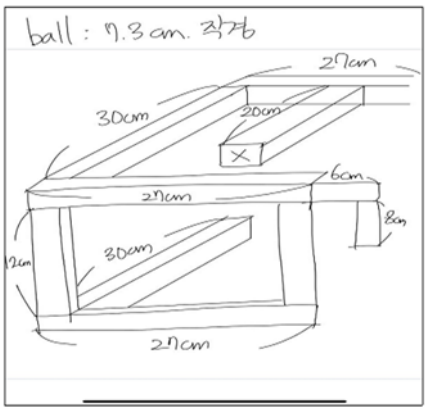


Figure 1

이러한 구조의 프레임을 사용할 예정인데 1층은 보관 및 분류에 사용되는 공간이라 여유공간이 없어 2층위로 적재하는 형식으로 설치하기로 하였다. 또한 다음과 같은 형태의 설치판을 새로 만들 로봇에 적합하게 바꾸어 만들기로 정하였다.

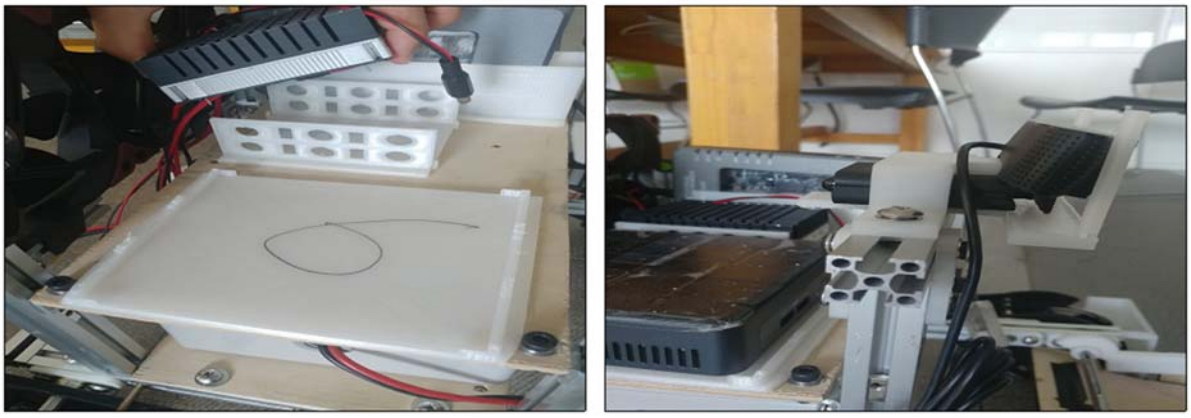


Figure 2

다음주 내에 관련 프린팅을 문의하고 구동부 및 픽업 모듈의 다이나미셀 설정을 할 예정이다.

2) 조현태

In this first week, since I was able to perform many intermediate skills in solidworks, because I studied modeling in Mechanics Design class, I worked hard to design many types of prototypes, and shared ideas with my peers. I'll show you many photos of my design.

Software	Hardware
오준영	조현태
이승언(Labview)	박성혁
박채리(openCV)	Zulfiqar
Farhad	장민석
장민석	

We first divided our roles in software team and hardware team.

And we made a brief schedule for the overall course.

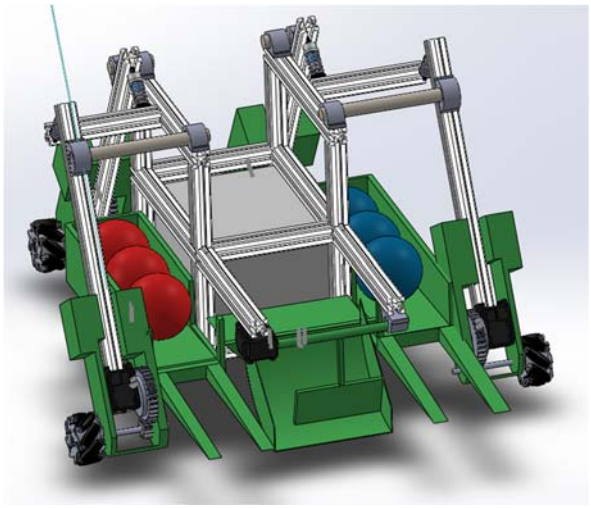


Figure 3

And this solidworks design was discarded because it was big and heavy.

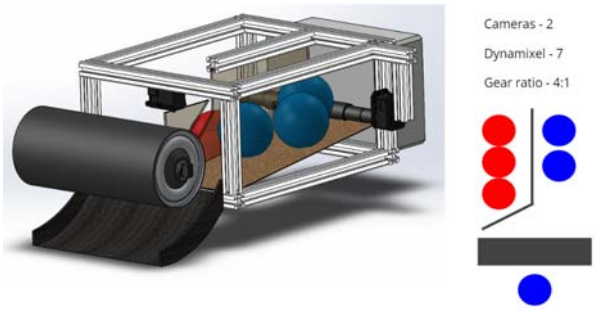


Figure 4

Figure 5

We will implement the gearbox and gear-ratio from the previous team.

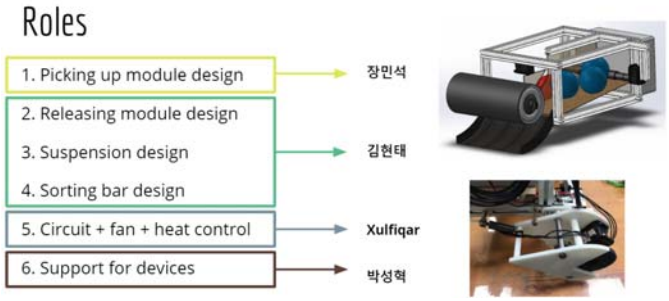


Figure 6

And we assigned our specific duties and obligations in order to avoid any free-riders.

3) Zulfiqar

Our group was divided into two main groups: hardware and software. I am part of the hardware group which is concerned with design, structure, circuitry and all physical specifications of our robot.

- Design of our Robot

The most important part of the design we had to consider was how to design our ball pickup, sorting and release mechanism. We had a group meeting in which we concluded to use the roller and ramp mechanism for pickup, sorting mechanism using a horizontal rotating gate and two separate pathways with the base as slack cloths to store separate balls. The releasing mechanism is such that the clothes are connected to a motor and when the motor spins the cloth becomes taut and the balls are released.

- Assignment of Roles

The roles of the Hardware part were divided into four categories:

- Pickup module Solidworks
- releasing module + suspension + sorting gate
- Circuitry + Heat analysis
- Assembly

I am in charge of the circuitry of our whole system and its heat analysis part. First of all, I listed down all the electrical components of our system and their electrical ratings :

1) Battery Output Voltage : 21.6 V

2) Camera x2 Input Voltage : 5 V

3) Motor x7 Input Voltage : 12 V

4) NUC Input Voltage : 19 V

5) myRIO Input Voltage : 12 V

6) GPU Input Voltage : 19 V

7) Fan x2 Input Voltage : 12 V

8) LIDAR Input Voltage : 5 V

9) Converters

In designing the circuit I have to consider the conservation of energy such that the energy input from each converter should be less than the energy consumed by the components. An Example of the circuit design is shown below :

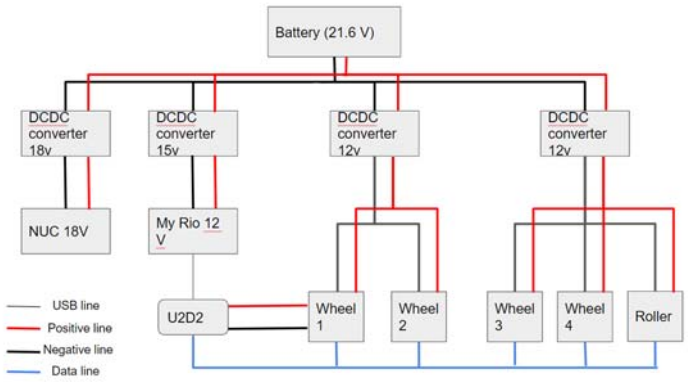


Figure 7

4) 장민석

지난주에 픽업 모듈을 정했다. <Figure 8>에서 볼 수 있듯이 롤러로 공을 경사면 위로 올리면 공은 경사면을 따라 로봇 안으로 들어간다. 이때 로봇 내부로 이어지는 갈색 경사면은 천 소재이다. 경사면으로 굴러가면서 다이나믹셀은 빨간공과 파란공을 분류한다. 공을 모두 주운 후, 공을 내보낼 때는 로봇 뒤에 달린 다이나믹셀로 천을 당긴다. 당겨진 천은 아래 그림에서 볼 수 있듯이 롤러쪽으로 경사를 만든다. 그리고 롤러를 반대쪽으로 회전시키면 공이 로봇 밖으로 빠져나간다.

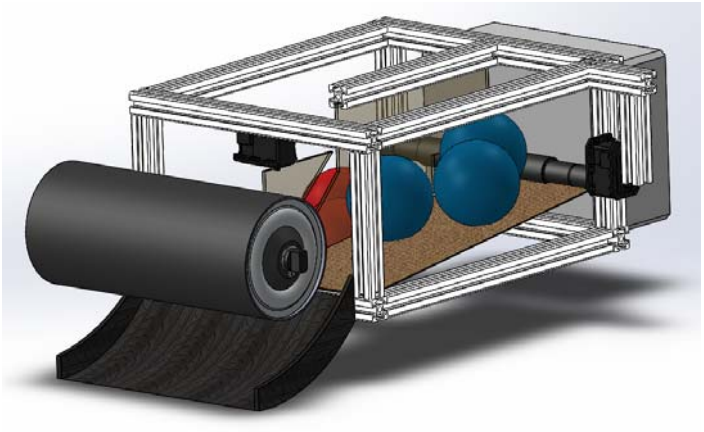
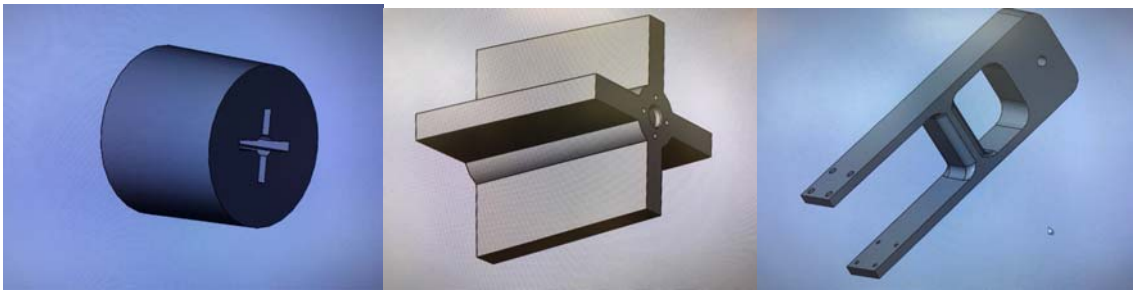
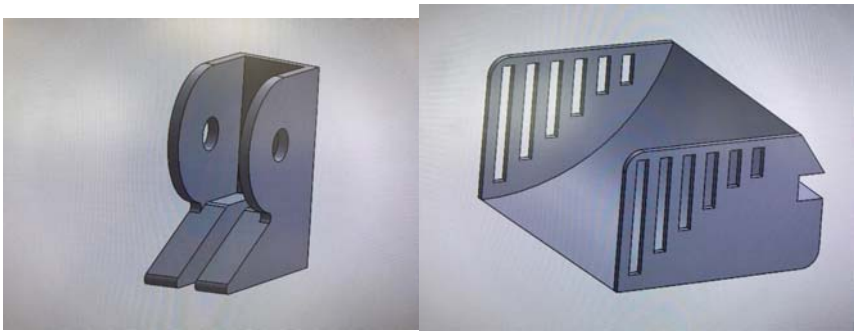


Figure 8

우리 조 하드웨어 팀은 총 4명으로 구성되어있으며, 각각 역할 배정을 하였다. 그 중 나는 픽업 모듈 제작을 맡았다. 그래서 아래에서 볼 수 있는 부품들을 제작하였다.





위에서 고안한 부품들을 3D 프린터로 출력하였고, 릴리즈 모듈을 맡은 조원과 아래 < Figure 9>에서 볼 수 있는 로봇을 조립하였다. 서스펜션은 지난 학기에 사용했던 로봇에서 그대로 떼어왔고, 나머지 부품은 새롭게 고안하였다.

Figure 9

하드웨어 팀에서는 10월 9일까지 하드웨어를 완성하려고 한다. 그래서 소프트웨어 팀이 로봇으로 더 많이 코드를 실험해보는 것이 목표이다. 다음 주까지 하드웨어 팀에서는 살짝 잘못 출력된 부품들을 수정하여 다시 조립해볼 것이고, 회로를 완성하여 로봇을 구동시킬 계획이다.

Week5:

1) 박성혁

금주에는 라이더, 그래픽카드, 배터리, 마이리오, nuc에 맞는 3d프린팅을 시도하였으나 생각보다 원활이 나오지 않아 시행착오를 겪었다. 공휴일도 껴 있어서 일단은 기존의 설치판을 분리하여 활용하기로 선택하였다.

랩뷰 부분에서는 기존 창시구1 때 사용하던 랩뷰 코드를 현 기기에 맞게 모터ID를 수정하여 소프트웨어 팀에서 슬램에 대한 시도를 도와주는 성과가 있었다.

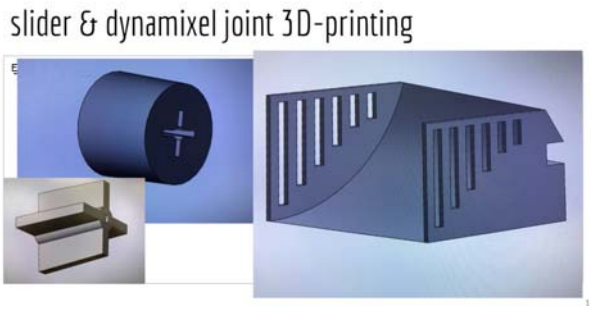
바퀴4개 구동도 문제없었고, 롤러(픽업모듈)도 그대로 사용하기 때문에 속도값 수정말고는 큰 변화가 없었다.

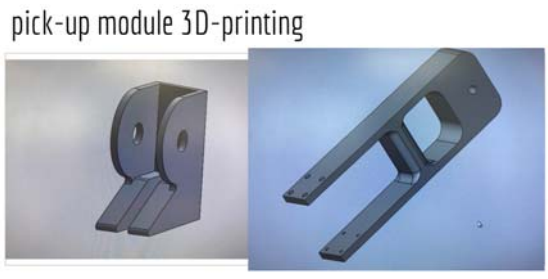
다만 문제점이라면 창시구 1에서는 조이스틱의 신호를 그대로 ros에서 입력하는 형식으로 코드를 짜 놓았던 것인데 기기의 움직임이 정해져 있다.(예시: 조이스틱을 위로 꺽었을 때 전방으로 1m/s의 속력으로 감, x키를 눌렀을 때 롤러가 시계방향으로 일정속도 회전)

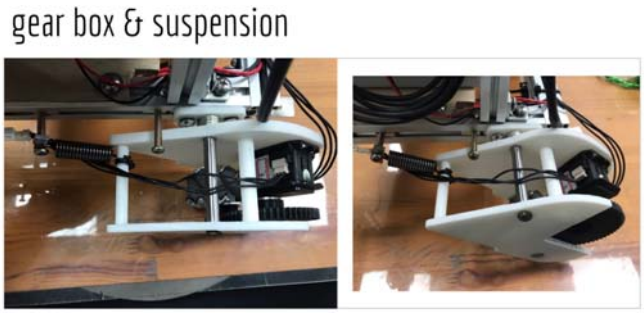
현재 소프트웨어 팀에서 슬램을 실험하는 것에는 지장이 없으나, 제대로 하기 위해서는 바퀴구동부에서 임의의 속도 값을 입력하였을 때 모터에서 구현할 수 있는 방식이 필요하다고 말하였다.

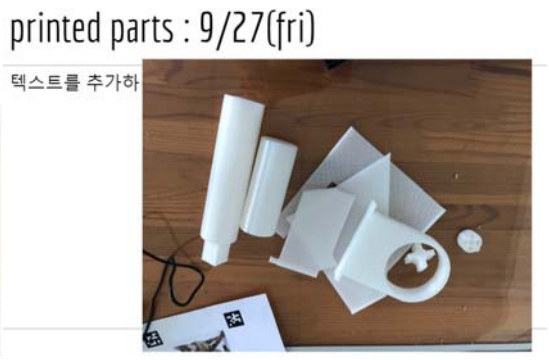
또한 모든 다이나믹셀을 바퀴 모드로 사용하고 있었는데, 솔팅과 릴리즈 부분에서 각도의 제한이 있는 회전 모드를 사용해야 하는 과제가 있다.

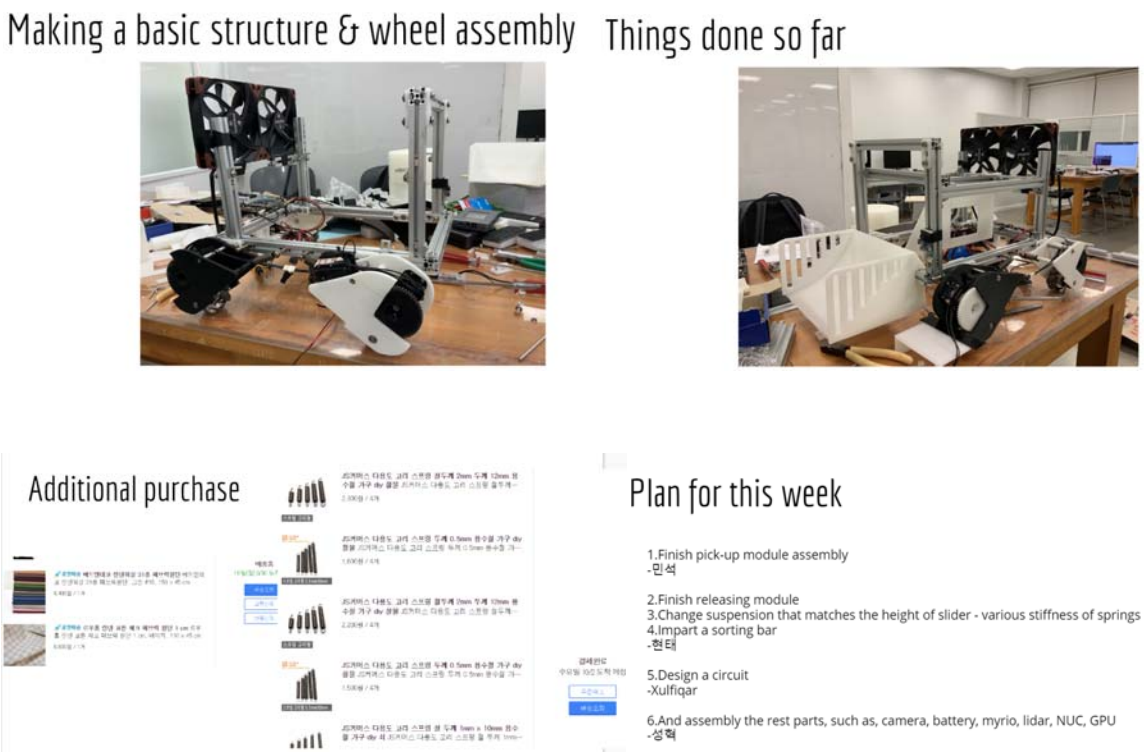
2) 조현태











3) Zulfiqar

(1) Design of Circuit

After specifying the Power input and the Electric rating of each electrical part in our system, I completed with the design of the circuit with the help of conservation of energy principle. Circuit schematic :

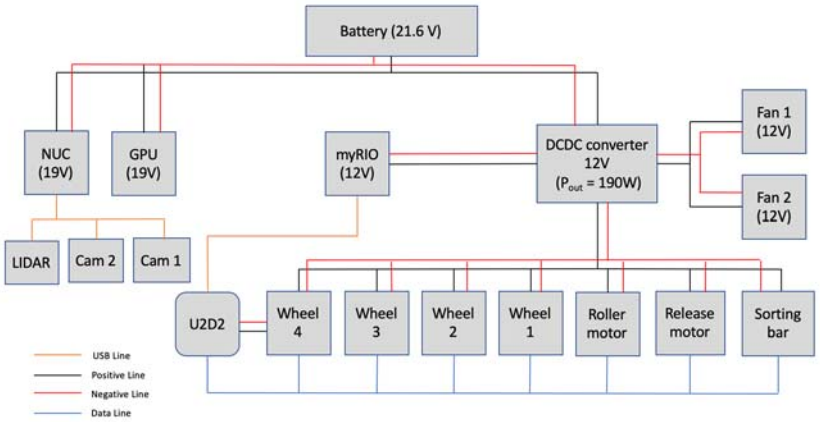


Figure 10

* We bought an expensive DCDC converter which has an high power output of 190 W and the total power required of each electrical component to it is 130 W (max).

(2) Implementing the Circuit into our System

I finished completing the circuit of our system as shown in the pictures below, tested it and the circuit is working fine :

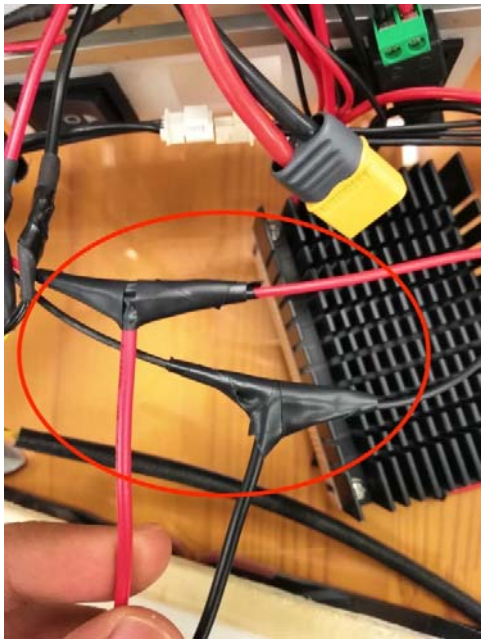


Figure 11. T wire joint installed for NUC and GPU

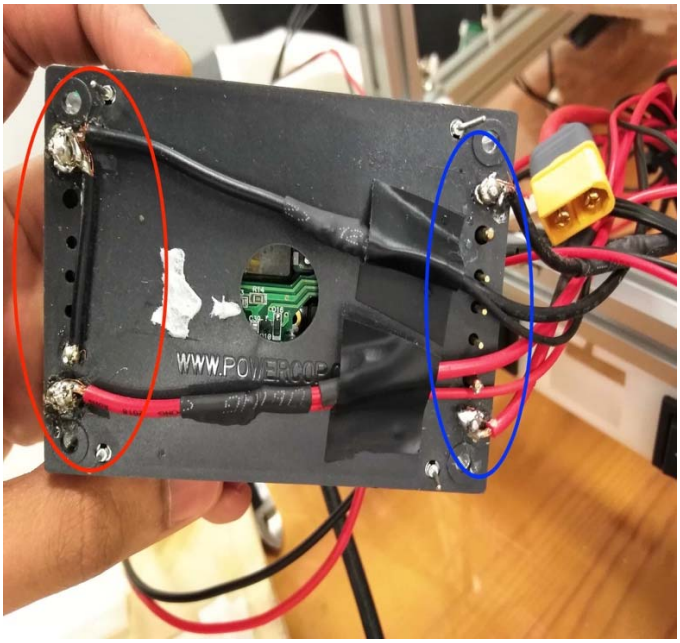


Figure 12. DCDC Converter battery input(red), Output(blue)

4) 장민석

지난주에 출력한 롤러와 램프에서 치명적인 문제가 발생하였다. <Figure 13>에서는 공이 램프와 롤러 사이에 잘 맞물려 올라가는 경우이다. 하지만 이 높이에서는 바닥에 있는 공과 롤러가 서로 만나지 못해 공을 픽업할 수 없었다. <Figure 14>에서는 롤러를 바닥에 있는 공과 닿을 정도로 내린 것이다. 하지만 이 상황에서는 롤러와 램프 사이의 거리가 너무 멀어 공이 램프 윗부분까지 올라가지 못했다. 그래서 이 문제를 해결하기 위해 램프를 다시 설계하였다.

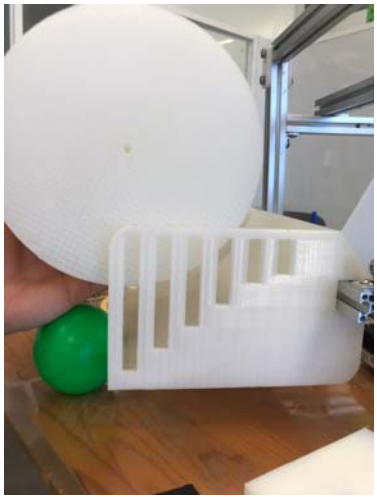


Figure 13

Figure 14

Figure 15

<Figure 15>에서는 롤러의 위치를 롤러암으로 고정하고 새로 디자인한 램프로 픽업이 되는지 확인해보았다. 다행히도 픽업이 잘 되었다. 현재 롤러의 무게는 1.3kg이다. 무게를 낮추고자 롤러 옆면에 여러 개의 구멍을 뚫었고, 내부를 비워 두어 원기둥 껍질로 다시 설계하였다. 새롭게 설계한 롤러는 다음 주 월요일에 출력이 완료되는데 이번 디자인 리뷰1 때까지만 큰 롤러를 사용하는 것으로 한다. 롤러의 사이즈를 줄여야 하는데 혹시 롤러의 사이즈를 줄였다가 공을 픽업하지 못하는 불상사가 일어날 수 있기 때문이다. 그래서 크기를 줄인 롤러도 뽑아볼 계획이긴 한데 일단은 픽업이 잘 되는 큰 롤러를 사용할 것이다.

Week6:

1) 박성혁

(1) 주말간 3d 프린팅 작업을 하여 맡기었으나 사용하기에 적합하지 못한 결과가 나와 부품 설치판에 대한 우선순위는 미뤄두었다. 또한 전체적인 기기 구조를 바꿔야하는 문제점이 발견되어 급하게 기기를 축소시켰다.

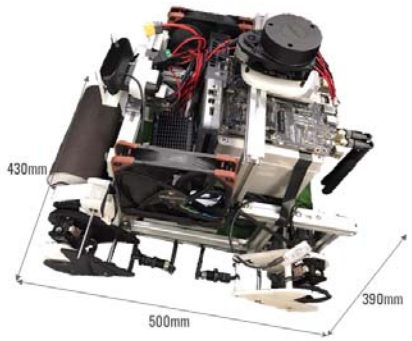
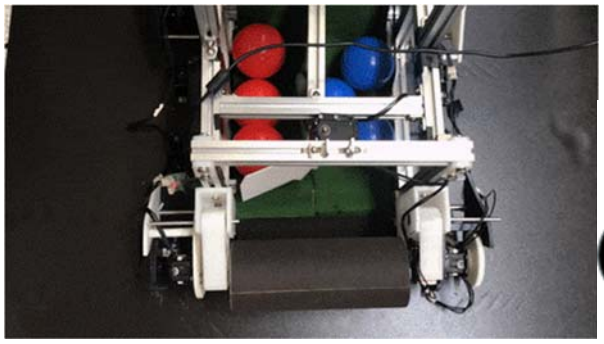


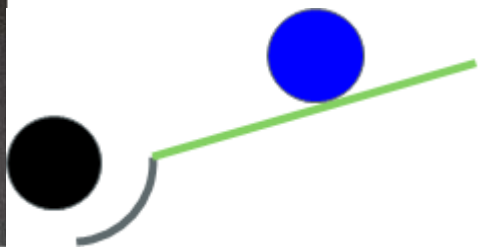
Figure 16. 바뀐 기기의 구조

지난주에 제시되었던 랩뷰에 대한 수정이 있었다.

(2) ros에서 임의의 element를 바로 모터 속도 값으로 입력하게 수정하였다. 이 경우 각각의 바퀴에 임의의 속도값을 받는 것으로 수정하였고, 메카넘 휠 방향에 대해서는 ros팀 쪽에서 이해를 하고 이에 따른 코드를 수정한 것으로 보인다. 기존 코드가 조이스틱 버튼에 따라 여러 가지 case로 정해져 있던 것에 비해선 랩뷰 코드가 더욱 깔끔해졌다. (단순히 각각의 모터에 case없이 element 요소로 속도값 입력)

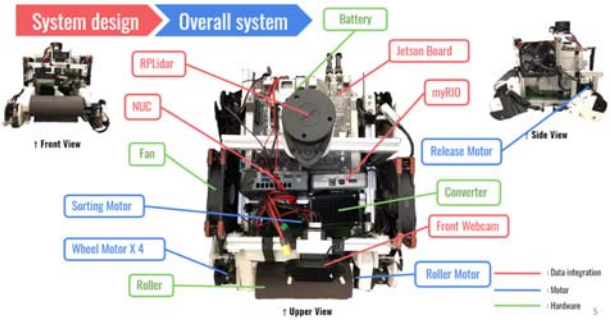
(3) 관절 모드의 다이나믹셀 구동에 성공하였다. 이미 설치를 해놓은 상태여서 켈레브레이션으로 정확하게 영점을 잡지는 않았고 실험적 데이터를 통하여 관절 모드 구현에 성공하였다. 다만 여기서 문제점이 하나 발생하였는데, 관절 모드의 구동값이 최대 360도인데 현 기기 상태의 릴리즈 모듈은 이러한 방식으로 구동이 불가능하다. 왜냐하면 360 회전만으로 충분히 천을 감았다 피기에 부족하기 때문이다. 따라서 첫 번째 디자인 리뷰가 끝난 뒤에 릴리즈 부분의 원통의 지름을 더욱 키울 예정이다.

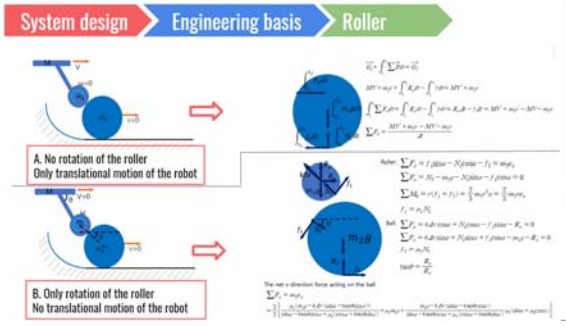


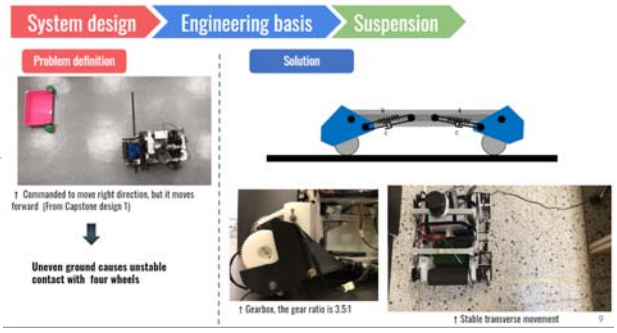


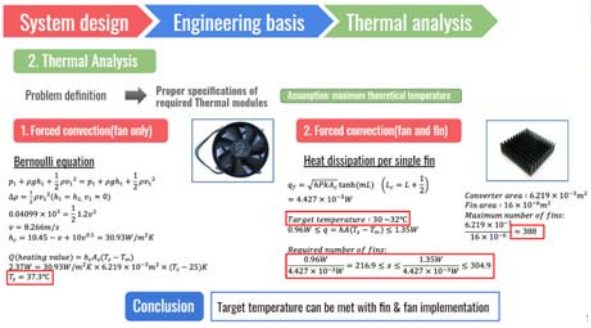
그림과 같이 릴리즈 모듈은 천을 감아서 경사를 만드는 것인데 이를 구현하기에 원통의 지름이 작다.

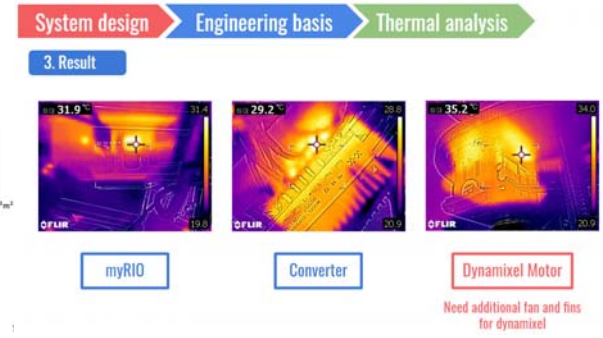
2) 조현태

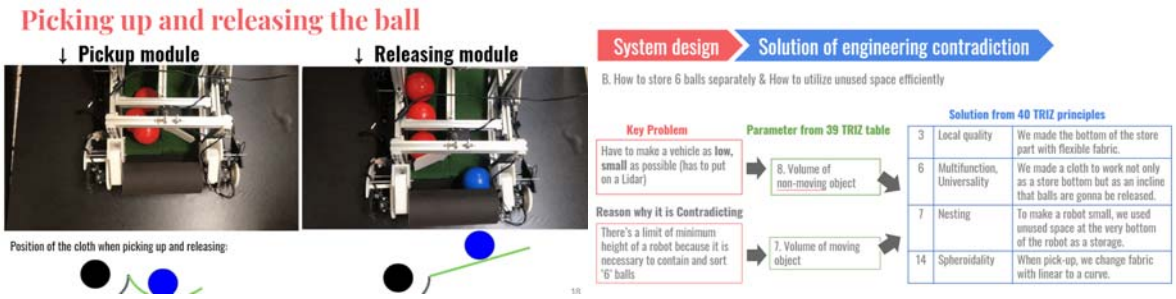












3) Zulfiqar

(1) External Structure of our System

Together with the members of the hardware group I helped in setting up the external structure of our system which includes setting the suspensions, structure skeleton.

(2) Thermal Analysis of our System

According to the Evaluation Criteria our system must be within the maximum temperature range of 30 to 32 degrees to score highest marks. It is my role to make sure of this.

Firstly, I used the Infrared Camera to check the hotspots of our system when it is under full operation.

The following top three hotspots were observed:

Figure 17. Myrio

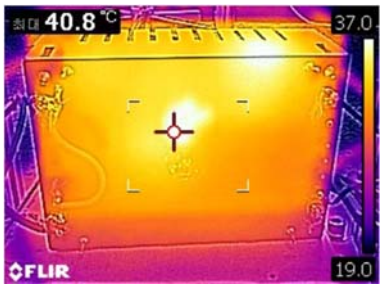


Figure 18. Converter

Figure 19. Dynamixel Motor

It can be seen that these hotspots are far from being in the maximum temperature range and thermal modules like fan and fin must be implemented to cool them down. For that I did the required calculations:

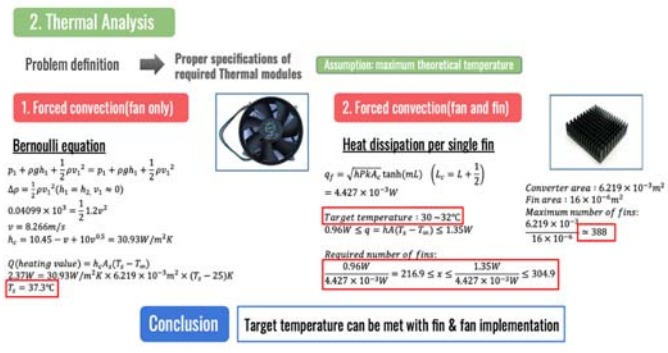


Figure 20

Implemented fan and fins in our system and took the IR camera pictures of the following components:

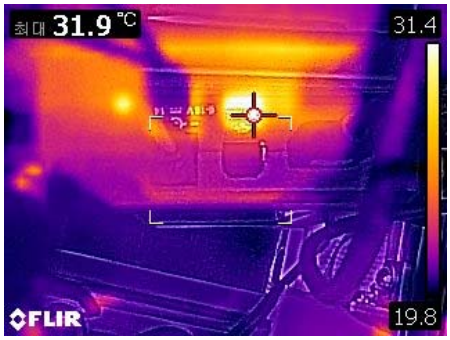


Figure 21. Myrio

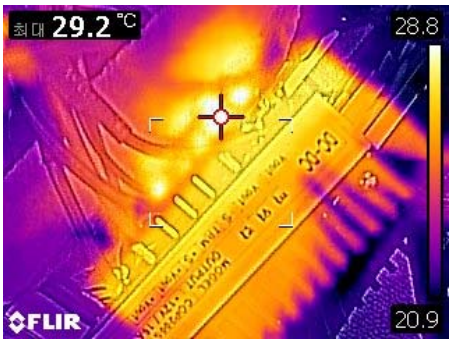


Figure 22. Converter

It shows that I was successful in the installing thermal modules for cooling according to my analysis. However, the fan and fin were not installed for the dynamixel motors as there are about 7 and they are each at different locations on our system hence additional fans and fins are required.

4) 장민석

원래 롤러의 지름이 18cm나 되었다. 지난주에는 롤러를 작게 뽑았다가 공을 먹지 못하는 불상사가 일어나지 않도록 롤러의 크기는 그대로 유지한 채로 데모 영상을 촬영하려고 하였다. 그런데 로봇의 크기가 전체적으로 커 SLAM에서 미로를 탈출하는데 힘겹다는 이야기가 있었다. 그래서 롤러의 크기를 최대한 줄이고자 하였다. 그 결과 <Figure 23>을 보면 롤러의 크기가 획기적으로 줄어든 것을 확인할 수 있다. 크기를 줄였는데도 불구하고 램프 구조를 바꾸었기 때문에 공도 잘 먹기도 하였다. 그러나 롤러의 크기가 작아지면서 롤러의 크기가 땅에 더 가까워졌다. 그러다보니 공을 뱉을 때 롤러가 들리지 않고 헛도는 문제가 발생하였다. 이 문제는 중간고사가 끝나고 해결할 계획이다.

Figure 23

롤러뿐만 아니라 로봇의 크기가 전체적으로 너무 커서 SLAM을 사용하여 미로를 탈출하기 쉽지 않았다. 그래서 설계팀이 모두 모여 로봇을 작게 만들기 위해 처음 프레임부터 다시 조립하였다. 라이다를 높이 40cm에 위치시키기 위하여 서스펜션의 높이를 3cm 이상 낮추었고, 공을 보관하는 부분을 더 짧게 다시 설계하여 길이 60cm에서 50cm로 줄였다. 아래 <Figure 24>를 보면 지난번보다 로봇이 많이 압축해진 것을 볼 수 있다. 창시구2는 1과 달리 로봇의 공간 활용을 어떻게 하느냐가 중요하기 때문에 빈 공간에 부품을 최대한 효율적으로 배치하려고 하였다. 라이다 밑의 공간에는 배터리가 들어가고 그 앞 부분에는 NUC, myRIO, 회로를 배치하였다. 가장 높은 온도까지 올라가는 부품의 열을 효과적으로 제어하고자 팬의 바람이 이 부품들을 직접적으로 닿을 수 있도록 하였다.

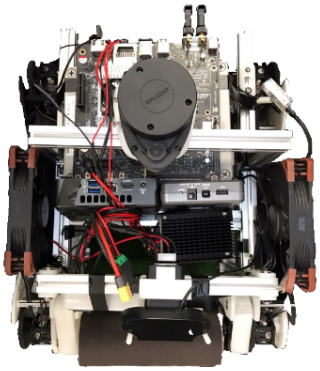


Figure 24

그리고 이번 주에는 Design review 1이 있었기 때문에 로봇을 완성하고 데모 영상을 촬영하고 발표 자료를 제작하였다. 앞으로 설계팀에서 남은 일은 부품을 제대로 고정하고, 롤러의 위치를 공이 잘 나가도록 수정하고, 릴리즈할 때 공이 3개 모두 들어가도록 천의 위치를 수정하고, 서스펜션을 완성하는 일이 남았다.

5) Farhad

This week our group spent much time on finalizing the vehicle assembly, code, and the design review presentation. I mainly worked on the animations for the presentation that would depict the working mechanism of our car. The animations had to describe three mechanisms: pickup, sorting, and release of the balls. They were created using Microsoft PowerPoint.

The first task was to animate the process of picking up the balls. The pickup module consists of a cylindrical roller, with a diameter slightly larger than that of the balls it picks up, a ramp with a somewhat steep incline angle, and a cloth, with one end attached to the sharp end of the ramp, and the other one linked to a motor. When the vehicle moves looking for the balls on its path, the roller continuously spins counterclockwise, and once a ball comes to contact with the roller, it starts a translatory motion with a clockwise spin. With the momentum transfer from the roller, the ball climbs up the ramp, and reaches the cloth and stays there due to the downward curvature of the cloth.

The second task was to demonstrate what happens once the roller picks up the ball. The sorting mechanism consists of a simple rod which rotates about its pinned end and, depending on the color of the ball, closes one of the two gateways, thereby opening the other one for the passage of ball.

The third task was the animation of the release module. The motor rotation raises the end of cloth attached to the motor, which, in turn, causes the downward curved cloth to straighten up and become inclined. Under the effect of gravity, the balls spin and move down the cloth and ramp, and with a clockwise spin of the roller are released outside.

The animations were created using the shapes and animations panels in PowerPoint and functions like spin, custom path, disappear, lines, and arcs.

The following pictures illustrate the mechanism animations:

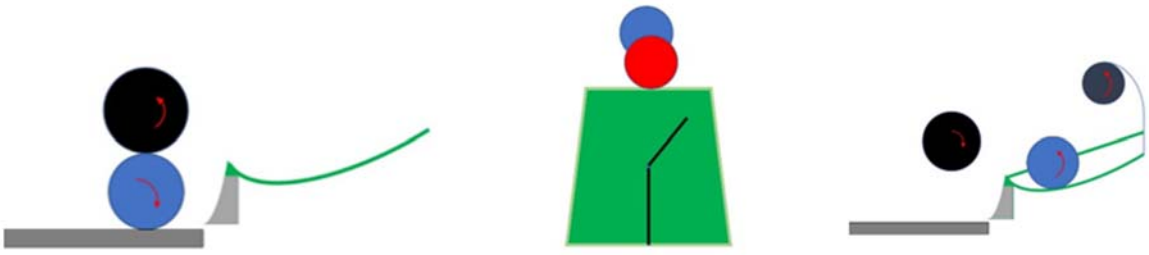


Figure 25. Illustration of the working principle of the vehicle. From left to right: Pickup module, Sorting module, and Release module

Week9:

1) 박성혁

기존의 모델의 디자인에 결함이 몇 가지 있어 전면적인 수정이 이루어졌다. 가장 중요하게는 천을 감아 올리는 방식이 아닌, 판을 CAM을 이용하여 들어 올리는 구조로 변경하였고 이에 따라 롤러와 램프의 디자인 변경도 같이 이루어졌다. 이 부분은 하드웨어팀의 다른 조원들이 진행하였고 본인은 이에 따라 다시 다른 부품들(배터리, nuc, myrio등)의 설치를 해야 하며 랩뷰 코드에서도 각도(sorting bar와 release model)에 대한 수정이 필요한 상태이다. 아직 변경된 부분이 진행되지 않아 대기중인 상태이며 거치대 설치를 위한 3d프린팅 요청을 준비중이다.

2) 조현태

Figure 26. Fixing the redundant and un-precise parts

3) Zulfiqar

(1) Changing the gear ratio from 1:2.5 to 1:3.5

When we tested out our car with the existing gear ratio however the car speed was not high enough and hence we needed to increase the gear ratio. After doing certain calculation it was concluded a gear ratio of 1:3.5 would suffice. Hence this week I started on changing all the gears in the gear box. I have 3D modelled the required gear and after it is printed will install in each gear box.

(2) Changing Suspension specifications

After experiments we noticed that our suspension is working fine however it is not that smooth and quick in performance for some extreme cases. After doing our analysis we found that it was a problem of transmission angle and hence concluded that a more higher K value of spring in the suspension would be better. So have ordered different springs of higher K value and will replace them with the existing spring in the suspension system.

4) 장민석

조원들과 상의하여 앞부분 픽업을 내가 수정하기로 하였다. 픽업 모듈에서는 바꿀 점이 크게 세 가지 있다. 첫째는 서스펜션의 높이가 생각보다 많이 낮아져서 램프의 높이를 바구니보다 높게 바꿔야 했고, 둘째는 롤러암 서포터가 서스펜션이랑 계속 부딪혀서 두께를 작게 해야했고, 마지막으로는 릴리즈할 때 롤러와 공이 잘 맞물리도록 롤러암의 각도를 바꿔야했다. 이것을 수정하고 난 후 부품의 모습은 <Figure 27>과 <Figure 28>에서 볼 수 있다. 이외에도 롤러암의 각도를 바꾸다보니 롤러암의 길이도 바꿔야해서 <Figure 29>와 같이 수정했으며, 다이나믹셀을 고정하기 위해 원래는 M2짜리 구멍을 뚫어두었는데, 조립의 편의를 위해 M5 구멍으로 수정하였다. 그리고 새롭게 수정한 부품들을 조립한 모습은 <Figure 30>에서 볼 수 있다. 롤러는 기존에 사용했던 것을 조립했고, 램프에는 사포를 부착하였다. 원래 계획은 조립을 완료한 후 픽업과 릴리즈가 잘 일어나는지 xbox controller로 확인하려 했으나 그 부분은 맡은 조원들이 준비가 덜 되어 픽업과 릴리즈는 확인하지 못했다. 이외에도 우리 로봇에 가장 효과적인 서스펜션을 찾기 위해 서로 다른 용수철 상수를 가진 서스펜션을 구매하였다. 다음 주에 간단한 실험을 통해 우리 로봇에 맞는 서스펜션을 찾을 계획이다.

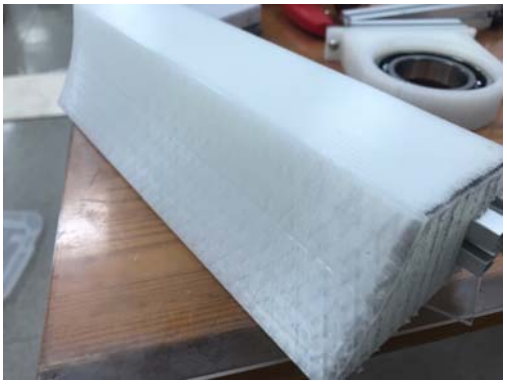


Figure 27

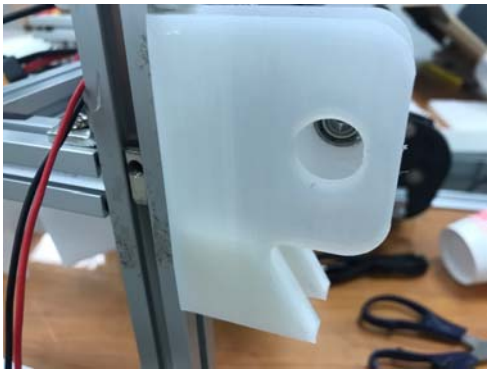


Figure 28

Figure 29

Figure 30

5) Farhad

For this week, me and my partner had to handle two of the most significant issues our team was facing in the post-midterm period. Firstly, our suspension system was not as good as required since the springs had a very low spring constant value, and, thus, in the default plain position the springs were fully compressed. As a

result, the vehicle's suspension system could not cope with disturbances to the wheels, when there was an uneven ground under the vehicle, since the spring force generated was not adequate for absorbing the impacts. Secondly, the vehicle was not able to generate high speeds due to the inherent gear ratio. Hence, the gearbox had to be modified.

To handle the suspension problem, we wanted to acquire four new suspensions (for each wheel) with a higher value of the spring constant, and keep the damping coefficient same. As the suspension mechanisms were taken from previous projects, the spring constant value was unknown. An experiment was implemented, whereby some weights were put on the spring and the consequent displacements were measured. The value calculated for the spring constant was used as a reference while looking for springs with higher k values. In addition, we were able to separate springs from dampers, and, thus, we only needed to acquire four new springs and install them on the previous dampers.

With regard to the gear ratio, for simplicity, we first decided to keep the small gear and design a new big one. All the necessary dimensions were taken, like the inner and outer diameters of the two spur gears, the number of the teeth in the two gears. The initial gear ratio was calculated to be 2.58:1. To gain higher speeds we determined that the ratio should be around 3.5:1. For that to happen, we designed a new big gear with a larger diameter and number of teeth, according to the ratios calculated. The SOLIDWORKS model was sent to the supervisor for 3D printing, and now we are expecting the printed gears. Two potential problems that may arise from this solution are that the newly-made gear teeth may not match perfectly with those of the small gear, and the gear housing may collide with the new gear since it is bigger. The former issue requires a new small gear to be made with the exact teeth shape and pitch, whereas the latter one can be easily solved by shifting the motor inside the housing so that the casing structures do not collide with the big gear.

The following pictures illustrate the newly-designed spur gear:

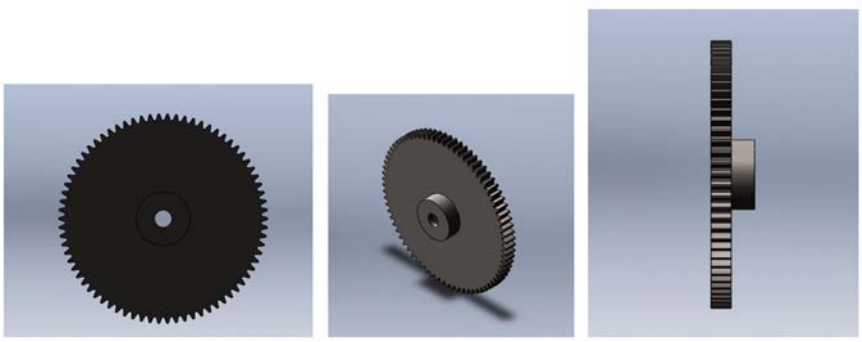


Figure 31. Different views of the new spur gears

Week10:

1) 박성혁

릴리즈 파트가 크게 변하면서 기계의 구성도 크게 변하였다. 금주 예비군 훈련이 있어(11/4~11/7) 서울지역을 다녀온 까닭에 개인적인 많은 진행은 없었다. 다만 부품(myrio, NUC, converter GPU ect)을 거치하기 위한 3d 프린팅을 몇 가지 요청하였다.

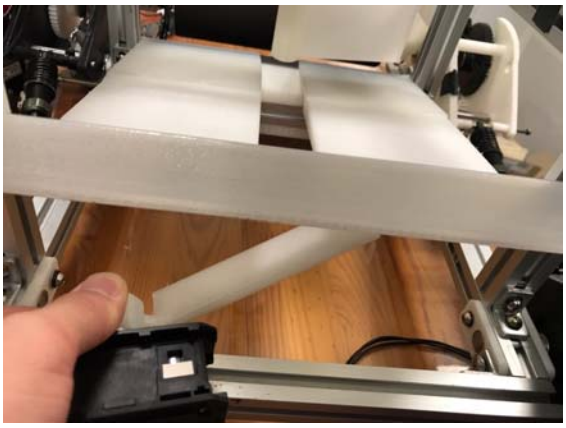


Figure 32. 바뀐 릴리즈 파트

2) 조현태

Figure 33

Rediscarded the cotton release model and changed to solid release model

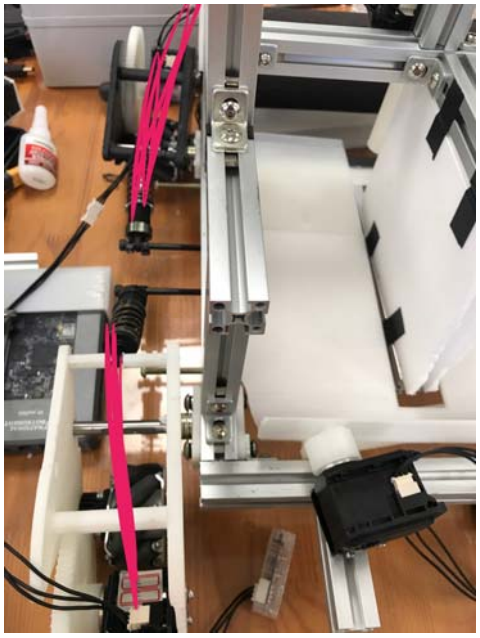


Figure 34

We implemented solid release part to the robot and made a whole new release part

3) Zulfiqar

(1) Completed changing the gear ratios from 1:2.5 to 1:3.5

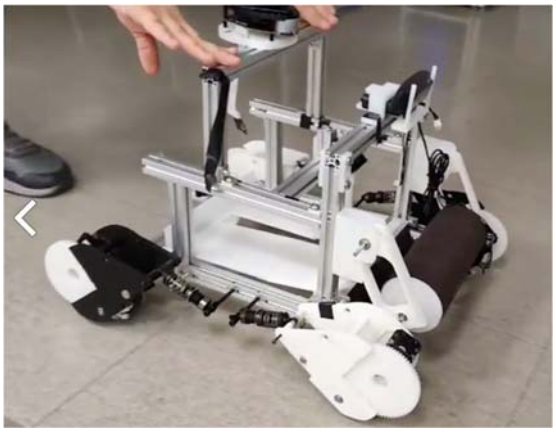


Figure 35

After I designed the new gears on Solidworks I worked on installing those into the previous gear box. It was a lot of work in terms of labor as the previous gear boxes had to changed and a lot of measurements had to be taken to install the position of gears such that the relative motion is smooth.

(2) Changing the value of K in suspension

After the springs we ordered arrived I tried the new springs on the suspension system. After testing the system on the uneven ground I concluded that the suspensions work well.

4) 장민석

원래는 조원들이 수정한 픽업과 릴리즈 모듈이 제대로 되는지 확인하기 위해 엑스박스 컨트롤러로 시험해볼 계획이었으나 랩뷰를 담당한 조원의 사정으로 이번 주에 진행하지 못했다. 그래서 엑스박스 컨트롤러로 로봇이 제대로 움직이고 공을 잘 줍고 뱉는지 확인하는 것은 다음 주에 진행할 것이다. 그리고 서스펜션의 경우, 원래 로봇에 장착한 스프링의 상수가 너무 작아서 서스펜션 역할을 제대로 하지 못해 더 센 것으로 바꾸었다. 원래 있던 스프링 상수에 500g 분동을 올려 얼마나 압축되는지 측정하여 용수철 상수를 얻어 이 값보다 큰 값을 가진 용수철을 여러 종류 구매하였다. 그러나 구매한 스프링 모두 damper와 규격이 맞지 않아 사용할 수 없었다. 그래도 다행히 조원 중 한 명이 실습동에서 적당한 스프링을 찾아 서스펜션 부분을 완성할 수 있었다. 그러나 <그림 17>을 보면 양쪽 서스펜션의 용수철 힘이 달라 비뚤어진 것을 볼 수 있다. 이를 해결하기 위해 용수철 길이를 조절하였고, 그 결과 양쪽의 균형을 어느 정도 맞출 수 있었다. 그러나 여전히 왼쪽과 오른쪽이 5mm 높이 차이가 있다. 조원들과 상의한 결과, 일단 차이를 그대로 두고 혹시 나중에 소프트웨어 팀에서 라이다를 사용할 때 오차가 너무 크다면 그때 문제를 해결하는 것으로 결론을 내었다. 그리고 <Figure 36>을 보면 롤러가 한쪽으로 기울어져 있는 것을 볼 수 있는데, 오른쪽 롤러암(Roller arm)을 더 짧게 출력하여 균형을 맞추고자 하였다. 다음 주에 롤러암 출

력이 완료되면 확인해볼 예정이다. 그리고 서스펜션의 용수철을 새로 바꾸니 로봇의 높이가 달라져 램프(ramp)의 높이도 바꿔야 했다. 그래서 높이가 더 높은 램프를 부착하였다. <Figure 37>은 서스펜션을 부착하고 로봇이 움직이는 사진이다. 위에서 언급한 문제 외에도 릴리즈의 경사면이 작다든지 부품 고정은 어떻게 할지에 대한 문제가 남아있다. 이러한 문제들을 최대한 빨리 해결하여 소프트웨어 팀을 위해 다음 주 수요일까지 하드웨어를 완성하는 것이 목표이다.

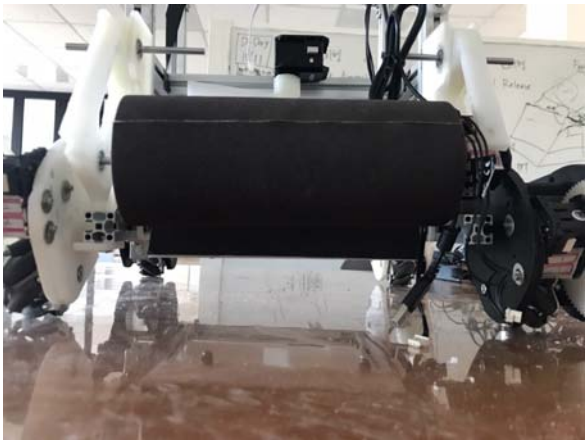


Figure 36

Figure 37

5) Farhad

We obtained the 3D-printed gears we designed last week. Our analysis and measurements of previous gears proved to be correct as the newly-printed gears perfectly matched with the smaller gears of the gearbox, which we decided not to change. Thus, one of the potential problems discussed in the previous week is naturally eliminated, namely, the requirement of making new smaller gears. The second issue, however, had to be tackled. For that, we first installed the newly-printed gears on the surface of the motors with bolts, after which the duo was placed inside the housing such that the collisions with the housing rod and walls were

avoided as well as the two gears were perfectly matched and aligned without exerting too much force on each other. The position of the motors and their bolt holes were marked on the corresponding surface on the housing directly below the motor surface. Next, the gear housings were peeled off with smoothing tools according to the marks made earlier. Subsequently, the diameter of motors' bolts was measured and corresponding drill bits were utilized in the drill press to make holes in the gear housings on the marked locations. These procedures took some time since they required a significant amount of manual work. Once the aforementioned tasks were completed, the motors with the gears attached to them were installed on the housings with the bolts, nuts, and washers. The finalized gearboxes were fixed back on the vehicle.

The next task completed was changing the springs in suspensions with newly-purchased springs with a higher value of k.

The following image illustrates the car after this week's task:

Figure 38. The vehicle after modification of the suspension and gearbox

Week11:

1) 박성혁

시현을 하면서 새로 발견된 문제점들은 다음과 같다.

1. 알루미늄 바의 수평과 라이더의 위치가 정확하지 않음

2. 기어박스의 고정이 정확하지 않음(다이나믹셀이 흔들림)

3. 서스펜션이 올바르게 작동하지 않음

소프트웨어 담당 팀에서 급하게 요구하는 점은 수평과 로봇의 이동이 가능하다는 것을 강조하였기에 위 문제점을 해결하는데 중점을 두었다.

이러한 문제들에 대하여 하드웨어 팀은 분담했고 본인은 주로 1번과 3번 문제에 집중하여 문제해결을 도왔다. 수평계를 이용하여 최하단의 알루미늄 바 위치 조정 작업을 실시하여 수평을 맞추었고 최상단 라이더의 위치 또한 로봇의 중심으로 옮기며 수평 작업을 동시 진행하였다. 이런 작업은 서스펜션의 수정과 함께 진행되었다.

이러한 작업을 진행하면서 기타 다른 부품의 설치도 완성하였다.

Figure 39

배터리 설치는 테이프를 이용하여 고정하였고, Myrio와 NUC 위 사진처럼 박스를 만들어 거치하였다.

2) Zulfiqar

(1) Fixing the Gear boxes

After fixing the gear boxes we encountered a problem of the gearboxes being tilted. We found out that the aluminum profile skeleton of our robot was not symmetric rather tilted from one side hence we have started to work on that.

(2) Fixing the suspension

The suspension initially worked fine but still it was not efficient enough and worked for only a little vertical displacement of uneven ground. Hence this week we tried to fix the suspension.

3) 장민석

모든 부품을 로봇에 고정하기 전에는 로봇의 무게가 크지 않아 서스펜션의 높이가 높았다. 그래서 예상했던 설계보다 로봇이 높아져 ramp가 바닥과 과하게 떨어져있어 공이 로봇 안으로 잘 들어오지 않았다. 그래서 ramp의 높이를 크게 하여 다시 출력하였다. 그리고 NUC, myRIO는 <Figure 40>에서 볼 수 있듯이 상자 모양으로 출력한 고정대에 고정하였고, 그 뒤에 있는 배터리, 컨버터, 팬은 로봇에 합판을 덧대어 평평한 바닥을 만든 다음에 <Figure 41>처럼 벨크로를 사용하여 부착하였다. 벨크로는 접착력이 좋으면서도 원할 때는 부품을 뗄 수 있기 때문에 추후에 로봇을 수정해야할 일이 생겼을 때 작업을 편리하게 해줄 것이라 생각한다. 부품들을 모두 고정하고 나니 서스펜션이 상당히 많이 압축된 것을 볼 수 있었다. 그래서 로봇의 높이도 낮아져 새롭게 설계한 ramp가 이제는 너무 바닥에 가까워졌다. 다행히 이전에 출력해놓은 작은 ramp의 높이가 바닥과 정확히 3cm 떨어져 있어 그 ramp로 대체하였다. ramp와 바닥 사이의 간격을 3cm 둔 이유는 공을 릴리즈하는 빨간색 바구니의 높이가 2.8cm 정도 되기 때문이다.

로봇의 서스펜션이 앞으로 지나치게 돌출된 형태여서 롤러를 회전시키는 다이나믹셀과 기어박스가 서로 걸리는 문제가 발견되었다. <Figure 42>을 보면 다이나믹셀의 나사와 기어박스가 서로 걸쳐있어 롤러가 원위치로 돌아가지 못한다. 다이나믹셀을 고정하는 나사 머리를 집어넣든지 기어 박스 위치를 조정한다든지 등의 방법으로 이 문제를 해결하기 위해 조원들과 더 상의해볼 필요가 있다.

앞으로의 계획은 소프트웨어팀이 로봇을 사용하면서 생기는 문제들을 수시로 수정할 것이고, 부러지기 쉬워보이는 부품의 경우 더 튼튼한 구조로 출력할 계획이다.

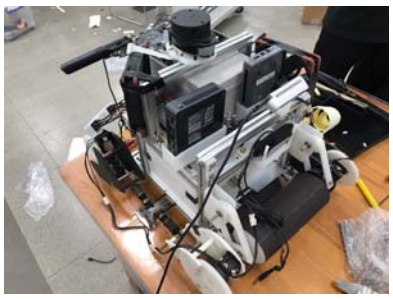


Figure 40

Figure 41

Figure 42

4) Farhad

After completing the tasks from previous weeks, namely gearbox and suspension design and installation, we waited for some feedback from our fellow software team and other people in the hardware team.

One of the first issues identified was that the newly-designed roller's dynamixel was colliding with the wall of the front left gearbox. Some of the propositions was to grind the surface of the gearbox as well as change the design of roller, and possibly its arm and the holder. However, while the latter solution was too complicated and could take too much time, the former was too risky to implement since the material was fragile, and gearbox had already sustained much grinding by that time; thus, it could easily break. Nevertheless, we were able to find a much simpler solution for this problem. I have observed that once some force is exerted on the roller in the right direction (looking from the perspective of the vehicle), there was no collision between the roller dynamixel and the gear housing, when the roller was to move up and down. Therefore, I have noticed that there was some empty space between the left roller arm and its holder, so it could slightly move in the right direction. In other words, the arm was a bit loose. The solution I proposed was to push the arm to the right and make the arm and the holder tighter. This could be done by designing a new holder for the left arm holder with adding some asymmetry and reprinting the part and changing the faulty one with it. However, it was much easier to eliminate the looseness by filling that empty space with some material, like a glue gun,

which could tighten the arm and avoid the collision of dynamixel and the housing. Now we are expecting some feedback about the new solution.

Some of the feedback received from our team, once they tried the car with an Xbox control were as follows:

- Sometimes some of the gears rotate without traction. This can be due to the fact that gears are slightly loose and should be fixed more tightly to the housing.
- A screw fixing one of the gear housings does not fully get inside the hollow shaft since there is a difference in the diameter along the shaft. This can affect the mobility of the vehicle when it tries to escape the maze since it can collide with the maze walls. One of the potential solutions is to cut the part of the screw sticking out.
- The suspension does not properly work since the spring constant might be too strong. One solution could be to change the springs to the ones with a constant larger than the initial one and smaller than current one, since in the initial case, the springs were almost fully compressed, and in the current one they are almost fully stretched in the default position.
- The gearboxes are not parallel and there might be some torsion. This can be fixed by checking the fasteners and how the housings are attached to the main frame.

We will have more discussions and try to solve all of the issues next week.

Week12:

1) 박성혁

기어박스 및 서스펜션 수정, 수평 작업, 롤러 부분 수정을 마치고 기계의 최종적인 디자인이 완성된 후 sorting bar와 releasing board의 회전각을 고려한 랩뷰 코드를 완성하였다. 기존에 주어졌던 관절 모드 코드의 적용이 원활하지 않아서 시험적으로 다양하게 시도했으며 우리가 원하는 수준의 동작을 구현할 수 있게 완성하였다.

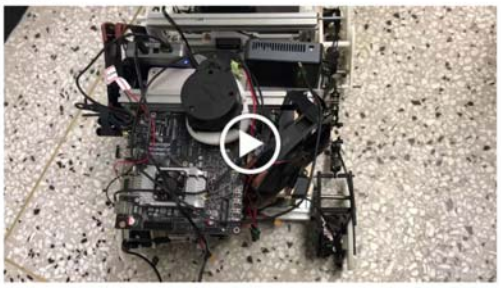
2) 조현태

Pick-up

Pick-up

Release

Movement



Movement

3) Zulfiqar

(1) Finish fixing the Gear boxes

After printing out new gears, I worked in fixing them back into the gear box. The main aim was to improve the interlocking of the gears to prevent vibration when the robot moved

(2) Finding problem in Circuit

I was informed by the software team that the NUC was not working or may have been faulted by the circuit. So I disassembled the circuit from the robot and measured the voltage being supplied to the NUC using multimeter. The voltage was 23.96 V which was voltage supplied directly from the battery and is within the working range of NUC . Hence it was concluded that the NUC did not fault because of problem in the circuit. One reason may have been short circuit in the open terminals of the converter which were not covered and when the + AND – terminal come into contact the aluminum skeleton of the robot it could cause short circuit. Hence I fixed the converter on the Robot such that it would not happen.

4) 장민석

기존에 있던 서스펜션 구조가 제대로 역할을 하지 못해 서스펜션 구조를 새롭게 바꾸었다. 기어박스를 새로 디자인하기에는 시간이 부족하여 원래 사용하던 기어박스를 최대한 유지하여 서스펜션을 바꾸고자 하였다. 서스펜션이 이 로봇과 고정되는 축을 위로 올렸더니, 기어박스의 방향과 서스펜션의 압축 방향이 달라 서스펜션 역할을 제대로 하지 못하는 것이 확인되었다. 그래서 회전 방향 반대쪽에 긴 나사를 박아 기어박스가 회전하면 그 위치에 놓인 서스펜션이 압축되는 구조로 설계하였다. 하지만 이때는 압축 스프링 대신에 인장 스프링을 사용하였는데, 서스펜션 역할을 잘 했는데도 불구하고 인장 스프링의 용수철 상수가 지나치게 커 기어박스가 고정된 알루미늄 프로파일이 뒤틀리는 현상이 발견되었다. 이 때문에 바퀴도 함께 뒤틀려 바퀴가 사선으로 놓이게 되었다. 그래서 진동도 굉장히 커졌고, 구조도 오히려 불안정해졌다. 인장 스프링의 용수철 상수를 더 작은 것을 사용하면 될 것 같은데 시간이 부족하여 다른 방안을 생각해보았다. 그래서 최종적으로 생각해낸 구조가 <Figure 43>이다. 다행스럽게도 서스펜션이 너무 안정적으로 제 역할을 하였고, 용수철 상수가 큰 인장 스프링 대신 용수철 상수가 적당한 압축 스프링을 사용하니 알루미늄 프로파일이 뒤틀리는 문제도 사라졌고, 로봇의 수평도 잘 맞게 되었다. <Figure 43>의 서스펜션 구조는 굉장히 단순해 보이지만 이 구조에 도달하기까지 수많은 시행착오가 있었다.

서스펜션 외에 또 다른 문제로, 기어가 잘 맞물리지 않는 문제가 있었다. 맨 처음에 기어를 디자인한 조원이 기

어가 서로 안 만나는 문제를 기어의 위치를 바꾸는 것이 아니라 다이나믹셀의 고정을 느슨하게 하는 것으로 해결해놓았다. 그래서 진동도 심해지고, 다이나믹셀이 계속 흔들렸다. 문제의 핵심을 해결하고자 다이나믹셀 기어의 치수를 살짝 수정하여 바퀴의 기어와 안정적으로 맞물리도록 하였다. 그래서 기어박스를 위에서 본 <Figure 44>에서 볼 수 있듯이 다이나믹셀의 기어와 바퀴의 기어가 서로 잘 맞물려있는 것을 볼 수 있다.

그 다음으로 발견된 문제는 공을 릴리즈할 때 롤러와 공 사이의 거리가 약간 부족한 것이다. 롤러의 지름을 1cm~2cm 정도 크게 해서 출력하면 해결될 수 있지만 롤러의 크기가 커지면 무게도 증가하고, 출력될 때까지 시간도 걸리고, 롤러를 크게 한다고 해서 해결되지 않을 거 같아 <Figure 45>처럼 기존의 롤러에 언덕을 만들어 공과의 살짝 부족한 거리를 언덕에 걸려 넘어올 수 있도록 하였다. 릴리즈 판에도 문제가 생겨 공이 제대로 릴리즈되는지 xbox controller로 확인해보지는 않았지만 손으로 롤러를 돌려보았을 때는 제대로 작동하는 것이 확인되었다.

다음 주에는 로봇을 크게 수정할 계획은 없다. 그러나 소프트웨어팀이 로봇을 사용하면서 고쳤으면 하는 부분을 말해주면 하드웨어팀 조원들과 상의하여 해결 방안을 찾아 로봇을 수정할 계획이다.

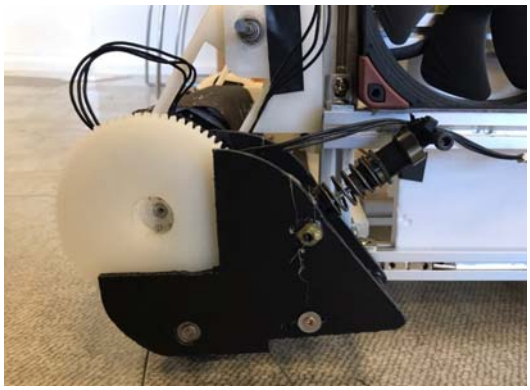


Figure 43

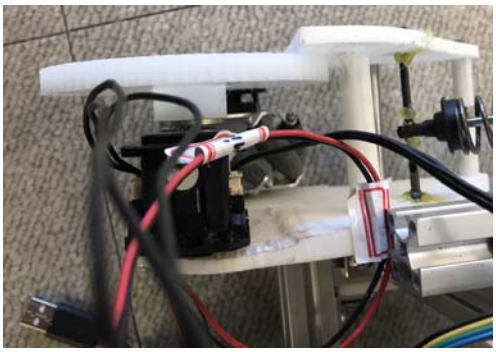


Figure 44

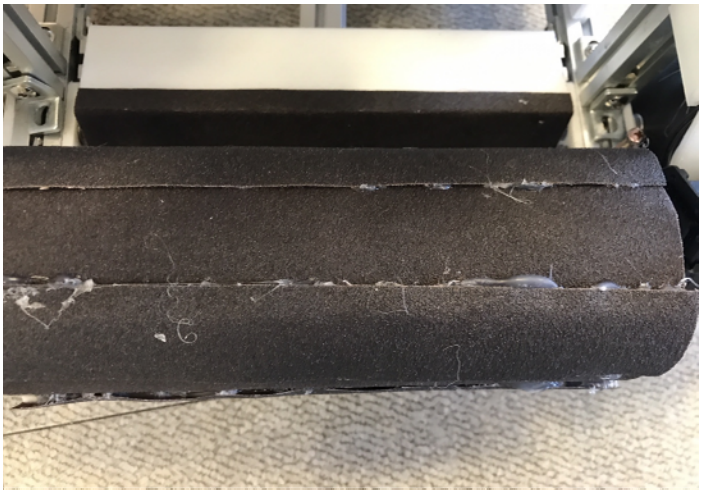


Figure 45

5) Farhad

First, the problems identified in the previous week were tackled.

Rotation of some of the gears without tractions was determined to be because of relative looseness in the attachment of the motors to the gearbox since some of the gears were fixed with two screws. For this reason, we determined additional positions to be drilled and added an additional screw to those positions along with washers. With regard to the existing screws, we fastened them much more to make the motors firmly attached to the gearbox housing.

The screw sticking out from one of the gearbox housings was first detached. Following this, the nuts, which were supposed to fix one end of the suspension to the gearbox, were removed. After that, the screw fully went inside the shaft and was not sticking out anymore.

Regarding the work of suspension, me and my partner decided to alter the locations of attachment of the suspension: the position at the gearbox as well as the one at the main frame. Two more aluminum rods were fixed at the main frame for both sides, at a position so as to make the fixing location at the main frame higher than before. Furthermore, an additional shaft was attached at the gearbox so as to make the fixing location at the gearbox again higher than before. In this way, suspension started to work properly: spring would absorb the vibration coming from an uneven ground, and the damper would dissipate this absorbed energy so as to minimize the vibration.

There was an additional vibration of significant strength during the motion of the vehicle, whose source was identified to be from the gearbox: the screwheads of the screws fixing the smaller gears would collide with the larger gear during the motion, and this would cause a strong and noisy vibration, especially during side motions. I have designed new big gears in order to solve this issue. The hollow cylinder on the gears used for fixing the gears were extruded more in order to shift the location of gear teeth farther from the screwheads of the smaller gears. The decision was justified since the possible collision in the new design with the gearbox housing could be simply solved by adding oil between the walls and gears or slightly grinding the housing.

The following image illustrates the new design:

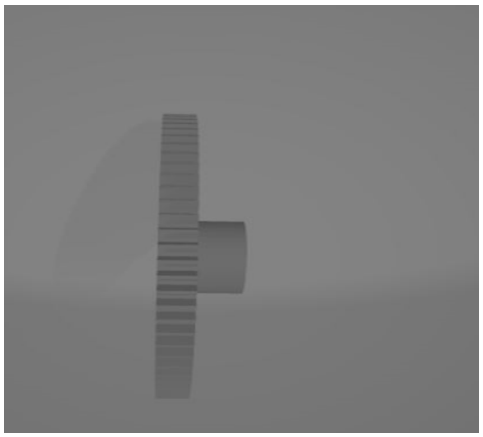


Figure 46. Longer gear extrusion

Another important problem was the problem of tilt which would also lead to vibration since some of the wheels would not even sometimes touch the ground, or the nuts of the wheels would touch ground. This was addressed by making that the aluminum profile was fixed properly without tilt since initially some of the rods were quite tilted. Moreover, adjusting the compression of the spring by changing the longitudinal location of the attachment at the main frame. To check the tilt after modifications the bubble tiltmeter was used. The outcome was acceptable.

Adjustment of the spring compression also solved the issue of the uneven weight distribution. This was checked by placing the four wheels on four scales and making sure that values were close to one another. Relocating some of the hardware like converter, fans, battery, NUC, myRIO also helped solve the problem.

The new issues identified in the hardware were as follows:

- Pickup by the roller during fast motions.
- Collision between the gearbox and basket during release, which resulted in some distance between the ramp and the basket. Thus, the balls could not be released inside the basket.

I have solved the first issue by simply increasing the roller angle slightly by adding some foam material between the roller arm and the holder. This created more space for the ball and it was able to get inside, experiments showed. After this modification, the vehicle was able to capture the balls at even the highest rpm of the wheels (speed of the car).

The second issue was solved by shifting the location of attachment of the shafts of the front wheel gearboxes on the main frame back by a distance measured between the ramp and the basket in the current configuration (around 80 mms). This was achieved by utilizing the drilling machine to make new holes in the aluminum bars for fixing the gearbox shafts. Subsequently, the front part of the vehicle was able to get fully inside the basket, and the balls could be released inside the basket.

Week13:

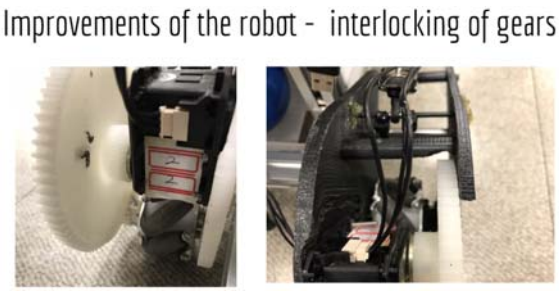
1) 박성혁

하드웨어나 랩뷰에 관련된 사항은 끝났고 시현에 대한 대기나 혹시 모를 수정에 대한 대기만 하고 있었다.

2) 조현태

Figure 47. Changed robot suspension

Figure 48. The robot lifted due to the fixation of robot suspension



3) Zulfiqar

(1) Circuit Problem Resolved

The voltage output directly from the battery to the NUC was supplied initially. The voltage output was of about 23.96 V when measured using multimeter. Hence the only problem according to our analysis was the high voltage input to the NUC which was causing it to fault. Hence, I used a 19 converter for NUC and installed in into our circuit to resolve the problem. Now the NUC works perfectly fine.

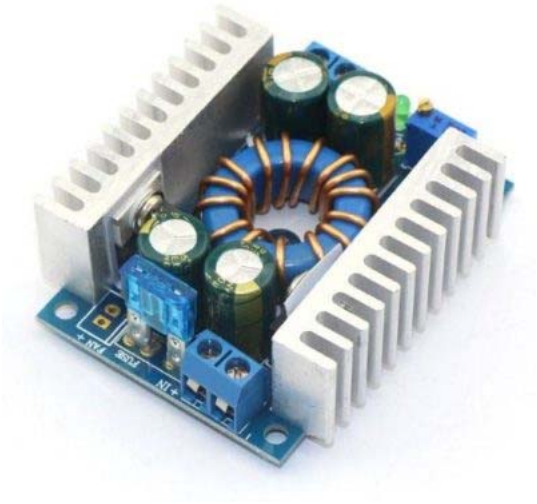


Figure 49. New converter installed into the circuit

(2) Finishing touches and arrangement of devices

After I setup the circuit I made sure the devices were installed in fixed position so that when the system moves the devices do not detach. Hence, I fixed the devices and assembled the large amount of wires all around the system in a mannered way.

4) 장민석

지난주에 수정한 기어박스를 시험해보기 위해 xbox controller로 로봇의 움직임을 확인해보았다. 서스펜션은 안정적으로 작동했고, 오른쪽 횡방향 움직임을 제외한 나머지 움직임은 모두 부드럽고 정확했다. 문제는 오른쪽 횡방향으로 움직일 때 오른쪽 대각선 뒤로 움직인다는 것이다. 네 바퀴의 움직임을 관찰한 결과 오른쪽 뒷바퀴가 제대로 회전하지 않는다는 것을 알아내었다. 그래서 오른쪽 뒷바퀴를 정면으로 세우고, 바퀴가 잘 회전하도록 하기 위해 살짝 휘어진 기어박스를 바로 세우고자 하였다. <Figure 50>에서 볼 수 있는 기어박스의 회전축을 연결하는

구멍이 수직이 아니어서 이를 수직으로 다시 뚫으면 기어박스를 곧게 세울 수 있을 거라 생각하였다. 그래서 기존에 뚫려있던 구멍 옆에 좀 더 정확한 구멍을 다시 뚫어 기어박스가 뒤틀리는 문제를 어느 정도 해결하였다. 그랬더니 다행히 오른쪽 횡방향 움직임이 훨씬 정확해졌다. 또한 오른쪽 횡방향으로 움직일 때 <Figure 51>에서 볼 수 있는 기어박스 지지대에 기어가 부딪히는 것이 발견되어 줄을 사용하여 이 부분을 갈아주어 여유공간을 확보하였다. 그래서 바퀴가 제대로 회전하지 않는 문제도 해결할 수 있었다.

<Figure 52>를 보면 롤러에 흰색 막대기가 추가된 것을 볼 수 있다. 롤러의 크기를 키우지 않으면서도 공을 안정적으로 픽업하고 릴리즈하기 위해 추가하였다. 이 막대기가 없을 때는 롤러의 회전에 의해 공이 빨려 들어가지 않고 공이 롤러와 회전하기만 하였다. 이 막대기를 추가해주니 회전하기만 하고 있는 공을 쳐주어 공이 성공적으로 픽업되고 릴리즈할 수 있게 되었다.

다음 주에는 발표 준비를 위해 피피티를 제작할 것이고, 하드웨어 팀에서는 데모 연습을 하며 발생할 수 있는 문제를 수시로 해결하기 위해 데모장에서 소프트웨어 팀을 보조하는 역할을 할 계획이다.

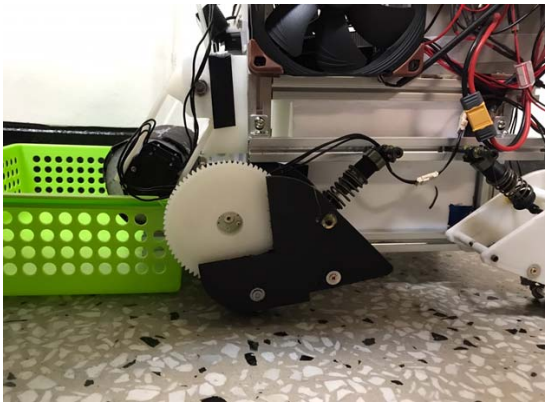


Figure 50

Figure 51

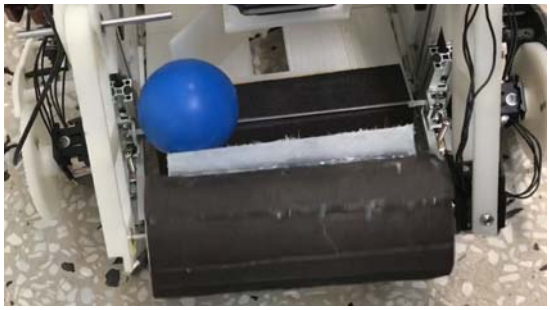


Figure 52

5) Farhad

Some modifications were made to the roller in order to be more efficient during the pickup and release parts of the mission. The location where the holder is fastened to the main frame of the vehicle as well as the roller arm length were modified to get an optimum distance from the floor to the car as well as the angle between the roller arms and the aluminum profile of the main frame.

A hole on one of the gearboxes for a shaft fixing the gearbox to the main frame of the car was drilled again since it was not perfectly aligned with the hole in the main frame of the car. This would cause the gearbox to be tilted, which, in turn, would lead to vibrations during the motion. The drilling machine was used for this purpose.

Furthermore, we tested how our robot behaves in a different environment, namely the floor of the auditorium room where the demo will be held. A particular focus was on how the suspension would work on an uneven ground of the auditorium.

Finally, we started our preparations for the final presentation.

3. ROS/System Integration and Machine Learning

Week4:

1) 박채리

CNN code review

- CNN_test.py and transfer_learning_tutorial.py

	torch , torch.nn, numpy : PyTorch 로 신경망 처리를 위한 필수 패키지 torch.optim for gradient descent PIL , PIL.Image , matplotlib.pyplot (이미지를 읽고 보여주는 패키지) torchvision.transforms (PIL 타입의 이미지들을 토치 텐서 형태로 변형해주는 패키지) torchvision.models (사전 훈련된 모델들의 학습 또는 읽기 패키지) copy for deepcopy with model
	data 를 전처리하는 과정이다. compose 는 여러 transform 들을 chaining 한다. train 에 있는 사진들만 random 하게 자르거나 가로로 접고 tensor 형식으로 바꾼 뒤 image data 들의 mean, std 값으로 normalization 을 거친다. accuracy 측정을 위한 validation set 은 image 의 사이즈를 256 으로 맞추고 가운데 부분만 잘라 tensor 형식으로 바꾼 뒤 train 과 같은 값으로 normalize 를 거친다.
	현재 위치에 있는 catdog_data 폴더 안에 있는 image 사진들을 불러와 16 의 batch size 로 나누고 섞는다. 여기서 num_workers 는 data 로딩을 위해 subprocess 의 개수를 지정해준다. (CPU 의 작업을 GPU 로 빨리 넘겨주기 위해 여러 개의 CPU 코어를 할당해준다.) 그리고 dataset_sizes 변수에 image 의 개수를 지정하고 train 안의 class 들(cat, dog)를 class_names 변수에 지정한다.

```
def visualize_model(model, num_images=9):
    was_training = model.training
    model.eval()
    images_so_far = 0
    fig = plt.figure()

    with torch.no_grad():
        for i, (inputs, labels) in enumerate(dataloaders['v
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)

            for j in range(inputs.size()[0]):
                images_so_far += 1
                ax = plt.subplot(num_images//3, 3, images_s
                ax.axis('off')
                ax.set_title('predicted: {}'.format(class_n
                imshow(inputs.cpu().data[j])

                if images_so_far == num_images:
                    model.train(mode=was_training)
                    return
        model.train(mode=was_training)
```

function visualize_model

-model : evaluation 에 사용되는 model, num_images 는 보여주고 싶은 image 개수

-일부 이미지에 대한 예측값을 보여주는 함수이다.

model 을 evaluation mode 로 설정해준 후 새로운 figure 를 만들어 창을 만든다.

with torch.no_grad 를 통해 auto gradient engine 을 deactivate 하고 memory 를 사용을 줄여 계산 속도를 빠르게 해준다. 그 후 val 폴더 안에 있는 data 들을 불러와 input 에는 사진을, label 에는 class 를 저장하고 model 에 사진을 넣어 최대 값을 갖는 class 의 index 를 pred 에 저장한다.

image 의 개수만큼 for loop 을 통해 한 장씩 print 할 때마다 images_so_far+=1 를 해주고 image plot 은 하나의 row 에 3 개의 image 가 들어가게 하고 predicted 된 class 를 제목으로 하여 imshow func 에 image 들을 하나씩 넣는다. for loop 중에 images_so_far 이 처음에 설정해준 num_images 와 같아졌을 때 model 을 visualize 하기 전으로 돌려주고 return 한다. 만약 num_images 가 val 안의 image 개수보다 클 경우 return 하지 못하고 model 을 돌려주고 끝난다.

Table 1

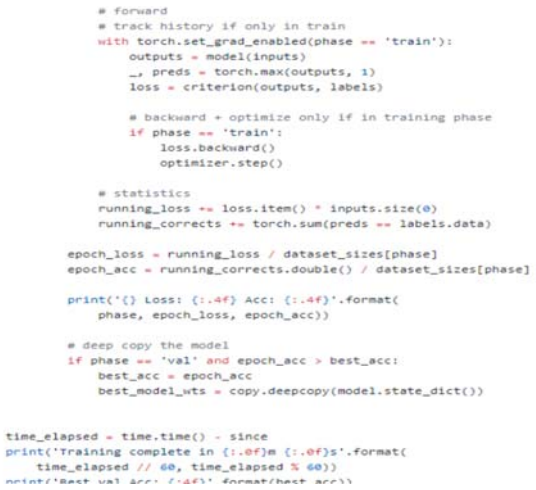
```
# Now, let's write a general function to train a model. Here, we will
# illustrate:
#
# -  Scheduling the learning rate
# -  Saving the best model
#
# In the following, parameter ``scheduler`` is an LR scheduler object from
# ``torch.optim.lr_scheduler``.

def train_model(model, criterion, optimizer, scheduler, num_epochs=25):
    since = time.time()

    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0

    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))
        print('-' * 10)

        # Each epoch has a training and validation phase
        for phase in ['train', 'val']:
            if phase == 'train':
                scheduler.step()
                model.train()  # Set model to training mode
            else:
                model.eval()   # Set model to evaluate mode

            running_loss = 0.0
            running_corrects = 0

            # Iterate over data.
            for inputs, labels in dataloaders[phase]:
                inputs = inputs.to(device)
                labels = labels.to(device)

                # zero the parameter gradients
                optimizer.zero_grad()
```

function training model

-scheduling learning rate

-saving best model

각 epoch 는 학습과 검증 단계를 가지므로 앞에서 정의된 phase 에 따라 train 일 경우 scheduler.step() (lr_scheduler.StepLR) 을 통해 every epoch 전에 parameters 를 update 해주고 model train 을 시작한다. 반면 validation 단계에서는 model

	eval 모드로 설정해준다. 그리고 나서 running loss 와 corrects 를 initialize 해준 뒤 dataloader 를 통해 phase 에 맞는 image 들과 label 을 불러온다. 그 후 param 의 gradient 를 initialize 한 뒤 시작한다.
	forward propagation 단계 torch.set_grad_enabled 를 통해 학습 시에만 연산 기록을 추적하도록 한다. torch.max(outputs,1)를 통해 가장 높은 값을 갖는(가깝다고 생각되는) class 의 index 를 pred 에 저장한다. 그리고 finetuning 에서 nn.CrossEntropyLoss()으로 정의된 criterion 함수를 이용해 loss 를 계산한다. train 단계인 경우 backward+optimazation 을 한다. backward 에서는 gradient 를 each param 에 대해 축적하고, optimizer.step 에서 current gradient 에 대해 param 을 update 한다. 따라서 이 때문에 epoch 시작할 때 step()을 부른 뒤 zero_grad()를 불러야한다. running loss 는 loss.item(avg loss over a batch of data)*inputs.size(batch_size)를 이용해 구하고 running_corrects 는 pred 와 label 이 같으면 1 씩 더하고 틀리면 0 을 더해 각각 data size 로 나누어 epoch_acc, loss 확률을 구한다. 그리고 phase 에 따른 acc, loss 를 print 한다.

	validation phase 에서 전에 저장되어있던 best_acc 와 current acc 를 비교해 더 큰 acc 와 model 을 best 에 저장한다. train 시작할 때 저장해 두었던 시간과 현재 시간을 비교해 train time 을 계산하고 train 후 최종 best 의 validation accuracy 를 print 한다. 그리고 가장 best accuracy epoch 의 model weight 을 불러와 return 한다.
	finetuning -pre-trained model(resnet18)을 불러온 후 마지막의 full connected layer 을 초기화한다. fc.in_features 를 통해 resnet18 model 의 마지막 단에서 출력 노드의 개수를 구해 pretrained-model 끝단에 fully connected layer 인 Linear 함수의 입력 노드에 넣고 출력 노드에는 class 개수인 2 로 설정해 노드를 연결시켜준다. 그 후 완성된 model 을 다시 device 에 넣어준다. train 에서 사용될 loss 함수는 crossentropy, optimizer 로는 stochastic gradient descent 를 사용하였다. lr 은 보통 0.001 과 0.01 을 사용하며, momentum 은 0.9, gamma 는 0.1 을 사용한다. 그리고 stepLR 에서 step_size 의 epoch 마다 0.1 씩 lr 을 감소시킨다.

Table 2

2) 오준영

Until week 4, software team worked for CNN, which is used for distinguishing whether the picture is dog or cat. For training, we downloaded the images of cats and dogs. We trained by using transfer_learning_tutorial.py which is in cnn_learning_code-master folder. After training, we get the

CNN_dogcat0810.pt file, which is used for validating whether the picture shown in camera is dog or cat. Also, there is a node named 'readmarker' which is in the package also named 'readmarker'. This node indicates 4 QR codes and cut the image in the range of these 4 codes. Catdog_cnn_network.py is used to show the result of validation. By running python files and the node, we can get the result like below.

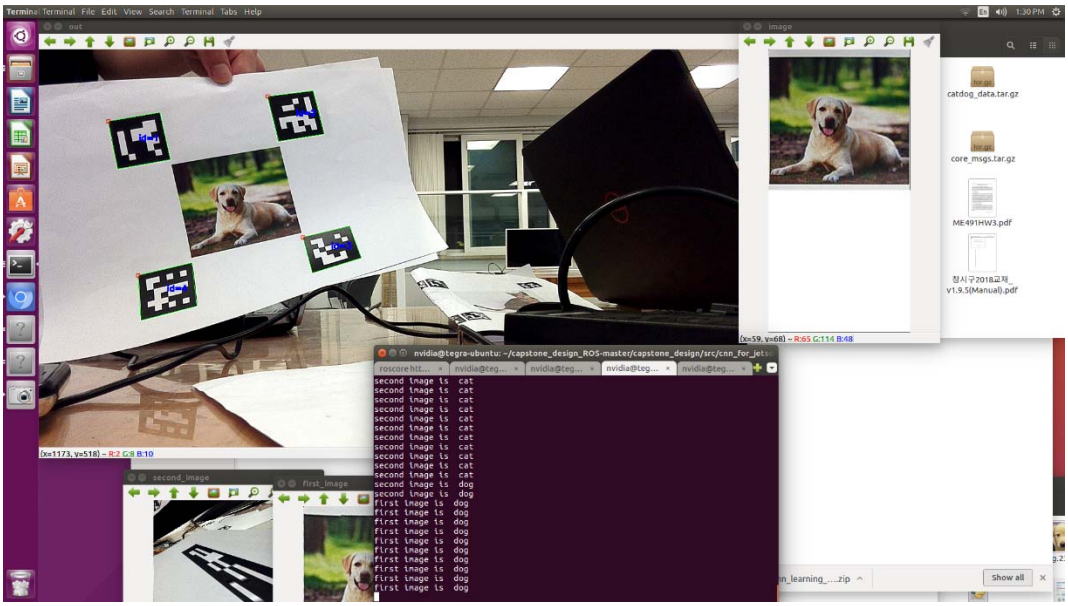


Figure 53. Result of CNN

The problem we had that it took too long to indicate after changing the picture. This problem was solved by changing code in catdog_cnn_network.py. The initial code had cv2.watikey(500) which was too big. We changed 500 to 50, and we could get the immediate response after changing the picture.

Also, we tried to change the step size, value of gamma. The accuracy for not changing the values was 98.6 %, and for the two other experiments changing the parameters, we got the same accuracy. We concluded that the resnet18 function is accurate enough and changing those parameters just a bit doesn't affect a lot.

For the next week, we are going to try other methods for higher accuracy. Also, we are going to work on SLAM which is used for getting out of the maze.

3) 이승언

(1) Machine Learning 및 Convolutional Neural Network(CNN) 공부

처음으로 machine learning 과 CNN 에 대해 배웠고 실습해 보았다. 우선 비교적 간단한 MNIST data 를 이용해서 기본적인 CNN model train 을 시켜 보았으며 모델의 구조는 다음과 같다.

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(28*28, 50)
        self.fc1_drop = nn.Dropout(0.2)
        self.fc2 = nn.Linear(50, 50)
        self.fc2_drop = nn.Dropout(0.2)
        self.fc3 = nn.Linear(50, 10)

    def forward(self, x):
        x = x.view(-1, 28*28)
        x = F.relu(self.fc1(x))
        x = self.fc1_drop(x)
        x = F.relu(self.fc2(x))
        x = self.fc2_drop(x)
        return F.log_softmax(self.fc3(x))
```

Figure 54

또한, Cifar 10 데이터를 이용해서 조금 더 복잡한 model 을 학습시켜 보았으며 그 model 의 구조는 다음과 같다.

```
class MyClassifier(nn.Module):
  def __init__(self):
        super(MyClassifier, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3)
        self.conv12 = nn.Conv2d(in_channels=32, out_channels=32, kernel_size=3)
        self.batchnorm1 = nn.BatchNorm2d(num_features=32)
        self.relu = nn.ReLU()
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        #self.dropout1 = nn.Dropout(0.2)
        #self.dropout2 = nn.Dropout(0.3)
        #self.dropout3 = nn.Dropout(0.4)
        #self.dropout4 = nn.Dropout(0.5)
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3)
        self.conv22 = nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3)
        self.conv3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3)
        self.conv32 = nn.Conv2d(in_channels=128, out_channels=128, kernel_size=3)
        self.batchnorm3 = nn.BatchNorm2d(num_features=128)
        self.conv4 = nn.Conv2d(in_channels=128, out_channels=256, kernel_size=1)
        self.fc1 = nn.Linear(in_features=256 * 1 * 1, out_features=256)
        self.fc2 = nn.Linear(in_features=256, out_features=output_dim)

  def forward(self, x):
      x = self.pool(self.relu(self.batchnorm1(self.conv12(self.conv1(x)))))
      x = self.pool(self.relu(self.conv22(self.conv2(x))))
      x = self.pool(self.relu(self.batchnorm3(self.conv32(self.conv3(x)))))
      x = self.pool(self.relu(self.conv4(x)))
      x = x.view(-1, 256 * 1 * 1)
      x = self.relu(self.fc1(x))
      outputs = self.fc2(x)
      return outputs
```

Figure 55

(2) Cat and Dog CNN training

우리가 최종적으로 만들어야 할 Cat and Dog classifier 를 resnet18 이라는 pretrained 된 model 을 이용해 25000 장의 data 를 바탕으로 추가로 training 시켰으며 10 분 12 초 만에 0.9868 의 accuracy 를 가지는 모델을 학습시킬 수 있었다. Training 과정에서 train loss, train accuracy, validation loss, validation accuracy 의 변화를 그래프로 나타내 보았고 다음과 같다.

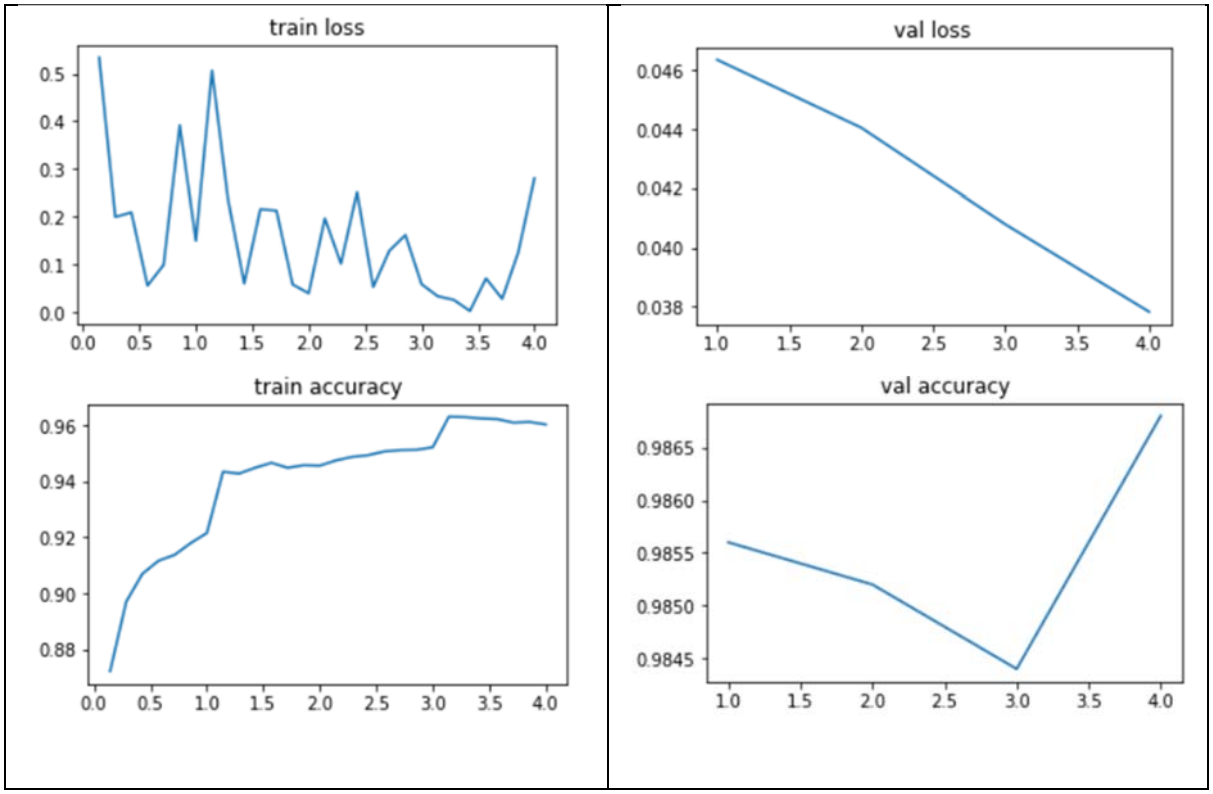


Table 3

(3) Train 된 데이터를 통한 test

앞서 train 한 model 을 이용해 실제로 webcam 으로 보여진 사진을 제대로 판별 할 수 있는지 확인하였고 노드들의 실행과정에서 많은 에러가 발생하였지만 모두 해결하여 다음과 같이 성공적으로 판별하는 것을 확인하였다.

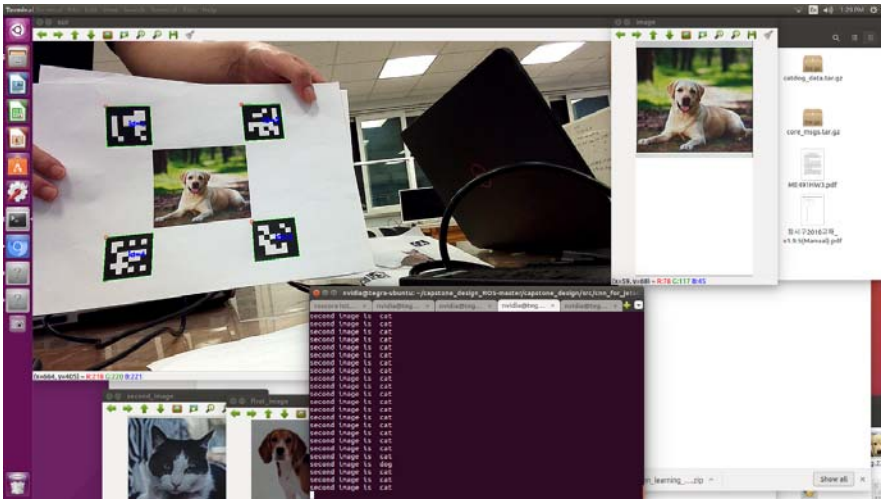


Figure 56

4) Farhad

The first week involved successful installation of Linux operation system with dual booting. Once this was achieved, I started getting familiar with the new OS as well as installed the ROS kinetic package and Anaconda.

The version of Linux OS installed on the computer was Ubuntu 16.04. The purpose of using Ubuntu is the ease with which certain tasks related to our project could be performed. Anaconda is a free and open-source distribution of Python and R programming languages for scientific computing. Command line installation method was utilized for Python 3.7 version of Anaconda. With Anaconda ready to be used, an Anaconda environment was created. Environments are convenient to use since everything I do there can remain there without affecting other software and packages. Afterwards, Robot Operating System (ROS) was installed. It provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. Since the project involves machine learning applications, GPU version of PyTorch was installed as well. PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing. GPU version was required in order to activate the GPU during heavy CPU processes like model training. This resulted in much faster training times.

Once the aforementioned prerequisites were completed, I developed my Python skills and learned the basics of PyTorch. This was necessary before starting practicing Convolutional Neural Networks (CNN). Firstly, I studied the theory of CNN. In deep learning, a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery. CNNs are regularized versions of multilayer perceptrons. CNN has a similar visual purpose for our project: detecting and differentiating cats and dogs, and depending on the differentiation, deciding which colored balls to drop in a particular basket. I also compared PyTorch with other deep learning frameworks, like TensorFlow and Keras, to see which one is more efficient and suitable for our tasks. I concluded that Tensorflow would be better for use in the project. However, since PyTorch is better supported by the project assistants, I continued studying PyTorch. I trained many models and tried to see the differences some parameters, like learning rate, batch size, preprocessing method, and epoch size, make and how they affect the final output.

Week5:

1) 박채리

(1) SLAM

수업 때 gazebo 를 이용해 m_robot_gazebo 에서 map 을 가져오고 find frontier package 의 nav_run launch 를 통해 가장 먼 방향의 좌표를 find frontier 에서 구해 move base 로 보내면 값을 계산해서 cmd vel 라는 토픽으로 gazebo 로 보내면 안의 robot 이 움직이는 것을 알 수 있었다. 실제로 rplidar 을 이용해 hector slam package 를 통해 rviz 에 map 을 그리는 것을 했을 때 rplidar 의 scan 토픽을 hector mapping 에서 받지 못하는 문제가 있었는데, tutorial.launch 파일의 use_sim_time 을 false 로 하여 simulation time 이 아닌 실제 시간을 이용하도록 수정하고 mapping_demo.launch 에서 base frame 과 odom frame 을 base link 로 바꾸었고 마지막 tf node 를 켜는 것을 바꿔서 해결하였다.
마찬가지로 find_frontier 의 nav_run 파일을 고쳤는데, hector_mapping 을 불러오면 error 가 생겨 mapping_demo 를 불러오도록 했더니 잘 되었다.

(2) Jetson, NUC 연결

NUC 과 jetson 의 bashrc 에서 rosmaster 를 쓰도록 설정하고(Ssh –X nvidia@~~~) NUC 에서 rosmaster 를 이용해 jetson 의 IP 를 받아와 둘을 연결하여 NUC 에 연결된 webcam 에서 받은 image data 를 jetson 의 cnn python 파일로 보내 개고양이 사진을 분류한다.

2) 오준영

For this week, we tried to modify our slam code. Our slam code had some problems if we launched them without modifying. By googling and learning the code, we could find that we should change some parameters and arguments. By running code from Hector slam, it worked but using launch file in find frontier file didn't work, so we changed them referring to Hector slam's code. Also, we changed parameter '/use_sim_time' from true to false in nav_run.launch file, which is using the simulation time.

By changing these parameters and arguments, we could do mapping by using launch file from find_frontier, after running rp_lidar. We checked that mapping works, but could not find out whether it publish the cmd_vel which is used for moving our vehicle.

What we are doing now is to get the cmd_vel, and use data[] to give signal to LabView, which will be main for running our motors.

For CNN, we checked that getting image from Nvidia camera and sorting whether it is cat or dog worked well. We wanted to get image from camera which will be connected by NUC, so we used ssh, which is used to connect NUC and Nvidia. We tried wireless connection, but we found that it is very slow, so we tried to use by connecting them with local cable, and found that it works much faster.

3) 이승언

(1) Nuc 과 Jetson 연결

저번주까지는 Jetson 안에서 모든 노드를 실행시켜서 CNN 이 정상적으로 작동하는 것을 확인하였다. 이번 주에는 Nuc 에 연결되어 있는 webcam 에서 이미지를 받아 Jetson 으로 넘겨 준 후 Jetson 에서 이미지를 판별하는 것을 해 보았다. 방법은 로스 마스터를 사용한 것으로 Jetson 과 nuc 의 bashrc 에서 로스 마스터 설정을 해 준 후 ssh –X nvidia@~~~라는 명령어를 이용하여 nuc 과 Jetson 의 연결이 잘 됨을 확인하였다.

(2) Slam 공부 및 cmd_vel 확인

TA Session 에서 배운대로 가제보를 통해 슬램의 작동원리에 대하여 공부하였고 이를 실제 라이더에 적용시키는 것을 시도하였다. 하지만 여러가지 문제가 발생하였다. 먼저 lidar 의 /scan 토픽이 hector_mapping 노드로 들어가야 하는데 들어가지 않았고 다른 방법으로 하면 move_base 노드는 아예 실행되지 않았다. 여러가지 방법을 고민하던 중 nav_run.launch 파일에서 hector_mapping.launch 를 불러오지 않고 mapping_default.launch 와 geotiff_mapper.launch 를 불러오는것을 바꾸었더니 rqt_graph 에서 정상적으로 모든 노드들이 연결되는 것을 볼 수 있었다. 그러나 rostopic echo cmd_vel 을 해 보았더니 cmd_vel 에는 아무것도 나오지 않았다. 다시 가제보를 통해 여러 번 실험을 해 본 결과 cmd_vel 명령이 가도 로봇이 움직이지 못하는 상황이면 cmd_vel 값이 나오지 않는 것을 관찰 하였다. 따라서 라이다를 로봇에 단 후 직접 실험해 봐야겠다는 결론을 가질 수 있었다.

4) Farhad

In this week, me and my team continued training CNN models, saved those models, integrated the NVIDIA Jetson TX2 module with a camera, and tried to see and evaluate predictions of the trained models with the NVIDIA GPU module connected to a monitor via an HDMI cable.

We tested the models with multiple cat and dog images encompassed by four QR codes. First issue that was observed and tackled was a significant delay while switching the dog and cat pictures. It would take some time before it could be determined if the new image contains dog or cat. The problem was solved by modifying the code running for outputting model's decision. As a result, the delay was reduced fiftyfold from 500 to 10. The unit was not milliseconds, since 500 ms would mean 0.5 seconds, but the delay we had experienced before was around half a minute. After the modifications, the computer would instantly respond to switching the image.

In addition, after some modifications we decided to use batch size and step size of 64 and 2, respectively. Before this, the step size was 4, and since the number of epochs was also 4, the step size was basically useless. Moreover, as there were detection issues with some dog breeds like Pomeranian, we considered adding pictures of some dog breeds that could potentially be mistaken for a cat.

For faster data transmission, it was decided to access Jetson with NUC through a LAN cable with IP address. This led to faster transmission times, which is a necessity in the project due to time limitations.

Following this, our team investigated SLAM principles and its execution. All the necessary installations were performed including ROS, Gazebo, and Hector SLAM. Afterwards, we simulated the maze and how the vehicle tackles finding frontiers and escapes the maze.

The next step was achieving the same performance in the real world. For that, the sensors, hardware, and circuitry had to be assembled. There were some discussions with the hardware team for successful execution of the tasks. One of important tasks is integration of LIDAR in the vehicle.

We have also discussed how to modify the codes for the mecanum wheels, which can move in any direction. This property requires additional considerations which will be investigated in the near future.

Week6:

1) 박채리

(1) SLAM

기존에는 차의 서스펜션이 불안정해 좌우 대각선 운동이 잘 되지 않을 것이라 생각하여 앞 뒤, 회전 운동만을 하였는데 parameter 들을 변경해 보아도 미로를 통과하는 시간도 오래 걸리고 벽에 자주 부딪히는 문제가 있어서 회전운동을 빼고 좌우 운동을 넣었더니 10 초 정도로 안정하고 빠르게 통과할 수 있었다.

(2) DQN simulator

기존에는 맵이 랜덤하게 인풋되었지만 실제 우리 demo 환경에 맞추어 3*2.5m 로 맵을 인풋해주었다. 또한 기존에는 가까운 공은 지나가고 거리가 있지만 공들끼리 모여있는 곳에 가장 먼저 움직였기 때문에 이를 해결하고자 공 사이의 거리를 멀게 하고 벽과의 거리도 떨어지도록 해주었다.

2) 오준영

This week, we mainly worked on running SLAM. We had found that our slam does do mapping, but could not publish cmd_vel. After changing some files, we could finally get cmd_vel and make our vehicle move. However, parameters were not good, and robot stopped when it was close to the wall. We tried changing parameters. First, we change footprint values, which means the size of the vehicle. Also, we changed maximum velocity because we just had to escape the maze for the first presentation, no need to do the task fast. Then we lowered the obstacle range, which made our vehicle still moving even the wall is close.
We also changed other parameters and did some simulations by Gazebo. After many attempts, we could finally succeed escaping maze. We also checked that for more complicated maze, our slam code worked well.

Also, we adjusted some data_integration node for CNN. Array named ‘data’ is used to send message to LABVIEW. Index 0 to 3 are used to define wheel velocity, which enable us to move our vehicle with desired translation and rotation speed. Index 4 is used for roller located at front of our vehicle. It will choose its rotating direction according to the value. Index 5 is used for roller lifting the ground cloth. Index 6 is used for sorting bar.

If string message 'dog' is received, sorting bar will rotate CW to open right side. Roller located in front will rotate CCW for release mode, and the ground cloth will be lifted. If string message 'cat' is received, sorting bar will rotate CCW to open left side.

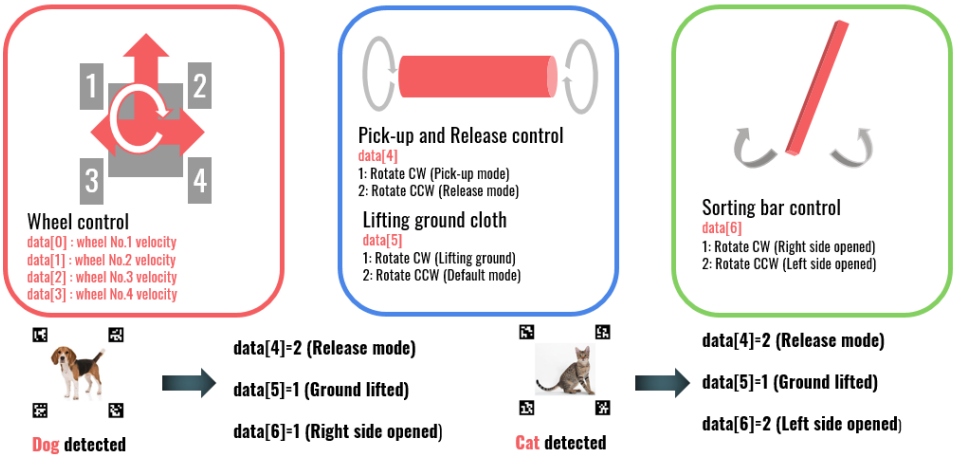


Figure 57

In the given CNN code, we just can use simple data augmentation such as normalizing, resizing, cropping, and rotating. For better training, we will going to use upgraded version which have Gaussian blur function. In this case, we can also handle situations getting blurred image from camera.

On Friday, we did our first presentation. We could briefly see how other groups did their works. For better design and output, we are going to make some feedbacks on next Monday.

3) 이승언

(1) CNN 으로 하드웨어와의 연결

우리가 이전 주에 짠 CNN 으로 개와 고양이 사진을 판별하는 알고리즘을 하드웨어에 적용하여 개 사진을 보여주면 파란색 공을 뱉고, 고양이 사진을 보여주면 빨간색 공을 뱉는 알고리즘으로 완성시켰다. 먼저 고양이 사진을 보여주면 가운데의 sorting 부분이 빨간색 공을 open 시키고 공을 담고 있는 부분의 천이 올라가며 롤러가 반대방향으로 돌아 빨간색 공이 밖으로 나오게 된다. 이어서 개 사진을 보여주면 sorting 부분이 파란색 공을 open 시키고 파란색 공이 마찬가지로 밖으로 나오게 된다.

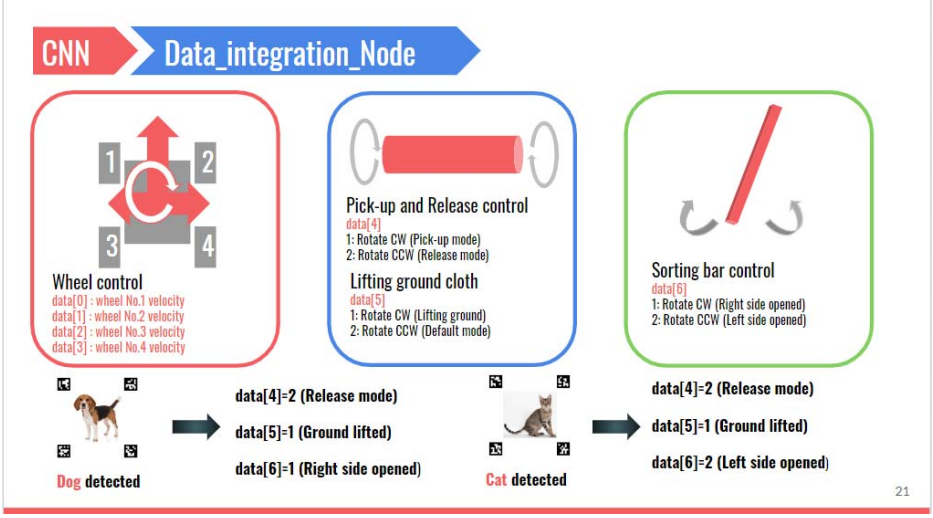


Figure 58

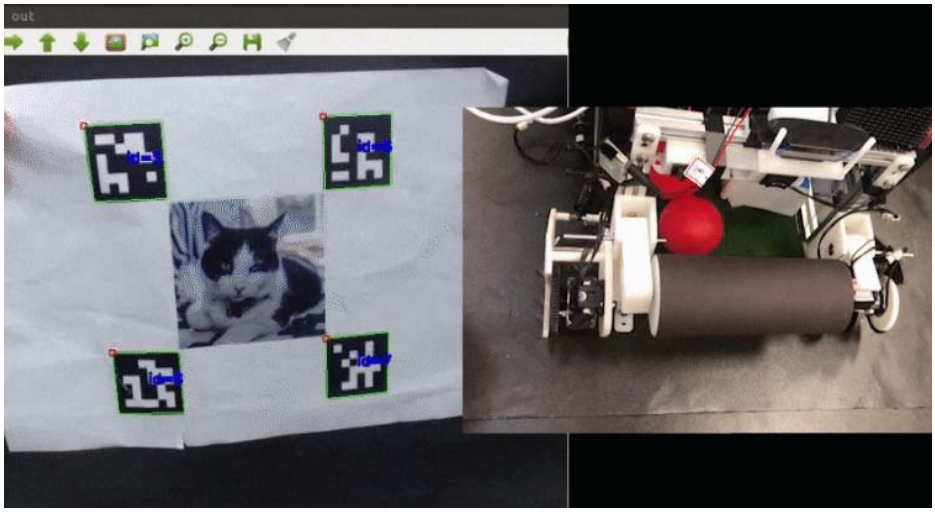


Figure 59

(2) SLAM 으로 미로 탈출 시키기

우리가 SLAM 을 사용하는 첫 번째 목표인 미로를 탈출하기 위해 SLAM 코드의 파라미터들을 조절하여 우리의 로봇이 미로를 탈출할 수 있게 하였다. 조절해야 할 파라미터들의 수가 너무 많기 때문에 처음부터 바로 로봇에 적용할 수는 없었고 따라서 GAZEBO 를 이용하여 여러가지 파라미터들의 성격을 알아본 후 우리의 로봇과 비슷한 크기의 가상 로봇에 실험 해 보고 그 최적값을 실제 우리 로봇에 적용하였다. 그 후에 실제 상황에 조금 더 최적화하여 그 값들을 완성하였다.

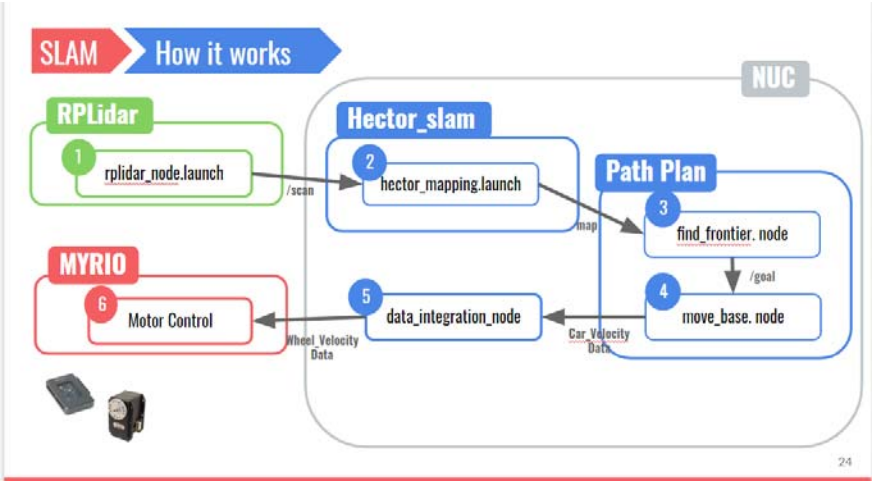


Figure 60

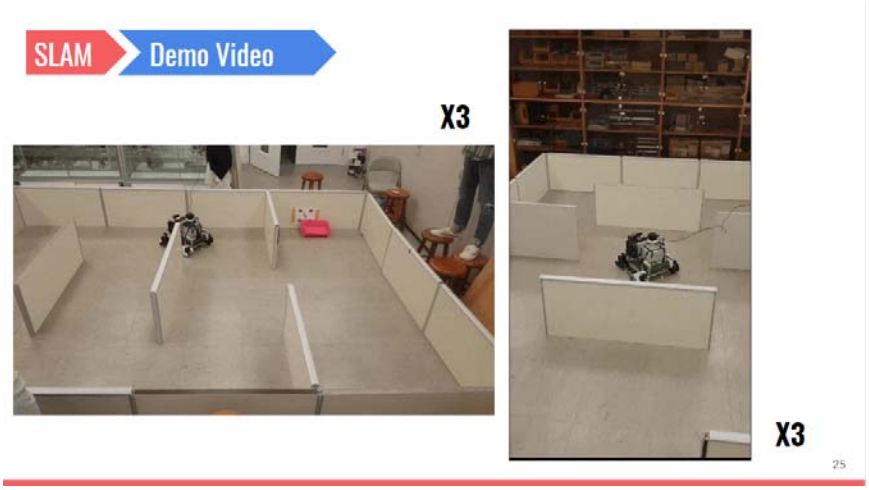


Figure 61

Week9:

1) 박채리

DQN 을 하기 위해 dqn_learning_code 의 전반적인 code 를 살펴보았다. 우선 main.py 에서는 dqn_model, dqn_learn, schedule, simulator 파일들을 import 해서 main 을 실행시켰을 때 학습이 실행되도록 하는 코드이다. 여기서 batch size, alpha, learning rate 등의 값들을 설정해줄 수 있다. 또한 debug_flag 를 통해 중간 중간의 영상을 확인할 수 있다.

Dqn_learn.py 에서는 100 episode 마다 평균 episode reward 값과 best episode 의 값을 비교해 더 큰값을 best mean episode reward 값으로 하고 화면에 현재의 time step 과 mean reward, best mean reward 와 함께 현재 얼마나 학습되었는지 episodes 개수를 print 하고 weight 값을 .pt 파일로 저장한다.

Dqn_model.py 에서는 dqn model 의 layer 를 설정해주는 파일이다. Convolutional layer 가 3 개, 그리고 linear 함수 두개와 forward propagation 을 위한 relu 함수 4 개가 쓰인다.

다음은 simulator.py 파일로, 여기서 reward 방법 및 전체적인 system 을 설계하는 파일로써 가장 많이 수정해야할 파일이다. 먼저 simulator 에 쓰일 wall 들을 만든다. 그리고 ball 들을 random 하게 생성하며, ball 이 wall 밖에 있으면 없앤다. 그 다음에 vehicle 을 만드는데, 차의 전체적인 크기를 표시하는 것은 빨간색, 차의 중심부를 표시할 때는 파란색 한 칸으로 나타낸다. 여기서 설정된 resolution 을 통해 한 칸은 5cm 인 것을 알 수 있고, 공은 한 칸 크기이므로 원래는 7cm 인 공이지만 5cm 로 했다는 것을 알 수 있다. 그리고 차체가 벽을 통과하면 안되기 때문에 벽을 넘어가지 못하도록 설정되어있다. 그 다음은 reward 방법인데, 차의 가운데로 공을 먹으면 reward 는 7 점, 그 옆 칸은 3 점, 마지막 칸은 1 점으로 되어있다. 또한 차가 움직일 수 있는 총 action 의 개수는 10 개로, 앞 뒤 좌 우와 함께 제자리 시계방향, 반시계 방향 회전과 앞과 뒤로 가면서 오른쪽 회전, 왼쪽 회전이 있다.

2) 오준영

For this week, it was a week after the midterm exam, so we didn't do something big, we just organized our schedule.

For the software team, we could do some experiments only after when hardware te am finish their work. Hardware team will going to finish their work until next week and then software will be done.

After the hardware is made, first thing we should do is change some parameters for SLAM. We just set our parameters to succeed the task, and the speed is really slow. To solve this problem and get some good

result on demo, we should do some more experiments on SLAM and find out which parameter we should change, and what are the best values for them to reduce our time.

Also, the challenging thing is DQN. We are taking classes about it and the basic lecture for DQN will be finished at Nov 4th. After the class, we will study more about the code and adjust it properly for our device.

3) 이승언

(1) RL 및 DQN 공부

공을 최대한 빠른 시간 안에 모두 줍기 위해 사용하는 RL 과 DQN 에 대하여 공부하였다. 우리는 기본적으로 Reinforcement learning 기법을 사용하지만 워낙 경우의 수가 많고 복잡하기 때문에 RL 에 CNN 을 접목한 Deep Q-Network 를 사용하게 된다. CNN 은 이전에 공부하였던 것이기 때문에 RL 에 대해 좀 더 공부하였다.

Week10:

1) 박채리

(1) DQN

① Map

우리의 차와 맞게 simulator 을 바꾸었다. 우선 차의 size 인 50cm 로 바꿔주었다. 그리고 공을 먹는 범위는 옆쪽으로 먹으면 공이 끼거나 제대로 차가 먹지 못할 수 있기 때문에 늘리지 않고 그대로 5 칸 이내에서 먹도록 하였다. 그리고 벽을 통과하지 못하도록 벽을 넘지 못하는 값을 더 키워 주었고, action 또한 앞뒤와 함께 제자리 회전까지 해서 총 4 가지만을 하도록 하였다. 좌우로 움직일 때 오차가 크기 때문에 그를 생략하였다. 또한 공은 차가 움직이면서 치지 않는 이상 벽과의 거리가 50cm 이상이기 때문에 벽과의 거리가 일정 이상이 되도록 해주었다.

② Reward

Simulator 를 돌렸을 때 차가 움직이지 않고 제자리에 있는 경우가 많았고, 살펴보니 simulator 에서 했을 때는 score 가 step 이 증가할수록 1 씩 감점이 되었는데, reward 는 감점이 반영되지 않아 처음에는 -0.5 로 하였다가 -1 로 바꾸었다. 또한 공을 먹을 때 reward 도 기존 7 에서 30 으로 바꾸었고 나머지도 각각 20, 10 으로 바꾸어 공을 꼭 먹도록 바꾸었다.

->Run file 을 tensorboard 로 확인이 잘 되지 않아서 그 점을 해결해야 하고 컴퓨터를 원격으로 조정하는 법을 배워 더 자주 학습시킬 수 있도록 해야겠다.

(2) RGBD Camera

dqn 에 사용될 map 을 그리기 위해 공을 정확하게 detect 해야 하므로 우리는 기존의 webcam 이 아닌 rgbd camera 를 사용하기로 했다. 우리가 사용한 카메라는 intel 의 d532i 로 이를 linux 에서 사용하기 위해 real sense package 등을 다운받았다. 그리고 launch file 을 실행시켜 rgbd camera 를 실행시키는 방법을 알 수 있었다. 그리고 ball detect node 에서 image 파일과 함께 depth image 를 같이 subscirbe 하여 image 에서 찾은 공의 중점 좌표를 depth image 에서 찾아 공의 거리를 찾으려 하였는데, 우선 launch 하였을 때 publish 되는 topic 들 중 sensormsgs :: Image 형식인 camera/depth/image_rect_raw topic 을 받아 cv image 로 바꾼 뒤 공의 x, y 좌표를 찾아 그 값을 print 하도록 하였는데 실제 거리 값과 다르게 매우 크거나 작은 값이 나와 다른 topic 이나 depth array 를 가져올 수 있는 방법을 생각해봐야한다.

2) 오준영

This week, we learned how to use DQN so we started our job. For DQN, we should have to change our simulator and some parameters to make it fully optimized for our vehicle. The biggest problem is that it takes a lot of time to train a model. So it takes a lot of time to check whether our result is useable or not. We tried three times and found that the vehicle in the simulator stops a lot. Instead of picking ball, it sometimes prefer just stopping. To fix this problem, we tried changing the reward values, and waiting for the result.

Another issue is camera. For DQN, we need accurate ball position, and the webcam we used give inaccurate coordinates. To handle this, we decided to use depth camera, which can give accurate coordinates for balls. However, the node and form is different from what we are given, and we have some difficulty getting this depth image coordinate values. We are still working on this problem.

The main problem is that if we try to fix something, even it is a small problem, it takes too much time to get the result. Also, comparing with Capstone design I, debugging is nearly unavailable which makes us difficult to indicate what to change. We are considering remote control to save time. When remote control is available, we can change and try more often.

For the SLAM, our vehicle is being changed, so we could not try changing some parameters. After when our vehicle is fixed, we are going to work on SLAM for faster movement.

3) 이승언

(1) 코드 분석

저번 주에 공부한 RL 과 DQN 을 바탕으로 구성된 dqn_learning_code 를 분석하고 공부하였다. 이 중 우리가 training 할 때 실제로 돌릴 코드는 main.py 이다. 하지만 이 파일 안에서 simulator.py 와 dqn_model.py, 그리고 dqn_learn.py 를 import 하여 사용하게 된다. simulator.py 에서는 키보드 인풋을 받아서 랜덤한 지도의 랜덤한 위치의 공들을 줍는 로봇을 만들게 된다. 이 simulator.py 를 바탕으로 모든 environment 와 state 이 정해지므로 실제 우리의 로봇과 실제의 환경에 맞춰 simulator.py 를 수정해 주어야 한다. dqn_model.py 에는 dqn 에서 사용할 모델이 정의되어 있고 dqn_learn.py 에서 모델이 학습되고 그 과정이 기록된다.

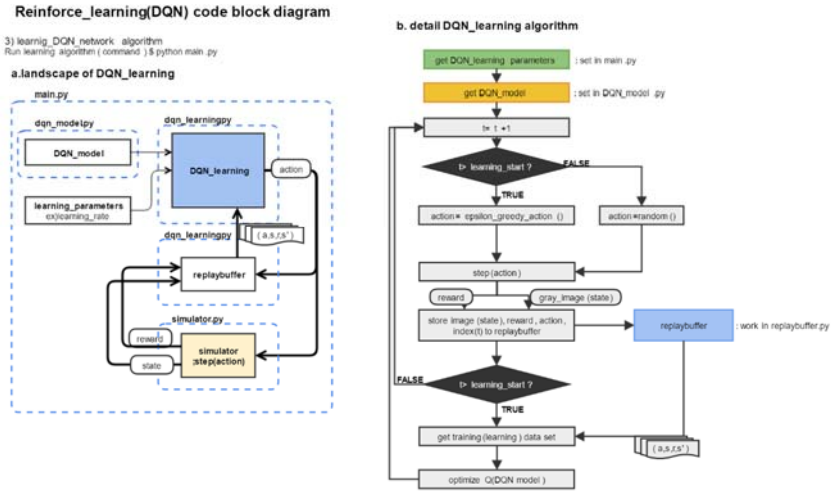


Figure 62

(2) 코드 수정 및 학습

우리의 로봇에 맞게 적절히 코드를 수정하여 학습까지 해 보았다. 먼저 우리의 로봇은 좌우 이동과 대각선 이동이 정확하지 않기 때문에 이를 제외하였다. 따라서 원래 총 10 개였던 action 을 직진, 후진, 왼쪽 회전, 오른쪽 회전 이렇게 네 개로 축소하였다. 확실히 action 의 수를 줄이니 수렴하는 score 가 낮아졌다. 또한 기존의 코드는 training 시에 시간이 지남에 따른 penalty 가 없기 때문에 학습에 시간이 반영되지 않았다. 따라서 우리는 매 step 마다 reward 를 -1 로 하여 시간에 따른 penalty 를 주기로 하였다.

Week11:

1) 박채리

이번주에 가장 대표적으로 한 일은 SLAM 알고리즘을 고친 것이다. 기존에 SLAM 에서는 직진운동 및 회전운동으로 장애물을 탈출하는 것이었는데 이 알고리즘으로는 parameter 를 바꿔서 속도를 최적화하기 매우 힘들었으며 계속 벽에 박는 현상이 발견되었다. 이를 해결하기 위해 우리는 직진+회전이 아닌 좌우 + 직진 운동을 사용하기로 하였다. 이 알고리즘으로 바꾼 이후 로봇의 움직임이 훨씬 단순해졌으며 벽으로 박는 현상을 피할 수 있었고 시간 역시 매우 단축할 수 있었다. 기존에 우리가 사용했던 SLAM 알고리즘이 약 27 초 걸렸다면 우리가 바꾼 최종 알고리즘으로는 약 10 초만에 장애물을 통과할 수 있었다.

또한 DQN 시뮬레이터를 바꾸었다. 기존에 제공된 시뮬레이터는 벽을 임의로 만들었는데 우리의 경우에는 벽이 항상 고정된 위치에 존재할 것이므로 우리의 상황에 맞는 벽을 만들어주었다. 또한 로봇이 시작하는 위치도 중앙이 아닌 좌 우 둘 중 하나로 랜덤하게 설정하였다. 이 외에도 공을 맵 안에 넣는 것도 제약조건을 주었다. 먼저 벽에서부터 50cm 떨어지지 않으면 추가하지 않는 코드를 넣었고 공과 공 사이의 거리도 50cm 이상 떨어져있을 때만 append 하게 설정하였다. 또한 learning 에 적합한 score 의 비와 gamma 값 등을 수정하면서 어느 경우에 우리의 robot 이 잘 작동할지 계속 learning 을 수행해보고 있다.

우리가 앞으로 해야할 일은 RGBD 카메라에 관련된 문제들을 해결하는 것이다. RGBD 카메라에서 depth 값을 불러오는 방법을 알아내야 하고 이를 해결했을 때 실제 ball position 으로 전환하여 확인해봐야 한다.

2) 오준영

The biggest thing we did this week is SLAM. Before, we used rotation and moving forward to escape the mirror. However the problem was that it was too slow, and also it was hard to find the parameters that will make this process faster. Every time we tried simulation with gazebo changing parameters, it bumped to the wall.

We thought that rotation is not essential for our design. By using x-y direction coordinate, it will be enough. After changing the basic idea and some optimization, we finally could get the time reduced to about 10 seconds, which is less than half of our previous version.

We also changed our DQN simulator. The simulator we got made wall randomly, which is too much for our demo, so we just set it to create wall of 1 version fit to our demo. Also, we changed the starting point, which can be the left side or right side. Furthermore, our balls will be placed 50cm each other and from the

wall , so we also changed this. We are trying changing some values such as scoring and gamma. We still have some problem so we will work on this more.

Also we started to use program named team viewer. This program enable us to remote the computer in the capstone design room, so we can train our model more frequently.

Next Monday, we will work with RGBD camera. After this is done, we will be able to do some experiments on our vehicle.

3) 이승언

(1) SLAM 최적화

이전에 하였던 SLAM 은 좌우 대각선 이동 없이 회전과 앞 뒤 운동만으로 구현하였다. 그랬더니 파라미터들을 아무리 열심히 조절 해 보아도 미로를 탈출하는 데에 20 초 넘는 시간이 걸렸다. 따라서 우리는 회전운동을 빼고 좌우와 대각선 이동을 추가하여 시도해보았다. 여러 번의 시도 끝에 10 초 정도로 시간을 축소할 수 있었다.

(2) Simulator 바꾸기

원래의 simulator 맵이 랜덤한 형태로 들어가게 되어 있다. 하지만 우리는 맵이 너비 3m, 길이 2.5m 의 직사각형 형태이므로 맵을 바꾸어 주었다. 또한, 공이 랜덤한 위치에 들어가므로 붙어있는 경우가 많았다. 그럴 경우에 로봇이 제일 가까운 공으로 가기 보다 뭉쳐져 있는 공으로 가게 train 되는 것을 확인 하였다. 따라서 우리는 공을 50cm 이상 떨어뜨려 놓고 뿐만 아니라 벽에서도 50cm 이상 떨어뜨려 놓았다. 또한, 공을 먹었을 때의 reward 와 step 이 진행될 때의 penalty 를 새롭게 지정해 가며 training 시켜 보았다.

Week12:

1) 박채리

Rgbd 카메라를 이용해 depth 값을 받아오는 것을 했다. 이 때 카메라를 실행할 때 rosrun realsense2_camera rs_camera aligh_depth:=true command 로 실행시켜 카메라와 depth 의 frame 을 일치시키도록 실행시켜야 한다. 그리고 받아오는 토픽들의 이름은 color/image 와 aligned_depth/depth 이고 image 에서 filter 를 통해 빨간 공과 파란 공을 각각 인식하고 그들의 pixel 값을 받아 depth 에서 같은 pixel 에 있는 depth 값을 받아오고, 그를 이용해 x, y 좌표의 calibration 을 하였다. 이를 이용해서 ball position node 를 완성하였고, 마찬가지로 read marker node 에서 depth 값을 받아와 마커 1, 2 번의 depth 값을 이용해 공을 release 할 때 align 할 수 있도록 하였다. 또한 일반 웹캠을 위한 ball_bottom node 도 만들어 롤러 앞에 있는 너무 가까운 공을 rgbd 카메라로 보지 못할 때 webcam 을 통해 공을 인식하도록 하였다.

cnn 에서 정렬하는 것은 marker 들의 x 와 거리 값을 이용했는데, 먼저 거리가 비슷하도록 회전을 시켜주었고, 그 다음은 두 마커의 x 값의 평균이 30 보다 클 때 좌우 이동을 하게 해주었고 만약 거리도 비슷하고 x 값도 30 이하일 경우 앞으로 가도록 해주었더니 속도를 느리게 했을 때 잘 되었다.

다음은 dqn 인데, pytorch 를 이용해 돌린 weight 값을 이용해 차를 움직여보기 위해 각각의 action 에 맞도록 각 바퀴에 들어갈 값들을 publish 하도록 해주었다. 또한 rgbd 카메라와 웹캠 카메라의 공 개수를 맞춰주기 위해 rgbd 카메라에서 보이는 공들은 전부 ball array 에 들어가도록 해뒀고 웹캠에서는 30cm 정도 안에 있는 공만 ball array 에 들어가도록 해주었다. (30cm 정도에서부터 rgbd 카메라로 detect 되지 않았다.)
또한 map 을 만들어줄 때 lidar 의 부호가 반대로 되어있어 부호를 바꿔주었고, 공들이 map 에 잘 들어갈 수 있도록 해주었다.

2) 오준영

This week, we finally solved the problem about RGBD camera. We could successively get the depth of the pixel, and convert it to the coordinate of the robot. By solving this problem, we could do lots of experiments.

First is aligning by markers. After we collect all the balls, we should store them in the basket. To successively do this task, alignment is very important. We set our algorithm like below.

1. Get the average of x coordinate for two markers. If bigger than our threshold, move right, smaller than our –threshold, move left.

2. If condition 1 is satisfied, we will compare the y coordinate for two markers. If the size of the y coordinate difference is bigger than our threshold, rotate according to it.

3. If condition 1 and 2 is satisfied, we will get the average of two depths. If the average is bigger than the distance, move forward.

We checked it works like what we desired, but not accurately. So we should try changing our thresholds or algorithm.

Second is testing DQN. We could get the gray image input. We had to calibrate for the coordinate of the ball because we used two cameras. We did some experiments and we could get the actions. However, it was hard to collect the balls so we should handle this problem.

3) 이승언

(1) RGBD 카메라를 위한 코드 작성

창시구 1 에서는 RGBD 카메라가 아닌 일반적인 웹캠을 사용하였다. 하지만 이번에는 비교적 공의 위치가 정확해야 했기 때문에 RGBD 카메라를 사용하기로 하였다. 하지만 RGBD 카메라의 depth 값을 subscribe 받고 이것을 다시 publish 하는 방법을 잘 몰랐기 때문에 윤국진 교수님의 랩에 방문하여 배워왔다. 또한 realsense 사의 RGBD 카메라를 사용하기 위해서는 여러 패키지를 깔아야 했고 에러가 많이 나와 어려움이 많았다. 하지만 결국 depth 값을 받아오는데 성공하였고 이를 바탕으로 로봇 바닥을 기준으로 한 공의 좌표를 받아올 수 있었다.

(2) CNN 과 바구니 앞 정열

개와 고양이 사진에 있는 marker 를 기준으로 하여 공을 뱉는 바구니에 로봇을 정렬시키는 것을 하였다. 위쪽 두개의 marker 까지의 거리를 받아와 그것을 바탕으로 로봇의 움직임을 제어하여 정렬할 수 있었다.

(3) DQN 로봇에 적용하기

우리가 simulator 를 바탕으로 training 한 pt 파일로 우리의 로봇에 적용하여 실험을 해 보았다. 먼저 data_intergration 노드에 rl 파트를 추가하였고 실험을 해 보았지만 원하는 action 대로 로봇이 잘 가지 않기도 했고 공의 좌표도 많이 보정을 해 주어야 했다.

Week13:

1) 박채리

Dqn 을 돌렸을 때 기존의 시뮬레이션으로는 트레이닝을 해도 시뮬레이션 상에서도 공을 잘 먹지 못해 많은 부분을 수정하였다. 첫번째로는, 맵을 그리는 것은 기존에 랜덤하게 벽을 만들거나 아니면 벽이 아예 존재하지 않았는데, 우리는 실제에 맞춰 3*2.5m 사이즈의 벽을 항상 일정하게 주었고, slam 을 통과하고 난 후의 위치를 넣어주기 위해 랜덤하게 오른쪽 또는 왼쪽으로 통과하도록 하였다. 그리고 공 사이의 간격을 50cm, 공과 벽 사이의 거리도 50cm 이상이 되도록 input 해주었다.

그리고 공의 개수에 대해 다양하게 트레이닝을 했었는데, 6 개 또는 그와 비슷한 경우에 트레이닝을 해도 공을 먹는 행위를 잘 하지 못하였다. 그래서 공을 15 개 이상 input 하였을 때 공을 상대적으로 훨씬 더 잘 먹어서 공의 개수는 최대한 많이 넣어주었다. 그리고 차의 action 에 대해 많은 부분을 수정하였는데, 기존에는 10 가지의 action 이 있었지만, 우리 시스템이 잘 움직이지 못하는 대각선 이동을 뺐고 또한 롤러를 이용해 공을 먹는 과정은 직진이 이루어져야만 가능하기 때문에 좌우 이동 또한 빼서 좌우 회전 및 앞뒤 운동 4 가지의 action 만을 가지고 트레이닝을 했다. 그리고 현재 코드 구조는 일정 시간이 지나면 수렴하게 되어있기 때문에 epsilon 값 등을 줄여 더 random 한 action 이 들어가고 수렴하는데 걸리는 시간을 좀 더 길게 하였다. 그렇게 트레이닝을 하다가 한가지 사실을 발견했는데, 전에는 트레이닝을 할 때 일정 값이 계속 지속되면 트레이닝이 끝난 줄 알았지만 시간이 지나고 다시 더 높게 올라가는 것을 관찰할 수 있었다. 그래서 우리는 최소 이틀 이상 동안 하나의 트레이닝만을 하였다. 그래서 가장 잘 된 pt 파일은 두가지가 있는데, 첫번째로 좌우회전 앞뒤 운동의 4 가지 action 의 pt 파일과 거기서 뒤로 가는 action 을 뺀 pt 파일을 만들 수 있었다.

Dqn 만을 실행했을 때 공을 잘 먹는 것 같아 dqn 과 slam 을 합쳐서 실험해보았더니 처음에는 코드에서 준 신호가 빠르게 모터에 전달되다가 dqn 중간 부분부터 모터에 굉장한 delay 가 생기는 문제가 있었다. 그래서 원인을 찾아보다가 마이리오에 write 을 해줄 때 다른 코드들을 참고하였을 때 for loop 안에서 계속 write 하는게 아니라 for loop 을 쓰지 않고 write 함수만 쓰는 차이점을 발견해서 코드를 수정하기로 하였다.

2) 오준영

This week, we really worked hard on Capstone design to make our device work well. Unfortunately, we still have lots of problems.

The biggest problem is delay. At first, the robot moves as our integration_node gives. However, after some time, it starts to act after some time. Due to this, our device works not properly. We wanted to handle this problem but still couldn't solve this.

Second problem is coming back to 0 point. We found that SLAM publish the topic named pose. We thought that this pose data would be accurate. However, this data was not accurate, so we think that we just have to use SLAM and send goal to make our device come back. The problem is that we modified SLAM code to move without rotation, so we think that it would be hard to come back if it is tilted a lot.

Third problem is NUC. We have done our task with using notebook. After some trials, we decided to do our jobs with NUC, but running python file didn't work.

Fourth problem is counting balls. We wanted to count the ball by adding 1 if it moves forward for pickup action. However, it doesn't count properly and sometimes it adds a lot instead of just adding once.

Due to these problems, we still failed to success. We really worked hard to solve these, 7days a week, about 60hours every week, but still got lots of things to do.

3) 이승언

(1) 모든 코드들 통합

앞서 개별적으로 작성하였던 CNN, SLAM, DQN 등 모든 코드들을 스위치를 이용하여 통합하였으며 그 외에 DQN 이 끝나고 원점으로 돌아오는 코드를 작성하였다. 우리는 move_base 로 돌아오지 않고, position 토픽을 이용하여 open-loop 으로 원점으로 보냈다. 이렇게 하니까 오류도 거의 발생하지 않았으며 다른 팀들 보다 빠른 속도로 올 수 있었다.

(2) 공을 먹는 코드 작성

DQN 에서 모든 상황을 model 에서 나온 action 을 주게 되니 공을 먹을 때의 정확도가 굉장히 떨어지는 것을 볼 수 있었다. 따라서 우리는 공이 롤러에서 20cm 정도 떨어져 있을 때에는 잠시 DQN 에서 나와서 align 을 하고 공을 먹는 과정을 거쳐 정확도를 올리도록 하였다.

(3) DQN 정확도 올리기

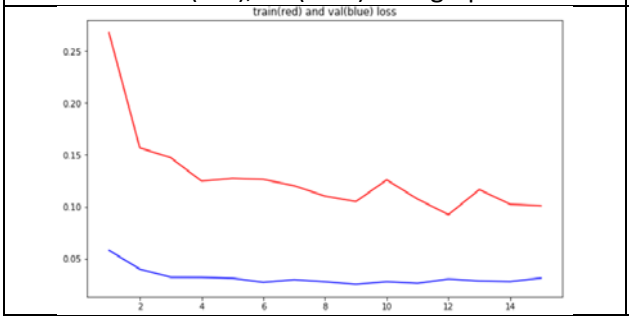
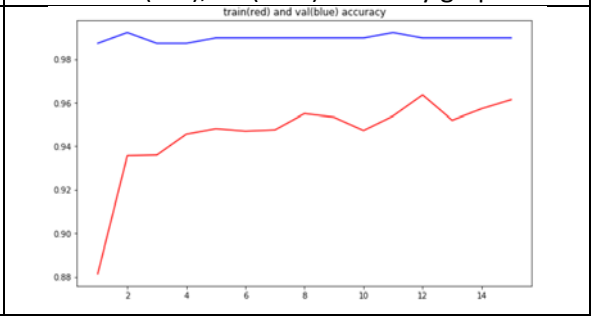
DQN 으로 공을 먹는 과정에서 정확도가 매우 낮았는데 그원인으로 우리가 판단 한 것을 딜레이였다. Action 은 제때 제때 나왔지만 로봇이 그 액션대로 빠르게 움직이지 않았기 때문에 많은 문제가 발생했다. 따라서 그 원인은 여러가지 노력으로 찾아보았는데 그 원인은 바로 labview 코드가 너무 무거워서 였다. 따라서 labview 코드에서 불필요한 부분을 모두 제거하여 delay 를 없앨 수 있었다.

4. ROS code explanation

4.1 Code Review

4.1.1 CNN : transfer_learning_tutorial.py

CNN model 을 학습시키는 코드로 개와 고양이 사진 20000 을 이용해 training 하였다. 이 때 사용되는 모델은 우리가 직접 만들지 않고 resnet18 이란 모델을 사용하였으며 pretrained=True 로 함으로써 이미 어느정도 학습된 모델을 사용하였다. 따라서 아무 수정도 하지 않은 채 training 을 시켜도 accuracy 가 0.98 이 나왔다. 우리는 이 accuracy 를 더 높이기 위해 여러가지 시도를 해 보았다. 우리는 batch size, learning rate, epoch size 를 중점적으로 바꾸어 보았는데 학습 속도는 조금 오르기도 했지만 accuracy 가 오르진 않았다. 여러 번의 실험 끝에 batch size 는 22, learning rate 은 0.0005, epoch size 는 15 로 함으로써 가장 빠르고 정확하게 train 시킬수 있었고 이렇게 최적화된 hyper parameter 값들을 이용해 train 한 결과는 다음과 같다.

Train(red), Val(blue) Loss graph	Train(red), Val(blue) Accuracy graph
	

이전의 결과들 보다 눈에 띄게 나아지진 않았지만 좀 더 적은 시간의 트레이닝으로 이전과 비슷한 validation accuracy 를 얻을 수 있었다.

4.1.2 DQN : simulator.py

98~136: 벽 생성

원래 코드의 벽은 랜덤한 형태로 생성되었다. 실제 우리가 실험해야 할 상황과는 많은 차이가 있었으므로 우리는 이 벽을 우리 상황에 최대한 맞추어 설정해 주었다. 먼저 세로 2.5m, 가로 3m 의 벽을 만들어 주었으며 slam 을 통과한 후 로봇의 위치는 오른쪽 혹은 왼쪽에 있을 수 있기 때문에 0.5 의 확률로 오른쪽 혹은 왼쪽으로 랜덤하게 생성되도록 벽을 만들어 주었다. 또한 우리의 로봇은 slam 에서 앞, 뒤, 좌, 우 이동만 하기 때문에 시작 할 때 map 을 회전 없이 만들어 주었다.

137~168: 공 생성

기존의 코드는 공의 개수가 많았으며 서로 붙어있는 경우도 많았다. 서로 붙어있다 보니 로봇이 붙어있는 공을 먹었을 경우 reward 가 한번에 많이 올라가기 때문에 로봇이 가까이 있는 공을 먹기 보다 붙어있는 공을 먹으러 가려는 경향이 보였다. 따라서 우리는 이러한 점을 보완하기 위해 공끼리의 거리를 45cm 떨어뜨려 놓았으며 공의 개수도 20 개로 한정하였다. 공의 개수를 6 개로 하지 않은 것은 공의 개수를 너무 적게 하니 training 하는 속도가 너무 느렸기 때문이다.

260~314: get_reward 함수

공을 어디로 먹느냐에 따라 받게 되는 reward 를 다르게 설정해 주었고 self.nosee 라는 변수를 만들어 시야에 공이 들어오지 않았을 때를 판별 할 수 있게 해 주었다. 또한 공이 10 번 이상 시야에 보이지 않을 때 done=true 가 되는 것을 없애고 대신에 ball_temp 에 공이 없거나 iteration 이 max_iter 를 넘을 경우에만 done=true 가 되도록 바꾸었다.

316~426: step 함수

먼저 우리는 action 의 개수를 앞, 뒤, 왼쪽 회전, 오른쪽 회전 4 개로 한정하였다. Action 의 개수가 많아질수록 각 상황에 따른 action 의 경우의 수가 많아져서 실제 로봇에서의 실험에서 불안정한 모습이 많이 보였기 때문이다. 또한 공이 시야에 들어오지 않았을 때(앞에서 정의한 nosee=0 일 때) 왼쪽으로 회전하는 action 이 들어가게 해 주었다. 또한 기존에는 시간이 지남에 따른 penalty 를 주는 것이 simulator 상에만 있고 학습에는 들어가지 않았기 때문에 이를 추가해 주었다. 마지막으로 로봇이 벽에 닿아 움직이지 못할 때 reward 를 -10 해줌으로써 벽에 부딪히는 상황을 최대한 피할 수 있었다.

4.1.3 DQN : testsimulator.py

앞서 설명하였듯 학습이 빠르게 되게 하기 위해서는 공의 개수를 6 개로 줄일 수 없었다. 하지만 데모에서는 공 6 개만 가지고 하기 때문에 공이 6 개만 있을 때 학습된 모델이 적절히 행동하는 지 확인할 방법이 필요했다. 따라서 공이 6 개만 있는 simulator 파일을 새로 만들어서 학습이 완료된 모델을 test 해 보고 6 개의 공을 잘 먹는지 확인하였다.

4.1.4 DQN : main.py

main.py 에서는 dqn_learn.py 의 parameter 들을 지정해주었다. 우리는 learning_rate, alpha, eps 를 중점적으로 바꿔가며 training 시켜보았으며 이 parameter 들이 수렴하는 위치에 영향을 미쳤다. 최대한 reward 가 오른 시점에서 수렴할 수 있도록 parameter 값들을 수정하였다.

4.1.5 convert_sensor_to_image.cpp

카메라로 들어온 이미지를 공과 벽의 좌표로 변환해주는 node 이다. 우리는 카메라 두대를 사용하였기 때문에 30cm 이상 떨어진 경우에는 위의 카메라로 측정한 값으로 변환하였으며 30cm 보다 가까이 있을 경우에는 아래의 카메라로 측정한 값으로 변환하였다.

4.1.6 readmarker.cpp

rgbd 카메라를 이용해 marker 를 인식하므로 이를 이용해 marker 번호 0,1 또는 4, 5 의 x, z(카메라에서의 거리), 차체에서의 거리 값을 calibration 한다. 그리고 이 값들을 core msg 의 mark 라는 메시지에 저장하여 mark 라는 토픽으로 메시지를 publish 하도록 하였다.

4.1.7 Ball_detect cpp

13 14 :Calibration 을 통해 얻은 값을 넣어주었다.
25 : car_height 변수에 카메라의 높이를 넣어준다.
40 ~ 51 : 파란색과 빨간색을 구분하는데 쓰이는 이미지 필터에 들어갈 h s v 값을 설정한다.
53 ~ 55 : 이미지 필터 및 calibration 함수이다.
56 ~ 57 : 카메라의 좌우 위치와 앞뒤 거리를 보정 해주기 위한 offset 으로 위 카메라와 아래 카메라에서 인식되는 값을 같게 보정해주는 역할과 함께 차 앞에 놓여있는 공의 위치를 정확하게 인식하도록 보정해주는 역할도 한다.
59 ~64 : 이미지 필터에 사용되는 변수들이다.
73 ~ : ball_detect 함수 시작
79 ~ 90 : buffer 리사이즈
Buffer1 과 buffer2 의 가로 너비가 320 일 경우 640*480 으로 사이즈를 바꾸고 그 외에는 각각 frame, frame2 에 buffer 들을 저장한다.
91, 92 : frame 을 넣어 calbration 을 통해 얻은 값들을 이용해 frame 을 캘리브레이션한다.
94 ~ 117 : 빨간 공과 파란 공을 찾기 위해 이미지를 필터해주는 과정이다.
118 : core msg 에 있는 ball_position messeage 를 msg 라는 이름으로 불러온다.
143 ~ 180 : 파란 공 인식
Min_enclosing circle 함수를 이용해 찾은 원들의 반지름이 처음에 설정한 min tracking ball size 변수의 크기보다 클 경우에만 공 개수를 세도록 하였다. 그 이유는 노이즈를 제거하기 위함이다. 또한 원의 중앙 좌표가 15 보다 크고 605 보다 작을 경우에만 개수를 세도록 하였는데, 그 이유는 공이 화면의 끝에 걸쳐져 있을 경우에 원을 찾는 것이 매우 불안정해 걸쳐져 있을 경우의 좌표값을 찾아 제외시켜주었다. 그리고 rgbd 카메라를 사용하기 때문에 depth 이미지에서의 원 중심 좌표에 있는 값을 받아 distance 에 저장하였다. 이렇게 받은 distance 값과 원의 중심을 받아 pixel2point 함수에 넣어 point ball2 에 리턴된 값을 저장한다. Ball2 에서 받은 x 값에 x_offset 을 뺀 후 cx 에 저장한다. 그리고 y 값을 받아 cy 에

저장하고 차체에서의 거리를 받아 distance 에 저장하고 난 후 파란 공 카운터에 하나 더한다. 그리고 공의 색을 구분해주는 color_ball 을 0 으로 하여 이러한 값들을 msg 에 넣어준다. 그리고 파란 공의 전체 개수를 알맞게 세기 위해 min enclosing 함수로 인해 발생한 공 개수 카운트 문제를 해결하였다. 그 방법은 i 번째 공의 좌표와 그 다음 i+1 번째 공의 좌표를 각각 x1, y1, x2, y2 에 저장하고 거리 차이를 계산하여 diff 라는 변수에 저장하고 현재 반지름을 l 에 저장한다. 그 후 마지막 177 번째 줄에서 diff 값과 l 값을 비교해 거리 차가 반지름보다 작을 경우 i 에 하나를 더해 인덱스를 올려줌으로써 겹치게 인식한 공의 개수를 정확하게 셀 수 있도록 하였다. 그리고 172~176 까지는 카메라 화면에 인식한 공의 원과 현재 위치를 나타내도록 하였다.

181 ~ 221 : 빨간 공 인식

파란 공의 개수를 세는 방법과 동일하며 단지 msg 에 넣어줄 때 공의 색깔을 나타내는 변수 colot_ball 에 1 을 저장하는 것에서만 차이난다.

222 : count_r 과 count_b 에 저장된 빨간 공과 파란 공 개수를 msg 의 size 에 넣어준다.

223 ~ 224 :

전체 공의 개수만큼 x, z, color 값을 프린트 해주도록 하였다.

228 ~ 230 : 이미지들을 화면에 띄우도록 하였다.

233 : core message 에 ball_position 메시지를 퍼블리시해준다.

235 : msg 의 color 벡터를 초기화해준다.

238 ~ 258 : 이미지 콜백 함수

Rgbd 카메라를 실행하는 노드에서 퍼블리시 해준 이미지 값을 받아 그 높이와 너비가 각각 480 과 320 이 일 경우 640*480 으로 리사이즈하도록 하였다. 그리고 그 외에는 아무것도 하지 않도록 했다. 또한 cv_bridge 를 이용해 msg 안의 rgb 형식의 image 를 buffer 에 저장하는 것을 시도하고 만약 cv bridge 에서 exception 이 발생한 경우 ros error 메시지를 프린트해주도록 하였다. 그리고 나서 ball_detect 함수를 호출한다.

259 ~ 284 : depth 이미지 콜백 함수

이미지 포인터를 만들고 depth 형식인 16UC1 에 맞춰 buffer2 에 msg 안의 image 를 저장한다. 만약 buffer2 가 비었으면 실패 메시지를 프린트하고 리턴하며, 이 때 cv bridge 에서 exception 이 일어나면 ros error 메시지를 프린트하고 리턴한다.

286 ~323 : main 함수

Realsense 카메라를 실행시키는 노드로부터 퍼블리시된 depth 와 color 이미지를 subscribe 한다. 그리고 core message 의 ball_position 메시지를 /position 이라는 이름의 토픽으로 publish 한다. 이를 로스 스핀을 통해 반복한다.

325 ~ 343 : pixel2point 함수

원의 중심 좌표와 실제 거리값을 받아 calibration 을 통해 얻은 값으로 Xc,Yc 값을 받고 distacnce 는 기존 카메라와 공 사이의 거리에서 차체에서의 거리로 바꾸어 리턴한다.

345 ~ 354 : morphOps 함수

이미지의 노이즈를 없애주는 필터 처리를 해준다.
356 ~ 364 : calibrate 함수
Calibration 을 통해 얻은 intrinsinc, distortion 값을 이용해 받은 이미지를 캘리브레이션하여 바꿔준다.
367 ~ 384 : 이미지 필터 함수
이미지 필터를 해줄 때 필요한 여러 필터들을 한꺼번에 해주는 함수이다. 예를 들면 medianblur, inrange, Gaussian blur, canny 등등의 필터가 사용된다. 이 때 사용되는 hsv 값들은 처음에 global 변수로 선언해준 값들을 불러온다.

4.1.8 Ball_bottom cpp

위의 ball_detect cpp 와 모두 같으나 공과의 거리를 계산하는 방법과 pixel2point 함수가 다르다.
위에서는 rgbd 카메라를 이용했기 때문에 depth 이미지에서 맞는 좌표의 값을 불러왔지만 ball_bottom 에 쓰이는 카메라는 웹캠이기 때문에 거리 계산은 실제 공 반지름인 3.5 를 이용해 거리를 계산하였다. 이는 pixel2point 함수 내부에서 공과의 거리를 계산하였으며 따라서 pixel2point 함수에 넣어주는 값 또한 공 중심 좌표와 거리가 아닌 중심 좌표와 인식된 공 반지름이다. 여기서 기존에는 공의 실제 반지름을 이용해 거리 측정을 했었지만 이에 오차가 매우 크기 때문에 실험적으로 공의 중심 좌표와 실제 차와의 거리의 관계를 그래프로 나타내어 관계식을 찾아 함수 안에서 중심 좌표를 넣었을 때 거리 계산을 할 수 있도록 하였다.

4.1.9 Data_integration cpp

89 ~ 98 : take_sleep 함수
Data[0] ~ data[3]까지 0 으로 초기화 해준 뒤 for loop 을 이용해 인풋으로 받은 t 값 동안 바퀴 속도를 0 을 myRio 에 준다.
99 ~ 108 : turn_left 함수
속도에 해당하는 size 값과 myRio 에 보내는 시간 t 를 인풋으로 받아 size 만큼의 속도의 왼쪽 회전을 data 에 넣어 받은 시간 동안 myRio 에 보낸다.
109 ~118 : turn_right 함수
위와 동일하며 오른쪽으로 회전하는 함수이다.
119 ~ 128 : go_front 함수
마찬가지로 직진 운동을 myRio 에 보내는 함수이다.
130 ~ 139 : go_back 함수
속도와 시간을 받아 받은 속도만큼의 후진을 해당 시간 동안 myRio 에 보내는 함수이다.
140 ~ 149 : go_left 함수
좌 이동을 myRio 에 보내는 함수이다.

150 ~ 159 : go_right 함수
우 이동을 myRio 에 보내는 함수이다.
160 ~ 169 : go_del 함수
SLAM 에서 받은 cmd_vel 에 맞도록 방향과 크기를 조절하고 평행 운동과 회전의 비율을 고려하여 각각의 바퀴에 대한 값을 계산해 받은 시간 만큼 동안 myRio 에 보낸다.
170 ~ 183 : closest_ball 함수
Ball distance array 를 받아 array 에 아무것도 없으면 -1 을 리턴하고 그 외에는 가장 거리가 작은 값의 index 를 리턴하는 함수이다.
184 ~ 210 : dataInit 함수
모든 data array 를 초기화해주는 함수이다.
211 ~ 213 : rgbd 콜백 함수
rgbd 에서 보낸 ball_position 메시지로부터 size 만 받아 count_num 변수에 저장한다.
214 ~ 225 : 웹캠 콜백 함수
Ball_bottom 메시지에서 size 를 받아 count2 변수에 저장하고 이를 count_num 에 더해주어 cont_num 이 시야에 보이는 공의 전체 개수에 해당하도록 설정한다. 그리고 closest_ball 함수에 z 값의 array 를 넣어 가장 가까이 있는 공의 index 를 받아 closest_idx 에 저장한다. 만약 아래 카메라에서 보이는 공의 개수가 0 이 아닐 때 closest_idx 에 해당하는 x, z, color 값을 받아 각각 x, z, c 에 저장한다.
226~228 : rl action 콜백 함수
DQN 을 통해 받은 data 를 action 변수에 저장한다.
229 ~ 235 : pose update 함수
Slam 에서 보내주는 pose 값을 받아 현재의 위치와 방향을 각각 posex, posey, posez 변수에 저장한다.
236 ~ 240 : msg(cmd_vel) 콜백 함수
Cmd_vel 의 linear x 값과 y 값을 각각 cmdvelx, cmdvely 변수에 저장한다.
251 ~ 263 : catdog cnn 콜백 함수
받은 msg 안의 data 으로부터 cat dog string 을 받아 만약 cat 일 경우 catordog 변수에 1 을 , dog 일 경우 2 를 저장하고 프린트한다.
271 ~ 278 :Subscriber
data integration 노드에서는 총 8 개의 subscriber 를 사용한다. 위와 아래 카메라에서 보이는 공의 개수를 /position, /position2 로부터 받고 DQN 으로부터 나온 action 을 action/Int8 으로 받는다. SLAM 에서 사용하는 /cmd_vel, /slam_out_pose 를 받아 차를 움직이고 현재 위치를 받는다. 그리고 marker 의 위치를 /mark 에서, /catdog/String 에서 개와 고양이를 분류한 값을 받는다.
280 ~ 288 : c_socket 리셋, myRio 통신 연결

Socket 함수를 이용해 C_socket 을 만들고 연결을 시도하고 만약 연결에 실패하면 메시지를 프린트하고 c_socket 을 닫고 -1 을 리턴한다.

290 ~ 308 : switch 0 SLAM

Switch 가 0 일 때 start 메시지를 프린트하고, /cmd_vel 에서 받은 값을 각각의 바퀴에 값을 계산해 넣어준다. 이때의 값과 부호는 다이나믹셀 키네마틱스에 따라 계산하였고 k2 는 slam 의 속도를 조절하기 위해 변수를 설정하였다. 이렇게 한 번씩 넣어줄 때마다 5 번씩 myRio 에 보내주도록 하였다. 그리고 /slam_out_pose 로부터 받은 현재 위치 값을 이용해 앞으로 1.9m 이동하였으면 switch 를 1 로 바꾸고 after_slam_goforward 스위치를 1 로 바꿔 킨다.

309 ~ 30 : switch 1 DQN

처음에 after_slam_goforward 가 1 일 때 go_front 함수를 통해 일정 거리를 앞으로 직진하고 after_slam_goforward 의 값을 2 로 바꿔 스위치를 끈다. 그 후 dqn 으로 공을 먹기 위해 롤러를 돌리도록 data 값을 넣어 준다.

만약 count2(아래 카메라에서 보이는 공의 개수)가 1 개 이상이고 공과의 거리가 30cm (실제 차와의 거리는 20cm 이다.) 안으로 들어오면 실험을 통해 구한 값을 이용해 그 거리를 이동하기 위해 myRio 에 보내줘야 하는 시간을 계산해 t3 에 저장한다. 그리고 거리가 30cm 이하이고 x 값이 -5 ~ 5cm 범위에 있으면 먹는 모드가 켜는데, 이 때 c 로부터 빨간 공과 파란 공을 인식한 값을 받아 소팅바의 각도를 바꿔준다. 그리고 전에 계산한 t3 를 이용해 go_front 함수를 이용해 직진하며 공을 먹고 ball count 에 1 을 더해주고 롤러가 공을 충분히 먹을 때까지 기다리기 위해 take_sleep 으로 잠깐 멈춘다. 만약 공이 30cm 안에는 있지만 x 값이 범위 안에 있지 않은 경우 정렬을 해주는데, x 값이 5cm 보다 크면 오른쪽 회전, -5cm 보다 작으면 왼쪽 회전을 한다.

그리고 만약 count_num (시야에 보이는 전체 공의 개수)이 0 일 때 공이 보일 때까지 왼쪽으로 회전하게 한다.

만약 위의 두가지 경우에 포함되지 않을 경우 DQN 으로부터 받은 action 에 따라 go_del 함수에 값을 넣어 차를 움직인다. Action 이 0 인 경우에는 직진, 3 은 후진, 1 은 왼쪽 회전, 2 는 오른쪽 회전을 하도록 하였다. 그 외의 경우에는 움직이지 않도록 하였다.

그리고 먹은 공의 개수가 6 개 이상이 되면 switch 를 2 로 바꾼다.

376 ~ : switch 2 원점으로 돌아감

Switch 2 안에서 차의 정렬을 위해 ssw 라는 switch 를 만들어주었다. ssw 가 0 일 때 차의 방향 조절을 하는데, 차가 원점을 수직하게 바라보고 있을 때의 pose z 값이 -1 또는 1 이기 때문에 현재 차의 방향에 따라 더 가까운 회전 방향을 계산해 왼쪽 또는 오른쪽 회전을 해서 정렬이 되도록 한다. 그리고 1 에 근접해졌을 때 ssw 를 1 로 바꾸고 take sleep 을 이용해 잠깐 멈추도록 해서 현재 차의 위치 값을 안정해지도록 하였다. 그리고 차의 방향 정렬을 끝낸 다음에 미로를 통과하기 위해 새로운 좌우 정렬 switch 를 tran1 로 설정했다. ssw 가 1 이고 trans1 이 0 일 때 좌우 1.25m 를 기준으로 처음에 거리 차를 계산해 한번에 빠른

속도로 해당 좌표에 도달할 수 있도록 값을 주었다. 그리고 trans1 을 1 로 바꾸고 그 다음에는 y 값이 1.13 ~ 1.25m 안에 들어오도록 느린 속도로 좌우 이동을 주었다. 그리고 y 값 정렬이 끝나면 data 를 전부 초기화 한 후 ssw switch 를 2 로 바꾸어 좌우 정렬을 끝낸다. 그 다음은 ssw 가 2 일 경우 좌우 정렬을 하면서 서스펜션의 불안정으로 원점을 수직하게 바라보는 방향에서 틀어지기 때문에 다시 z 값을 -1, 1 을 갖도록 회전운동을 주어 방향 정렬을 한다. 그 후 ssw 를 3 으로 바꾸고 take_sleep 을 통해 움직임을 멈추고 현재의 위치 값이 안정해질 수 있도록 하였다. 그 다음 ssw 가 3 일 경우 차의 위치가 원점으로부터 1.5m 보다 멀리 있을 때 앞으로 가는 움직임을 넣었고 만약 범위 안에 들어오면 마찬가지로 take_sleep 을 하도록 하였고 ssw 는 4 로 바꾼다. 다음 단계에서는 원점을 바라보도록 하는 좌우 정렬을 해야하기 때문에 trans2 라는 switch 를 설정하였다. 만약 trans2 가 0 일 경우 현재의 차의 y 값과 0 의 차를 이용해 실험을 통해 구한 비율로 myRio 에 보내줘야하는 값을 계산해주어 한번에 0 까지 갈 수 있도록 하였고 그 다음엔 바로 trans2 를 1 로 바꾸어 switch 를 끝냈다. Trans2 가 1 일 경우 y 값이 0.2 ~ -0.2m 안에 들어오도록 느린 속도로 좌우 정렬을 하도록 했다. 그리고 이 범위 안에 들어오면 data 를 초기화하고 ssw 를 5 로 바꾼다. ssw5 에서는 원점(<0.1m)에 차가 위치할 수 있도록 직진시키고 take_sleep 을 한 후 ssw 를 6 으로 바꾼다. 6 에서는 개와 고양이 사진이 있는 벽을 보도록 90 도 왼쪽 회전을 시키고 ssw7 로 바꾼 후 7 에서는 사진이 시야에 들어올 수 있도록 앞으로 일정거리 직진하도록 하였고 다음 단계인 CNN 을 위해 sw switch 를 3 으로 바꾼다.

491 ~ : switch 3 CNN

Sw 3 은 크게 바구니에 다가가는 정렬과 공을 릴리즈하는 두가지의 과정으로 나눌 수 있다. 처음에 data 를 초기화시켜 롤러와 소팅바 둘 다 초기화하고 slep 이란 변수를 통해 4 초 동안 차를 멈추고 slep 에 1 로 바꾼다. nvidia_Jetson 을 사용하지 않았기 때문에 NUC 만으로 이미지를 구별하는데 시간이 소요되기 때문에 4 초동안 움직임을 멈춰서 안정적으로 이미지를 인식하여 구분할 수 있도록 하였다. 그 후 readmarker 노드에서 subscribe 한 marker 와의 거리, x, y 값을 이용해 차를 정렬하도록 하였다. Marker 가 인식이 되지 않을 경우 x_1 과 y_1 에 default 로 0 이 저장되기 때문에 498 줄과 같이 둘 다 모두 0 이 아닐 경우에만 marker 의 좌표를 이용해 정렬을 하도록 하였다. 정렬의 순서는 회전, 좌우, 직진으로 해주었고 처음에 회전의 경우에는 y 값의 차이를 이용해 c2 로 지정한 일정 범위 안에 들어오도록 차를 회전시켰다. 그리고 y 값이 범위 안에 들어오면 x 값을 이용해 c1 을 통해 지정한 일정 x 범위 안에 들어오도록 (마커의 중앙에 차가 오도록) 좌우 움직임을 해주었다. 그리고 이 두 범위 안에 모두 들어오면 다시 y 값을 이용해 거리가 c3 안에 들어오도록 계속 직진 운동을 넣어주었다. 이 과정은 바구니에 롤러가 들어오도록 거리를 가깝게 하는 정렬의 마지막 과정이다. 이 세가지의 경우가 계속 반복되며 정렬 후 바구니에 알맞게 가도록 하였다.

그 후 CNN 을 통해 구분한 cat dog 를 이용해 소팅바의 각도를 바꾸고 data[4]=2 로 하여 공을 릴리즈 할 수 있도록 롤러를 반대방향으로 돌리도록 하였다. Nvidia Jetson 을

사용하지 않아서 이미지를 인식하는데 시간이 소요되므로 시간을 단축하기 위해 처음에 인식한 이미지를 switch 를 통해 기억하여 반대편의 벽에 릴리즈를 할 때 그와 반대되는 방향의 소팅바로 바꿀 수 있도록 sssw 라는 switch 를 설정해주었다. 만약 처음 릴리즈할 경우 sssw 는 0 이며 이 때 catordog 를 통해 cat 인 경우 1 이기 때문에 소팅바의 각도를 그에 맞게 바꾼다. 그리고 나서 secondrelease 와 ssww 에 0 을 저장한다. 만약 dog 로 인식하여 catofdog 값이 2 인 경우 소팅바의 각도를 바꾸고 secondrelease 에 0 을, ssww 에 1 을 저장한다. 두가지 과정 중 하나라도 끝나면 sssw 값을 1 로 바꾼다. Sssw 값이 1 인 경우 두번째 릴리즈를 시작하며 만약 앞에서 cat dog 를 인식하고 저장한 switch 인 ssww 값이 0 인 경우 파란 공을 릴리즈 할 수 있도록 소팅바의 각도를 바꾸고 반대로 ssww 값이 0 이 아닌 경우 빨간 공을 릴리즈 할 수 있도록 그에 맞게 소팅바 각도를 바꾼다. 만약 secondrelease 가 1 인 경우 sw 를 4 로 바꾸어 끝나도록 하였다. 그리고 myRio 에 보내주는 for loop 를 넣어 앞에서 설정한 소팅바와 함께 릴리즈를 5 초 동안 하도록 해서 시간을 충분히 주었다.

그리고 secondrelease 가 0 인 경우 실험적으로 구한 값들을 이용해 open loop 로 일정 거리 후진하고 180 도를 돌아 직진하도록 하였다. 이 과정을 매우 빠른 속도를 주어서 시간을 단축할 수 있었고 반대편 벽에 있는 사진이 시야에 들어올 수 있도록 하였다.

4.2 Philosophy and strength of our code

4.2.1 SLAM with no rotation

처음에 미로를 통과하기 위해서 SLAM 에서 yaml 파일의 parameter 의 값들을 바꿔 시뮬레이션을 돌려 미로를 빠져나가는데 적합한 조합을 찾고 실험을 반복함으로써 가장 빠르고 안전하게 미로를 탈출할 수 있는 값들을 구하려고 하였다. 문제는 같은 parameter 값들을 사용하였는데도 불구하고 로봇의 움직임이 약간씩 다르거나 우리가 쉽게 예측할 수 없는 행동을 하는 경우가 존재하였다. 우리는 이 불확실성의 근원을 회전이라고 생각하였고 yaml 파일에서 rot_vel 값의 최대 및 최소 값을 둘다 0 으로 설정함으로써 SLAM 에서 회전운동을 배제하게끔 하였다. 이러한 방법을 채택함으로써 미로를 통과하는데 다소 불필요하다고 생각했던 회전운동을 배제, 오직 직진 및 좌우 운동만으로 미로를 통과하게 하였고 벽에 부딪히거나 우리의 예측에 벗어나는 행동이 현저히 줄어들게 되었다.

```
DWAPlannerROS:
  # Robot configuration parameters
  acc_lim_x: 4.0
  acc_lim_y: 4.0
  acc_lim_th: 0

  max_vel_x: 4.5
  min_vel_x: 0.1
  max_vel_y: 4.5
  min_vel_y: -4.5

  max_trans_vel: 3
  min_trans_vel: 0.05
  max_rot_vel: 0
  min_rot_vel: 0
```

4.2.2 Rostopic 중 pose 사용

6 개의 공을 다 수집한 이후 다시 starting point 으로 되돌아 오는 방법으로 goal point 를 (0,0)으로 설정하여 나온 cmd_vel 을 바탕으로 움직이는 것을 처음에 고려하였다. 그러던 중 SLAM 에서 publish 하는 topic 중 pose 라는 것의 존재를 알게 되었는데 이는 시작한 지점을 원점으로 기준 삼아 현재의 로봇의 위치와 기울어진 정도를 나타내는 topic 이었다. 이 topic 을 이용함으로써 우리의 로봇이 map 에서 어느 위치에 존재하는지 실시간으로 알 수 있었고 이러한 정보를 통해 어느 방향으로 얼만큼 움직여야 미로를 통과하고 다시 원점으로 돌아올 수 있는지 계산할 수 있었다. 이러한 계산 값을 통해 우리가 원하는 경로대로 움직일 수 있게 함으로써 다시 출발점으로 돌아오는데 걸리는 시간을 현저히 줄일 수 있었다.

4.2.3 Open loop 과 real-time feedback 의 조합

우리는 로봇의 움직임에 있어 open-loop (비 피드백 컨트롤, 즉 한번에 일정 시간 동안 같은 입력
값을 주는 것) 과 real-time feedback (실시간 피드백, 즉 현재의 상태를 계속 업데이트 받아서 이에 따른 입력값을 주는 것)을 혼용하여 사용하였다.
만약 계속하여 open-loop 을 이용하여 우리의 로봇을 제어한다면 로봇에게 주는 명령이 현재 로봇의 상태에는 적합하지 않은 것일 확률이 높아진다.
만약 계속하여 real-time feedback 을 이용하여 우리의 로봇을 제어한다면 우리 로봇의 실시간 상태에 대하여 명령을 주기 때문에 적합한 행동을 취할 수는 있으나 이 행동이 실시간으로 계속 빠르게 바뀌게 되면 움직임이 비연속적이라 비효율적이고 slip 현상이 일어날 수 있다.
이러한 문제점들을 해결하기 위하여 우리는 다음과 같은 방법을 채택하였다

a. 현재 상태를 기준으로 어떤 action을 얼만큼의 시간 동안 줘야 하는지 계산하고 이만큼 실행 (open loop)

b. 로봇의 상태를 계속 업데이트 하며 이에 맞는 action을 실시간으로 실행 (real-time feedback)

c. 어느정도 오차 범위 내에 조건 만족 시 real-time feedback 종료

위와 같은 방법을 사용함으로써 처음에 계산한 값으로 한번에 행동을 함으로써 우리의 목표 지점에 빠르고 거의 정확하게 도달할 수 있고, 이 후 실시간 피드백을 통해 미세 정렬을 실시해줌으로써 작은 오차도 잡을 수 있었다.

4.3 Factors we can improve more

4.3.1 Full DQN

우리가 공을 수집하는 알고리즘은 공이 롤러로부터 20cm 밖에 존재하면 DQN 에서 결정한 action 을 바탕으로 움직이고 20cm 안에 존재하면 DQN 이 아닌 우리의 개입으로 조정 후 pick up 을 하게 하였다. 이러한 방식을 채택함으로써 더 효율적이고 정확한 공 수집을 수행할 수 있었다. 다만 아쉬웠던 점은 우리 로봇의 pickup 방식이 롤러였기 때문에 다른 픽업 방식에 비해 단순히 직진만으로 공을 먹을 수 있다는 점에서 따로 개입 없이 순수 DQN 만으로도 Task 가 가능하였는데 이를 성공하지 못했다는 점이다. 공을 먹는 과정에서는 문제가 없었으나 먹은 공의 개수를 세는데 계속 변수가 발생하였다. 순수 DQN 을 사용할 시 공의 개수를 세는 방법으로 공이 30cm 밖에 있다가 15cm 이내로 들어올 경우 +1 을 해주는 것을 고려하였는데 이러한 경우에 공의 좌표가 튀거나 다른 변수들로 인해 공의 개수를 한 두개 정도 잘못 세는 경우가 다소 발생하였다. 이를 해결하기 위해 조금만 시간이 더 존재하여 정확한 원인을 분석했거나 다른 방법으로 먹은 공의 개수를 구했다면 순수 DQN 으로 만으로도 정확하고 빠르게 과제를 수행할 수 있었을 것이라는 아쉬움이 존재한다.

4.3.2 Shorter path planning

우리의 최종 demo 에서 두번째 공을 집을 때 가까운 공 대신 멀리 있는 공을 향해 갔었는데 이는 우리 카메라의 시야 때문인 것으로 보인다. 카메라의 위치나 각도를 잘 조정하여 볼 수 잇는 시야를 넓히면 공이 조금 옆에 있으면 안보이는 현상을 줄일 수 있을 것이며 공을 먹는 경로 역시 좀더 최적의 경로에 가까워질 것이다.

4.4 Tips for future capstone design 2 students

4.4.1 디버깅에 충실할 것!

문제가 발생하였을 경우 이 문제가 어디에서 발생한 것인지 정확하게 인지하는 것이 매우 중요하다. 기본적인 debugging 능력은 창시구 1 에서 많이 길렀을 것이다. 이번에 우리 조에서 가장 시간을 오래 먹은 것이 delay 문제였는데, ROS 에서는 맞는 타이밍에 적합한 명령을 주고 있었으나 우리의 모터의 움직임에서는 delay 가 발생하여 우리가 원하는대로 움직이지 않은 현상이 존재하였다. ROS code 를 계속 바꾸고 형식을 뜯어 고쳐가며 원인을 파악하려고 했는데 결국 Labview 코드 문제였었다. 문제가 Labview 코드에서도 존재할 수 있으니 꼭 문제가 ROS 코드에서만 존재할 것이라고 단정짓지 말고 모든 가능한 경우의 수를 고려해보고 어디서 문제가 발생했을지 정확히 인지하는 것이 매우 중요하다.

4.4.2 Launch 파일과 단축기를 잘 사용할 것

창시구 1 과 달리 창시구 2 에서는 task 를 수행하기 위해서 실행시켜야 할 node 가 매우 많다. 우리 조의 경우에는 10 개 정도의 node 를 실행시켰다. 아무래도 node 의 개수도 많고 경우에 따라서는 python file 을 실행시켜야 해서 directory 를 바꿔서 실행시켜야 하는 경우도 많다. 창시구 2 역시 매우 많은 trial 을 반복해야 하기 때문에 매번 node 를 하나하나 키고 긴 명령어를 치는데 걸리는 시간이 쌓이면 매우 크다. 처음부터 launch 파일을 만들어 이러한 노고를 줄이거나 단축키를 정의해줌으로써 trial 을 실행하는데 걸리는 준비시간을 줄이면 생각보다 매우 많은 시간을 아낄 수 있을 것이다.

4.4.3 Rostopic list 를 꼭 확인해볼 것

창시구 2 에서, 특히 SLAM 을 사용하면서 여러가지 topic 을 publish 하는 것을 확인할 수 있을 것이다. 이러한 topic 들이 어떤 의미를 가지고 있는지 간략하게라도 알아보자. 우리의 경우에는 SLAM 에서 시작점에 대한 현재 로봇의 위치를 좌표값으로 알려주는 pose 라는 topic 을 알게 되었고 이를 이용함으로써 출발점으로 다시 돌아가는데 걸리는 시간을 다른 조에 비해 현저히 줄일 수 있었다. 이미 publish 되고 있는 topic 에 대해 잘 인지하고 이를 활용할 수 있으면 다른 조와 다른 방법으로 task 를 수행할 수 있을 것이며 선택할 수 있는 방법의 폭이 넓어질 것이다.

4.4.4 CNN 인식에 대하여

CNN 인식의 정확도는 높았지만 사진을 보고 그것이 개인지 고양이인지 인식하는 속도가 느렸다. 그것의 첫번째 원인은 catdog_cnn_network.py 의 callback 함수에서 waitkey 가 500 으로 설정되어 있는 것이었다. 이를 50 으로 줄이니 속도가 많이 향상되었지만 그래도 아주 빠르진 않았다. 이는 눅의 연산 속도 때문이었는데 눅 대신에 zetson 을 사용하니 이 문제가 해결되었다. 하지만 이 속도를 조금 더 빠르게 하기 위해 zetson 을 추가로 사용하기에는 로봇에 공간도 충분하지 않았으며 우리의 알고리즘 특성상 readmarker 를 통해 어느정도 align 하여 정지한 후에 개 고양이 사진을 인식하였기 때문에 그렇게 빠른 속도를 요하지도 않았다. 차라리 우리는 zetson 을 사용하지 않는 대신 개 고양이 사진을 판별하는 시간을 3 초동안 주기로 하였다. 하지만 알고리즘에 따라 회전하면서 개 고양이 사진을 판별해야 하는 경우에는 매우 빠른 속도를 요하므로 zetson 을 사용하는 게 나을 수도 있을 것이다.

4.4.5 SLAM 속도 문제

SLAM 은 여러가지 파라미터들을 조금씩 튜닝해 가며 최적의 값을 찾아가야 한다. 하지만 그전에 완료하여야 할 것이 cmd_vel 값을 모터에 줘야할 속도로 변환하는 것이다. move base 노드가 find frontier 로 부터 goal 값을 받으면 global path 와 local path 를 정한 후 그대로 움직일 수 있도록 cmd_vel 값을 publish 한다. 그러면 이 cmd_vel 값을 받아서 data_intergration 에서 바퀴는 모터에 적절한 속도를 주어야 한다. 만약 이 변환이 정확하지 않다면 이 로봇을 통해 튜닝하는게 아무 의미가 없어질 수 있다. cmd_vel 로 준 값보다 더 빠른 속도로 움직이게 되면 벽에 박을 가능성이 굉장히 올라간다. 따라서 원하는 cmd_vel 값으로 로봇이 정확히 움직이는 지 확인하여야 한다. 변환 식은 dinamixel kinematics 를 참조하면 되며 여러 번의 실험 결과 cmd_vel 보다 속도가 약간 느린 것을 큰 문제를 만들지는 않았다.