창의적 시스템 구현

Final Report

20150352 박진욱

Index

1. Progress report ( Solid works and ROS)
2. Presentation material
3. 후기

1) Progress report( Two main parts of our work are Solidworks and ROS)

- Solid work

3/19~3/25

월요일 교수님과의 미팅에서 공을 집는 방법에 대해 토론. 공을 집는 방법으로 로봇 팔을 이용, 롤러를 사용, suction을 이용한 방법 이 3가지가 나왔다. 최소한의 무게로 최대 효율을 내기 위해 suction을 이용하기로 결정하였다. suction으로 결정한 뒤 세부적인 디자인을 간단하게 정리하였다.

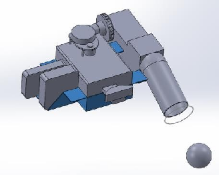
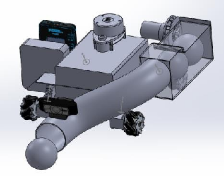
  

-회의 중에 나온 공을 집는 방법들

또한 진동을 절감하기 위해 suspension system을 사용하기로 결정하였다. 수요일에 진공펌프를 조사하고 진공청소기 안에 들어있는 펌프를 사용하기로 결정하였다. 그리고 suction관을 PVC관 같은 매끄러운 것으로 하라는 조언을 받았고 suction system을 어느정도 디자인 하게 되었다.

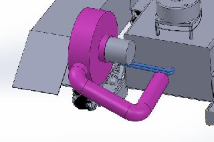
3/26~3/31

월요일에 교수님과의 미팅에서 suction system의 전체적인 디자인에 대해 주로 토의하였다. 공을 넣는 바구니의 높이가 2cm라서 suction을 밑에 달자는 의견도 나왔다 그러나 이 방법은 차체를 너무 높여서 안정성을 보장하지 못해서 기각했다. 또 만약 카메라를 석션관 위에 바로 단다 하면 카메라와 차체의 중심이 어긋나서 차량의 움직임을 카메라로 제어하는데 어려움이 생길 수 있다고 판단하여 석션관을 옆에서 길게 연장해 차체의 중앙에서 공을 빨아 들일 수 있게 디자인 하였다.

-공을 측면에서 빨아들이는 디자인과 중앙에서 빨아들이는 디자인 비교

추가적으로 쿨링시스템에 관련해선 진공펌프에서 빠져나오는 공기를 이용해 열을 식히는 방안을 교수님께 제시하였다.



금요일날 디자인 리뷰에서 이 디자인은 전력소비가 너무 많다는 문제가 있었고 suction 의 넓이와 압력의 관계, 석션관에서의 마찰이 얼마나 영향을 줄지 등을 좀 더 자세히 알아야 된다고 생각해 김형수 교수님께 이 내용들을 문의해보기로 결정했다.

4/2~4/8

Q1 공을 빨아들인 뒤 파워가 약해지지 않는가?

A1 일단 공이 들어가면 관에 걸리지 않는 이상 그대로 빨려 들어갈 것이다.

Q2 마찰력 때문에 공이 끼지 않나?

A2 공의 요철까지 고려한 반지름으로 관을 만든다면 그렇지 않을 것이다.

Q3 3D 프린터 재료의 거칠기 때문에 압력이 떨어지지 않는가?

A3 그 정도로는 문제 없다.

Q4 테프론을 이용하려면 어떻게 해야 하는가?

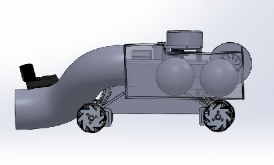
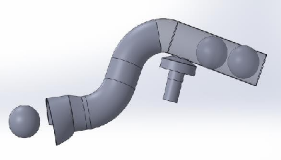
A4 테프론 블록을 깎아 만들긴 무리고 테이프나 스프레이를 알아 볼 것.

Q5 석션펌프에서 방출 되는 공기로 열을 식히려는데 괜찮은가?

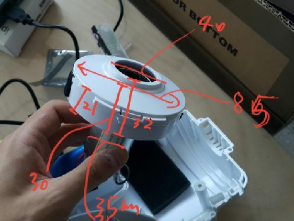
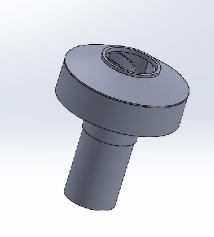
A5 좋은 아이디어이지만 석션 모터에서 방출되는 열로 공기가 뜨거워지지 않게 경로를 잘 만들어야 할 것이다.

4/9~4/14

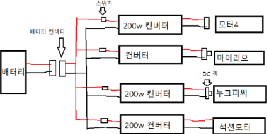
흡입관의 보관함을 박스형태가 아닌 원통형으로 좀 더 간결하게 설계

구입한 청소기의 진공펌프의 수치에 맞게 새로 펌프 디자인을 함.

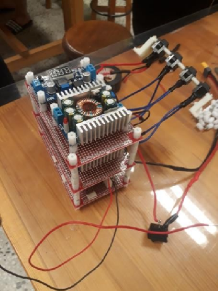
 

컨버터 및 각종 전자부품을 구입하고 납땜 작업을 일부 진행함.

4/16~4/22

납땜 작업을 완료함



흡입관 보관함의 뒷문을 열수 있는 메커니즘을 추가하고 진공펌프와 파이프가 연결된 부위를 설계하였다.



4/23~4/28

-서스펜션을 장착한 차체를 제작

교수님과의 미팅 후 수직으로 올라간 프레임 기둥을 제거하고 스프링을 수평으로 장착시켜 차체의 공간을 확보해 흡입관을 좀더 차량중심에 가까이 위치시키는 방안을 교수님께서 제시해주셨다.



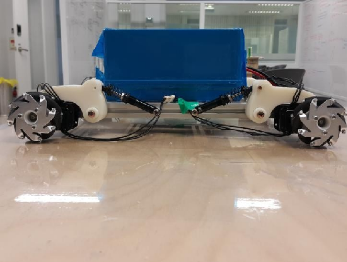
4/30~5/6

흡입관을 완성함.

-테이프를 이용해 임시로 관을 결합하고 진공모터를 장착해 실험해본 결과 공이 관을 따라 막힘 없이 잘 들어갔다.



-새로운 서스펜션을 완성.

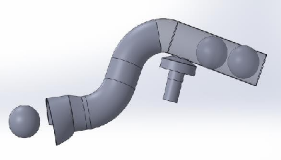
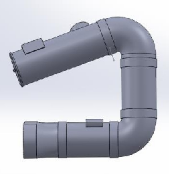


만일 차체의 속도를 기어를 통해 올리거나 하면 진동저감에 유리가 있을 것 같아 서스펜션 시스템을 유지했다.

5/7~5/13

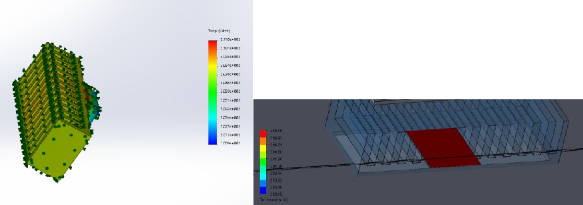
-흡입관 디자인을 수정함.

기존의 디자인으로는 차체가 바구니로 가고 나서 다시 180도를 돌고 문을 열어야 했는데 이는 시간을 굉장히 많이 소비해서 흡입관 디자인을 공을 앞에서 빨아들이고 앞에서 뱉을 수 있게 수정하였다.

-간단한 쿨링 시뮬레이션 실시.

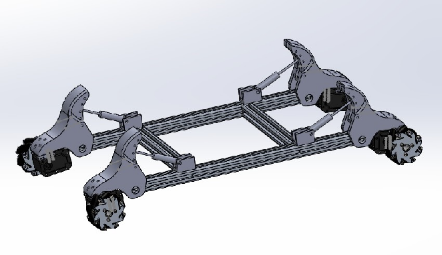
자연대류의 결과 38도 강제대류의 결가 22도로 확연한 차이를 보여주었다.



5/14 ~5/20

-서스펜션을 바꿈

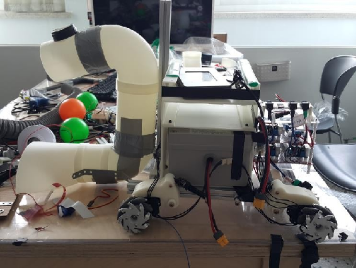
서스펜션의 모양을 수정해 스프링에 가해지는 힘을 줄여주었다. 또한 서스펜션의 휨 모멘트가 작용해 바퀴가 옆으로 퍼지는 것을 줄이는 디자인으로 수정.



-스프링을 업그레이드함

디자인만 바꾼 것이 아니라 스프링 자체도 더 강한 것으로 바꾸었다.

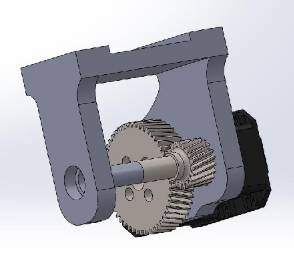
-U자관 완성



5/21 ~ 5/26

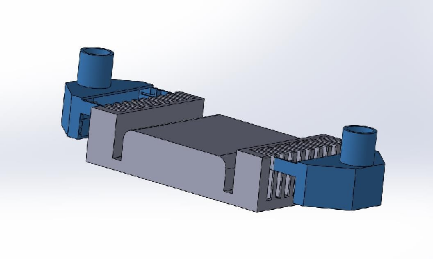
-기어를 제작.

더 빠른 속도를 내기 위해서 기어를 설계하고 제작하였다.



-쿨링 시스템을 디자인함

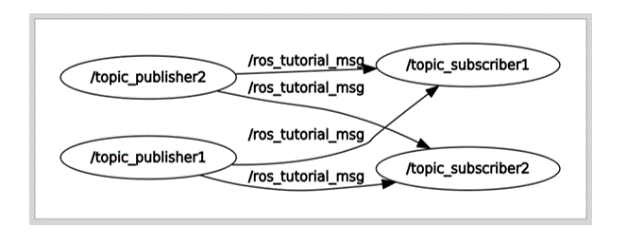
효율적인 쿨링을 위해 컨버터의 알루미늄 핀에 직접적으로 바람을 분배시킬 부품들을 디자인 하였다.



-ROS

3/19 ~ 3/25

ROS 과제가 두가지 있었는데 예제를 구현하는 것과 추가적으로 해야한는 과제가 있었다. C++을 처음 접하는 나로서는 혼자하기 상당히 버거웠다. 그래서 유튜브에 있는 표현식 선생님의 강좌를 보면서 예제르 따라했다. 그렇게 해서 예제를 구현하는 과제는 쉽게 했지만 이것을 응용하는 추가과제는 상당히 어려웠다. 이 과제는 같이ROS를 담당하시는 분의 도움을 받아서 해결을 했다. 이번 창시구를 하기 위해서는 C++을 전혀 모르면 절대 안 된다고 생각해 공부 계획을 세웠고 ROS도 많이 접해야겠다고 생각했다. 이번 주는 그냥 ROS를 한 번 경험해보는 시간이었다.



3/26 ~ 3/31

이번 주는 저번 ROS 강의 시간에서 배웠던 webcam, rplidar를 실습해보는 시간을 가졌다. 웹캠이나 rplidar의 소스코드는 klms에 조교님께서 큰 틀에서 만드신 코드를 사용하였다. Webcam은 USB를 노트북과 연결뒤 Videocapture 부분을 0에서 1로 바꾼뒤 실행하면 웹캠으로 보는 시각정보들이 노트북에 보였다. my\_publisher 라는 node에서 이미지 정보를 출력하고 my\_subscriber라는 node에서 받고 터미널에 출력했다. 다음은 rplidar를 사용했는데 rplidar위의 원판이 돌아가면서 주위를 scan하게 된다. 이것을 사용하기 위해서는 rplidar 패키지 안의 node와 client를 실행해야 한다. 둘을 실행하고 다시 새로운 터미널을 열어서 rviz를 열고 frame을 맞춰준뒤 add 에서 scan항목을 찾아 추가하게 되면 주위가 빨간선으로 뜨게 되는 것을 볼 수가 있다. 그리고 xbox와 노트북을 연결해 xbox정보가 노트북으로 잘 가고 있는지도 확인했다.

4/2 ~ 4/8

3월 달 까지는 최종 완성본에 필요한 node를 실행시키고 실험해 보았다. 그래서 이번 주는 각각의 패키지 안의 node들을 할수 있는 데 까지는 해석해보기로 했다. C++을 이번 창시구를 통해 배우는 나로서는 당장 하기는 어려웠지만 유튜브나 구글에 C++ 초심자를 위한 강의가 많았다 그것을 차근차근 보면서 node들을 한줄 씩 할수 있는데까지 해석을 시도했다. 이 과정을 통해 C++의 기본 함수들이나 float형 int형 등 기본적으로 알아야 할 것들을 많이 알게 되었다. 그리고 저번주에 xbox를 나의 노트북과의 통신을 했다면 이번에는 Labview와의 통신을 해보기로 했다. Labview에서 myrio를 통해 서버를 열어주면 ROS에서 myrio의 서버주소를 입력하고 xboxnode를 통해 서로 통신을 할 수 있다. 우리 조는 터미널에서는 연결이 잘 되었다고 뜨는데 xbox버튼을 누르면 모터에 아무런 반응이 없었다. 아마 Labview 코드에 살짝 문제가 있는 것으로 보였다.

4/9 ~ 4/14

이번 주는 Opencv의 Calibration node를 ROS의 ball detection node를 합치는 시도를 했다. 일단 ROS 내에 ball detection node를 공부하기로 했다. 애초에 코드가 너무 길고 복잡해서 한줄 한줄 읽어도 제대로 와닿지가 않았다. 그래도 조교님께서 한줄 한줄마다 주석을 달아놔서 이 줄은 무엇을 뜻하고 어떤 역할을 하는지 간단하게 설명해준 덕분에 대충은 이해를 했다. 하지만 코드라는 것이 주석을 보고 이해를 한다고 해도 코드를 짜는 것은 전혀 상관이 없다. 아무튼 둘이 합치려면 Open cv 코드도 알아야 하기 때문에 Calibration node도 같이 보면서 공부를 했다. 일단 Open cv이에 있는 void함수는 통합할 때 무조건 필요할 것 같아 죄다 복사해서 ROS의 ball detection node에 넣었다. Include도 마찬가지로 넣었다. 그리고나서 opencv이의 main함수를 보면서 어떻게 통합을 할지 생각을 해보았다. 단순히 합치기엔 Opencv 코드도 이미지 정보를 받고 있고 ROS코드도 이미지 정보를 받고 있어서 합칠 때 이미지 정보를 두 번 받지 않게 해결도 해야 한다고 생각했다. 우리로선 최선의 방법으로 생각을 해서 코드를 합쳐보았다. 이렇게 코드를 다 합치고 실행한 결과 erorr message는 뜨지가 않았다. 그래서 희망을 가지고rviz를 통해 opencv에서 나오는 것들이 똑같이 나오나 실험을 해 본 결과 나오지 않았다. 아직 C++지식이 얕아서 제대로 되지 않는다고 생각했다. ROS를 맡은 이상 안일하게 대충 공부하면 안 된다고 생각을 했다.

4/16 ~ 4/22

이번 주는 일단ball detection node의 통합은 뒤로 제쳐두고 저번에 실패했던 Labview와의 xbox통신을 무조건 성공시키다는 각오로 xbox통신을 시도했다. ROS에서 xbox를 통해 내보내는 정보는 24개의 float이고 우리는 a,b,x,y 버튼을 써서 컨트롤 하기로 했다. 저번에 통신에 실패한 데에 이유를 찾기 위해 조교님을 모셔와서 도음을 받기로 했다. 그랬더니 노트북에서 xbox 를 통해 보내는 정보는1인데Labview상으로는 4곱하기 10의 마이너스 40승 가까이 되는 것이었다. 이러니 당연히 통신은 되고 있지만 ROS에서 주는 정보를 Labview에서 제대로 받지 못하니 모터가 작동하지 않은 것이었다. 이 문제를 조교님 덕분에 알게 되었고 조교님께서 조언을 해주어서 Labview상에쇼 다시 1로 인식할 수 있도록 되게 하였다. 그리고 나서 실험을 해본 결과 모터가 xbox 버튼을 누르자 돌아가기 시작했다. 최종데모에서는 xbox를 통해 하는 것이 아니라 무인로봇이기 때문에 이를 data integration node에서 xbox에서 정보를 주는 것처럼 우리가 특정 상황에서 다른 데이터를 주면서 어떻게 Labview와 통신할지에 대해 고민해보기로 했다.

4/23 ~ 4/28

이번 주는 저번주에 랩뷰와의 통신을 끝냈기 때문에 홀가분한 기분으로 ball detection node의 통합을 끝내는 것을 목표로 삼았다. 여기서 통합이란 opencv의 main.cpp라는 코드를 ROS의 ball\_detection\_node를 잘 합쳐 실행했을 때 opencv를 통해 감지된 ball들이 ros rviz상에서 나오고 감지외어야 한다는 것이다. 두 코드가 따로 놀아서도 안되고 완전한 하나의 코드가 되어야 한다. 처음에는 opencv에서 필요로 하는 해독 파일들을 전부 끌어다 놓고 필요한 코드를 복사 붙여넣기 해서 합쳤다. 이렇게 하니 너무 바쁘다라는 내용의 error가 떴다. 그래서 코드에서 웹캠을 실행하는 것을 지웠다. Webcam이 두번 켜지는 것 같아서이다. 아무튼 이렇게 하고나서 실행해보니 ball이 잘 감지 되는 것처럼 보였지만 rviz를 실행하고 marking을 해보니 전처럼 여전히 공이 방방거리고 제대로 detect되지 않는 것을 볼 수 있었다. 이 부분은 나로서는 너무 해결하기 어려웠다. 둘의 function도 너무 다르고 해서 C++초보인 나로서는 상당히 고충이었다. 그런데 같이 ROS를 하시는 분께서 잘하셔서 rivz에서marking되는 ball들도 opencv에서 보는 ball과 같도록 고치셨다. 통합에 성공한 것이다. 아직 잘 통합되는 과정과 코드를 제대로는 모르지만 코드를 다시 한번 읽고 조원에게 물어보면서 이해해야 할 것 같다. 그리고 ball을 봤을 때 어떻게 가고 어떻게 ball을 suction할 것인지 고민을 해보았다.

4/30 ~ 5/6

이번 주는 금요일에 2차 발표가 있어서 모두가 바빴다. 하드웨어 쪽도 바쁘고 소프트웨어쪽도 다들 바빴다. ROS 쪽에서는 Labview와 통신도 성공했고 opencv와의 통합도 성공했기 때문에 data integration node를 짜기 시작했다. 이번 발표에 보여줄 영상을 만들기 위해 파란공 3개를 인식해서 다 suction하는 코드를 짜기로 했다. 빨간 공을 피하는 코드는 나중에 넣기로 했고 당장 시간이 없어서 파란 공만 가지고 코드를 짰다. 통합된 ball\_detection\_node에서 출력된 공의 정보들을 이용해 가장 가까운 파란 공의 거리를 가지고 그 거리가 얼마 이하이면 직진하고 직진하면서 각도가 벗어나면 시계방향으로 돌고 시계 반대방향으로 돌고 각이 맞고 어느 거리 안이면 그대로 직진하면서 suction해 공을 먹는 코드를 구현했다. 이렇게 짜고나서 로봇을 움직여 보니 파란공 3개를 잘 찾아서 가서 먹는 것을 볼 수 있었다. 지금까지 ROS하면서 가장 보람찬 순간이었다.

5/7~5/13

저번 주에 파란공 만 찾아가서 먹는 코드를 구현했으니 이제는 빨간 공을 보면 피하는 것과 바구니위에 green ball을 통해 인식하게 해놓았기 때문에 green ball을 인식해서 바구니까지 가서 공을 내려놓는 코드에 대해 생각해보았다. 일단 코드가 4개의 상황으로 나누어졌다. 파란 공과 빨간 공 다 있는 경우, 파란 공은 있고 빨간 공은 없는 경우, 빨간 공은 있고 파란 공은 없는 경우, 둘 다 없는 경우 4가지로 나누었다. 이제 생각해야 할 것은 처음에 시작할 때이다 처음에 공이 아무것도 안보이면 보일 때까지 가는 path planning을 해야 하는데 이것을 하기 위해서 우리는 while문이 0.001초에 한 번 씩 도는 것을 이용했다. While문 밖에서 t를 정의하고 whiile 문 안에서 계속 더해주면서 이 t가 10초 이하일 때까지 계속 직진하게 만들었다. 이렇게 하니 충분히 공을 보일 수 있는 데까지 로봇이 이동할 수 있었다. 그리고 빨간 공이 보이면 x좌표를 가지고 얼마 이하이면 왼쪽 오른쪽으로 피하고 얼마 이상이면 무시하도록 했다. 그리고 중요한 것은 볼 3개를 다 먹고 어떻게 다시 basket까지 갈까였다. 이것을 하기 위해 counting을 하였다. Suction이 켜질 때 마다 1씩 count가 올라갔고 3이 될 때 오직 green ball의 정보만 받아 바구니까지 가도록 하기 위해서이다. 처음 green ball을 찾을 때 Orientation을 찾기 위해 애초에 turn left, turn right를 할 때마다 특정 각도를 계속 더해주었고 이것을 가지고 green ball을 찾을 때 얼마로 나누어 주어서 특정 범위안에 들어가게 해주어서 green ball이 있는 basket을 볼 수 있게 하였다. 이렇게 green ball을 detect하게 되면 두 green ball의 중점을 이용해 중점을 얼마 범위 안에 계속 맞추게 함으로써 로봇을 움직이게 했다. 이렇게 코드를 만들고 로봇을 움직였더니 green ball까지 가는 과정은 좋았으나 회전속도가 빨라서 그런지 counting이 마음대로 하나 먹을 때마다 1씩 올라가지 않았다. 이 문제를 해결해야한다고 생각했다.

5/14 ~ 5/20

저번 주에 파란공 3개를 찾고 green ball을 인식해 돌아오는 것까지 코드를 구현했기 때문에 이번 주는 기존 코드를 업그레이드 시키는 쪽으로 진행했다. 현재 알고리즘의 문제는 파란공의 x범위를 맞추기 위해 회전하는데 범위를 맞추려다 보니 또 벗어나고 또 안으로 들어오고를 계속 반복했다. 이것을 해결하기 위해 일단 x범위를 웹캠과 공이 이루는 각도록 변경했다. 그래서 특정 각 안에 들어오면 직진하게 알고리즘을 수정했다. 그리고 교수님과의 미팅에서 얻은 아이디어로 하나를 더 구현했는데 각도를 두개 만들어서 더 큰 각 범위안에 들어올때가지느 빠르게 돌게 하고 더 작은 각 범위 안에 들어오면 느리게 돌게 했다. 이렇게 하니 예전 문제인 로봇이 렉 걸린 것처럼 왔다갔다 하는 걸 피할 수 있었고 counting도 1씩 자연스럽게 늘어났다. 또 바구니에 갔을 때 먹은 공들을 떨어뜨리기 위해 서브모터도 작동하게 코드로 구현했다. 이렇게하고 로봇을 작동시켰더니 공을 정교하게 찾지않아도 앞에 깔대기가 있어서 공을 모아주기 때문에 잘 먹는 것을 볼 수 있었다. 더 좋은 알고리즘이 생각난다면 바로바로 바꿀 것이다.

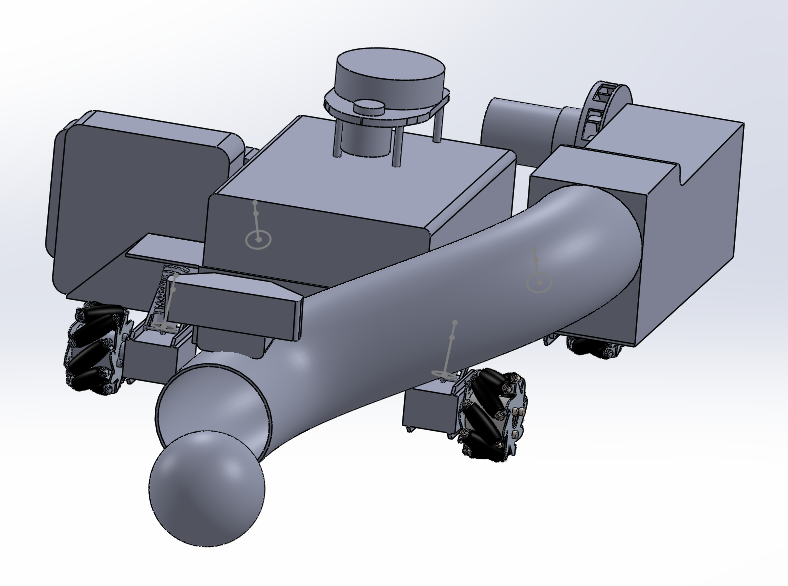
5/21 ~ 5/26

교수님과의 미팅에서 이렇게 각도를 두개 정하고 큰 각도 범위에서는 빠르게 돌고 작은 각도 범위에서는 천천히 도는 것 대신에 아예 pd제어기를 사용하는 것이 어떠냐고 제안을 하셨다. 그래서 우리는 ROS 코드 상에서 pd control을 사용한 새로운 알고리즘을 짜기로 했다. 이 방법이 전 방법보다 더 smooth하게 공 근처에서는 속도를 줄여버리기 때문에 상당히 괜찮은 방법이라고 생각했다. pd제어기를 설계하기 위해서 많은 변수가 필요했다. 시스템 모델링 제어 시간에서 배운 pd제어기를 떠올리며 파란공과 웹캠이 이루는 각도를 현재 error로 설정하고 목표는 error를 0으로 만드는 것이라고 생각해 error값과 error를 수치미분한 값에 kp와 kd를 곱했다. 다만 error를 미분한 값이 많이 튈수도 있기 때문에 low pass filter를 써서 튀는 것을 막아주었다. 이 filter도 수치적으로 접근해 코드를 구현했다. 이렇게 pd제어기를 만들고 로봇을 작동시켰더니 전에는 각도 범위 2개에서 딱딱 끊겨서 회전했다면 이것은 정말 부드럽게 느려지면서 공과 웹캠의 사이 각도를 0도로 만들었다. 물론 p gain값과 d gain값을 잘 맞춰야 가능 한 것이었다. 이 gain의 이상적인 값들을 찾기 위해서 정말 수도 없는 실험을 했다. 또 하드웨어의 기어를 달게 되었는데 이것 때문에 회전 속도가 상당히 빠른것에 pd제어기가 좋은 역할을 할 수 있었다. 다만 green 공을 찾기 위해 계산하는 각도가 pd제어기로 연속적으로 각도를 변하면 제대로된 orientation을 찾을 수가 없었다. 그래서 그냥 green공이 무조건 보인다고 가정하고 파란공 3개를 다 먹으면 회전하도록 했다.

2) Presentation material

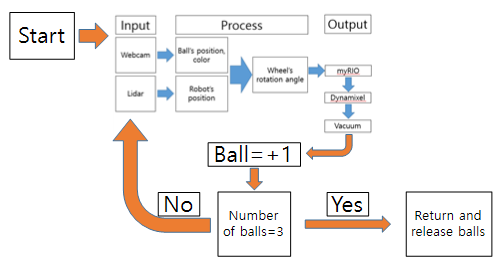
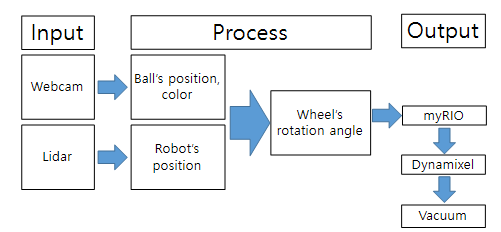
★First presentation

-Overview



배터리를 최대한 중심에 낮게

-Mechanism



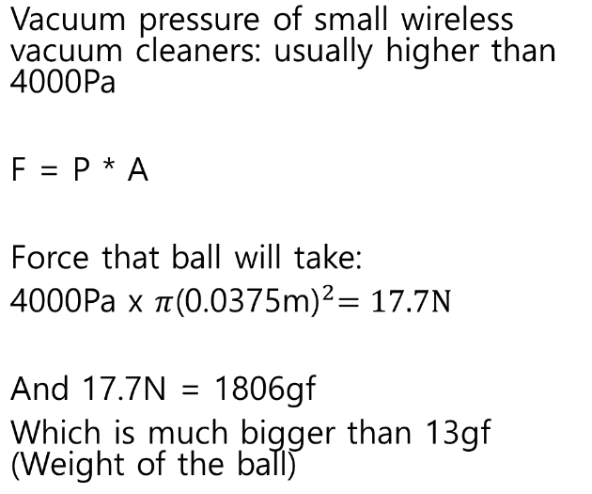
-Why we choose suction?

1) Characteristic of balls

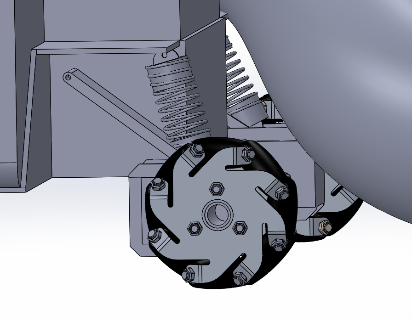
2) Enerby efficiency

3) Time efficiency

-Is it possible?

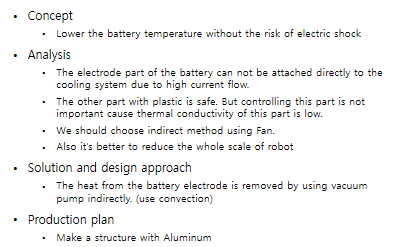


-Why we use suspension?

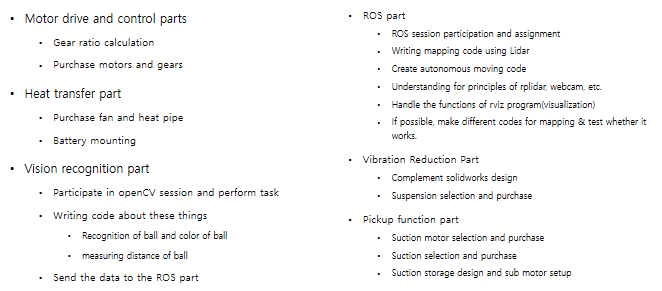
 Shape of the Mecanum wheel generates vibaration

Vibration could be bad for webcam and rplidar

-Cooling system

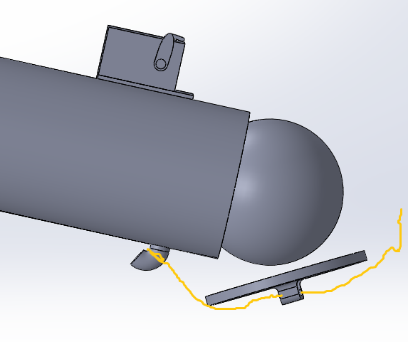


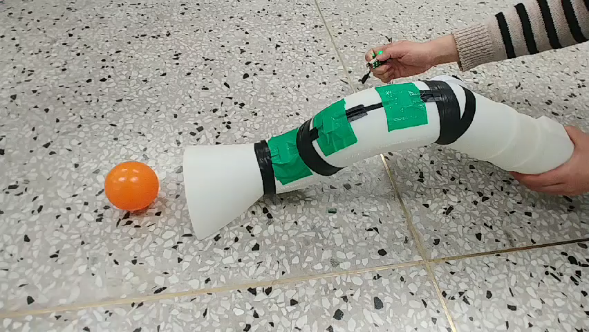
-Future plan



★Second Presentation

- Designing vacumm system





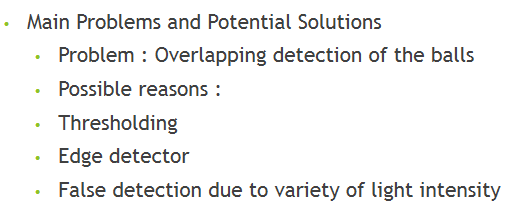


-Designing suspension system

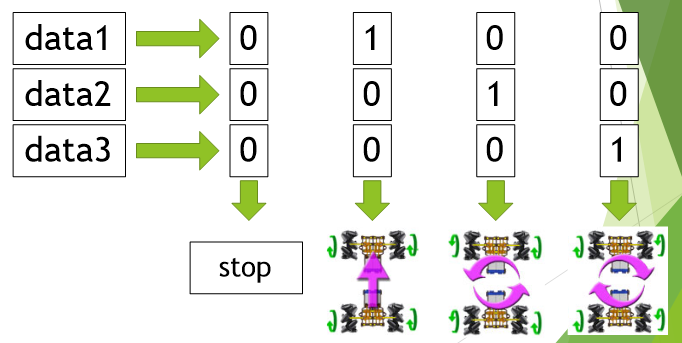
We assumed 3kg for the whole robot

The weight of car is about 1500kg

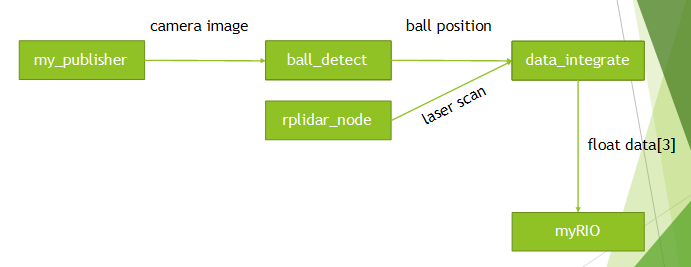
-Opencv



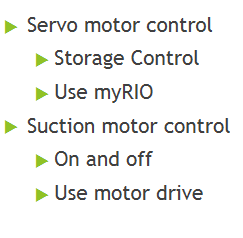
-Labview



-ROS algorithm

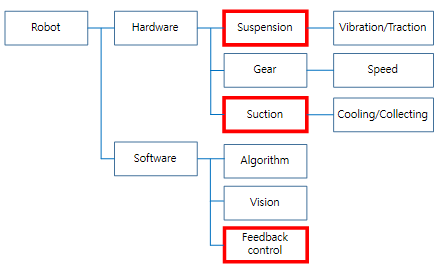


-Future work



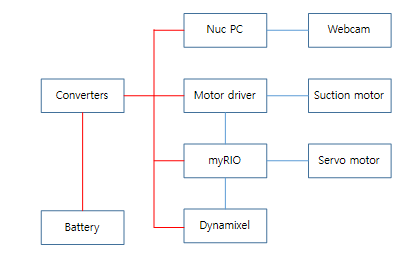
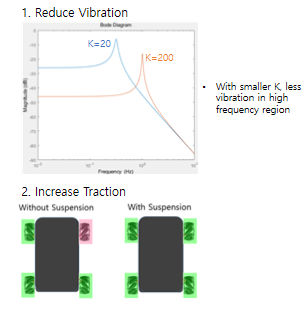
★Final presentation

-system introduction

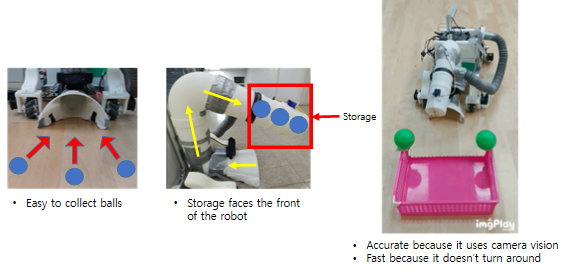


-Hardware

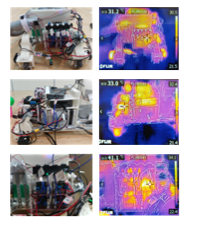
-Circuit -Suspension

-Suction Process

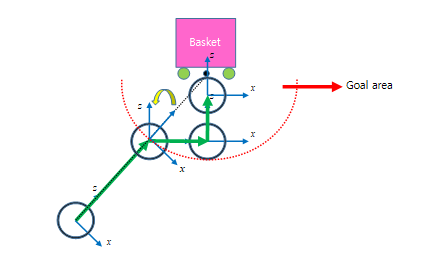
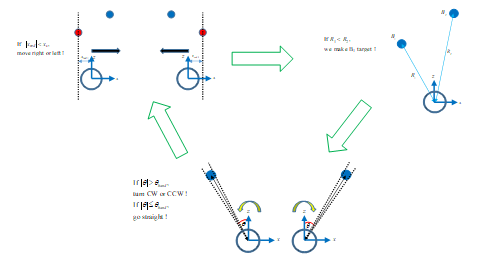


-Cooling

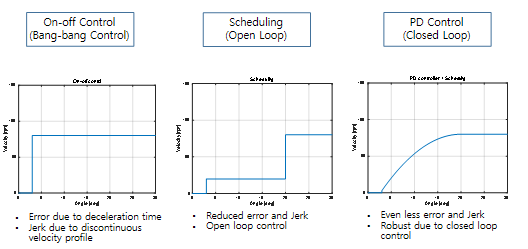


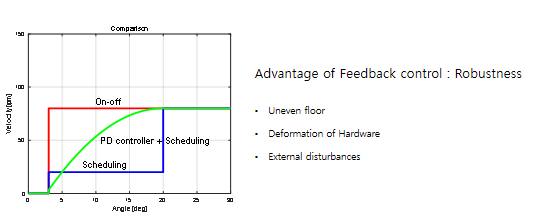
-Software

-Algorithm



-PD control





3) 후기

이번 학기부터 창시구가 전면적으로 바뀐다는 것을 첫 시간에 알았다. 공학설계의 연장점이라고 생각했었는데 모든 조들이 같은 주제, 같은 목표를 가지고 경쟁하는 것이었다. 공을 줍고 바구니에 넣는 로봇을 만드는 것이 목표라고 들었을 때는 과연 이것이 정말 가능한가라고 속으로 생각했었다. 이번 창시구 부터는 각자 파트를 맡아서 진행하였는데 나는 전체 통합파트인 ROS를 맡게 되었다. 다른 소프트웨어쪽은 해본 것이 많아서 새로운 것을 배우고 싶어서 선택했다. 하지만 C++을 배워본 적도 없는 나는 처음에 상당한 고충을 겪었다. 당연히 따라갈 수 있을 정도의 난이도였을 줄 알았는데 1을 알려주고 10을 하라는 것이다. 나로서는 너무 당황스러웠다. 이런 프로젝트를 할려면은 그 전에 기계과를 위한 프로그래밍이라든지 강좌가 개설되어야 했을 것이다. 갑작스럽게 창시구가 바뀌어서 나 같은 사람들이 많은 불만감을 표출했을 것이다. 이 문제는 독학이라든지 겨우겨우 이겨낼 수는 있었다. 또 하나의 문제점은 열심히 코드를 만들어서 로봇을 작동시키면은 코드 문제가 아닌 통신문제로 도저히 해결할 수 없는 상황이 생긴다. Myrio 의 와이파이가 잘 잡히지도 않을 뿐더러 로봇과 통신에도 제대로 되지 않으니 진이 빠질 수 밖에 없다. 개인적으로는 이번 창시구가 갑작스런 변화를 시도해서 많은 문제점도 나왔지만 지금 언급한 문제점만 없다면 충분히 재밌을 수 있을 것 같다. 처음에 이것이 과연 가능할까 라고 생각한 나는 로봇이 완성되고 몇번의 데모에서 우리 로봇이 공 3개르 줍고 바구니에 떨어뜨릴 때마다 절대 안될 것 같은게 되니깐 정말 보람찼다. 결론은 이번 창시구를 통해 많은 것을 배울 수도 있었지만 너무 갑작스럽게 창시구가 바뀌어서 사전 준비를 제대로 하지 못했다는 것이 아쉬웠다. ROS같은 것처럼 C++을 사용하는 소프트웨어는 미리 공부할 수 있게 하던지 학기 창시구를 하면서 C++수업을 많이 해주었으면 정말 좋았을 것 같다. 이런 문제점만 해결된다면 졸업하기전에 정말 좋은 경험을 기계과에서 얻을 수 있을 것 같다. 다음 후배들 부터는 이런 문제점이 해결되어서 불만 없이 재밌게 창시구에 참여했으면 한다.