ME400 : 창의적 시스템 구현

최종보고서

작성자 : 20150388 사공철

Team : 걷지말고 기어

교수님 : 박용화 교수님

조교님 : 김수용 조교님

조원

20150094 김덕영 (ROS)

20150121 김민경 (Solidworks)

20150388 사공철 (LabVIEW)

20150510 윤상원 (LabVIEW)

20150579 이예찬 (Solidworks)

20150760 최 원 (Open CV)

20150813 홍성웅 (ROS)

목차

1. Process

1) Plan

Plan in March

Solidworks, LabVIEW, Open CV, 그리고 ROS 각자 담당 파트에 대해서 공부하는 것을 목적으로 하였으며, picking 방법을 정하고 대략적인 디자인을 구성하는 과정을 거쳤다. 4월부터 디자인의 확실한 구상 및 각 파트의 통합을 시작하였다.

2) Whole Process

실제 프로세스는 Plan 과는 조금 다르게 진행되었다.

실제로는 Plan 보다 살짝 빠르게 진행되는 경향이 있었다.

이 내용들은 각 파트의 progress report 를 참조하여 작성하였다.

(1) Solidworks

3월 19일 – 3월 25일

롤러 방식으로 결정된 아이디어에 대해서 solidworks 설계를 시작하였으며, 차체 프레임의 두 가지 아이디어를 조교님께 컨펌 받았다.

3월 26일 – 4월 1일

메카넘휠과 기어, dynamixel, 차체의 어셈블리를 제작하였고, 몸통과 picking module 을 추가적으로 제작하였다.

4월 2일 – 4월 8일

팀미팅에서 정한 dimension 을 바탕으로 롤러 부분과 공이 저장소로 넘어가는 부분을 제작하였다.

4월 9일 – 4월 15일

1차적으로 제작한 받침대를 접착하여 프로토타입을 만들었으며, 팀미팅을 통해서 수정된 받침대와 롤러를 다시 제작하였다.

4월 16일 – 4월 22일

롤러 부분과 경사로의 3D 프린팅이 완료되었다. 이를 과학상자와 합쳤다.

4월 23일 – 4월 29일

플라스틱을 이용하여 프로토타입을 제작했지만, 문제가 많아 아크릴판을 이용하여 다시 제작하였다. 그러나 크기가 크고 무거워서 문제가 많다.

4월 30일 – 5월 6일

기어와 motor 를 합쳤으며, 롤러 모터와 롤러 축을 연결하여 롤러를 완성시켰다.

5월 7일 – 5월 13일

기어축에 슬립링을 끼워 슬립을 방지하였고, 축의 이동으로 인한 뒤틀림을 보정하였다. 바닥면에도 추가적인 프로파일을 끼워 기어로 인한 차체의 뒤틀림을 방지하였고, 기어에 의한 모터의 회전을 방지하기 위해 외부에 L자형 고정단을 추가로 부착하였다.

5월 14일 – 5월 20일

Backdoor 부분을 제작하여 작동하는 것을 확인하였다. Storage 의 경사가 완만하여 공이 잘 내려가지 않는 문제가 발생하였다. 롤러 부분을 다시 연결하였다.

5월 21일 – 5월 27일

차체가 rigid 하여 공동강의실의 평평하지 않은 바닥에서 4점 지지가 되지 않아 문제가 발생하여 프로파일을 나누어 자유도를 주었고, 차체 네 축의 고정을 느슨하게 하였다.

5월 28일 – 6월 1일

중간에 롤러 모터의 고장이 발생하는 바람에 기존에 사용하던 모터보다 힘이 센 모터를 사용하게 되어 picking 부분의 가림막을 다른 형태로 계속 바꾸었다.

(2) LabVIEW

3월 19일 – 3월 25일

LabVIEW 에서 myRIO 프로젝트를 만들어 랩뷰 코드와 myRIO 사용에 대해 이해하는 기간을 가졌다.

3월 26일 – 4월 1일

Dynamixel 을 PC Command 를 이용해서 회전시키고, TCP/IP 통신에 대해서 공부하는 기간을 가졌다.

4월 2일 – 4월 8일

TCP/IP 통신을 할 수 있는 코드를 직접 짰고, 그를 이용해 모터를 제어하는 데에 성공했다.

4월 9일 – 4월 15일

X Box 를 이용하여 모터를 제어할 수 있게 하기 위하여, example 에서 주어진 bi-directional connection 이 가능하게 짜진 코드를 변형시켰다.

4월 16일 – 4월 22일

차체의 움직임을 총 10가지 mode(앞, 뒤, 좌, 우, 그리고 4가지 대각선 방향, 시계 방향, 반시계 방향 회전) 로 나누어 제어할 수 있게하는 코드를 짰다.

4월 23일 – 4월 29일

ROS 와 통합하였다. 24개의 float 을 받아오므로 96 바이트를 읽어오도록 설정하였으며, 느린 반응 속도는 loop time 을 1ms 로 바꿔서 해결하였다.

4월 30일 – 5월 6일

롤러 모터를 구동시키는 코드를 추가하였다. 바퀴와 같은 wheel mode 를 사용하도록 코드를 짰다.

5월 7일 – 5월 13일

공을 내려놓기 위한 backdoor 를 여는 코드를 추가하였다. 이는 다른 모터들과는 다르게 position mode 를 사용하기로 하였다.

5월 14일 – 5월 20일

Backdoor 코드에서, 작동시킬 때마다 시작하는 position 이 달라지는 문제가 있음을 확인하였다.

5월 21일 – 5월 27일

Backdoor 또한, wheel mode 를 사용하여 일정시간 동안 신호가 들어오면 계속 돌아가는 방식으로 바꾸었다. 또한, 공동강의실 바닥 문제 때문에 대각선 방향으로의 이동 속도를 증가시켰다.

5월 27일 – 6월 1일

최종 demo 때까지 롤러 모터의 알맞은 회전 속도를 맞추기 위해서 계속해서 값을 바꾸었다.

(3) Open CV

3월 19일 – 4월 1일

Camera Calibration 을 수행하였다. 그리고 noise 를 제거하여 물체의 경계면을 정확하여 하였으며, Thresholding operation 을 통해 공의 색을 인식하였고, Laplace Operation, Canny edge detector, Hough circle transform, finding contour, creating bounding boxes and circles for contours 등의 방법을 이용하여 공의 모양을 인식하였다.

4월 2일 – 4월 8일

Ball detection 예제 코드를 이용하여 공을 인식하고, 위치를 detect 하였다. HSV 색상과, canny edge detector 를 이용하였다.

4월 9일 – 4월 15일

Ball tracking 방식과 mapping 방식, 그리고 물체의 위치를 파악하는 데에 stereo camera 가 유리할 수 있어 추가적인 카메라 사용을 고민하였다.

4월 16일 – 4월 22일

카메라가 최대 시야를 확보하기 위하여 16:9의 비율로 영상을 받는다. 또, 기존에 주어진 정보를 바탕으로 공이 위치할 군집의 위치를 지정한다.

4월 23일 – 4월 29일

ROS 의 ball detection node 에 기존에 작성했던 코드를 합쳤다.

4월 30일 – 5월 6일

웹캠의 최적의 위치와 각도를 결정하였다. 높이 32cm 에 지면으로 23.5 도 기울이는 것이다.

5월 7일 – 5월 13일

초록색 공을 detect 하는 코드를 추가하였고, 실험적으로 카메라의 각도를 23.5 도로 확정지었다.

5월 14일 – 5월 20일

진동분석을 위해 영상을 촬영해 데이터를 수집하는 코드를 작성하고 작동하였다.

5월 21 – 5월 27일

진동의 원인을 찾으려 했으며, 초록공을 두개로 인식하는 현상을 수정하였다.

(4) ROS

3월 19일 – 3월 25일

ROS를 사용하여 node 생성, 한 node 가 여러 역할을 하도록 하는 방법, topic 통신, service 통신, roslaunch 함수를 사용하여 여러 패키지를 동시에 실행하는 방법 등을 익혔다.

3월 26일 – 4월 1일

NUC 에 우분투와 ROS 를 설치하였다. Lidar 를 실행하였고, rviz 로 lidar 의 데이터를 시작적으로 보여주는 등의 실행을 해보았다.

4월 16일 – 4월 22일

Xbox_ctrl node 가 어떻게 구동되는지 확인하였고, 노트북에서 서버를 열고 통신이 되는지 확인하였다.

4월 23일 – 4월 29일

랩뷰에서 byte 와 case 가 반복되는 시간을 수정하여 ROS 에서 xbox 의 데이터를 전송하는 것을 myRIO 에서 실시간으로 받아들일 수 있도록 하였다. (ROS 와 LabVIEW 통합) 또한 Open CV 와 통합하였으나, 이미지 출력이 되지 않는다는 문제가 존재하였다.

4월 30일 – 5월 6일

공을 detect 하고 자동으로 공을 주우러 가는 알고리즘을 완성하였다.

5월 7일 – 5월 13일

파란공 3개를 주웠다는 사실을 판단하는 방법에 대해서 생각해보았다. Pick up motor 의 회전횟수, 그리고 sensor 를 동시에 이용하기로 했다.

5월 14일 – 5월 20일

Pick up motor 가 세번 작동하면 공을 모두 주운 것이라고 판단하여 basket 으로 돌아오도록 하는 코드를 짰다. 초록색 공들의 좌표들의 평균값을 목표값으로 하여 돌아가도록 하자는 의견이 있었다.

5월 21일 – 5월 27일

기존의 알고리즘이 공을 바구니에 담지 못하는 것으로 판단되어 두 초록색 공의 수직이등분선 위에 차체를 위치하여 정확이 바구니를 보도록 한 후 180도 회전시키는 방식으로 코드를 바꾸었다.

5월 28일 – 6월 1일

롤러 모터가 힘이 센 모터로 바뀌면서 공이 높이 튀어 오르는 등의 문제가 발생하여 코드의 수정을 거쳤다.

(5) 전체 (팀 미팅)

3월 19일 – 3월 25일

팀 미팅을 통해서 picking 방식과 heat transfer 등의 방식을 이야기하였고, decision matrix 를 통해서 좋은 방식을 결정하였다. 이를 페트병과 고무줄을 이용하여 만들어보았다. 하지만 내려놓는 방식에 대해서 아이디어를 낼 수 없어 롤러 형식으로 바꾸었다.

3월 26일 – 4월 1일

롤러의 문제점인 '공을 쳐낼 수 있다' 는 부분을 해석하였고, 첫 번째 발표 준비를 하였다.

4월 2일 – 4월 8일

발표 이후, 여러 조가 비슷한 아이디어를 향해 가고 있음을 알 수 있었고, 차별화 전략에 대해서 이야기해보았다. 컨버터 스펙에 대해 의논하였으며, Mobile Platform의 빠른 속도를 위하여 기어를 사용하기로 하였다.

4월 9일 – 4월 15일

컨버터를 인터넷으로 주문하였다. 롤러와 바닥 부분에 대한 수정이 필요해져 의논을 거쳤다.

4월 16일 – 4월 22일

컨버터가 도착하여, 모터를 구동하고 NUC 와 myRIO 에 전원을 공급할 회로를 짜고 납땜하였다.

4월 23일 – 4월 29일

롤러 모터가 지급받은 코드로 작동하지 않는 것을 확인하여 다른 모터로 다시 주문하였다. 그리고 기어가 도착하였다.

4월 30일 – 5월 6일

공을 바닥에 놓고 실제 상황처럼 파란공 3개를 수집하고 빨간공은 피하는 과제를 자율주행으로 수행하는 데에 성공하였다. 다만 모터의 토크가 약해 다른 모터를 찾아보았다. 열과 진동에 대한 분석도 하였으며 2차 발표 준비를 하였다.

5월 7일 – 5월 13일

기존에 차체가 틀어져 기어가 잘 작동하지 않는 일이 벌어져서 하드웨어를 재조립하였다. 그리고 롤러 모터를 바꾸는 것보다는 picking 의 경사를 곡선이 아닌 직선으로 바꿔서 해결하자는 결론이 나왔다. 따라서 picking 과 경사로의 전체적인 디자인이 바뀌어 다음주에 하드웨어가 새로 완성될 예정이다.

5월 14일 – 5월 20일

공동강의실에서 데모를 수행할 기회가 주어졌으나, 공동강의실에 랩뷰를 실행할 수 있는 컴퓨터가 없었기 때문에 작동시켜보지 못하였다. 따라서 ball detection 만 확인하였고, 공동강의실의 환경에 맞게 ball detection 코드를 수정하는 과정을 거쳤다.

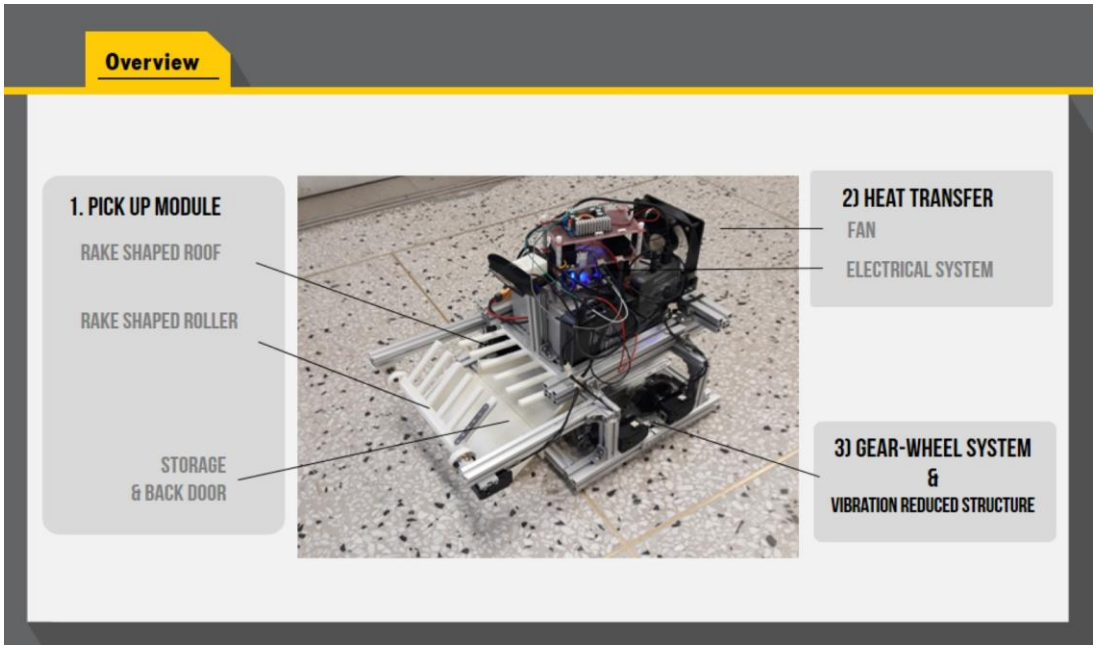
5월 21일 – 5월 27일

공동강의실에서 mobile platform을 가동시켰을 때, 바닥이 울퉁불퉁하여 4점지지가 일어나지 않아 대각선이나 좌우 방향으로의 이동에 어려움이 있음을 발견하였다. 따라서 이를 해결하기 위해 자유도를 주고, 대각선 방향으로의 속력을 올렸다.

5월 28일 – 6월 1일

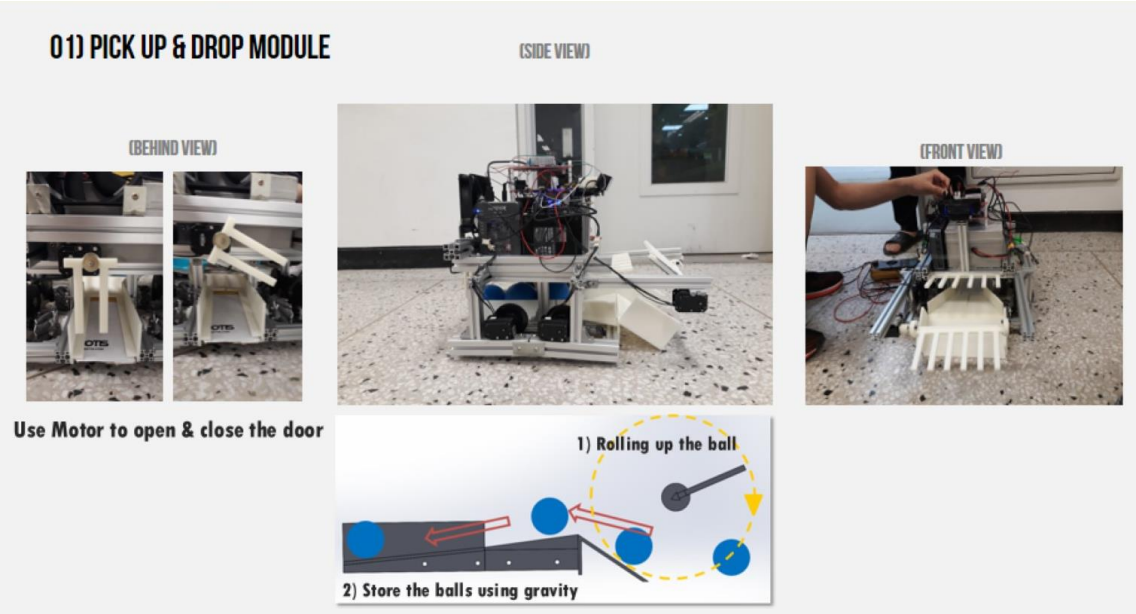
고장난 모터를 바꾸고 난 후, 롤러의 힘이 필요 이상으로 강해졌기 때문에 기존의 picking design 에서는 문제가 많았다. 하지만 발표 기간이 다 되어서 전체적인 구조를 바꿀 수 없었기 때문에 picking 이 잘 되는 최적의 각도를 여러 번의 시행을 통해서 찾았고 공이 튀어 나가지 않도록 하는 가림막의 디자인을 살짝 바꾸었으며 롤러 모터의 회전 속력을 낮추어 문제를 해결하였다.

2. Hardware



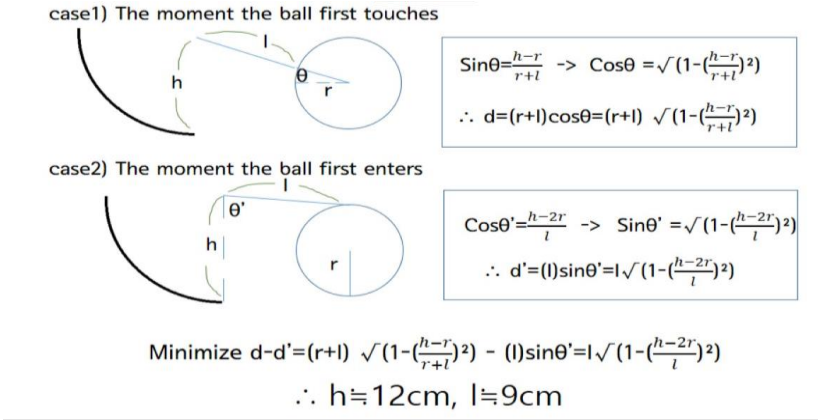
전체적인 디자인은 위와 같다. 핵심은 picking module 과 fin and fan, 그리고 기어이다.

1) Picking module



위 그림은 picking module 의 전체적인 구조이다. 공은 빗 모양의 롤러를 통해서 수집되어 저장고로 이동되고, basket 앞에서 뒷문을 회전시켜 공이 basket 에 들어가도록 하였다.

(1) 롤러



롤러의 경우, 공이 쓸려 들어오지 않고 오히려 바깥으로 밀어낼 수도 있다. 따라서 위와 같은 분석을 통하여 공을 밀어낼 확률을 최소화 하는 값으로 롤러의 중심축의 위치와 길이를 결정하였다.

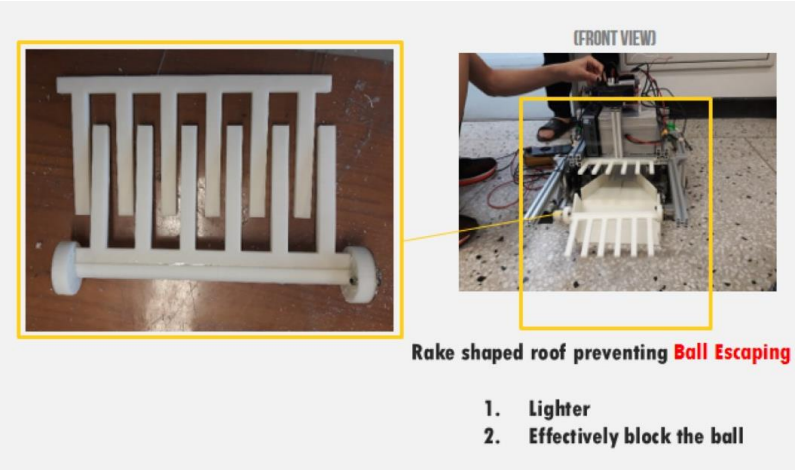
(2) 저장고

저장고를 약간의 경사를 이용하여 만들었다. 따라서 공을 내려 놓기 위해서 별도의 에너지를 사용하지 않고, 중력만으로 공을 내려놓을 수 있었다.

(3) Backdoor

Backdoor 의 경우에도 위로 열거나 아래로 여는 등의 아이디어가 있었고, 에너지 사용을 줄이기 위해 모터를 사용하지 않는 방법들이 논의되었으나 뒷문을 여는 데에는 단 몇 초만 필요하기에 모터를 사용하기로 하였다. 뒷문의 경우, 단순히 회전시키는 것만으로 공을 내려놓을 수 있도록 간단한 디자인을 선택하였다.

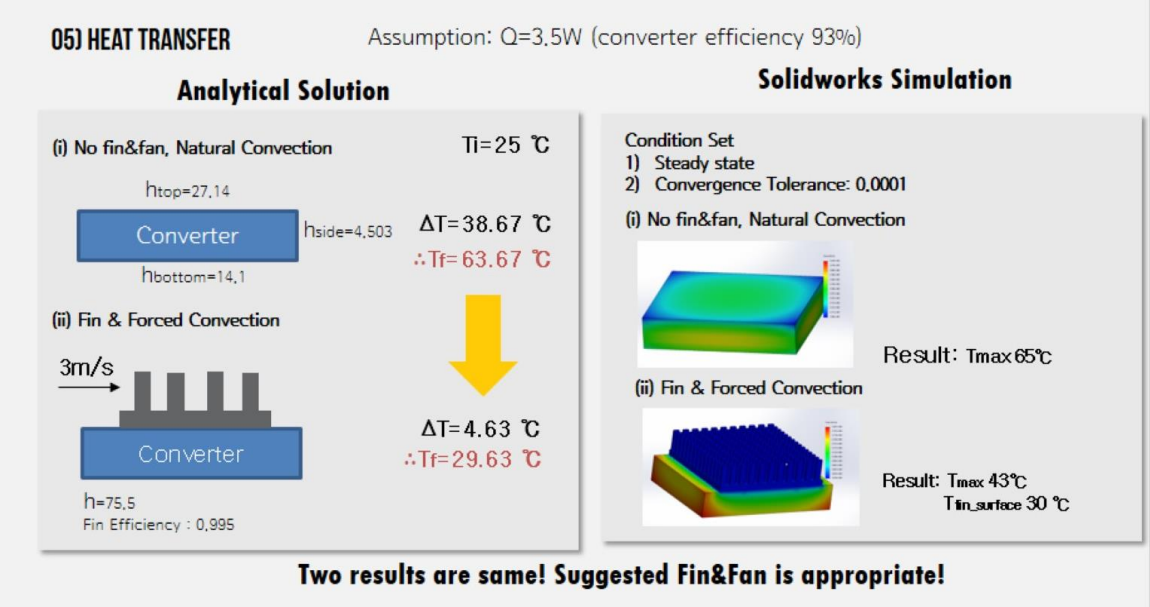
(4) 롤러와 가림막의 형태



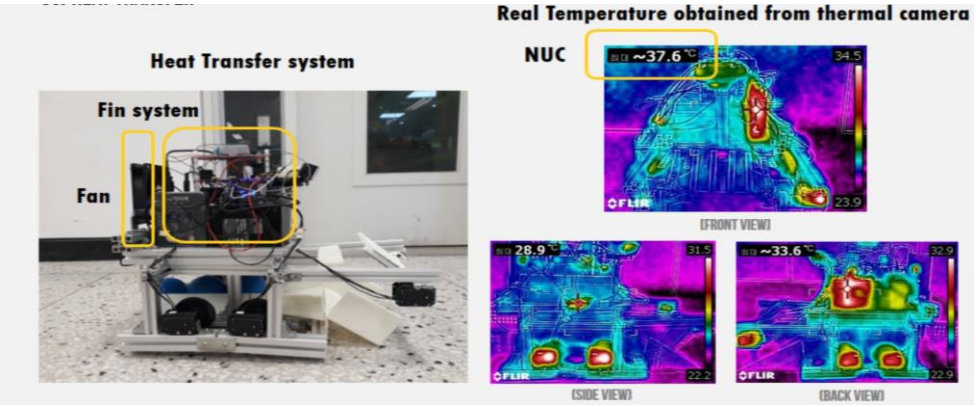
가림막이 없을 경우, 공이 관성으로 인해서 다시 바깥으로 튀어나갈 가능성이 존재한다. 따라서 가림막을 설치하였다. 또한, 롤러의 경우에도 하나의 평판으로 만들 경우에 무거워지기 때문에 가림막과 롤러가 교차해서 지나갈 수 있도록 rake 형태를 띠도록 설계하였다.

2) Fin and Fan

Mobile Platform 을 작동시켰을 때, 가장 온도가 높게 올라가는 부분이 모터를 구동시키는 데에 필요한 converter 라는 것을 확인할 수 있었다. 따라서 열전달을 효과적으로 할 수 있는 방법을 찾기 시작했다. 처음에 conduction 을 생각하였으나, 이는 디자인이 복잡해질 우려가 있어 convection 을 사용하기로 하였다. 따라서 fin 과 fan 을 같이 사용하여 forced convection 을 하기로 하였다.

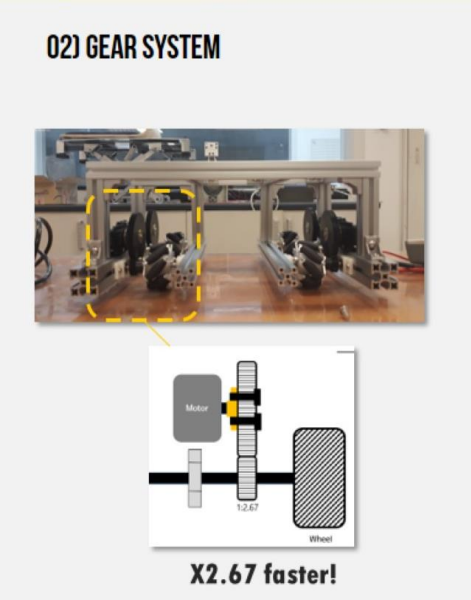


위 그림은 이론적으로 계산한 것과 solidworks simulation 을 통해서 얻어낸 fin 과 fan 을 사용하고 난 후의 온도 분석이다. 위 결과에 따르면 온도를 약 20도 이상 냉각시킬 수 있다.



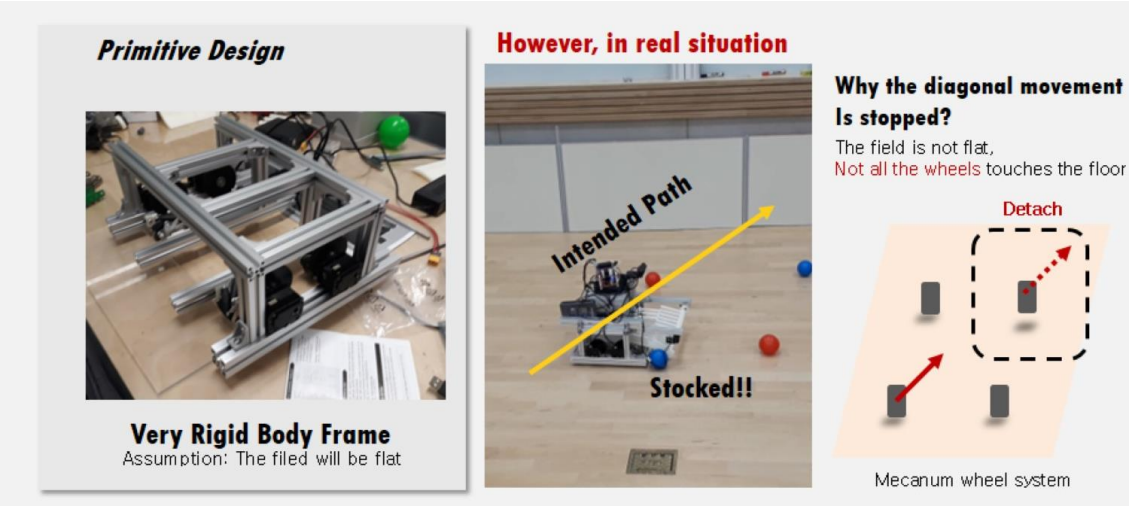
또한, 실제 상황에서도 온도가 확연히 내려 갔음을 확인할 수 있었다.

3) 기어

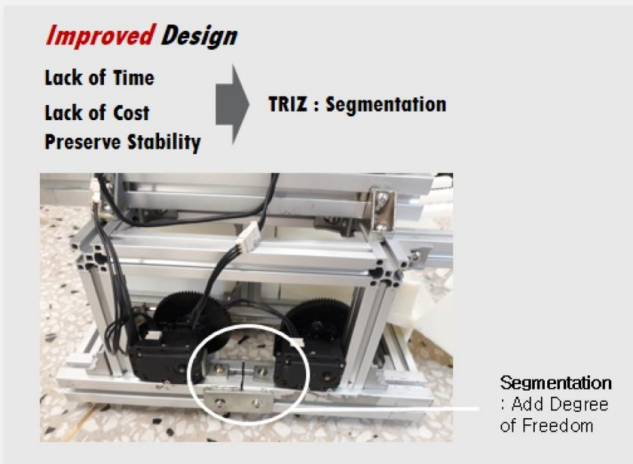


기존의 속력은 약 6cm/s 로 상당히 느렸다. 따라서 시간적인 부분에서 이득을 보기 위하여 기어를 사용하여 속력을 빠르게 하였다. 작은 기어의 날이 30개, 큰 기어의 날이 80개여서 2.67 배 빠른 속력을 낼 수 있었다. 또한, 바퀴를 두 축으로 고정하여 진동으로 인해 생기는 문제를 줄였다.

4) 차체 (4점 지지)

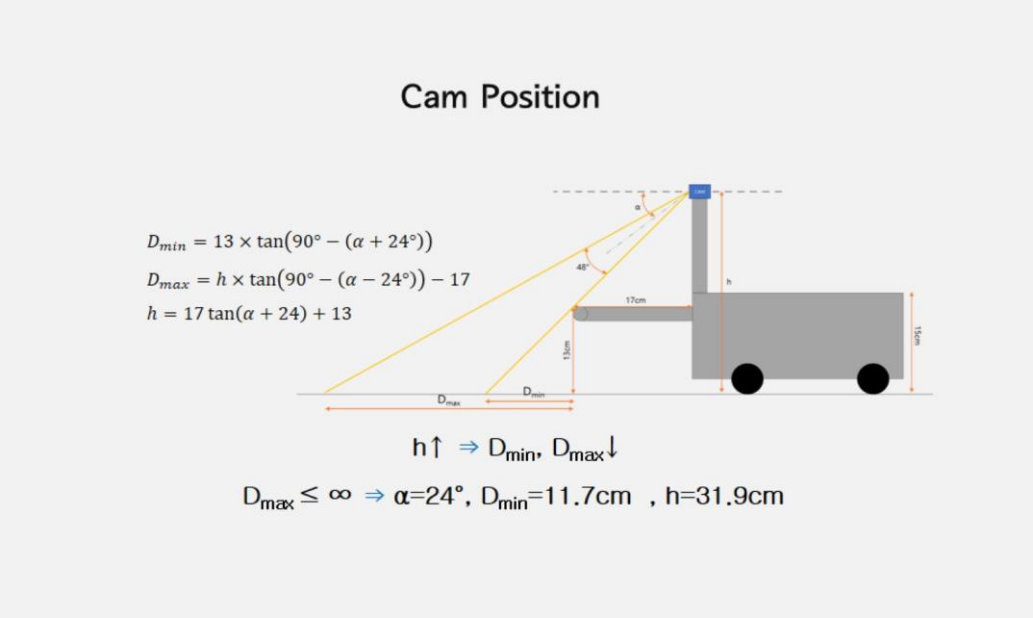


공동강의실의 경우, 바닥이 창시구실과는 다르게 평평하지 않았다. 바닥이 올록볼록하여 4개의 바퀴가 모두 바닥에 닿지 않아 대각선이나 좌우 방향의 이동이 어렵다는 문제점이 생겨, 프로파일을 나누어 자유도를 높였고 대각선 방향 속력을 올렸다.



위 그림과 같이 TRIZ 를 이용해 얻어낸 segmentation 이라는 결론이 나왔다. 따라서 프로파일을 잘라서 자유도를 주어서 문제를 해결하였다.

5) 캠의 위치와 각도

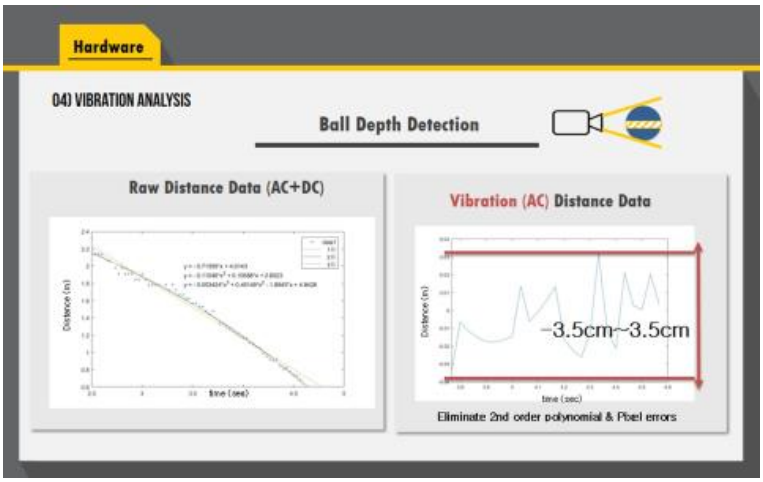


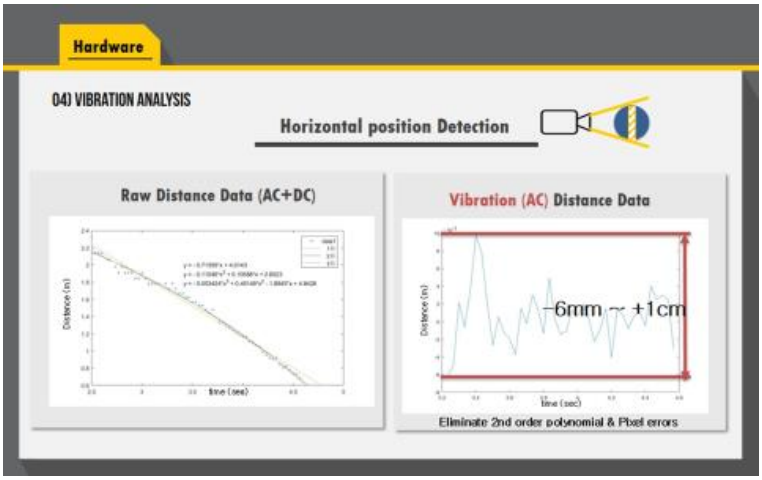
최대한 가까운 것부터 먼 곳까지 볼 수 있는 각도를 지정하였다. 대각선 시야각(FOV)이 76°라는 사실을 이용해 가로(66°), 세로(48°)의 시야 각을 추정해 계산했다.

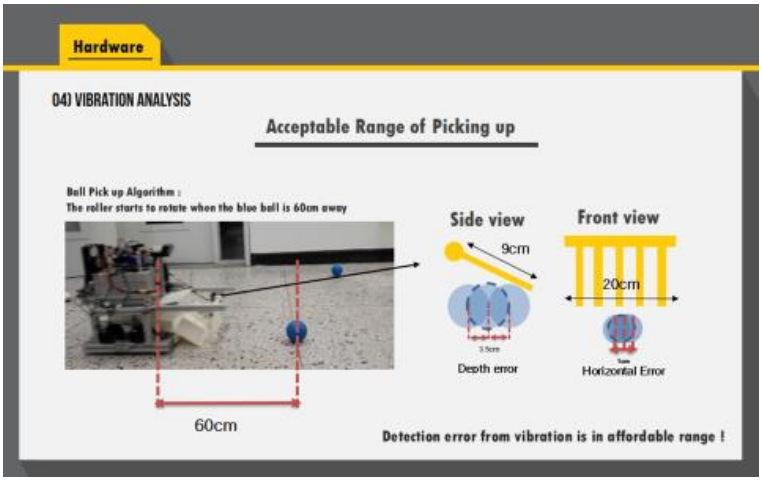
6) 진동

진동의 경우, 공을 고정시켜 놓고 mobile platform 을 직진시키거나 대각선으로 이동시키면서 차체의 진동이 생기더라도 ball detection 능력에는 문제가 없는지 판단하는 과정을 거쳤다. 조금의 진동이 발생하기는 하지만 detecting 에는 문제가 없다고 결론을 내렸기 때문에 진동을 줄이기 위한 damper 등의 사용은

하지 않았다.



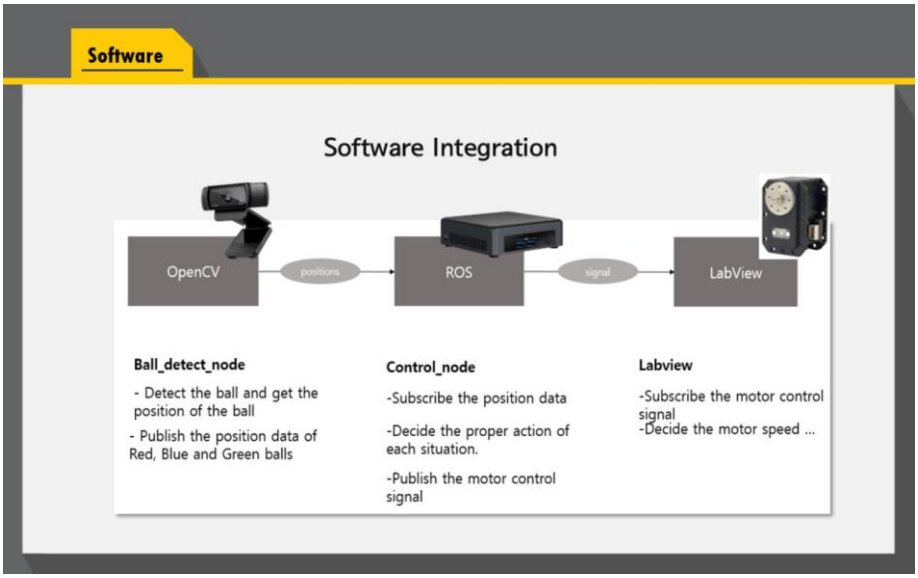




위의 그림을 볼 때, ball detection 이 가능한 허용 범위 내에서만 진동이 일어나기 때문에 진동을 상쇄시키기 위한 추가 디자인의 필요성이 없음을 확인할 수 있다.

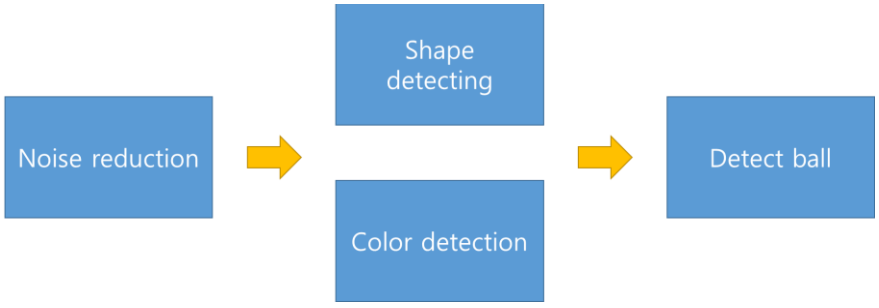
3. Software

1) Software Integration



각 부분에서 수행하는 작업은 위와 같다. Open CV 를 통해서 ball 의 위치와 색에 대한 정보를 얻어 ROS 에서는 연산을 하여 수행할 행동에 대한 데이터를 LabVIEW 로 넘겨주면 LabVIEW 는 이 데이터를 바탕으로 dynamixel 을 가동시켜 mobile platform 이 주어진 과제를 수행할 수 있도록 한다.

2) Open CV



Open CV 의 주요 과제는 ball 을 detect 하여 위치와 색을 결정하여 ROS 로 전달하는 것이다. 위는 ball detection 의 과정이다.

첫 번째로, Ball 을 뚜렷하게 볼 수 있도록 하기 위하여 noise 를 제거하는 작업을 먼저 수행해야한다. Blur 효과나 contrast 를 높이는 방법을 이용하여 noise 자체를 제거하거나 물체의 경계면을 확실하게 한다. 이러면서 너무 작은 픽셀을 차지하는 부분이나 갑자기 확 튀는 부분을 제거하는데, 너무 멀리 있는 공의 경우에는 noise 로 인식하여 제거해버릴 수도 있다는 문제점이 존재한다. Demo 장소가 좁았기 때문에 문제

가 없었지만, 만약 Demo 장소가 기존처럼 9m 의 넓은 공간이었다면 문제가 생겼을 수도 있다.

다음으로는, 공의 모양과 공의 색을 인식하는 단계를 거쳐야한다. 공의 모양을 인식하는 데에는 Canny edge detector 를 사용하였고, 공의 색을 인식하는 데에는 Thresholding operation 을 이용하였다.

위 과정들을 통해서 ball 을 detect 할 수 있다. 그리고 ball 의 위치는 ball 의 크기를 설정하여 ball 이 차지하는 픽셀의 수를 이용하여 ball 까지의 좌표와 거리를 알 수 있도록 하였다.

Open CV 에서는 아주 큰 문제점이 있었는데, 조명에 아주 민감하다는 것이었다. 공동강의실과 창시구실의 조명이 달랐기 때문에 공동강의실에서는 basket 을 빨간 공으로 인식하지 않는 반면, 창시구실에서는 빨간 공으로 인식하여 demo 환경과는 살짝 다르게 상황을 설정해놓고 연습을 하였다.

3) LabVIEW

LabVIEW 에서는 ROS 에서 결정된 행동에 대한 데이터를 바탕으로 dynamixel 을 가동시키는 역할을 하였다. 여기서 controller 는 myRIO 이며, 와이파이를 통해서 NUC 와 통신을 하였다. 즉, 실제로 LabVIEW 가 설치된 컴퓨터에서 코드 실행하는 버튼을 누르더라도 실제로 그 코드를 실행하는 역할은 myRIO 가 담당한다.

Motor 를 컨트롤 하기 위해서는 첫 번째로 motor 의 ID 를 설정해야한다. 이 때, motor ID 를 설정할 때, motor 하나만 연결하고 설정해야한다는 주의점이 있다. 두 번째로, NI 로부터 배부 받은 motor 를 돌릴 수 있는 코드를 이용하여 모터가 작동하는지를 확인해야한다. NI 로부터 배부 받은 코드로는 AX 시리즈와 MX 시리즈의 dynamixel 만 구동시킬 수 있었기 때문에 처음에 샀던 XL 시리즈는 사용할 수 없어 모터를 새로 사는 일이 벌어지기도 했다.

LabVIEW 팀에서는 치명적인 문제가 있었는데, myRIO 의 통신이 생각보다 심각했다. myRIO 와 노트북이 잘 연결이 되지 않는 문제가 벌어지기도 했고, 만약 연결되더라도 통신의 속도가 약 200ms 이상으로 아주 느려서 딜레이가 생겨 mobile platform 의 작업이 늦게 수행되기도 하여 공을 주워야 되는 환경에서도 줍지 못하는 일이 벌어지기도 하였다. 이는 myRIO 기기 자체의 문제이기 때문에 해결할 수 있는 방법은 없지만, ROS 에서 코드를 간략히 하여 데이터 처리를 최대한 간소화 시켜서 조금이라도 줄여보았다. 통신이 느려지는 원인으로는 창시구실에서 여러 팀들이 같이 myRIO 를 쓰고, 그러다보니 다들 아이피와 통신되는 주파수가 같아서 혼선이 생기는 것으로 생각된다.

4) ROS

ROS 에서는 Open CV 에서 받은 공의 위치와 색에 대한 데이터를 받아 적절한 행동을 결정하여 LabVIEW 로 전달하는 역할을 한다. 즉, 사람의 뇌와 같은 역할을 수행한다고 생각할 수 있다.

파란공 데이터가 들어올 때까지 계속 mobile platform 을 회전시키는 명령을 내리다가 일정 범위 내로 들어오면 직진하는 명령을 내린다. 이 때, 회전이 너무 빠르면 공을 인식하지 못하는 경우가 발생할 수 있다. Open CV 에서 영상이 5Hz 로 찍히기 때문에 공을 영상에 담지 못하고 지날 수 있는 상황이 발생할 수 있어 적절한 수치를 정하는 것이 중요하다.

Mobile platform 이 직진하여 공이 일정 기준치의 거리에 들어오면 롤러를 회전시키는 명령을 내린다. 다만 딜레이로 인해서 롤러가 기준치보다 더 짧은 거리에서 작동되기 시작한다는 문제가 존재한다. 따라서 롤러 회전 명령을 내리는 적절한 기준을 여러 번의 실험을 통해서 알아내야한다. 우리 팀은 60cm 이내에 들어오면 롤러를 가동시키도록 결정하였다.

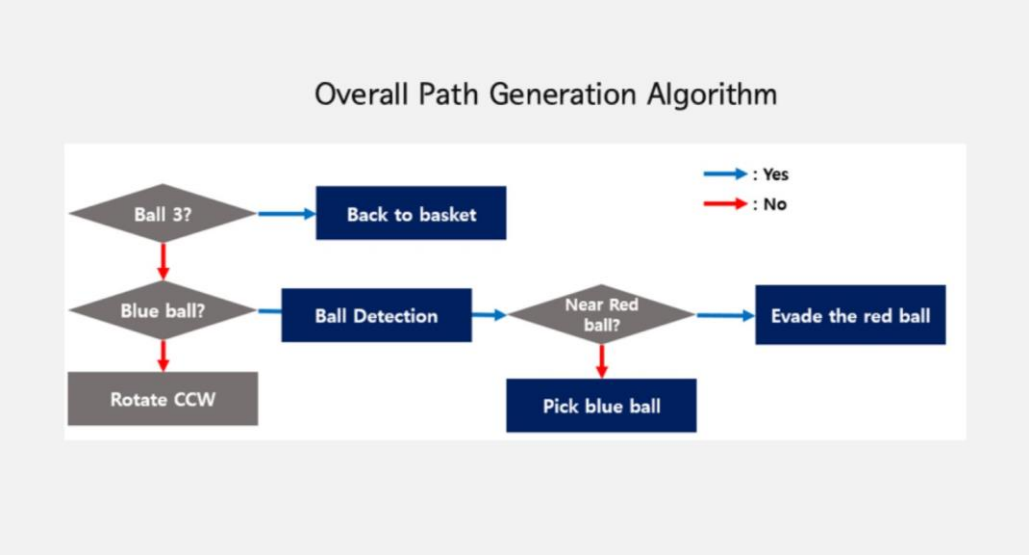
롤러의 회전에 대해서도 살짝 이야기하면, 공이 들어왔는지 판단하기 어렵기 때문에 롤러가 공을 먹을 수 있는 충분한 시간이라고 판단되는 1.5초에서 2.25초까지를 바꾸어 가면서 롤러의 회전 시간을 결정하였다. 우리 팀의 Mobile Platform은 롤러가 회전하면서 직진하는 형태이기 때문에 만약 롤러의 회전 시간이 너무 길지 않을 경우 벽에 들이받을 수 있고, 짧을 경우 공을 먹지 못하는 문제가 생길 수 있었다. 따라서 이 또한 실험을 통해 적절한 수치를 결정하였다. 그리고 공을 3개를 수집했는지, 롤러가 회전되는 코드가 작동되는 횟수를 통해서 판단하기로 하였다. 롤러가 3번 돌아가면 공을 모두 수집했다고 판단하여 basket 으로 돌아가는 것이다.

빨간공이 있는 경우에는 대각선으로 피하는 명령을 내리도록 하였다. 이는 빨간공이 특정 범위 내에서 벗어날 때까지만 차체가 이동하면 되었기 때문에 딱히 어려운 문제는 존재하지 않았다.

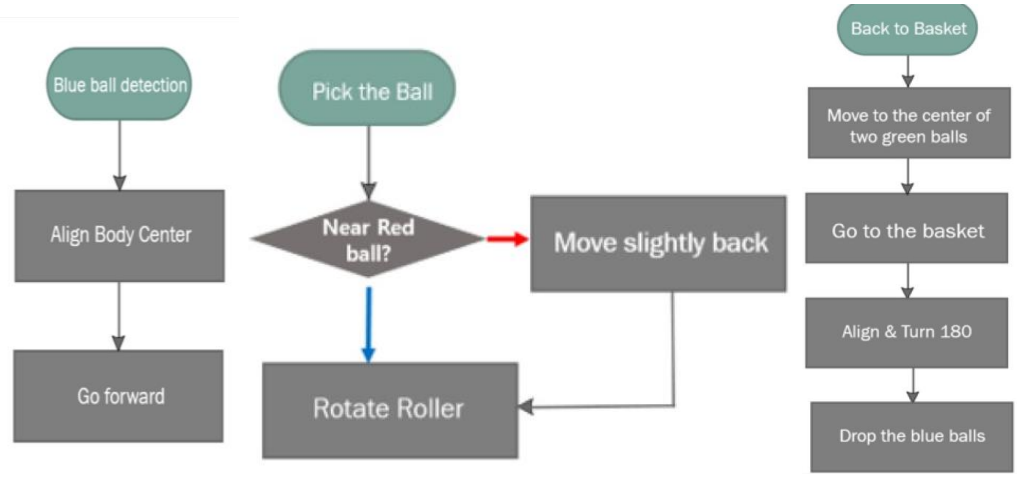
마지막으로 basket 을 향해서 돌아가는 코드에서 문제가 많았다. 초록공 두 개의 수직이등분선에 차체가 올라간 후, 두 공의 좌표 사이를 바라보고 직진한 후 180도 돌아서 공을 내려놓는 방식을 선택하였다. 처음에 공동강의실에서 대각선 방향으로의 진행이 어려운 것을 확인하여 수직이등분선에 올라가는 명령을 회전과 직진으로만 수행하여야 해서 시간이 너무 오래 걸렸지만 hardware 에서 자유도를 주면서 대각선 방향으로의 진행이 가능하게 하여 align 시간을 조금 줄일 수 있었다. 이 작업을 수행할 때 수직이등분선에 가까워질수록 motor 의 회전속도를 아주 느리게 주었기 때문에 LabVIEW 에서 설정했던 일정 기준치보다 낮은 값이 들어가는 문제가 있었다. 이

문제는 어쩔 수 없이 LabVIEW 에서 기준치를 없애면서 해결되었다. 정확한 연산을 통해서 모터의 운동을 결정하였기 때문에 기준치를 없애도 문제가 생기지는 않았다.

5) Overall Algorithm



전체적인 알고리즘은 위 그림과 같다. 파란공을 인식할 때까지 회전하고, 발견하면 가까지 접근하여 공을 줍고, 롤러가 3번 가동되면 basket 으로 초록공을 보고 돌아간다. 이 때, 공을 줍는 과정에서 빨간공이 나타나면 대각선 방향으로 피한다.



4. 후기

한 학기 창시구를 수강하면서 여러가지 문제점이 많았다. 첫 번째로, 우리 조의 경우는 컨버터가 고장이 났다. 컨버터의 스펙이 약한 것을 좋지 않은 것을 사용하면서 모터를 돌리다보니 컨버터가 무리하여 발표 직전에 하나가 고장이 났다. 그래서 다른 컨버터를 사용하였다. 컨버터는 가변 컨버터를 사용하여 필요에 따라 전압과 전력을 조절할 수 있는 것을 사용하는게 좋았을 것이다.

두 번째로, dynamixel 이 생각보다 복잡하다는 것을 알 수 있었다. Dynamixel 의 경우, 각각의 dynamixel 을 병렬으로 연결할 경우, 한 dynamixel 이 데이터 선과 전기 선이 반대로 연결되지만 하더라도 모든 dynamixel 이 작동하지 않는 것을 확인할 수 있었다. 또한, 맞는 방향으로 연결하더라도 단선된 부분이 있어 데이터가 dynamixel 으로 들어가지 않는다면 모든 모터가 작동하지 않는 현상이 발생하였다. 그리고 dynamixel 에 과한 전류가 흐르게 되면 전원은 들어오지만 모터를 인식하지 못하게 내부가 망가지는 경향도 있는 등 dynamixel 을 다룰 때에는 주의할 점이 상당히 많았다.

세 번째로, 위에서도 언급하였던 myRIO 의 통신 문제가 있다. 아마 여러 조들이 다들 좁은 공간에서 myRIO 를 쓰고, 그 myRIO 들이 같은 주파수를 사용하기 때문에 통신에 혼선이 왔을 것이라고 예상된다. 창시구실을 제외한 곳에서도 demo 연습을 할 수 있는, LabVIEW 가 설치되어 있는 컴퓨터가 있는 연습실이 따로 존재하였으면 한다.

네 번째로, 최종 demo 를 수행하는 장소의 공지가 늦었음으로 인해 생긴 문제점이 있었다. 처음에는 기계동 중앙현관에서 demo 를 수행한다고 하였고, 이 때는 바닥이 평평하였기 때문에 단지 공이 미끄러질 우려만 생각하였으나, 공동강의실으로 장소가 바뀌면서 평평하지 않는 바닥 때문에 suspension 을 설계하여야 할지, 아니면 다른 방법으로 해결하여야 할지 등의 논의가 계속해서 오갔다. 또한 처음에 9mX2m 의 넓은 공간이라고 하여 시간적인 면에서 이득을 보기 위해서 기어를 설계하였으나 장소가 좁아짐에 따라서 시간적인 면에서 볼 수 있는 이득은 적어지고 오히려 딜레이가 생긴다면 더 큰 부정확도만 생길 수 있는 가능성이 생기기도 하였다.

또, demo 를 연습할 수 있는 기간이 조금 더 길었으면 하는 아쉬움이 남는다. Mobile Platform 을 빠르게 완성하였더라도 공동강의실과 창시구실의 환경이 다르기 때문에 창시구실에서 잘 되던 기능들이 공동강의실에서는 잘 되지 않을 수도 있었다. 공동강의실의 환경에 맞도록 수정할 수 있게 조금 더 빠르게 공동강의실을 개방하였으면 demo 환경에 더 알맞은 상태로 수정을 할 수 있지 않았을까 하는 아쉬움이 있다.