# ME400

# 창의적 시스템 구현 1

# Final Report

**Group IKOH**

**20140156 김태홍**

## Index

## 1. Understanding the Project

첫 OT 때 자신이 원하는 파트를 작성하여 제출하면 조교님들께서 파트가 겹치지 않도록 조를 배정해 주신다. 조 배정부터 1차 발표, 2차 발표, 3차 발표까지 각각의 기간에 전체 프로젝트 중 어떤 부분을 다뤘는지 설명하겠다.

### A. Design Review 1

조 배정이 완료되고 Solidworks, LabVIEW, ROS, OpenCV로 자신의 파트를 정하게 된다. 각 파트에 대한 1차 발표까지는 미션을 이해하고, 미션의 목표를 파악하는 것이 가장 중요하다. 5mx3m의 세트에 blue ball 3개와 red ball 3개가 놓여져 있고 바구니 앞에서 출발해서 blue ball 3개를 집어서 바구니에 담는 것이 미션이다. 이 때 blue ball 하나당 10점, red ball 하나당 -5점을 얻게 된다. 바구니 입구의 양쪽에 배치된 green ball을 인식해서 돌아와야 한다. 미션을 완수하는 데 걸린 시간과 미션이 끝난 직후 플랫폼의 온도에 따른 점수도 있다. 또한 미션 이전에 진행되는 발표에서는 Heat transfer, Pick-up system, Motor control, Vibration control, Vision & ROS integration에 대한 설계가 잘 진행되었는지 평가하게 된다.

이렇게 미션이 정해져 있을 때, 네 파트 각각에 어떤 사람이 적합할 것인지 개인적인 의견을 적어보면 우선 Solidworks는 3월에 여유로운 사람이면 좋다. Solidworks로 진행하는 것은 첫 설계가 거의 대부분이기 때문이다. 이후에는 Solidworks software보다는 직접 플랫폼을 제작하고 수정하게 되는데, 기계 부품에 대해 잘 알고 있으면 좋다. 실제로 우리 그룹에서는 LabVIEW 담담 조원이 중간부터 hardware 제작에 전념했다.

우리 그룹이 특수한 경우일 수도 있지만 LabVIEW를 담당한 두 학생 모두 중간 이후로 각각 hardware와 software로 합류했기 때문에 LabVIEW에 적합한 사람이 어떤 사람일지는 모르겠다. LabVIEW에서 필요한 것만 한다면 가장 배우기 쉽고 간단하다고 생각한다.

내가 담당한 ROS의 경우 프로그래밍 경험이 많고, 미션을 위한 알고리즘에 관심이 많은 사람이 적합하다. Linux Ubuntu라는 익숙하지 않은 OS 환경에서 작업해야 하고, 작업 내용이 시스템의 Path planning algorithm을 구상하고 이를 C++을 이용해 코드를 작성해야 한다. ROS가 LabVIEW와 OpenCV를 이어주는 역할을 하기 때문에 다른 파트 조원과 자주 컨택해야 한다.

OpenCV 역시 프로그래밍 경험이 많으면 좋다. 우리 그룹에서는 조교님들께서 배포해 주신 코드를 수정해서 사용했고, HSV value를 조정해서 좀 더 잘 인식할 수 있게 하는 정도에서 그쳤다. OpenCV를 적극 활용한 그룹에서는 보다 정확한 detecting을 통해 정확하게

ball의 위치 정보를 제공했다고 한다.

이렇게 파트 배정이 되면 각자 파트에서 전체 미션 수행을 위해 어떤 기능을 수행할 수 있는지 명확히 목표를 설정하고 2차 발표까지 완수해내는 것이 중요하다. 3월 한 달 간은 대부분 연습반을 통해 각 파트의 기본적인 것과 미션 수행을 위한 내용을 배우게 되는데 이 내용을 바탕으로 어떤 기능까지 수행할 수 있는지, 다른 파트에서 어느정도까지 요구하는지 등을 적극적으로 공유하는 것이 중요하다.

B. Design Review 2

1차 발표까지 정한 아이디어를 바탕으로 실제 시스템 제작을 시작한다. Solidworks의 경우 1차 발표의 피드백을 바탕으로 보다 현실적인 디자인을 설계하고, 주어진 재료나 3D 프린터를 활용하여 차체를 제작한다.

LabVIEW, ROS, OpenCV에서는 각 파트 연습반에서 배운 내용을 바탕으로 서로 정보를 주고 받는 통신을 시도한다. 우리 그룹의 경우 2차 발표까지는 blue ball과 red ball을 하나씩 놓고 red ball을 피해 blue ball을 pick-up 하는 것까지를 목표로 설정했다.

LabVIEW와의 통신은 ROS에서 motor input을 LabVIEW에서 받을 수 있는 형태(24개의 float)로 정리해서 보내는 것이다. LabVIEW에서는 이를 받아서 원하는대로 motor를 구동시키는 것이다. Xbox Controller를 활용하면 통신의 어느 부분이 잘 못 됐는지 확인할 수 있다. 조교님들께서 수업해주신 내용만으로는 이를 쉽게 해내기 힘드니 수시로 조교님들께 질문을 드리는 것이 좋다.

ROS와 OpenCV를 연결하는 것은 두 파트 각각에 주어진 코드를 통합하는 것이 끝이다. 하지만 코드의 전체적인 구조에 대해서 정확히 이해하지 못한 상황이고, 이해하기도 힘들기 때문에 여러 번의 실패를 겪을 것이다. 코드를 한줄한줄 분석하며 어느부분을 어디에 넣을지 고민을 많이 하면 둘을 합친 코드를 얻을 수 있을 것이다. 이 부분에 대해서는 조교님께 많은 질문이 쏟아졌지만 이를 해결하는 과정에서 배울 것이 많다고 하시며 가르쳐주시지 않으셨다.

각각의 파트끼리 통합이 완료되고, hardware가 구동 가능한 상태가 되면 목표로 설정했던대로 red ball 하나를 피해서 blue ball을 pick-up 하는 정도의 간단한 작동은 해낼 수 있을 것이다.

C. Design Review 3

2차 발표까지 OpenCV를 통해 바닥에 놓여진 ball을 detect해서 ball의 좌표와 색상을 ROS로 전송하고, 이를 바탕으로 모터에 명령을 내려 차체가 움직이는 것을 확인했다. 이제 구상한 알고리즘을 바탕으로 Path Planning을 완료하면 된다.

우리 그룹의 경우 2차 발표 때 이미 Path planning에 대한 계획이 완료되었고, 발표에도 포함시켰었다. RPLidar를 이용해서 Navigation처럼 자신의 위치를 파악할 수 있기는 하지만, 이는 가을학기에 배울 SLAM과 관련된 내용이라고 하셔서 빠르게 포기하고 Webcam으로만 미션을 수행할 수 있도록 algorithm을 구상했다. Algorithm에 구상에 대한 설명은 다음 장에서 설명하겠다.

Algorithm대로 복도에서 간이로 미션을 수행하다보면 예상치 못한 케이스가 발생하게 되고, input에 따른 output도 예상과 다른 경우가 많이 발생하기 때문에 한 달 내내 튜닝만 한다고 보면 된다. 이 때 hardware의 변경사항도 있기 때문에 최대한 빨리 최종 시스템을 구축하고 튜닝만 해야 할 것이다.

그리고 2차 발표까지 생각만 했던 heat transfer와 vibration control의 경우 지도교수님들의 분야를 고려하여 직접 연락을 드리면 연구실 장비를 활용하여 온도 측정이나 진동 측정 등을 할 수 있다. 이 자료를 바탕으로 시스템에 적합한 장치를 설치하면 된다.

모든 발표 통틀어서 가장 중요한 것은 교수님께서 공지해주신 기준을 참고하여 각 기준에서 요구하는 내용을 모두 포함시켜야 한다는 것이다. 예를 들어 Vibration Control이 필요없다고 생각이 된다면, 필요없는 합당한 이유를 찾아 설명해야 할 것이다.

## 2. ROS

네 파트 중 내가 맡은 파트는 ROS 파트였다. Solidworks와 LabVIEW는 기계기초실습과 시스템 모델링 및 제어 수업을 들으며 어떤 프로그램인지 어느 정도 파악했지만, ROS와 OpenCV는 전혀 모르는 프로그램이었다. 개인적인 생각으로 OpenCV보다는 ROS가 기계과에서 더 유용할 것이라고 생각해서 ROS를 담당하게 되었는데, 딱히 차이는 없는 것 같다.

ROS는 기존에 사용하던 Windows가 아닌 리눅스 Ubuntu 환경에서 사용하는 프로그램이기 때문에 시작부터 익숙하지 않음에 어려움을 겪었다. 또한 ROS에 익숙해진 이후에는 본격적인 프로그래밍을 해야 하는데, 이는 C++을 기반으로 작업해야 하기 때문에 기본적인 프로그래밍도 알아야한다. 이렇듯 기계과 4학년이 한 학기 동안 해내기에는 낯선 일들이 많기 때문에 프로젝트를 위해 필요한 지식이 무엇인지 파악하고 효율적으로 해내는 것이 중요하다.

이 장에서는 ROS의 기본과 활용으로 나누어 설명할 것인데, ROS의 내용에 대해 설명할 것이 아니라 어떤 것을 어떻게 배우고 어디에 활용하는 지에 대해서만 설명할 것이다. 실제 ROS 내용은 주어지는 수업자료나 ROS 책 그리고 조교님들의 연습반에서 충분히 잘 배울 수 있다.

## A. ROS의 기본

ROS를 배우는 것이 가장 힘든 첫 번째 이유는 Windows가 아닌 Linux Ubuntu를 사용하는 것에 있다. Linux Ubuntu와 ROS를 설치하는 데만 최대 1주 정도 소요되니 첫 번째 과제가 오래 걸리지 않는다 해서 직전에 설치하지는 않아야 할 것이다. 우리 그룹의 경우 NUC이 두 번이나 다운되는 바람에 최종 발표 전날 밤을 비롯하여 총 세 번의 설치를 경험했다. 한 번 깔고 말 것이라는 생각말고 제대로 배우면서 설치하는 것이 좋다.

ROS 설치가 완료되면 ROS의 가장 기본으로 node를 만들고 node 사이에 message를 주고 받는 것을 배운다. 우리 그룹의 경우 새로운 node를 생성하지 않고 기존에 제공된 node를 수정하여 사용했다. 하지만 node를 만들고 node의 구성을 정확히 알아야 이렇게 수정해서 사용하는 것도 가능하다. 첫 번째 과제를 하며 정확히 익히고, 연습반을 통해 한 번 더 복습한다면 충분할 것이다.

다음으로 갑작스러울 수 있지만 바로 Webcam, RPLidar, Xbox Controller를 이용하여 data를 받는 것을 배운다. 이 때 조교님들께서 필요한 node를 전부 제공해 주시는데 우리 미션에 바로 적용하기에는 부족한 상태이니 상황에 맞게 적절히 수정해서 사용해야 한다. 특히 ball_detection_node.cpp 의 경우 OpenCV에 제공되는 main.cpp와 결합하여 완전한 node로 만들어지는데, OpenCV 담당 조원과 함께 한줄한줄 코드를 분석하며 결합해야 한다. RPLidar의 경우 RViz라는 시각화 프로그램으로 데이터를 받아오는 것을 확인하는 데에서 그쳤고, Xbox Controller를 통해 LabVIEW에 어떤 형태로 input을 제공해줘야 하는지, LabVIEW에서는 어떤 형태로 data를 받아야 하는지 알 수 있다. LabVIEW와의 통신이 안 될 때 Xbox_ctl.cpp 라는 node를 활용하여 어느 쪽에 문제가 있는지 판단할 수 있다.

여기까지가 ROS의 기본이고 이것들이 완성된다면 위에서 말한 2차 발표 목표, 즉 red ball을 피해 blue ball을 pick-up 하는 것까지 성공할 수 있을 것이다.

## B. ROS의 활용

이제 ROS를 활용하여 실제 미션을 수행하는 것에 대해 얘기해보고자 한다.

우선 아래의 psudo code를 보면 우리 그룹의 전체적인 Path palnning algorithm을 알 수 있다. 간단히 말하면 가장 오른쪽의 blue ball 부터 pick-up하는 것이다. Algorithm을 구상할

```
void path_planning(i){   //i=number of blue ball in field

        if i=0{ // after checking that there's no blue ball in field
                scan green balls while turn CW
                after scan 2 green balls, go to center of 2 balls
                turn 180 degree
                go back for 5 seconds
                end
        }

        elif i=1{ // after picking twe blue balls
                scan blue balls while turn CW
                after scan a blue ball, turn toward the blue ball
                approach to the blue ball with avoid red balls
                go straight with operating the propeller(?)
                scan blue balls while turning 360 degree
                path_planning(i)
        }

        elif i=2{ //after picking one blue ball
                scan blue balls while turn CW
                after scan a blue ball, turn toward the blue ball
                approach to the blue ball with avoid red balls
                go straight for 3 seconds with operating the propeller
                path_planning(1)
        }

        else{ //at first
                scan blue balls and choose the rightest one
                turn toward the blue ball
                approach to the blue ball with avoid red balls
                go straight for 3 seconds with operating the propeller
                path_planning(2)
        }
}
```

때, 흔히 shortest path를 찾을 때 사용하는 가까운 blue ball부터 pick-up하는 algorithm을 생각했다. 하지만 매번 그 자리에서 360 °를 회전하여 가장 가까운 blue ball을 찾는데 오래걸릴 것이고, ball 들의 위치정보를 기억해서 처리하는 것도 쉽지 않을 것이라고 생각했다. 그리고 기하학적으로 계산해 봐도 path의 길이 차이가 크지 않아 오른쪽부터 pick-up하는 algorithm을 채택했다.

오른쪽부터 pick-up 하기로 한 뒤에 red ball이 경로에 있으면 red ball이 시야의 왼쪽에 있으면 오른쪽으로, 오른쪽에 있으면 왼쪽으로 피하도록 했다. 또 가장 오른쪽에 있는 blue ball이 아닌 blue ball이 경로에 있는 경우 red ball처럼 피하기로 했다. 이는 이 플랫폼이 지능적이지 않기 때문에 일관성 있는 algorithm이어야 했기 때문이다. Blue ball을 pick-up 할 때마다 count를 했고, 3이 되면 360 ° 회전해서 실수해서 남아있는 공이 있는지 확인한 뒤에 green ball을 detect해서 바구니에 가까이 가게 했다. 마지막으로 green ball 두 개의 위치정보를 통해 두 green ball의 중앙에 오게 했고 정확히 180 °를 회전한 뒤 후진하도록 했다.

이렇게 미션을 수행하면서 다른 그룹에서는 어떻게 했는지 모르겠지만 우리 그룹이 미션을 수행하기 위해 사용한 나름 특별한 방법이 하나 있다. 그것은 어떤 신호를 일정 시간동안 또는 일정횟수만큼 지속적으로 보내는 것이다. ROS에서 LabVIEW로 message를 보낼 때 time

duration과 관련된 함수가 있는데, 이를 이용하면 webcam으로 촬영하는 지금 당장의 정보를 배제하고 원하는 움직임을 수행할 수 있다. 이를 위해서는 wheel kinematics라는 것도 필수적인데, 바퀴의 지름, 차체 중심과 바퀴 사이의 거리를 이용하여 input에 따른 output을 정확히 계산하는 것이다. 이를 이용하면 신호 한 번에 몇 도를 회전할 지 계산할 수 있다. Time duration과 결합한 결과 정확히 180 °를 회전할 수 있었던 것이다. 이 time duration을 찾아서 정확히 적용하는 데에 2주 정도 걸렸던 것 같다. 좀 더 빨리 알았다면 더 빨리 적용했을 것이다.

아쉬운 점도 있다. OpenCV를 이용하면 ball의 중심 좌표와 size를 이용하여 x, y, z 좌표를 얻을 수 있는데, 이를 이용해 우리 시스템에 맞게 distance를 제대로 정의하지 못했다는 것이다. 우선 ROS 팀에서 OpenCV 팀에 제대로 요구를 하지 않아 OpenCV 팀에서는 어느정도에서 만족하고 다른 역할로 옮긴 것이 발단이었다. 때문에 z좌표에 대한 신뢰도가 부족했다. 다음으로는 초기에 distance를 $\mathrm{d} = \sqrt{x^2 + z^2}$ 라고 정의를 했는데 이를 고쳐서 정확한 distance를 측정하고 그에 따라 새롭게 움직임을 변경하기에 시간이 촉박했다.

이렇게 많은 고민을 하고 작성한 코드 임에도 모의 미션 수행을 진행할 때마다 각종 오류가 발생했다. 완벽한 algorithm을 위해 밤을 새워가며 준비했지만, 실제 미션은 훨씬 쉽게 출제되었다. 물론 카메라 정렬, 통신 문제 등으로 첫번째 시도에서 마지막에 바구니에 공을 내려놓지 못했지만 우리가 기울인 대부분의 노력이 red ball을 피하고 blue ball을 pick-up하는 것이었음을 생각하면 어느정도 허탈함도 있다.

ROS의 처음부터 끝까지 돌아보면 아직도 답답하고 막막하다. 처음 배울 때부터 목표가 무엇인지 인지하고, 미션을 수행하기 위해 필요한 것이 무엇인지 생각하면서 차근차근 공부하면 잘 할 수 있을 것이다. 무엇보다도 조교님들께서 정말 잘 도와주시니 적극적으로 물어보고 배우는 것이 좋다.

## 3. Progress Report

1st week (3/19 ~ 3/25)

3/19 ROS TA Session

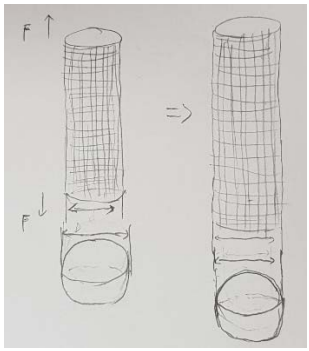
- Ubuntu와 ROS의 컨셉에 대한 설명을 들음

- Assignment로 주어진 publisher와 subscriber 노드 작성 및 실행 / service server와 client 노드 작성 및 실행 에 대해 설명을 들음

3/20 Group Meeting

- Group의 Concepts에 대해 각자 Idea 준비 및 발표

- Idea에 대한 evaluation 진행 및 결정(아래는 제시한 두 개의 idea 중 하나)

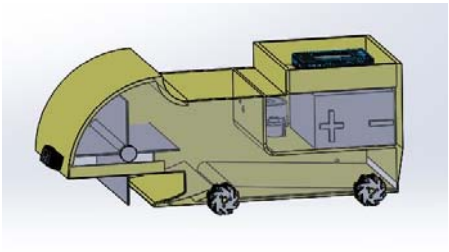



3/23 ROS TA Session / Group Meeting / Assignment

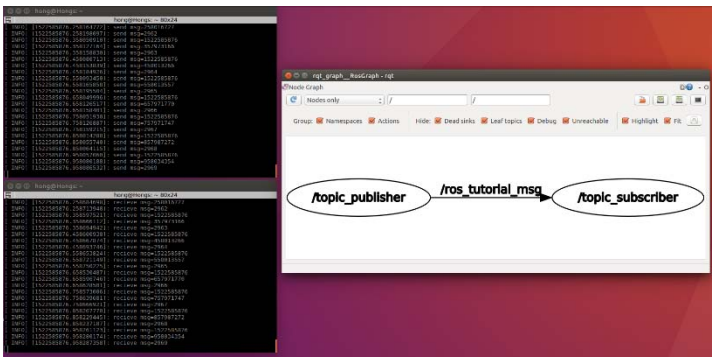
- ROS를 이용한 data integration에 대한 설명을 들음

- RP Lidar, Webcam, Xbox Controller로부터 ROS로 신호를 받는 것에 대한 설명을 들음

- 20일의 idea를 보완하여 concepts를 finalize함(아래는 3/23까지의 design)



- ROS assignment 수행(아래는 1번 과제에 대한 사진)



3/25 Group Meeting

- concepts와 관련된 토의 진행 및 수정
- Design Review #1 발표에 대한 논의
- 각 파트의 발표 내용 논의

2nd week (3/26 ~ 4/1)

3/27 Group Meeting

- 발표자의 모의 발표를 듣고 피드백

3/28 Group Meeting

- 발표자의 모의 발표를 듣고 피드백
- RPLidar를 이용한 주변 장애물 정보 받음
- Webcam을 이용한 ball detection을 진행(size가 변하는 것을 통해 확인)
- RPLidar와 Webcam의 좌표를 transformation을 통해 일치시킴
- Xbox Controller로부터 입력을 받는 것을 확인

3/30 Design Review #1

- 첫 번째 Design Review 참관

3rd Week (4/2~4/8)

4/3 Group Meeting

- 2 차 발표까지의 계획 수립
- RPLidar, Webcam, Xbox 관련 node 점검 및 transformation

- 각 센서로부터 입력 받은 데이터를 Rviz 와 터미널로 확인

4/4 Meeting with OpenCV students

- NUC 에 OpenCV 설치

- OpenCV 팀에 제공된 main.cpp 코드 확인

4/5 Meeting with LabVIEW students

- xbox_ctrl_node 를 이용한 MyRIO 와 통신 시도

  - xbox_ctrl_node.cpp 의 IP, port number 등을 수정한 뒤 서버 연결을 시도 했으나 "fail to connect" 라는 error 가 뜸

- OpenCV 에 제공된 main.cpp 코드 분석

  - 코드의 각 부분이 어떤 역할을 하는지 대략적인 분석 시도

4/8 Meeting with OpenCV students

- ROS에 제공된 webcam_node.cpp, ball_detection_node.cpp와 OpenCV에 제공된 main.cpp 통합 시도

  - 분석한 내용을 바탕으로 OpenCV 요소가 포함된 ball_detection_node가 작동하도록 코드를 수정했으나 webcam 관련 오류 발생

4th Week (4/9~4/15)

4/10 Group Meeting

- 지난 주까지 경과 보고 및 계획 renewal

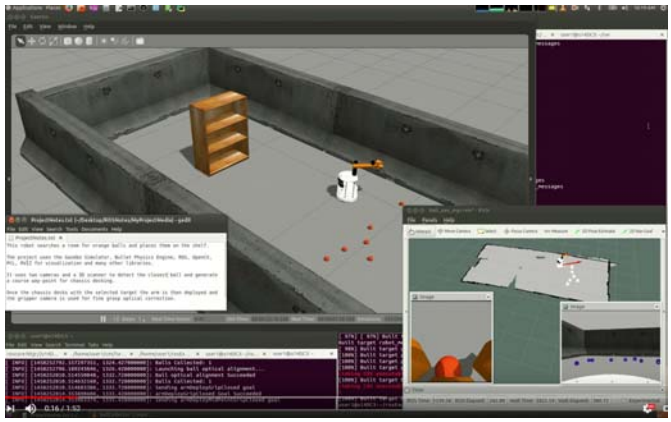
- ROS 의 경우 LabVIEW, OpenCV 와의 연결을 성공하고, Hardware 가 제작되는 동안 Path planning, Picking Algorithm 등을 연구하기로 함.

4/12 Path Algorithm 관련 ROS 프로그램 조사

- SLAM & Navigation 활용 방안 조사

- Gazebo 를 이용한 가상 현실 구현

  - 아래 영상과 같이 final mission 과 같은 조건으로 가상 공간을 구현하고 Platform 의 Hardware 가 제작되는 동안 이를 이용하여 Algorithm 을 구상할 계획



(출처: https://www.youtube.com/watch?v=NFiTfMdTR4w)

4/13 Meeting with LabVIEW(본인은 불참)

-MyRIO 와 통신에 성공했으며 xbox controller 를 이용한 모터 구동이 가능하다고 함.

5th Week (4/16~4/22)

4/16~4/18 Midterm exam

4/18 Path Planning

- 공을 인식한 뒤에 어떤 path 로 움직일 지 algorithm 구상

  - 가장 가까운 ball 부터 picking 하는 algorithm 과 가장 오른쪽(또는 가장 왼쪽)에 위치한 ball 부터 picking 하는 algorithm 을 구상

  - 그 밖에 ball picking 과 관련된 algorithm 및 code 검색

6th Week (4/23~4/29)

4/23 ROS team Meeting

- ball_detection_node.cpp 를 수정

- Path planning algorithm 에 대한 논의

  - 가까운 ball 부터 picking 하는 algorithm 과 오른쪽에 위치한 ball 부터 picking 하는 algorithm 을 group meeting 에서 논의해보기로 함

4/24 Group Meeting

- 각 파트 별 진행 상황 보고 및 2 차 발표까지의 계획 수립

- ROS 의 경우 openCV 와 함께 ball_detection_node.cpp 를 수정하여 ball 이 너무 가까이 있는 경우, 멀리 있는 경우, ball 끼리 겹치는 경우, ball 이 아닌데 ball 로 인식하는 경우 등을 확인

4/27 ROS TA Session

- rqt_graph 로 전체 시스템 구성 발표 및 피드백

- data_integration.cpp code 수정과 관련된 질의응답

  - data_integration.cpp code 에서 webcam 으로부터 data 를 받고 myRio 에 신호를 주는 것까지 포함시키기로 함

4/28 Group Meeting

- 발표 전 각 파트 진행 상황 확인 및 계획 수립

  - LabVIEW 파트에서 wheel kinematics 를 도입하여 원하는 위치로 보다 정확하게 이동할 수 있도록 할 계획

  - Path planning algorithm 에 대한 논의를 통해 오른쪽에 위치한 ball 부터 picking 하는 algorithm 으로 결정

  - 여러가지 error cases 에 대해 논의가 진행되었고 이를 반영하여 code 를 작성할 예정

4/29 Path Planning

- 회의에서 채택된 algorithm 과 함께 논의한 error cases 를 포함하여 pseudo code 작성

```
void path_planning(i){   //i=number of blue ball in field

        if i=0{ // after checking that there's no blue ball in field
                scan green balls while turn CW
                after scan 2 green balls, go to center of 2 balls
                turn 180 degree
                go back for 5 seconds
                end
        }

        elif i=1{ // after picking twe blue balls
                scan blue balls while turn CW
                after scan a blue ball, turn toward the blue ball
                approach to the blue ball with avoid red balls
                go straight with operating the propeller(?)
                scan blue balls while turning 360 degree
                path_planning(i)
        }

        elif i=2{ //after picking one blue ball
                scan blue balls while turn CW
                after scan a blue ball, turn toward the blue ball
                approach to the blue ball with avoid red balls
                go straight for 3 seconds with operating the propeller
                path_planning(1)
        }

        else{ //at first
                scan blue balls and choose the rightest one
                turn toward the blue ball
                approach to the blue ball with avoid red balls
                go straight for 3 seconds with operating the propeller
                path_planning(2)
        }
}
```

## 7th Week(4/30~5/6)

5/1 Group Meeting

- Design Review #2 발표 컨텐츠에 대한 정리

  - 열, 진동, Picking, Vision, ROS integration, Motor control 로 나누어 진행
  - ROS integration 의 경우 motor control, vision system 과 통신하고 path planning algorithm 을 code 로 작성하기로 함

5/2 Group Meeting

- 가까이에 있는 red ball 을 피해 blue ball 을 picking 하는 것까지 algorithm 작성

- webcam_node, ball_detection_node, data_integration_node 를 구동시키고 data 를 정상적으로 보내는지 확인

5/3 Group Meeting

- 발표 자료 준비

- 5/2 에 구동한 node 를 이용하여 실제 hardware 에 적용하여 하나의 blue ball picking 까지 성공

5/4 Design Review #2

- 2 차 발표 참관

8th Week(5/7~5/13)

5/8 Group Meeting

- Design Review #2 에 대한 피드백
  - 열 관리의 필요성과 우리가 실행하는 열 관리법에 대한 설명이 중요
  - Ball picking, 열 관리, 시간 단축도 중요하지만 창의적인 부분 어필의 필요성
  - Hardware 가 완성된 직후부터 최대한 많이 연습할 수 있도록 미리 path planning algorithm 의 완성도를 높여야 함.
- 최종발표까지 열 분석 및 관리와 경로 설정으로 팀을 나눠 진행하기로 함.

5/9 Path Planning Algorithm

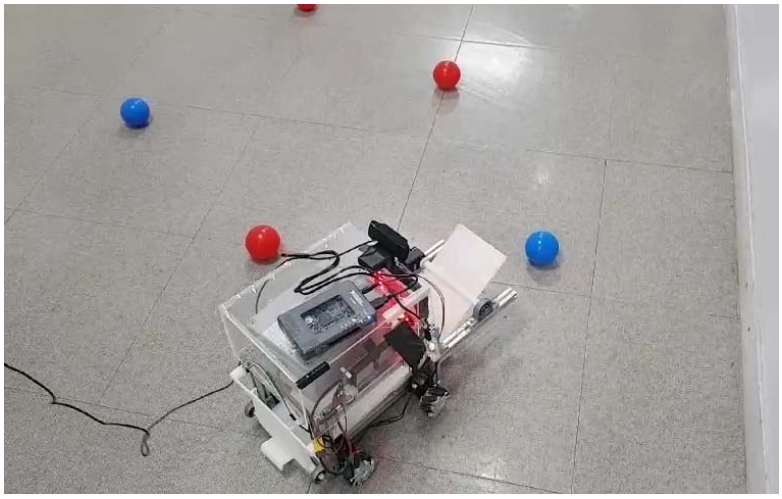
- 결정된 Path Planning Algorithm 에 창의적인 요소를 추가할 수 있는지 고려

- Red ball 을 피해 Blue ball 3 개를 모두 picking 하는 것까지 code 수정

5/10 Software Group Meeting

- 실제 하드웨어와 실습동 복도 환경에 적합하도록 parameter 를 수정한 뒤 구동

- Error 를 통해 수정해야 할 부분 확인



5/11 Software Group Meeting

- 전날 발견한 error 를 수정하고 전체 mission 수행을 위해 코드 추가
  - 3 개의 blue ball 을 picking 한 뒤에 green ball 을 detect 하여 멈추는 것까지 진행
  - 가장 오른쪽에 위치한 blue ball 을 picking 하는 algorithm 의 일관성을 위한 최초 위치 설정 추가
- mission 에 적합하도록 ball_detection_node.cpp 수정
- green ball 을 detect 한 뒤에 release 하기까지의 algorithm 고안

5/13 Software Group Meeting

- 5/10, 5/11 에 test 한 결과를 보완하여 code 수정 및 test 진행

## 9th Week (5/14~5/20)

5/15 Group Meeting

- 교수님과의 식사 및 경과보고
- 이후 창시구실에서 코드 수정 및 실행

5/17 Software Group Meeting

- 큰 바퀴로 교체한 후 다음날 있을 demo 에 바로 사용할 수 있을 지 test
  - 바퀴의 size 가 바뀜에 따라 직진 거리, 회전 각도 등 새롭게 계산
  - 계산한 내용을 반영하여 data_integration_node.cpp 의 수치 변경

5/18 Demo

- 공동강의실에서 첫 demo
- 교체한 바퀴의 축이 안쪽으로 기울어져 있고, 작동 시 이탈하는 현상 발생
  - 바퀴 고정 장치 추가하기로 함
  - OpenCV 의 HSV 값을 조정하여 특정한 조명에서 ball 을 가장 잘 detect 하도록 함

5/20 Software Group Meeting

- 주말 동안 지난 demo 에서 발견된 바퀴 관련 오류를 수정한 뒤 월요일 demo 를 위한 test
  - 바퀴의 모터 축이 기울어지지 않도록 하는 장치가 추가됨
  - 고정된 바퀴에 맞게 수치 조정

## 10th Week (5/21~5/27)

5/21 Demo

- 보완된 hardware 를 이용하여 demo 실행
  - 전반적으로 mission 을 잘 수행하는 것을 확인
  - 바닥의 mic 단자 덮개가 금속 재질이고 지면을 불 균일하게 만들어 slip 이 생기는 것을 확인
  - Green ball 을 detect 하여 바구니에 release 하는 algorithm 보완 필요성 확인

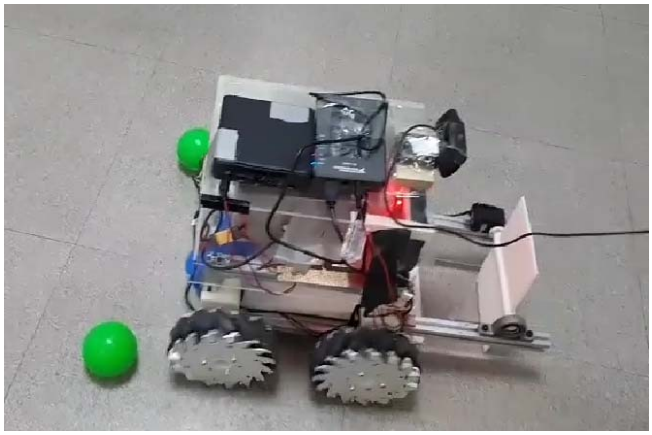
5/22 Group Meeting

- 경과 보고 및 최종발표까지 할 일 정리

- 발표자 선정 및 발표 준비 시작

- green ball detect 후 release 하는 code 수정 및 실행

- NUC 로 전체 시스템 구동



두 개의 green ball 을 detect 하여 정 가운데로 후진

5/23 Software Group Meeting

- 바퀴 높이 조정을 통해 보다 정확하게 움직이도록 조정

- right most blue ball 로 가는 경로에 다른 blue ball 이 있는 경우 오류가 발생할 수 있음을 확인하고 red ball 과 같이 피해가도록 code 수정

5/25 Group Meeting

- Hardware team 에서 cooling fan 을 추가하여 duct 를 새롭게 수정했고, 잘 작동하는지 test

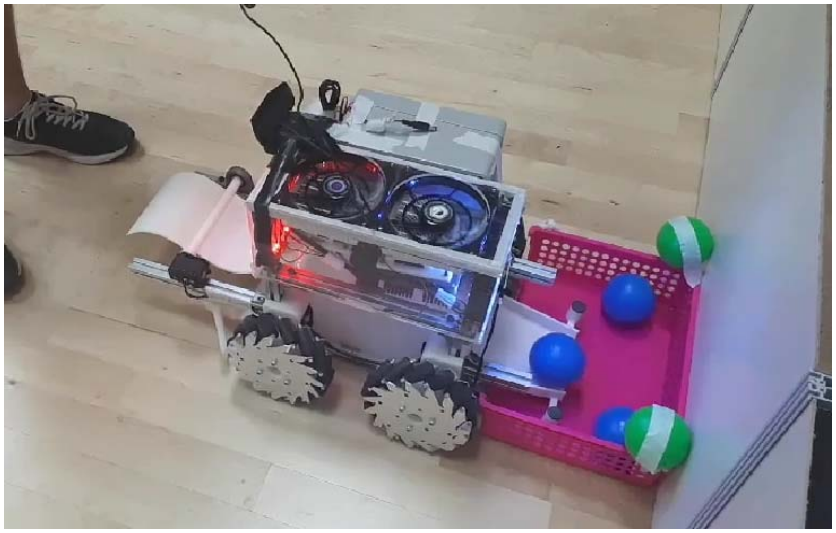
  - 이전처럼 잘 작동하는 것을 확인
  - 바퀴 높이 차이 문제로 직진 시 왼쪽으로 치우치는 현상 발생

5/26 Demo

- 전반적으로 mission 을 잘 수행하나 몇 가지 오류 해결 필요성 확인

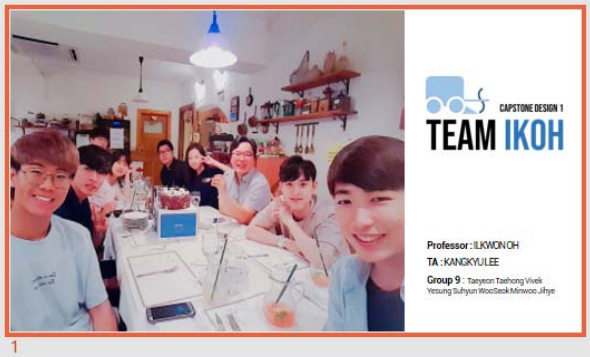
  - Red ball 을 피할 때 간혹 반대로 움직이는 경우가 발생

- Green ball 을 detect 하여 바구니에 release 할 때 정확하지 않는 경우가 발생



## 4. Presentation Materials

1, 2, 3차 발표 중 한 학기 동안 진행한 내용이 모두 포함된 Final Presentation 슬라이드를 아래에 첨부했다. 이 프로젝트를 진행하면서 고려해야 했던 Heat transfer, Pick-up system, Motor control, Vibration control, Vision & ROS integration을 모두 포함하여 발표 자료를 만들었고, 실제 프로젝트를 위해 고민한 흐름에 맞게 순서를 배열했다. OpenCV와 ROS 파트의 분량이 적다고 생각될 수 있는데, OpenCV와 ROS로 새롭게 배우고 연구할 수 있는 내용은 2차에서 모두 끝났기 때문이다.


CAPSTONE DESIGN 1
TEAM IKOH
Professor : ILKWON OH
TA : KANGKYU LEE
Group 9 : Taeyeon Taehong Vivek
Yesung Suhyun Woo Seok Minwoo Jihye

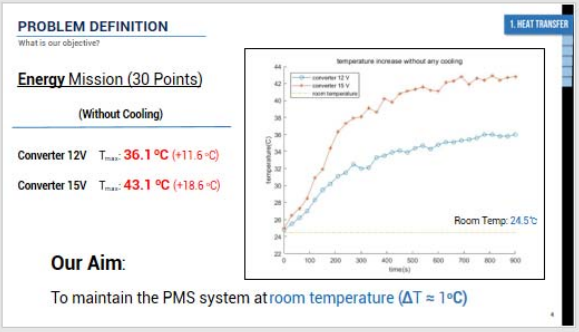
1


1 HEAT TRANSFER
2 PICK-UP SYSTEM
3 MOTOR CONTROL
CONTENTS
4 VIBRATION CONTROL
5 VISION & ROS ALGORITHM
6 SUMMARY

2


1. HEAT TRANSFER
1. HEAT TRANSFER

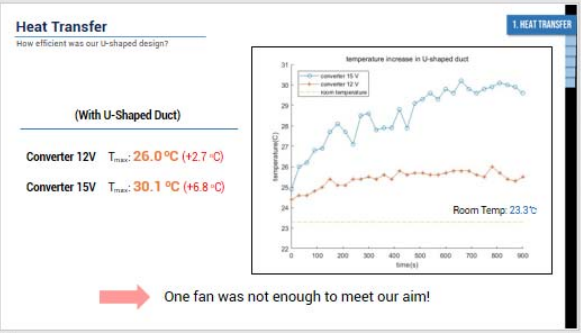
3


PROBLEM DEFINITION
What is our objective?
Energy Mission (30 Points)
(Without Cooling)
Converter 12V Tmax: 36.1 °C (+11.6 °C)
Converter 15V Tmax: 43.1 °C (+18.6 °C)
Room Temp: 24.5°C
Our Aim:
To maintain the PMS system at room temperature (ΔT ≈ 1°C)
1. HEAT TRANSFER

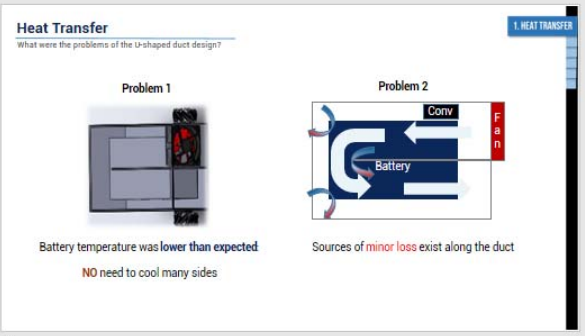
4


Heat Transfer
What is our previous design?
Battery
Conv
Fan
Can cool many surfaces (Battery (6 sides), Converter, myRio, etc) with only one fan
1. HEAT TRANSFER

5


Heat Transfer
How efficient was our U-shaped design?
(With U-Shaped Duct)
Converter 12V Tmax: 26.0 °C (+2.7 °C)
Converter 15V Tmax: 30.1 °C (+6.8 °C)
Room Temp: 23.3°C
One fan was not enough to meet our aim!
1. HEAT TRANSFER

6


Heat Transfer
What were the problems of the U-shaped duct design?
Problem 1
Problem 2
Conv
Fan
Battery
Battery temperature was lower than expected
NO need to cool many sides
Sources of minor loss exist along the duct
1. HEAT TRANSFER

7


Heat Transfer
How can we improve our duct design?
One fan was not enough to meet our aim!
Use CPU fan to efficiently extract heat
FAN
New Linear Duct
High Temp Parts (myRio, U2D2)
Side View
Use one fan for each converter!
Is this enough?
1. HEAT TRANSFER

8

## Heat Transfer

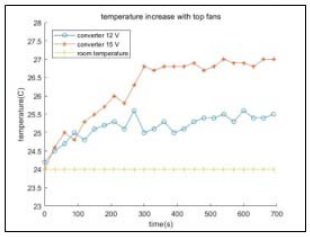
How efficient is the 2-fan design?

1. HEAT TRANSFER

(With Top Fans)

Converter 12V $T_{max}$: 25.6 °C (+1.6 °C)

Converter 15V $T_{max}$: 27.0 °C (+3.0 °C)




**How can we improve this even more?**

9

## Heat Transfer

Heat Transfer & Fluid Mechanics Analysis*

1. HEAT TRANSFER

$Q = h_{conv} A (T_s - T_{air})$

How can we increase h?

$h_{conv}$ ↑ ⟸ $velocity$ ↑

Fans are **pumps** that **increases fluid head** to induce fluid flow

$H_{pump} = \frac{P_2 - P_1}{\rho g} + \frac{(\alpha_2 V_2^2 - \alpha_1 V_1^2)}{2g} + (z_2 - z_1) + h_{turbine} + h_{friction}$

where $h_{friction} = \frac{V^2}{2g}(\sum \frac{fL}{D} + \sum K)$

Our f ↑ so our V ↓

$H_{pump}$ ↑ ⟹ $v_{fluid}$ ↑ ⟹ $h_{conv}$ ↑ ⟹ $R$ ↓ ⟹ Steady state temp ↓

*(White 8th Fluid Mechanics)

10

## Heat Transfer

How does the improved cooling system work?

1. HEAT TRANSFER

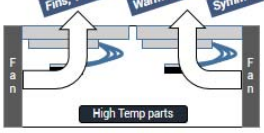
Add more pumps (fans) in series!

Fins, Thermal Pads, Thermal Grease

Warm Air Rises (Efficient)

Symmetrical (Even Cooling)

Focused Cooling of Parts




Symmetry Color Proportion

**Aesthetics is important in engineering**

11

## Heat Transfer

How does the improved cooling system work?

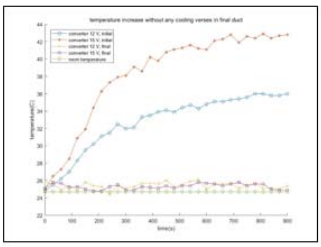
1. HEAT TRANSFER

12

## Heat Transfer

How effective is our new design?

1. HEAT TRANSFER




(With Final Duct)

Converter 12V $T_{max}$: **26.1 °C** (+1.4 °C)

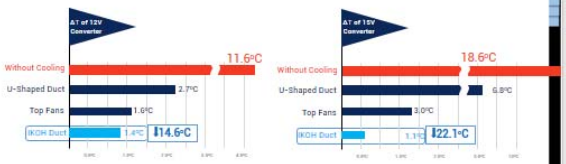
Converter 15V $T_{max}$: **25.8 °C** (+1.1 °C)

13

## Heat Transfer

What is ΔT of each design?

1. HEAT TRANSFER




Aim: ΔT ≈ 1°C ✓

**I**mproved groundbrea**K**ing co**O**ling met**H**od

14

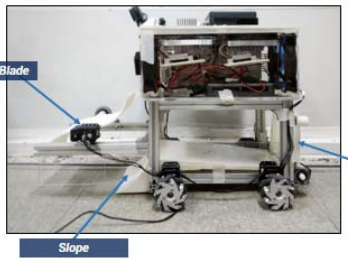
2. PICK-UP SYSTEM

15

## Pick-Up System

How do we pick up the ball?

2. PICK-UP SYSTEM



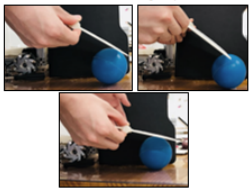

16

## Pick-Up System

How did we modify the blade?

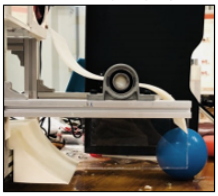
2. PICK-UP SYSTEM

Previous blade (plate)



Problem : Balls get stuck in many angles

Modified blade (curved) shape



Solution : Balls rarely get stuck

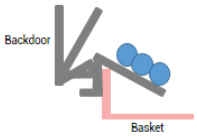
17

## Pick-Up System

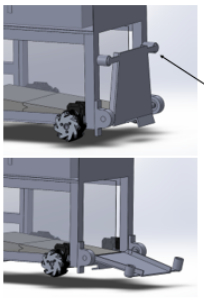
How do we release the ball?

2. PICK-UP SYSTEM

Non Actuator Mechanism




Cost is important in engineering!

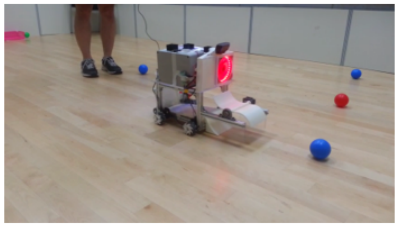



18

## Pick-Up System

Ball pick-up

2. PICK-UP SYSTEM

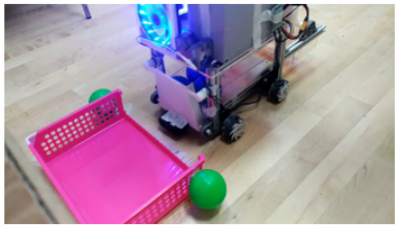


19

## Pick-Up System

Ball release

2. PICK-UP SYSTEM



20

3. MOTOR CONTROL

21

## Motor Control

Wheel Kinematics

3. MOTOR CONTROL

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \frac{R}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{l_1+l_2} & \frac{1}{l_1+l_2} & \frac{1}{l_1+l_2} & \frac{1}{l_1+l_2} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \rightarrow \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{R} \begin{bmatrix} 1 & 1 & -(l_1+l_2) \\ 1 & -1 & l_1+l_2 \\ 1 & -1 & -(l_1+l_2) \\ 1 & 1 & l_1+l_2 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix}$$

By using wheel kinematics...

- Accurate control is available.
- Green ball location can be estimated.

22

## Motor Control

How to reduce vibration?

3. MOTOR CONTROL

Vibration reduction

- Reference filtering → Motor control (Software)
- Vibration isolator → Vibration control (Hardware)

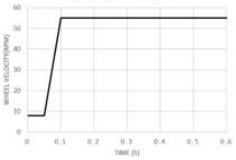
23

## Motor Control

Vibration reduction using software

3. MOTOR CONTROL

STEP SPEED INPUT




**Problems of step function**

- Sudden acceleration and braking induce vibration can damage motors.
- Rotation in place was not operated well because of webcam delay and low rpm.

➡ Motor input should be modified.

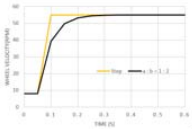
24

## Motor Control

Vibration reduction using software

3. MOTOR CONTROL

SPEED DAMPING



**Solution**

- In each loops, the mean values of **previous and present storage variables** are used to calculate what is required in the next step
- We fixed the degree of damping for rotation as 1:2.

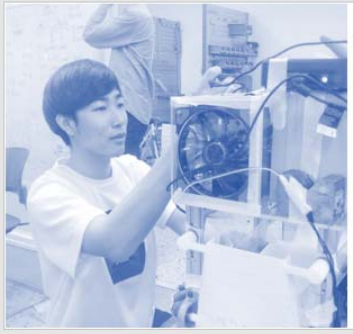
$T = nT = 2(s), n = 40 \gg \tau = 0.05 \gg H = \frac{1}{\tau s + 1}$

By Z transform, $\frac{Y}{U} = \frac{0.6321}{z - 0.3679}$, $(1 - 0.3679z^{-1})Y = 0.6321z^{-1}U$

$y(n) - 0.3679y(n-1) = 0.6321u(n-1)$

$\therefore y(n) = 0.3679y(n-1) + 0.6321u(n) \simeq \frac{y(n-1) + 2u(n)}{3}$ ➡ Act as low-pass filter : Reference filtering

25 ★



4. VIBRATION CONTROL

## 4. VIBRATION CONTROL

26

## Vibration Control

Cause of vibration

4. VIBRATION CONTROL

$f_{vib} = 7.150Hz$

$\omega_{f,vib} = 2\pi f_{vib} = 2\pi \times 7.150Hz = 44.981rad/s$

$\omega_{f,Hz} = 2\pi f = 2\pi\omega_{wheel}$

$= 2\pi \times (55rpm/60sec) = 46.077rad/s$

∴ Excitation by Mechanum Wheel

$\omega_f = 46.077rad/s$

27 ★

## Vibration Control

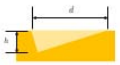
Vibration system design

4. VIBRATION CONTROL


Sudden acceleration

Sponge's Material property

$k_{sponge} = \frac{F}{\Delta x} = \frac{0.82kg \times 9.8m/s^2}{0.5mm} = 1.607 \times 10^4 N/m$



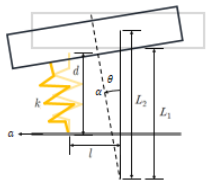

$k = k_{sponge} \times \frac{dh/2}{\pi D^2 h/4} = 247.421 N/m$

28 ★

## Vibration Control

Vibration system modeling

4. VIBRATION CONTROL




$\alpha = L_2 a$

$x = -l$

$x' = -L_1 \sin\theta - l\cos\theta$

$y = d$

$y' = -L_1(1 - \cos\theta) + d - l\sin\theta$

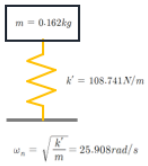
$\Delta x = x - x'$

$\Delta y = y - y'$

$m L_1^2 \ddot{\theta} = -k\left(\sqrt{\Delta x^2 + \Delta y^2}\right) l\cos\theta$

$\therefore m\ddot{\theta} = -k'\theta$

$k' = 108.741 N/m$




$\omega_n = \sqrt{\frac{k'}{m}} = 25.908rad/s$

29 ★

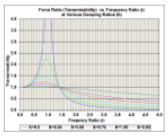
## Vibration Control

Reducing hardware vibration

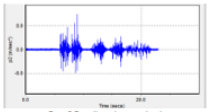
2. Vibration control

$\omega_n = \sqrt{\frac{k'}{m}} = 25.908rad/s$

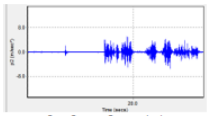
$r = \frac{\omega_f}{\omega_n} = \frac{46.077rad/s}{25.908rad/s} = 1.778$



∴ Transmissibility ≃ 0.5

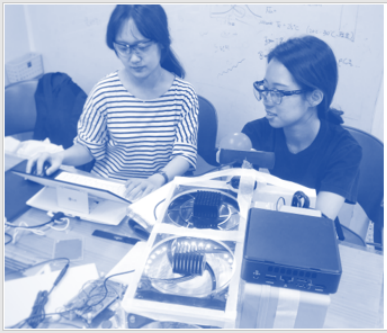


Cam&Car directly attached



Cam-Sponge-Car attached

30



5. VISION & ROS

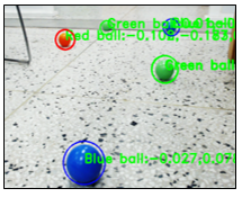
## 5. VISION & ROS ALGORITHM

31

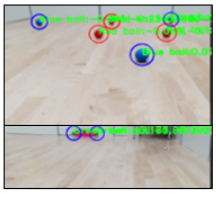
## VISION & ROS

Vision processing modification

5. VISION & ROS

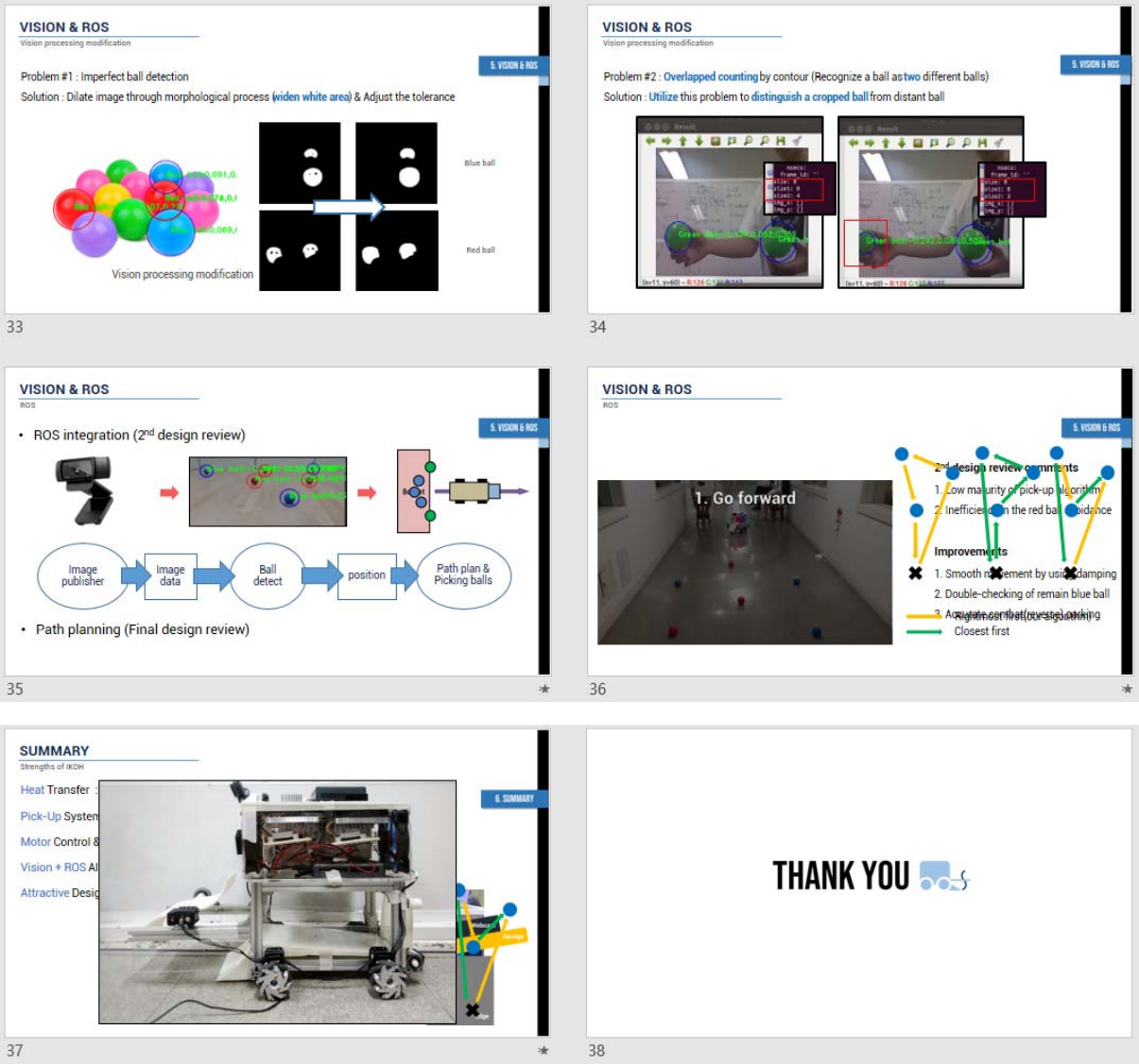



Actual recognition(in Capstone room)



Actual recognition(in Lecture room)

32




## 5. Conclusion

창의적 시스템 구현은 기계과 학생이라면 누구나 기대하고 또 걱정하는 과목일 것이다. 나는 3학년 가을학기에 공학설계와 시스템 모델링 및 제어를 들으며 다음학기도 저녁시간을 버리고 싶지 않다는 생각을 해서 1년 뒤인 5학년 봄학기에 듣게 되었다. 작년에 수강한 친구들의 조언도 들으며 미리 마음의 준비를 하고 있었는데, 갑자기 수업이 완전히 달라졌다.

새로운 방식의 수업에 학생들도, 조교님들도, 교수님들도 모두 낯설고 새로웠지만, KAIST 기계과답게 3달이라는 짧은 시간동안 잘 해낸 것 같다. 딱 정해진 미션만 수행하면 되니까 밤샐 일도 없을 줄 알았는데, 발표 전마다 밤을 새고, 새벽 5시에 데모 시간을 배정받아 공동강의실로 갔던 경험은 추억으로 평생 남을 것이다. 데모가 끝나고 돌아온 창시구실은 저녁 9시처럼 밝고 분주했

었다. 이렇게 모두가 함께 고생했고, 각자 고생하면서 내년에 들을 후배들은 덜 고생하기를 바라며, 또 이 수업이 잘 정착되길 바라며 많은 피드백을 남긴 것으로 알고 있다. 비록 이 수업을 처음 듣는 학생으로써 고생하며 배운 것도 많지만, 수업이 더 다듬어져서 필요 이상의 고생은 안 했으면 좋겠다.

마지막 Conclusion을 넣을 생각이 없었는데, final report를 완성하고 나니 일반 수업 같지 않아서 이렇게 짧게 나마 감상을 적어봤습니다. 3월부터, 이 수업을 준비한 교수님들과 조교님들은 1월부터 6월 초까지 정말 고생하셨습니다. 가을학기에도 다함께 열심히 해서 좋은 결과 있기를 바라겠습니다.