**ME400: 창의적 시스템 구현 - 최종 보고서**

2018-spring

**Team: 걷지 말고 기어**

**20150094 김덕영**

**§목차**

**1. Hardware Design**

**1.1 Pick-up module**

**1.2 Fin & Fan**

**1.3 Gear-Wheel System**

**1.4 차체**

**2. Software design**

**2.1 Overall algorithm**

**2.2 Ball detection**

**2.3 Pick the ball**

**2.4 go to the basket**

**3. Process**

**3.1 Monthly plan**

**3.5 ROS**

**1. Hardware Design**

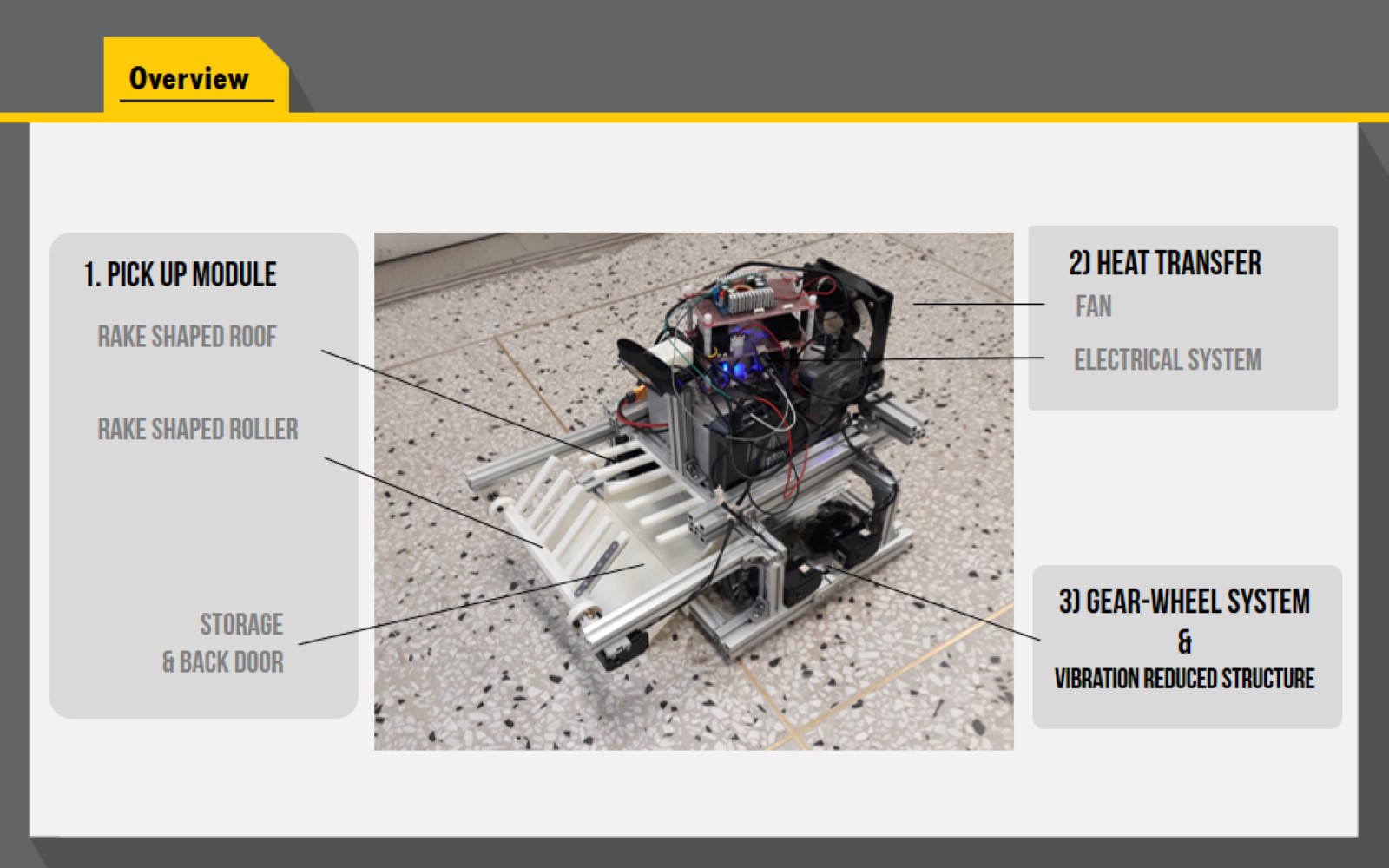
****

Figure 1 Overview of hardware

Mobile platform의 구조는 위와 같다. 특징적인 구조는 pick-up module, 열 전달 설계(fin & fan), 그리고 기어 시스템이 있다. 카메라 위치, 차체의 크기 또한 최적으로 설계하였다.

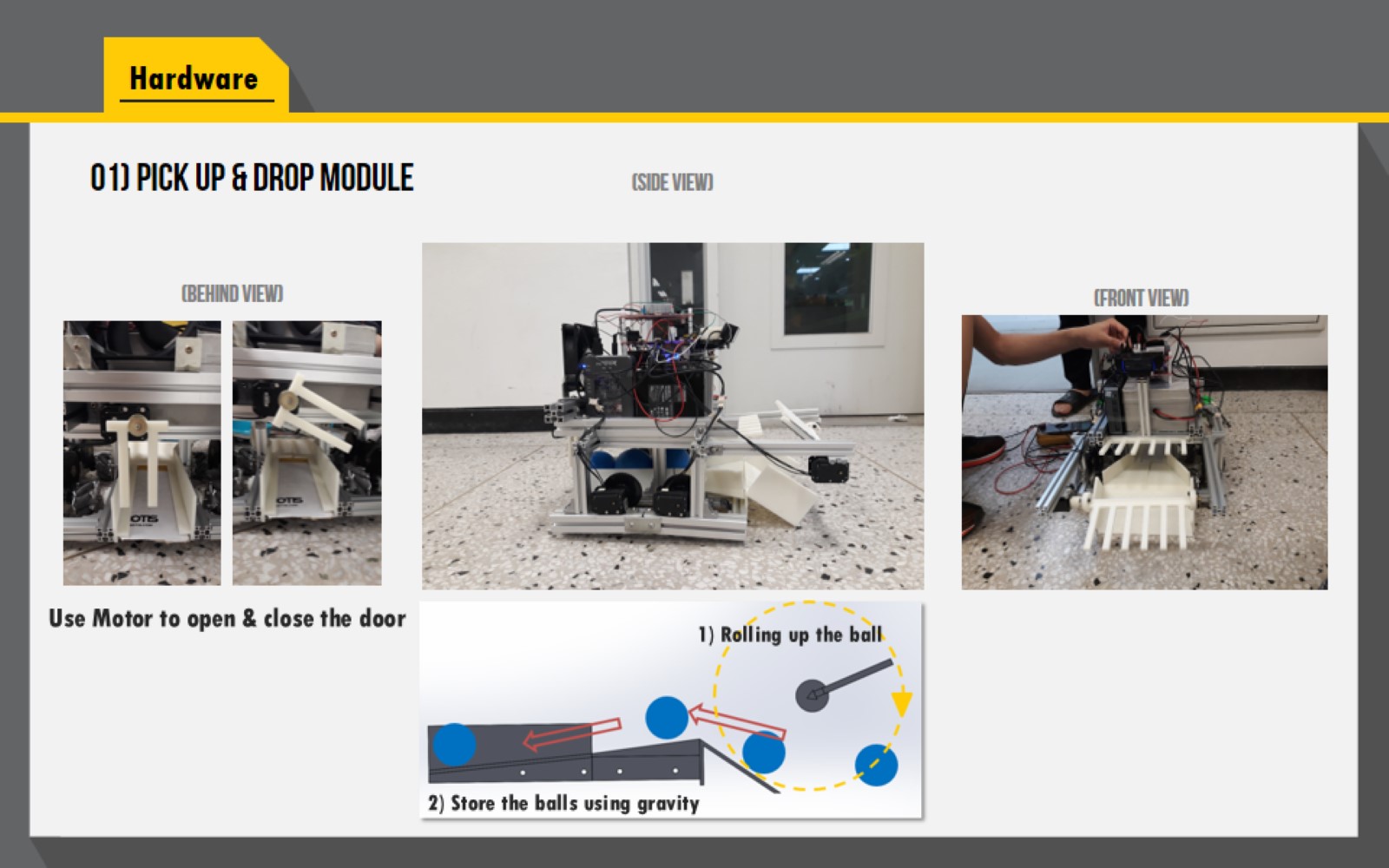
* 1. **Pick up module  
     **

Figure 2 pick-up & drop module

다음은 공을 주워 담고, 내려놓는 모듈의 앞, 옆, 뒤에서 본 모습이다. 공은 차가 앞으로 이동하면서 앞 쪽에 있는 롤러에 의해서 앞쪽 경사면을 타고 들어올려진다. 경사면을 지나고 난 이후에는 중력에 의해 뒤쪽으로 굴러갈 수 있도록 경사로 형태로 저장소를 설계하였다.  
**1) Roller design**

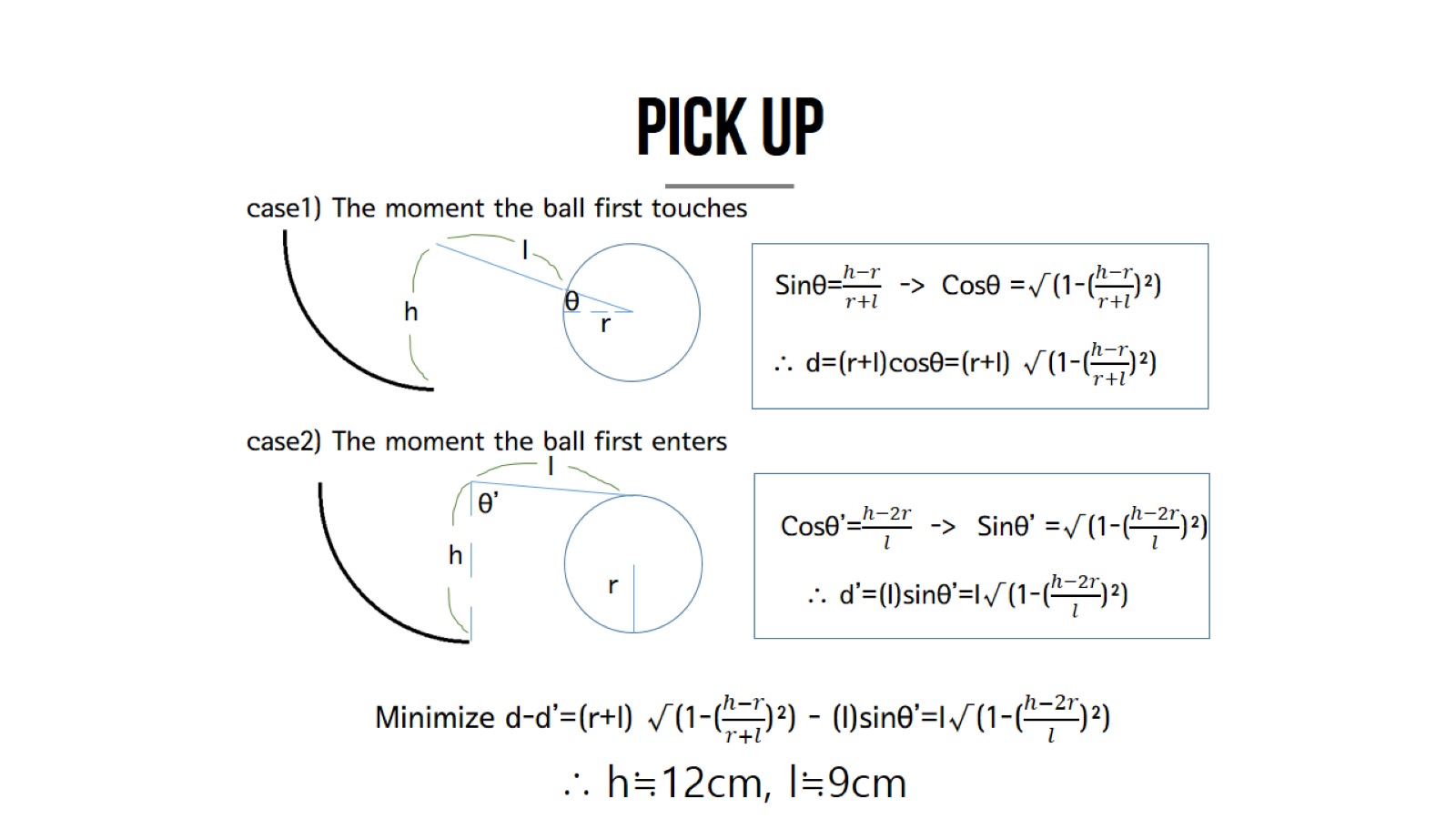


Figure 3 optimal dimension for roller

Figure 3은 공을 잡을 확률이 가장 높은 roller의 길이와 roller 중심 높이의 위치를 계산한 것이다. 그러기 위해서는 공을 놓치는 공의 위치 범위를 최소로 만들어야 한다. 그 결과 roller의 길이는 9cm, 중심의 높이는 12cm로 설정되었다.  
또한 roller에 의해 공이 다시 빠져나갈 수 없도록 위쪽에 가림 막을 설치하였다.

**2) Drop module**

저장고가 뒤쪽으로 경사져 있기 때문에 뒤에 아무런 장애물이 없다면 공이 자연스럽게 떨어지도록 되어있다. 따라서 공을 주울 때는 저장소의 뒤쪽을 막아 두었다가 공을 떨어뜨릴 때에만 막아둔 것을 치우도록 했다. 다른 전원을 쓰지 않는 방법도 존재했지만 설계의 복잡도가 올라간다. 따라서 모터를 이용해 뒤의 장애물을 회전시켜 떨어뜨리도록 했다.

**3) Rake shape**

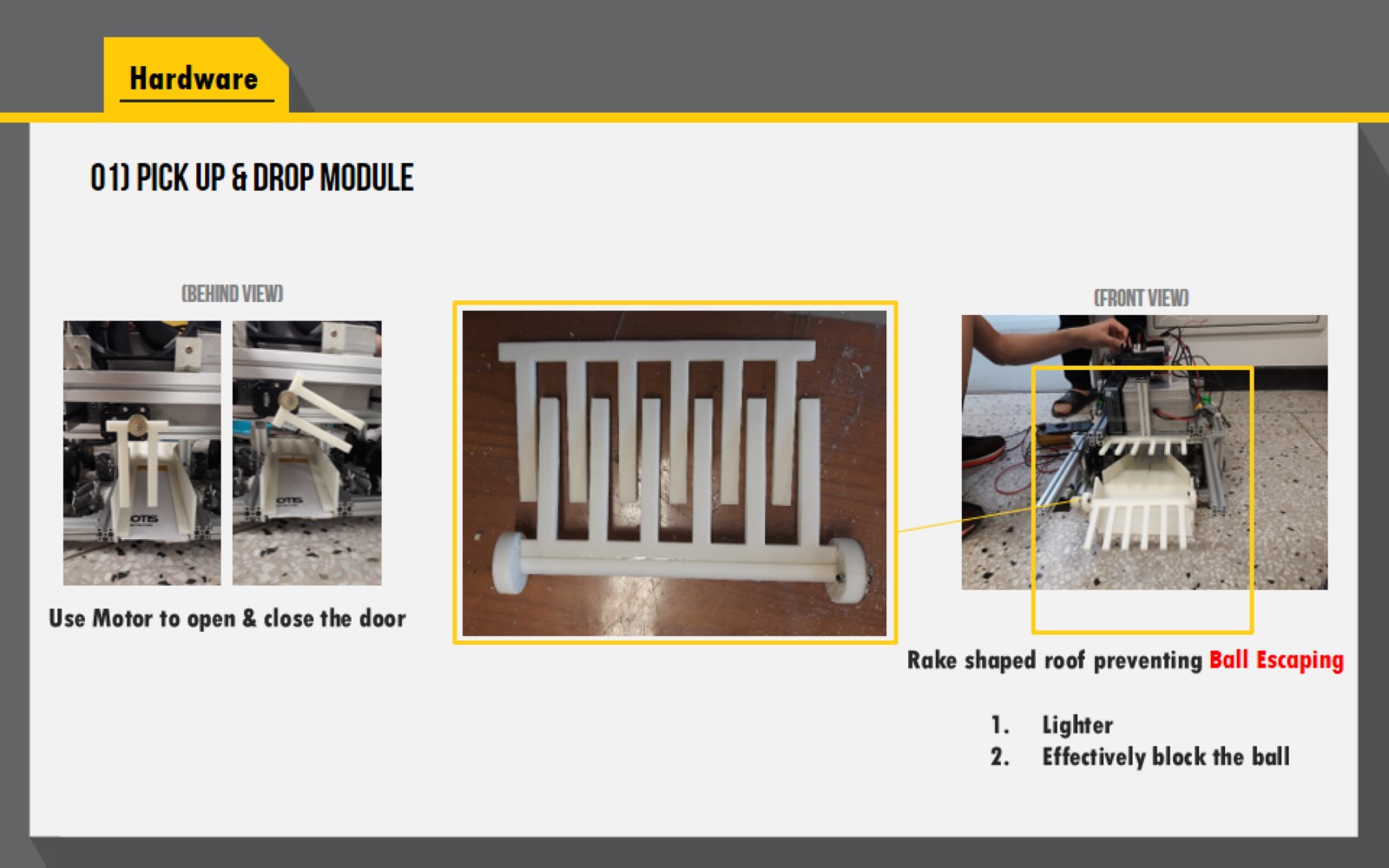


Figure 4 rake shaped

가림 막의 설치가 용이하고, 무게를 줄여 에너지 효율을 높이기 위해서 TRIZ 공간의 분리를 이용해 위와 같은 빗 모양의 구조를 사용하였다.

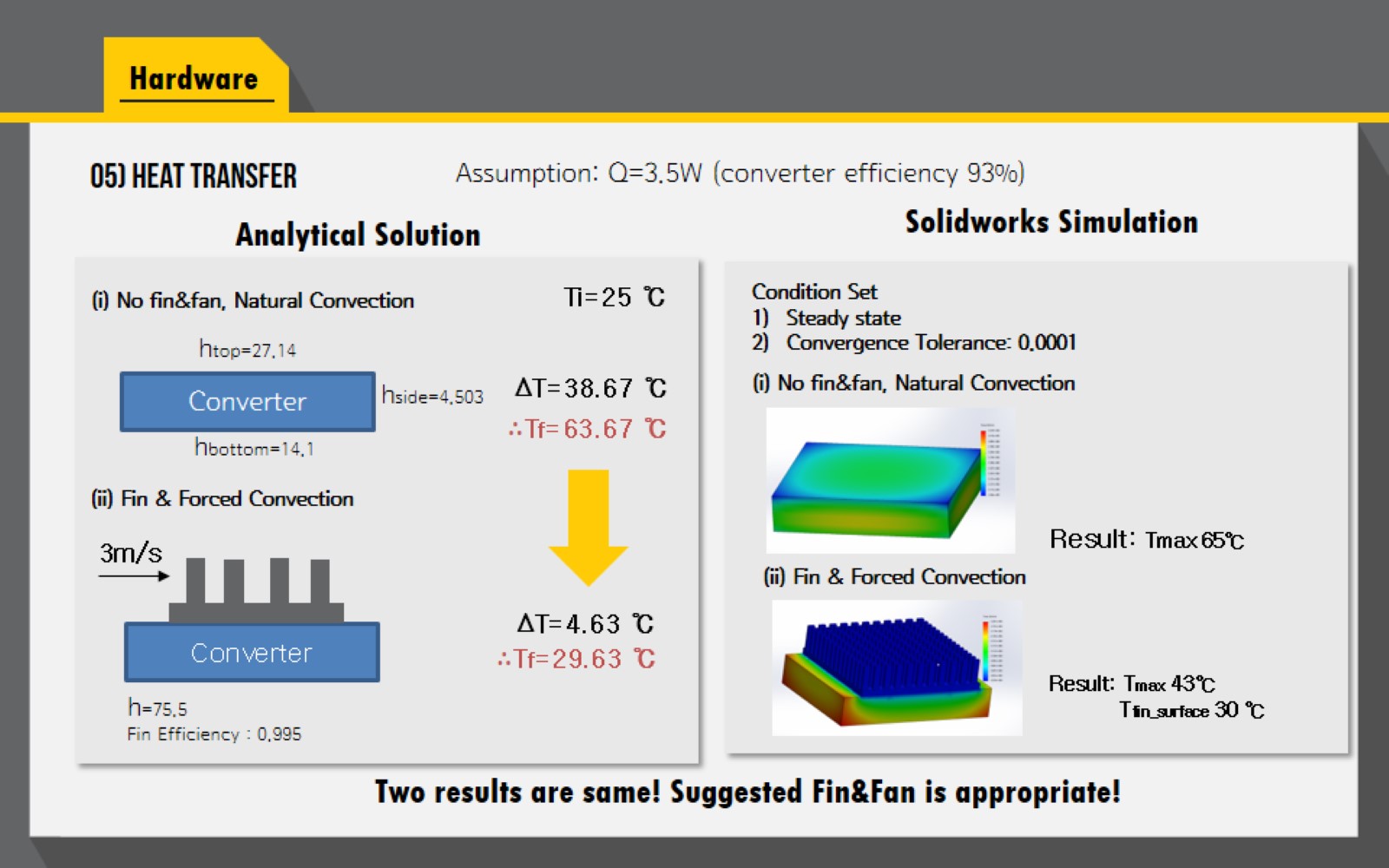
* 1. **Fan & Fin**차체의 최고 온도가 기준 온도인 70°C를 넘지 않았지만 가장 많은 에너지가 사용되기에 온도가 가장 높아졌던 모터 컨버터 위주로 냉각을 위한 설계를 진행하였다. Conduction은 냉매를 이용해야해서 설계의 복잡성이 커지고, radiation은 건물 내벽의 온도가 차체의 절대온도와의 차이가 크지 않기 때문에 영향이 적을 것으로 판단되어 fin과 fan을 이용한 forced convection을 사용하였다. ****

Figure 5 Heat analysis

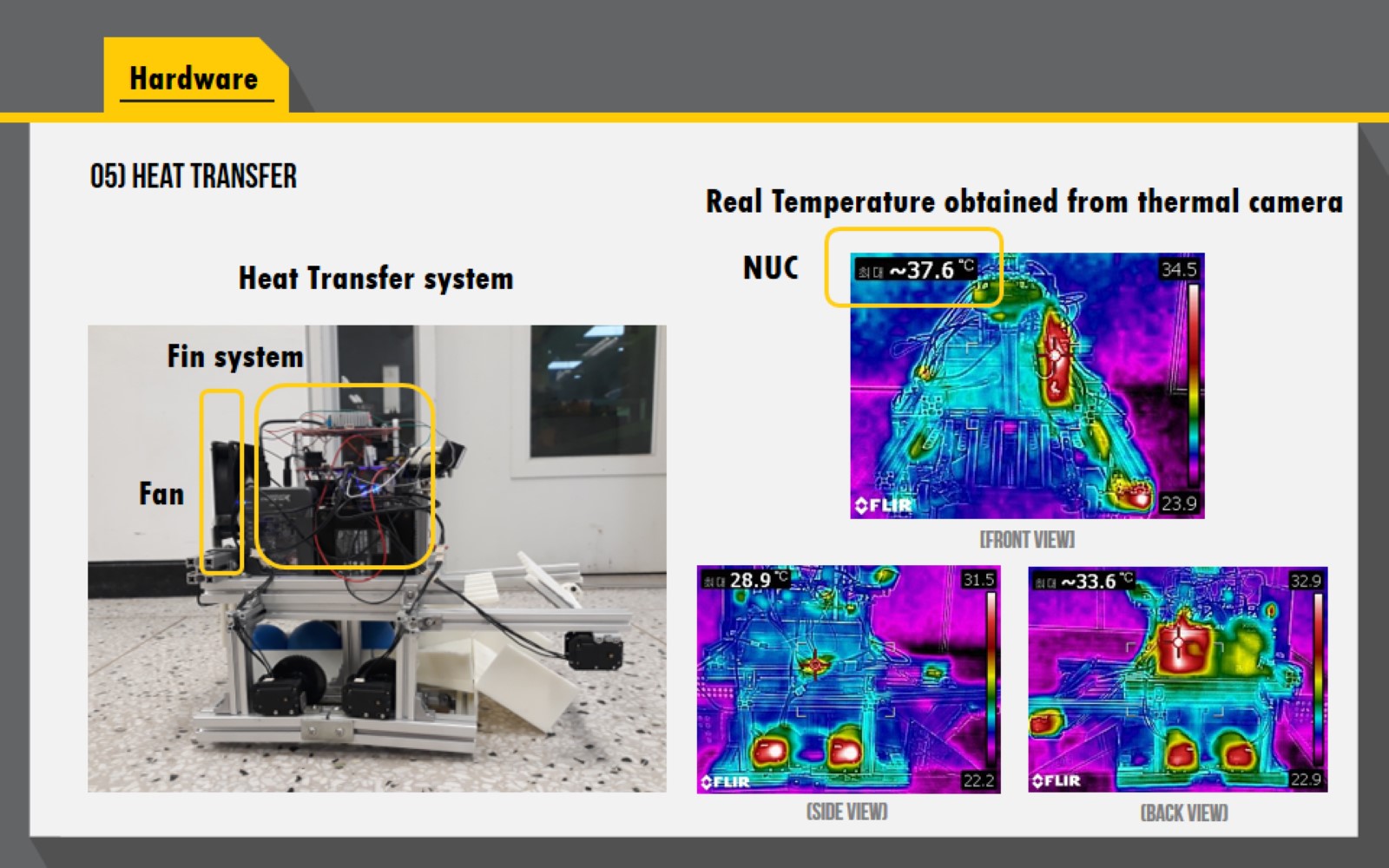
온도를 20°C이상 냉각시킬 수 있다는 것을 확인하였다.  
****

Figure 6 Temperature measured  
또한 실제 온도 측정 결과 또한 이론적인 분석을 뒷받침했다.

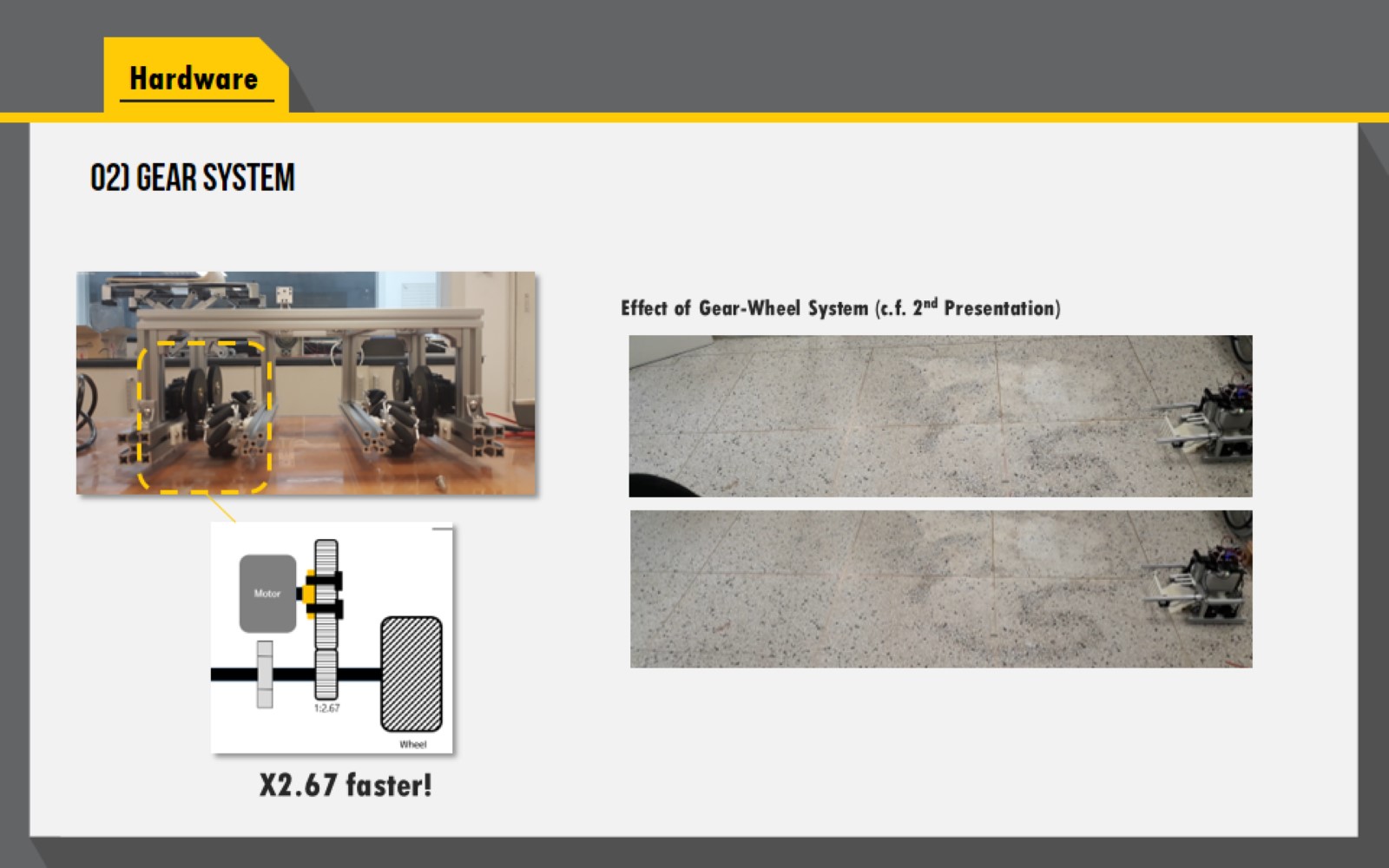
* 1. **Gear-Wheel system  
     **

Figure 7 Gear-Wheel system

기존에 주어진 모터 MX-28AT는 최대 55rpm으로 매우 느렸다. 따라서 기어를 사용하여 속도를 높였다. 기어는 바퀴의 크기 등을 고려하여 20:80 즉 기어 비 약 1:2.67로 설정했다. 결과적으로 약 2.6배 정도 속도가 빨라진 것을 확인할 수 있었다.  
또한 기어를 설계하면서 바퀴를 두 축으로 고정하면서 바퀴의 진동으로 인해 생기는 문제를 줄였다.

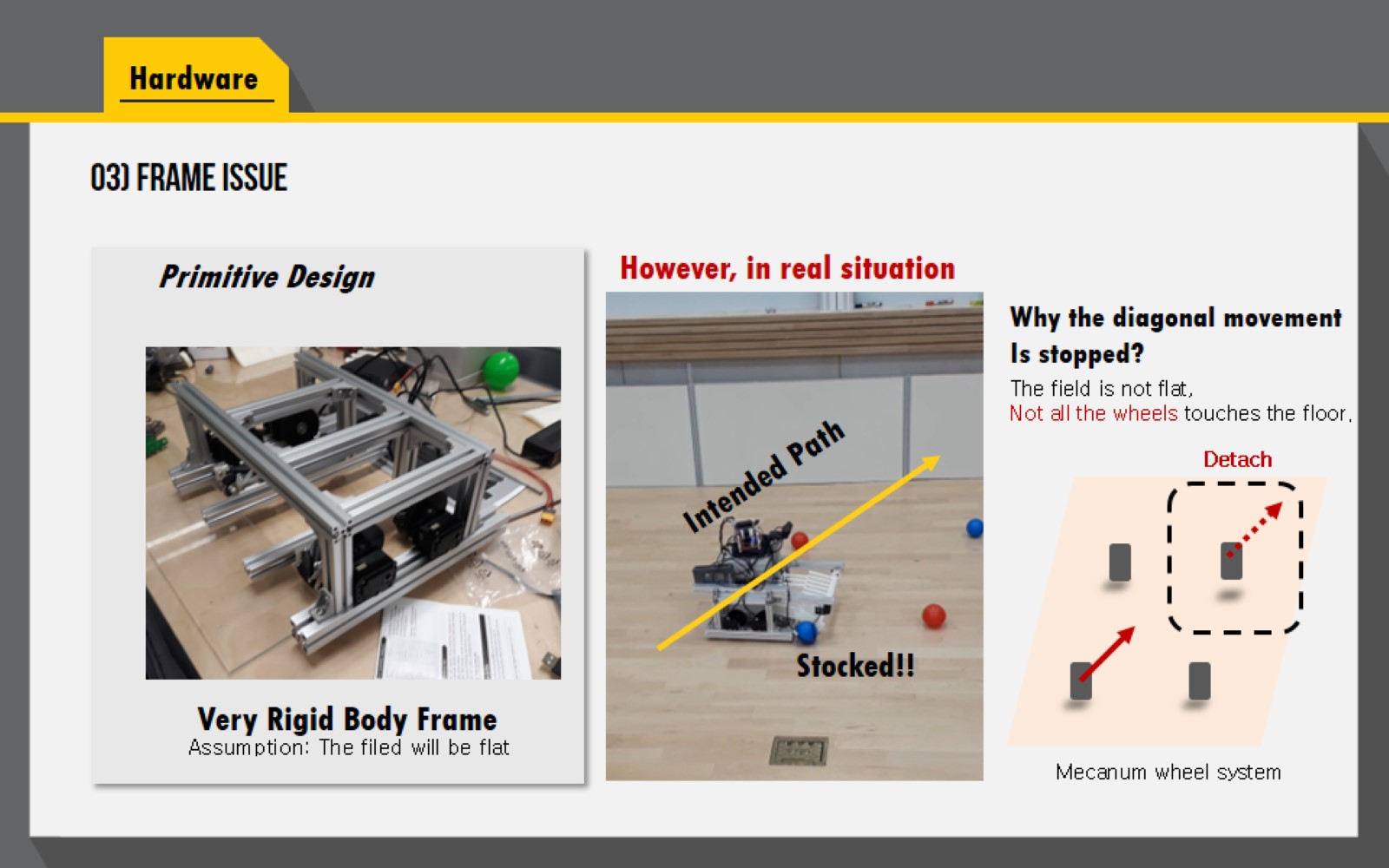
* 1. **차체  
     1) rigid body problem  
     **

Figure 8 rigid body problem

차체를 알루미늄 프로파일로 제작하고 바퀴를 두 축 고정하는 설계는 차체를 rigid하게 만들었다. 따라서 지면의 형태라는 noise로 인해 모든 바퀴가 땅에 고정되어 있지 않는 상황이 발생하였고, 이는 대각선 방향 진행에 큰 문제점으로 작용되었다. 따라서 차체에 자유도를 줄 수 있는 방법을 생각했었고, TRIZ기법을 이용하여 차체를 분리하여 회전 자유도를 부여할 수 있었다.  


Figure 9 Frame segmentation

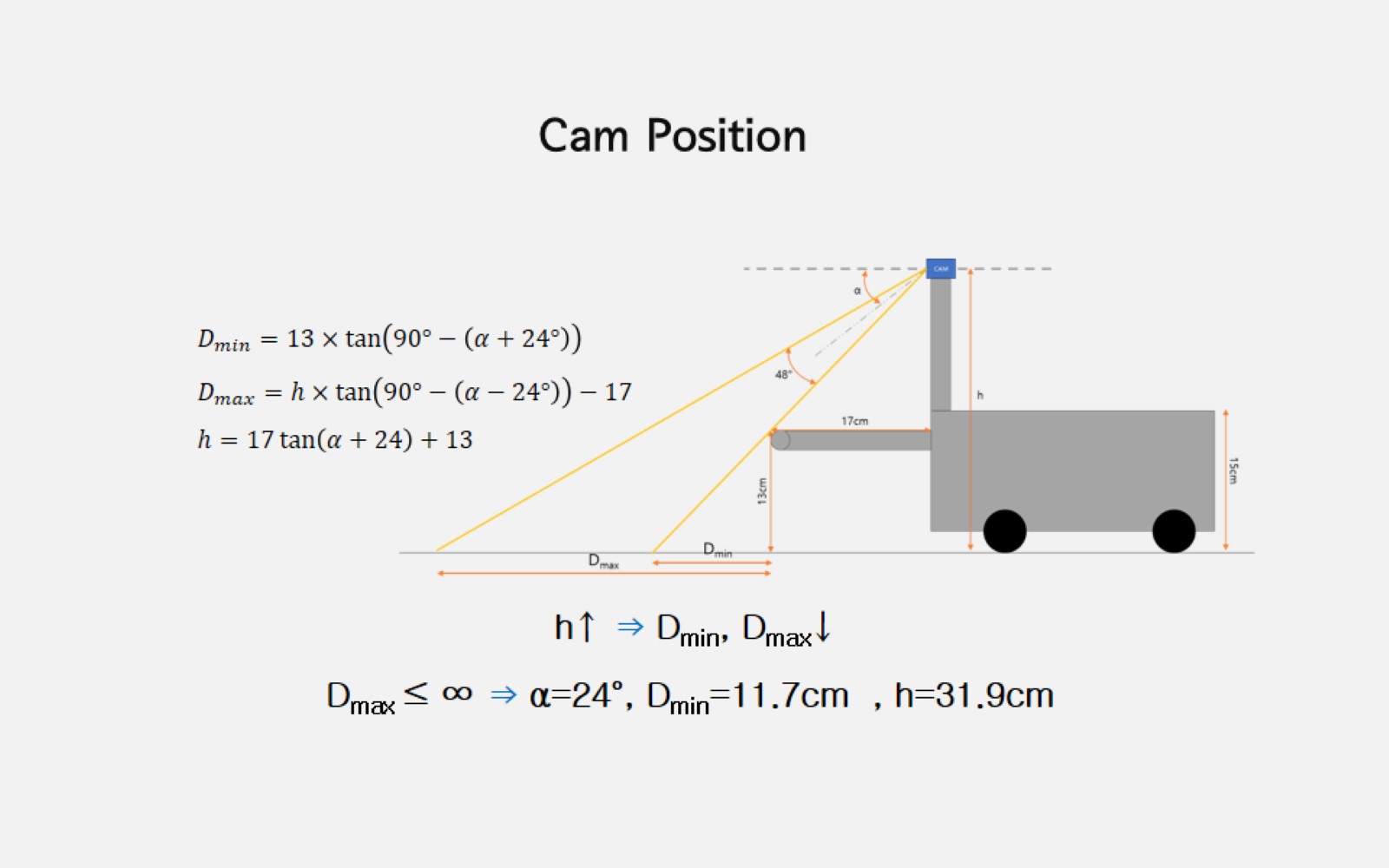
2)webcam위치 정하기  
최대한 가까운 곳까지 보면서 멀리 있는 공 또한 인식할 수 있도록 웹캠의 위치를 계산하였다. 대각선 시야 각(FOV)이 76°라는 사실을 이용해 가로(66°), 세로(48°)의 시야 각을 추정해 계산했다.  


Figure 10 Webcam position

**2. Software Design**

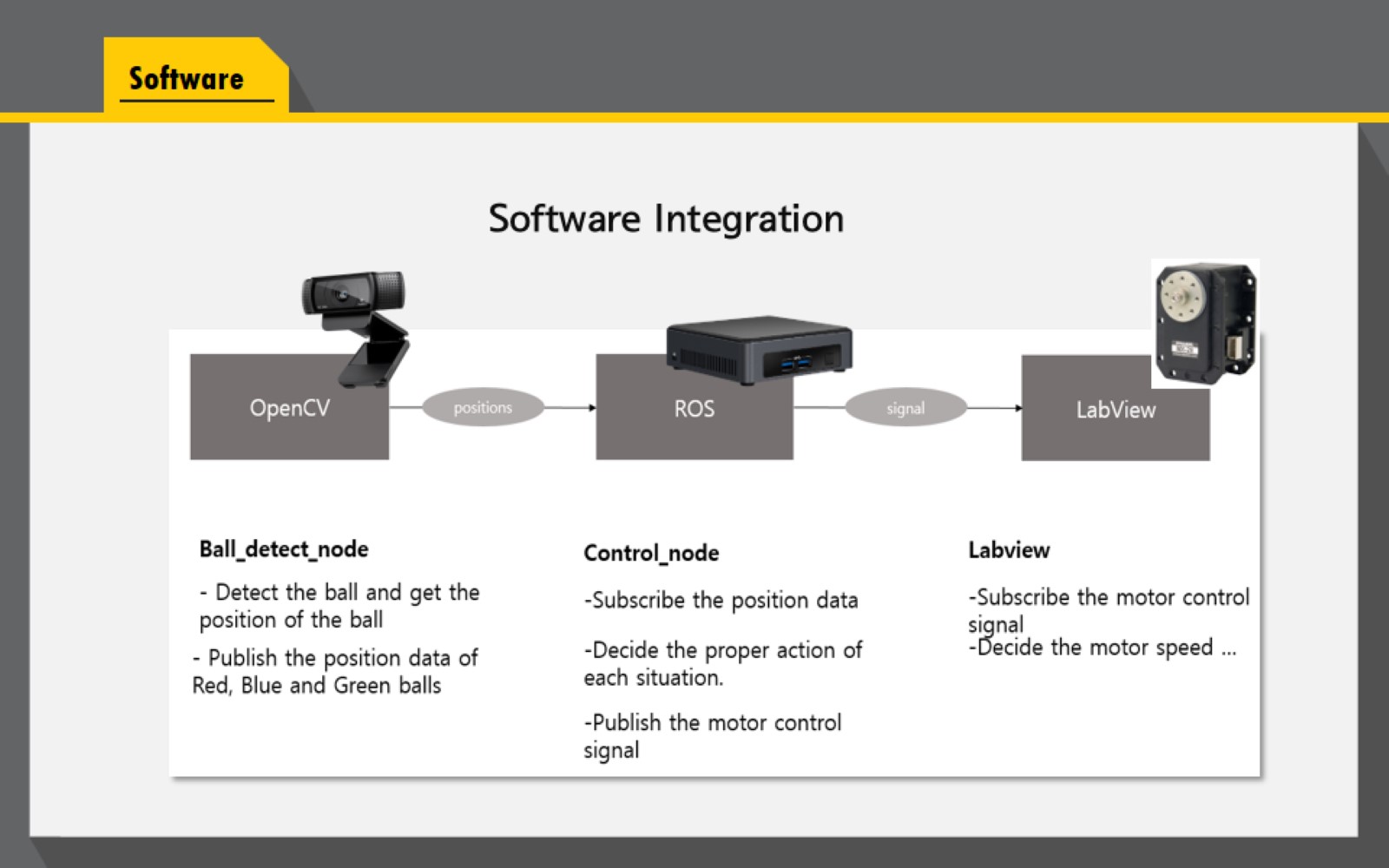
****

Figure 11 Software Integration

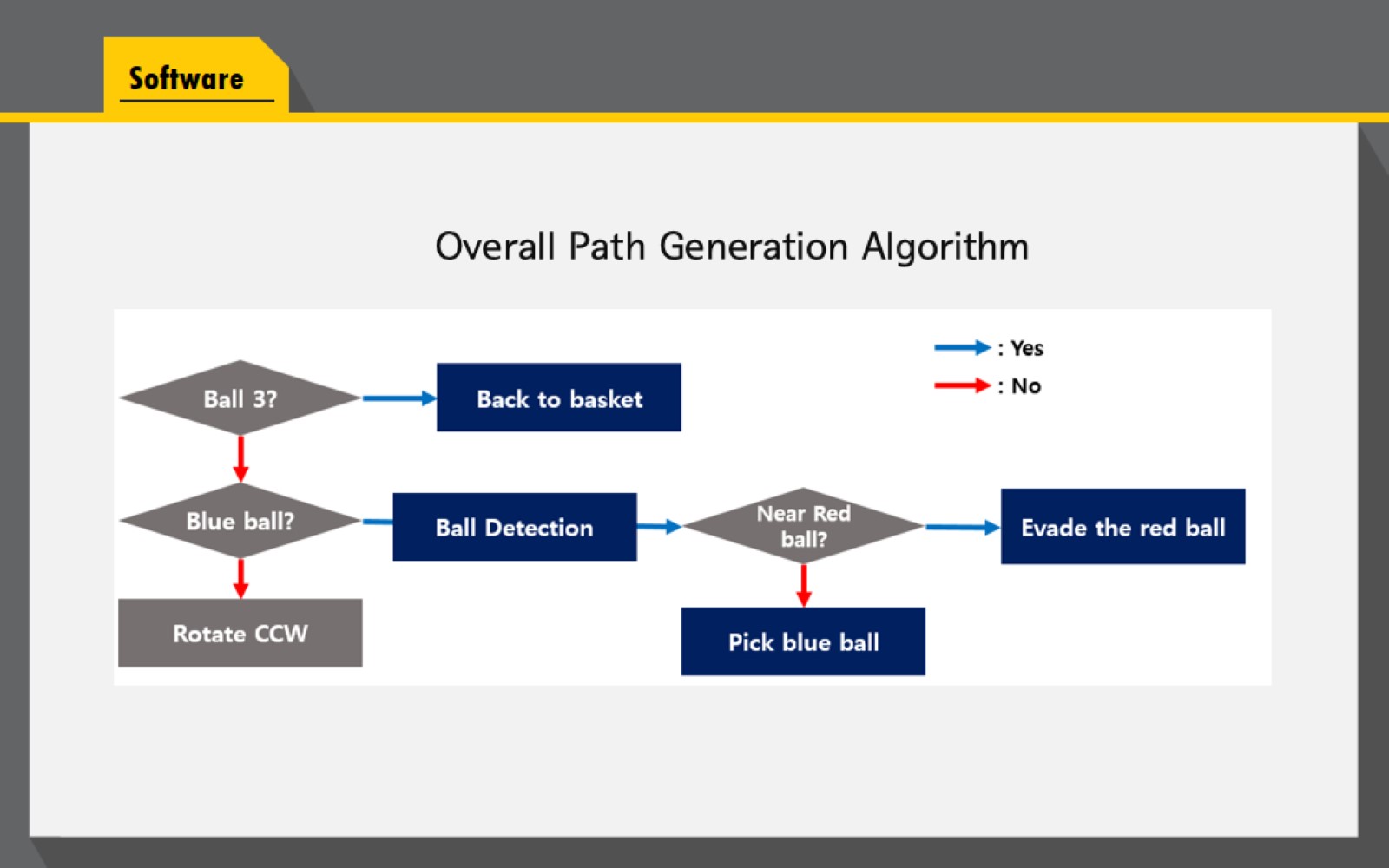
1. Overall Algorithm  
   

Figure 12 Overall Algorithm

전체 알고리즘은 다음과 같다. 공을 인식하고, 공에 접근하고, 파란 공이 가까워지면 차체와 파란 공 사이에 있는 빨간 공을 피하고, 공을 줍고, 공을 세번 다 주우면 바구니로 돌아간다.

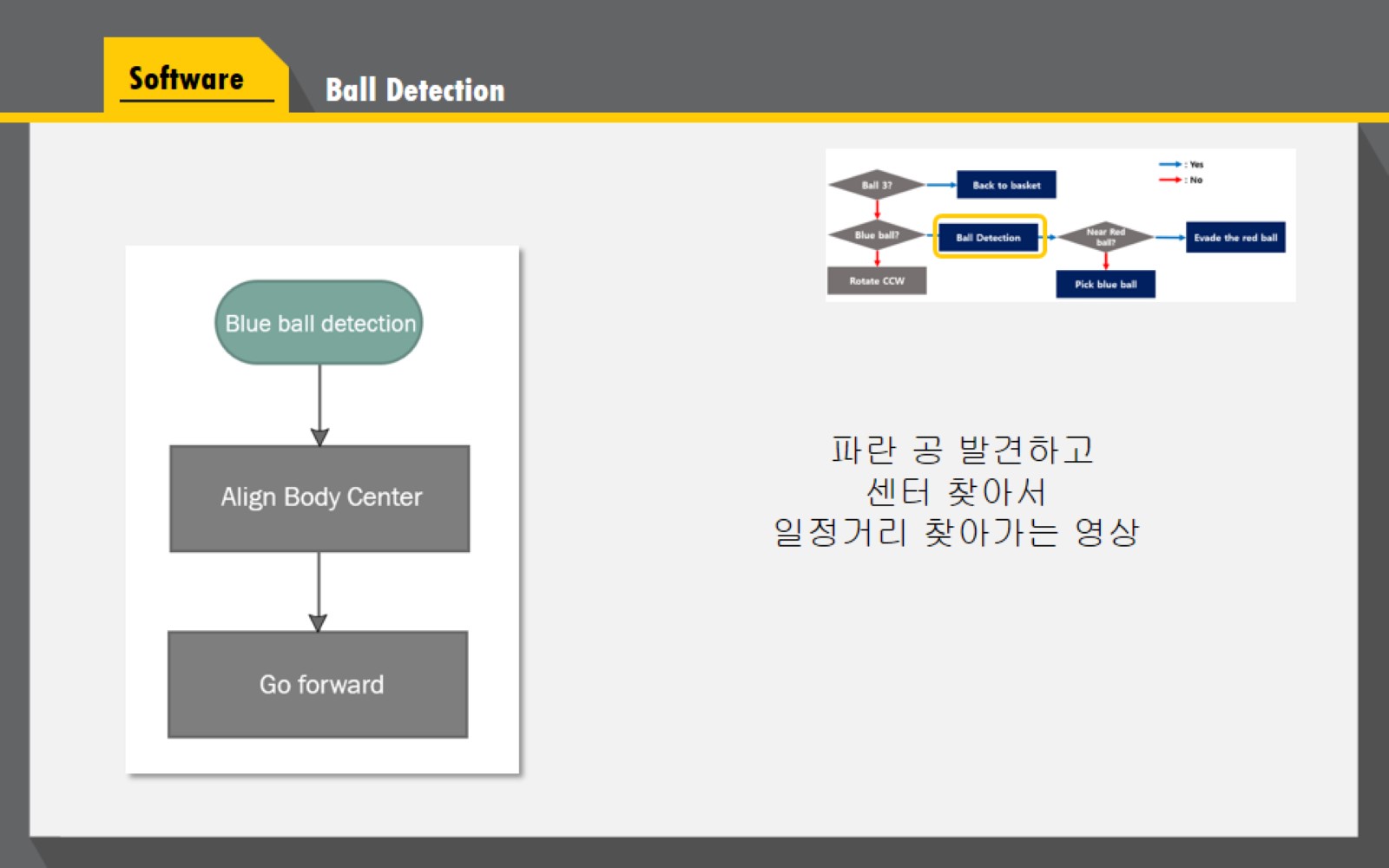
1. Ball detection  
   

Figure 13 Blue ball detection

파란 공을 찾을 때까지 제자리 회전을 하다가 파란 공을 발견하면 차체가 파란 공을 바라보도록 한다. 이때 파란 공과 웹 캠의 각도를 활용하여 feedback control을 하는 방식으로 속도와 안정성을 잡을 수 있었다. 파란 공을 바라본 다음엔 파란 공을 줍는 모션을 실행하기 전까지 파란 공을 따라 직진을 하는데, 그 과정에서 빨간 공을 만날 경우 빨간 공이 있는 반대 방향 대각선으로 움직여 회피하는 방식을 사용하였다.

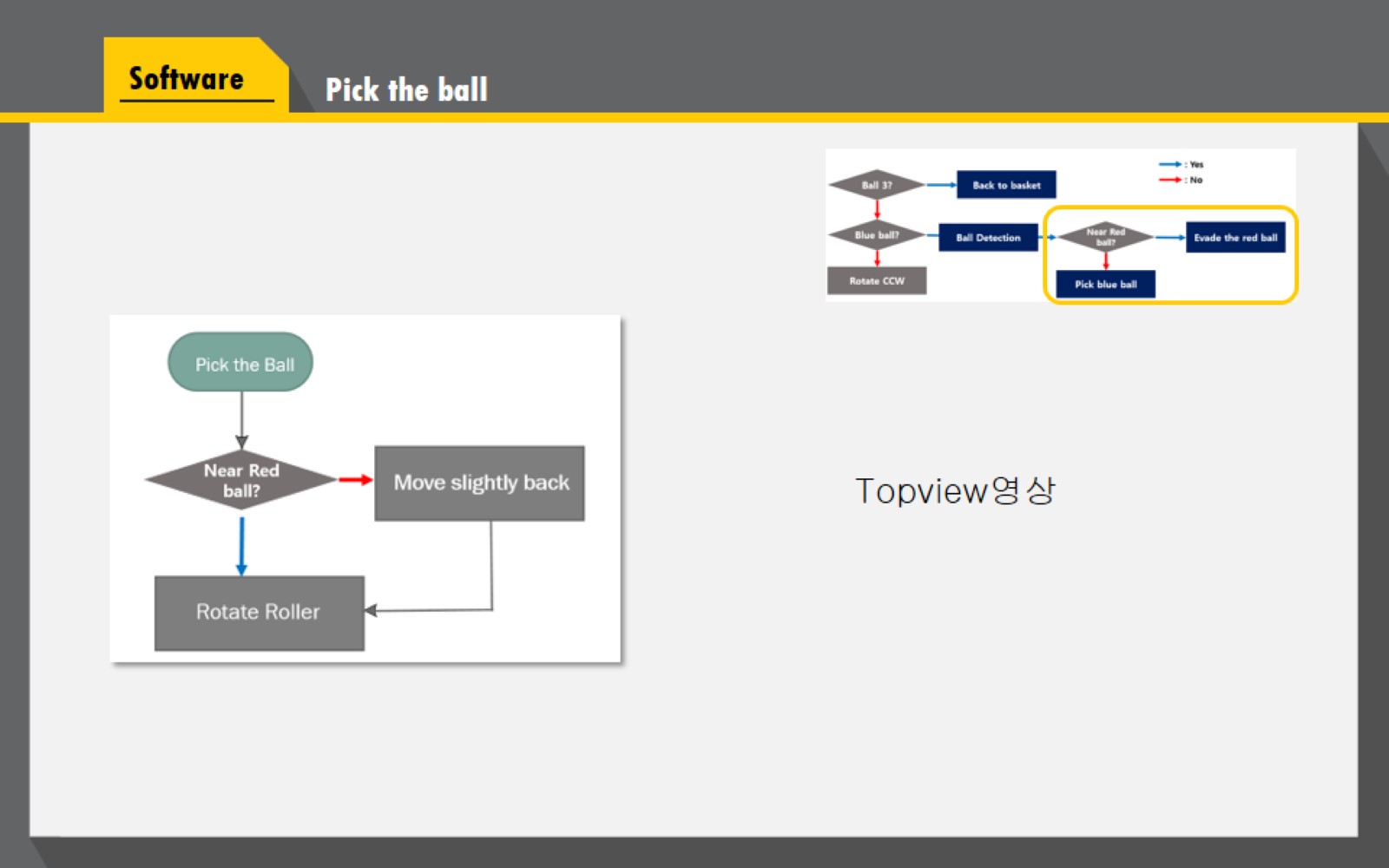
1. Pick the ball  
   

Figure 14 pick the ball

파란 공을 항해 가다가 파란 공이 일정 거리 내로 들어오면(~70cm) pick up 모션이 실행된다. 이 때는 이동 속도를 늦춰 가다가 일정 시간 후부터 롤러가 돌기 시작한다. 만약 파란 공 뒤에 빨간 공이 있다면 빨간 공을 잡지 않기 위해 살짝 뒤로 가는 움직임을 추가한다.

1. Back to the basket  
   

Figure 15 Back to the Basket

3번의 picking을 하면 다시 바구니로 돌아가는 알고리즘을 실행한다. 이 때의 function인 Back to basket 는 크게 4개의 부분으로 나누어 볼 수 있는데, 첫번째는 일단 제자리에서 공을 찾을 때까지 회전 한 후 blue ball detection 에서 사용한 알고리즘과 동일한 알고리즘으로 녹색 공을 따라 일정거리(~100cm) 까지 이동한다. 두번째로는 대각선 이동을 통해 두 녹색 공 사이의 수직이등분선 위로 차체를 이동시킨다. 세번째는 수직이등분선 위에서 두 녹색 공의 중심까지 일정 거리를 이동 한 후 두 공의 z좌표를 맞추어 차체와 바구니가 평행하도록 한다. 마지막으로 180도를 회전하고 뒤로 이동하여 바구니와 차체를 밀착시킨 후 후방 모터를 작동시켜 공을 내놓는다.

**3. Process**

1. Monthly Plan  
   3월은 각자 파트를 공부하고 익히는데 중점을 두고 dynamixel이 도착한 4월, 5월에 본격적으로 프로젝트를 계획하고 진행하였다.  
   

Figure 16 Plan in April

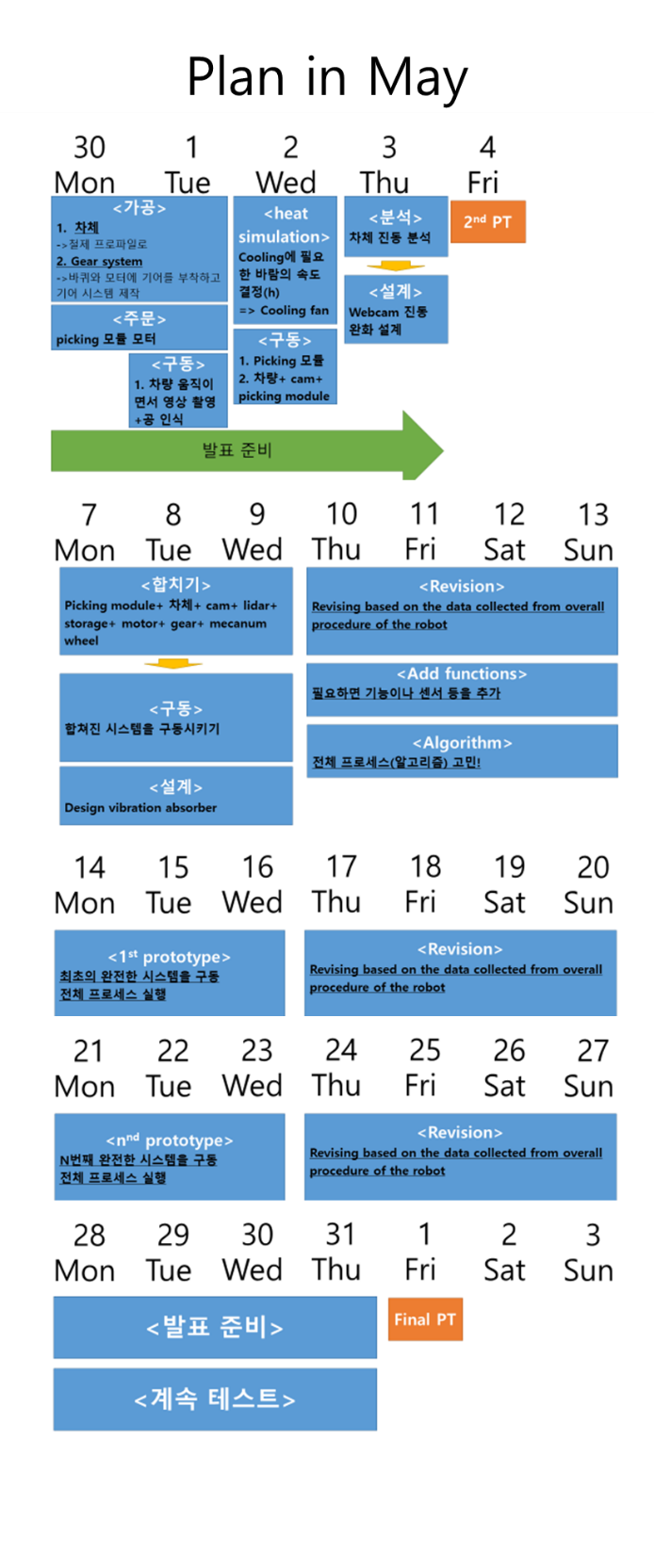


Figure 17 Plan in May

1. ROS  
   1주-2주차

Ros는 무엇인가, ros가 어떤 역할을 하게 되는지를 학습하는 시간을 가지고 컴퓨터에 우분투, ros를 설치하고 실행해 보는 과정을 거쳤다. 어려운 과정은 아니지만, 리눅스 환경에 익숙하지 않았기 때문에 다소 어려움을 겪었다.

3주차

과제를 같은 조 ROS 담당 학생(홍성웅) 과 같이 해결하였는데, 문제를 해결 하면서 topic, service가 어떻게 작동하는지 실제로 볼 수 있었으며, 여러 개의 node의 묶음을 편리하게 한번에 실행시켜주는 roslaunch 를 어떻게 사용하는지 배울 수 있었다.

금요일 session에서는 조교님들이 구현해 놓으신 기본 소스코드를 받고, 그 코드들이 어떻게 작동하는지를 배우는 과정을 거쳤다. 그 과정에서 webcam, lidar 가 어떻게 사용되는지 실제로 볼 수 있었고, xbox controller 를 통해 조종 데이터를 수신하고, tcp/ip서버를 구성하여 송신하는 방법에 대해 배울 수 있었다.

4주차

ROS파트 에서는 저번 주에 받은 코드들을 읽어보고, 실제로 xbox controller, lidar를 연결하여 사용해 보는 과정을 거쳤다. ROS 파트의 특성상 전체 파트가 구성되기 전까지는 주도적으로 역할을 하지 않기 때문에 받은 코드들에 대해 인지하고, 실제 사용할 때는 어떤 부분을 수정하고 사용해야 할 지에 대해서 공부하는 과정을 중심으로 하고, nuc와 개인 노트북을 어떻게 연결하여 사용하는지에 대해서도 공부했다. 또한 주어진 코드가 대부분 C++이고, 속도, 리소스 측면에서도 interpreter인 python보다는 C++이 유리하므로 ROS에서 메인으로 사용하게 될 C++에 대해서 공부를 하여 과목 후반부에 ROS에 필요할 프로그래밍 실력을 미리 키워 놓을 수 있었다.

5주차

전반적으로 하드웨어 파트에 집중한 주였다. ROS파트에서는 LIDAR와 WEBCAM을 사용하는 코드들을 읽어보면서 어떻게 작동하는지 알아보고 해당 코드를 다른 파트와 어떻게 통합시켜 발전시켜야 할 지에 대해서 생각 해보는 시간을 가졌다.

6주차

해당 주에 시험이 많이 있는 관계로 많은 내용을 커버할 순 없었고, 창시구 소프트웨어 단체 톡방에 올라온 질문들을 확인해보고 그 질문의 답변들을 확인하면서 어떤 문제가 있을 수 있고 어떻게 해결하면 되는지를 알아 볼 수 있었다

7주차

Ros파트에서 transfer function에 대한 강의가 있었는데 시간상 참여하지 못했기 때문에 시험이 끝난 후 해당 강의 자료를 보며 복습 하는 과정을 거쳤고, transfer function을 어떻게 사용할 지에 대한 정보를 알 수 있게 되었다.

다만, 현재 우리의 시스템에서 rplidar가 하는 역할이 매우 적고, 실질적으로 매핑이 쉽지 않은 만큼 transfer function이 과연 정말로 필요할지, 실제로는 어떻게 사용할지에 대한 생각이 필요할 것으로 예상된다. 만약 rplidar가 필요 없다면, transfer function은 사용할 필요가 없거나, 사용하더라도 웹 캠과 플랫폼의 중심 위치를 보정해 주는 역할로 바뀌게 될 것이기 때문에 우선 빠르게 웹 캠을 이용한 ball detection을 하고, 이 데이터를 활용하여 공들을 주워 오는 것을 해 보아야 전체적인 시스템 구현의 그림이 바뀌기 때문에 최우선적으로 해야 할 목표라고 할 수 있다.

따라서, 주어진 ball detection node가 실제로 어떻게 작동하는지 확인 하고, 코드가 어떻게 쓰여 있는지를 공부하여 opencv와의 통합을 쉽게 하도록 하였다.

8주차

ros파트에서는 첫째로 tcp ip통신을 my rio와 하는 것을 랩뷰파트와 함께 하여, 모터를 실제로 xbox 컨트롤러를 통해 조종할 수 있었다.

또한, opencv 파트의 ball detection 에 관련된 코드를 ball detection node에 통합시키고 data show node에 결과를 보이도록 하는 과정을 거쳤다. Opencv 에서 구현한 코드와 주어진 ball detection node의 코드가 상당히 달라서 그 차이를 이해하고 통합하는데 많은 노력을 해야 했다. 이 과정에서 webcam node를 사용하는 것 대신 ball detection node에 통합시켰고, 메시지에 푸른 공, 붉은 공에 대한 데이터를 모두 보내주도록 수정했다.

결과적으로 ball detection node를 열면 웹캠으로부터 이미지 데이터를 받아 data show 와 data integration쪽으로 인식된 공들의 데이터를 publish해주는 역할을 하도록 할 수 있었다.

또, 전체적 process를 노드/토픽 단위로 다음과 같은 그림으로 만들어 보는 과정을 거치며 실제로 각각의 노드들이 어떤 일을 할 것이며 해당 노드들을 어떻게 구현해야 할지를 더 구체적으로 알아 볼 수 있었다.

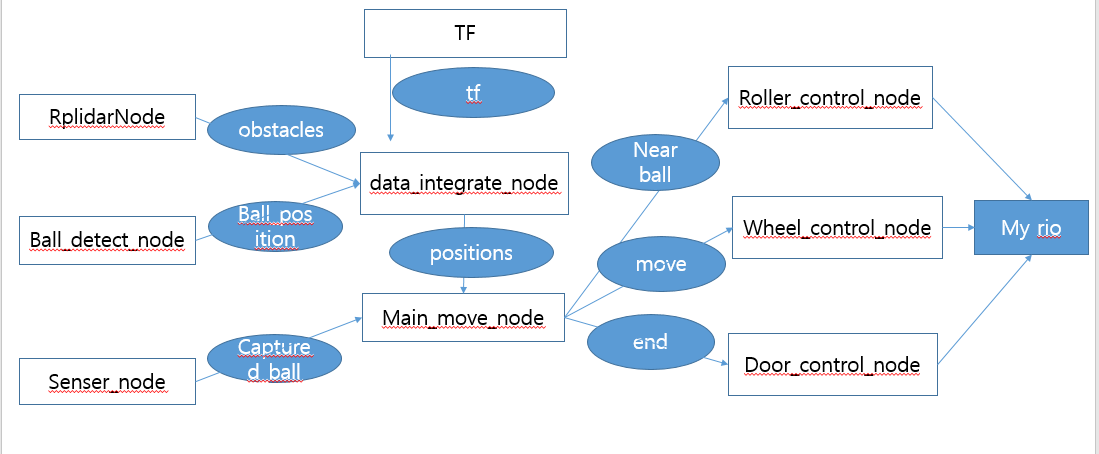


Figure 18 algorithm prototype

9주차

해당 주차에는 control node를 생성하여, 간단한 수준의 자율 주행을 하는 것에 중심을 뒀다. 주어진 data integration node 를 기본으로 하여, 웹 캠에서 온 공의 좌표 값에 따라 플랫폼이 취해야 할 행동을 결정하고, my rio에 xbox control node에서 수행했던 것처럼 값을 넣어주는 역할을 하도록 했다.

구체적으로 알고리즘을 설명하자면, 파란 공을 발견하지 못했을 땐, 파란 공을 찾을 때까지 제자리에서 회전하고, 파란 공을 발견하면, 빨간 공을 피해야 하는지 확인을 하고 피해야 한다면, 빨간 공이 있는 반대쪽 대각선으로 이동한다. 빨간 공을 피하지 않아도 된다면, 파란 공이 플랫폼의 정면에 있도록 회전을 한 후, 파란 공 쪽으로 직진한다. 마지막으로, 파란 공을 향해 직진 하다가 가까워지면 일정시간동안 계속 직진하면서, 롤러의 모터를 돌려 공을 잡는다.

실제 작동시킨 결과, 전반적으로 생각했던 것 보다도 잘 작동했지만 몇가지 문제점이 있었다. 첫째로 너무 빠르게 회전해서 공이 정면에 오기 힘든 문제가 있었는데, 간단한 P컨트롤을 사용하고 기존에는 정면에 오는 기준을 Y좌표를 통해 계산했는데, 이렇게 계산하면, 너무 먼 거리에 있는 공은 약간의 회전만으로 많은 Y값이 변하므로, 각도 조건을 추가하여 둘 중 하나만 만족하면 되도록 하였다. 다만 웹 캠으로 공을 인식하고 가는 방식의 특성상 공이 너무 가까워져 시야를 벗어났을 때 생기는 문제는 해결하지 못했다.

10주차

Rplidar 사용에 있어서는 일단 SLAM을 전반적으로 구현하기 힘들고, 공 이외의 다른 장애물이 없는 상황에서 특별히 중요한 역할을 하긴 힘들 것으로 보인다. 다만 벽을 인식하여, 벽을 피하는 용도, 마지막에 공을 모두 집고 돌아온 후 공을 바구니에 넣을 때 공이 뒤로 나오게 되므로 공을 놓기 위해 180도 회전할 때 초록 공이 보이지 않기 때문에 벽을 통해 기준 축을 정하는 용도, 공을 잡는 과정에 문제가 있을 때, 다시 대략적으로 맵 가운데로 돌아와 다시 행동을 수행하도록 하는 용도 등 부가적인 요소로는 사용하면 좋을 것 같다는 생각을 했다.

또, 저번 주에 문제가 있었던 회전속도를 조정하는 시간을 가졌는데 간단한 P컨트롤을 통해 조절을 해 준 결과 회전속도가 충분히 빨라도 된다는 결론을 내리고 현재의 빠른 속도를 유지하기로 했다.

11주차

전반적인 설계가 끝나고 rplidar 사용을 위해 rplidar를 활용하는 방안에 대하여 생각해 보았다. Hector mapping, gmapping 등의 package를 활용하여 rough한 mapping을 활용할 순 있지만, 가속도계나 encoder가 없는 상황에서 이 mapping이 큰 의미를 갖기 힘들다는 생각을 했고, rplidar의 raw data인 각도와 거리를 활용하는 방식에 대해서 생각을 해 보았고, 바구니나 공을 인식하여 결과를 얻거나, 벽을 인식하여 align하는 방안이 있었으나, 결국 웹 캠을 활용하여 조종하는 것에 비해 큰 의미를 갖지 못한다고 판단이 되었으며, 결국 rplidar를 설계에서 제외하는 방안으로 가기로 했다.

그 외의 ros파트에서는 일단 데모가 진행되는 과정을 지켜본 결과 실제로 공 3개를 안정적으로 잡기 위해 여러가지 값들 예를 들어 빨강색 공을 장애물로 인식하는 범위나 파란 공 앞에서 롤러를 돌리기 시작하는 거리 등 구체적인 값들을 튜닝하여 안정적으로 공을 잡을 수 있도록 했고, 초록색 공을 인식하여 picking이 끝나고 난 뒤 다시 바구니로 돌아와 공을 놓는 방법에 대해 생각을 해 보았는데, 일단 2개의 초록색 공 사이를 목표로 두고 이동한 후 두개의 공의 Z좌표가 같아지도록 회전을 하여 바구니와 수평이 되도록 align을 한 후, 180도 회전이 되는 시간을 측정하여 그 시간 동안 회전하도록 명령을 주어 완벽하게 바구니 앞에 멈추어 서는 것을 목표로 코딩을 하고 있다.

12주차

이번주는 데모 전 주인 만큼 최대한 모든 할 일들을 미리 끝마치려 했으나 시스템적인 문제가 너무 많아 어려움이 있었다. 일단 모터가 계속 인식이 안되어서 롤러 부분이나 뒷문 쪽이 제대로 돌아가지 않는 문제가 있었고, 초록 공을 활용하여 align을 하려고 원래 방식보다 이상적인 위치로 이동하게끔 두 공의 수직 이등분선 위로 플랫폼을 이동 시키고, 거기서 두 공 사이를 보고 이동하도록 하는 방식을 고안하여 보았다.

초록색 공의 인식이 튀어서 공 하나가 같은 위치에 2번 찍히는 문제점이 있었는데 그 문제가 화면상에서는 잘 보이지 않았고, 그래서 그 문제를 잘 모른 채 코드 문제로 알고 수정하느라 시간을 낭비하였다. 이 문제를 찾아낸 후 open cv 파트에서 같은 위치에 있는 공의 인식을 없애는 코드를 넣었고 ros파트에서 혹시 여러 개의 공이 인식이 되어도 상관 없게끔 2개의 공을 사용할 때, 원래는 0번 1번공을 활용하여 놓던 것을 가장 왼쪽 오른쪽의 공을 활용하도록 코드를 바꾸어서 문제를 해결하였다.

대각선 움직임 자체가 전체 움직임 중 일부분에 불과 하다는 것을 고려하였을 때, 완벽하진 않지만 충분히 괜찮다고 볼 수 있고, 최대한 코드를 짜는 과정에 있어서 대각선 이동을 배제하고 대각선 이동을 한다면, 오래 가만히 있으면 탈출하는 (전후 좌우로 움직이는) 방식을 사용한다면 문제를 해결 할 수 있다는 결론에 이르렀다.

13주차

데모 전 마지막으로 여러가지 수치들(거리, 시간 값들)을 튜닝하는 것을 목적으로 두고 반복하여 보았다. 이론적인 값과 실제 그 값들을 적용하였을 때 작동하는 것을 보면서 최대한 높은 가능성으로 성공 하도록 코드를 튜닝하는 것을 중심으로 삼았다.