1. Group meeting의 전체적인 진행 과정

초기에는 Evaluation criteria가 명확하지 않고, battery, lidar, motor등의 주요 부품들도 아직 도달하기 전이라 구체적인 계획 수립이 불가능했다. 각 sensor들의 physical dimension 및 spec등을 알아야만 차체의 design, mission 수행을 위한 algorithm 계획 등이 가능했지만, 여건상 그러지 못했고 초반에는 진행이 거의 이루어지지 않았다. 되돌아보면 이것이 학기말에 업무량이 상당히 치중된 원인이라 생각된다.

따라서 초반에는 추상적인 업무를 진행할 수 밖에 없었다. 추상적으로 전체적인 task들을 예상해보고, brainstorming을 거치며 project definition을 했다. 교수님과의 첫 meeting때는 Gantt chart의 구체화 및 합리적인 decision 과정, project의 진행 방향에 대한 조언 등을 얻었다. 또한 Mechanical design 수업에서 배운 design process에 기반을 두고, problem definition, project planning, product definition, concept generation, concept evaluation의 과정을 부분적으로 진행하고, 각자의 업무 part에 따라 task들을 배분했다. 나는 concept generation & Evaluation과정에 초점을 맞추어 진행했고, 구체적으로는 TRIZ를 통한 idea generation 및 heat transfer & ball collecting에 대한 이론적, 수치적인 접근, 즉 analytic analysis를 진행했다. Prerequisite으로 수강했던 dynamic, system modeling and control등을 바탕으로, 다양한 alternative들에 대한 energy, time consumption을 이론적으로 계산해보고, 결과값을 바탕으로 decision matrix를 만들고 prototype의 concept을 결정했다.

1) Gripper

Energy consumption by vertical motion of arm

$$E = \frac{2R_1}{{K_{T,1}}^2}\left[\frac{2{I_{tot}}^2{\theta_{f,1}}^2}{t_{1,1}{t_{1,2}}^2} + \frac{\left(m_{tot}g x_{cm,tot}\right)^2}{\omega_{max,1}}\left(\frac{1}{2}\theta_{f,1} - \frac{1}{4}sin\left(2\theta_{f,1}\right)\right)\right]$$

$$+\frac{R_2}{{K_{T,2}}^2}\left(\frac{2{I_{grip}}^2{\theta_{f,2}}^2}{t_{2,1}{t_{2,2}}^2}\right) + \frac{R_2}{{K_{T,2}}^2}\left(\frac{m_{ball}g}{2\mu x_{grip}}\right)^2 \Delta t_{grip}$$

Energy consumption by horizontal motion of gripper

Energy consumption by holding ball

2) Sweeper

Energy consumption by sweep ball

$$E = \frac{R}{{K_T}^2}\left(\frac{2I^2{\theta_f}^2}{t_1{t_2}^2} + \frac{{m_{ball}}^2(1.2gh)}{{x_{sweep}}^2}\frac{1}{\Delta t_{sweep}}\right)$$

Energy consumption by rotational motion of sweeper

3) Vacuuming

$$E = P_{vacuum}\Delta t_{suction}$$

4) Sticky device

$$E = 0$$

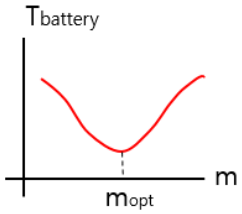
(Figure1. Analytic approach about energy consumption of the alternatives)

$$E = \frac{R}{K_T{}^2}(\mu m g r_{wheel})^2 t_{wheel,oper} + P_{fan}(m)\ t_{fan,oper} + E_{collect\ ball}$$

$$q_{battery} = f(E)\ \text{(increasing function)}$$

$$T_{battery} = T_\infty + \frac{q_{battery}}{\eta_o h A_t}$$

Let assume, h(convection heat transfer coefficient is function of system mass (cause fan,fin mass increases, then h increases)



$$\left.\frac{dT_{battery}}{dm}\right|_{m_{opt}} = 0$$

$$m_{cooler,opt} = m_{opt} - m_{sys,\,w/ocooler}$$

(Figure2. Analytic approach about relationship between heat transfer and total mass)

1) If we use many motor, system reliability is good but Energy, Complexity is High.

-Reliability & Energy

11. Beforehand Cushioning
19. Periodic Action — Rotating sweeper
21. Skipping
27. Cheat Short - Living

-Reliability & Complexity

1. Segmentation
13. The other way around — Sticky device & Vacuum
35. Parameter Changes

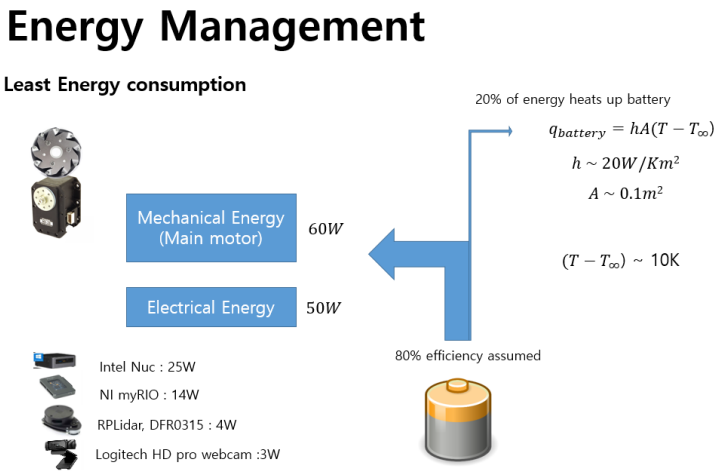
2) Collecting ball is easy when collector area is large, but energy increases

-Area of moving object & Energy

9. Preliminary Action
19. Periodic Action
29. Pneumatic and Hydraulic
32. Color Changes

(Figure3. Results of TRIZ)

그 후 각자 실시한 task들을 취합하고, 서로 feedback을 주고 받으며 발전시키는 시간을 가지고, 이를 바탕으로 1st design review를 위한 presentation flow 및 contents를 제작했다. 또한 지금까지 해온 task들을 교수님께 feedback을 받았다. 주요 feedback으로는 1) Analytic analysis의 수식이 너무 복잡해 의미가 없으므로 간략하게 하고 의미 있는 결과값 만을 강조하기, 2) Heat transfer 및 vibration reduction을 무작정 고려할 것이 아닌, 전체적인 system에 대한 이해를 바탕으로 그 필요성을 결정하는 과정을 진행하며, 이 때 간단한 실험이나 수식을 통한 계산을 바탕으로 결정하기. 3) decision matrix의 criteria와 weight를 조금씩 바꾸기. 예를 들어서 collecting system을 공이 앞에 있는 special case에만 사용되므로 total energy consumption에 비해 그 비율이 적으므로 크게 고려할 부분이 아니라 weight를 줄이는게 좋을 것이며, 오히려 reliability등이 훨씬 중요할 것이라 조언해주셨다. 4) 전체 system에 사용되는 motor, sensor, electric device들을 파악하고 이들의 관계를 정리해 본 뒤, 전체적인 energy consumption을 수학적으로 계산해보고 각 element들의 fraction을 파악하기 등이 있었다. 따라서 우리는 추후 meeting 때 vibration reduction 및 energy management를 중점적으로 고려해보았다. 나는 sensor, device등의 catalog를 분석하고, motor에서 소모되는 energy등을 계산함으로써 system의 energy flow를 체계화했다. 또한 이를 바탕으로 발열량을 계산했고, 약 10도 정도의 온도 증가가 있을 것이라 예상했다.



(Figure4. Schematic of energy management)

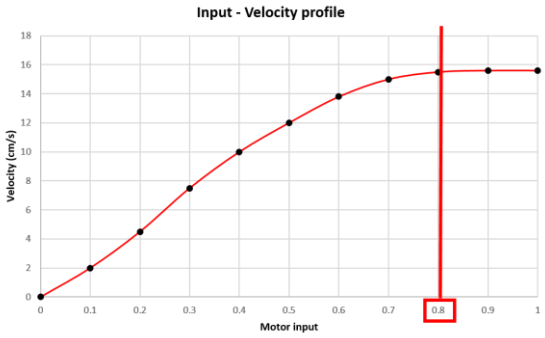
4 월 3 일, 드디어 battery 와 dynamic cell 을 받았다. 교수님께 1) Open CV 를 통해 공과의 거리를 측정하는 방법, 2) webcam 및 lidar 의 최적 위치, 3) I beam 또는 알루미늄 등을 통한 빠른 골격 제작 및 방법 등에 대해 조언을 받았다. 구체적인 조언 내용으로는 1) webcam 으로 공의 테두리를 sensing 하여 크기를 통해 거리를 측정하는 것은 너무 부정확하므로 공의 중심을 찾고, webcam 의 centerline 과 닮음비를 통해 거리를 측정하는 방법을 이용해라, 2) webcam 같은 경우는 높을수록 시야각이 넓어지고, 두 개의 공을 구분할 수 있는 능력이 증가하고, 이를 통해 거리 측정 등에 요긴하게 사용할 수 있으므로 높게 위치시키는 것이 유리하다. Lidar 를 이용하는 방법 및 배치하는 위치는 lidar 를 통한 추가 실험 및 더욱 심도 있는 논의가 필요해 보였다. 저번 주에 lidar 를 통해 간단한 실험을 해본 결과, 공이 매우 작은 점 1~2 개 정도로 detecting 되어 lidar 를 통해 공과의 거리를 측정하는 과정 및 정확도에 의문을 가졌다. 따라서 더욱 정밀한 실험을 통해 공의 거리 측정 가능 여부 및 이를 통한 slam 및 path planning 의 실현가능성을 파악해보아야 했으며, 만약 불가능이라는 결론을 얻었을 경우, 공의 위치가 아닌 벽면을 sensing 하여 SLAM 및 위치 파악, 골대로의 귀환 등을 수행하자는 결론을 도출했다. 3) 교수님께서는 중간 고사라 시간이 부족하고, 2 차 발표까지 시간도 얼마 남지 않았으므로 최대한 빨리 prototype 을 만들어보라고 조언해주셨다. 제일 중요한 부분은 바퀴가 고정될 차체로써, 이를 플라스틱 등 연약한 재질을 사용하면 약 5kg 의 차체 무게를 고려했을 때 뒤틀림 등이 발생하여 추후 여러 문제가 발생할 수 있으므로, 이는 무게적 손실을 보더라도 튼튼한 철제로 제작하는 것이 좋다고 하셨다. 또한 현재 사용하는 4) 2D Lidar 에 motor 를 연결하여 nodding 시킴으로써 3D space 에 대한 visualization 및 sensing 하는 아이디어가 나왔으며, 추후 진행 상황을 보고 이를 추가시켜 봄직했다. 모든 팀원들과 함께 공을 집는 collecting 방법, 공을 저장하고 release 하는 방법, 차체의 physical dimension 및 최소화 방법 등을 최종적으로 결정했고, 이를 바탕으로 각자 업무를 분담했다. 우리 ROS 팀의 역할은 Lidar 를 통한 data acquisition 방법 연구, 실제 실험을 통한 최적의 lidar 위치 및 분해능 등을 파악, 최소 및 최대 탐지 거리 측정 등이었다.

교수님에게 SLAM 에 대한 의미 있는 조언을 많이 들었다. 처음에는 SLAM 의 필요성에 의구심을 제시하셨지만, 결국 골대를 찾아오기 위해서는 의미 있다고 하셨다. 하지만 이를 위해서는 여러 sensor 들의 구체적인 spec 등을 확인해 볼 필요가 있고, 따라서 다음 meeting 때 이를 진행하기로 했다. 또한 pose_estimation 등을 쉽게 하기 위해서 나침반을 구입하여 활용 가능한 sensor data 의 수를 늘리라는 조언을 해주셨다.

4 월 24 일, 교수님께서 창시구실에 방문하셔서, 1) x-box controller 를 통한 motor 구동, 전진, 측면 이동, 회전 등의 진행 현황, 2) open-cv 를 통한 ball_detection 의 accuracy, 3) rp-lidar 의 성능 확인 등을 진행하셨다. 개별 part 의 진행은 만족하셨고, 빠르게 integration process 로 들어가라 하셨다. 또한 sensor 의 성능을 확인하시고, 초기 위치에서 모든 공의 sensing 이 불가능하므로, ball collection 과정 이전에 공들의 위치를 파악하는 additional process 의 필요성을 언급하셨다. 또한 실험을 통해 motor input voltage 에 대한 velocity profile 을 작성하라고 하셨다.

이렇게 얻은 velocity profile 은 굉장히 유용하게 사용되었다. 첫번째로, ball collecting 과정에서 공에 충분히 가깝게 접근하여 webcam 의 시야각을 벗어났을 때는 어떠한 sensor 정보도 없이 일정한 거리를 진행해야 한다. 이 때의 적절한 motor input 은 이 profile 을 기반으로 결정했다. 또한 motor input 가 velocity 가 이론대로 linear 한 관계가 아닌 input 이 증가함에 따라 velocity 의 증가량이 감소하는 경향을 보였고, input 이 0.8 이상일 때는 속도가 거의 증가하지 않았다. 만약

motor input 이 2 배가 되면, 전력 소모량은 4 배가 되므로 최대한 motor input 을 줄이는 것이 energy consumption 을 줄이고 heat dissipation 을 감소시키는 측면에서는 유리하지만 속도가 줄어드므로 mission 수행 시간이 늘어나는 단점이 있다. 이러한 trade-off 관계에서 optimal motor input 을 찾는 것은 중요했고, 우리는 input-velocity profile 을 기반으로 0.8 의 motor input 을 이용했다.



(Figure5. Motor input-velocity profile)

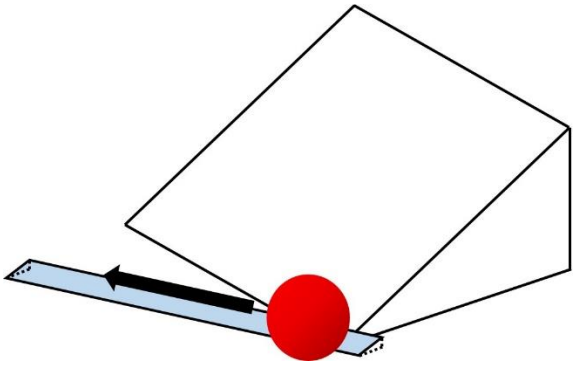
5 월 4 일 있는 Design Review2 를 중점으로 조모임을 진행했다. 발표 전까지 1) ball 을 발견하고, ball 의 중심과 로봇의 중심 align, 2) ball collecting 에 적합한 거리까지 직진, 3) sweeper 작동을 통한 ball collecting 까지를 목표로 했으며, 이 과정에서 발생되는 문제점들을 해결해 나갔다. 또한 외부 전력 공급 없이 독립된 차체를 만들기 위한 작업을 진행했다.

다음 교수님과의 meeting 날, 분명히 잘 작동되던 로봇이 갑자기 작동하지 않아서 크게 꾸중을 들었다. 교수님께서는 이러한 연습 기간에 10 번에 9 번 꼴로 성공한다면, 훨씬 변수가 많은 실전에서는 거의 실패한다고 봐도 무방하다고 하시며 테스트 시 성공률을 100 프로에 가깝게 해야 한다고 하셨다. 또한 hardware 제작보다 software 의 debugging 이 훨씬 많은 시간이 필요하고, 중요하고 조언해 주셨고, 우리는 최대한 빨리 시스템을 완성시킨 뒤 debugging 에 집중하기로 했다.

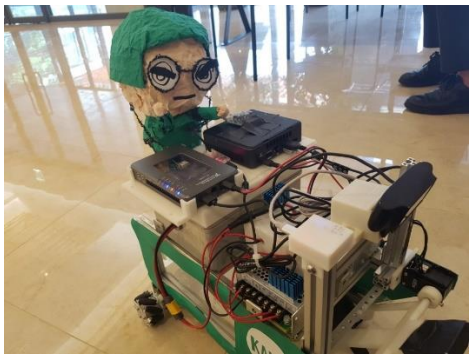
5 월 15 일 창시구실에서 교수님과의 meeting 이 있었다. 당장에 기계동 공동강의실에서 진행될 field test 에서 얻어야 할 data 및 결과들에 대해 토의해보는 시간을 가졌다. 교수님께서는 단순히 field test 과정에서 제대로 동작할 때까지 parameter 들을 calibration 하는 것을 의미가 없으며, data 들을 logging 하여 정확한 문제 원인 파악 과정이 필요하다고 조언해주셨다. 따라서 우리는 rosbag record -a 기능을 이용하여, 모든 test 과정에서의 ball_position_data, xbox_ctrl data 등을 저장하고, test 과정에서 발생한 문제를 data 관점에서 파악해보는 시도를 진행했다.

저번 주 진행한 ball collecting process 의 reliability 를 증가시킨 결과를 교수님께 시연했다. 그러나 시연 과정 중, 차량이 파란 공까지는 정확히 이동했으나, collecting process 가 가끔씩 제대로 작동하지 않는 문제가 발생했다. 우리는 이러한 이유가 sweeper 의 motor torque 에 비해 차량 전면부에 위치된 빗면의 경사가 너무 가파르고, sweeping 각도 또한 문제가 있다고 판단하여 hardware 적인 design 의 변경이 필요함을 느꼈다. 따라서 solid work team 에서 개선을 진행했다. 교수님께서는 혹시라도 공 3 개를 다 못 줍는 경우가 발생했을 때 최소 점수를 확보하기 위 대한 대비책도 강구해야 한다고 말씀하셨다. 따라서 ros 의 timer 기능을 이용해 3 분 동안 귀환하거나 공 3 개를 다 줍지 못하면 강제로 귀환 code 를 작동 시키는 등의 algorithm 에 대해 생각해보았다. 또한 최종 발표 시 강조 할만한 우리 팀 만의 창의적인 요소들을

찾아보았다. 첫 번째는 converter 에서 발생하는 열을 감소시키기 위해 대용량 converter 를 사용한 것이고, 이를 통해 실제로 발열량을 크게 줄여 10 분 정도 가동시키더라도 converter 가 실온과 큰 차이가 없음을 확인할 수 있었다. 두 번째는 sweeper 에 counter mass 를 인가한 것이다. 차량이 움직이는 내내 motor torque 를 통해 sweeper 를 open 시키고 있어야 하는데, 이전에는 motor 에 걸리는 load 가 커서 motor 에서 발열이 크게 일어나는 문제가 있었다. 하지만 counter mass 를 통해 motor torque 를 크게 감소시켜 발열 문제를 거의 해결할 수 있었다. 세 번째는 빨간색 공을 다루는 방법으로, 일반적인 타 팀들은 빨간 공이 발견되면 우회하는 방법을 사용하지만, 우리는 빗면의 hardware 적인 design 을 통해 빨간 공은 옆으로 밀어내는 방법을 사용했다. 따라서 software 가 가벼워졌으며, 미션 수행 시간 또한 단축시킬 수 있었다.



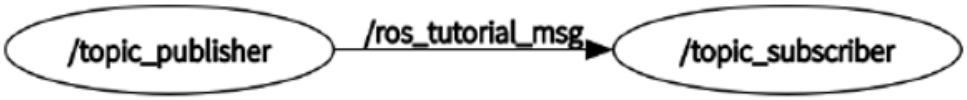
(Figure6. A ramp design for pushing out the red ball)



(Figure7. Final design)

2. ROS part의 전체적인 진행 과정

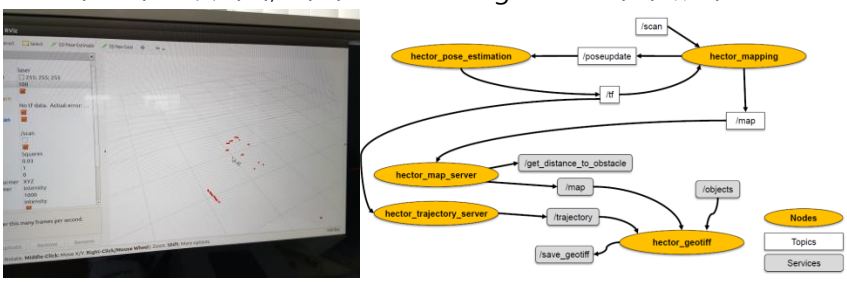
우선 ROS 사용을 위한 설치 과정 및 환경설정 방법을 배웠으며, 크게 1) ROS의 원활한 구동을 위한 Linux 운영체제인 Ubuntu를 개인 laptop에 설치하는 과정, 2) Terminal을 열고 스크린을 분할하거나, cd, ls, sudo...등의 간단한 명령어 및 Ubuntu의 주요 기능, 3) ROS를 설치하고 원활한 작업을 위한 환경설정 방법 등으로 나눌 수 있다. 그 후 본격적으로 ROS를 통한 message 송수신 방법을 배웠다. 1) Package file을 생성하는 방법, 2) Message file을 생성하는 방법, 3) Publisher node 및 Subscriber node를 생성하고, CMakelist를 이에 맞게 수정하는 방법, 4) rosrun 및 roslaunch를 통해 만들어진 node와 package를 구동하는 방법, 5) rqt_graph, ros echo등을 통해 node간의 관계를 가시화하고 message를 확인하는 방법 등을 배웠다. 이를 바탕으로 ROS통신을 하는 node를 만들어 보는 과제를 진행했다.



한편 추가적으로 요청이 있을 때만 작동하는 일회성 통신인 service방식을 과제를 하며 공부했으며, server와 client를 만들고, 두 숫자를 입력했을 때 client에서 그 합이 출력되는 system을 실습해보았다.

이후 본 system에 필요한 software package를 이용하는 방법을 배워보았다. Xbox controller를 통해 값을 입력 받는 방법, webcam과 lidar 신호를 수신하는 방법을 배웠다. 이를 바탕으로 실제 우리의 NUC를 xbox controller와 연결하고, lidar로 받은 신호를 rviz를 통해 2D로 가시화시키는 과정을 진행했다.

Lidar 를 NUC 에 연결시킨 뒤, RVIZ 를 켜고 최적의 visualization 을 위한 환경설정을 했다. 그 뒤 Lidar 를 이용한 실험을 진행했다. 실제 demo 조건과 유사하게 장애물이 적은 평지에 Lidar 를 놓고, 공을 Lidar 에서 점점 떨어뜨려가면서 RVIZ 로 이를 확인해보았다. 실험 결과, 공이 Lidar 와 너무 가까워도 인식이 되지 않음을 확인할 수 있었고, 최소 15cm 정도, 안전하게는 20cm 정도 떨어져 있어야 식별이 가능했다. 따라서 나중에 Lidar 를 통해 공의 거리 및 위치를 파악하기 위해서는 차체가 공과 너무 가까워질 때까지 이동하면 안되고, 이는 dynamic cell 을 통한 구동 control 시 유의해야 함을 깨달았다. 또한 공을 잡기 위해서는 공과 15cm 보다 더 가까워져야 하는데, '이 과정에서는 공을 어떻게 식별해야 하는가'라는 문제에 봉착했고, 이 과정에서는 lidar 가 아닌 webcam 이 필요함을 느꼈다. 즉, 15cm 보다 멀리 있을 때는 lidar + webcam data, 그보다 가까워졌을 때는 webcam data 만을 이용해야 한다. 한편 최대 식별 가능 거리는 약 6m 정도로, lidar 의 spec 과 거의 일치했다. 여기서 문제가 발생했는데, demo 시 기계동 중앙 로비의 면적이 6m 보다 넓으므로, 초기 위치에서 lidar 를 통해 공의 위치를 모두 파악하지 못하게 되는 경우가 발생할 수 있게 되었다. 이는 초기 위치에서 lidar 를 통해 공의 위치 파악 -> 이를 바탕으로 지도 제작 -> 최단거리 path planning -> 공 collecting 이라는 기존의 모델이 제대로 작동하지 않을 수 있는 위험성을 의미했다. 따라서 차체가 완성된 뒤 본격적인 실험 등을 통해 collecting algorithm 을 개선하거나, 바꾸어야 함을 인식했다. 한편 lidar 실험 외에도 인터넷의 여러 블로그 및 ROS 홈페이지에서 SLAM 방법을 찾아보았다. SLAM 을 지원하는 여러가지 package 들이 있었는데, 그 중 우리에게 가장 적합하다고 생각된 package 는 Hector_SLAM 으로, 기본적인 process 는 mapping 을 통해 map 을 만들고, mapping 을 하며 pose_update 를 지속적으로 진행하며 pose_estimation 과 feedback 을 한다. 이를 바탕으로 map_server 및 trajectory_server 를 구성하는 것이며, 자세한 block diagram 은 아래 있다.

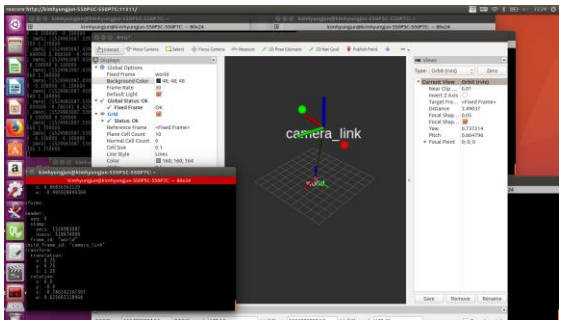


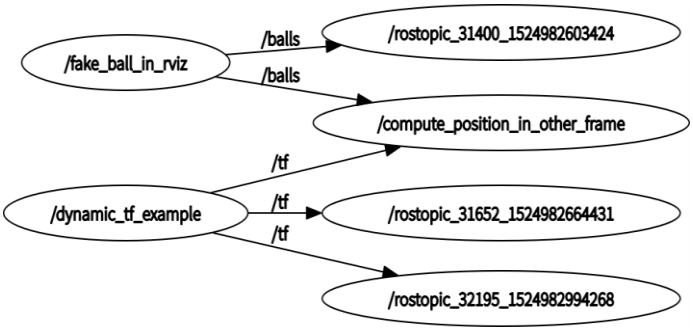
http://wiki.ros.org/hector_slam
https://hollyqood.wordpress.com/2015/12/01/ros-slam-2-hector-slam-2d%E5%9C%B0%E5%9C%96%E5%BB%BA%E7%BD%AE/amp/

4 월 20 일 금요일 연습반 session 이 있었으나 시험이 있는 탓에 참여하지는 못했고, 따라서 개인적으로 github 에 업로드 된 transformation matrix 와 ROS tutorial 을 공부해보았다. Transformation matrix 란 relative coordinate 를 absolute coordinate 로 변환 시켜주는 역할을 한다. 본 mission 수행을 위해 node construction 을 해보면, webcam 이나 rp_lidar node 를 통해 얻은 raw data 를 이용하, 특정 ball_detection node 를 통해 ball_position 및 color data 를 얻는다. 이는 차체에 대한 상대적인 data 로, fixed point 에 대한 절대 좌표 data 로 변환을 위해서는 tf node 가 필요하다. 본 tutoria 에서는 ball_detection node 를 지나서 output 으로 나온 relative ball position 을 absolute ball position 으로 변환하는 방법을 소개한다. 또한, 두 종류의 tf, static_tf 와

dynamic_tf 의 의미와 implementation 방법을 배울 수 있었다. 이를 우리 collector-bot 에 적용시켜보면, 차체에 대해 고정되어있는 webcam 및 lidar 에 대해, 우선 static_tf 를 적용시킨 뒤, bot 이 이동하므로 dynamic_tf 를 거치게 하면 absolute coordinate 를 얻을 수 있을 것이라 예상된다. 또한 두 package 의 source code 를 분석해본 결과, static_tf node 를 작성할 때는 차체에서 lidar 및 webcam 의 물리적인 위치 정보를 알아야 하며, 즉 x,y,z,rotation with x,y,z axis 의 총 6 개 data 가 필요하다. dynamic_tf node 를 작성 할때는 이동변위 delta_pos 와 회전인 rot 정보가 필요하다. Tutorial 에서는 이 값들이 constant 지만, 실제로는 시간에 따라 변하는 값이므로, 실시간의 position 과 rotation 정보가 필요하며, 이는 encoder, webcam, lidar, IMU, compass 등의 sensor data 를 input 으로 하는 추가적인 node 의 필요성을 알게 했다. 그러나 ball collecting algorithm 에 따라 각 tf 들의 필요성이 결정되므로, 우선 mission 수행을 위한 algorithm 을 체계화하고 결정하는 과정이 우선시 돼야 한다는 결론을 얻었고, 추후 이를 진행했다.

또한 ball_detect node 의 cpp file 과 open cv source code 를 이해하고, 통합해보려는 시도를 계속 진행했다.

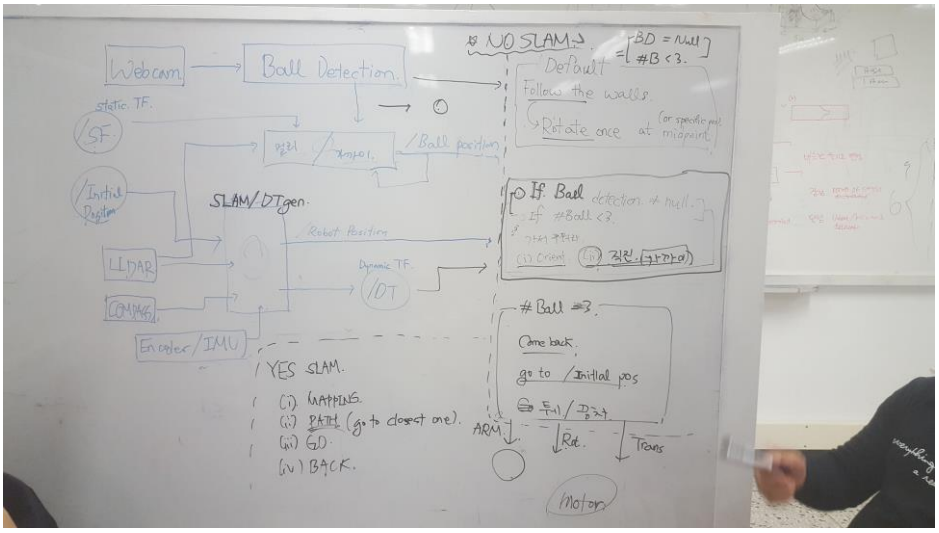




4 월 26 일, ball collecting algorithm 을 최종적으로 결정하고, 이를 통해 필요한 data 들과 node 를 생각해보고, node 의 관계를 정리하는 과정을 진행했다. SLAM 의 어려움과 불확실함, sensor 의 한계 등으로 인해 기존의 algorithm 에서 크게 변경하였다. 기존 'ball & map sensing → path planning → collecting'이라는 과정에서, '공이 detect 될 때까지 이동 → map 의 테두리를 따라 이동하고, 중간중간 회전하며 ball 을 detection → collecting → 3 개의 공 획득할 때까지 이를 반복'의 과정으로 변경했다. 이 때 default 값을 map 의 테두리를 따라 이동, 회전하며 target ball detection 으로 두어, 공이 발견되지 않으면 찾을 때까지 이룰 반복 하게끔 했다. 또한 공 3 개를 모두 저장하면(collector 에 연결된 servo motor 가 3 번 작동되면), compass 및 robot position data, webcam 을 통한 초록색 공 detection 등의 input 이 필요한 귀환 node 를 통해 돌아오는 mechanism 을 설계했다. 다음 발표까지 목표를 설정했다. 목표는 한 개의 공을 detect 한 뒤, robot 을 회전시켜 공의 center 와 robot 의 center 를 align 하고, 직진하고, 공을 collect 하는 것이다.

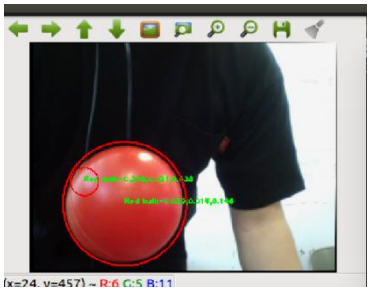
이렇게 결정된 algorithm 을 4 월 27 일 발표하는 시간을 가졌다. 창시구실에서 모든 조들이 모여 각자의 node construction 및 algorithm 을 발표하고, 조교님들께 피드백을 받았다. 주요 피드백으로는 1) SLAM 을 이용하면 mapping 이 잘 안되고, 어려우므로 시간이 얼마 남지 않음을 고려하면 사용하지 않는 편을 추천, 2) 최대한 빨리 구현해보고 실패해 보는 것이 중요, 3)

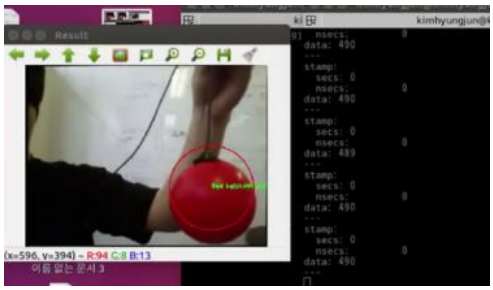
algorithm 을 node & msg level 로 최대한 구체화 시킬 것이었다. 우리의 algorithm 을 최대한 빨리 구현해보아 trial 의 횟수를 높이는 것이 중요함을 느꼈다.



우선 기존 주어진 ball_detection node 와 open cv 에서 만든 filter, depth calculation code 를 통합하는데 성공했다. 빨간 공이 detection 될 때만 position message 를 보내도록 설정하여 빨간 공만 tracking 할 수 있도록 했다. Webcam_node 를 켜고, 통합된 ball_detection node 를 켠 뒤, rostopic echo /balls, position 을 입력한 결과 빨간 공의 position message 가 잘 publish 됨을 확인했다. 그런데 여기서 문제가 발생했다. 아래 사진과 같이 공 표면에 빛이 반사되는 현상 등에 의해, 공이 하나로 lumping 되지 않고, 여러 개의 원으로 detect 되었다. 이렇게 되면 동시에 여러 개의 position message 가 보내지고, 이 중 어떠한 data 를 사용해야 하는지 모르는 문제가 발생한다.

뿐만 아니라, webcam node 의 publishing Hz 와 ball_detection node 의 subscribing Hz 를 일치시켰음에도 불구하고, 가끔씩 zero data 가 보내지는 문제가 발생했고, 추후 구동에 문제를 야기할 수 있다. 이러한 문제점들을 해결하기 위해, 우리는 data_massage 라는 package 를 만들고, 그 안에 가장 크게 detecting 된 공의 data 만을 보내주고, zero data 를 없애주는 node 를 만들었다. Ball_detection node 를 켠 뒤, 이 data_massage node 를 켜니 가장 크게 detecting 된 boundary 에 대해서만 단일 position data 를 보내주는 것을 확인할 수 있었다.





이렇게 만들어진 position message 를 subscribe 하여 TCP-IP 통신을 통해 my-Rio 로 구동에 대한 data 를 보내주는 node 를 만들어야 했다. 우리는 x-box control node 를 변형하여 이를 만들었다. Ball position 의 x 좌표가 240 이 될 때까지 회전 data 를 보내주고, 그 이후 y 좌표가 특정 값이 될 때까지 직진 data 를 보내주고, 그 다음 sweeper motor 작동 data 를 보내주는 algorithm 을 code 상으로 구현했다. 이를 토대로 실험해 본 결과, task 를 잘 수행함을 확인할 수 있었으나

몇가지 문제가 존재했다. Ball position data 가 publish 된 뒤, 이에 대해 차량이 구동할 때까지 약 1~2 초 이상의 motion delay 가 발생하는 것이 문제였다. 이를 해결하기 위해 1) data publishing Hz 을 낮추고, 2) image size 를 압축했으며, 3) open cv 에서 만든 track bar 및 window 들을 제거했고, 4) ball 과의 align 을 위한 회전 속도를 낮추었다. 이러한 방법들을 시도해 본 결과 delay 가 줄어듬을 확인할 수 있었다. 한편 갑자기 확 튀는 data, 즉 singular value input 에 의한 영향을 줄이기 위해, 이전 data 를 참조하는 filter code 를 구현했다.

그 뒤, 가장 기본적이면서 필수적이지만 귀찮다고 미뤄왔던 task 들을 처리했다. 빨간공을 줍는 mission 으로 착각해 빨간공을 tracking, collecting 하도록 했었는데, 이 ball_detection node 에서 파란공의 position 을 publish 하도록 바꾸었다. 또한 기존에 webcam node 실행 → ball_detection node 실행 → data massage node 시행→ xbox control node 실행으로 이루어졌던 과정을 roslaunch file 을 통해 한번의 command 로 실행할 수 있도록 개선했다. 또한 기존에는 nuc 의 작동 상황을 살펴보기 위해 HDMI 케이블을 모니터와 연결해야 해서 차량의 운용이 불편했다. 비록 master-slave 를 선언했지만, 여전히 roscore 를 nuc 에서 실행해야 하는 불편함이 있었고, 이는 team viewer 를 설치하여 개인 laptop 으로 현황을 파악할 수 있도록 개선했다.

한편 xbox control node 를 개선하는데 집중했다. 차량이 sweeper 를 통해 공을 collecting 한 이후에는 sweeper 가 움직이지 않아야 하는데, 이상하게 계속 까딱까딱 움직이는 현상이 발생했고, rostopic echo 를 통해 확인해 본 결과 xbox control node 의 sweeper 작동 command 가 my-Rio 로 전달되는 것을 확인했다. 분명 공이 특정 거리 내로 들어와야 sweeper 가 작동되도록 코드를 만들었으나 이런일이 발생해 당황스러웠고, 따라서 debugging 이 필요했다.

또한 차량이 파란공 3 개를 모두 줍고 골대로 되돌아오기 위해서는 차량의 현재 위치 및 방향 등에 대한 정보가 필요하므로, IMU 를 이용하기로 했고, acceleration, direction 등을 측정할 수 있는 MPU 9150 모델을 구입했다. 이를 위해 필요한 ros package 를 찾고, arduino 를 adaptor 로 이용하여 nuc 와 연결하는 방법을 모색했다.

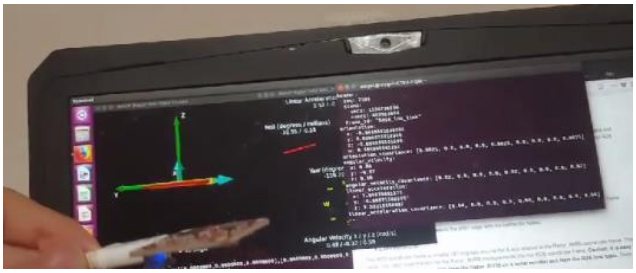
차량의 움직임이 delay 되는 문제가 webcam node 에서 ball_detection_node 로 image data 를 publish, subscribe 할 때 발생한다고 생각되어 두 node 를 통합했고, 그 결과 delay 를 획기적으로 감소시킬 수 있었다.

또한 Ball collecting process 의 reliability 를 높이기 위한 debugging 을 진행했다. 파란공 3 개를 배치시키고, ball collecting algorithm 을 실행시켰을 때, 가끔씩 차량이 멈추거나 부자연스럽게 움직이는 문제가 발생했다. Debugging 을 위해서 xbox_ctrl node 의 사이사이에 data 를 출력하는 코드를 입력하여 log file 을 생성해 문제를 파악해보았다. 그 결과 차량이 공과 정확히 align 되었을 때, ball_detection node 에서 subscribe 한 공의 position data 가 아주 미세하게 변화하여 차량에 회전 command 가 조금씩 입력되기 때문임을 확인했고, 따라서 차량과 공이 align 되어 공의 x 좌표가 특정 값 이내로 들어오면 0 으로 수렴되도록 code 를 변경하여 문제를 해결할 수 있었다.

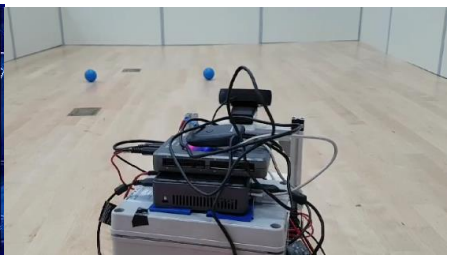
Ball collecting 이 제대로 수행되자 다음 단계인 귀환 algorithm 을 생각해보았다. Algorithm 은 다음과 같다. 파란 공 3 개를 모두 collecting 하면 360 도 회전하며 초록색 공을 detecting 한 뒤, 특정 거리까지 직진한다. 만약 초록색 공을 발견하지 못하면 초록색 공을 발견할 때 까지 random motion 등을 진행하며, 이에 대한 자세한 algorithm 은 추후 개발하고자 했다. 초록색 공과 가까워지면 방향을 basket 과 로봇의 center line 이 직각이 되도록 방향을 align 하는 과정을

진행한다. 이 때 imu 을 통해 얻은 yaw data 를 이용하여, 차체의 yaw 가 특정값이 될 때까지 회전시킨다. 그 뒤 좌우 운동을 통해 두 개의 초록 공 사이로 차체를 위치시킨 뒤, 360 도 회전시켜 hatch 를 개방하여 공을 release 한다.

이를 위해서는 우선 imu 를 nuc 에 연결하여 원하는 acceleration data 를 얻도록 하는 과정이 필요했다. 이를 위해 arduino 에 imu 를 연결하여 firmware 를 다운로드 해주는 과정을 진행했고, data 가 정확히 도출됨을 확인했다.



또한 laptop 을 통해 nuc 를 원격으로 조종하는 방법을 찾아보았다. 원래는 teamviewer 를 설치하여 원격 조종하려 했으나 tcp/ip 통신을 위해서는 nuc 가 인터넷을 지원하지 않는 my-Rio wifi 에 연결되어있어야 하는 문제점이 있었다. 따라서 vnc 를 이용하는 방법을 찾아보았고, 그 중 tightvnc 라는 program 을 설치했다. 그 결과 nuc 에서 vncserver 를 실행시키면 my-Rio wifi 에 연결되어 있더라도, 개인 노트북을 my-Rio wifi 에 연결시키고 vncviewer 를 실행시킴으로써 원격 조종이 가능했다. 두 번째로 공동강의실에서 field test 를 진행했다. 중점적으로 얻고자 했던 것은 imu 를 통해 측정한 yaw data 였다. 이를 통해 basket 과 align 될 때까지 회전해야 할 각도를 찾고자 했다. 이를 통해 test 과정동안의 모든 data 들이 저장된 bag file 을 생성했고, 창시구실에서 이를 분석하여 적절한 reference yaw value 를 찾아보았다. 세 번째로 heat transfer 문제를 알아보기 위해 공동강의실에 비치된 열화상 카메라를 통해 차체의 온도를 측정해보았다. 가장 발열이 심했던 부분은 motor 부분으로, 약 32 도 정도로 크게 높지는 않았으나 그래도 온도를 낮출 방법의 필요성을 느껴서 motor 에 heat sink 와 thermal grease 를 붙여주었다. 또한 nuc, my-Rio 등의 electric device 에서 발생하는 열 문제 해결을 위해 fan 을 장착하기로 결정했다.



이후 field test를 반복적으로 진행하며 debugging을 하는데 집중했다. 사실 software적인 bug는 크게 발견되지 않았고 대부분의 문제는 hardware적인 측면에서 발생했다. 특히 ball collecting 과정의 성공률이 약 70%정도로 불안함을 보였다. 이를 해결하기 위해 최대 torque가 높은 모터를 사용하고, sweeper의 design을 변경하는 과정을 내내 반복했다.

최종 demonstration의 결과는 다음과 같다.

첫번째 시도 때는 공 3개를 모두 collecting하는데 성공했고, 골대로의 귀환도 성공했다. 하지만 골대와 차체의 align 이후 직진하는 과정에서 직진하는 거리가 아주 살짝 모자라 공 3개를 골대 안에 제대로 넣지 못했다. 이는 compass를 사용하는 우리 팀의 특성상, 전날 calibration을 정확히 해 놓았음에도 불구하고, 최종 demo날 수많은 사람들의 전자기파 영향 등으로 calibration이 틀어진 것으로 추측된다.

두번째 시도 때는 공 3개 중 하나를 줍지 못한 채 귀환했다. 이러한 결과의 주 원인은 my-Rio와 laptop간의 tcp-ip 통신이 계속 끊겼기 때문이다. 우리 조는 공을 주울 때 빗면과 sweeper라는 두 개의 장치를 이용한다. 차체가 직진하며 빗면으로 공을 밀어 올리면서 sweeper로 쓸어 담는 방식으로 collecting을 하는데, 최종 demo때는 연결이 계속 끊겨 차체의 진행이 매끄럽지 못하고 계속 끊기면서 직진하는 모습을 보였다. 따라서 공을 밀어 올리는데 문제가 발생했고, 하나의 공을 줍는데 실패했다. 첫번째 시도에서 귀환 거리에 대해 다시 calibration을 진행했기에 귀환에는 성공했고, 골대에 공 2개를 넣으며 demo를 마쳤다.

아쉬움이 굉장히 많이 남은 결과였다. 그날 통신이 잘 끊기지 않았더라면, calibration이 틀어지지 않았더라면 등의 여러 생각이 들었다. 이러한 문제 원인들이 demo 전에도 예상되었으나 마땅히 해결방법이 없어서 그대로 둘 수 밖에 없었고 그저 이런 문제가 발생하지 않기를 바랬는데 그것은 우리의 발목을 잡았다. 우리 팀 모두 엄청난 노력을 쏟고, 같이 밤을 지새우며 힘든 시간을 보냈기에 더욱 아쉬울 수 밖에 없었다. 하지만 힘들었던 과정만큼 배운 것도 많았고 돌이켜보면 참 즐거운 시간이었던 것 같다. 몸은 피곤하고 찌들어 있었지만 나도 모르게 밤을 새우고 몰입하고 있는 나를 발견할 수 있었다. 함께 노력해준 팀원들이 너무 고맙고, 앞으로 같은 길을 걸어갈 능력 있고 착한 친구들을 만나게 된 계기가 된 것 같아 의미 있는, 강의 이상의 의미를 지닌 강의였다.