창의적 시스템 구현 Report 1, (3.19~25)

20140344 양예준

오준호 교수님 그룹, Labview part

Labview part

Labview 2번째 과제를 해결하며 공부하는 주였다. 1번째 과제 때 while loop안에 while loop를 돌리는 과정이나 while loop안의 값이 return되는 조건을 몰라서 헤맸었는데, 2번째 과제를 하기 위해 찾으면서 labview의 기본적인 구동방식을 이해하였다. 2-1과제는 LED 4개를 다른 주기로 깜빡이는 것인데, 첫 주에 미리 만들었을 때에는 하나의 while loop내에서 몇 번 실행됐는지 알 수 있는 변수 i를 적당한 값으로 나누어서 구현하였었지만, 구동방식을 이해한 뒤에 여러 while loop를 만들고, 각 loop의 주기를 설정하여 돌아가게 만들어주었다. 2-2과제는 순서대로 짜는 것이고, 이 또한 여러 방식으로 구현해보았다. 먼저 위에서 언급한 while loop내의 변수를 4로 나눈 나머지 값이 각각 0, 1, 2, 3일 때 해당하는 led가 켜지도록 만들었다. 다른 방법으로는 1개의 T, 3개의 F값을 초기값으로 지정한 쉬프트 레지스터를 사용하여, i번째 레지스터값이 다음 루프에서는 i+1(mod 4)값이 되도록 짜보기도 하였다. 2-3과제는 1번째, 2번째 과제를 합치는 과정이었다. Myrio 기본 project를 만들면 Main.vi 내에 accelerometre가 구현되어 있어 이를 사용하였다. (첫 번째 과제 때에는 해당하는 기능을 찾아 직접 만들어보았다.) 이 x, y, z 벡터 합의 크기가 중력의 1.5가 넘을 때 led가 불이 들어오게 만들었다. 3번째 과제의 일부를 미리 진행해 보았다. Robotis 자체 프로그램을 통하여 후에 여러 모터를 동시에 제어하는데 필요한 모터의 id를 설정해주었다. DXL motor 패키지를 labview에 설치하고, 해당하는 패키지를 찾아 코딩하였다. 이 과정에서 motor 컨트롤에 관련된 같은 이름의 패키지가 여럿이어서 input, output에 필요한 형식이 달라서 제대로 된 코딩을 못하고 헤맸었으나, 한 패키지를 포함하는 패키지 안에서 다른 패키지를 찾아서 만드는 방식으로 해결하였다. 이를 통하여 motor를 제어해보았다. 그런데 이 때 12V input을 배부받은 것이 없어서 NUC 구동에 필요한 19V input을 사용하였다.

Other part

NUC의 기본적인 환경세팅을 끝냈다. 아직 필요하진 않지만, 중간 점검 삼아서 코드들을 옮겨NUC에서도 실행을 시도해보아 환경이 달라서 생기는 문제를 줄이기 위해서이다.

ROS part에서 환경이 익숙하지 않아 생긴 에러 몇 개를 같이 해결했다. 라이다 센서에서 publish하는 데이터를 rviz로 가져오거나 frame을 잡을 때 이름이 달라 생긴 같은 에러들이 있다.

OpenCV에서 C++ 문법에 익숙하지 않아 생긴 에러들을 같이 해결하였다. 함수의 정의를 사용되는 부분보다 위에 해야한다던지, 변수를 두 번 정의하여 생긴 에러들이 있다.

창의적 시스템 구현 Report 2, (3.26~4.1)

20140344 양예준

오준호 교수님 그룹, Labview part

Labview part

Labview 3번째 과제 제출을 했다. 3-1과제는 모터를 돌려보고, TCP/IP 통신을 시도하는 것이었다. 저번 주에는 모터 속도 컨트롤은 직접 만들어보았고, 이번에는 예제를 가져다 써서 wheel mode, joint mode를 모두 사용해 보았다. TCP/IP 통신은 코드의 이해만 하였고, 직접 만들기엔 오래 걸리므로 예제를 가져다 썼다. 통신은 기본적으로 state machine을 이용하여 연결 상태를 관리하며, ip와 포트가 일치한다면 연결은 알아서 해준다. Code의 client부분에서 접속을 시도하는 ip를 MyRio의 ip로 바꾸어서 PC와 MyRio간의 소켓 연결을 정상적으로 하였으며, 특정 키보드 입력에 특정 값의 array를 대응시켜서 서버에서 이 array를 읽어 화면에 표시된 Xbox 컨트롤러 인풋 값이 바뀌는 것을 확인하였다. 다음으로 3-2과제의 경우 PC를 통하여 MyRio에 신호를 주고, 이 신호를 이용하여 모터를 제어 하는 것이다. Tcp/ip 연결은 3-1과제와 같이 만들었으며, 이 때 전달하는 data array를 mode, change에 해당하는 두 개의 값만 가지도록 바꿔주었다. 그 후 mode, change 값에 따라서 모터의 구동 방식을 바꿔주거나, 현재 position 혹은 speed 값에 change를 더해주었다. 마지막으로 Pc, MyRio간의 통신이 아니라 ROS, MyRio간의 통신을 시도해 보았다. ROS는 제공된 xbox\_ctrl폴더 내의 xbox\_ctrl\_client 노드를 사용하였다. Xbox\_ctrl\_node의 경우 코드에 socket 부분이 없어 xbox컨트롤러에서 ROS로 정보는 받지만 myrio로 통신을 안하기 때문이다. Ip, port를 맞추고 ros, Myrio 통신을 이용하여 xbox 컨트롤러의 신호를 받는 것을 시도해보았는데, MyRio server의 state가 ros에서 node를 켰을 때 한번 바뀌고 더 이상 신호를 받지 않았다. 따라서 연결 시도까지만 되고 성립이 안된다던가, 성립이 됐으나 데이터를 안 보낸다거나, 형식이 맞지 않는 것 같다.

Other part

OpenCV에서 특정 함수는 GRAY input만 받는데, HSV 시스템을 사용하고 있었고, HSV->GRAY 변환 기능이 없어서 에러가 나서 HSV->RGB->GRAY순으로 변환하여 해결했다. 공만 존재할 때는 상관 없지만, 화면상에 다른 물체로 인해 물체 하나에 대한 컨투어 집합을 제대로 찾지 못하게 되면, 서로 섞여서 더 많은 컨투어 집합이 생기고 각 컨투어 집합에 대하여 에러가 작은 가장 알맞은 원을 찾기 때문에, 수 많은 원의 얻게 된다. 따라서 이 에러의 범위를 조절하여 노이즈를 제거하려고 시도해보았는데 너무 많은 폭의 에러가 있었다. 따라서 얻은 공의 지름을 이용하여 에러가 지름과 상관없이 만들었는데, 이전보다는 좋은 결과를 보였으나, 이 또한 노이즈의 범위를 포함하였다. 최종 시연 장소 환경에 따라 이대로 써도 되거나, 개선이 필요한지 나뉠 것 같다.

창의적 시스템 구현 Report 3&4, (4.2~4.15)

20140344 양예준

오준호 교수님 그룹, Labview part

Labview part

먼저 해결하지 못했던 에러를 해결하였다. ros 기기에서 xbox input을 받아 labview 프로그램에서 데이터를 표시해주었다. 이는 example code에서 처음에 읽을 데이터의 크기를 받아오는 작업이 있어서 생긴 문제였다. 따라서 ros에서 보내는 데이터의 크기를 고정으로 받아오도록 바꾸어 주며, write과정을 없애 read과정만 남겼다.

다음으로 한 작업은 각 바퀴에 id를 할당하는 작업이었다. 모든 바퀴를 따로 돌려야 하므로 기본적으로 할당돼 있던 id가 아닌 각각 0~4의 id를 할당해주었다.

그리고 ros에서 각 바퀴에 목표 rpm 정보를 보내주면 그 정보를 받아 각 바퀴에 들어가도록 코드를 짜주었다. 기본적으로 xbox input을 받는 것과 같으며, 4개의 double값을 받아와서 순서에 맞춰 각 바퀴로 보내준 것이다.

처음 생각한 회로에서는 모든 바퀴가 돌아가지 않고, 특정 바퀴만 돌아갔다. 이 회로는 myrio-u2d2-왼쪽 바퀴들-SMPS2Dynamixel-오른쪽 바퀴들이었다. U2d2와 바퀴 사이에 전원이 있는 건 괜찮지만, 바퀴와 바퀴 사이에 전원이 있는 건 안되는 것으로 판단하여 myrio-u2d2-SMPS2Dynamixel-바퀴들 순서로 연결시켜주었고, 잘 동작했다.

그 후 Myrio가 컴퓨터와의 연결없이 스스로 작동을 해야하므로, 배포하는 법을 인터넷에서 찾아서 적용하였다. 컴파일을 해준 뒤, myrio 저장공간에 넣고 재시작 시 자동으로 실행되도록 설정해줬다.

자동으로 켜지도록 하는 것은 좋은데, 이 경우 구동 중인지, 에러가 났는지 확인하기 힘들기 때문에 제대로 실행이 되었다면 LED0에 불이 들어오도록, TCP/IP 연결이 성립되었다면 LED1에 불이 들어오도록 만들어 주었다.

참고자료에 올라온 회로를 따르면 data선만 병렬로 연결하였는데, 나중에 특정 바퀴의 데이터를 받고 싶을 때 (토크 상태 확인 같은 작업) 문제가 생길까봐 모두 직렬로 연결해주었다. 이 기능이 필요없고, 구조상 병렬로 연결해야한다면 바꿀 것이다.

앞으로 할 일은 다음과 같다. 새 모터가 연결된다면 그 모터를 넣어주기만 하면 된다. 재시작에 너무 오랜 시간이 걸리므로, 에러가 났을 시 재시작이 없이도 다시 시작하도록 만들어 줄 수 있다면 좋겠다. Myrio와 NUC 사이의 wifi 자동 연결도 생각해봐야 할 것 같다. 바퀴가 미끄러지므로 labview단계에서 가능하다면 이 에러를 보정해주어야한다.

ROS part

Labview part와 연결되는 부분의 코드를 작성하였다. 기본적으로 xbox\_ctrl.cpp을 수정한 것이다. 처음 xbox input을 보내주기 위해서는 바꿀 필요가 없었다.

다음으로 각 키에 맞춰서 모터를 컨트롤 하도록 바꿔주었다. 왼쪽 조이스틱으로 상하좌우의 움직임을, 오른쪽 조이스틱으로 회전을 컨트롤 한다. 왼쪽 조이스틱의 각도와 중심으로부터의 거리를 통하여, x방향과 y방향의 속력을 계산한다. 오른쪽 조이스틱의 각도와 중심으로부터의 거리를 통해, 중심을 기준으로한 회전 속력을 계산한다. 그 뒤, 랩뷰 파트 조원이 미리 조사한 공식을 적용하여 움직임에 필요한 각 바퀴의 회전속도를 계산하였다. 각 방향의 입력으로 생기는 움직임의 최대치는 60rpm을 넘지 못한다. 만약 여러 방향의 입력이 더해져 60rpm을 넘는다면, 바퀴 중 제일 빠른 바퀴의 속력이 60rpm이 되도록 같은 비율로 줄여준다.

Ros파트 조원이 미리 조사한 gmapping이 잘 되지않아 같이 실행시켜 보려고 하였다. 튜토리얼을 따라하며 시행착오를 겪은 결과 Rqt\_graph로 확인했을 때 각 노드들은 잘 연결이 되었다. bag파일도 생성되는 듯 하였다. 하지만 실제로 작동을 시키고, map data를 rostopic echo를 통해 확인했을 때에도 아무 데이터를 전송하지 않았다. 발표 전까지 문제를 해결해야한다.

Total part

DC-DC컨버터를 정했다. 모터 스펙을 봤을 때 stall A가 1.7A이다. 따라서 모터 4개에 Myrio를 돌리고, 설계 상 2개가 추가 될 것을 생각하여 여유를 생각해서 24V-12V 10A 컨버터를 구매했다. 이 컨버터로 Myrio와 모터들을 모두 구동 시킬 것이다. 21.6V와 NUC에 필요한 19V는 큰 차이가 아니기 때문에 조교님의 조언에 따라서 NUC은 배터리에서 직접 연결하기로 하였다.

각 파트 작업이 합쳐진 1차적인 결과가 나왔다. solidwork파트에서 설계한 대로 차체를 만들고 바퀴를 달았다. 그리고 ros파트의 작업물을 통해 myrio로 신호를 보내고, myrio에서는 신호를 받아 모터를 컨트롤하는 것이다. 이 합쳐진 결과물이 실제로 잘 이동하는지 구동시켜 보았다.

창의적 시스템 구현 Report 5&6, (4.16~4.29)

20140344 양예준

오준호 교수님 그룹, Labview part

Labview part

ROS와 연동하여 어떤 정보를 주고 받을지 결정하고 있다.

SLAM을 위해서 MyRio에 내장된 가속도 센서 정보와 바퀴의 회전수를 받아와야 할 수도 있다. 바퀴의 회전은 slip이 일어나므로 사용에 대한 토의나 실험이 더 필요하다.

ROS part

OpenCV와 합치는 작업을 진행했다. OpenCV part Simeneh가 쓴 코드에 ROS node와 공 위치를 담은 publisher를 넣는 것이다.

처음 합쳤을 때 정상적으로 구동이 안되어, openCV 버전 문제라고 생각했다. ros내에 설치된 opencv와 직접 설치한 opencv버전이 다르기 때문이다. 그래서 컴파일 시에 따로 설치한 opencv를 사용해보려 시도를 해봤지만 실패했다. 하지만 다른 곳에서 간단한 실수를 잡으니 해결되었다. 알고 보니 videoCapture함수에서 0번 카메라가 아닌 1번 카메라의 데이터를 받아오려 한 것이 문제였다.

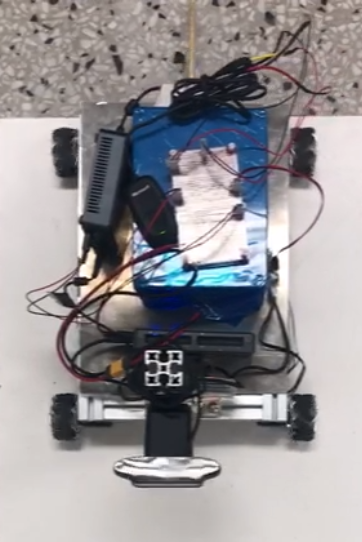
조원들과 전체 구조에 대해서 이야기를 하여, navigation & slam이 될 때와 안될 경우를 나누어서 전체 구조를 정하였다.

카메라가 바라보는 방향이 정면이면 문제가 없겠지만, 기울어져 있다면 좌표 변환이 복잡해질 것이다. 이를 ROS의 TF를 이용해서 쉽게 할 수 있는 듯하다. 이에 대한 공부가 더 필요하다.

공 위치를 받아서 따라 다니도록 하는 코드를 작성 중이다.

Total part

모든 파트 작업 결과물을 더해서 확인하는 작업을 하였다. 납땜이 진행되었으니, 차체 프로토타입 위에 모든 부품을 놓고, 배터리 전력을 통하여 구동시켜 보았다.



NUC

Camera

Battery

MyRIO

납땜기판

converter

Xbox 수신기

위 내용을 진행하면서 지금까지 만든 ROS 코드들을 구동시켜 보았다. Xbox를 통하여 조종을 하면서 화면을 통해 얼마나 흔들림이 있는지 보았다. 직진할 때는 거의 흔들림이 없으나, 회전 시에는 흔들림이 생겼다. 공 수거 알고리즘을 짤 때, 회전을 최소화하는 것이 좋을 것 같다.

창의적 시스템 구현 Report 7, (4.30~5.6)

20140344 양예준

오준호 교수님 그룹, Labview part

Labview part

Tcp/ip 통신에서 기존의 정보에 더해 state 정보를 주기로 했다. State 정보로 픽업 파트의 컨트롤을 labview에서 처리해준다.

일정 시간 동안 픽업 파트 상자를 내려준 뒤에 실을 풀고 상자를 더 내려준다. 그 후 공까지 내려갔으면 실을 다시 당겨주고 상자를 올려주는 것이다. 이 작업을 하는 동안 다른 컨트롤 메시지가 버퍼에 쌓이지 않도록 하기 위해, ROS쪽에서는 작업 완료 메시지를 받을 때까지 대기한다.

ROS part

카메라를 하나 더 설치하였다. 첫 카메라와 같이 프레임을 설정하여 주었다.

Control 노드 내의 state를 저장하여 그 state에 따른 다른 작업을 해준다. 카메라에 공이 없을 때, 카메라 1을 쓸 때, 2를 쓸 때, 공을 집을 때, 그리고 공 3개를 잡아서 돌아오는 작업으로 나누어져 있다.

빨간 공을 피하는 알고리즘을 정하려했다. 카메라에 들어오면 공을 중심으로 원으로 돌자는 이야기, potential field method를 이용하자는 이야기가 나왔다.

바퀴를 큰 것으로 바꿀 것을 생각하고 픽업파트를 만들었기에 각 파트 구동은 가능하나 합쳐서 구동을 할 순 없다. 따라서 차 프로토타입이 완성되면 직접 실행을 해보며 공을 잡는 작업에 필요한 수치들을 재조정 할 것이다.

창의적 시스템 구현 Report 8, (5.7~5.13)

20140344 양예준

오준호 교수님 그룹, Labview part

Labview part

픽업 파트를 들어올리면서 다음 작업을 하자는 이야기가 나왔다. 현재 랩뷰 코드는 픽업 파트가 움직이는 동안 다른 작업을 못하기 때문에, 이를 수정해야한다. Notifier를 사용하여 여러 while loop를 돌릴 계획이며, 기존에 픽업 파트 작업이 끝나면 보냈던 확인 메시지를 픽업파트가 올려지기 시작할 때 보내는 것으로 바꿀 것이다.

ROS part

바퀴를 큰 바퀴로 바꾸면서, 빨간 공을 피하지 말고 차체 아래로 흘려 보내자고 이야기를 했다. 바퀴를 달면 실험을 해보고, 잘 되면 파란 공만 집을 것이고, 아니라면 피하는 코드를 작성할 계획이다.

최적의 카메라 위치를 찾기 위한 작업을 진행 중이다.

보고서 7에서와 같이 최적의 동작을 위한 수치를 실험적으로 찾아야한다.

OpenCV part

공 내부에 빛으로 인해 생기는 원을 없애는 코드에서 오류를 잡는 작업을 진행하였다. 공의 반지름과 공 사이의 거리를 이용하여서 제거하였다.

저번 발표를 보니 Morphological Transformations을 이용하여 제거한 조가 있었다. 이를 보고 Simeneh가 구현을 해놨으며, 위에서 만든 기능과 같은 것이며, 나중에 더 빠른 방법만 적용하기로 하였다.

시연 장소에 대한 정보를 듣고 생각한 점들이 있다. 지금 사람이 입은 옷에서도 원이 잡히는데, 이를 보완해야 할 것 같다. 벽 높이에 따라 카메라 각도를 잘 조절해서 아예 못 보도록 하거나, transform 작업을 진행할 때 지면과의 높이가 예측 값을 많이 벗어나면 제거하는 방법을 생각중이다.

창의적 시스템 구현 Report 9&10, (5.14~5.27)

20140344 양예준

오준호 교수님 그룹, Labview part

Labview part

온도에 따라 팬을 컨트롤하는 것을 시도하였다. 처음엔 트랜지스터와 Myrio의 PWM기능을 이용하여 만들었다. 잘 돌아가고, 회전 속력도 조절하였으나 트랜지스터가 너무 빨리 뜨거워지는 문제가 있었다. 따라서 다음으로 시도한 방법이 MOSFET을 이용하는 것이었다. 트랜지스터와 달리 사용할 MOSFET에는 자체적으로 방열판이 달려있어 덜 뜨거워지기 때문이다. 그런데 PWM 기능이 달려있는 Myrio digital output의 최대 전압이 3.3V이고, MOSFET의 문턱 전압이 3.8V이기 때문에 작동이 되지 않았다. 그래서 1V정도를 다른 핀에서 가져올 생각을 해보았지만 실패했다. 아마도 Myrio의 모든 GND는 같은 전위라서 실패한 듯 하다. 따라서 최대 5V의 전압을 표현가능한 analog output을 이용해서 on/off만 제어하기로 하였다.

AX 시리즈 관절모드에서 전체 범위 컨트롤이 불가능하며 MX시리지의 다중 관절모드 설정도 불가능했다. NI 직원분이 오신 질문 세션에서 이에 대해 원인을 파악하고 주어진 라이브러리를 약간 수정하여 이 문제를 해결했다. 이를 통해서 실을 조이는 범위를 폭넓게 제어 가능해졌고, 픽업 파트의 경우 속도로 제어하여 높이 제어에 불확실한 점이 있었는데 이 문제를 해결했다. 하지만 만약 기어가 헛돌면 계속 잘못 돌기 때문에 큰 문제가 될 수 있다. 이를 해결할 방법을 잘 모르겠다.

또한 픽업파트를 속도로 제어할 때 불확실한 점을 최대한 제거하기 위해 고정된 시간으로 반복하는 루프를 하나 더 뺐었는데, 다중관절모드로 제어하면서 필요가 없어졌다. 따라서 이 루프로 인해 생긴 시리얼포트에 명령을 내릴 때 픽업파트 제어와 속도제어 명령 둘이 충돌하는 문제가 사라졌다.

진동 실험을 하기 위한 코드를 짜주었다. Myrio를 고정시키고 속도에 따른 myrio의 가속값을 파일로 저장하여 진동 해석을 위한 data를 수집했다.

매 실험마다 myrio를 껐다 켜야했었는데, 상위 루프를 하나 더 만들어서 연결이 끊기면 알아서 초기화 후 재시작하도록 만들었다.

전체적으로 처리 속도는 충분하지만, 시작 초기에만 렉으로 인해 버퍼에 메시지가 쌓이는 문제가 있었다. 이를 준비 확인 메시지를 통하여 해결하였다.

모터는 자체 온도센서로 온도를 확인하나, 나머지에는 온도센서를 달아야한다. 온도 센서의 회로와 코드를 만들었다.

ROS part

초록 공 2개로부터 수평하면서 카메라로 인식 가능한 최소 거리의 위치를 계산하는 식을 통해 얻은 좌표를 파란 공에 다가가는 방법에 대입하여 골을 찾는 코드를 짰다.

파란 공에 다가가는 코드를 좀 더 부드럽게 이동 할 수 있도록 바꿔 주었다.

가속을 위한 코드를 넣었었으나, 진동 문제로 잠시 빼놓았다. 속력의 최저값을 높게 설정해 생긴 문제인 듯하다. 그렇다고 속력 최저값을 낮추자니 느리다고 생각된다. 급가속에 큰 문제가 없으면 가속의 최대 값을 높일 생각이다.

골대로 다가갈 때 카메라로 보다가 rplidar를 사용하여 벽에 다가갈 생각이다. Rplidar의 점을 통해 점들의 기울기와 절편을 얻어 이를 이용하여 다가가볼 것이며 구현해야한다.

빠른 속도를 위해 픽업파트가 올라가는 동안 다른 공을 찾으려 했다. 그런데 이 방법은 아래쪽 카메라가 가려져서 멀리 있는 공부터 집는 경우가 있다. 이를 위해 픽업파트가 다 올라가는 것을 기다려야할지 아닐지 고민 중이다.

Rplidar를 직접 노트북과 연결하면 잘 작동하지만, 연장선을 이용하면 작동하지 않는다. 이를 해결 할 수 있으면 편할 듯하다.

빠른 실행을 위해 launch 파일을 만들었다.

OpenCV part

내부 공을 잡는 알고리즘을 만들었기에, 최소 공 크기에 대한 제한이 사라졌다. 기존에는 공 내부의 반사로 인한 원을 크기를 이용해 잡았었다. 따라서 이 부분의 코드를 제거하여 공 크기 제한이 사라졌기에 더 먼 공을 잡을 수 있게 됐다.

기존엔 고정 값으로 공 크기를 보정해주었는데, 사이즈에 따라서 약간의 선형으로 보정하기로 하였다.

Total part

카메라 높이, 보는 각도를 정하였다.

빨간 공을 무시하고 지나가는 것이 목적이었는데, 아래쪽 카메라에 걸리는 문제가 있었다. 차 전체적으로 올려야 할지 고민을 했었으나, 아래쪽 카메라를 뒤로 밀었다. 그랬더니 잘 지나가졌다.

창의적 시스템 구현 Report 11, (5.28~6.1)

20140344 양예준

오준호 교수님 그룹, Labview part

Labview part

픽업 파트 부분에 포지션 컨트롤을 사용하면 공이 걸렸을 때 모터가 힘을 최대로 주는 경우가 있다. 이때 기어가 고장이 난다. 따라서 토크 컨트롤을 사용하기로 하였다. 이를 위해 모터 스펙을 읽어보고, 이를 위한 코드를 새로 만들어주었다.

Ros part

공을 다 집은 후 골에서 먼 경우 초록 공을 인식하지 못하는 경우가 있다. 따라서 rplidar에서 얻은 점들의 평균 위치 (경기장의 중심)로 이동하는 코드를 넣어주었다.

최종 발표를 위해 테스트를 해보면서 생기는 마이너 버그들을 꾸준히 수정하였다.

첫 시연 때 공에 다가간 후 위치 미세 조정에 오랜 시간이 걸렸다. 이는 최소 속력값이 없기 때문에 생긴 문제로 두 번째 시연 때는 최소 속력을 넣고 시연을 했다. 시간이 감소 되었다.