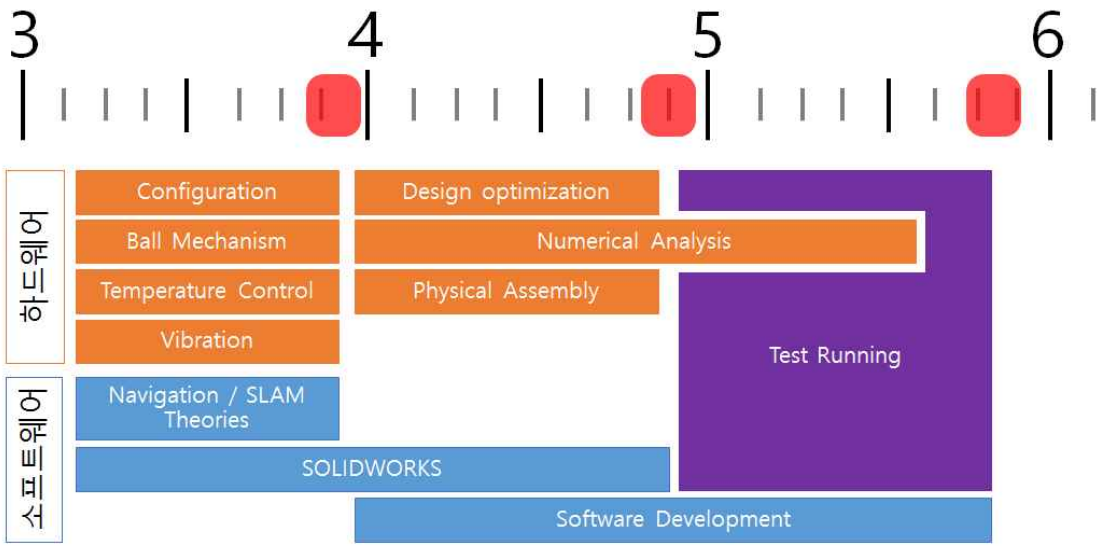
Final Report

<목차>

학번 : 20150516
이름 : 윤영호
조명 : 배달의 민족(7조)
지도교수 : 김진환 교수님

Ⅰ. 프로젝트 일정

<Gantt chart>



창의적 시스템 구현1의 최종목표를 수행하기 위해 위와 같이 프로젝트 일정을 구성하였다. ROS, Open CV, Labview, Solid works 총 네 개의 파트로 나누어진 이번 프로젝트를 크게 하드웨어와 소프트웨어로 나누어 프로젝트를 수행할 계획을 세웠다. 3, 4월 동안은 병렬적으로 각자의 파트에 충실한 기초 지식 습득이 이루어진다. 그 중, Ball mechanism, Temperature control과 같이 전원의 의견이 반영되면 성능이 향상될 수 있는 파트들은 조원 전체의 의견을 반영한 디자인이 이루어 질 수 있도록 진행하였다. 3, 4월 중으로 프로토타입이 완성하는 것을 목표로 한 후, 많은 test running을 통해서 하드웨어와 소프트웨어가 각자 수정할 수 있는 부분을 체크하여 성능을 향상시킬 계획을 세웠다. 또한, Test running동안 로봇의 성능이 향상되는 부분을 수치적으로 체크하여 프로젝트 진행동안 얼만큼의 향상이 이루어 졌는지 꾸준히 관찰할 계획을 세웠다.

Ⅱ. Progress report

Progress Report(3/19 ~ 3/25)

20150516 윤영호(파트 : Labview)

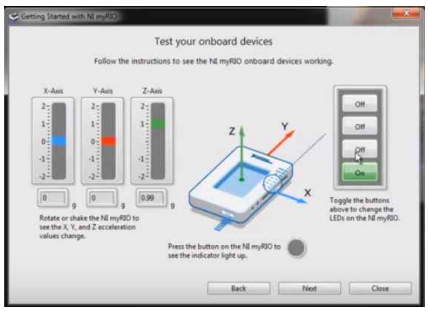
김진환교수님조(7조)

1. Labview 파트 과제수행

<Homework 1>

1) Launch myRIO example 수행을 통한 myRio 설치(Blink LEDs, measure accelerometer data example)

먼저 myRIO 설치를 위해 설치프로그램을 연 후, 교과서에 첨부된 다음 링크를 따라 myRIO test 수행을 위한 작업을 시행하였다. https://www.youtube.com/watch?v=CT7a3okNG8M myRIO를 pc와 연결한 후 설치프로그램을 열게 되면, 아래의 그림과 같이 myRIO의 상태를 test해볼 수 있는 화면이 뜬다. 이때 LED blink, accelerometer data를 수집해 볼 수 있다.



2) myRIO configuration을 통한 pc와 myRio연결 (setting name, wireless connection)

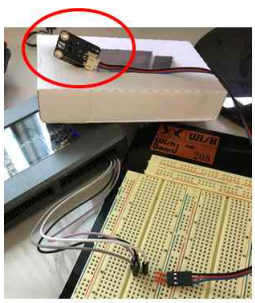
교재에 첨부되어 있는 https://www.youtube.com/watch?v=_ZAMC1XWR2Q 링크를 따라서 myRIO ip주소를 입력받은 후 와이파이연결을 통해 무선으로도 pc와 myRIO의 통신이 가능하도록 하였다. 또한, 연결이 잘 이루어졌는지 판별하기 위해 LED blink labview코드를 제작하여 실제로 동작하는지 알아보았다.

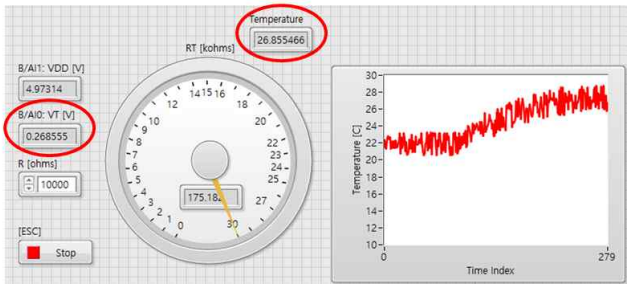
<Homework 2>

1) LED blink 순서 및 시간에 대한 코드 제작

Blink LED 1~4 in different time interval, LED 1~4 blink in order (LED1->LED2->LED3->LED4->LED1…)와 같이 순서와 시간을 중요시하는 알고리즘을 제작하여 LED가 깜빡이도록 설계를 해보았다.

2) accelometer data 수집 및, 온도센서 회로 설계를 통한 온도 data수집과 LED blink를 연계시켰다. 아래의 그림은 회로설계 제작사진 및 랩뷰 코드결과값을 보여주는 화면이다. 일정 가속도 이상, 또는 일정 온도이상의 데이터가 수집될 경우 LED가 켜지도록 설계함으로써 향후 로봇제작 및 실험 시에 안전상의 위험을 감지할 수 있는 코드를 제작하였다.





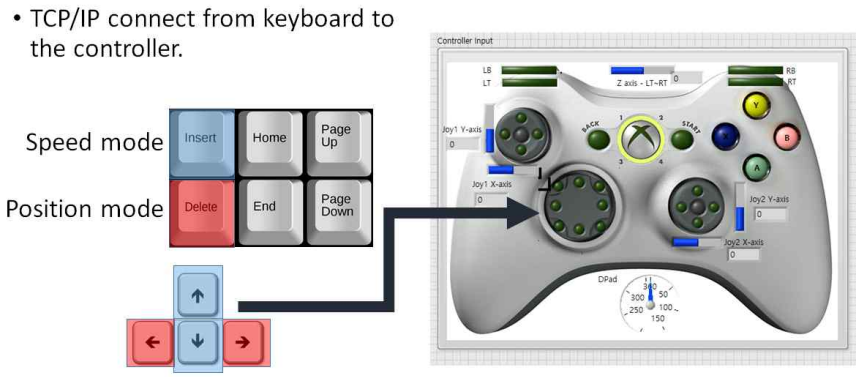
Progress Report(3/26 ~ 4/1)

1. Labview Part 과제수행

<Homework 3>

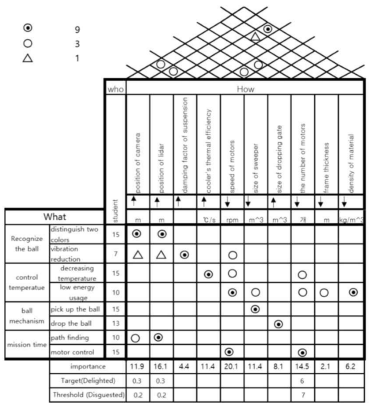
1) DXL example and TCP/IP example을 통한 모터구동

myRIO를 server로 두고, pc를 client로 하여 두 하드웨어가 tcp/ip 통신을 할 수 있도록 labview코드를 제작하였다. 또한, 그 코드에 모터구동을 첨가하여 모터제어 또한 가능하도록 하였다. 아래의 그림과 같이 두 모드, 즉 speed mode와 position mode로 전환될 수 있도록 키보드 자판에 insert와 delete를 각각 누를 경우 mode가 변형되도록 하였다. 또한, 방향키를 사용하여 speed mode에서 속도 증가 및 감소, position mode에서 angle의 크기 증가 및 감소가 이루어질 수 있도록 코드를 제작하였다.



2. 로봇 설계 spec 중요도 파악을 위한 QFD 제작

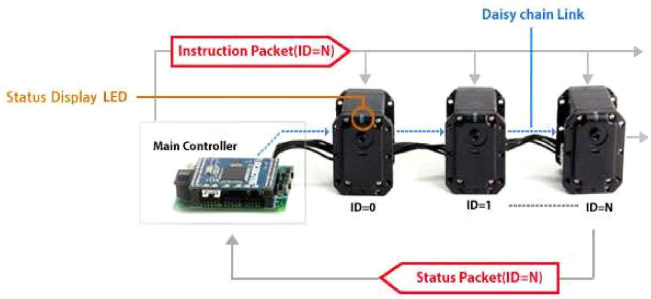
미션수행에 있어 수행해야할 criteria 및 engineering specification의 관계를 알아볼 수 있는 QFD를 제작하였다. 또한, 그 specification에 알맞은 target을 설정하였으며, 그 target이 이번 criteria 충족에 있어 얼마나 큰 비중을 차지하는지 알아낼 수 있었다. QFD의 결과, speed of motors spec이 target 충족에 가장 큰 중요도를 가진다. 또한, The number of motor에 대한 spec은 motor control 및 온도제어에 영향을 주며, target 충족에 큰 중요도를 가진다. 마지막으로, camera 및 lidar의 위치는 색상인식, 경로설정에 영향을 준다는 것을 알 수 있었다.



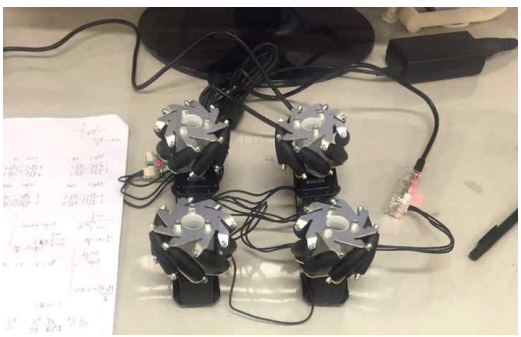
Progress Report(4/2 ~ 4/8)

1. multiple motor 사용을 위한 각 motor ID 지정

3월부터 시행된 과제에서는 모터를 단 하나만 사용하여 여러 가지 설정을 시도하였기에 나머지 3개의 모터를 사용할 필요가 없었다. 그러나 이제 4개의 모터를 제어하여 로봇이 이동할 수 있어야 하므로 4개의 모터를 모두 pc와 연결할 필요가 있다. 이를 위해 교재의 그림에서 나와 있듯이 각 모터들을 서로 선으로 연결해 시리얼 통신이 가능하도록 해야 한다. 아래의 그림은 2개 이상의 모터를 사용하기 위한 선연결 그림이다.

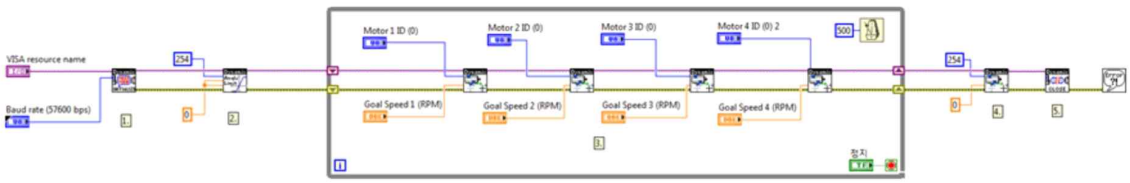


선연결 이전에 Roboplus를 통해 각 모터들의 ID를 부여받을 필요가 있다. 이를 위해 모터를 각각 하나씩 pc에 연결하여 서로 다른 ID를 부여 받았다. 또한 아래의 그림과 같이 4개를 한 번에 직렬로 연결하여 시리얼 통신이 가능하도록 하였다.



2. Labview example(multiple motor operation(Wheel mode)) 활용

4개의 모터를 연결한 후 여러개의 motor를 한번에 제어할 수 있는 labview example을 통해 한번에 네 개의 모터의 속도를 제어할 수 있도록 하였다. 여기서 유의할 점은 Roboplus에서 설정한 motor ID와 Labview 코드에서 설정한 ID가 서로 일치해야 하는 것이다. 이를 통해 각 모터가 동시간대에 각기 다른 속도로 회전할 수 있다.



Progress Report(4/9 ~ 4/15)

1. Mecanum wheel사용법 조사

현재 미션수행을 위해 필요한 사항 중 하나가 바로 시간을 단축하는 것이다. 이를 위해서는 로봇이 원하는 방향으로 쉽게 이동할 수 있게 하여 특정장소로 빠르게 이동하는 것이 시간단축과 직결된다. 이를 위해 홀로노믹 시스템, 즉 전방향으로 이동할 수 있는 시스템을 구축하기 위해 본 창시구1 과목에서는 Mecanum wheel을 사용한다. Mecanum wheel은 일반적인 바퀴와는 달리 사선으로 모터가 굴러갈 수 있도록 하여 네 바퀴마다 다른 벡터방향을 두어 어느 방향으로도 벡터 합을 통해 이동이 가능하도록 만들 수 있다. 정확한 사용방법을 알기 위해 논문을 조사하여 아래와 같이 어떤 모터방향과 속도를 가질 때 벡터합이 달라질 수 있는지에 대하여 연구하였다.

2. KINEMATIC

Figure 2 shows the configuration of a robot with four omnidirectional wheels.

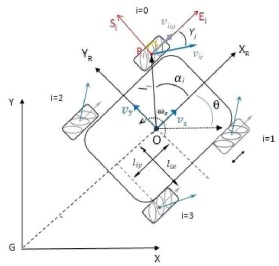
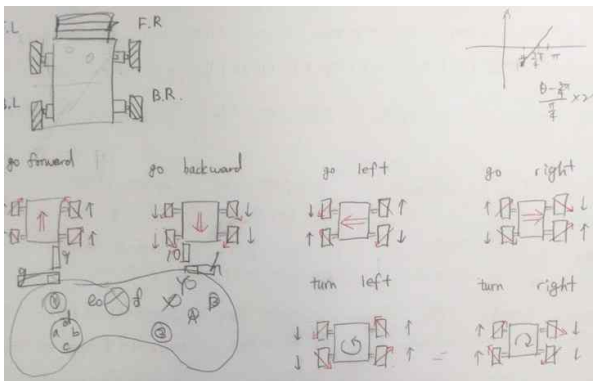


Fig 2: Wheels Configuration and Posture definition

2. Joystick array정보 분석 및 dynamixel 구동

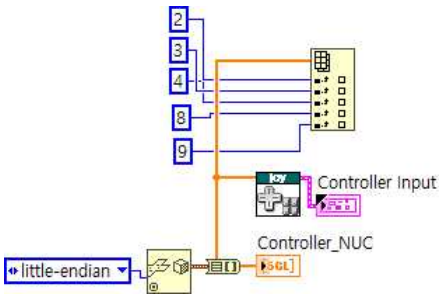
Macanum wheel에 대한 분석이 끝나면 이제 조이스틱에서 control하는 방향과 같은 방향으로 모터가 작동하도록 해야 한다. 그러기 위해서는 먼저 조이스틱에서 어떤 정보가 전달되는지부터 파악해야 한다. 조이스틱에서 전달되는 array에는 16개의 디지털신호와 8개의 아날로그신호로 구성되어 있다. 그 중, 8개의 아날로그신호를 통해 조이스틱의 방향과 세기를 알아낼 수 있다. 이 8개의 신호에는 x-y좌표값을 r-theta값으로 변환해준 정보까지 전달된다. 따라서 r값을 모터속도와 비례하게 만들고, theta값을 벡터방향과 비례하게 만들도록 알고리즘을 개발했다. 조이스틱과 통합하기 전, 키보드를 통해 전진, 후진, 왼쪽, 오른쪽, 왼쪽회전, 오른쪽회전에 대한 정보를 준 결과, 정확한 행동을 취하는 것을 확인할 수 있었다.



Progress Report(4/16 ~ 4/22)

1. xbox controller 정보처리 및 모터구동코드 제작

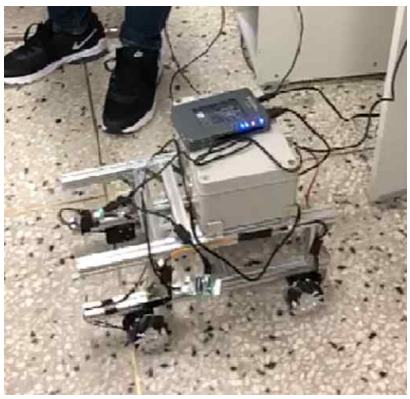
앞 주에 있었던 모터구동 메커니즘을 분석한 후 labview코드로 구현한 후, 실제로 차체에 모터를 달아 xbox controller로 구동해 보는 시간을 가졌다. 먼저, xbox controller의 정보를 얻기 위해서는 tcp/ip 통신을 통해 서버로 전달된 array정보를 추출하기 위한 코드가 필요하다. 이를 위해 아래 그림에 명시되어 있듯이, client가 준 정보가 들어 있는 주황색 선을 끌고와 원하는 배열값을 추출하는 코드를 짰다. 이를 통해 위에서부터 왼쪽 angle값, 왼쪽 magnitude 값, 오른쪽 x축 값, left top 아날로그 신호, right top 아날로그 신호 총 5개의 정보를 추출 할 수 있었다. 이 값 중 앞의 3개 값을 이용하여 차체가 모든 방향, 그리고 다양한 속도로 이동할 수 있는 코드를 제작했다.



2. 차체에 모터 부착 및 디버깅

앞선 코드제작을 모두 마친 후, 디버깅을 위해 차체에 모터를 직접 단 후 제어를 해보았다. 그 결과, 모터속도 제어는 매우 잘 이루어졌지만 방향제어가 이루어지지 않는 것을 확인할 수 있었다. 분석해본 결과, 모터의 방향이 좌우가 반대이기 때문에 전진을 하기 위해서는 두 개의 모터가 쌍으로 다른 방향으로 회전해야 한다는 것을 확인할 수 있었다. 이를 고려하여 다시 코드를 제작한 결과, xbox controller에서 방향을 가리킨 대로 모터가 전방향으로 움직일 수 있는 것을 쉽게 확인할 수 있었다.

또한, 그림에서는 myrio와 모터에 전원을 외부에서 공급해주고 있어 배터리에 연결하게 된다면 완벽한 무선제어가 가능할 것이다. 또한, 배터리를 사용할 때, 모터구동속도가 어디까지 제어가능한지까지 알아보아야 할 것이다.



Progress Report(4/23 ~ 4/29)

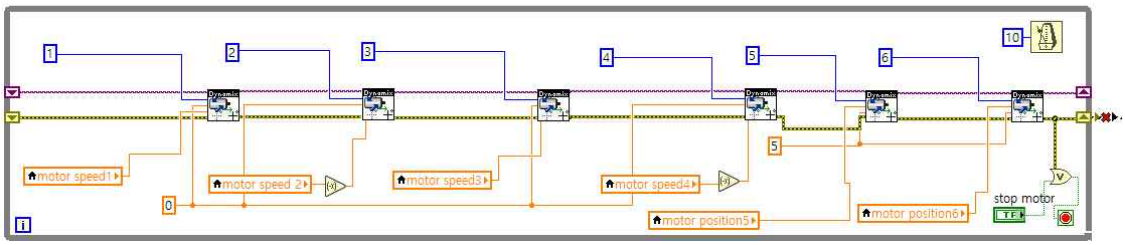
1. Sweeping을 위한 모터 구매

현재 로봇의 공 줍는 방법 및 빼내는 방법을 고려해 보았을 때, 두 개의 모터가 더 필요하다. 이를 위해 모터에 부착하게 될 플레이트의 무게를 고려하고 각도제어가 가능해야 한다는 점에서 아래의 로보티스 사의 MX-12W모터를 구매하였다. 앞에 장착하게 될 sweeper가 정밀한 각도제어가 필수적이며, 뒤편에 달게 될 문닫이용 플레이트가 여닫이가 가능해야 한다는 점을 고려해야 했다. 이를 위해 로보티스 서보모터가 모터 내부에 PID제어기가 장착되어 있어 쉽게 각도제어가 가능하다는 점을 염두해 두고 구매하였다.

모델명	MX-12W
무게	54.6 g
크기	32mm x 50mm x 40mm
기어비	32 : 1
Operation Voltage (V)	12
Stall Current (A)	0.6
No Load Speed (RPM)	470

2. 모터구동 및 서보모터 동시제어를 위한 코드 제작

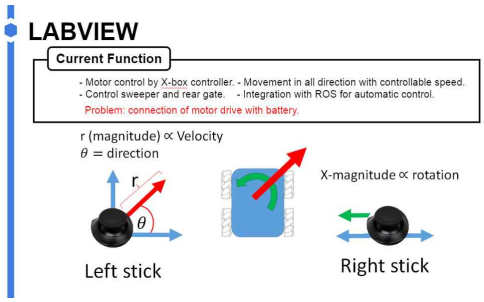
저번 주에 진행했던 모터구동을 위한 코드에서 앞서 구매한 두 개의 서보모터를 동시에 제어하기 위해 코드를 수정했다. 3월에 진행했던 첫 코드제작 방안은 3월에 연습했던 Joint mode 코드와 Wheel mode코드를 각각 while 루프에 돌리는 방안으로 제작하였다. 그 결과, 두 while루프는 따로 돌았지만, while 루프를 돌기위한 구속조건들이 각각의 루프에 영향을 주는 사태가 발생하였다. 이를 해결하기 위해 아래의 코드와 같이 wheel mode와 joint mode를 한 while루프에 넣어 동시에 제어할 수 있도록 하였다. 그렇게 한 결과, 모터구동도 제대로 이루어지면서 각도제어까지 완벽하게 할 수 있음을 알 수 있었다. 이를 통해 앞선 xbox controller array정보 중 left top 아날로그 신호, right top 아날로그 신호에 비례하는 각도를 두 서보모터에 주어 0~300도의 모든 각도로 제어가 가능하도록 할 수 있었다.

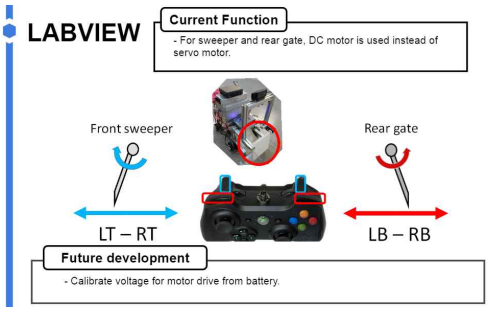


Progress Report(4/30 ~ 4/6)

1. 2차 발표를 위한 ppt 제작

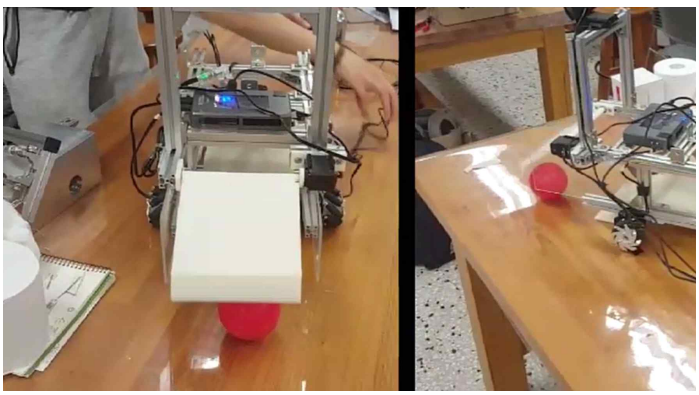
지난 일정 동안 Labview 파트에서 진행하였던 과정들을 설명하기 위해 아래와 같은 ppt를 제작하였다. 먼저, Macanum wheel을 사용하여 모든 방향으로 전진 및 후진, 회전이 가능하다는 것을 설명하기 위해 첫 번째 슬라이드와 같이 표현하였다. xbox-controller의 왼쪽 stick은 지정된 방향 및 세기를 입력받아 그 방향 및 속도로 차가 이동할 수 있도록 알고리즘을 구현했다는 것을 의미한다. 또한, 오른쪽 stick은 각 방향 및 세기를 입력받아 차가 왼쪽 및 오른쪽 방향으로 회전한다는 것을 의미한다. 두 번째 슬라이드는 차의 앞쪽에 위치한 sweeper 및 뒤쪽에 위치한 gate를 controller로 조작하는 방법을 설명하고 있다. controller 위쪽에 부착되어 있는 버튼이 analog input인 것을 이용하여 input의 세기를 각도로 전환하여 sweeper 및 gate의 각도를 조절할 수 있도록 하였다.





2. Auto ball tracking 및 Sweeping algrithm 구현

TCP-IP통신을 통해 받는 data의 24개의 집합으로 이루어진 array에 대한 정보를 ROS파트에게 전달하였다. 그 후, 어떤 data가 전송될 때 차가 어떤 행동을 취하는 지에 대하여 정리한 후, ball detecting을 연동하여 차가 자율적으로 빨간공까지 가게 한 후 공을 집을 수 있도록 알고리즘을 구현하였다. 카메라의 위치 및 sweeper의 위치를 고려했을 때, 빨간공이 매우 가깝게 되면 화면에서 아랫부분에서 사라진다는 것을 고려하여 공이 보이지 않게 되면 적절한 속도 및 각도를 고려하여 공이 보이지 않아도 집을 수 있도록 했다.



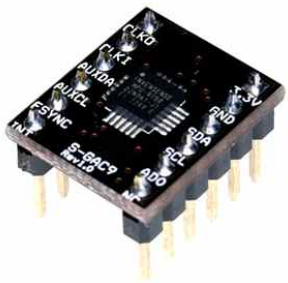
Progress Report(5/7 ~ 5/13)

1. 2차 발표 후 피드백 분석

2차 발표 결과, 온도상승에 대한 정밀한 측정이 부족했다는 부분을 지적받았다. 대부분의 조에서 나타나는 현상이듯이, 보통 시스템을 개발하면 컨버터에서 온도가 가장 많이 상승하는 것이 기본적이다. 그러나, 우리 조에서는 5분간 로봇을 실행시켜 본 결과, 모터에서 가장 큰 온도상승이 일어나는 것으로 추측되었다. 이에 대하여 보다 정확한 측정이 필요할 것이며 그 원인 및 컨버터 구매 시 고려했던 온도상승 스펙을 추가로 발표할 필요가 있을 것으로 보인다.

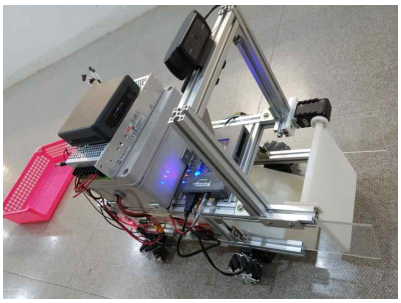
2. 문제점 개선을 위한 추가구매

차체 뒤에 달려 있는 gate가 부착된 모터가 오작동을 일으켜 현재 A/S신청을 해둔 상태이다. 보다 빠른 실험이 필요하므로 모터를 추가구매 하였다. 또한, 모터와 Wheel사이가 차체의 무게에 의해 휘어지는 현상이 발생하여 모터에 무리가 갈 것으로 추정되어 Ball wheel을 추가구매하여 각 Wheel에 작용하는 수직항력을 분배시키도록 하였다. 마지막으로 Compass를 구매하여 자율주행 시 로봇이 보다 많은 정보를 습득할 수 있도록 하였다.



3. 복잡한 배선 문제 개선

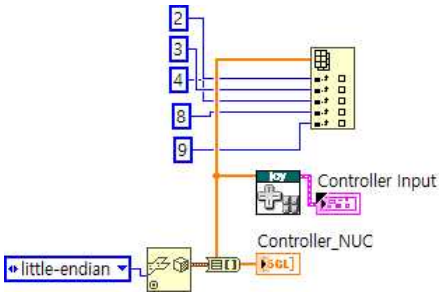
다 수의 device를 사용하여 복잡한 배선들이 추가되면서 전선들이 엉키는 문제가 발생하였다. 전선이 끊기는 것은 매우 치명적이므로 긴 선들을 다시 짧게 제작하여 전선이 꼬이는 사고를 막았다. 특히, 로봇 아래로 오가는 전선들의 길이를 축소시켜 바퀴에 걸리는 일이 발생하지 않도록 하여 로봇의 움직임에 문제가 없도록 하였다.



Progress Report(5/14 ~ 5/20)

1. Labview code 개선

시스템 안정화를 위해 Labview code를 일부 개선할 필요가 있음을 인지했다. 마이리오와 nuc의 통신과정 중 통신이 장시간 동안 지속되면 일부 신호가 잠깐 끊겨 0데이터를 보내는 현상이 발생했다. 그렇게 될 경우 앞 스위퍼와 뒤에 부착된 스위퍼가 각을 유지하지 못하고 다른 각으로 바뀌어 버리는 사태가 나타났다. 이를 막기 위해 initial position을 ros데이터 신호가 0일 때로 맞추도록 하여 갑자기 신호가 끊기더라도 스위퍼가 움직이지 않도록 하였다. 그렇게 하기 위해서는 기존에 오던 데이터인 1을 0으로 받을 수 있도록 하고, 0으로 오는 데이터를 1로 받을 수 있도록 calibration할 필요가 있었다. 이를 위해, -x+1의 함수를 받는 데이터신호 뒤에 달아 앞에서 원하는 데이터로 변경될 수 있도록 하였다. 그 결과, 기존의 ros 신호를 받더라도 원하는 각으로 작동이 잘 이루어졌으며, 신호를 받지 못해 0데이터가 들어오는 순간에도 스위퍼가 움직이지 않게 할 수 있었다.



2. IMU9250 compass신호 사용

Compass정보를 얻기 위해 다양한 센서를 알아보던 중, ROS를 통해 바로 신호를 얻을 수 있는 IMU센서를 구매하였다. 공을 모두 수집한 후, 초록공으로 돌아오기 위해서는 처음 있었던 위치로 가는 정확한 방향을 알아야 한다. 이를 위해, 나침반신호를 받아 정확한 방향으로 돌아올 수 있는 알고리즘을 고안하였다. 움직이기 전, 나침반 신호를 받아 저장한다. 그 후, 공을 다 집고 나면 그 방향의 180도방향으로 로봇이 회전한 후, 두 초록공의 중심을 찾아 올 수 있도록 하였다. 현재상태는 나침반신호를 통해 바구니방향을 잘 찾아내었지만, 오는 도중 중간지점에서 잠시 멈추는 에러가 발생하여 그 부분을 수정할 필요가 있을 것으로 보인다.

Progress Report(5/21 ~ 5/27)

1. 열화상카메라를 이용한 열 분석

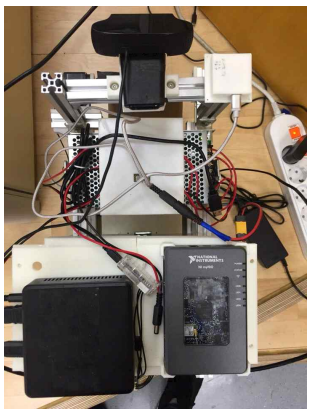
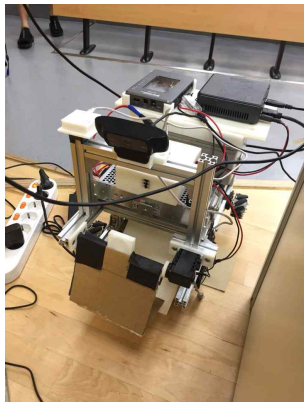
공동강의실에 비치된 열화상카메라를 이용하여 현재 로봇의 열 관련 스펙을 알아보았다. 로봇을 5분간 가동한 후, 각 파트가 어떤 온도를 가지는지 알아보았으며 아래의 사진이 열화상카메라를 캡쳐한 결과이다.

왼쪽 사진은 로봇을 위에서 촬영한 사진이다. 컨버터와 배터리에서 많은 열이 발생할 것이라는 예상과는 달리, 앞 스위퍼를 작동시키는 모터에서 가장 높은 온도가 나타난다는 것을 확인할 수 있었다. 그러나 30도 내외의 온도를 가져 위험한 정도의 온도까지 올라가는 것은 피할 수 있었다. 다른 조와는 다르게 컨버터의 온도가 올라가지 않는 것은 높은 100W의 전력을 낼 수 있어 열용량이 크기 때문으로 추정할 수 있다. 따라서 열용량이 적은 컨버터를 여러 개 사용한 조와는 반대로 컨버터가 큰 열을 발생시키지 않은 것을 알 수 있다.

2. 로봇 부품 재배치 및 스위퍼 구조 개선

기존 로봇은 컨버터, nuc, 마이리오가 한 곳에 있어 열방출이 다소 잘 이루어지지 않았다. 이를 개선하기 위해 3D 프린터로 틀을 만든 후 각 부품들을 분산시켜 열이 잘 방출될 수 있도록 재배치시켰다.

또한, 앞 스위퍼가 파란 공과 계속해서 끼이는 현상을 막기 위해 기존의 스위퍼보다 더 꺾인 구조로 변경하였다. 그 결과, 파란 공이 더 이상 끼이지 않앗 공을 보다 정확히 주울 수 있었다.

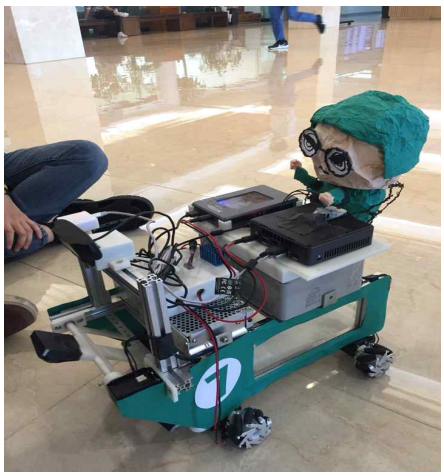
Ⅲ. 최종 데모 및 결과

일시 : 6/1 (금)
장소 : ME building Rm. #1501

최종 데모에서는 각 조별로 한 팀씩 궁동강의실에서 데모를 하는 것으로 진행되었다. 프로젝트의 최종목표인 파란 공 세 개를 주어 오는 것을 성공적으로 마무리하기 위해 마지막까지 로봇의 성능을 유지하기 위한 최종 체크가 이루어졌다. 먼저, 각 부품들의 볼트, 너트가 잘 결합되어 있는지, 배터리가 충분히 충전되어 있는지, Myrio, NUC 등 소프트웨어 물품들의 전원이 잘 켜지는 지 등을 데모 전에 체크하였다.

1차 결과, 약 2분 만에 세 개의 공을 바구니 앞까지 가져오는 것은 성공하였으나, 바구니 앞에서 서버의 지연으로 인해 뒤로 가는 거리가 약간 줄어들어 바구니에 담는 것을 실패하였다. 2차 결과에서도 같은 원인으로 공을 줍는 과정에서 서버의 지연으로 인해 공 앞까지 가지 못하여 하나의 공을 놓쳤다. 최종 결과는 2분 47초로 2개의 공을 집는 것으로 만족했다.

서버와 클라이언트 사이의 통신이 끊기는 것을 해결하지 못한 점이 목표성공에 큰 차질을 남긴 것으로 생각되었다. 따라서, 앞으로의 로봇개발 시 다른 프로그램 사이에서는 통신이 원활하게 이루어질 수 있는 프로그램 구성이 이루어져야 하며, WIFI 개별화를 통해 다른 통신으로 인해 장애를 일으키지 않도록 로봇을 구축해야 할 것으로 생각되어진다.



<로봇 최종본>

<Team 단체사진>