
Final Report
: Making Robot for Mars Exploration

Group	1
	하재민 / 2013314714
	안찬우 / 2013311067
Name	전현준 / 2012312582
& I.N.	황성택 / 2013312819
	이반석 / 2012312854
	정상익 / 2012311126
Class	EME3057(42)
Affiliation	Sungkyunkwan University, School of Mechanical Engineering
Date	2017. 6. 17.

<TABLE OF CONTENTS>

1. Team Story & Introduction	-----1
2. Preliminary Process	
2.1 Theory	-----4
2.2 Mission Analysis	-----7
2.3 Customer Requirements	-----10
2.4 Technical Requirements	-----11
2.5 Relationship Map	-----12
2.6 Technical Requirements quantification	-----13
2.7 QFD	-----14
2.8 CTQ	-----15
2.9 Analysis of QFD & CTQ	-----16
3. Concept Design	-----17
4. Robot Making & Detail Design	-----22
5. Algorithm & Code	-----26
6. Conclusion	-----39
7. Reference	-----41

1. Team Story & Introduction



공대의 역경 속에 탄생한
“율전 THE SIX”



그리고 그들의 로봇
“율전동 여섯바퀴”



MISSION MEMBER

WORK DIVISION



안찬우

- 탁월한 조종수
- 코드 설계
- 모의 맵 제작
- 로봇 제작
- 로봇 디자인
- 리포트 작성



이반석

- 디자인의 시초
- 로봇 제작
- 코드 설계
- 로봇 디자인
- 리포트 작성



전현준

- 아이디어 뱅크
- 코드 설계
- 로봇 제작
- 로봇 디자인
- 리포트 작성



정상익

- 최적화 장인
- 로봇 제작
- 모의 맵 제작
- 코드 설계
- 로봇 디자인
- 리포트 작성



하재민

- 리포트 달인(조장)
- 모의 맵 제작
- 로봇 제작
- 코드 보완, 수정
- 로봇 디자인
- 리포트 작성



황성택

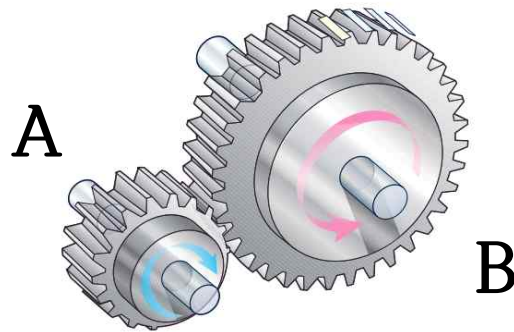
- 로봇 개조대가
- 모의 맵 제작
- 로봇 제작
- 코드 설계
- 로봇 디자인
- 리포트 작성

2. Preliminary Process

2.1 Theory

2.1.1 Gear

기어는 간단한 구조로 동력을 전달함에 있어 동력손실이 적고 회전의 전달을 정확하게 하여 많은 기계구조에 널리 사용되고 있다. 기어의 속도와 힘은 기어수를 이용하여 조절하게 되는데 다음 식을 통해 알아보자.



<Figure. 2-1 Gear[2]>

A기어의 기어수(N_A)와 B기어의 기어수(N_B) 비는 두 기어의 각속도의 비와 동일하다.

$$\frac{N_A}{N_B} = \frac{w_B}{w_A} \quad (1-1)$$

그리고 두 기어의 한 일은 같기 때문에 다음과 같은 식으로 표현될 수 있다.

$$T_A w_A = T_B w_B \quad (1-2)$$

위 두식을 정리하면 우리는 기어수가 작은 기어에서 기어수가 큰 기어를 연결시키면 더 큰 힘을 얻을 수 있다는 것을 확인 할 수 있다.

$$\frac{N_A}{N_B} = \frac{w_B}{w_A} = \frac{T_A}{T_B} \quad (1-3)$$

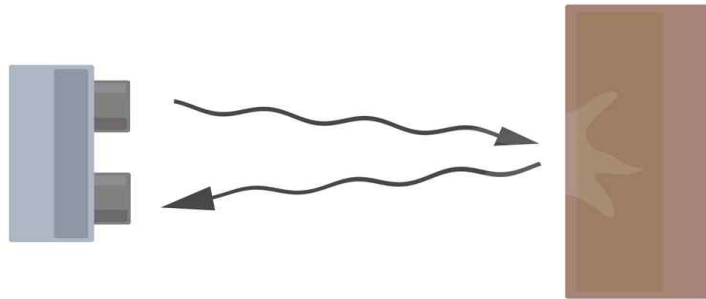
모터에 기어비가 작은 기어를 연결하고 기어비가 큰 기어를 연결시켜 캐터필더에 연결시키면 캐터필더가 더 큰 힘을 얻을 수 있게 된다. 기어비를 통해 더 큰 힘을 얻을 수 있지만 대신에 로봇의 속도는 줄어들 수밖에 없다.[2]

2.1.2 Sonar sensor

적외선 센서는 음파를 이용해서 물체와의 거리를 측정하는 장치이다.[3] 특정 주파수의 음파를 발생시켜 반사된 음파를 감지해서 거리를 측정한다. 음파가 발생된 시간과 물체에 반사되어 다시 측정될 때까지 소요된 시간과 음파의 속도를 이용하는 것이다. 가령 공기 중에서 소리는 344m/s로 이동하므로 음파가 이동한 총 거리를 측정한 시간에 344m/s를 곱해서 파악할 수 있다. 이 때 음파가 이동한 총 거리는 센서와 물체 사이의 거리에 2배이다. 그래서 물체까지 거리는 음파가 이동한 총 거리에 절반을 나누면 알 수 있다.

$$\text{거리} = \frac{\text{음속} \times \text{총 소요시간}}{2}$$

초음파 센서가 감지할 수 없는 물체도 있다. 이는 물체가 뾰족하거나 혹은 음파가 물체에 반사되었지만 심하게 왜곡되는 경우이다. 혹은 물체가 너무 작아 충분히 음파를 반사시키지 못해 센서에서 감지를 못하는 경우도 있다. 옷, 카펫처럼 음파를 모두 흡수해서 센서가 정확하게 찾지 못하는 경우도 있다. 이 모두 초음파 센서를 사용하는 로봇을 디자인, 프로그래밍할 때 고려해야하는 중요한 요소이다.



<Figure. 2-1 Diagram of the basic ultrasonic sensor operation>

2.1.3 Motor

모터는 전력을 이용해서 회전운동의 힘을 얻는 기계이다.[4] 즉 전류가 흐르는 도체가 자기장 속에서 받는 힘을 이용하여 전기에너지를 역학적 에너지로 바꾸는 장치이다. 전원의 종류에 따라 직류 전동기와 교류 전동기로 분류되며, 교류 전동기는 다시 3상 교류용과 단상 교류용으로 구분된다.

모터는 플레밍의 왼손 법칙에 따라 전자기장 에너지를 운동에너지로 변환해준다.[5] 전류가 흐르는 도체를 자기장 속에 놓으면 자기장의 방향에 수직인 방향으로 전자기적인 힘(로렌츠 힘)이 발생한다. 즉, 자기장 속에 있는 도선에 전류가 흐르면 움직이는 전하에 작용하는 로렌츠힘에 의해 도선도 힘을 받는다. 이것이 전기에너지를 사용하여 회전운동을 하는 전동기의 기본 원리가 된다.

2.1.4 Center of mass

질량중심은 물체를 이루는 각 입자의 질량에 위치 벡터를 곱한 것을 모두 더해서 물체의 전체 질량으로 나누어 구한 값으로 물체를 구성하는 질량을 가진 모든 입자들의 평균적인 위치이다[6]. 그래서 중력장 내에서 물체가 받는 중력의 원천이 되는 지점인 질량중심이 곧 무게중심이 될 수 있다.

2.1.5. Caterpillar(Continuous tracks)

긴 바퀴로 된 주철제(鑄鐵製) 벨트를 바퀴에 걸고 돌리면 벨트가 움직여 차체가 움직이도록 하는 장치로, 캐터필러라고도 한다.[7] 동륜(動輪)을 축에 고정시켜 놓고, 종륜(從輪)은 무한궤도가 늘어지지 않게 해 축을 조절할 수 있도록 되어 있다.

궤도에 사용되는 밴드는 일반적으로 군용 차량 및 중장비의 경우 모듈러 강판으로, 경량의 농기구 및 건설 차량의 경우 강철 와이어로 보강된 합성 고무로 만들어진다.

트랙의 넓은 표면적은 차량의 무게를 쇠 혹은 고무 타이어보다 고르게 무게를 분산시키므로 무한 궤도가 갖춰진 차량이 진흙과 같은 상대적으로 부드러운 지면에 가라 앉을 가능성이 적다. 즉, 바퀴만 있는 차에 비하면 속도는 느리지만, 지면에 닿는 면이 넓으므로 요철이 심한 도로나 진흙 바닥에서도 자유로이 달릴 수 있고, 회전 속도를 조정해 방향 전환을 자유로이 할 수도 있다. 무한 궤도는 오늘날 불도저, 굴삭기, 탱크 및 트랙터를 비롯한 다양한 차량에 일반적으로 사용되고 있다.

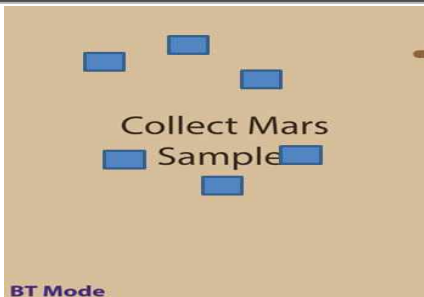


<Figure 2-2 Caterpillar>

2.2 Analysis of Mission

Mission 1	Scar of Mars
	
핵심 내용	<ul style="list-style-type: none"> -미로 지역을 장애물과 접촉 없이 지나간다. -각 장애물의 크기는 10cm이다. -장애물이 한 칸 이상 밀려나면 감점 및 재시작을 하게 된다. -장애물 조합은 총 4가지 중 무작위로 결정된다. -자동 모드로 이동한다. -로봇은 반드시 센서를 이용하여 종료 선에 도달하면 다음 미션을 수행해야한다. -로봇은 반드시 2번째 열에서 정면을 바라보고 출발해야한다
해결 방안	<p>초음파 센서를 이용해서 장애물의 유무를 파악하고, 유연하게 통과해야 한다. 먼저 컵의 유무를 파악하기 위해 센서가 인식하는 문턱값을 알맞게 설정해야 한다. 코드를 설계할 때 장애물 조합이 총 4가지이므로 이를 모두 고려해야한다. 회전 시 장애물을 건드리면 안되므로 장애물을 인식하는 센서의 위치, 회전 방식과 지점, 로봇의 크기와 장애물 사이의 폭을 고려해야 한다. 장애물을 모두 지나온 후, 빛 센서를 이용해서 미션 종료 지점에 해당하는 검은색 라인에 정지해야 한다. 정지 후에는 수동 모드로 전환할 수 있어야 한다.</p>

<Table. 2-1 Analysis of Mission 1>

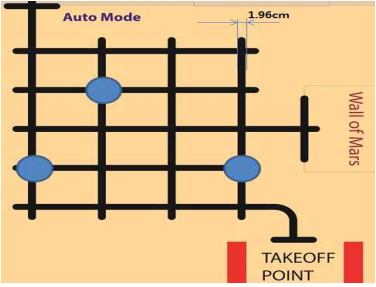
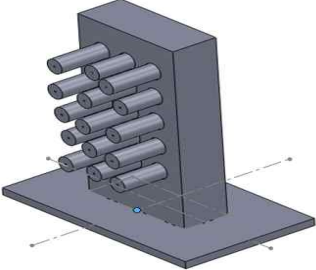
Mission 2	Collect Samples
	
핵심 내용	<ul style="list-style-type: none"> -지도 전체에 흩어져 있는 블록을 수집하시오. -6개의 샘플이 맵 전체에 흩어져 있다.

	<p>-흩어진 샘플을 원하는 만큼 획득한 후 다음 스테이지로 이동한다.</p> <p>-수동모드로 이동한다.</p>
해결 전략	<p>최단시간에 많은 샘플을 수집하는 것이 목표이다. 수동 모드로 진행되므로 로봇의 수동 조작 방식을 세분화한다면 더욱 정밀한 조작이 가능할 것이다. 또한 수집 전후의 로봇의 이동 속도와 샘플을 수집하는 속도가 빠를수록 좋다. 수집된 샘플이 넓고 안전한 공간에 보관되도록 해야한다.</p>

<Table. 2-2 Analysis of Mission 2>

Mission 3	Sand Swamp / Hall of Mars
	
핵심 내용	<p>-Hall of Mars와 Sand Swamp가 존재하며 Sand Swamp는 반드시 수행하여야 한다.</p> <p>-수동모드로 진행한다</p> <p>-Hall of mars와 Sand Swamp의 폭은 30cm이다.</p> <p>-통과할 경우 기본점수와 획득한 샘플 수에 따른 추가 점수를 얻는다</p> <p>-Sand Swamp의 낭떠러지를 지나 “Escape” 지역에 도착하면 다음 스테이지로 이동한다.</p>
해결 방안	<p>로봇이 경사면을 올라갈 수 있을만큼의 파워, 힘이 있어야한다. 그리고 차체(바퀴)의 높이가 경사의 문턱을 넘는데 크게 방해해서는 안 된다. 언덕을 올라가거나 내려갈 때, 그리고 낭떠러지에서 떨어질 때 로봇이 전복 및 파손이 안되며 수집된 샘플이 빠져나가서도 안 된다. 장애물을 hall 충분히 채워서 지나가거나 혹은 채우지 않고서 잘 넘어갈 수 있어야한다. 원통의 회전을 무시할만큼의 마찰 면적을 가져야 한다.</p>

<Table. 2-3 Analysis of Mission 3>

Mission 4	Line Tracer / Wall of Mars
	 
핵심 내용	<ul style="list-style-type: none"> -자동 모드로 라인 트레이싱하여 이륙 지점(TAKEOFF POINT)으로 이동한다. -검은 선을 따라 가야 한다. -1개의 경유 지점이 존재하며, 선택 미션인 Wall of Mars를 수행할 수 있다. -자동 모드로 수행해야 한다. -도착지점으로 로봇이 들어오면 경기를 종료하며 반드시 센서를 이용하여 라인을 따라 들어와야한다.
	<ul style="list-style-type: none"> -최대한 많은 기둥을 집어 놓아야한다. -크기는 130x250x50mm (w x h x d)이다 -기둥을 4cm 이상 밀어 놓아야 인정된다. -기둥의 총 개수는 15개이다. -자동모드로 수행해야 한다.
해결 방안	<p>라인 트레이싱을 위해 센서의 문턱값을 적절하게 선택해서 검은 선과 바탕을 잘 구분할 수 있어야 한다. 적절한 경로 선택과 더불어 교차 지점을 정확하게 인식해야 한다. 회전을 하는 경우, 회전이 끝난 뒤 센서를 검은 선에 다시 놓이도록 해야한다. 원기둥을 충분히 밀 수 있을만큼 속력을 내고 거리를 이동해야 한다.</p>

<Table. 2-4 Analysis of Mission 4>

2.3 Customer Requirements

CUSTOMER REQUIREMENTS	IMPORTANCE
MISSION 1 – Scar of Mars	
작은 차체의 크기(회전 용이)	3
장애물의 인식	5
센서의 민감도	5
적절한 회전 방법	4
인식 시간	3
MISSION 2 – Collect Samples	
적절한 수집능력	5
수집공간의 안정성 및 크기	3
조작의 용이성	4
이동 및 회전속도	4
MISSION 3 – Sand Swamp / Hall of mars	
차체의 안정성	4
장애물 제거 능력	3
충분한 출력 마찰	5
수집 샘플 보관의 안정성	3
MISSION 4 – Line Tracer / Wall of Mars	
라인 트레이싱	5
교차점에서 회전	4
차체벽의 크기	3
벽을 밀 수 있는 로봇의 힘	3
교차점 인식	5
COMMON FEATURES	
적절한 디자인	4
내구도	4
속도	3
배터리에 대한 적은 영향	4
수정의 용이	3

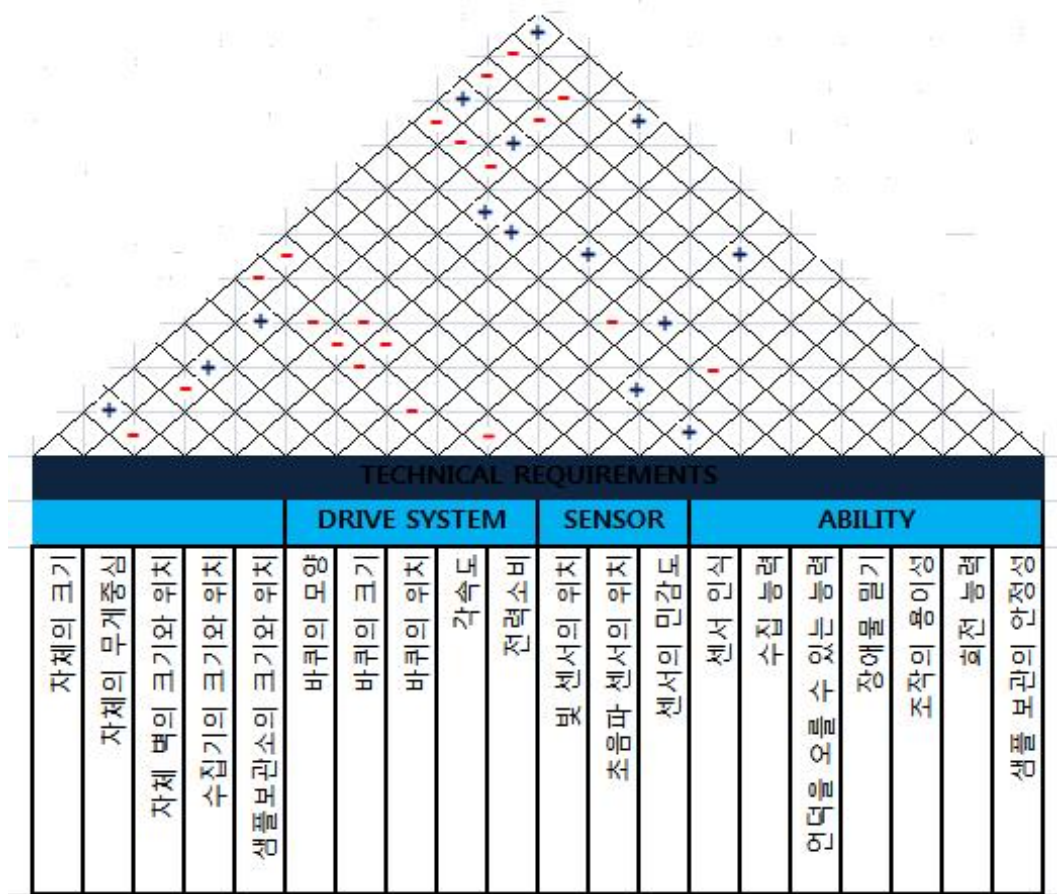
<Table. 2-5 Technical requirement>

2.4. Technical Requirement

TECHNICAL REQUIREMENTS
BODY
차체의 크기
차체의 무게중심
차체 벽의 크기와 위치
수집기의 크기 위치
샘플보관소의 크기와 위치
DRIVE SYSTEM
바퀴의 모양
바퀴의 크기
바퀴의 위치
각속도
전력소비
SENSOR
빛 센서의 위치
초음파 센서의 위치
센서의 민감도
ABILITY
센서 인식
수집 능력
언덕을 오를 수 있는 능력
장애물 밀기
조작의 용이성
회전 능력
샘플 보관의 안정성

<Table. 2-6 Technical requirement>

2.5. Relation Map



<Figure. 2-3 Relation map>

2.6. Technical Requirements Quantification

TECHNICAL REQUIREMENTS QUANTIFICATION	UNIT
BODY	
차체의 길이	cm
차체의 폭	cm
차체의 높이	cm
차체의 무게중심의 위치	cm
장애물용 벽의 크기	cm
장애물용 벽의 위치	cm
차체의 무게	N
DRIVING SYSTEM	
이동속도	m/s
모터의 개수	cm
바퀴의 크기	cm
각속도	Rad/s
출력	W
SENSOR SYSTEM	
빛 센서의 높이	cm
초음파 센서의 위치	cm
빛 센서의 민감도	LUX
초음파 센서의 민감도	cm
COLLECTING SYSTEM	
수집기의 크기	cm
수집기의 높이	cm
수집기의 속도	Rad/s
수집기의 무게	N
샘플보관소의 부피	cm³
샘플보관소의 높이	cm

<Table. 2-7 Technical requirements quantification>

2.7. QFD

High = 5 Intermediate = 3 Low = 1		Technical Importance	Technical Requirments																			
			Body					Drive system					Sensor		Ability							
			차체의 크기	차체의 무게중심	차체 밖의 크기와 위치	수집기의 크기와 위치	샘플보관소의 크기와 위치	버퀴의 모양	버퀴의 크기	버퀴의 위치	각속도	전력소비	빛 센서의 위치	초음파 센서의 위치	센서의 민감도	센서 인식	수집 능력	장애물 오를 수 있는 능력	장애물 밀기	조각의 용이성	회전 능력	샘플 보관의 안정성
Customer Requirnts																						
MISSION 1 Scar of Mars	작은 차체의 크기 (회전 용이)	3	5	5															3	3	1	
	장애물의 인식	5												5	5	5						
	센서의 민감도	5											3	3	5	5						
	적절한 회전 방법	4	1					1	3	1	3									5		
MISSION 2 Collect Samples	인식 시간	3									1			5	5	5						
	적절한 수집능력	5				5	5										5		3		5	
	수집공간의 안정성 및 크기	4	3	3	3	3	5										3				5	
MISSION 3 Sand Swamp & Hall of mars	조작의 용이성	4	3	3							5								5	5		
	차체의 안정성	4	5	5	3	3	3		3													
	장애물 제거 능력	3			5							3						5				
MISSION 4 Line Tracer & Wall of Mars	충분한 출력	5										5					5					
	라인 트레이싱	5											5		5	5						
	교차점 인식과 회전	4											5		5	5						
	차체밖의 크기	3			5													5				
COMMON FEATURES	벽을 밀 수 있는 로봇의 힘	3										5										
	적절한 디자인	3	3	3	3	3	3						1	1							3	
	내구도	4	5	5				3	3	3												
	속도	4						5			5	5										
	배터리에 대한 적은 영향	4										3										
	수정의 용이	3	1	3	3	3	3															
	Point		95	97	72	67	75	36	36	16	55	81	63	58	110	110	37	25	30	44	49	57
	Rank		4	3	7	8	6	16	16	20	12	5	9	10	1	1	15	19	18	14	13	11

<Figure. 2-4 QFD>

2.8. CTQ

<div>High = 5</div> <div>Intermediate = 3</div> <div>Low = 1</div>			Technical Importance	Part Characteristics																				
				Driving Part						Collector					Board			Sensor					Basket	
				모터의 수	바퀴의 지름	바퀴의 마찰계수	바퀴의 종류	무게중심의 위치	모터의 힘	수집기의 크기	무게중심의 위치	수집하는 속도	수집기의 높이	수집기의 무게	판의 높이	판의 폭	판의 길이	센서의 반응 시간	조음파 센서의 측정범위	조음파 센서의 높이	빛 센서의 높이	두 빛 센서의 거리	샘플의 수용량	바구니 입구의 크기
Technical Requirents																								
Body	차체의 크기	95	5	1			3		5	1		1		1	1	1						1		
	차체의 무게중심	97					5			3				1	1	1								
	차체 벽의 크기와 위치	72					3						3	3	3									
	수집기의 크기 위치	67		3			3		5	5	5	5	5								1			
	샘플보관소의 크기와 위치	75		1			3		1	3		3	1								5			
Drive system	바퀴의 모양	36	1	3				5			5										5			
	바퀴의 크기	36			5	3	3	3		5		1						1	1					
	바퀴의 위치	16		3		3	5																	
	각속도	55			5	5		5																
	전력소비	81	5		3	5		5																
Sensor	빛 센서의 위치	63															3			5	5			
	조음파 센서의 위치	58															3	5	5					
	센서의 민감도	110															3	3	5	5	5			
Ability	센서 인식	110															5	5	5	5	5			
	수집 능력	37							3		5	5									5	5		
	언덕을 오를 수 있는 능력	25	3	1	5	3	5	5																
	장애물 밀기	30												5	5	5								
	조작의 용이성	44			3	3	5	5																
	회전 능력	49		1	1	1		3																
	샘플 보관의 안정성	57																			5	5		
	Point		991	601	1004	1092	1945	1460	996	1126	700	876	410	558	558	558	1243	1170	1426	1451	1415	1025	632	
	Rank		13	17	11	9	1	2	12	8	15	14	21	18	18	18	6	7	4	3	5	10	16	

<Figure. 2-5 CTQ>

2.9 Analysis of CTQ, QFD

2.8.1 QFD

Ranking	
1	센서의 민감도
1	센서의 인식
3	차체의 무게중심
4	차체의 크기
5	전력소비

<Table 2-8 Ranking of QFD>

점수의 순서를 보면 센서의 민감도와 센서인식이 가장 중요하게 나왔다. 이는 초음파 센서와 빛 센서가 두 개의 미션을 해결하는데 있어서 중요한 역할을 하기 때문이다. 우리는 센서가 주어진 상황에 인식을 정확히 수행하여 미션을 수행하도록 할 수 있게 로봇을 설계하여야 한다. 그리고 그 다음으로 중요한 부분이 차체의 무게중심과 크기 이다. 무게중심이 맞지 않으면 물체의 구동이 불안정해지고 언덕을 올라갈 때 뒤로 넘어가거나 내려올 때 앞으로 넘어갈 수 있다. 그리고 작은 크기로 틈 사이를 통과하여야 하기 때문에 적당한 크기와 그 크기에서 가질 수 있는 최고의 무게중심을 잡아야 할 것이다.

2.8.2 CTQ

Ranking	
1	무게 중심의 위치
2	모터의 힘
3	빛센서의 높이
4	초음파 센서의 높이
5	두 빛센서의 높이

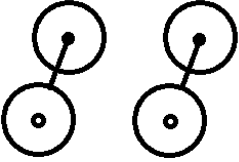
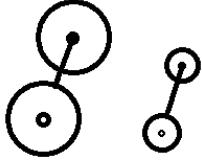
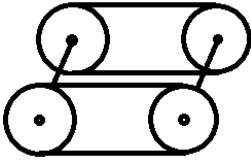
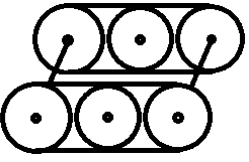
<Table 2-9 Ranking of CTQ>

CTQ에서도 QFD와 비슷한 결과가 나타났다. CTQ에서는 무게중심의 위치가 가장 높은 점수를 받게 되었다. 미션을 해결 하는데 있어 차체의 안정성과 구동성에 영향을 끼치기 때문에 무게중심이 잘 잡힌 로봇은 전체적으로 안전한 시스템으로 구동 될 것이다.[8] 그 뒤로 높은 점수가 모터의 출력인데 높은 모터의 출력이 나와야지 험난한 지형을 통과 할 수 있을 것이다. 그리고 우리가 설계한 프로그램을 정확히 구동시키기 위해서도 항상 최고의 출력을 유지 하여만 한다. 그리고 뒤를 잇는 게 초음파 센서와 빛 센서의 높이이다. 초음파 센서가 컵을 정확히 인식하기 위해 컵보다 아래에 가장 컵을 잘 인식할 수 있는 높이에 설치하여야만 하고 빛 센서도 라인을 정확히 따라가기 위해 빛 센서가 좋은 문턱값을 설정할 수 있도록 빛 센서를 낮게 부착해야 할 것이다.

∴ 종합적으로 봤을 때 무게중심과 두 센서의 위치가 이번 로봇 설계에 가장 많은 영향을 미치게 될 것이다. 그리고 추가적으로 콜렉터와 판을 설치하여 모든 미션을 완벽히 수행하도록 설계하여야 할 것이다.

3. Concept Design

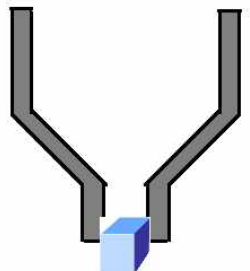
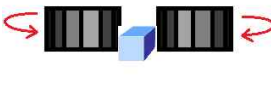
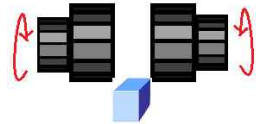
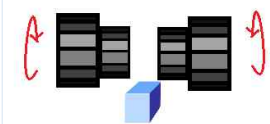
3.1 Wheel Design

General Wheel		Chain Wheel	
			
<ul style="list-style-type: none"> · 안정하다. · 속도가 빠르다. · 방향전환이 쉽다. · 조립하기 쉽다. · 무겁다. 	<ul style="list-style-type: none"> · 앞바퀴가 작아서 언덕을 오르기 쉽다. · 조립하기 쉽다. · 언덕을 내려올 때 불안정하다. 	<ul style="list-style-type: none"> · 장애물을 통과하기 용이하다. · 조립하기 힘들다. · 마찰력이 강하다. · 방향전환이 힘들다. 	<ul style="list-style-type: none"> · 장애물을 통과하기 용이하다. · 조립하기 힘들다. · 마찰력이 강하다. · 안정적이다.

<Table. 3-1 Wheel design>

디자인 초기에 바퀴와 Chain Wheel을 고안했다. Sand swamp mission을 통과하기에 Chain Wheel이 적합하다 생각하여 Chain Wheel로 디자인을 시도했다. 더불어 4개의 바퀴로 구성된 Chain Wheel을 만들었으나 방향전환을 할 때에 각도가 부정했는데, 차체가 덜컹거리어서 방향이 틀어지는 것이라 추정하여 가운데에 바퀴를 하나씩 더 추가했다.

3.2 Sample Collector

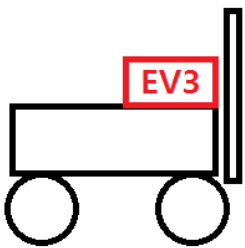
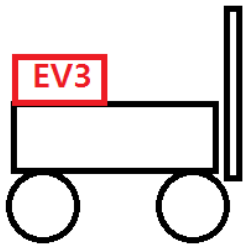
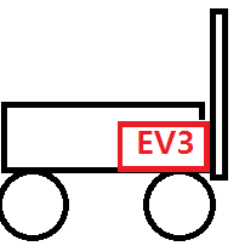
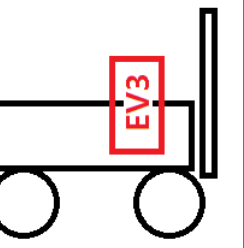
집게	Horizontal Roller	Vertical Roller 1	Vertical Roller 2
			
<ul style="list-style-type: none"> · Sample을 잡아서 뒤로 넘긴다. · Sample을 잡기 어렵다. · 시간이 오래 걸린다. 	<ul style="list-style-type: none"> · Sample을 수집하기 매우 쉽다. · 단시간 내에 수집이 가능하다. · 무겁다. 	<ul style="list-style-type: none"> · Sample을 수집하기 쉽다. · 또 다른 바퀴로 이용 가능하다. · 무겁다. · 차량을 안정적으로 만들어준다. 	<ul style="list-style-type: none"> · Sample을 수집하기 쉽다. · 또 다른 바퀴로 이용 가능하다. · 무겁다. · 차량을 안정적으로 만들어준다.

<Table. 3-2 Sample collector desgin>

최초의 디자인은 집게로 하였는데 Sample을 잡기도 힘들고 시간도 오래 걸려서 Roller 방식으로 바꾸었으며 Horizontal Roller의 경우 무거워서 언덕을 오르지 못하여서 또 다른 동력을 이용하기 위해 Vertical Roller로 바꾸었다. Vertical Roller1에서 바퀴를 4개나 쓴 이유는 무

게중심을 맞추기 위해서였으며, Vertical Roller 1의 경우 Sample이 흡입 과정에서 옆으로 빠지는 경우가 생겨서 개선시킨 Vertical Roller 2로 디자인을 했다.

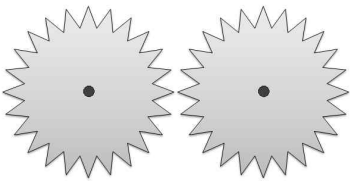
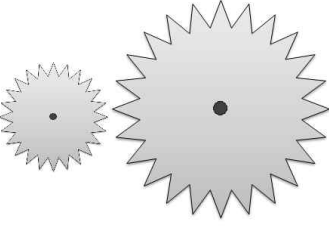
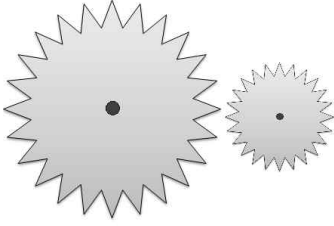
3.3 EV3 Position

Front side Horizontal	Back side Horizontal	Front side Bottom Horizontal	Front side Bottom Vertical
			
무게중심 앞쪽 위	무게중심 뒤쪽 위	무게중심 앞쪽 아래	무게중심 약간 앞쪽 중간높이

<Table. 3-3 EV3 position desgin>

EV3가 가장 무겁기 때문에 가운데에 위치시켜서 무게중심을 가운데로 맞추는 것이 가장 이상적이다. 하지만 가운데의 경우, 모터 때문에 EV3를 안착시킬 공간이 없어서 앞쪽과 뒤쪽 중 한 곳에 안착시켜야 했다. Sample Collector가 뒤에 설치되어 있어서 후면의 무게가 상대적으로 컸기 때문에 Front side를 선택했다. Front side Horizontal의 경우 무게중심이 앞으로 크게 쏠려서 무게중심이 앞으로 덜 쏠릴 수 있는 Front side Bottom Vertical 으로 디자인을 결정했다.



3.4 Gear Ratio

입력축 잇수 : 출력축 잇수		
1:1	1:n	n:1
		
$E = \omega \times T$	$E = \frac{1}{n} \omega \times nT$	$E = n\omega \times \frac{1}{n} T$
Output = T	Output = nT	Output = $\frac{1}{n} T$

<Table. 3-4 Gear ratio desgin>

언덕을 잘 올라가기 위해서는 출력축의 토크가 강해야 하므로 기어비가 높아야 한다. 그래서 모터의 출력을 높이기 위해 적절한 Gear Ratio를 사용했다.

3.4 Ultra Sonic Sensor

Horizontal	Vertical
	
· 상하 오차 x	· 좌우 오차 x

<Table. 3-5 Ultra sonic sensor desgin>

우선 장애물 컵이 왼쪽에 위치해 있기 때문에 Sonic Sensor를 차량 왼쪽에 달았으며 장애물을 마주했을 때 상하 오차 보다 좌우 오차가 더욱 크고 중요하기 때문에 Vertical 방식으로 디자인을 했다. 그리고 Ultra Sonic Sensor의 높이도 중요한데 컵의 높이가 20cm내외 정도였기 때문에 15cm보다는 낮게 설계했다.

3.5 Light Sensor

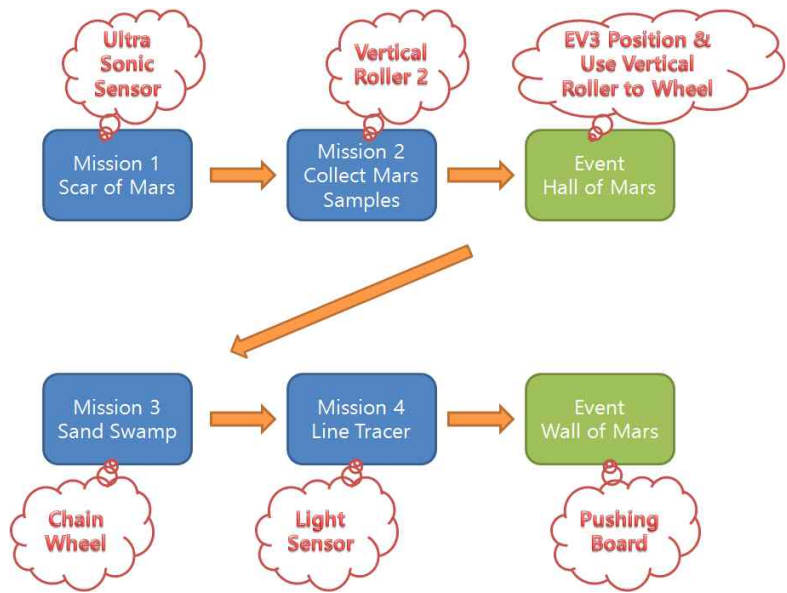
Line tracing을 성공적으로 시행하기 위해서는 Light Sensor의 높이와 Left sensor와 Right sensor의 간격이 중요하다. Light Sensor의 높이가 너무 높으면 빛이 퍼져서 감지를 잘 못하게 되고 너무 낮으면 언덕 장애물에서 턱에 걸리기 때문이다. 그리고 두 센서의 간격이 너무 넓으면 일직선으로 가지 못하고 너무 지그재그로 움직일 것이며 너무 좁으면 탈선을 할 위험성이 증가한다. 최종적으로 적절한 높이에 Light Sensor를 달았으며 line의 넓이가 약 2cm정도이기 때문에 Light Sensor 간격이 line보단 살짝 넓은 정도로 디자인했다.

3.6 Pushing Board

Wall of Mars 맵에서 모든 원기둥을 집어넣기 위해서 원기둥들이 포진해있는 면적과 일치하게 설계하였고, Board 사이의 구멍이 원기둥보다 크지 않도록 했다.

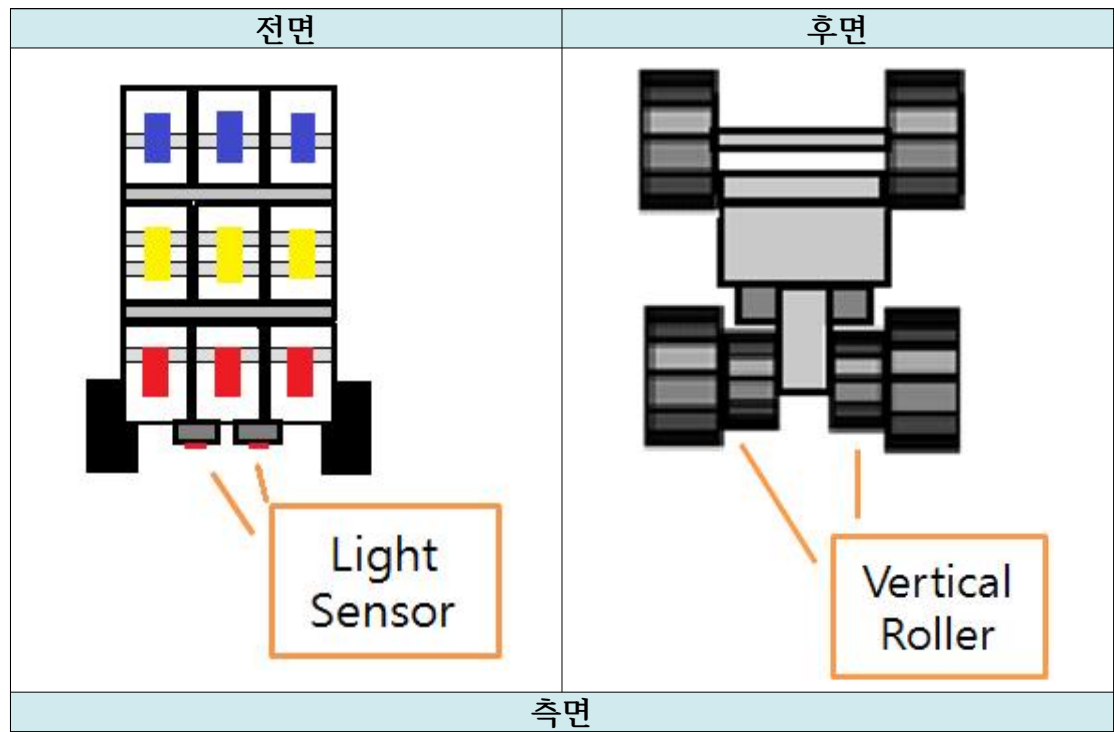
3.7 Final Concept Design

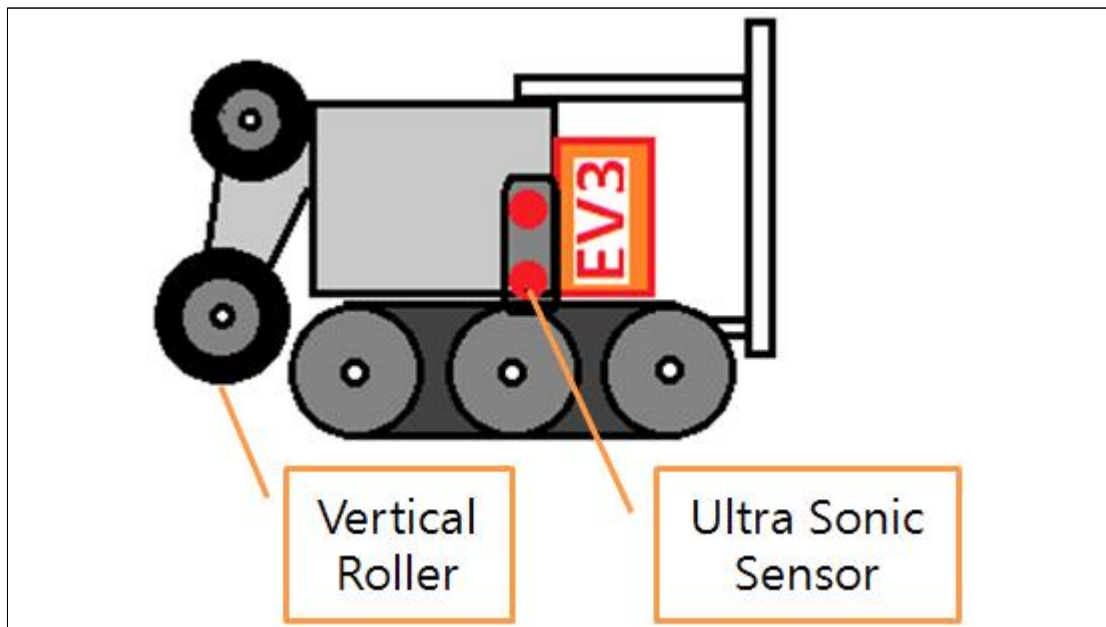
- Solving Missions



<Figure. 3-1 Schematic diagram of robot>

- Robot Image





<Table. 3-6 Robot design>

4. Robot making & Detail design

4.1 Catching Problems

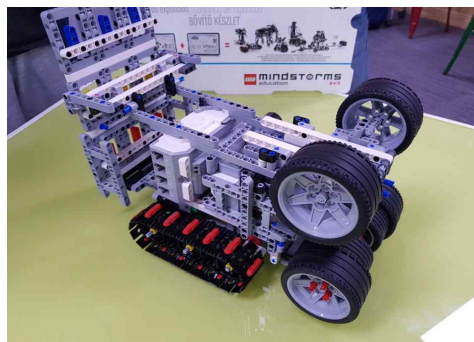
로봇을 제작하며 테스트하는 전 과정에서 발생한 Design 문제들은 다음과 같다.

Mission	Problem
1	<ul style="list-style-type: none">- 초음파 센서의 위치가 너무 높음.- 로봇의 회전 반경이 넓어 컵을 밀어냄.
2	<ul style="list-style-type: none">- 천천히 지나가야 블록을 흡입.- 흡입구 위치가 잘 보이지 않아 컨트롤이 어려움.
3	<ul style="list-style-type: none">- 언덕을 올라갈 때 Moment 때문에 뒤로 넘어감.- Hole을 지날 때 로봇의 길이가 짧아 빠지게 됨.- 무게중심이 높아서 언덕을 내려갈 때 넘어짐- 언덕을 올라가는 마찰력이 부족함- 충격에 의해 블록을 떨어트림
4	<ul style="list-style-type: none">- 벽을 미는데 밀리지 않는 기둥이 있었음.- 라이트 센서의 위치가 높아서 정밀하지 못함.

<Table. 4-1 Design problems>

4.2. Detail design

앞서 언급한 문제점을 보완해서 만들어진 로봇은 다음과 같다. 이제 문제들을 어떻게 보완해 나갔는지 파트별로 분석을 해보았다.



<Fig. 4-1 Full size of robot>

-Wheel



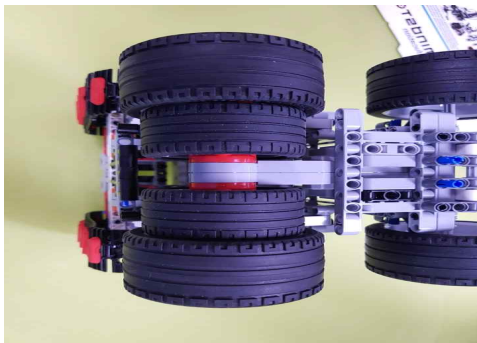
<Figure. 4-2 Side of wheel>



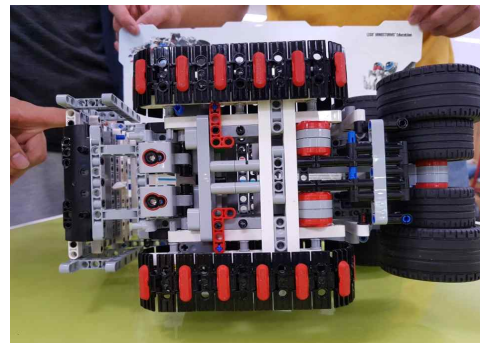
<Figure.4-3 Upper side of wheel>

언덕을 올라갈 때 로봇의 바퀴에 마찰이 부족한 문제점이 발생했다. 로봇을 주행할 때 해당하는 지면과 로봇 간의 마찰을 고려해야한다.[9] 그래서 궤도 바퀴를 사용하였고 예상보다 언덕을 올라가는데 마찰력이 부족해서 바퀴에 고무패드를 추가적으로 끼웠다. 또한 언덕을 올라가는 과정에서 로봇의 가운데 부분이 뜨지 않도록 하기 위해 안쪽에 바퀴를 하나씩 더 추가했다. 이를 통해 로봇이 이동하면서 궤도의 떨림으로 인해 원하는 만큼 움직이지 않던 점을 보완했다.

-Sample Collector



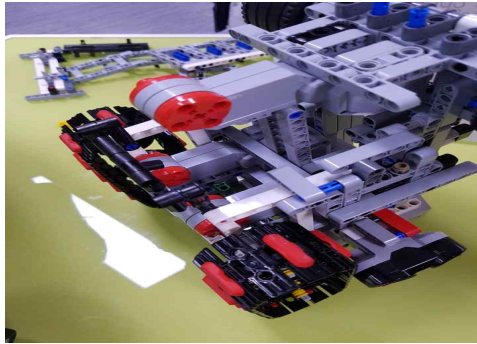
<Figure. 4-4 Front of collector>



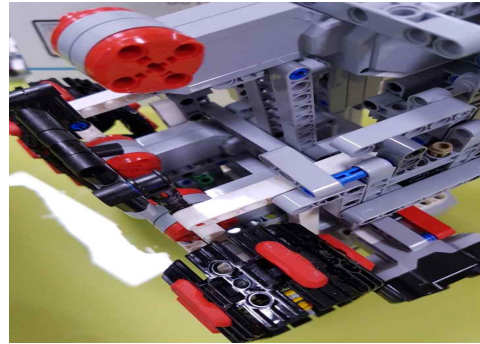
<Figure.4-5 Bottom side of collector>

가장 큰 고무바퀴를 이용해 블록을 눌러서 안쪽 공간으로 흡수 할 수 있도록 설계를 했다. 모터에서 고무바퀴로 동력이 바로 전달이 된다. 이는 손실이 최소화된 강한 힘으로 바퀴를 돌릴 수 있으므로 천천히 가지 않아도 블록을 모을 수 있다. 또한 가운데 부분이 눈에 잘 보여서 컨트롤을 하기가 용이하다.

-Cavity for storing block



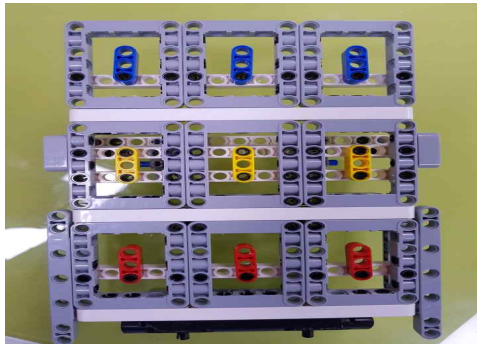
<Figure.4-6 Front of cavity>



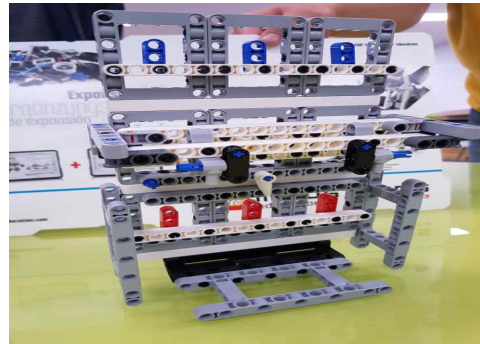
<Figure.4-7 Closed shot of cavity>

컬렉터가 모은 블록을 잘 보관하기 위한 공간도 중요하다. 블록을 보관하는 공간이 6개를 보관하기에 딱 알맞게 만들었더니 3번 4번 미션에서 충격에 의해 블록이 떨어지는 일이 자주 발생했다. 이를 보완하기 위해 내부 공간을 최대한 넓혀서 먼저 수집된 블록이 뒤쪽으로 이동하고 큰 방해없이 움직일 수 있도록 설계를 했다.

-Wall



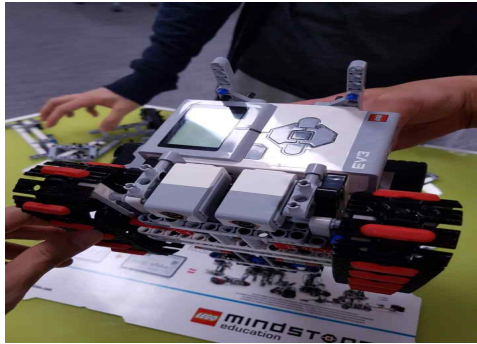
<Figure.4-8 Front of wall>



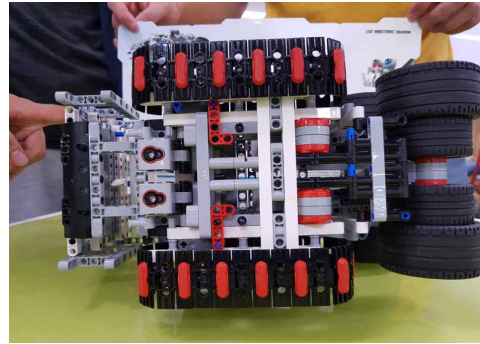
<Figure.4-9 Back of wall>

벽을 설계할 때 기둥의 지름보다 큰 구역을 막지 않았더니 2,3개가 남는 경우가 발생했다. 그래서 사이를 작은 블록들로 채웠다. Figure. 5.8을 보면 벽이 로봇에 딱 붙어있지 않고 살짝 떨어져 있음을 알 수 있다. 그 이유는 무게중심을 가운데로 맞춰주기 위함과 Mission 3에서 언덕을 올라갈 때 뒤로 넘어지는 모멘트의 작용점을 뒤로 옮기기 위한 목적도 있다. 실제로 최종 디자인에서 벽의 위치 조정만으로 로봇이 뒤로 넘어지지 않았다.

-Light Sensor



<Figure.4-10 Light sensor>



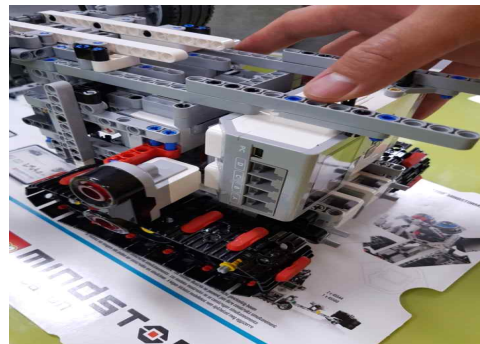
<Figure.4-11 Bottom light sensor>

두 개의 빛 센서를 로봇에 뒤쪽에 나란히 배치하였고 지면과의 거리가 3~4cm 되도록 설계했다. 빛 센서 간의 간격은 검은 선의 두께와 라인을 따라가며 바퀴가 위치를 조정하는 속도에 맞춰 4~6cm가 되도록 설계했다.

-Ultrasonic Sensor



<Figure.4-12 Front of Ultrasonic sensor>

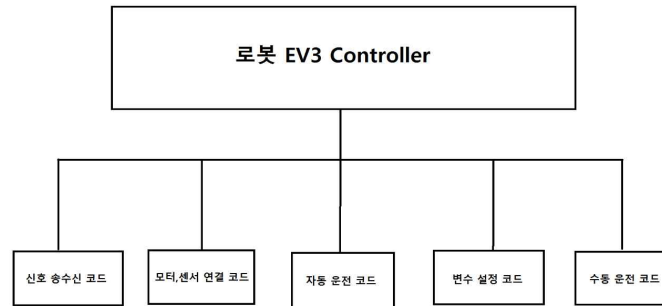


<Figure.4-13 Side of Ultrasonic sensor>

초음파 센서를 로봇이 회전의 중심이 되는 지점인 중간지점에 배치했다. 그래서 Mission 1에서 컵이 없는 부분을 발견한 뒤 회전을 하는 코드를 설계할 때 이동 속도 및 거리의 직관성을 높였다. 더불어 초음파 센서를 가로 혹은 세로로 배치할지 실험을 했고, 그 결과 세로 방향으로 설치하였을 때 초음파 센서의 좌우 범위가 더욱 정밀하다는 사실을 알 수 있었다.

5. Algorithm & Code

알고리즘은 로봇이 움직이는 방식을 지배하는 로봇의 뇌와 같은 부분이다. 그래서 다음과 같이 전체적인 구조를 구성을 하였다.



<Figure. 5-1 General Concept of Algorithm>

5.1 Entire frame

5.1.1 Signal transmit / receive code

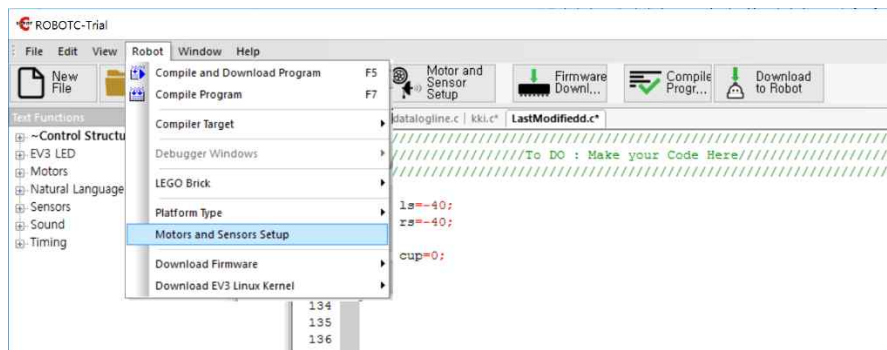
EV3 에는 두 가지의 신호 교환 방법이 있다. 한 가지는 'USB'를 통해서, 또 다른 방법으로 'Blueetooth'를 통해서 이다. 실제 로봇이 화성 탐사를 할 때 컴퓨터와 선 연결을 유지할 수 없으므로 원격으로 신호를 보낼 수 있는 Bluetooth를 사용해야 한다. Bluetooth는 mailbox 라는 기능을 사용하여 Cellphone Application 상에서 보내진 신호를 수신하여 코드에 따라 로봇을 움직이게 해준다. Mailbox의 코드는 아래와 같다.

Code	Explanation
<pre> task listen() { //mail box setup char mb_name[10]; sprintf(mb_name,"mymb"); int idx=5; //////////////// mailbox index number int msg_len = 0; //open mail box openMailbox(idx,mb_name); //listening while(true) { waitForMailboxMessage(idx); //playSound(soundBeepBeep); // } check recieve msg_len = getMailboxMessageSize(idx); readMailbox(idx,buf,msg_len); displayTextLine(1,buf,msg_len); char_get = buf[msg_len-2]; if(char_get>='0' && char_get<='9') com = char_get; wait1Msec(10); } </pre>	<p>Task 라는 Robot c 상에서의 하나의 '틀'에서 코드가 실행된다. 괄호안의 코드는 listen 을 통해서 실행된다.</p> <p>Mailbox 에서 '5'라는 숫자를 입력해야 신호교환 채널이 개방이 된다.</p>

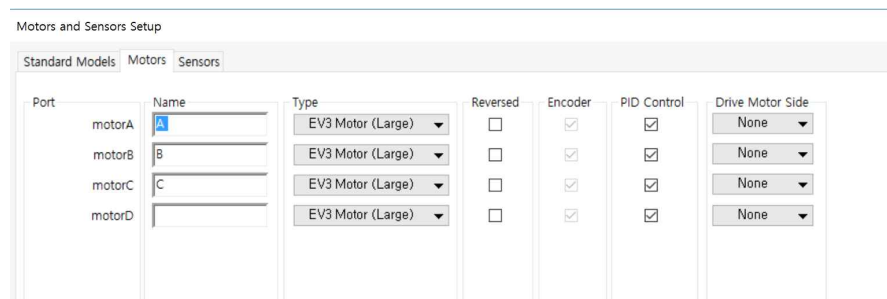
5.1.2 Motor, Sensor connecting code

로봇이 움직이고 mission 을 통과하기 위해서는 모터, 센서가 필요하다. 세부적인 코드에 앞서서 반드시 Robot C 상에서 모터와 센서를 연결해주는 코드를 구성해주어야 한다. 코드와 설정은 다음과 같다.

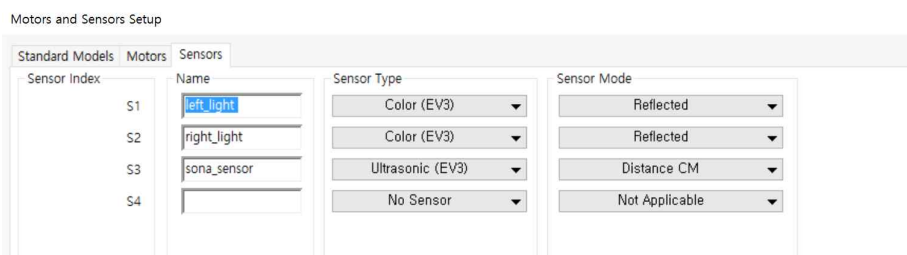
- ① Motors and Sensors Setup 을 들어간다.



- ② Motor 항목에서 보기의 그림과 같이 변수 이름을 지정해준다.



- ③ Sensor 항목에서 빛 센서와 초음파 센서의 변수 명을 지정해준다. 이 때 Sensor Type 은 빛 센서의 경우 Color, 초음파 센서의 경우 Ultrasonic으로 설정해준다.



설정이 끝나고 코드창으로 가보면 맨 위에 다음과 같은 코드가 자동적으로 생성된다.

```
#pragma config(Sensor, S1, left_light, sensorEV3_Color)
#pragma config(Sensor, S2, right_light, sensorEV3_Color)
#pragma config(Sensor, S3, sona_sensor, sensorEV3_Ultrasonic)
#pragma config(Motor, motorA, A, tmotorEV3_Large, PIDControl, encoder)
#pragma config(Motor, motorB, B, tmotorEV3_Large, PIDControl, encoder)
#pragma config(Motor, motorC, C, tmotorEV3_Large, PIDControl, encoder)
//**Code automatically generated by 'ROBOTC' configuration wizard **//
```

5.1.3 Automatic operation code

자동 운전 코드는 Mission 1, Mission 4에서 실행된다. 로봇은 빛 센서 와 초음파 센서로 구성되어 있고, 각각의 센서에서 수신되는 값을 피드백한다. 그래서 조건문에 따라서 임무를 완료하면 조건문을 나가는 과정을 거친다.

5.1.4 Parameter setting code

코딩을 효율적으로 설계하고 시험 과정에서 발생한 문제에 따라 신속하게 설정을 바꿀 수 있도록 변수를 개별적으로 설정했다.

① Threshold 변수 설정 및 컵 개수 변수 설정

```
#pragma config(Sensor, S1, left_light, sensorEV3_Color)
#pragma config(Sensor, S2, right_light, sensorEV3_Color)
#pragma config(Sensor, S3, sona_sensor, sensorEV3_Ultrasonic)
#pragma config(Motor, motorA, A, tmotorEV3_Large, PIDControl, encoder)
#pragma config(Motor, motorB, B, tmotorEV3_Large, PIDControl, encoder)
#pragma config(Motor, motorC, C, tmotorEV3_Large, PIDControl, encoder)
/*!!Code automatically generated by 'ROBOTC' configuration wizard !!*/

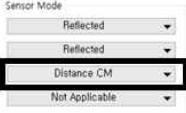
int com=0; //control number
int char_get;
char buf[256];
#define THRESHOLD 30
#define THRESHOLD2 20

int cup=0;
```

← 빛 센서 문턱값 설정

← 컵 개수 0으로 초기화 설정.

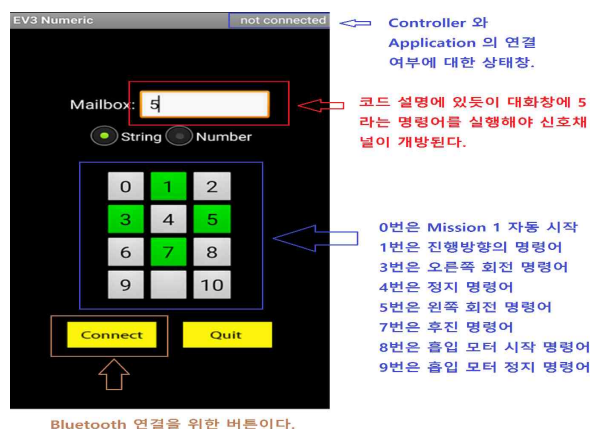
② 로봇 행동 관련 명령어 설정

Code	Explanation
<pre>void encstraight(int distance, int sp) //cm { int angle = distance/12.8*360; nMotorEncoder[motorA] = 0; nMotorEncoder[motorB] = 0; while (nMotorEncoder[motorA] > -angle && nMotorEncoder[motorB] > -angle) { if (nMotorEncoder[motorA] > -angle) { motor[motorA] = -sp; } if (nMotorEncoder[motorB] > -angle) { motor[motorB] = -sp; } motor[motorA] = 0; motor[motorB] = 0; playSound(soundBeepBeep); wait1Msec(500); } void encright(int rotangle, int sp) //deg { int angle = (rotangle+1)*7 ;</pre>	<p>Void 란 값을 보내고 다시 반환받지 않는다는 뜻의 명령어이다. 이 부분은 단순히 값을 반환받을 필요가 없는 것이기 때문에 void 함수를 쓴다.</p> <p> 단위: CM</p> <p>보기와 같이 SETUP 에서는 CM로 거리 단위를 설정하였다. Encstraight(1 2) 에서 1부분은 앞으로가고 싶은 거리, 그리고 2는 가고 싶은 속도를 설정하는 부분이다. Encoder 설정을 통하여 엔코더 회전각이 'angle' 변수까지 도달하면</p> <p>여기서 '삐빅' 하고 0.5 초 동안 소리가 울린다.</p> <p>오른쪽으로 회전하는 부분이다. Rotangle은 회전하고 싶은 각도, sp 는 속도이다.</p>

<pre> nMotorEncoder[motorA] = 0; nMotorEncoder[motorB] = 0; while (nMotorEncoder[motorA] > -angle && nMotorEncoder[motorB] < angle) { if (nMotorEncoder[motorA] > -angle) { motor[motorA] = -sp; } if (nMotorEncoder[motorB] < angle) { motor[motorB] = sp; } } motor[motorA] = 0; motor[motorB] = 0; playSound(soundBeepBeep); wait1Msec(500); } void encleft(int rotangle, int sp) //deg { int angle = (rotangle+1)*7 ; nMotorEncoder[motorA] = 0; nMotorEncoder[motorB] = 0; while (nMotorEncoder[motorA] < angle && nMotorEncoder[motorB] > -angle) { if (nMotorEncoder[motorA] < angle) { motor[motorA] = sp; } if (nMotorEncoder[motorB] > -angle) { motor[motorB] = -sp; } } motor[motorA] = 0; motor[motorB] = 0; } </pre>	<p>sp의 속도로 angle 까지 각도가 도달하면</p> <p>0.5초동안 소리가 울린다.</p> <p>왼쪽으로 회전하는 부분이다. 오른쪽으로 회전하는 부분과 같은 설명이다.</p>
--	--

5.1.5 Manual operation code

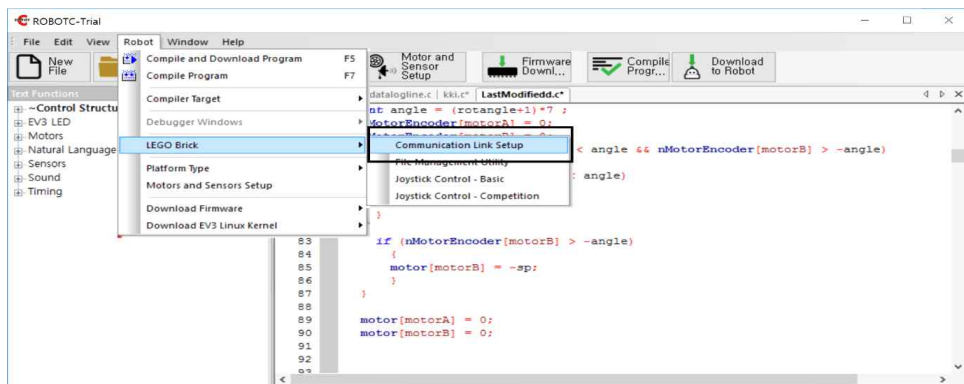
Mission 1와 Mission 4와 달리 Mission 2, 3은 수동 조작을 해야한다. 그래서 휴대폰 앱인 'EV3 Numeric'에서 숫자 버튼을 통해 로봇을 조종하는 코드를 넣어주어야 한다. 앱을 실행하면 다음과 같은 화면이 뜨게 된다.



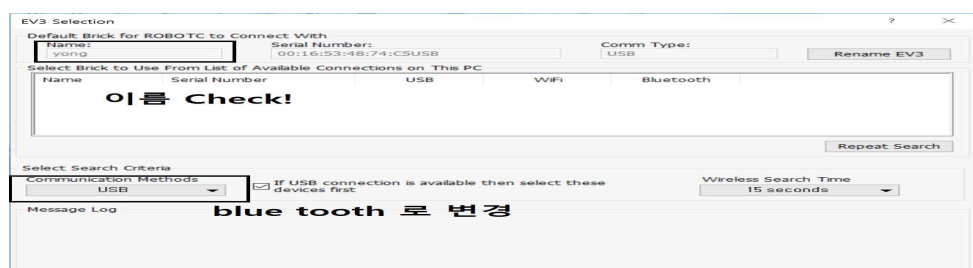
Connect 부분을 누르면 다음과 같은 창이 뜬다.



블루투스와 연결하는 기기가 많이 존재할 경우, 자신의 로봇을 알기 위해서는 다음의 과정을 거치면 된다.



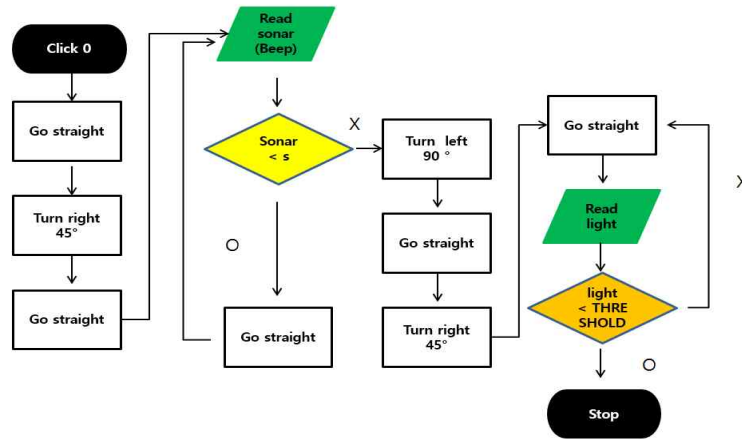
보기와 같이 Communication Link Setup 부분을 누르면



Name 부분의 이름을 어플리케이션에서 선택하면 된다.

5.2 Mission & Code Explanation

5.2.1 Mission 1 : Scar of Mars



<Figure. 5-2 Algorithm of Mission 1>

Mission 1에서는 컵 1, 2, 3, 4에서 무작위로 뽑힌 컵의 위치를 지나가기 위해서 초음파 센서가 사용되었고, 검은 선에 정지하기 위해서 빛 센서를 사용했다.

먼저 0 버튼을 누르면 설정된 만큼 앞으로 가고 45도 꺾는다. 그리고 앞으로 일정거리를 간다. 그 후 반복문에 의한 초음파 센서값을 검증한 뒤에 조건이 맞으면 앞으로 가고 만약 센서값이 S보다 크면 그곳이 컵이 없는 빈 공간이므로 왼쪽으로 90도 꺾는다. 그리고 일정 거리 가고 오른쪽으로 45도 꺾는다. 끝으로 또 하나의 빛 센서를 읽어들이는 반복문에 의해서 미션 1의 끝에 있는 검은 선을 만나면 자동으로 멈추게 되어있는 알고리즘이다. Mission 1의 코드는 다음과 같다.

Code	Explanation
<pre> void running_case0() { encstraight(17,30); encright(23.9,30); encstraight(45,50); while (1) { int s = SensorValue(sona_sensor); if(s < 30) { encstraight(11.5,50); cup+=1; playSound(soundBeepBeep); wait1Msec(500); } if(s >= 30) { encleft(47.8,30); encstraight(32,50); encright(23.9,30); break; } } } </pre>	<p>명령어 0을 누르면 실행하게 하는 부분.</p> <p>앞으로 30의 속도로 17cm 만큼 가라 오른쪽으로 30의 속도로 23.9만큼 회전하라. 앞으로 50의 속도로 45만큼 회전하라. 괄호 안의 구문을 계속 반복하게 하는 부분.</p> <p>초음파 센서로부터 값을 받아서 s에 정의 만약 센서 값이 30보다 작으면 앞으로 50의 속도로 11.5 cm 만큼 간다. 그리고 cup 변수에 1을 더한다.</p> <p>센서값이 30보다 클 경우, 즉 컵이 없는 빈공간에 도달한 경우 왼쪽으로 90도 꺾고 일정 거리 32 cm 만큼 간 후 오른쪽으로 45도 꺾는다. 그리고 반복문에서 빠져나온다</p>

<pre> if(s<30 && cup==2) {break; } if(cup==2) { motor[motorA]=10; motor[motorB]=-10; wait1Msec(900); motor[motorA]=-20; motor[motorB]=-20; wait1Msec(900); motor[motorA]=10; motor[motorB]=-10; wait1Msec(1300); motor[motorA]=-20; motor[motorB]=-20; wait1Msec(2000); } while(1) { int right = SensorValue(right_light); int left = SensorValue(left_light); if (right < 20 left < 20) { motor[motorA] = 0; motor[motorB] = 0; break; } else { motor[motorA] = -80; motor[motorB] = -80; } } sleep(3000); </pre>	<p>센서값과 cup이 2가 되었을 때 반복문을 빠져나온다. (cup 4 를 위한 조건문)</p> <p>Cup4 를 지나가야 할 경우 컵을 3개까지 인식하고 cup 변수가 2가 된 순간 반복문에서 빠져나와 느린속도로 왼쪽으로 살짝 꺾고 앞으로 조금 간다음에 또 왼쪽으로 살짝 꺾고 앞으로 살짝 가는 코드이다. 이 코드는 엔코더가 아니라 모터 출력으로 설정했기 때문에 실험이 필요한 항목이었다.</p> <p>이 부분은 컵의 빈공간을 지난후에 검정 선을 만날 때까지 진행하도록 하는 코드 부분이다.</p> <p>왼쪽 이나 오른쪽 센서 둘 중 하나라도 THRESHOLD 값인 20보다 작아질 경우 멈추고 반복문을 빠져나오도록 했다.</p> <p>모든 과정이 끝나고 MISSION2 를 시작하기위해서 SLEEP 명령을 통해 수동 조작 버튼을 활성화할 수있도록 하였다.</p>
--	---

5.2.2 Mission 2: Collect Mars Samples

이 미션에서는 수동 조작이 필요하다. 위 항목에서 언급한 수동 조작 버튼을 통하여 운전자가 ‘화면’을 보고 원격 조작을 한다. 이 부분에서 특이한 점은 화면을 통해서 조작을 한다는 점인데 화면으로 보면 회전방향이 직관적으로 생각했을 때 반대방향으로 나온다. 그래서 코드를 반대로 설정하였다. 그 부분과 앞, 뒤, 블록 흡입 모터의 코드 짜임새는 다음과 같다.

CODE	EXPLANATION
<pre>void running_case1() { Motor[motorA]=-40; Motor[motorB]=-40; } void running_case3() { motor[motorA]=-15; motor[motorB]=15; } void running_case4() { motor[motorA]=0; motor[motorB]=0; } void running_case5() { motor[motorA]=15; motor[motorB]=-15; } void running_case6() { motor[motorC]=0; } void running_case7() { motor[motorA]=40; motor[motorB]=40; } void running_case8() { motor[motorC] = -100; }</pre>	<p>앞으로 가는 버튼 명령이다. 속도 40의 속도로 가도록 설정하였다.</p> <p>원래 설정에서는 A가 15이고 B가 -15였는데 실제 경연장에서 해보니 좌우 반대여서 값을 바꿨다. 화면상으로 왼쪽 회전이다.</p> <p>진행 모터를 멈추는 코드이다.</p> <p>같은 이유로 화면상으로 오른쪽 회전이다.</p> <p>흡입 모터를 멈추는 코드이다.</p> <p>후진 버튼이다.</p> <p>흡입 하는 방향으로 100(최대 속도) 로 motor C를 돌리는 코드이다.</p>

5.2.3 Event : Hall of Mars + Mission 3 : Sand Swamp

샘플 수집이 끝나면 추가 미션으로 진행할 수 있다. 구간을 통과하기 위한 방법으로 블록을 밀어서 홈을 채운 뒤 지나가거나 로봇 설계를 통해 홈을 지나가는 방법이 있다. 실제 경연에서 후자를 택하기로 했고 로봇의 디자인으로 충분히 극복가능하기 때문에 개별적으로 코드를 설계하지는 않았다. 그리고 Sand Swamp는 앞으로 로봇을 전진시켜서 충분히 통과할 수 있었다.

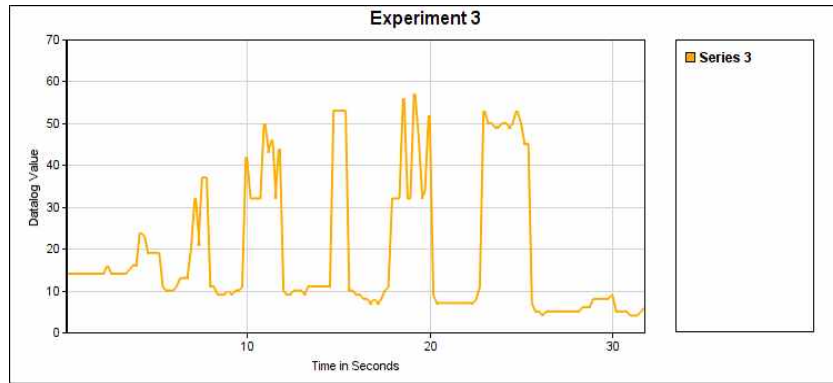
5.2.4 Mission 4 : Line Tracer + (6) Event: Wall of Mars

알고리즘의 시작은 intersection 과 intersection3을 0 으로 초기화 하는 것에서 시작된다. 앞서 설정해둔 Threshold(실제 설정은 Threshold 2임)을 두 센서가 모두 넘지 못하면 검은색 선, 즉 일자 검정 선에 도달했다는 뜻이므로 intersection 값에 1을 더해주고 계속 반복하여 '4'에 도달하면 오른쪽으로 90도 회전 하고 '8'이 되면 코딩처럼 좀 더 전진해서 EVENT STAGE에서 원기둥을 밀고 180도 회전한다. 그리고 앞과 같은 방식으로 intersection 3이 '1'이 되면 왼쪽으로 90도 회전, 그리고 '3' 이 되었을 때 왼쪽으로 90도 회전, 조건문에 따라서 '5'에 도달하면 조금 더 전진해서 END point에 도달하고 멈춘다.

알고리즘을 나타나듯 두 센서값 모두 Threshold보다 작으면 교차 지점에 도착했으므로 1을 더하고 하나 센서값이 문턱값보다 작고 나머지 센서값이 문턱값보다 큰 경우, 로봇 차체가 line 에서 벗어난 상태라는 것을 의미하므로 알맞게 회전해서 설계된대로 실행하게 되어있다. 이를 계속 반복해서 Line Tracing을 완료할 수 있다.

* Data Logging

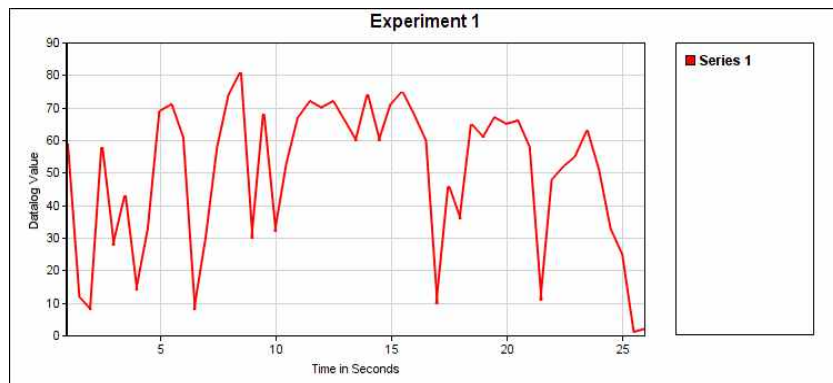
설계한 코드가 수치적으로 어떻게 표현되는지 알기 위해서 다음과 같이 Data logging을 해보았다. 앞서 만든 알고리즘으로 초안 코드를 설계했고, 모의 맵을 만들어서 실험을 진행했다. Figure 는 Mission 1 의 초음파 센서 값을 주기적으로 측정한 그래프이다.



<Figure. 5-4 Sensor value of sonar sensor>

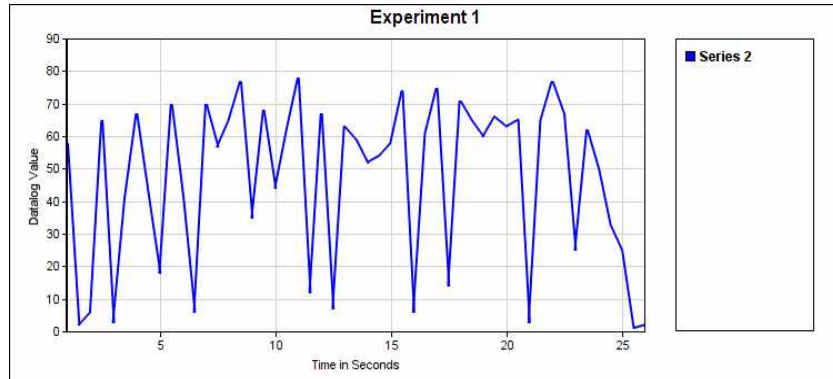
그래프에서 값이 30 이상으로 솟아있는 데이터 군집의 경우에 컵과 컵 사이의 빈공간을 의미한다. 그리고 그 데이터 사이의 낮은 데이터는 SENSOR VALUE 경계값을 30으로 설정해서 그보다 낮을 때 전진하도록 설정했기 때문에 로봇이 컵 벽면을 일직선으로 따라가면서 자동적으로 생기는 부분이다. 실험 종료 시점에 해당하는 상대적으로 낮고 긴 부분 (Time=30 전후 부분)은 무작위하게 뽑힌 컵의 빈 공간을 지나는 부분이다.

그리고 또 다른 자동모드인 Line Tracing에서 센서의 값을 측정 해보았다. 로봇의 왼쪽 빛 센서에 해당하는 Data logging 값이다.



<Figure. 5-5 Sensor value of light sensor(left)>

오른쪽 센서의 data logging 이다.



<Figure. 5-6 Sensor value of light sensor(right)>

이 그래프에서 찾을 수 있는 특징은 초음파 센서와 크게 다르지 않다. 빛 센서는 값이 크면 밝은 부분이고 값이 작으면 어두운 부분을 감지하고 있다는 특징이다. 따라서 그 원리에 따라 로봇이 가면서 검정 색 부분을 어느 시간에서 만났는지 그리고 인식했는지 안했는지 설정한 THRESHOLD 값 보다 밑에 있는 데이터 군집의 개수를 세면 알 수 있다. Line Tracing 에서는 총 13개의 교차지점을 지나야 한다. 하지만 그래프를 보면 왼쪽 센서는 설정한 THRESHOLD 값 20보다 아래의 데이터 군집이 6개 , 그리고 오른쪽 센서는 10개 밖에 없는 것을 알 수 있다. 이는 로봇의 속도 및 센서의 태생적인 지연 인식이 문제 일 것이라고 생각이 들었고 실험과 코드 수정을 통해서 보완하였다.

이를 통해서 최종적으로 보완한 코드는 다음과 같다.

CODE	EXPLANATION
<pre> void running_case2(){ int b=0; int d=0; int intersection = 0; int intersection3 =0; while (intersection<9) { int left_light_value = SensorValue(left_light); int right_light_value = SensorValue(right_light); if ((left_light_value > THRESHOLD2) && (right_light_value > THRESHOLD2))/? { b = 0; motor[motorA]=50; motor[motorB]=50; } if ((left_light_value > THRESHOLD2) && (right_light_value < THRESHOLD2))/? { b = 0; motor[motorA]=-20; motor[motorB]=20; } if((left_light_value<THRESHOLD2) && (right_light_value>THRESHOLD2))//AAE? { b = 0; motor[motorA]=20; motor[motorB]=-20; } } </pre>	<p>2번 버튼을 누르면 line tracing 을 시작하도록 하는 부분이다.</p> <p>이 부분은 변수 설정에서는 언급하지 않았지만 검은 색 교차지점에 도달했을 때 intersection, intersection3 값을 무한정 늘어나지 않게 하는 역할의 변수이다.</p> <p>교차 변수가 8이 될 때 까지 센서 값에 따라 오른쪽 ,왼쪽으로 조절하면서 두 센서가 모두 밝을 때는 진행하고 검정 색 선에서는 변수 값을 증가시키는 일련의 코드들이다. 또한 Sensorvalue 로 센서값을 받는다.</p> <p>이 부분의 속도 설정이 수동조작에서의 후진인 이유는 원기동을 밀기 위해서 벽을 앞으로 보게 시작하기 때문이다.</p> <p>조건문을 만족할 때마다 b=0 을 선언해준다. 이유는 앞에서 설명한 바와 같다.</p>

<pre> } if ((left_light_value < THRESHOLD2) && (right_light_value < THRESHOLD2))/? { if (b == 0) { motor[motorA]=50; motor[motorB]=50; wait1Msec(100); } b = 1; intersection += 1; displayTextLine(1,"%d",intersection); playSound(soundUpwardTones); } if (intersection == 4) { motor[motorB]=-10; motor[motorA]=50; wait1Msec(1100); } if (intersection == 8) { motor[motorA]=20; motor[motorB]=20; wait1Msec(100); motor[motorA]=-50; motor[motorB]=-50; wait1Msec(600); motor[motorA]=0; motor[motorB]=0; wait1Msec(800); nMotorEncoder[motorA] = 0; nMotorEncoder[motorB] = 0; while (nMotorEncoder[motorB] > -700 && nMotorEncoder[motorA] < 700) { if (nMotorEncoder[motorB] > -700){ motor[motorB] = -30; } if (nMotorEncoder[motorA] < 700){ motor[motorA] = 30; } } break; } while (intersection3<6) { int left_light_value = SensorValue(left_light); int right_light_value = SensorValue(right_light); if ((left_light_value > THRESHOLD2) && (right_light_value > THRESHOLD2))/? Z { d = 0; motor[motorA]=50; motor[motorB]=50; } if ((left_light_value > THRESHOLD2) && (right_light_value < THRESHOLD2))/?i { d= 0; </pre>	<p>교차지점을 만날때마다 확인을 하기 위해서 넣은 소리나는 코드이다.</p> <p>네 번째 교차지점에 도달 했을 때 왼쪽으로 약 90도를 꺾는 포인트이다. 모터 출력으로 하기 때문에 배터리 용량에 따라 다르지만 실험을 통해서 어느정도 평균값을 찾아서 설정하였고 어차피 조건문을 통해서 보정이 되기 때문에 크게 무리가 가지 않는다.</p> <p>원기등을 만나는 부분이다. 교차지점 8이 되었을때</p> <p>앞으로 20의 속도로 아주 조금 간다. 이유는 원기등과 8 교차 지점이 매우 가깝기 때문에 그렇다.</p> <p>뒤로 살짝 빼는 코드이다. 빼지 않으면 회전할 때 원기등 벽에 부딪힌다.</p> <p>이 부분은 정확히 하기 위해서 엔코더 회전수를 설정하여 180도를 회전하도록 하였다. 대략 700정도가 180도 회전각이다. 이 때 180도를 회전하는 이유는 처음에 뒤로 진행하려고 하였는데 sensor 위치가 처음과 반대이고 설정값과도 맞지 않아서 처음처럼 벽이 앞을 볼 수있도록 회전을 시키는게 맞다고 생각해서 넣은 코드이다.</p> <p>마지막 교차지점 intersection3 =5 가 될때까지 진행하도록 하는 반복문의 시작이다.</p> <p>앞에서 b와 같은 역할을 하는 변수 d이다.</p>
--	---

<pre> motor[motorA]=-20; motor[motorB]=20; } if((left_light_value < THRESHOLD2) && (right_light_value>THRESHOLD2))//AAE? { d= 0; motor[motorA]=20; motor[motorB]=-20; } if ((left_light_value < THRESHOLD2) && (right_light_value < THRESHOLD2))/? { if (d == 0) { motor[motorA]=50; motor[motorB]=50; wait1Msec(30); d = 1; intersection3 += 1; displayTextLine(4,"%d",intersection3); playSound(soundDownwardTones); } if (intersection3 == 1) { motor[motorA]=20; motor[motorB]=20; wait1Msec(1000); nMotorEncoder[motorA] = 0; nMotorEncoder[motorB] = 0; while (nMotorEncoder[motorA] < 335 && nMotorEncoder[motorB] > -335) { if (nMotorEncoder[motorA] < 335){ motor[motorA] = 30; } if (nMotorEncoder[motorB] > -335){ motor[motorB] = -30; } } } if(intersection3 ==5) {motor[motorA]=95; motor[motorB]=95; wait1Msec(1000); motor[motorB]=25; motor[motorA]=-25; wait1Msec(1300); motor[motorA]=95; motor[motorB]=95; wait1Msec(1000); motor[motorA]=0; motor[motorB]=0; break; } if(intersection3 ==3) { motor[motorA]=20; motor[motorB]=20; wait1Msec(1000); nMotorEncoder[motorA] = 0; nMotorEncoder[motorB] = 0; while (nMotorEncoder[motorA] < 335 && nMotorEncoder[motorB] > -335) { if (nMotorEncoder[motorA] < 335){ motor[motorA] = 30; } if (nMotorEncoder[motorB] > -335){ motor[motorB] = -30; } } } } } } } } </pre>	<p>첫번째 교차 지점에서 왼쪽으로 90도를 꺾는 조건문이다.</p> <p>마지막end point 앞의 교차지점 5까지 진행된 경우 완벽하게 end point 에 로봇이 들어갈 수있도록 살짝 넣은 코드이다. 앞으로 1초만큼 가고 오른쪽으로 살짝꺾고 다시 살짝 앞으로 가고 멈추고 반복문을 빠져나오는 진행순서이다.</p> <p>코드 순서가 보기 좋지 않게 바뀌기는 했지만 intersection3 == 5 보다 먼저 실행되는 조건이 만족되는 부분이다. 진행하다가 맵의 끝지점, 세번째 교차지점에서 왼쪽으로 90도 꺾는 코드이다.</p>
--	--

6. Conclusion

6.1 First try

Mission 1 - Scar of Mars

Mission 1에서는 첫 번째 조가 3번을 뽑아서 3번째 컵을 위치를 지나야 했다. Mission 1에서는 최초의 45도 회전이 관건이었다. 약간만 각도가 틀어져도 거리가 늘어날수록 많은 오차가 생겼기 때문이다. 경연 전날에 많은 실험을 해 보았는데 처음에 45도 회전이 실험을 할 때마다 조금씩 달라졌던 경험을 바탕으로 경연 당시 실험을 할 때와 동일한 위치에 정확하게 놓으려고 했다. 그래서 큰 오차는 없이 회전을 할 수 있었지만, 3번째 컵 지점을 통과할 때에 왼쪽 컵에 살짝 걸려서 컵을 밀어 10점 감점을 당했다. 이후 진행된 다른 조들도 상위권에 위치한 조를 제외하면 컵을 밀어서 감점을 자주 당했기 때문에 이부분에 코드를 수정을 따로 하지 않았다.

Mission 2 : Collect Mars Samples

Mission 2에서는 얼마나 Sample을 빠르게 수집하는지도 중요했지만 조종사의 침착함과 숙련도가 더욱 중요한 미션이었다. 특히 카메라의 좌우가 반대여서 조종하는데 애를 먹었고, 결국 6개 중에 2개의 샘플을 수집했다. 샘플 하나당 20점 획득인데 반해서 Mission 2에서 40초 이상 소요되었으므로, 2차 시도에서는 샘플 모으는 데 시간이 오래걸릴 것 같다면 포기하고 바로 다음 미션을 진행하기로 전략을 바꿨다. 다른 팀들의 경우도 상위권 팀을 제외하면 수집을 2,3개 정도밖에 못했다. 하지만 상위권 도약을 위해서는 많이 수집해야 했으므로 차별화된 전략이 필요했다.

Event : Hall of Mars

Hall of Mars는 경연에서도 앞에 달린 Vertical Roller 덕분에 안정감 있게 Hall을 통과할 수 있었다. 다만 첫 시도에서 바퀴가 실험할 때 보다 많이 헛돌아서 시간 소요가 상대적으로 많았다. 다른 팀들도 무난하게 통과를 했기 때문에, 특별한 전략을 세울 필요는 없었다.

Mission 3 : Sand Swamp

Mission 3은 바퀴를 Chain Wheel로 설계하여 표면적이 넓었기 때문에 아무런 문제없이 통과할 수 있었다.

Mission 4 & Event : Line Tracer & Wall of Mars

Line Tracer는 코딩을 얼마나 차량에 최적화되게 잘 했는지가 관건이었다. 수많은 실험을 통해서 선을 따라가다가 회전할 때에 두 바퀴 사이의 회전속도 차이를 어떻게 해야 매끄럽게 탈선을 하지 않고 갈 수 있는지를 알아내어 적용한 결과 첫 번째 turn과 원기둥을 밀 때까지 아주 깔끔하게 성공했다. 하지만 원기둥을 밀고 나서 180도 회전을 할 때에 실험했을 때보다 더욱 많이 회전하여서 탈선을 했다. 그래서 점수를 전부 받지 못했고, 완주한 팀과 상당한 격차가 벌어졌다. 원기둥을 미는 것보다 완주를 제대로 하는 것이 중요하다는 것을 깨달았다. 2차 시도 전에 각도값을 수정을 했다.

Result : 160 points

6.2 Second try

Mission 1 - Scar of Mars

2차 경연 때에는 4번째 컵 위치를 통과해야 했다. 4번째 컵은 가장 멀리 있고, 통과할 때에 공간이 좁아서 4가지 case 중에서 가장 난이도가 높았다. 1~3번 case까지의 방법대로 turn을 하게 되면 옆에 있는 컵들을 모두 밀쳐버리기 때문에 3번 컵에 컵이 있을 경우 서서히 앞으로 가면서 turn을 하게 만들어 컵을 최소한으로 밀고 지나가 감점 기준을 벗어나지 않게 코딩을 해놓은 상태였다. 하지만 역시 약간의 오차 때문에 오른쪽 컵에 살짝 걸리는 바람에 10점 감점을 당했다. 그래도 어려운 구간을 잘 통과한 덕인지 상대적으로 편안 마음을 가지고 다음 미션에 들어갔다.

Mission 2 - Collect Mars Samples

2차 경연에서는 1차 경연에서의 경험을 바탕으로 카메라 좌우 반전을 반영해 코딩 좌우 방향을 바꿈으로써 조종의 편의성을 높였다. 그리고 Roller를 Concept Design에서 Vertical Roller 2로 바꾸자는 의견을 반영해서 Sample 수집 효율을 높였다. 그 결과 6개중에 5개의 Sample을 모았고, 수집하는 데 소요된 시간도 적어 많은 점수를 획득할 바탕을 마련했다.

Event & Mission 3 - Hall of Mars, Sand Swamp

Hall of Mars는 Roller를 추가적인 실험없이 Vertical Roller 2로 바꾼 탓인지 언덕을 올라가는 데에 조마조마 했지만 다행히도 언덕을 올라갈 수 있었다. 대칭이라는 점을 고려했을 때 무게 중심이 그대로 유지된다는 것과 대각선 길이가 길어지더라도 토크값이 크게 안바뀌꺼라는 과감한 판단덕 이었다. 이후에 Hall과 Mission 3는 1차 때와 마찬가지로 무난하고 빠르게 통과했다.

Mission 4 & Event - Line Tracer & Wall of Mars

Line Tracer에서는 1차 때에 180도 회전이 살짝 더 많이 돌았던 점을 개선하여 모터 회전수를 줄였습니다. 그 결과 원기둥을 밀치고 난 이후의 수순도 실험해 보았을 때와 같이 매끄럽게 통과하였습니다. 다만 원기둥 2개를 깔끔하게 밀지 못해 약간 감점을 당했다. 원인을 분석한 결과 급하게 차량을 보완하던 도중 Pushing Board를 상단 부분을 잘못 조립했기 때문이었다. 도착지점에 들어왔을 때 시간도 상대적으로 많이 남아있어서, Sample을 더 수집했다면 최고 기록을 세울 수 있지 않았을까라는 아쉬움이 남았다.

초초한 마음으로 다른 조들의 경연을 지켜보았고, 그 결과 예상치도 못했던 1등이라는 결과를 손에 쥐게 되었다. 그동안 함께 밤새고 로봇의 코드에 울고 웃던 우리의 시간들을 보상해준 것 같아서 팀원 모두가 기뻐했다. 서로가 나서서 문제를 찾고 고쳐나가고, 첫 시도에 좌절하지 않고 끝까지 노력한 덕분에 1등으로 치고 오르는 반전을 만들 수 있었다.

Result : 356 points

7. Reference

- [1] JOHN J.UICKER,JR.GORDON R.PENNOCK, JOSEPH E.SHIGLEY, 2012, "Theory of machines and mechanisms 4/e", Info-Tech Corea
- [2] ZUM 학습백과, 'Gear', Internet:
<http://study.zum.com/book/12098>, unknown [Jun. 9. 2017]
- [3] unknown, 'Sonar sensor', Internet:
http://education.rec.ri.cmu.edu/content/electronics/boe/ultrasonic_sensor/1.html, unknown [Jun. 10. 2017]
- [4] Wikipedia, 'Motor', Internet:
<https://ko.wikipedia.org/wiki/%EC%A0%84%EB%8F%99%EA%B8%B0>, unknown [Jun. 9. 2017]
- [5] Kimmy-Ro, 'Motor & Fleming's left hand rule', Internet:
http://kimmy-ro.kimm.re.kr/learn/lea_environment_v10.html, unknown [Jun. 9. 2017]
- [6] 한국물리학회, "질량중심", Internet:
<http://terms.naver.com/entry.nhn?docId=3537291&cid=58577&categoryId=58577>, unknown [Jun. 10. 2017].
- [7] 자동차 용어사전, "무한궤도", Internet:
<http://terms.naver.com/entry.nhn?docId=1653943&cid=42330&categoryId=42330>, May.25 2012 [Jun. 14. 2017]
- [8] 이명수, 김상섭, 「차량무게중심의 측정 및 추정에 관한 연구」, (Transactions of KSAE, Vol. 18, No. 5, 2010) pp.91-99
- [9] 강현석, 곽윤근, 최현도, 정해관, 김수현, 「주행로봇 제어를 위한 험지의 최대마찰계수 추정」, (제어 · 로봇 · 시스템학괴 논문지 제 14 권, 제 10호 2008, 10), 11쪽