

프로젝트 보고서

<F1TENTH 제작 및 HECTOR SLAM 구동>

프로젝트
프로젝트 기간
이 름

2021학년도 2학기 학부연구생
2021.09.01 ~ 2021.11.30
신건희

1. Abstract	2
2. Building the F1TENTH Car	
1. Base Chassis 구축	2
2. Powerboard 제작	3
3. Platform Deck 설계 (3D-Print)	3~5
4. Platform Deck에 부품 부착	6~7
5. 부품 연결	7~8
3. Configure F1TENTH Car	
1. Lidar Sensor(Hokuyo 10LX) Xavier 연결	8~10
2. VESC를 통한 모터 제어	10~13
3. Logitech joypad-f710를 이용하여 F1TENTH 구동	14~16
4. Hector Slam 구동	
1. Hector Slam	17
2. 구동	17~20

1. Abstract

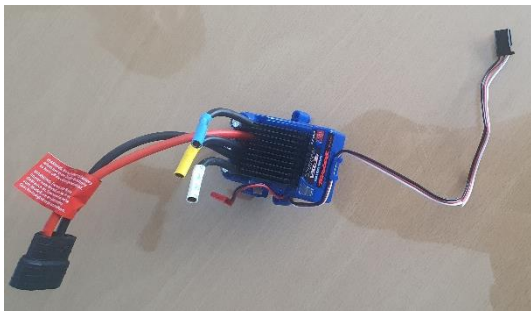
2021학년도 2학기 학부연구생 프로그램에 참여하여 4개월동안 인하대학교 전기공학과 LCIS 연구실에서 진행한 프로젝트이다. F1TENTH는 차량을 1/10크기로 축소한 자율주행 플랫폼이다. 하드웨어를 구축하고 소프트웨어 조작을 통해 차량을 구동했고 오픈소스 Hector Slam을 이용하여 Slam을 구현했다. 차량 부품들과 센서를 직접 부착하고 조작하는 과정에서 자율주행 자동차에 사용되는 기술에 대해 포괄적으로 이해할 수 있었다.

2. Building the F1TENTH Car

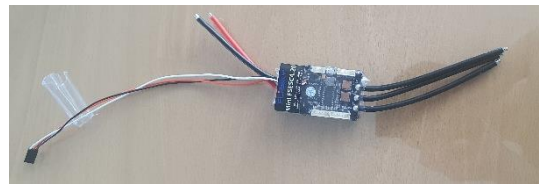
1. Base Chassis 구축

1. RC Car 차체에서 불필요한 부품 제거

- 기존에 Traxxas Slash 4x4 Platinum Edition; 1/10 Scale Brushless Pro 4WD의 VESC는 PID 제어를 위한 Gain 조절이 불가능하여 제거하고 flipsky Vesc Mini 로 대체하였다.

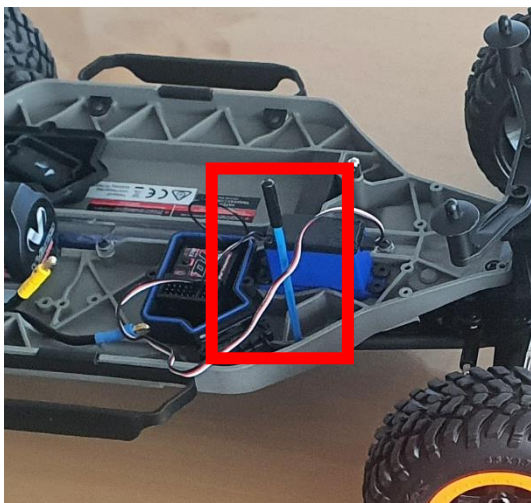


<FIG.1 F1/10 Scale Brushless Pro 4WD>

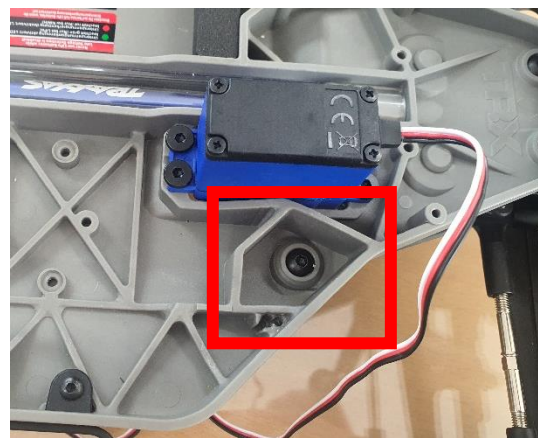


<FIG2. Flipsky Vesc Mini>

- 본 프로젝트는 자율 주행을 목표로 연구하기 때문에 RC카 조종을 위한 수신기를 제거했다.



<FIG.3 F1/10 수신기 제거 전>



<FIG4. 수신기 제거 후>

2. Powerboard 제작

- Manual에 따라 소자 극성을 고려하여 부착하고 납땜을 했다. 0.1uf 커패시터는 극성과 무관하다.
- 납땜을 진행하던 중 2단자 블록을 반대 극성으로 납땜했고 수정하기 위해 소자를 뽑는 과정에서 플라스틱 기판이 분리되는 일이 발생했다. 그래서 새로운 기판으로 다시 납땜을 진행했다.



<FIG.5 Powerboard 뒷 면>

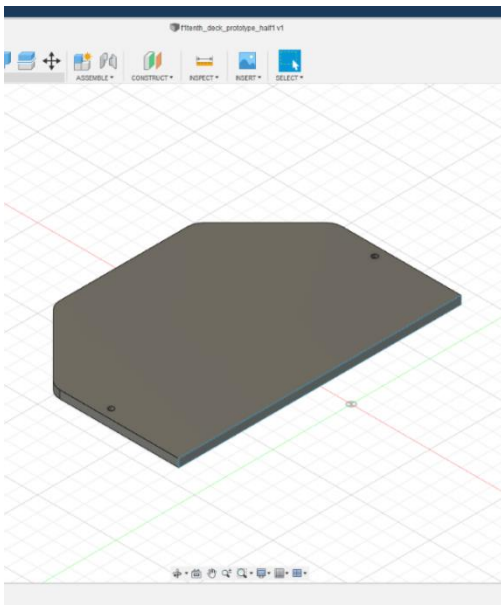


<FIG6. Powerboard 앞 면 >

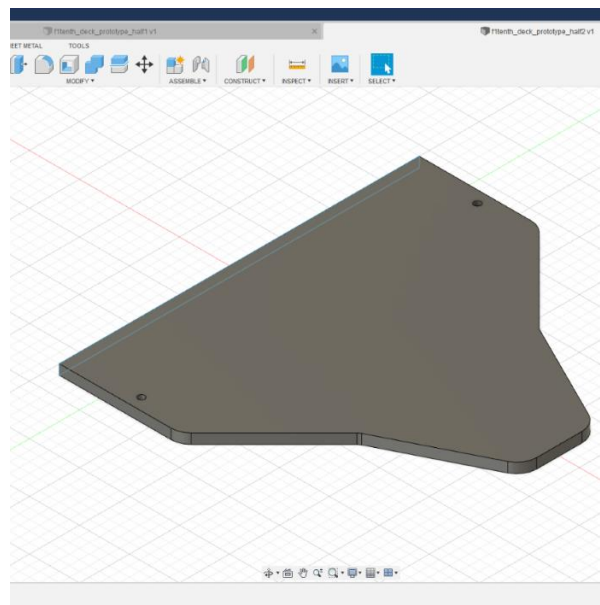
3. Platform Deck 설계 (3D-Print)

1. 프로토타입 제작

- F1TENTH 홈페이지의 build guide의 Slash 모델과 본 연구에서 보유한 Slash 모델이 달라 홈페이지에서 제공되는 Platform 도면과 치수가 일치하지 않아 사용이 불가능했다.
- Chassis의 치수를 직접 재고 Autodesk Fusion 360을 이용하여 도면을 설계했다.
- 사용하려고 하는 3D 프린터인 Ender3 모델의 최대 가능 규격보다 설계한 도면의 크기가 커서 반으로 나누어 재설계했다.



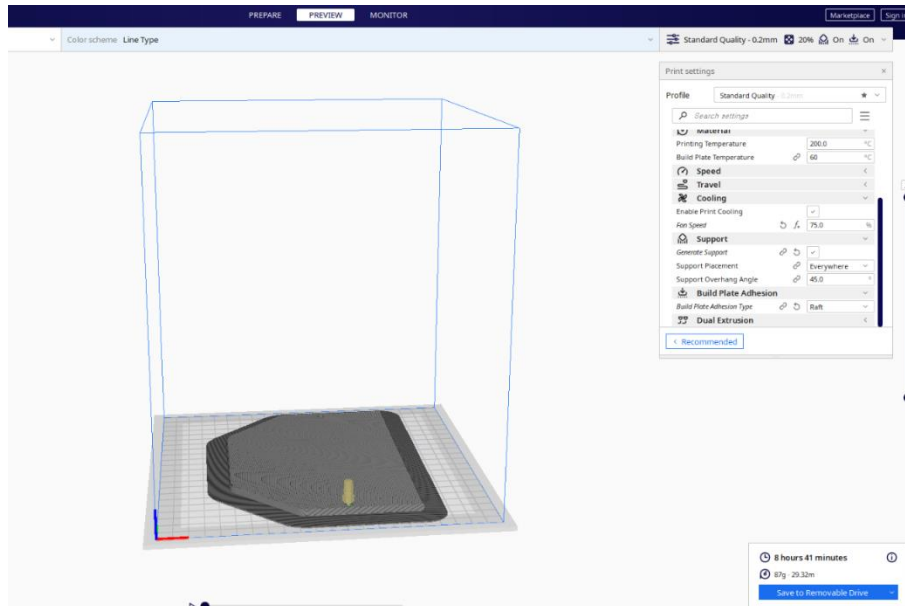
<FIG.7 Deck 뒤쪽>



<FIG8. Deck 앞쪽>

2. 3D 프린터 출력 과정

- 제작한 3D 도면을 STL파일로 변환했다.
- 변환한 STL 파일을 3D프린팅소프트웨어 CURA에 업로드.
- 프린팅 환경 설정(support의 유무, buildplate의 유무, nozzle 및 bed 온도 설정, 노즐의 속도 설정 등등)



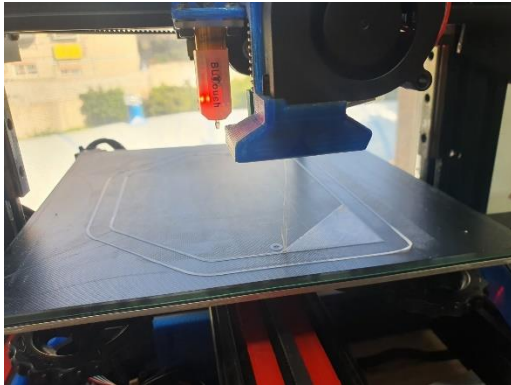
<FIG.9 CURA에 STL 파일을 업로드하여 설정을 끝낸 UI>

- 설정한 것을 Slice하여 G-code 파일로 변환 후 USB 저장
- USB를 3D 프린터에 장착 후 노즐과 베드의 온도를 각각 200°C, 60°C 로 올려주고 필라멘트를 뽕족하게 만든 후 노즐에 장착 (위 작업을 거치지 않으면 노즐 부위 고장 발생)
- 3D 프린터 UI의 'Print' 버튼에서 해당 G-code 파일을 선택하여 프린트 실행

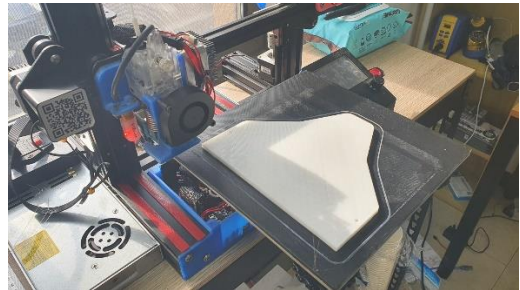


<FIG.10 출력이 진행될 때 3D 프린터 UI>

- 제작 및 완성 과정(약 6시간 소요)

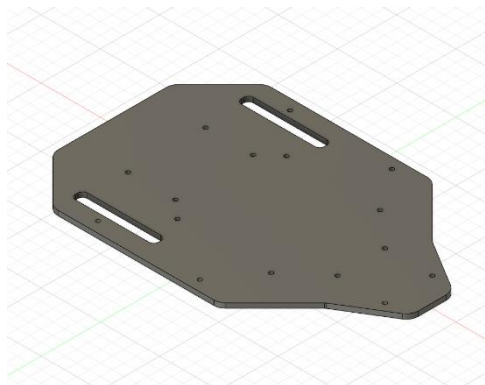


<FIG.11 출력 초기 모습>



<FIG12. 출력 완성 모습>

- 제작 후 약간의 오차가 발생했고 다시 치수를 정밀하게 재어 도면을 재설계했다. 도면을 반으로 분리해서 제작하면 안정적이지 못하고 이어 붙이기 어렵다고 판단해 대형 3D프린터(LE3340)에서 단일 부품으로 제작하기로 했다. 또한, 모터와 VESC가 연결할 커넥터, 배터리와 Powerboard 연결선이 지나갈 자리가 필요하여 양 옆에 큰 구멍을 뚫어 주었다. Jetson Xavier가 Deck에 고정되어야 할 자리와 차체와 Deck을 고정하기 위해 나사가 들어갈 구멍을 뚫어 주었다.



<FIG.13 Platform Deck 최종본>



<FIG.13 대형 3D프린터(LE3340)에서 나온 결과물>

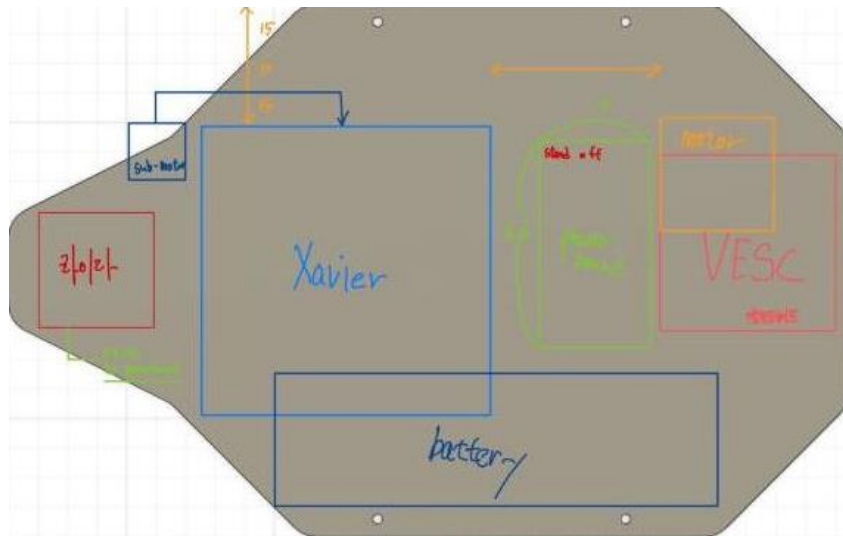


<FIG14. 최종 완성본>

4. Platform Deck에 부품 부착

1. Platform Deck에 부품 부착 구상도

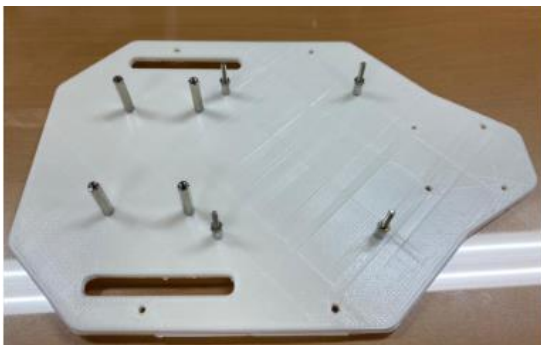
- Deck에 부착해야 할 부품은 라이다(LiDAR, Hokuyo 10LX), Xavier, Vesc, Powerboard로 4가지이다.
 - Vesc: Main motor와 Sub motor와 연결되어 종방향, 횡방향 모터 제어
 - Powerboard: 외부 배터리를 통해 전원을 공급받고 Vesc, 라이다에 전원을 공급한다.
- F1TENTH RC Car 매뉴얼에 맞게 구상도를 제작했다.



<FIG14. 부품 부착 구성도>

2. Platform Deck에 Xavier 부착

- Xavier 초기 구성품에는 회로를 보호하기 위한 Leg가 두개 있었다. 그러나 본 연구에서 Deck에서 Leg를 관통하여 Xavier를 고정할 수 있는 길이의 나사를 구하기 힘들었다.
- Leg를 제거하고 작은 standoff와 너트를 활용하여 보유한 나사를 통해 회로를 보호하면서 Xavier를 고정할 수 있었다.



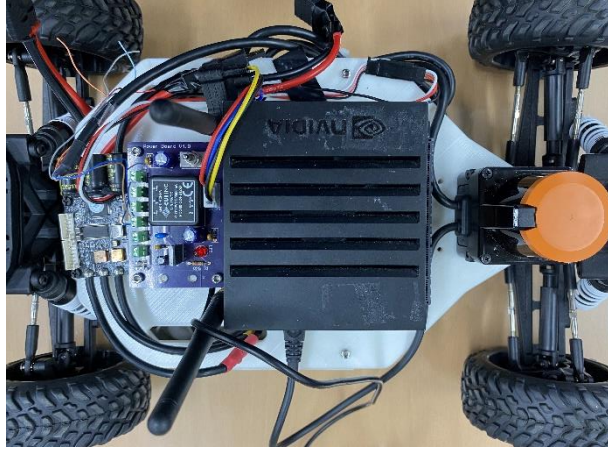
<FIG.15 Xavier 부착 전 Deck의 모습>



<FIG16. Leg를 제거하고 부착한 모습>

3. Platform Deck에 Esc, Powerboard, 라이다 부착

- Deck의 앞부분에 라이다를 부착했다.
- Xavier 바로 뒤에 Standoff를 이용하여 Powerboard를 고정했다.
- Powerboard 뒤에 테이프를 이용하여 Vesc를 부착했다.



<FIG.17 Platform에 부품을 부착한 모습>

5. 부품 연결

1. Vesc – Motor 연결

- Vesc 연결 부위와 Motor 연결 부위가 모두 Male인 것을 확인했다.
- Male 간의 연결하는 Connector를 구매하는데 실패했다.
- 그래서 Male 접합 부분을 납땜을 하고 수축 튜브로 고정했다.



<FIG.18 Male 접합부 납땜 후 수축튜브로 고정>

2. 배터리- Vesc -Powerboard 연결.

- Vesc와 Powerboard에 배터리로 전원을 공급하기 위해서 3s Connector를 사용했다.

- 3s Connector에서 Powerboard와 결합되는 단자가 4핀이어야하나, 보유중인 케이블은 3핀이어서 4핀에 맞게 납땜을 진행하였다. 하지만 정상 작동되지 않았고, 멀티미터로 전압을 측정했을 때 Powerboard 소자에서 전압과 전류가 측정되지 않는 것을 확인하고 Connector를 알맞게 다시 주문했다.
- 새로 구매한 Connector의 배터리 연결부가 보유한 배터리 연결부와 규격이 다른 것을 확인했다. 그래서 3s Connector의 배터리 연결부를 절단하고 배터리 연결부와 규격이 맞는 연결부로 교체했다. 납땜 후 수축튜브, 절연 테이프로 고정했다.



<FIG19. 연결부 교체 후 고정된 모습>

3. Configure F1TENTH Car

1. Lidar Sensor(Hokuyo 10LX) Xavier 연결

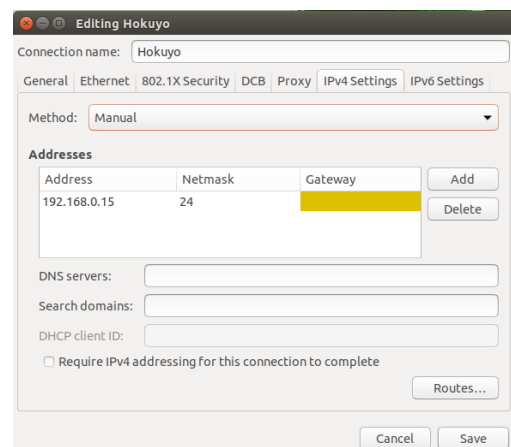
1. Configure the eth() network

1. Hokuyo ip address 설정

\$ifconfig

```
lclis@lclis-desktop:~$ config
bash: config: command not found
lclis@lclis-desktop:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu
1500
    inet 172.17.0.1 netmask 255.255.0.0 bro
adcast 172.17.255.255
    ether 02:42:ae:c7:05:50 txqueuelen 0 (E
thernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame
    0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrie
r 0 collisions 0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
mtu 1500
    inet 192.168.0.15 netmask 255.255.255.0
broadcast 192.168.0.255
    inet6 fe80::9e35:4d45:6c99:7346 prefixle
n 64 scopeid 0x20<link>
    ether 00:04:4b:e5:7a:dd txqueuelen 1000
(Ethernet)
```

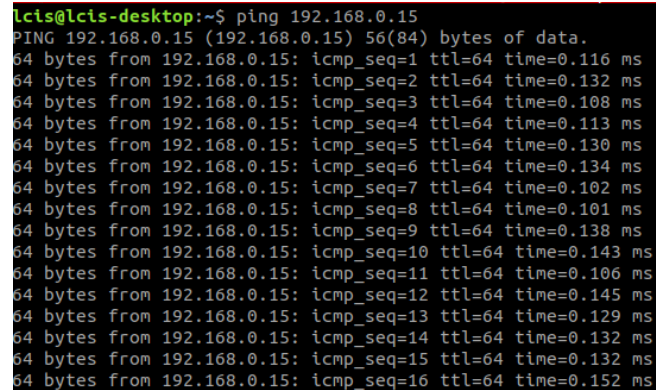
<FIG.20 eth() network Address 확인>



<FIG.21 Hokuyo 연결을 위한 Address 설정>

2. Ping으로 확인하기

```
$ ping 192.168.0.15
```

A terminal window showing the execution of a ping command. The prompt is 'lcis@lcis-desktop:~\$'. The command is 'ping 192.168.0.15'. The output shows 'PING 192.168.0.15 (192.168.0.15) 56(84) bytes of data.' followed by 16 lines of '64 bytes from 192.168.0.15: icmp_seq=1 ttl=64 time=0.116 ms' through 'icmp_seq=16 ttl=64 time=0.152 ms'.

```
lcis@lcis-desktop:~$ ping 192.168.0.15
PING 192.168.0.15 (192.168.0.15) 56(84) bytes of data.
64 bytes from 192.168.0.15: icmp_seq=1 ttl=64 time=0.116 ms
64 bytes from 192.168.0.15: icmp_seq=2 ttl=64 time=0.132 ms
64 bytes from 192.168.0.15: icmp_seq=3 ttl=64 time=0.108 ms
64 bytes from 192.168.0.15: icmp_seq=4 ttl=64 time=0.113 ms
64 bytes from 192.168.0.15: icmp_seq=5 ttl=64 time=0.130 ms
64 bytes from 192.168.0.15: icmp_seq=6 ttl=64 time=0.134 ms
64 bytes from 192.168.0.15: icmp_seq=7 ttl=64 time=0.102 ms
64 bytes from 192.168.0.15: icmp_seq=8 ttl=64 time=0.101 ms
64 bytes from 192.168.0.15: icmp_seq=9 ttl=64 time=0.138 ms
64 bytes from 192.168.0.15: icmp_seq=10 ttl=64 time=0.143 ms
64 bytes from 192.168.0.15: icmp_seq=11 ttl=64 time=0.106 ms
64 bytes from 192.168.0.15: icmp_seq=12 ttl=64 time=0.145 ms
64 bytes from 192.168.0.15: icmp_seq=13 ttl=64 time=0.129 ms
64 bytes from 192.168.0.15: icmp_seq=14 ttl=64 time=0.132 ms
64 bytes from 192.168.0.15: icmp_seq=15 ttl=64 time=0.132 ms
64 bytes from 192.168.0.15: icmp_seq=16 ttl=64 time=0.152 ms
```

<FIG.22 Ping으로 Hokuyo와 연결이 되는지 확인하는 모습>

2. ROS workspace setup.

1. Clone the following repository into a folder on your computer

```
$ cd ~/sandbox
```

```
$ git clone https://github.com/f1tenth/f1tenth_system
```

2. Create a workspace folder if you haven't already, here called f1tenth_ws, and copy the f1tenth_system folder into it.

```
$ mkdir -p f1tenth_ws/src
```

```
$ cp -r f1tenth_system f1tenth_ws/src/
```

3. You might need to install some additional ROS packages.

```
$ sudo apt-get update
```

```
$ sudo apt-get install ros-melodic-driver-base
```

4. Make all the Python scripts executable.

```
$ cd f1tenth_ws
```

```
$ find . -name "*.py" -exec chmod +x {} \;
```

5. Move to your workspace folder and compile the code

```
$ catkin_make
```

6. Finally, source your working directory into your shell using

```
$ source devel/setup.bash
```

3. Test the Lidar

1. Hokuyo Lidar 는 Xiver 와 Ethernet 으로 연결했기 때문에 F1system 내의 hokuyo node 패키지를 urg_node 패키지로 교체.
2. Urg_node 패키지에 필요한 의존성 패키지를 다운

```
$ cd ~/sandbox/f1tenth_ws/src/f1tenth_system/rug_node
```

```
$ rodep install -from-paths . -ignore-src -r -y
```

```
$ rosdep update
```
3. Complie the F1tenth_ws

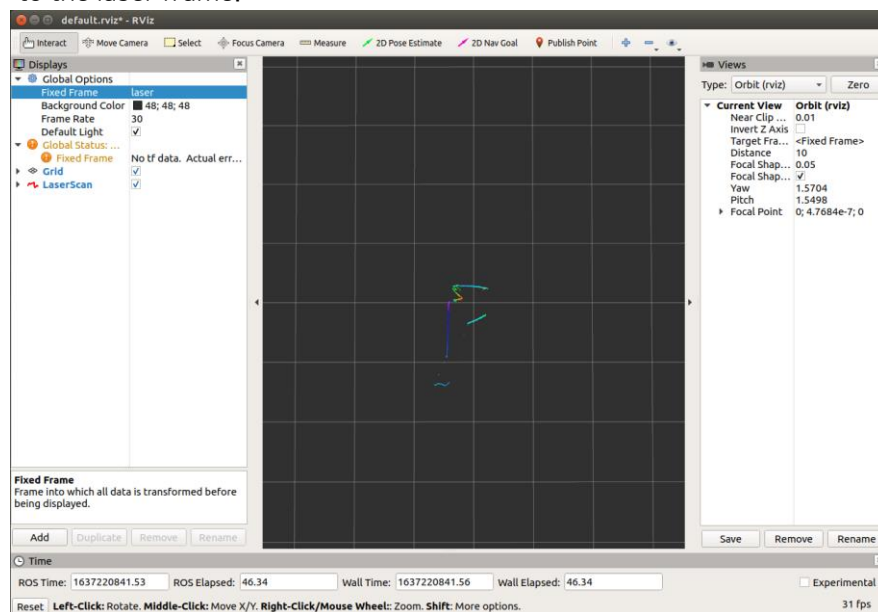
```
$ cd ~/sandbox/f1tenth_ws
```

```
$ catkin_make
```
4. Visulaize point cloud of Lidar through Rviz

```
$ roscore
```

```
$ rosrn urg_node urg_node _ip_address:="192.168.0.10"
```

```
$ rosrn rviz
```
5. In Rviz tool, highlight the LaserScan topic and switch from viewing in the map frame to the laser frame.



<FIG23. Rviz를 통해 확인한 Lidar Point Cloud>

2. VESC를 통한 모터 제어

1. BLDC 모터, FOC 제어

- 회전자가 브러시리스 영구 자석이므로 회전자에 전류 인가가 필요 없다. 고정자의 전류를 제어함으로써, 임의의 방향과 크기의 자기장이 생성될 수 있다. BLDC 모터의 토크는 회전자와 고정자 필드 사이의 인력과 반발력에 의해 만들어진다. BLDC 모터의 토크는 회전자인 영구 자석의 회전으로 인한 역기전력과 고정자에 흐르는 3상 전류에 비례한다.

- 따라서 본 연구에서는 VESC Tool에서 BLDC 모터의 고정자 전류를 제어함으로써, 토크와 속도를 제어했다.
- 제어 방법으로는, VESC TOOL에 내장된 FOC 제어법을 사용하였다. FOC 제어는 토크를 3상 전류 공간 벡터 i 의 q 축 성분만을 조절하여 제어할 수 있음을 의미한다. 즉, 자속을 기준으로 3상 공간상에서 전류의 크기와 방향을 제어하는 기법을 자식 기준 제어(Field Oriented Control; FOC)라고 부른다.
- Flux(d) 및 Torque(q) 두개의 생성 성분으로 분해하여 고정자 필드에 직각인 회전자의 영구 자석에서 Flux가 계속 생성되도록 모터의 고정자 권선의 전류를 제어한다.

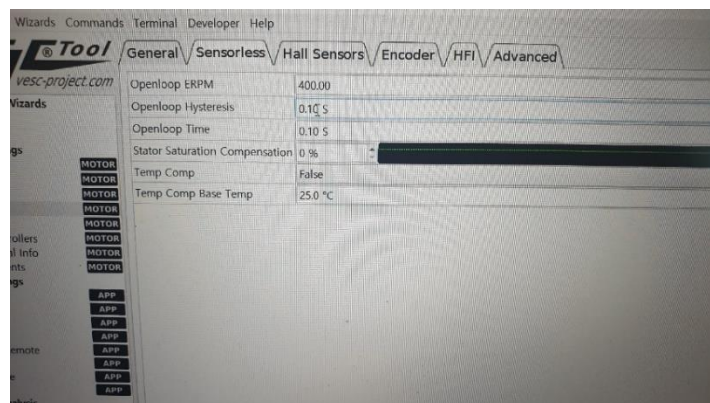
2. VESC Tool 설정

1. AutoConnect 연결
2. VESC 펌웨어 업데이트
3. Load Motor Configuration XML을 업로드 후 M버튼을 눌러 모터 구성을 적용한다.
4. Tool 화면 하단의 화살표 방향을 따라 4개의 버튼을 눌러 Parameters를 측정한다.



<FIG.24 VESC Tool 화면 구성>

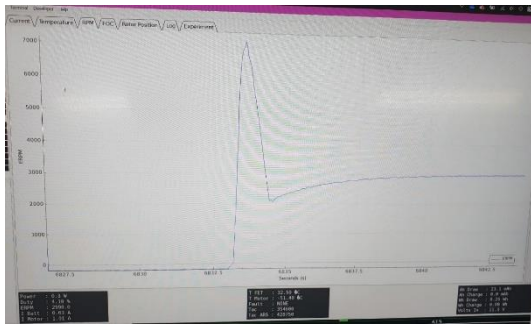
5. Sensorless Tab서 Openloop Hysteresis 및 Openloop Time을 0.01로 변경



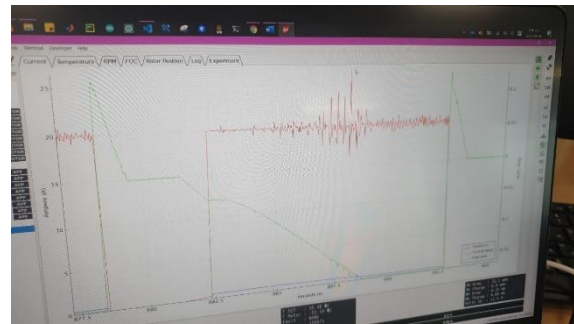
<FIG.25 Openloop parameters 설정>

3. PID Gain 튜닝

- RT 버튼을 누르고 PID CONTROLLER에서 PID게인을 변경하며 Realtime Data 탭에서 RPM의 계단 응답 파형을 확인하면서 정상상태 오차가 적고 Stable한 RPM 파형을 만들 수 있도록 PID 게인 튜닝을 진행
- Kp 설정, 목표 속도인 3000RPM에 빠르게 도달하게 하기위해 P게인을 증가시키니 RPM의 파형에서 Overshoot이 증가하였고 전류의 파형의 불안정성이 심해졌다.
- Kp 값이 크면, 시스템은 목표 값에 빨리 도달하지만 시스템은 불안정한 상태가 되고, 목표 값 근처에서 계속 진동한다.
- Kp 값이 작으면, 목표점에 늦게 도달하게 되고, 목표 값 아래에서 일정한 오차를 가지며 안정된 상태가 유지된다. 시스템이 목표 값에 도달하는데 필요한 충분한 힘이 공급되지 못하기 때문에 목표 값에 도달하지 못했다.

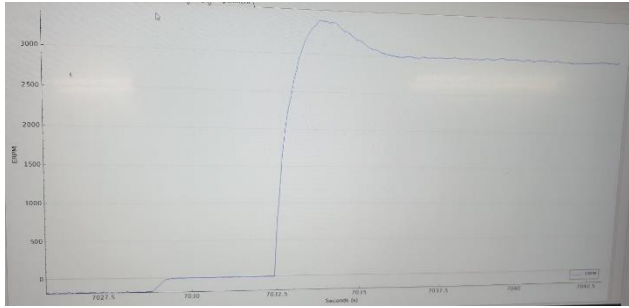


<FIG.26 Kp 조정시 RPM 파형>

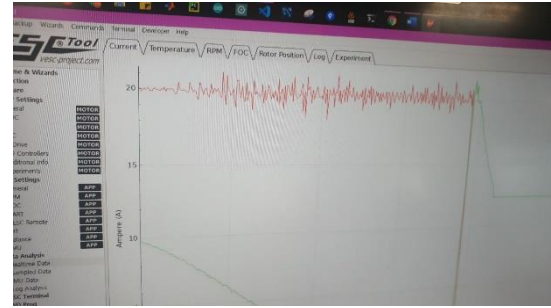


<FIG27. Kp tuning 전류 파형>

- Ki 설정, P게인은 목표치에 정확히 수렴하지 못하기 때문에, I게인을 튜닝하여 정상상태 오차를 감소시켜 목표 값에 최대한 근접하게 함.
- Ki 값이 크면, 시스템은 목표 값에 빨리 도달하지만 하지만 파형의 Oscillation이 커진다. 높은 상승부가 있다 하더라도, P제어와는 달리 시스템이 불안정한 상태로 남지 않고 몇번의 진동 후 목표 값에 도달하고 안정된 상태로 유지된다.
- Ki 값이 작으면, 늦게 목표 값에 도달하고, Oscillation이 적다. 진동에 비해 빠르게 stable해진다. 시스템이 도달하는 시간과 진동의 정도를 맞추기 위해 최적화가 필요하다.

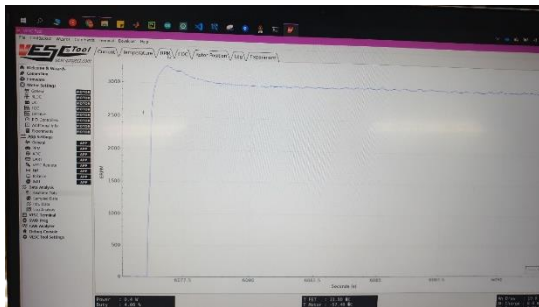


<FIG.28 Ki 조정 시 RPM 파형>

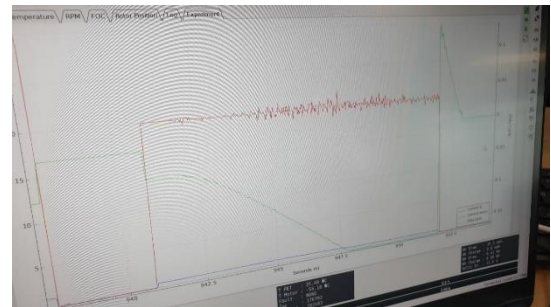


<FIG.29 Ki 조정 시 전류 파형>

- Kd 설정, D게인을 증가시켜 RPM의 파형과 전류의 파형의 Overshoot을 감소시키고 파형을 안정적으로 만들었다.
- Kd 값이 크면, Oscillation과 Overshoot를 감소시키고 시스템의 안정성을 증가시킨다. 또한 시스템의 반응속도를 향상시킨다.

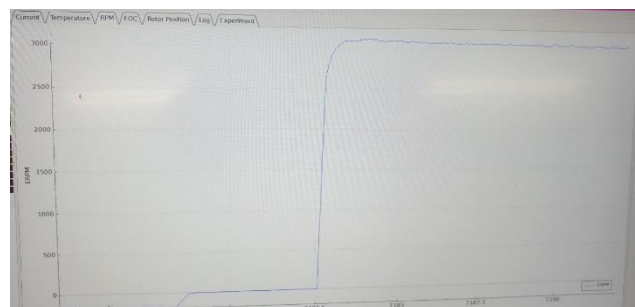


<FIG.30 Kd 조정 시 RPM 파형>



<FIG31. Kd 조정 시 전류 파형>

- PID gain을 적절히 조절하면서 응답 속도가 빠르고, Overshoot가 적은 Stable한 RPM파형을 찾을 수 있었다.



<FIG.32 최종 PID 값에 따른 RPM 파형>

3. Logitech joypad-f710를 이용하여 F1TENTH 구동

1. 초기 설정

- 본 연구에서 Joypad 활용 방안은 Joypad와 Xavier 연결 및 데이터 통신, 종방향, 횡방향 제어이다.
- 처음에 다운 받은 F1TENTH 파일에 joy 패키지가 존재하므로 추가적으로 다운 받아야 할 패키지는 없다.
- 초기 설정에는 여러 문제가 존재했다. 먼저 Joypad와 Xavier 연결 시 Joypad의 전원을 키면 연결이 끊기는 현상, 횡방향 제어 시 조이스틱이 아닌 LT버튼으로만 제어가 가능, 종방향 제어 시 속도를 제어하지 못하고 항상 최대 속력으로 고정되었다.

2. 문제점 해결

- Joypad 와 Xavier 연결 시 Joypad 전원을 키면 연결이 끊기는 현상은 Joypad의 X/D모드를 변경해서 해결했다. X mode는 X input 방식으로 최근 출시한 게임과 호환되고 D mode는 DirectInput 방식으로 구형 게임과 호환이 된다. 기존의 D mode에서는 Joypad 전원을 키면 연결이 끊겼지만 X mode에서는 Joypad 전원을 켜도 연결이 끊기지 않았다.



<FIG.33 LogitechJoypad-f710>

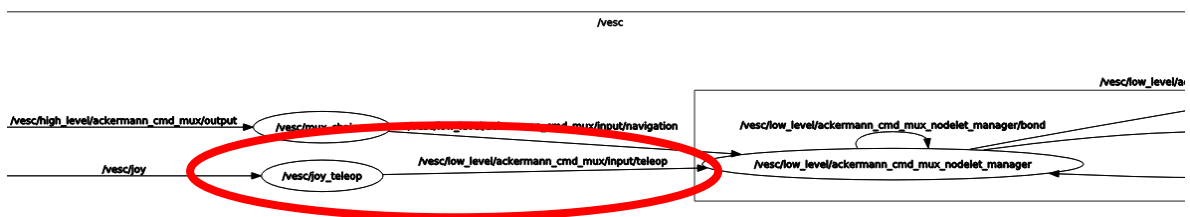
- LT 버튼으로만 횡방향 제어가 가능 한 문제는 소스 코드 분석을 통한 Parameter 변경으로 해결하였다.
- Joypad 입력 값을 확인하여 우측 조이스틱을 움직이면 Axis 3의 값이 변하는 것을 알 수 있었다. 이를 통해 우측 조이스틱의 입력에는 문제가 없다는 것을 확인 할 수 있었다.
- Rqt Graph를 통해 조이스틱 입력 데이터의 흐름을 확인했다. 조이스틱 데이터가 입력되는 /vesc/joy_node, /vesc/joy_teleop 노드와 /vesc/low_level/Ackermann_cmd_mum/input/teleop 토픽에 문제 있을 것으로 예상했다.
- /vesc/joy_teleop 노드의 yaml 파일에서 drive.steering_angle의 default axis가 2로 설정

되어 있어 LT버튼으로만 횡방향 제어가 가능했음을 확인할 수 있었다. 이를 우측 조이스틱의 Axis 값인 3으로 변경하여 우측 조이스틱을 이용한 횡방향 제어를 할 수 있었다.

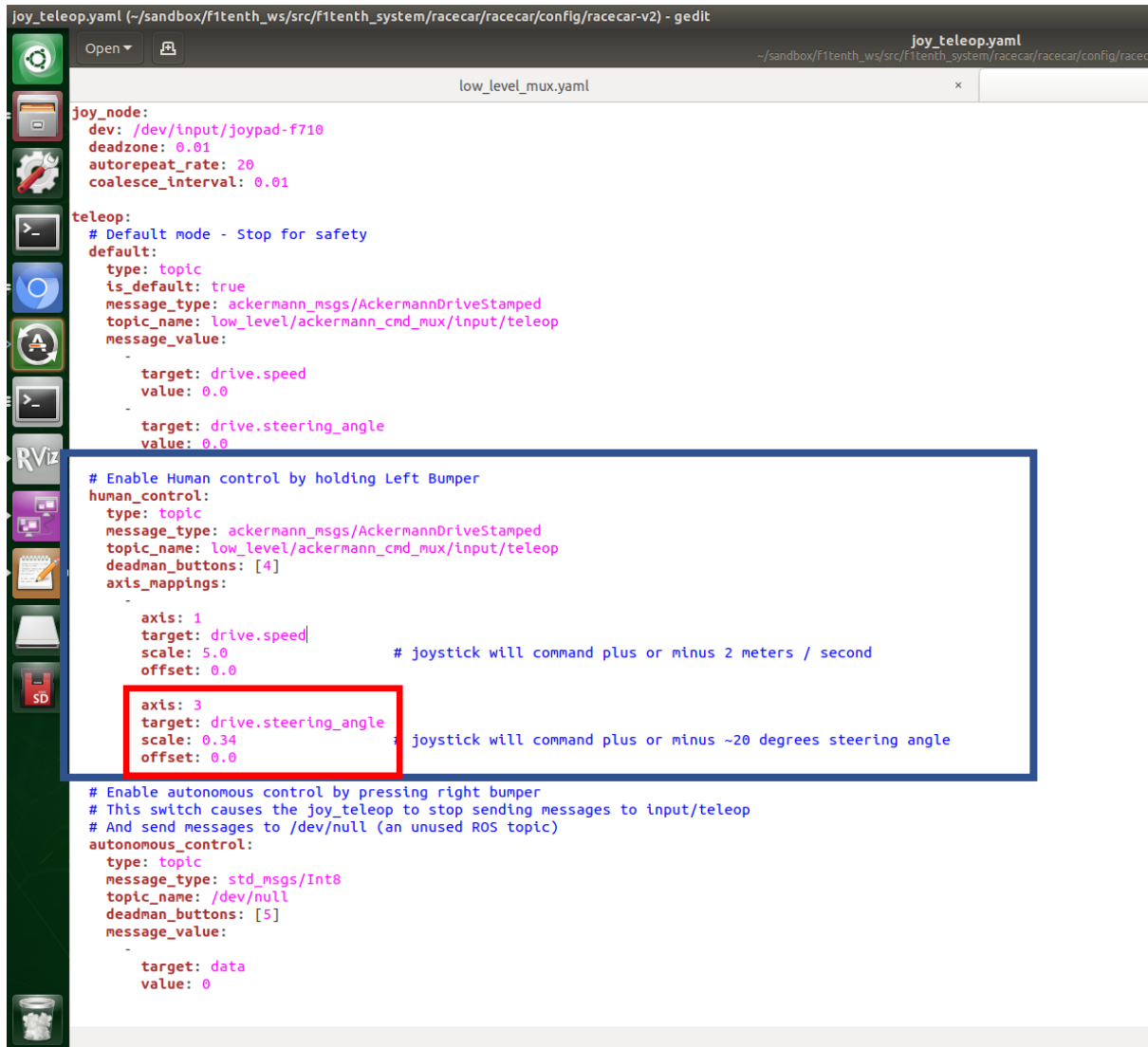
```

lclis@lclis-desktop: ~
lclis@lclis-desktop: ~ 80x24
0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9
Axes: 0: 0 1:-19687 2:-32767 3: 777 4: -2 5:-32767 6: 0 7:
0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9
Axes: 0: 0 1:-17874 2:-32767 3: 777 4: -2 5:-32767 6: 0 7:
0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9
Axes: 0: 0 1:-16579 2:-32767 3: 777 4: -2 5:-32767 6: 0 7:
0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9
Axes: 0: 0 1:-15802 2:-32767 3: 777 4: -2 5:-32767 6: 0 7:
0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9
Axes: 0: 0 1:-14507 2:-32767 3: 777 4: -2 5:-32767 6: 0 7:
0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9
Axes: 0: 0 1:-12953 2:-32767 3: 777 4: -2 5:-32767 6: 0 7:
0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9
Axes: 0: 0 1:-11917 2:-32767 3: 777 4: -2 5:-32767 6: 0 7:
0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9
Axes: 0: 0 1:-9585 2:-32767 3: 777 4: -2 5:-32767 6: 0 7:
0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9
Axes: 0: 0 1:-3628 2:-32767 3: 777 4: -2 5:-32767 6: 0 7:
0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9
Axes: 0: 0 1: -2 2:-32767 3: 0 4: -2 5:-32767 6: 0 7:
0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9
:off 10:off
  
```

<FIG.34 Joypad 입력 값 확인>



< FIG.35 Rqt_Graph 를 통해 입력 데이터 흐름 확인>



<FIG.36 /vesc/joy_teleop 노드의 yaml 파일>

- 종방향 제어 시 속도제어가 불가하고 항상 최대 속력으로 고정되는 문제 또한 횡방향 제어 문제 해결과 같이 소스 코드 분석을 통한 Parameter 변경으로 해결하였다.
- Rqt_Graph를 통해 조이스틱의 input 값들이 joy 패키지에서 Publish 되어 vesc/joy_teleop가 Subscribe하는 것을 확인했다. 현재 low_level_mux.yaml 파일의 drive.speed의 scale이 5로 설정되어 있어, 좌측 조이스틱의 output data값이 RC차량의 최대 속도값보다 커서 해당 차량 속도 제어가 되지 않는 문제점이 발생했다. 따라서 drive.speed scale을 낮추어 좌측 조이스틱의 output data값이 차량 속도 제어에 적절한 값이 되게 설정했다.

4. Hector Slam 구동

1. Hector Slam

- Joypad로 F1TENTH를 구동 할 수 있고 Lidar를 통해 데이터를 받을 수 있으므로 Mapping을 할 수 있는 구조가 갖췄다고 판단하여 Slam Open Source Package를 찾아보았다.
- 그 중 Hector Slam 오픈 소스를 찾았고 본 연구에 적용했다.
- 출처: https://github.com/tu-darmstadt-ros-pkg/hector_slam

2. 구동

- 차량의 좌표계를 'base link'로 설정

```
$ cd ~/catkin_ws/src/hector_slam/hector_mapping/launch/
```

```
$ gedit mapping_default.launch
```

```
1  <?xml version="1.0"?>
2
3  <launch>
4    <arg name="tf_map_scanmatch_transform_frame_name" default="scanmatcher_frame"/>
5    <arg name="base_frame" default="base_footprint"/>
6    <arg name="odom_frame" default="nav"/>
7    <arg name="pub_map_odom_transform" default="true"/>
8    <arg name="scan_subscriber_queue_size" default="5"/>
9    <arg name="scan_topic" default="scan"/>
10   <arg name="map_size" default="2048"/>
11   <node pkg="hector_mapping" type="hector_mapping" name="hector_mapping" output="screen">
12
13     <!-- Frame names -->
14     <param name="map_frame" value="map" />
15     <param name="base_frame" value="$(arg base_frame)" />
16     <param name="odom_frame" value="$(arg odom_frame)" />
17
18     <!-- Tf use -->
19     <param name="use_tf_scan_transformation" value="true"/>
20     <param name="use_tf_pose_start_estimate" value="false"/>
21     <param name="pub_map_odom_transform" value="$(arg pub_map_odom_transform)"/>
22
23     <!-- Map size / start point -->
24     <param name="map_resolution" value="0.050"/>
25     <param name="map_size" value="$(arg map_size)"/>
26     <param name="map_start_x" value="0.5"/>
27     <param name="map_start_y" value="0.5" />
28     <param name="map_multi_res_levels" value="2" />
29
30     <!-- Map update parameters -->
31     <param name="update_factor_free" value="0.4"/>
32     <param name="update_factor_occupied" value="0.9" />
33     <param name="map_update_distance_thresh" value="0.4"/>
34     <param name="map_update_angle_thresh" value="0.06" />
35     <param name="laser_z_min_value" value = "-1.0" />
36     <param name="laser_z_max_value" value = "1.0" />
37
38     <!-- Advertising config -->
39     <param name="advertise_map_service" value="true"/>
40     <param name="scan_subscriber_queue_size" value="$(arg scan_subscriber_queue_size)"/>
41     <param name="scan_topic" value="$(arg scan_topic)"/>
42
43     <!-- Debug parameters -->
44     <!--
45     <param name="output_timing" value="false"/>
46     <param name="pub_drawings" value="true"/>
47     <param name="pub_debug_output" value="true"/>
48     -->
49     <param name="tf_map_scanmatch_transform_frame_name" value="$(arg tf_map_scanmatch_transform_frame_name)" />
50   </node>
51
52   <!--<node pkg="tf" type="static_transform_publisher" name="map_nav_broadcaster" args="0 0 0 0 0 map nav 100"/>-->
53 </launch>
54
```

<FIG.37 Launch 파일 변경 전>

```

1  mapping_default.launch
2
3  <?xml version="1.0"?>
4
5  <launch>
6    <arg name="tf_map_scanmatch transform frame name" default="scanmatcher_frame"/>
7    <!--<arg name="base_frame" default="base_footprint"/>-->
8    <arg name="base_frame" default="base_link"/>
9    <!--<arg name="odom_frame" default="nav"/>-->
10   <arg name="odom_frame" default="base_link"/>
11   <arg name="pub_map_odom_transform" default="true"/>
12   <arg name="scan_subscriber_queue_size" default="5"/>
13   <arg name="scan_topic" default="scan"/>
14   <arg name="map_size" default="2048"/>
15   <node pkg="hector_mapping" type="hector_mapping" name="hector_mapping" output="screen">
16
17     <!-- Frame names -->
18     <param name="map_frame" value="map" />
19     <param name="base_frame" value="$(arg base_frame)" />
20     <param name="odom_frame" value="$(arg odom_frame)" />
21
22     <!-- Tf use -->
23     <param name="use_tf_scan_transformation" value="true"/>
24     <param name="use_tf_pose_start_estimate" value="false"/>
25     <param name="pub_map_odom_transform" value="$(arg pub_map_odom_transform)"/>
26
27     <!-- Map size / start point -->
28     <param name="map_resolution" value="0.050"/>
29     <param name="map_size" value="$(arg map_size)"/>
30     <param name="map_start_x" value="0.5"/>
31     <param name="map_start_y" value="0.5" />
32     <param name="map_multi_res_levels" value="2" />
33
34     <!-- Map update parameters -->
35     <param name="update_factor_free" value="0.4"/>
36     <param name="update_factor_occupied" value="0.9" />
37     <param name="map_update_distance_thresh" value="0.4"/>
38     <param name="map_update_angle_thresh" value="0.06" />
39     <param name="laser_z_min_value" value = "-1.0" />
40     <param name="laser_z_max_value" value = "1.0" />
41
42     <!-- Advertising config -->
43     <param name="advertise_map_service" value="true"/>
44     <param name="scan_subscriber_queue_size" value="$(arg scan_subscriber_queue_size)"/>
45     <param name="scan_topic" value="$(arg scan_topic)"/>
46
47     <!-- Debug parameters -->
48     <!--
49     <param name="output_timing" value="false"/>
50     <param name="pub_drawings" value="true"/>
51     <param name="pub_debug_output" value="true"/>
52     -->
53     <param name="tf_map_scanmatch transform frame name" value="$(arg tf_map_scanmatch transform frame name)" />
54   </node>
55   <node pkg="tf" type="static_transform_publisher" name="base_to_laser_broadcaster" args="0 0 0 0 0 base_link laser 100" />
56   <!--<node pkg="tf" type="static_transform_publisher" name="map_nav_broadcaster" args="0 0 0 0 0 map nav 100"/>-->
57 </launch>

```

<FIG.38 Launch 파일 변경 후>

➤ 이와 같이 설정한 후 catkin_make를 진행하면 정상적으로 build가 된다.

1. Ros 실행

\$ roscore

2. Hokuyo Lidar 실행

\$ rosrunc urg_node urg_node_ip_address:="192.168.0.10"

3. Joystick node 실행

\$ roslaunch racecar teleop.launch

4. Hector Slam 실행

\$ roslaunch hector_slam_launch tutorial.launch

5. Mapping 한 이후 이미지 파일 저장

\$ Rosrun map_server map_saver -f ~/FILE NAME"

➤ 처음 Hector Slam을 구동하였을 때 Map이 잘 그려졌으나 Rviz 상에서의 Map Size가 맞지

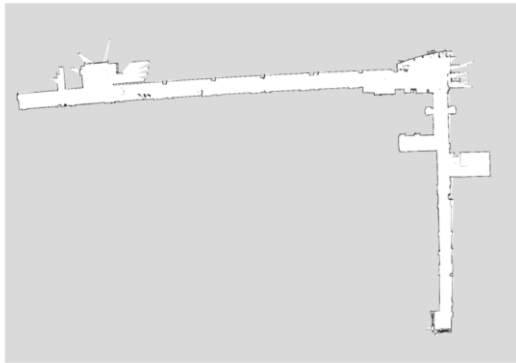
않아 Map의 크기에서 벗어나는 곳으로 가면 Mapping이 중단되는 것을 확인했다.

- 따라서 Hector Slam의 Package 상에서 Map Size를 조절해야 하는데 이것 또한 위에서 수정한 Launch 파일에서 수정한다.

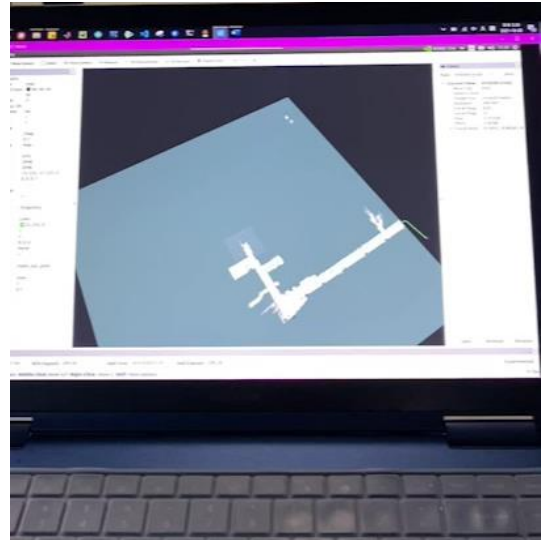
```
mapping_default.launch
1 <?xml version="1.0"?>
2
3 <launch>
4   <arg name="tf_map_scanmatch_transform_frame_name" default="scanmatcher_frame"/>
5   <!--<arg name="base_frame" default="base_footprint"/>-->
6   <arg name="base_frame" default="base_link"/>
7   <!--<arg name="odom_frame" default="nav"/>-->
8   <arg name="odom_frame" default="base_link"/>
9   <arg name="pub_map_odom_transform" default="true"/>
10  <arg name="scan_subscriber_queue_size" default="5"/>
11  <arg name="scan_topic" default="scan"/>
12  <arg name="map_size" default="2048"/> Map의 크기를 조절하고 싶다면 '2048'을 조절하면 된다.
13  <node pkg="hector_mapping" type="hector_mapping" name="hector_mapping" output="screen">
14
15    <!-- Frame names -->
16    <param name="map_frame" value="map" />
17    <param name="base_frame" value="$(arg base_frame)" />
18    <param name="odom_frame" value="$(arg odom_frame)" />
19
20    <!-- Tf use -->
21    <param name="use_tf_scan_transformation" value="true"/>
22    <param name="use_tf_pose_start_estimate" value="false"/>
23    <param name="pub_map_odom_transform" value="$(arg pub_map_odom_transform)"/>
24
25    <!-- Map size / start point -->
26    <param name="map_resolution" value="0.050"/>
27    <param name="map_size" value="$(arg map_size)"/> "map_start_x"와 "map_start_y"는 좌표계가 시작하
28    <param name="map_start_x" value="0.5"/> 는 지점을 map 상에서 비율로 나타낸 것이다. 각
29    <param name="map_start_y" value="0.5" /> 각 0.2,0.2로 바꿔주었다.
30    <param name="map_multi_res_levels" value="2" />
31
32    <!-- Map update parameters -->
33    <param name="update_factor_free" value="0.4"/>
34    <param name="update_factor_occupied" value="0.9" />
35    <param name="map_update_distance_thresh" value="0.4"/>
36    <param name="map_update_angle_thresh" value="0.06" />
37    <param name="laser_z_min_value" value = "-1.0" />
38    <param name="laser_z_max_value" value = "1.0" />
39
40    <!-- Advertising config -->
41    <param name="advertise_map_service" value="true"/>
42    <param name="scan_subscriber_queue_size" value="$(arg scan_subscriber_queue_size)"/>
43    <param name="scan_topic" value="$(arg scan_topic)"/>
44    <!-- Debug parameters -->
45    <!--
46    <param name="output_timing" value="false"/>
47    <param name="pub_drawings" value="true"/>
48    <param name="pub_debug_output" value="true"/>
49    -->
50    <param name="tf_map_scanmatch_transform_frame_name" value="$(arg tf_map_scanmatch_transform_frame_name)" />
51  </node>
52  <node pkg="tf" type="static_transform_publisher" name="base_to_laser_broadcaster" args="0 0 0 0 0 base_link laser 100" />
53  <!--<node pkg="tf" type="static_transform_publisher" name="map_nav_broadcaster" args="0 0 0 0 0 map nav 100"/>-->
54 </launch>
```

<FIG.39 Launch 파일에서 Map Size 변경>

- 위와 같이 수정한 후 다시 Hector Slam을 구동했더니 알맞은 Map을 얻을 수 있었다.



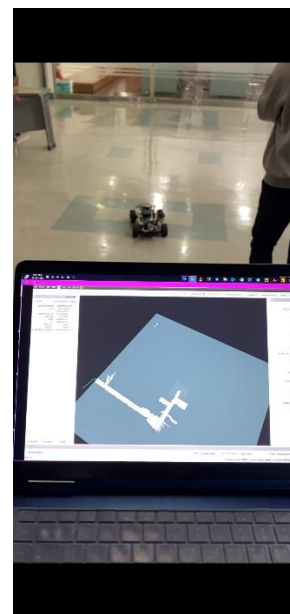
<FIG.40 인하대학교 하이테크 6층 Map>



<FIG.41 처음 구동시 Mapsize로 인한 오류>



<FIG.42 F1/10 구동 모습>



<FIG.43 F1/10 Slam 모습>