작성일	2025.03.17
작성부서	SW개발팀
작성자	조환영

A1로봇 – sensor_interface

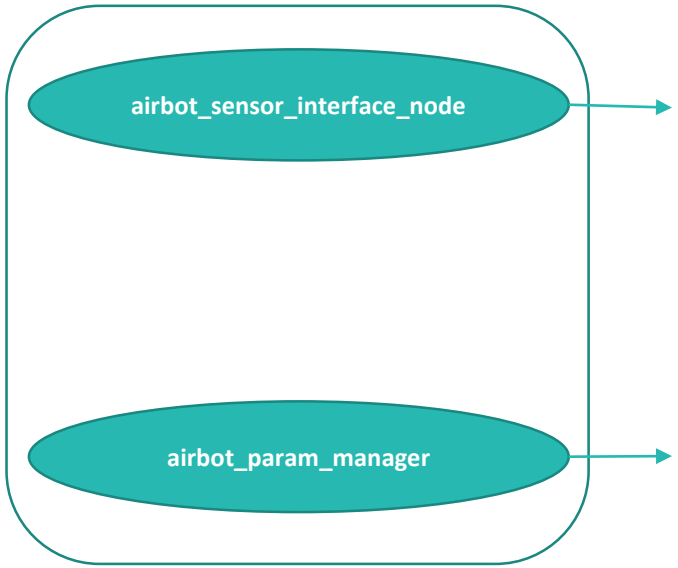
SW개발팀_조환영 연구원
2025.03.17

everybot.

CONTENTS.

1. “airbot_sensor_interface” Package

airbot_sensor_interface



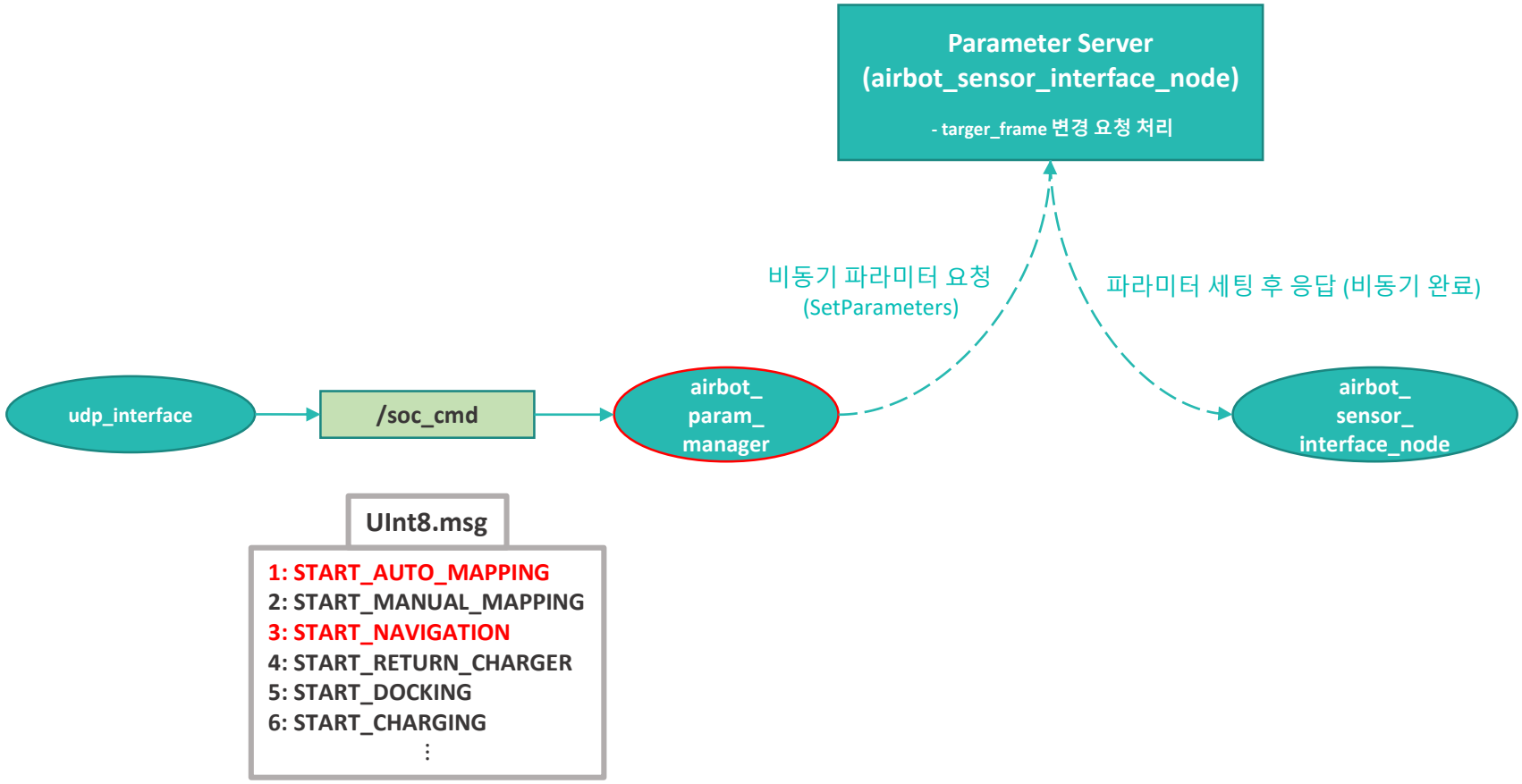
센서 데이터 기반 주행 알고리즘에 사용되는 전처리 데이터를 발행하는 노드

1. PointCloud2 형태로 변환한 데이터
2. 카메라 사물인식 bounding box 데이터
3. 멀티존 ToF 3, 4행 관련 데이터

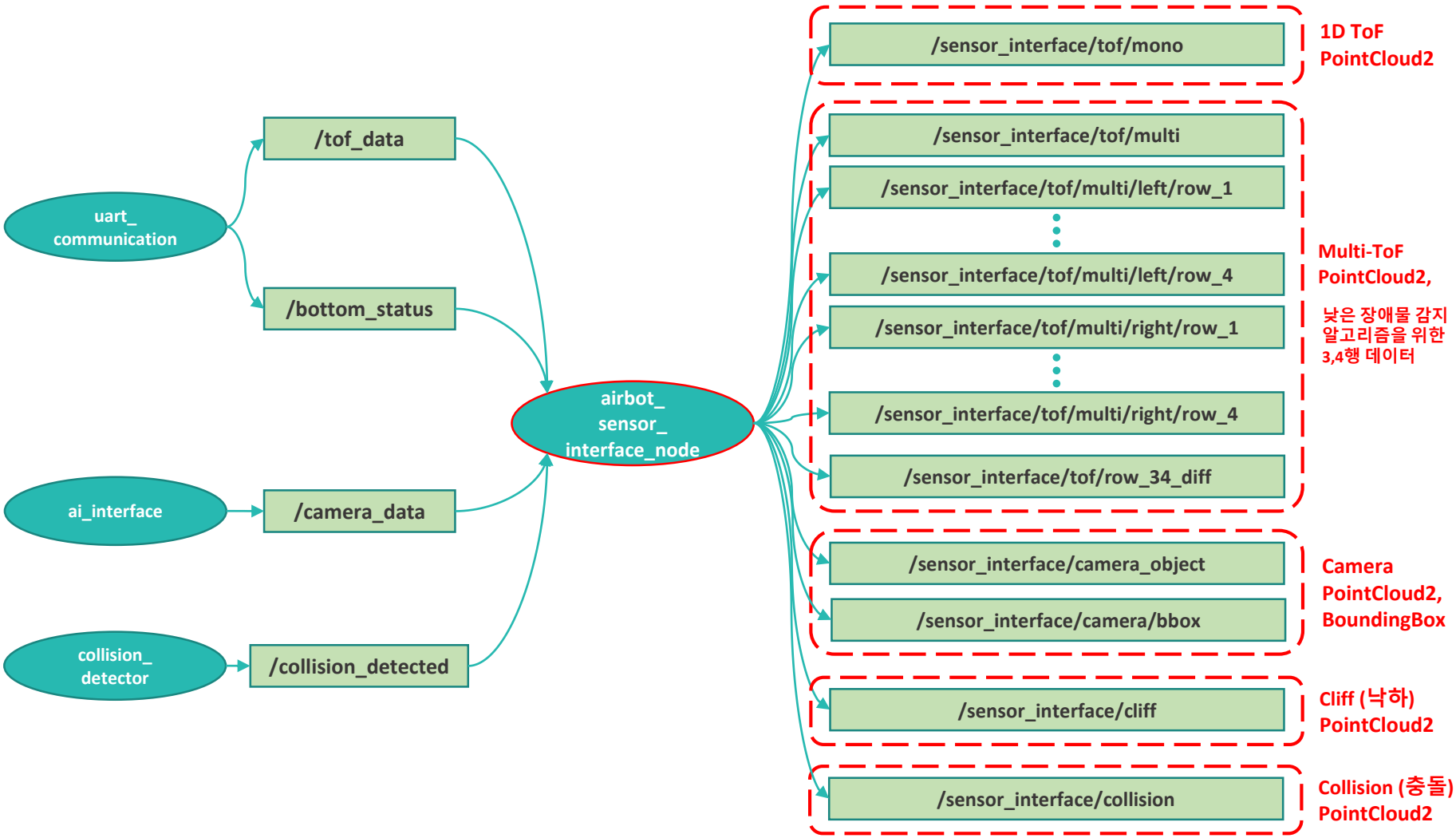
airbot_sensor_interface_node의 파라미터를 프로세스 런타임에 동적으로 변경하는 노드

(주로 pointcloud의 좌표계를 변환하는 용도로 사용)

1-1. “airbot_param_manager” Node



1-2. “airbot_sensor_interface_node” Node



1-3. “airbot_sensor_interface_node” Parameter

sensor_interface_node 관련 파라미터 (사용자 수정 가능)

sensor_interface_param.yaml 파일에서 수정 가능
(파일 위치: **~/airbot_ws/install/airbot_sensor_interface/share/airbot_sensor_interface/config/**)

```yaml
airbot_sensor_interface_node:
  ros__parameters:

    target_frame: "map" # "map" or "base_link"

    tof:
      all:
        use: true
      1D:
        use: true
        publish_rate_ms: 10
        tilting_angle_deg: 45.0 # double type
      multi:
        publish_rate_ms: 50
        left:
          use: true
        right:
          use: true
        row:
          use: true
          publish_rate_ms: 50
```

```yaml
camera:
  use: true
  publish_rate_ms: 100
  pointcloud_resolution: 0.05
  class_id_confidence_th: # 형식: "class id: confidence score"
    - "0: 50" #cable
    - "1: 75" #carpet
    - "2: 60" #clothes
    # - "5: 80" #obstacle 미사용
    - "6: 30" #poop
    # - "8: 80" #threshold 미사용
  object_direction: false  # 정방향(CCW+):True, 역방향(CW+):False
  logger:
    use: true
    margin:
      distance_diff: 0.5
      width_diff: 0.1
      height_diff: 0.1
```

```yaml
cliff:
  use: true
  publish_rate_ms: 10

collision:
  use: true
  publish_rate_ms: 10

use_sim_time: false
```

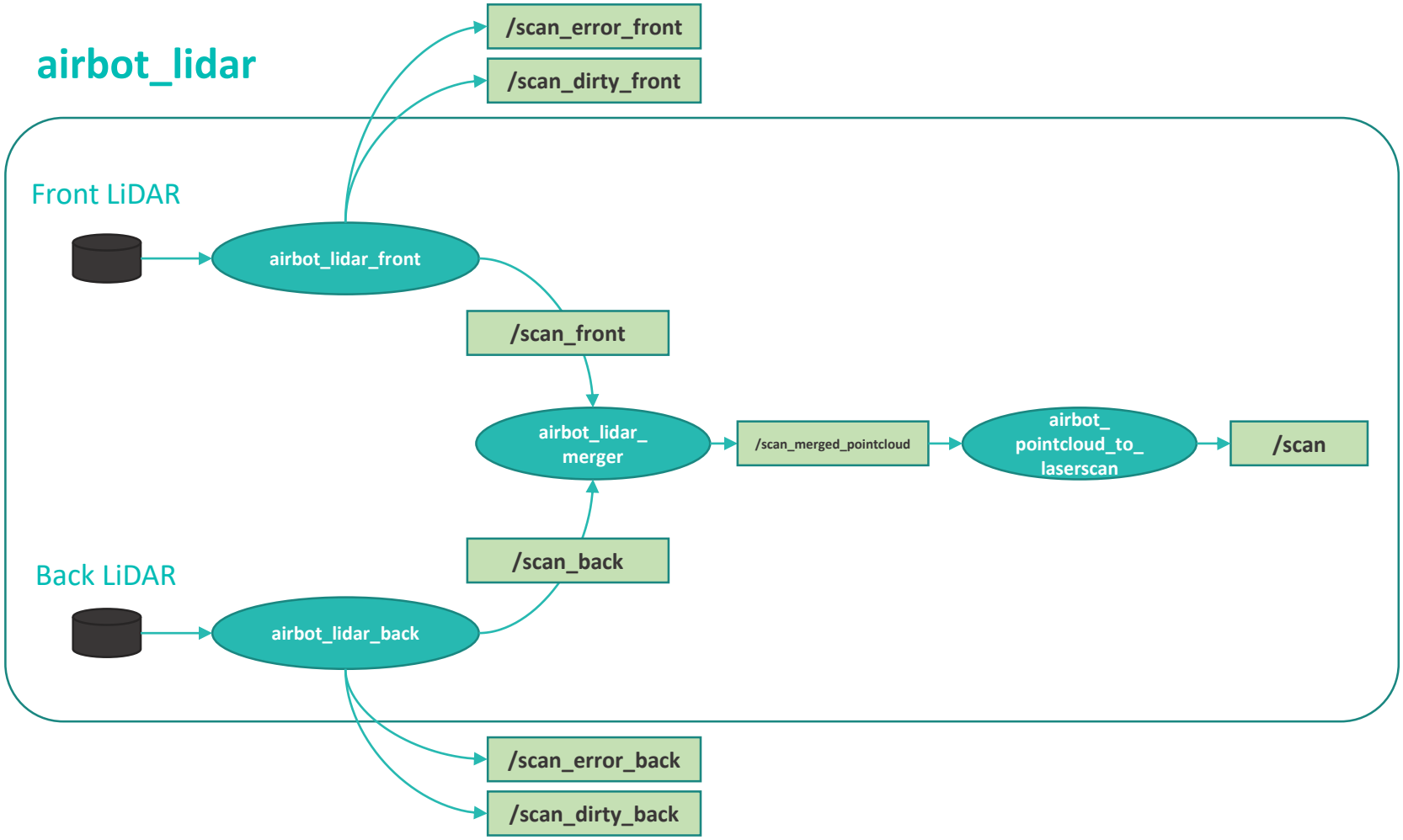
1-4. “airbot_sensor_interface” Node – Parameter

“**센서 사양 및 기구적 제원**”에 관한 파라미터 (사용자 수정 불가)

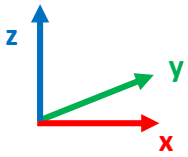
sensor_interface_node.cpp 파일에서 정의

```
// Robot, Sensor Geometric Specification
double tof_top_sensor_frame_x_translate = 0.0942;       //[meter]
double tof_top_sensor_frame_y_translate = 0.0;          //[meter]
double tof_top_sensor_frame_z_translate = 0.56513;      //[meter]
double tof_bot_sensor_frame_x_translate = 0.14316;      //[meter]
double tof_bot_sensor_frame_y_translate = 0.075446;     //[meter]
double tof_bot_sensor_frame_z_translate = 0.03;         //[meter]
double tof_bot_left_sensor_frame_pitch_ang = -2.0;      //[deg]
double tof_bot_right_sensor_frame_pitch_ang = -2.0;     //[deg]
double tof_bot_left_sensor_frame_yaw_ang = 13.0;        //[deg]
double tof_bot_rihgt_sensor_frame_yaw_ang = -15.0;      //[deg]
double tof_bot_fov_ang = 45;                            //[deg]
double camera_sensor_frame_x_translate = 0.15473;       //[meter]
double camera_sensor_frame_y_translate = 0.0;           //[meter]
double camera_sensor_frame_z_translate = 0.5331;        //[meter]
double cliff_sensor_distance_center_to_front_ir = 0.15; //[meter]
double cliff_sensor_angle_to_next_ir_sensor = 50;       //[deg]
double collision_forward_point_offset = 0.25;           //[meter]
```

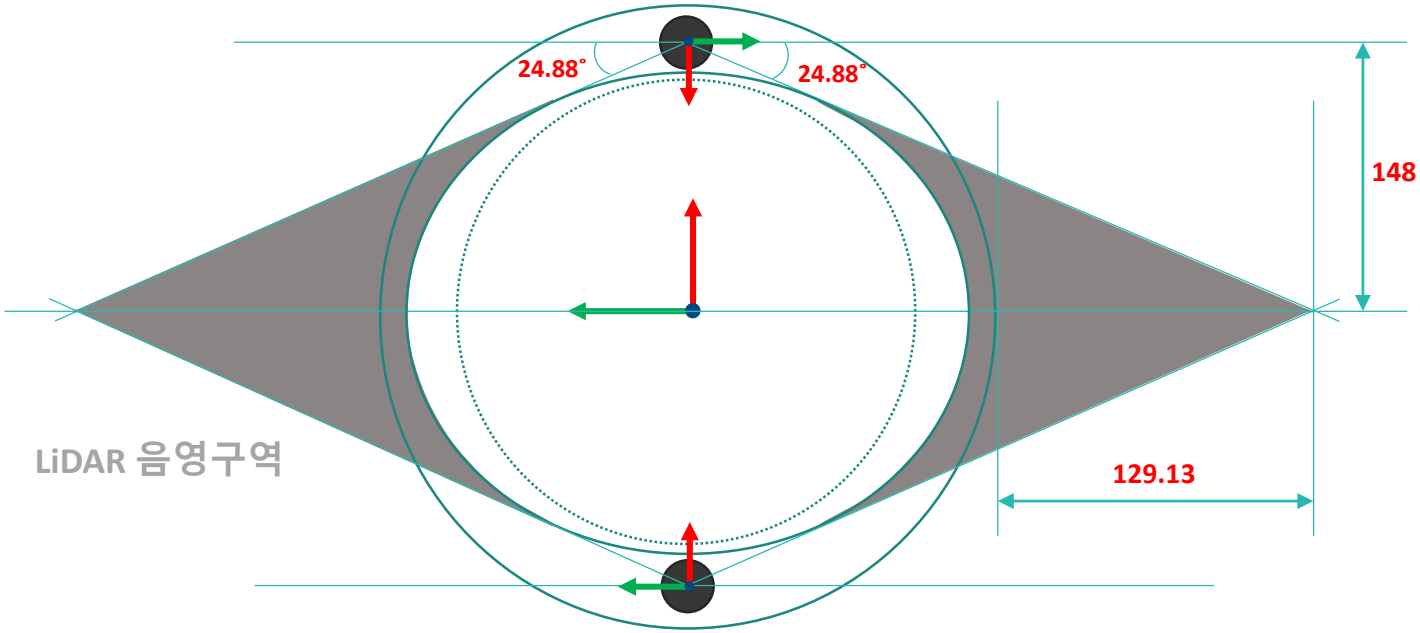
2. “airbot_lidar” Package



2-1. Lidar - FoV

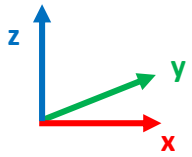


, Unit: [mm], [deg]

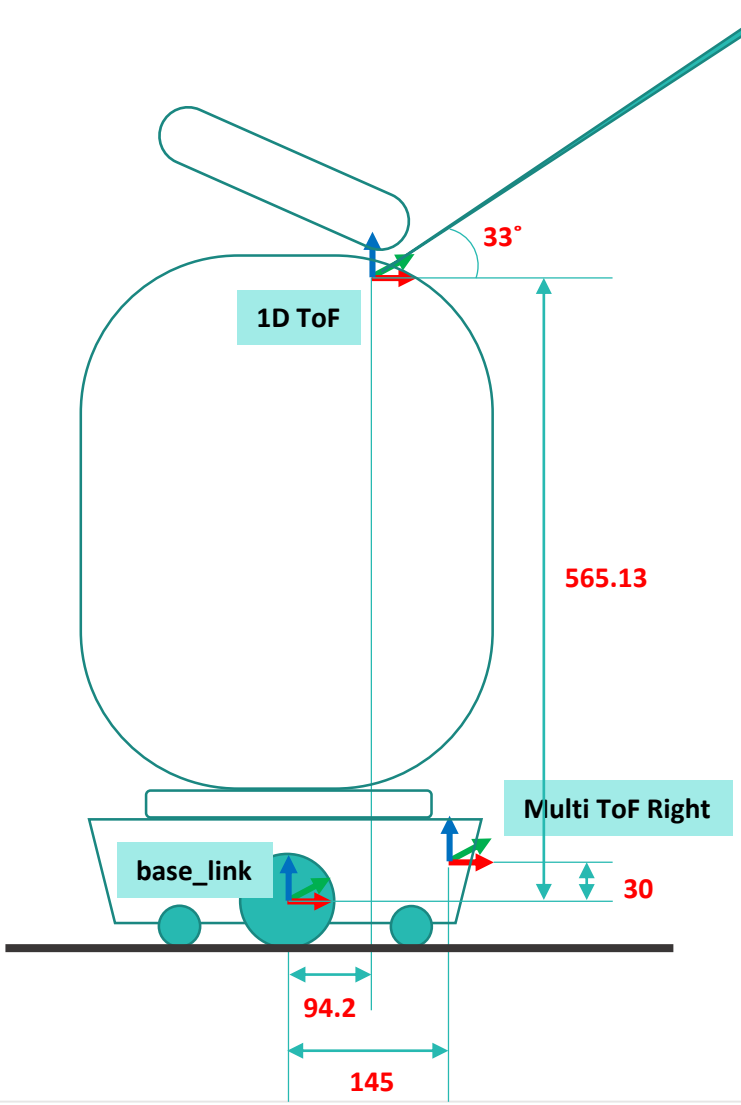


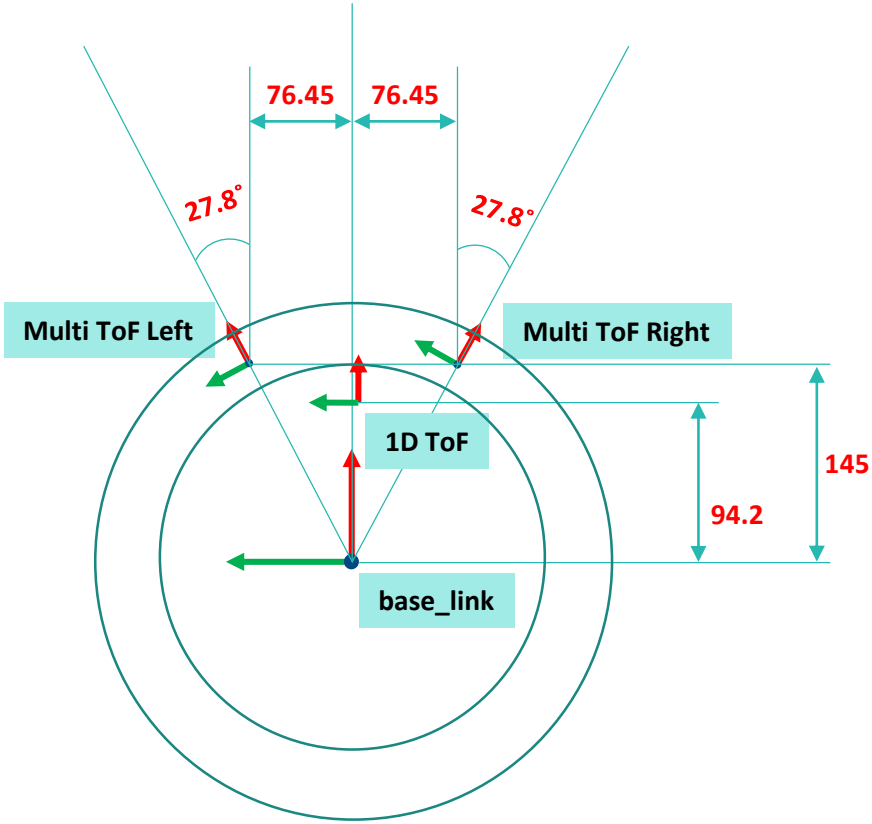
LiDAR 각도 범위 [60,300] deg로 설정

3. Sensor TF Geometry - ToF

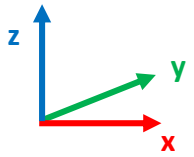


, Unit: [mm], [deg]

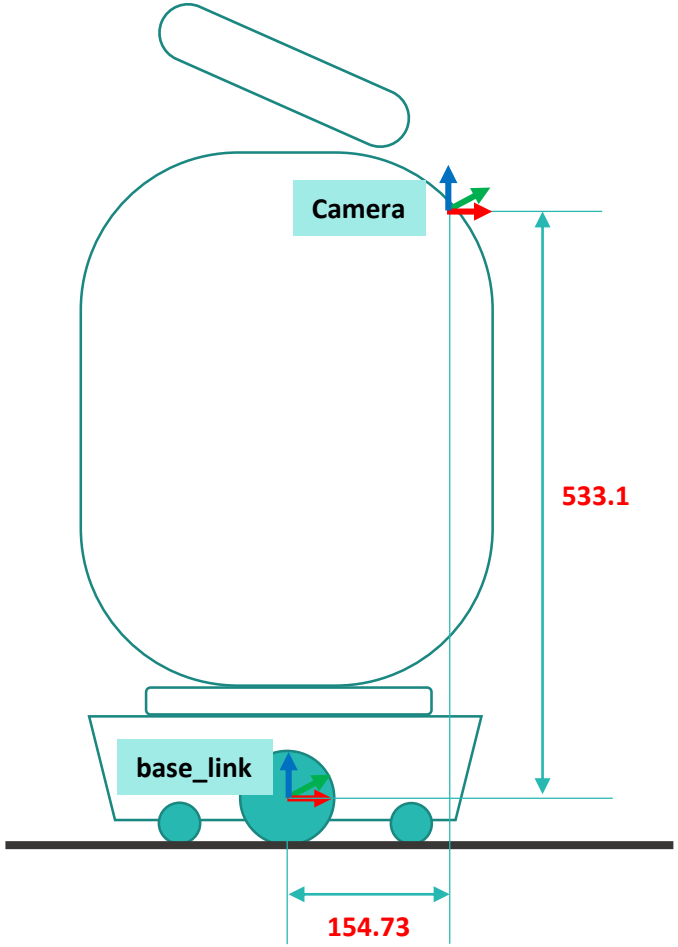


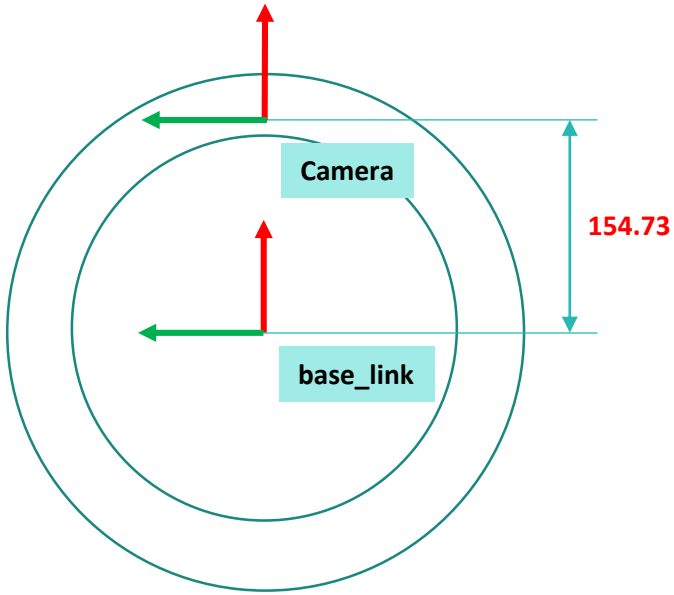


3. Sensor TF Geometry - Camera

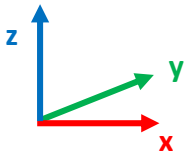


, Unit: [mm], [deg]



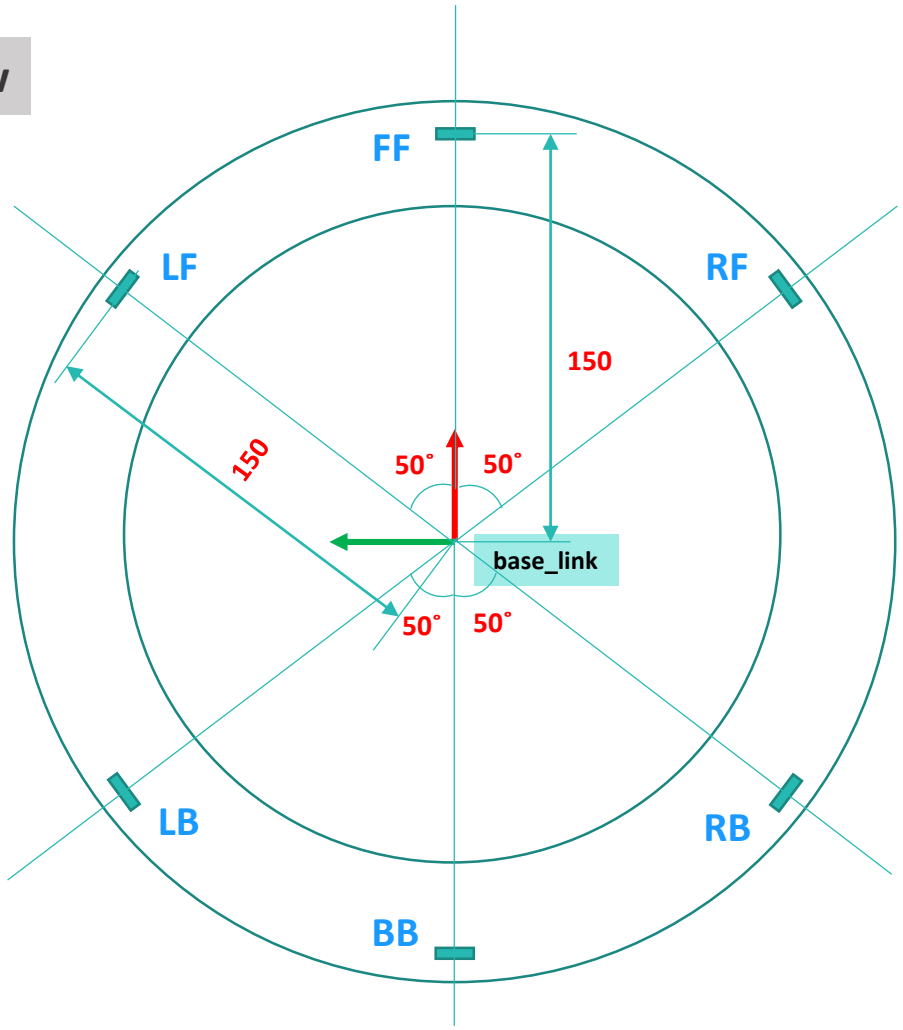


3. Sensor TF Geometry - Cliff



, Unit: [mm], [deg]

Top View



4. Sensor Data Type - ToF

Raw Data

/tof_data

PointCloud2 Data

/sensor_interface/tof/mono

/sensor_interface/tof/multi

실제 sensor->MCU 로 들어올 때 데이터 배열

3	2	1	0
7	6	5	4
11	10	9	8
15	14	13	12

LEFT

12	13	14	15
8	9	10	11
4	5	6	7
0	1	2	3

RIGHT

숫자의 의미는 배열의 index (데이터 순서)입니다.

AP에서 re-mapping하여 주행에 사용하는 데이터 배열

Re-Mapping

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

/tof_data
bot_left, bot_right

"ros2 topic echo /tof_data" 로 보는 데이터입니다.

TofData.msg

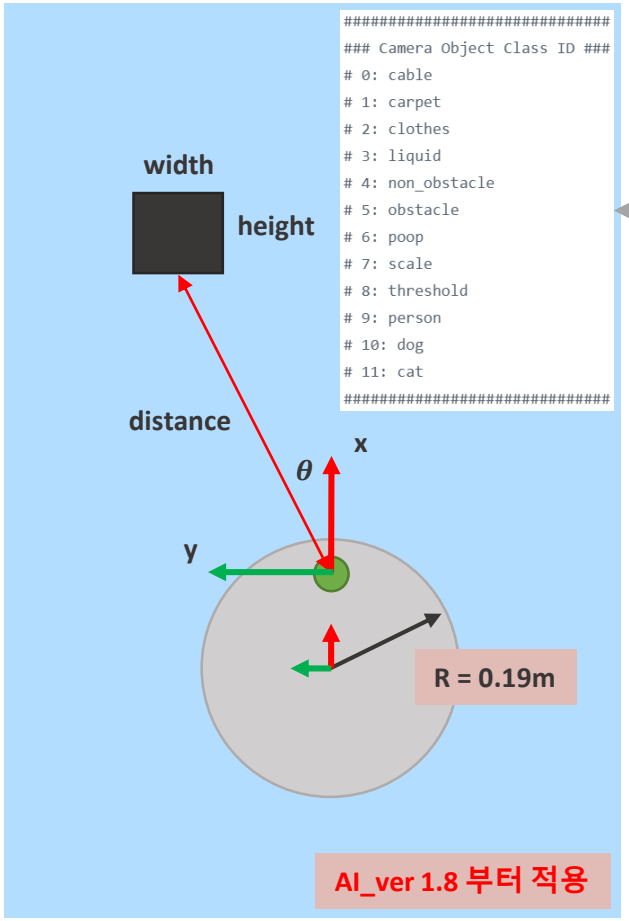
```
builtin_interfaces/Time timestamp

float64 top
float64[16] bot_left
float64[16] bot_right

uint8 top_status
uint8 bot_status

float64 robot_x
float64 robot_y
float64 robot_angle
```

4. Sensor Data Type - Camera



Raw Data

/camera_data

PointCloud2 Data

/sensor_interface/camera_object

AIData.msg

```
uint8 id
uint8 score

~~float64 x~~
~~float64 y~~ (무시)
float64 theta
float64 width
float64 height
float64 distance
```

AIDataArray.msg

```
builtin_interfaces/Time timestamp

uint8 num

AIData[] data_array

float64 robot_x
float64 robot_y
float64 robot_angle
```

4. Sensor Data Type - Cliff

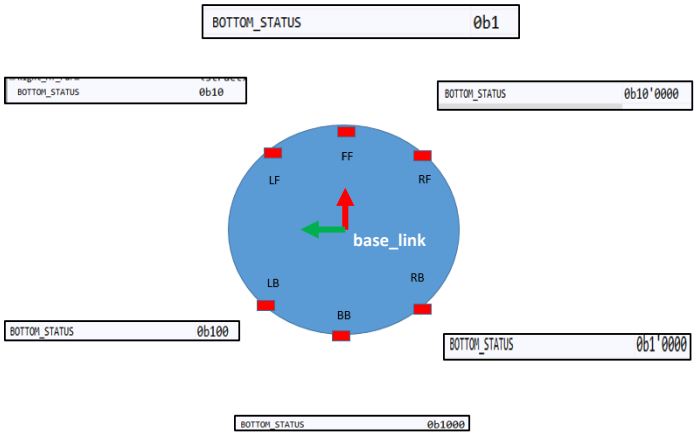
Raw Data

/bottom_status

/bottom_ir_data

PointCloud2 Data

/sensor_interface/cliff



std_msgs/UInt8.msg

uint8 data

BottomIrData.msg

builtin_interfaces/Time timestamp

bool ff
bool fl
bool fr
bool bb
bool bl
bool br

float64 robot_x
float64 robot_y
float64 robot_angle

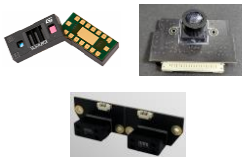
FF	1	1	0x01
LF	10	2	0x02
LB	100	4	0x04
BB	1000	8	0x08
RB	10000	16	0x10
RF	100000	32	0x20

예시) FF, LF, RF가 감지되었을 때

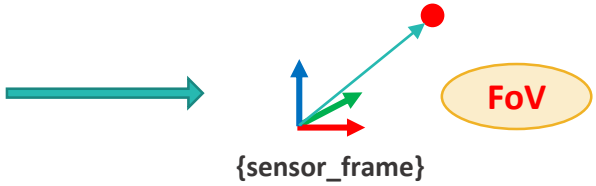
FF,LF,RF 100011 35 0x23

“ros2 topic echo /bottom_status” 로 보는 데이터입니다.

5. Converting Sensor Data to PointCloud2 in Map Frame

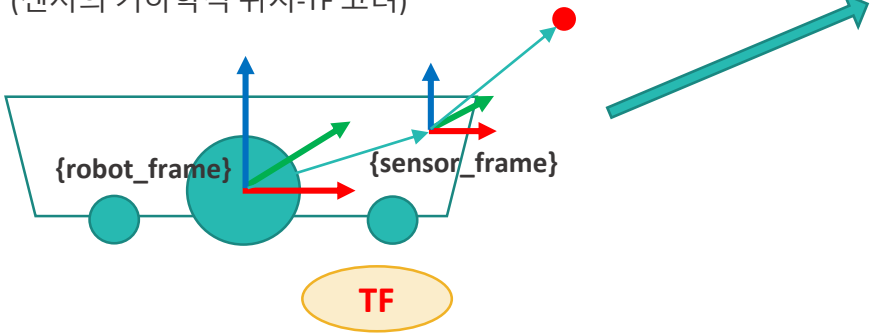


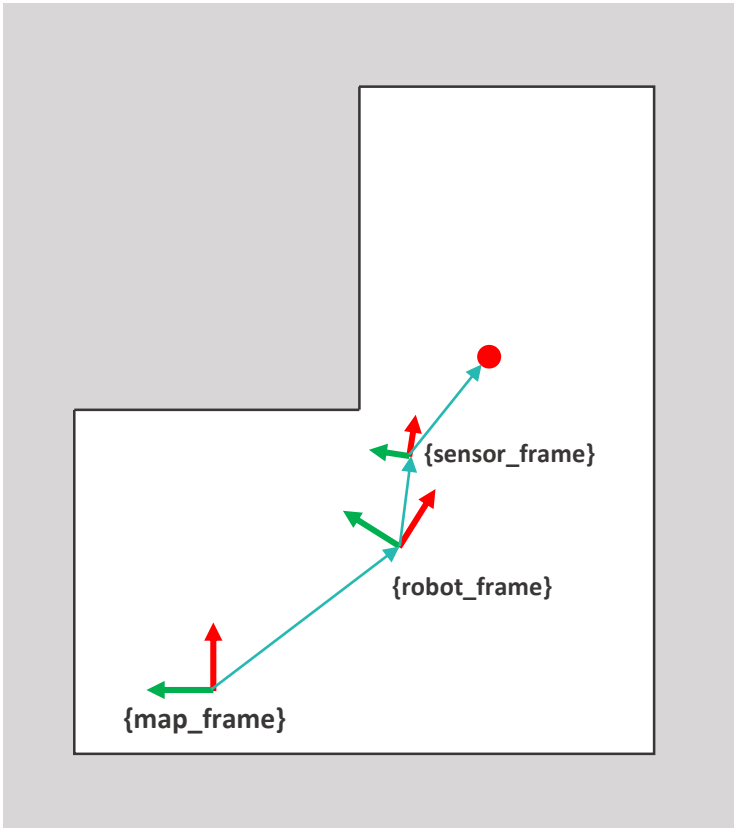
센서 감지



감지된 데이터를
센서 좌표계 기준으로 표현
(센서 FoV 고려)

센서 좌표계의 데이터를
로봇 좌표계 기준 데이터로 변환
(센서의 기하학적 위치-TF 고려)

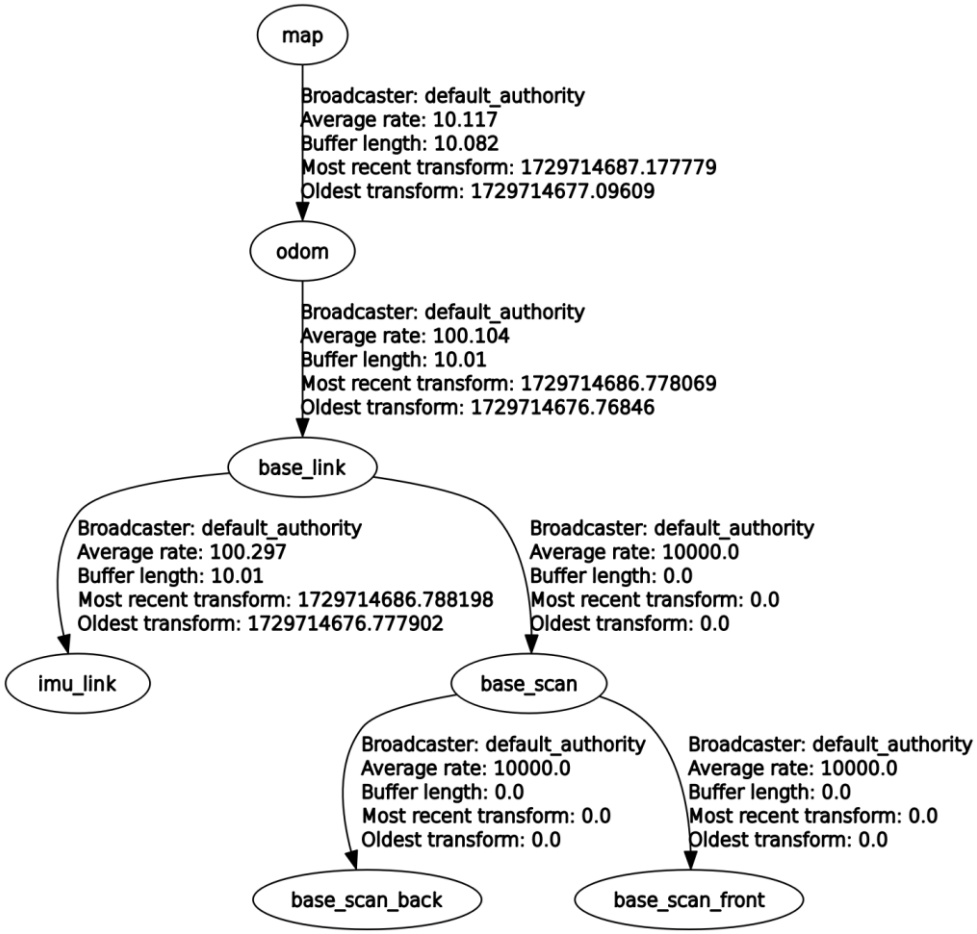




최종 글로벌 좌표계(map frame) 기준으로 변환 후
PointCloud2 형태 데이터로 발행
(현재 로봇 위치-amcl_pose 고려)

amcl_pose

6. Robot TF Tree



7. 참고

- sensor_msgs/PointCloud2 메시지 타입

PointCloud2.msg

```
std_msgs/Header header
uint32 height
uint32 width
sensor_msgs/PointField[] fields
bool is_bigendian
uint32 point_step
uint32 row_step
uint8[] data
bool is_dense
```

height, width: PointCloud2의 크기 결정
(1차원이면 height=1 / 2차원이면 height가 2 이상)

field: pointcloud의 데이터를 해석하는 방법을 알려주는 설명서
(point들의 데이터 타입, 크기, 이름등을 결정)
(ros의 다른 노드들이 해당 토픽을 해석할 때 필요한 format)

bigendian: 빅엔디안인지, 리틀엔디안인지 명시 (True : 빅, False: 리틀)

point_step: 각 포인트가 차지하는 bytes (데이터 타입 bytes * PointField 개수)

row_step: 한 열의 bytes (point_step * width)

data: 실제 바이너리 데이터를 저장하는 곳

is_dense: 모든 포인트가 유효한지 (NaN or Inf가 없는지) (안전하게 false)

End of Document.

everybot.