| | |
|---|---|
| 작성일 | 2025.02.27 |
| 작성부서 | SW개발팀 |
| 작성자 | 조환영 |

# A1로봇 센서 데이터 및 TF

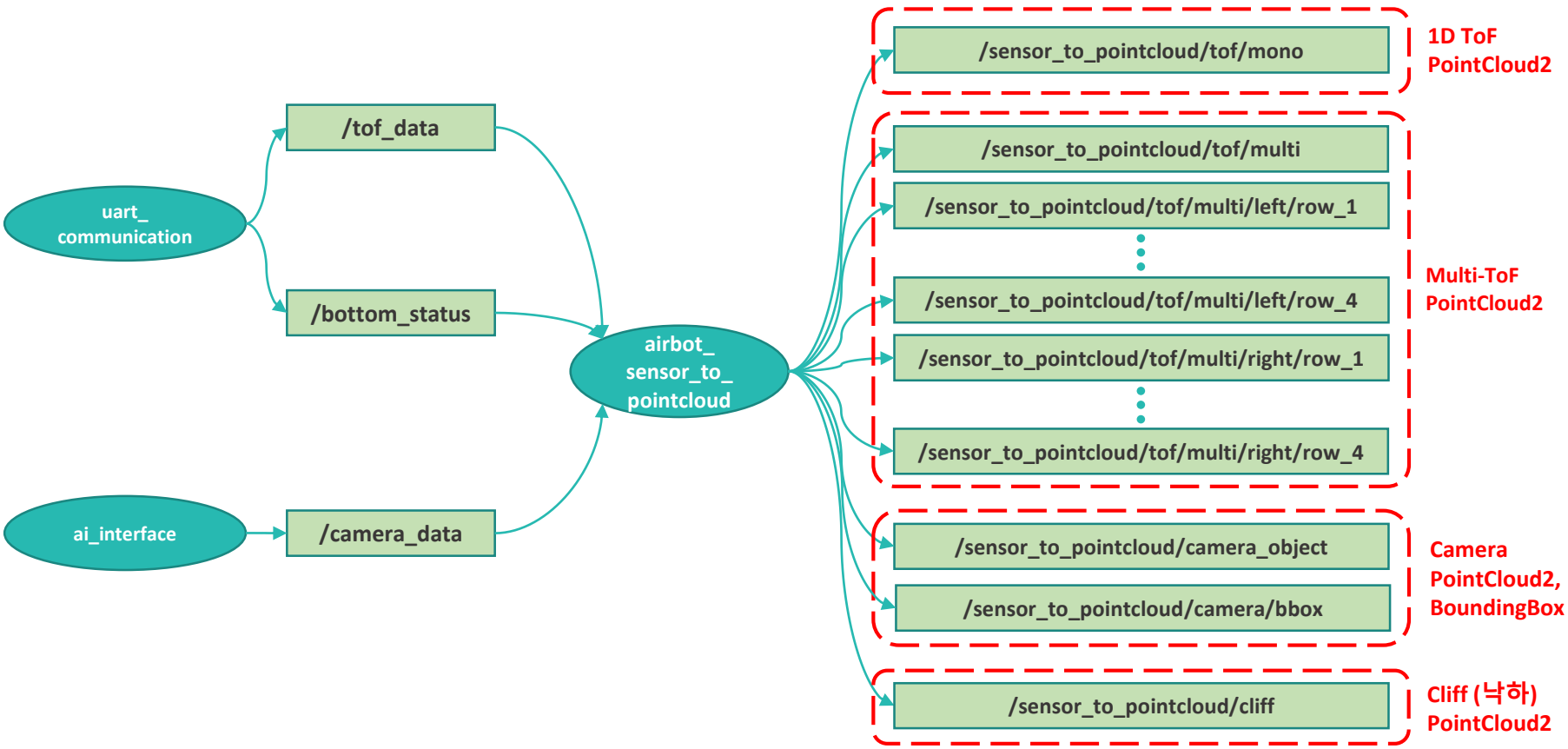
SW개발팀_조환영 연구원
2025.02.27

everybot.

# CONTENTS.

# 1. sensor_to_pointcloud Node – Topic Graph

"base_link", "map" 좌표계 선택 가능
(파라미터로 선택 가능)



# 1. sensor_to_pointcloud Node – Parameter

sensor_to_pointcloud Node의 “**사용 방법**”에 관한 파라미터

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

```yaml
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


  cliff:
    use: true
    publish_rate_ms: 10
```

```yaml
camera:
  use: true
  publish_rate_ms: 100
  pointcloud_resolution: 0.05
  class_id_confidence_th: # 형식: "class id: confidence score"
    - "2: 55"
    - "5: 55"
    - "6: 55"
  object_direction: true  # 정방향(CCW+):True, 역방향(CW+):False
  logger:
    use: true
    margin:
      distance_diff: 0.5
      width_diff: 0.1
      height_diff: 0.1
```

```yaml
lidar:
  use: false
  publish_rate_ms: 100
  front:
    range:
      angle_max: 270.0
      angle_min: 90.0
    geometry:
      alpha: 180.0
      offset:
        x: 0.15
        y: 0.0
        z: 0.0
  back:
    range:
      angle_max: 270.0
      angle_min: 90.0
    geometry:
      alpha: 0.0
      offset:
        x: -0.15
        y: 0.0
        z: 0.0
```

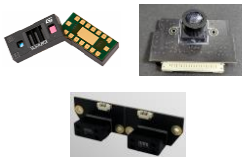
# 1. sensor_to_pointcloud Node – Parameter

"**센서 사양 및 기구적 제원**"에 관한 파라미터

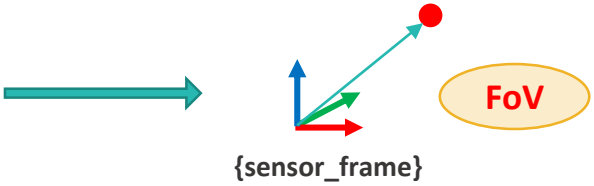
**sensor_to_pointcloud.cpp** 파일에서 수정 가능 (웬만하면 안바뀜)

```
// 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]
```

# 2. Converting Sensor Data to PointCloud2 in Map Frame

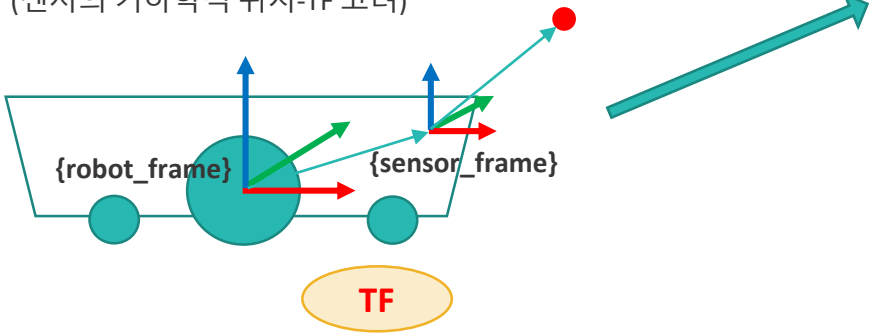


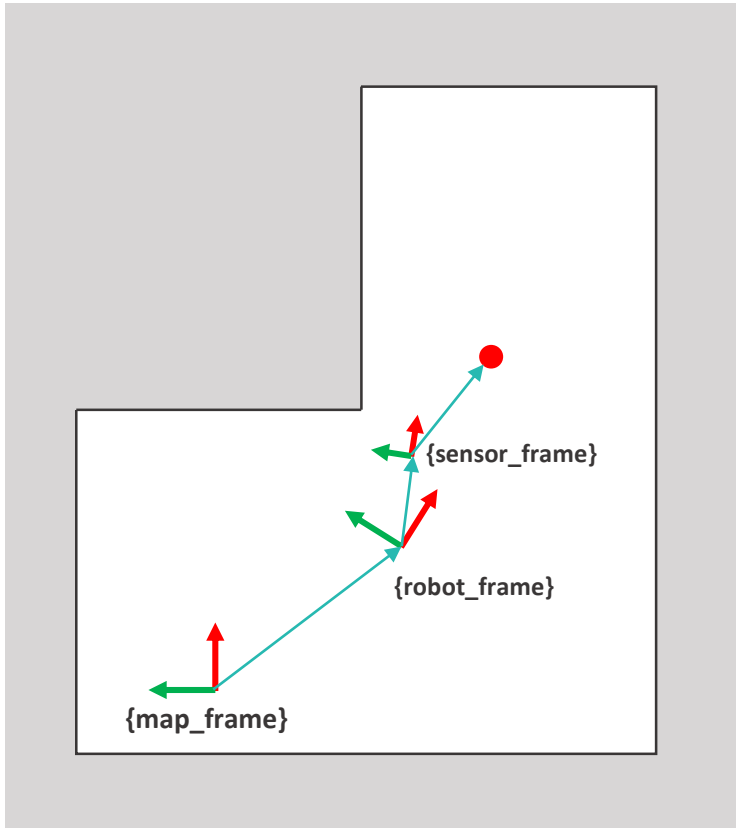
센서 감지




감지된 데이터를
센서 좌표계 기준으로 표현
(센서 FoV 고려)

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

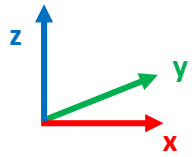


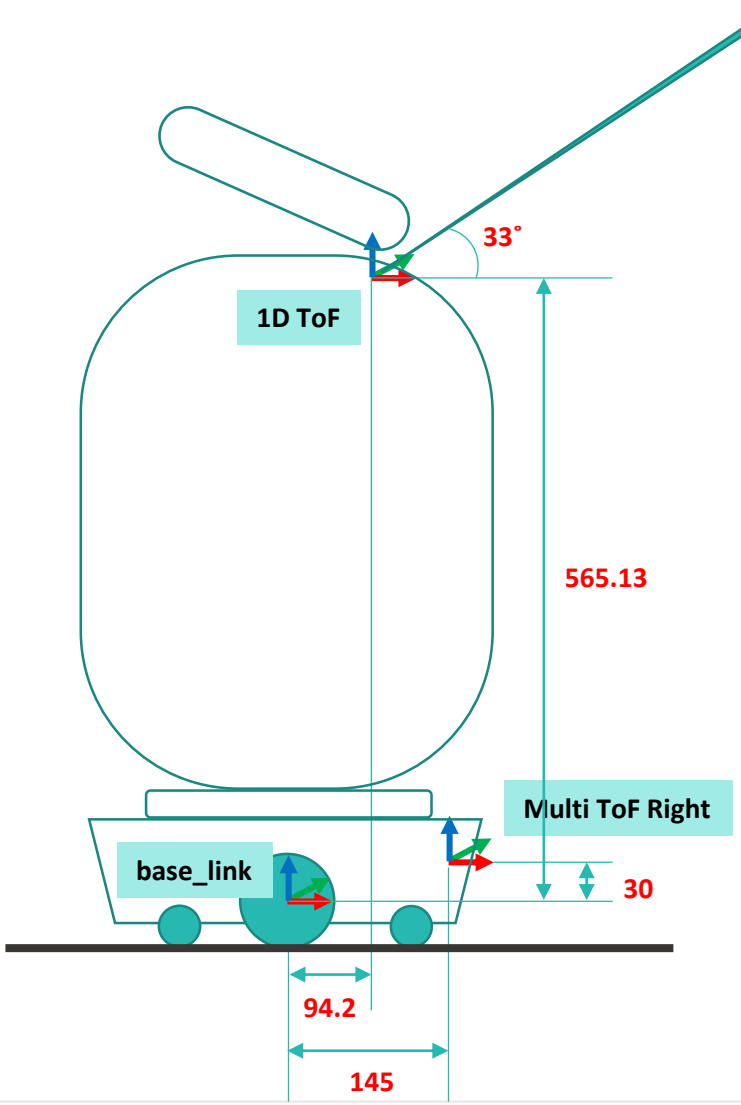

최종 글로벌 좌표계(map frame) 기준으로 변환 후
PointCloud2 형태 데이터로 발행
(현재 로봇 위치-amcl_pose 고려)

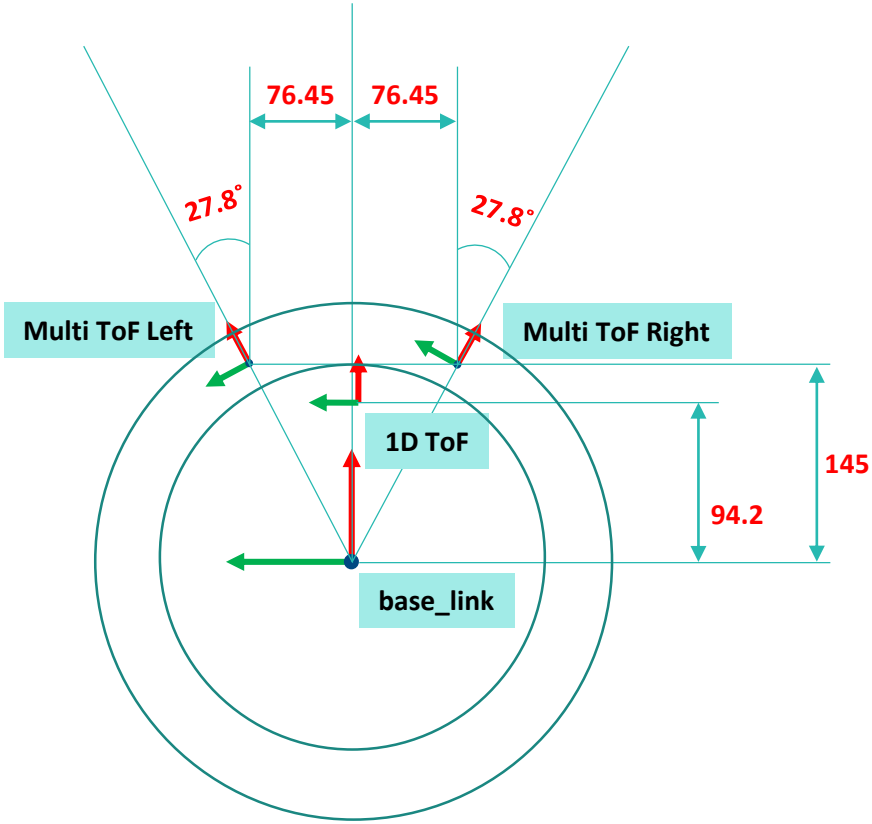
amcl_pose

# 3. ToF TF




, Unit: [mm], [deg]



# 3. ToF Data

**Raw Data**

/tof_data

**PointCloud2 Data**

/sensor_to_pointcloud/tof/mono

/sensor_to_pointcloud/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 |

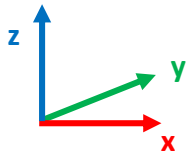
**/tof_data**
bot_left, bot_right

**"ros2 topic echo /tof_data" 로 보는 데이터입니다.**

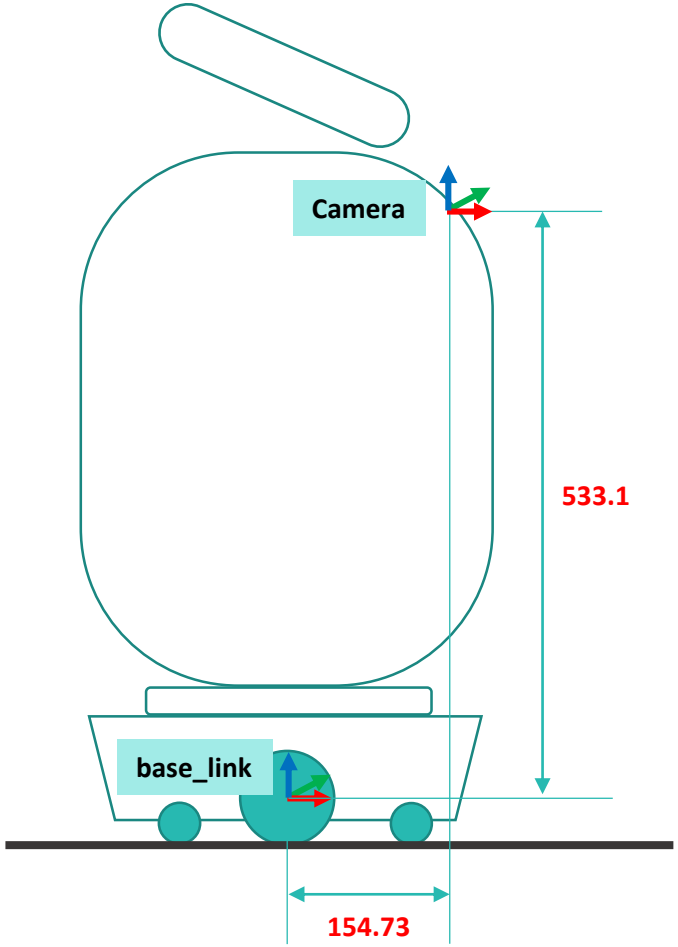
**TofData.msg**

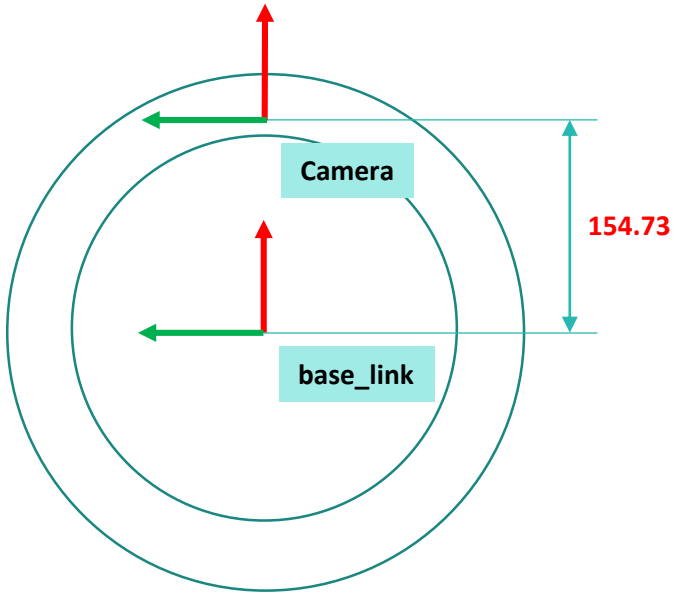
```
builtin_interfaces/Time timestamp

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. Camera TF

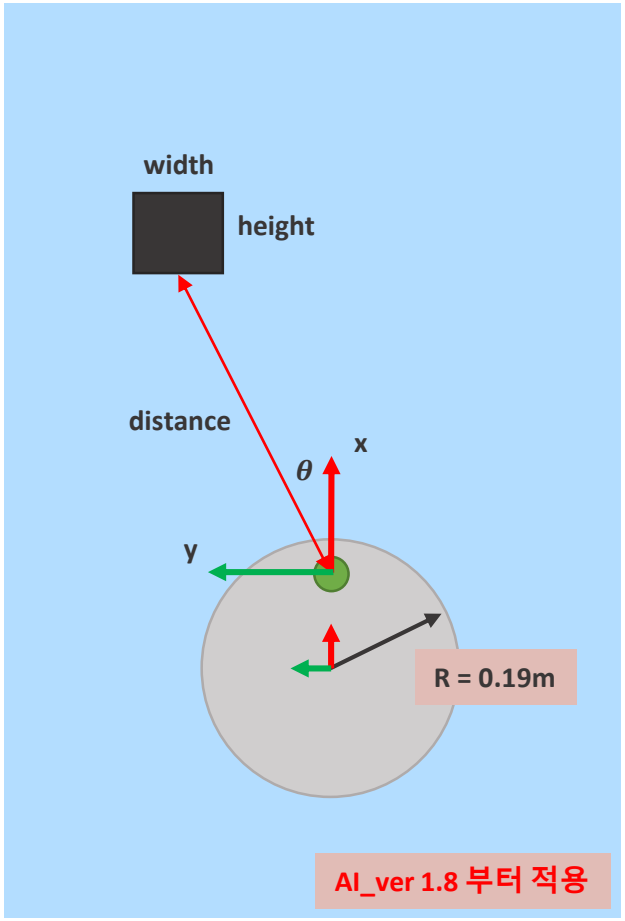



, Unit: [mm], [deg]



# 4. Camera Data




**Raw Data**

`/camera_data`

**AIData.msg**

```
uint8 id
uint8 score

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

```
##############################
### Camera Object Class ID ###
# 0: cable
# 1: carpet
# 2: clothes
# 3: liquid
# 4: non_obstacle
# 5: obstacle
# 6: poop
# 7: scale
# 8: threshold
# 9: person
# 10: dog
# 11: cat
##############################
```

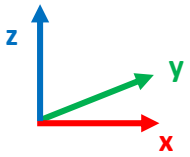
**PointCloud2 Data**

`/sensor_to_pointcloud/camera_object`

**AIDataArray.msg**

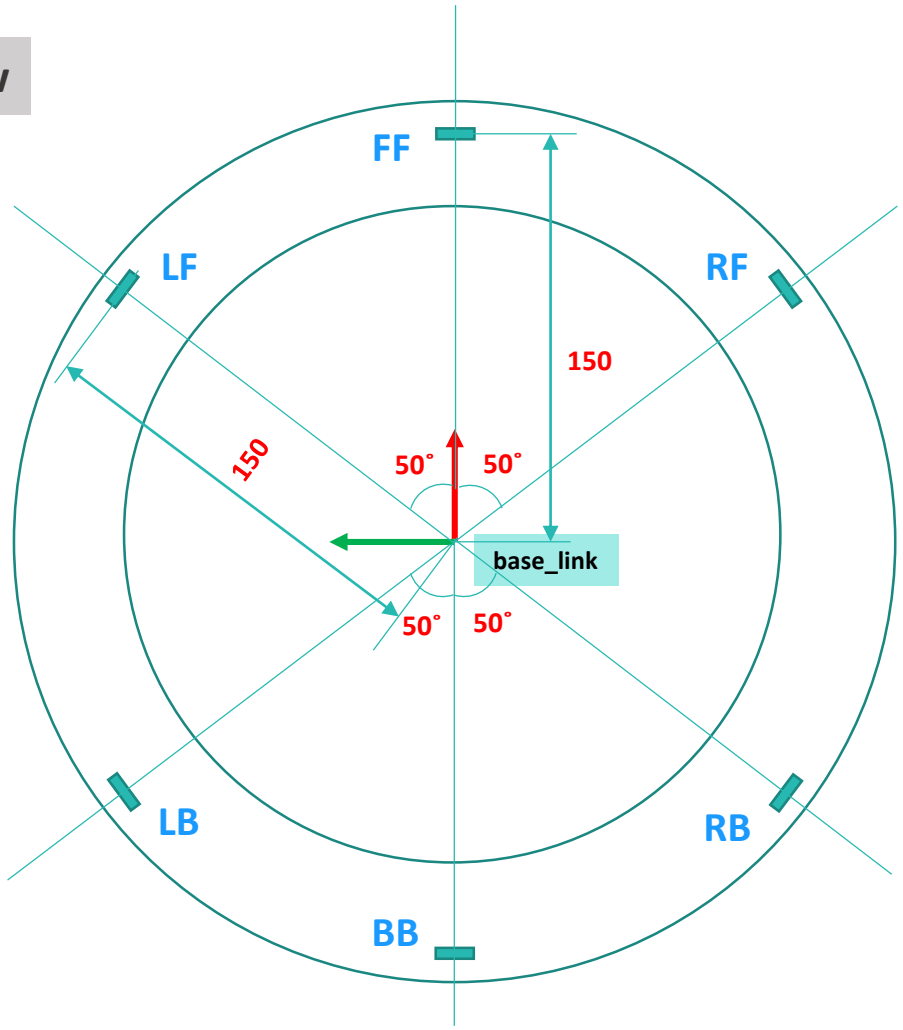
```
builtin_interfaces/Time timestamp

uint8 num

AIData[] data_array

float64 robot_x
float64 robot_y
float64 robot_angle
```

# 5. Cliff TF




, Unit: [mm], [deg]

Top View



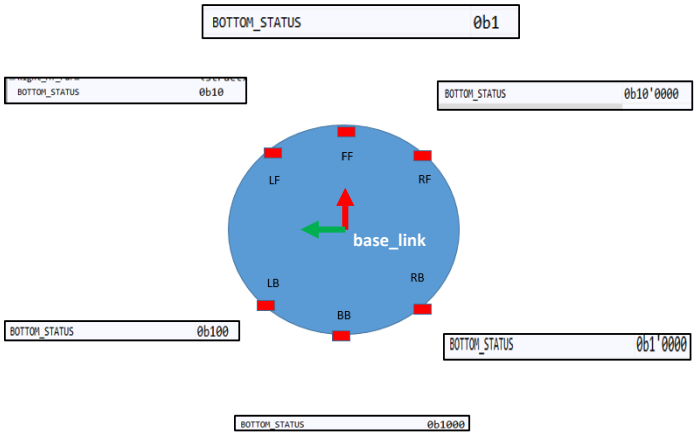
# 5. Cliff Data

**Raw Data**

**/bottom_status**

**PointCloud2 Data**

**/sensor_to_pointcloud/cliff**




**std_msgs/UInt8.msg**

**uint8 data**

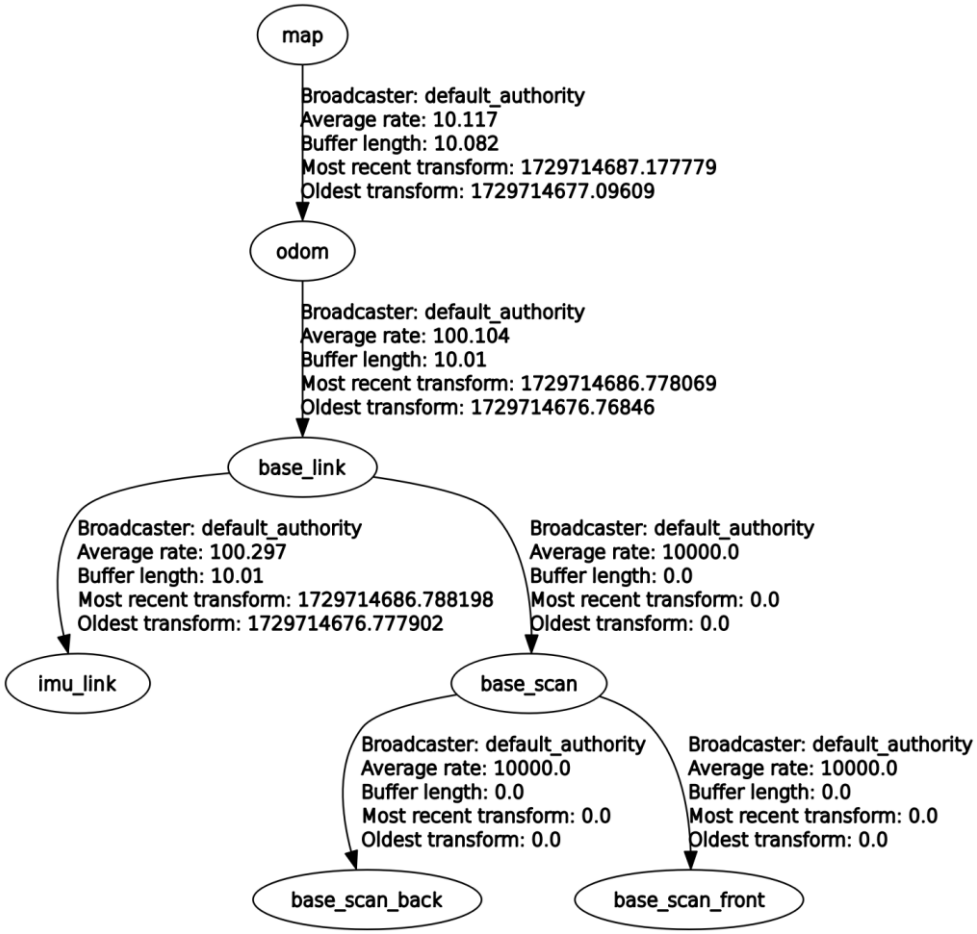
| | | | |
|---|---|---|---|
| FF | 1 | 1 | 0x01 |
| LF | 10 | 2 | 0x02 |
| LB | 100 | 4 | 0x04 |
| BB | 1000 | 8 | 0x08 |
| RB | 10000 | 16 | 0x10 |
| RF | 100000 | 32 | 0x20 |

예시) FF, LF, RF가 감지되었을 때

FF,LF,RF 100011 35 0x23

**“ros2 topic echo /bottom_status” 로 보는 데이터입니다.**

# 6. Robot TF Tree



# 7. 데이터(Topic) 확인 방법

모든 환경 세팅이 되어있다는 가정 하에
(ros2 humble, linux(ubuntu22.04 권장) 설치 및
~/airbot_ws/install/robot_custom_msgs 파일 확보 – 자세한 내용은 문의)

터미널 창에서 아래 명령어 실행

**export ROS_DOMAIN_ID = 30**

**source ~/airbot_ws/install/setup.bash**

**ros2 topic echo {토픽명}**

{토픽명}에 원하는 데이터의 토픽명을 입력한다. (e.g. ros2 topic echo /camera_data)
(이전 페이지들의 raw data 토픽명 확인)

{토픽명}은 터미널 창에 **ros2 topic list** 를 입력해서 확인할 수도 있다.

**/camera_data 토픽의 출력 예시**

```
---
timestamp:
  sec: 1729104105
  nanosec: 734337639
num: 3
data_array:
- id: 5
  score: 52
  x: 0.001
  y: 0.001
  theta: -1.0995574287564276
  width: 0.46
  height: 0.614
  distance: 0.771
- id: 5
  score: 45
  x: 0.001
  y: 0.001
  theta: -1.3089969389957472
  width: -0.232
  height: 0.326
  distance: 1.909
- id: 5
  score: 27
  x: 0.001
  y: 0.001
  theta: -1.5707963267948966
  width: 0.414
  height: 0.12
  distance: 0.027
robot_x: 0.0
robot_y: 0.0
robot_angle: 0.0
---
```

# 8. 참고

- Roll, Pitch, Yaw 데이터 확인 토픽 — /odom — odom_msg.pose.pose.orientation 에서 확인
- acc (Twist) 데이터 확인 토픽 — /imu — 현재 발행 막아 놓음, 추후 발행 예정
- 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.