ROS gazebo

HW1

```
  cmd_vel_pub_ = this->create_publisher<geometry_msgs::msg::Tw

  // Initialise subscribers
  scan_sub_ = this->create_subscription<sensor_msgs::msg::Las
    "scan", \
    rclcpp::SensorDataQoS(), \
    std::bind(
      &Turtlebot3Drive::scan_callback, \
      this, \
      std::placeholders::_1));
```

깃허브 클론한 패키지에서 다음과 같은퍼블리셔와 라이다 받아오는

“turtlebot3_drive.cpp”로 토픽 이름들을 확인하여 이 과제의 흐름을 이해했다.

```
  this->declare_parameter<double>("desired_distance_from_wall
  this->declare_parameter<double>("desired_distance_front", 1

  // 변수에 파라미터 값 저장
  this->get_parameter("desired_distance_from_wall", desired_d
  this->get_parameter("desired_distance_front", desired_dista

  cmd_vel_pub_ = this->create_publisher<geometry_msgs::msg::Tw

  scan_sub_ = this->create_subscription<sensor_msgs::msg::Las
    "scan", 10, std::bind(&TurtlebotTraceWall::scan_callback,
```

생성자
벽과의 거리, 정면과의 거리를 상수값으로 선언해 주었다.

이 상수값 거리들을 기준으로 벽에 가까이 있는지 없는지 를 판단하게 하였다.

‘cmd_vel’이라는 토픽 퍼블리셔 선언

‘scan’이라는 라이다 데이터 토픽을 서브스크라이버 scan_callback 함수로 전달

- 스캔 값 콜백 함수

```
void TurtlebotTraceWall::scan_callback(const sensor_msgs::msg
{
  int right_index = (msg->ranges.size() * 3) / 4;  // 오른쪽 (약
  int front_index = 0;                             // 정면 (약 0

  float right_distance = msg->ranges[right_index];
  float front_distance = msg->ranges[front_index];

  RCLCPP_INFO(this->get_logger(), "Right distance: %.2f, Fron

  geometry_msgs::msg::Twist cmd_vel;

  // 정면의 거리가 너무 가까운 경우 제자리에서 왼쪽으로 회전
  if (front_distance < desired_distance_front_)
  {
    cmd_vel.linear.x = 0.0;
    cmd_vel.angular.z = 0.07 * 2; // 왼쪽 회전
  }
  else
  {
    // 오른쪽 벽의 거리 값에 따라 TurtleBot 조정
    if (right_distance < desired_distance_from_wall_)  // 벽에
    {
      cmd_vel.linear.x = 0.1;
      cmd_vel.angular.z = 0.07; // 왼쪽으로 회전
    }
    else if (right_distance > desired_distance_from_wall_)  /
    {
      cmd_vel.linear.x = 0.1;
      cmd_vel.angular.z = -0.07; // 오른쪽으로 회전
    }
    else  // 벽과 적정 거리 유지
```

```
    {
      cmd_vel.linear.x = 0.1;
      cmd_vel.angular.z = 0.0; // 직진
    }
  }

  cmd_vel_pub_->publish(cmd_vel);
  RCLCPP_INFO(this->get_logger(), "Published cmd_vel: linear.
}
```

‘right_index’ : 라이다 오른쪽 값

‘front_index’ : 라이다 정면 값

‘right_distance’ : 오른쪽 거리값

‘front_distance’ : 정면 거리값

- 여기서 ‘msg → ranges’는 LaserScan 메시지의 멤버 변수로, 각도별로 측정된 거리 값들의 **배열**

  배열의 각 요소는 레이저가 특정 각도에서 측정한 거리 값을 나타냄

  예를 들어, msg→ranges[0]는 보통 **정면(0도)** 방향에서 측정된 거리 값

- 벽 따라가는 로직

  정면 거리 기준

  가까울 경우 : 제자리에서 왼쪽으로 회전(오른쪽 벽을 수월하게 볼 수 있도록

  정면에서 멀 경우 : 오른쪽 거리 기준

  오른쪽 벽에서 가까운 경우 : 왼쪽으로 회전하며 직진(즉 좌회전)

  오른쪽 벽에서 먼 경우 : 오른쪽으로(벽 쪽으로) 회전하며 직진(우회전)