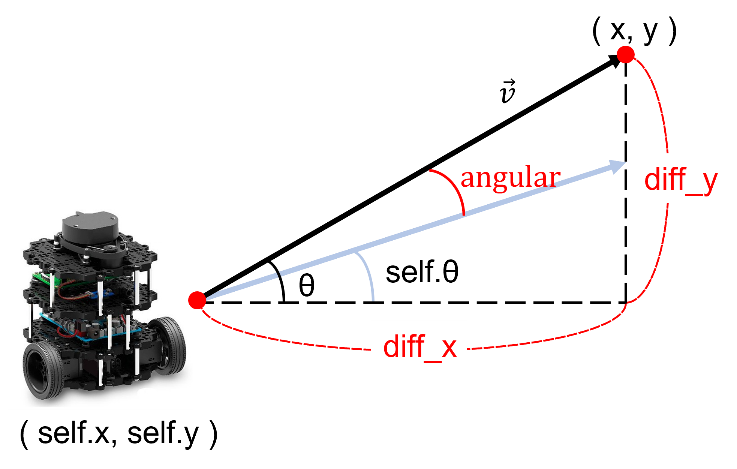
(3) Waypoint 기능 구현

Waypoint란 특정 point를 따라 이동하는 것을 의미한다. 도로의 차선과 축구장, 야구장과 같은 원하는 map을 그리기 위해선 정교한 waypoint를 수행할 수 있어야 한다. 터틀봇3에서는 Odometry 즉, 로봇의 위치 정보를 subscriber를 통해 받아올 수 있으며, cmd\_vel 즉, 선속도와 각속도 정보를 publisher를 통해 로봇에게 줄 수 있다. 이 두가지를 가지고 터틀봇3의 waypoint 기능을 구현하였다.

앞서 언급하였듯이 Odometry를 통해 터틀봇3의 x, y, θ 값을 받아들일 수 있다. 이때 각각의 값을 self.x, self.y, self.θ라고 하자. 도착 지점의 좌표는 명령을 통해 넣어주는 값으로 x, y으로 정의하였다.

이를 활용하면 (1)의 식과 같이 터틀봇3와 목표점까지의 벡터 를 구할 수 있으며, 터틀봇3와 목표점까지의 거리를 나눠주면 (2)의 식과 같이 크기가 1인 단위벡터 를 구할 수 있다. (1), (2) 식을 활용하여 이후 터틀봇3를 목표점까지 일직선으로 이동시킬 수 있다.

터틀봇3를 안정적으로 이동시키기 위해선 먼저 self.θ 값 즉, 로봇이 바라보는 방향과 목표점까지의 θ값을 일치시켜야 한다. self.θ와 θ가 일치한다는 것은 angular가 0이 되도록 만든다는 것이다. 이를 위해선 cmd\_vel으로 각속도를 조절해야 한다. 각속도는 (3) 식에서 볼 수 있듯 θ - self.θ를 계속 업데이트하는 방식으로 조절할 수 있다. 이후 PID 제어기를 추가하여 각속도의 성능과 안정성을 향상시킬 수 있다.

angular 값을 0으로 맞춰주었다면 터틀봇3를 바라보고 있는 방향 즉, x축 성분의 선속도 명령만을 주어 목표점으로 이동시킬 수 있다. 그러나 이 과정에서 내부와 외부 요인으로 로봇이 일정한 방향으로 갈 수 없는 상황이 생길 수 있으므로 각속도 명령도 포함되어야 한다. 먼저, 선속도는 (4) 식과 같이 계산할 수 있다. 여기서 단위벡터 는 초기 로봇 위치와 목표점으로부터 계산한 값이며, 벡터 는 이동 중에 계속해서 변화하는 벡터 값이다. 이 두 벡터를 내적하여 로봇과 목표점 사이의 직선 성분만 남길 수 있다.

위와 같은 과정으로 거쳐 터틀봇3의 waypoint 기능을 구상 및 구현하였다.