2019학년도 창업연계공학설계입문

AD 프로젝트 결과 보고서

국민대학교 소프트웨어융합대학

소프트웨어학부

4분반 5조

20191632 윤상건

20191638 이민재

20191644 이연주

20150752 김인규

1. 서론

1.1. 초기 구현 목표

초기 구현 주제는 도로가 없고 장애물과 진입금지 표지판이 곳곳에 위치해 있는 상황에서 실시간으로 자신의 위치와 장애물의 위치를 파악하여 목적 위치까지 도달하도록 하는 것이었다.

1.2. 초기 구현 방법

1.2.1. 자신의 위치 파악 방법

xycar의 IMU센서를 이용하여 현재 xycar가 향하고 있는 방향과 선형 가속도 $\vec{a}$를 알 수 있다. 이를 이용하여 xycar의 현재 속도 벡터 $\vec{v}$를 계산할 수 있고, 이를 통해 xycar의 위치 벡터 $\vec{p}$또한 계산할 수 있다.

프레임 f일 때의 가속도, 속도, 위치 벡터를 각각 $\overrightarrow{a_f}$, $\overrightarrow{v_f}$, $\overrightarrow{p_f}$라 하고 프레임 f, $f+1$의 시간 t_f, t_{f+1}사이의 시간 간격을 Δt_f라 하자. 그렇다면 각 벡터는 Trapezoidal rule을 이용한 수치적분을 이용하여 다음과 같이 계산된다.

$$\vec{v}_{f+1} = \int_{t_f}^{t_{f+1}} \vec{a}(t)dt + \vec{v}_f \approx \frac{\vec{a}_{f+1} + \vec{a}_f}{2} \times \Delta t_f + \vec{v}_f$$

$$\vec{p}_{f+1} = \int_{t_f}^{t_{f+1}} \vec{v}(t)dt + \vec{p}_f \approx \frac{\vec{v}_{f+1} + v_f}{2} \times \Delta t_f + \vec{p}_f$$

물론 Δt_f의 값이 0에 가까울수록 즉, 두 프레임 사이 간격이 작을수록 오차는 작아진다.

이를 통해 초기 위치를 기준으로 어느 지점에 위치해 있는지를 파악할 수 있다.

1.2.2. 장애물 위치 파악 방법

xycar의 초음파 센서를 이용하여 어느 위치의 초음파 센서에 얼마만큼의 거리에 장애물이 존재함을 파악할 수 있다. 따라서 어느 초음파에서 감지한 것인가에 따라 IMU 센서에서 탐지한 값을 바탕으로 계산한 xycar의 방향 벡터 $\vec{d}$를 통해 탐지한 장애물까지의 방향벡터를 구할 수 있다. 이후 측정된 거리 정보를 이용하여 장애물의 위치를 특정할 수 있다.

xycar에 장착된 초음파 센서 i의 수직벡터 $\vec{u}$의 xycar 정면을 향하는 방향 벡터 $\vec{d}$에 대한 반시계방향 각을 θ_i라고 하면, $\vec{u}$의 벡터 $(1,0)$에 대한 각 θ는 다음과 같다.

$$\theta = \arccos\frac{\vec{d}\cdot(1,0)}{\|\vec{d}\|} + \theta_i$$

따라서 $\vec{u}$의 단위벡터는 다음과 같다.

$$\frac{\vec{u}}{\|\vec{u}\|} = \begin{bmatrix}\cos\theta \\ \sin\theta\end{bmatrix}$$

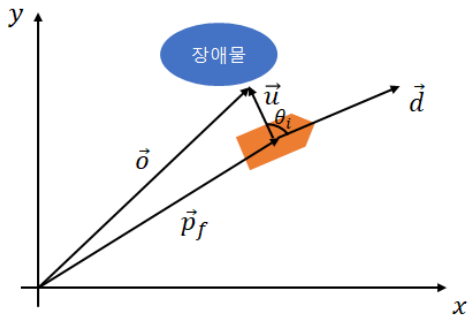
고로 $\vec{u}$는 초음파 센서가 탐지한 거리 s를 통해 다음과 같이 나타낼 수 있다.

$$\vec{u} = s\frac{\vec{u}}{\|\vec{u}\|} = s\begin{bmatrix}\cos\theta \\ \sin\theta\end{bmatrix} = s\begin{bmatrix}\cos\left(\arccos\frac{\vec{d}\cdot(1,0)}{\|\vec{d}\|} + \theta_i\right) \\ \sin\left(\arccos\frac{\vec{d}\cdot(1,0)}{\|\vec{d}\|} + \theta_i\right)\end{bmatrix}$$

해당 벡터는 xycar의 위치를 기준으로 한 것이므로 장애물의 위치벡터 $\vec{o}$는 다음과 같다.

$$\vec{o} = \vec{p}_f + \vec{u}$$

그림으로 표현하면 다음과 같다.



주황색이 xycar를 타나낸다.

1.2.3. 진입금지 표지판 파악 방법

진입금지 표지판은 YOLO (You only look once) 실시간 사물 인식을 이용한다. YOLO의 학습 모델을 가져와 xycar의 카메라 센서가 촬영한 이미지를 입력으로 주어 진입금지 표지판을 인식하도록 한다.

해당 표지판을 인식하면 해당 표지판과 xycar 사이의 거리를 가늠하고 해당 표지판 이후 지역은 방문하지 않도록 한다. 해당 지역은 앞서 보인 것과 비슷한 방식으로 계산할 수 있다.

1.2.4. 이동 경로 계산 방법

사용자가 설정한 xycar의 목적지까지 도달하는데 필요한 경로를 계산하는 과정이 필요하다. 이는 Grid-based A* algorithm을 적용하여 해결할 수 있다.

우선 xycar가 현재 프레임까지 인식한 지도를 grid형태로 나타내고, 해당 grid에서 현재 xycar가

위치한 셀에서 목적지가 위치한 셀까지 장애물을 피해 가장 빠르게 도달할 수 있는 경로를 효율성이 좋은 A* 알고리즘을 이용하여 구한다.

새로운 장애물을 발견하면 A* 알고리즘을 다시 수행하여 경로를 수정해 준다.

1.2.5. 이동 경로 기반 xycar 조향 방법

이동 경로가 계산되었으면 해당 경로에 따라 xycar가 이동해야 한다. 이는 grid로 표현된 지도에서 현재 xycar의 위치에서 다음 셀까지 방향과 xycar가 향하고 있는 방향벡터를 이용하여 이동 방향을 결정하는 방식으로 해결한다. 해당 각도는 두 벡터의 내적을 이용하여 구할 수 있다.

1.3. 제약사항에 따른 개발 목표 설정

1.3.1. IMU 센서의 문제 및 해결 방안

초기 설정한 개발 목표를 달성하기 위해서는 xycar의 IMU 센서의 정확도가 굉장히 중요하다. 하지만 xycar의 IMU센서는 캘리브레이션(Calibration)조차 되지 않은 상태였고, 이에 따라 IMU센서 사용이 불가능 해졌다.

IMU센서 사용이 불가능한 상황에서 개발 목표를 달성하기 위해 xycar가 격자 구조 위에서 동작하는 것과 같이 성능을 제약하여 해결하기로 했다. 즉, xycar의 이동을 맨해튼 거리 에서와 같이 제약하는 것이다. 이와 같은 조건을 가하면 xycar가 임의 위치 (즉, 임의의 가상의 격자)까지 이동하기 위해 필요한 이동에 대한 연산은 90도 제자리 회전 연산과 일정 거리를 이동하는 이동 연산 2가지뿐이다.

1.3.2. 위치 파악 및 장애물 위치 파악 방법의 변경

IMU 센서 사용 불가에 따라 맨해튼 거리를 이용하게 되었다. Xycar의 이동 방향이 x축, y축 각각에 평행하고 이동 거리가 일정해졌기 때문에 위치 파악 또한 90도 연산, 이동 연산을 기준으로 격자 내에서 어디에 위치해 있는지 파악할 수 있다. 따라서 위치 벡터를 이용하는 것이 아닌, 움직임 연산을 기준으로 위치를 추정하기로 했다.

장애물(정지 표지판)의 위치 또한 xycar의 이동 방향이 한정되면서 격자 상에서 xycar의 정면 방향의 격자에 위치한다는 점을 이용해 쉽게 구할 수 있다.

1.3.3. 경로 계산 알고리즘의 변경

IMU센서 사용 불가에 따라 정밀한 grid가 아닌 단순한 맨해튼 거리 grid에서 경로를 계산하게 되었다. 때문에 A*알고리즘을 사용하지 않고 BFS알고리즘 만으로 경로를 빠르게 계산할 수 있다. 따라서 BFS 알고리즘을 이용해 경로 계산 알고리즘을 구현하기로 했다.

1.3.4. 최종 구현 목표

맨해튼 거리 공간에서 움직이는 xycar가 실시간으로 그리드 위에서의 위치를 파악하고 진입금지 표지판을 인식하여 사용자가 설정한 목적지 까지의 적절한 이동 경로를 찾아 자율적으로 이동하는 것을 구현 목표로 했다.

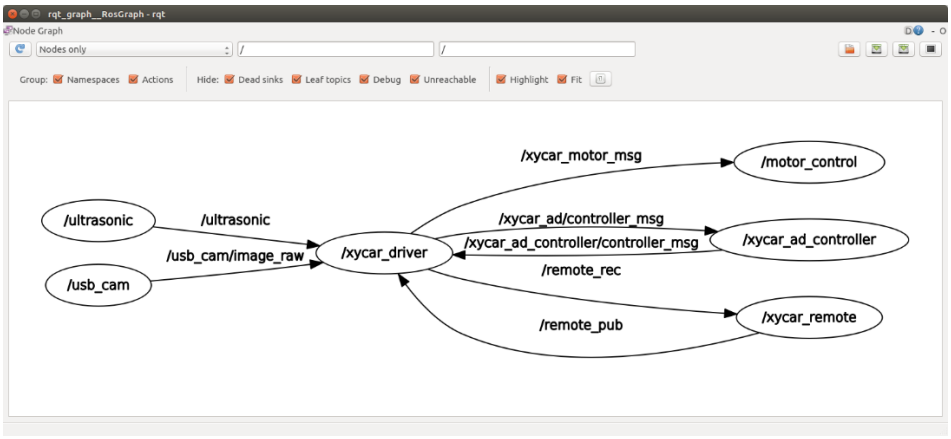
2. 구현

2.1. ROS 노드 및 토픽 구조 설계

사물 인식을 위한 YOLOv3를 사용하기 위해서는 3.4.2 버전 이상의 OpenCV가 필요하다. 하지만 xycar의 개발 환경의 한계로 인해 xycar에서 직접 YOLOv3를 가동하기는 어려웠다. 이를 해결하기 위해 xycar에서 직접 YOLOv3를 실행하지 않고 xycar에 접속하는 원격 컴퓨터(노트북)에서 xycar의 ROS master에 접속하는 노드를 만든 뒤, 해당 노드에서 이미지 데이터를 구독하고 YOLOv3를 실행한 결과를 발행하는 노드를 구상했다.

또한 xycar의 현재 위치와 경로, 장애물의 위치를 출력하고 사용자가 목적지를 설정할 수 있는 GUI기반 인터페이스를 보이는 것이 필요하다. 이는 PyQt5를 이용하기로 했는데, PyQt5의 이벤트 루프를 실행하기 위해서는 또 다른 노드가 필요했다. 이를 위해 현재 지도 상태에 대한 정보를 구독하고 사용자가 명령한 목적지를 발행하는 노드를 구상했다.

구상한 내용을 바탕으로 설계 및 구현된 ROS 노드-토픽 구조는 다음과 같다.



각 노드의 역할은 다음과 같다.

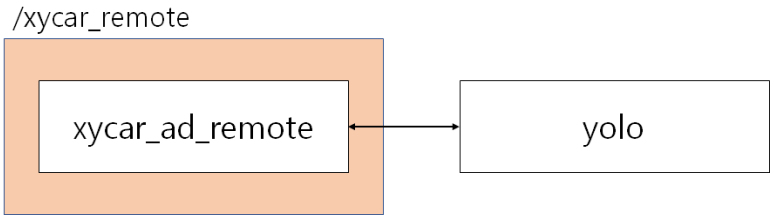
- Xycar부분에서 실행되는 노드
  - /xycar_driver: 자율주행 수행 노드
  - /xycar_ad_controller: Xycar이동 경로 및 이동 명령 창 출력 노드
  - /motor_control: 모터 제어 노드
  - /ultrasonic: 초음파 센서 수신 노드
  - /usb_cam: usb camera에서 영상 데이터 수신 노드
- Remote부분에서 실행되는 노드
  - /xycar_remote: YOLOv3를 이용한 사물인식 노드

각 토픽의 역할은 다음과 같다.

- /xycar_ad/controller_msg: 경로를 포함한 지도 정보를 전달
- /xycar_ad_controller/controller_msg: 사용자가 목적지를 설정한 정보를 전달
- /remote_rec: xycar가 촬영한 이미지 정보 전달
- /remote_pub: YOLOv3가 판단한 사물 인식 정보 전달
- /xycar_motor_msg: 모터 제어 정보 전달
- /ultrasonic: 초음파 센서 정보 전달
- /usb_cam/image_raw: usb camera가 촬영한 이미지 정보

2.2. 각 ROS 노드 별 Python 모듈 구조 설계

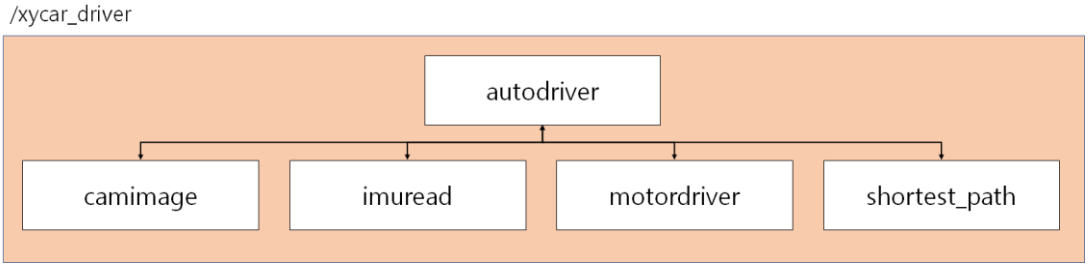
우선 YOLOv3를 구동하는 역할을 담당하는 Remote부분의 /xycar_remote 노드는 다음과 같이 모듈이 구성되어 있다.



yolo의 경우, xycar_ad_remote에서 프로세스로 구동하기 때문에 /xycar_remote의 노드에 포함되지 않는다. 즉, xycar_ad_remote는 YOLOv3결과를 받아올 때 yolo를 프로세스로 실행하면서 이미지 정보를 인자로 넘겨주고 yolo는 인자를 통해 받은 이미지를 YOLOv3를 이용해 처리한 결과를 표준 출력하고 xycar_ad_remote는 이를 받아 문자열 파싱을 통해 사물인식 결과를 받아온다. 각 모듈의 역할을 정리하면 다음과 같다.

- xycar_ad_remote.py: /remote_rec토픽으로 부터 이미지 데이터를 받아 YOLO를 수행하고 결과를 /remote_pub토픽으로 발행하는 모듈
- yolo.py: 이미지 데이터를 입력받고 인식 결과를 표준출력으로 출력하는 모듈

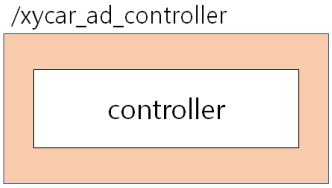
다음으로 Xycar에서 가동되는 노드 중, /xycar_driver 노드는 다음과 같이 모듈이 구성되어 있다.



각각의 모듈의 역할은 다음과 같다.

- autodrive.py: YOLO인식 결과인 /remote_pub토픽과 사용자가 입력한 목적지 정보인 /xycar_ad_controller/controller_msg토픽으로 부터 정보를 받고 shortest_path 모듈로 부터 계산된 경로를 바탕으로 자율주행을 수행하는 모듈
- camimage.py: /usb_cam/image_raw토픽에서 카메라가 촬영한 이미지를 받아오는 모듈
- imuread.py: IMU센서 데이터를 가져오는 모듈 (IMU센서의 문제로 사용하지 않음)
- motordriver.py: /xycar_motor_msg토픽으로 모터 제어 정보를 발행하는 모듈
- shortest_path.py: 현재 위치, 장애물 정보, 목적 위치를 관리하고 최단 경로를 계산하는 모듈

마지막으로 xycar에서 가동되는 /xycar_ad_controller 의 모듈 구조는 다음과 같다.



모듈의 역할은 다음과 같다.

- controller.py: 현재 xycar의 위치와 앞으로의 경로를 출력하고 사용자로 부터 목적지를 명령받는 PyQt를 이용한 GUI창 출력하는

2.3. 주요 구현 내용

2.3.1. Xycar의 90도 회전 연산

그리드 구조에서 상, 하, 좌, 우로 이동하기 위해서는 Xycar의 방향을 시계방향 또는 반시계방향으로 90도 1회전하는 연산을 구현하는 것이 필요하다. 이때, 중요한 조건은 다음과 같다.

- 회전을 수행하는 동안 현재 격자를 벗어나지 않도록 해야 한다.
- 회전한 이후, Xycar의 중심이 회전하기 전의 위치와 동일해야 한다.
- 회전한 각이 90도가 되어야 한다.

Xycar의 모터를 정밀히 조작하는 것은 불가능 하기 때문에 위 조건을 최대한 만족하도록 구현했다.

2.3.2. Xycar의 전진 연산

그리드 구조에서 나아가기 원하는 방향으로 회전했다면, 전진하는 연산이 필요하다. 이때, 전진은 다음과 같은 조건을 만족해야 한다.

- 전진하는 속도는 일정해야 한다.
- 전진하는 시간은 일정해야 한다.

회전 연산 때와 마찬가지로, Xycar의 모터를 정밀히 조작하는 것은 불가능 하기 때문에 첫 번째 조건은 최대한 느린 속도로 움직이도록 하여 오차를 최대한 줄이도록 했다. 두 번째 조건은 소프트웨어 상에서 쉽게 구현할 수 있다. 하지만 해당 명령이 모터에 전달되는 과정에서 발생하는 시간에 의해 미세한 오차가 발생할 수 있다.

2.3.3. YOLOv3를 이용한 사물인식

앞서 ROS 노드 및 토픽 구조에서 설명한 바와 같이, YOLOv3를 사용하기 위해 원격 노드를 구현했고, 해당 원격 노드에서 YOLOv3를 이용한 사물인식을 수행하기 위해 python의 subprocess모듈을 사용하여 yolo모듈을 프로세스로 실행 및 표준 출력을 통해 결과를 받아왔다.

2.3.4. 경로 계산 연산

격자 구조에서 현재 위치에서 목적지까지 이동 경로를 계산하기 위한 연산이 필요하다. 각 격자 사이의 거리가 1이므로, 가중치가 1일 때 최단 경로를 계산할 수 있는 BFS 알고리즘을 이용했다.

BFS를 수행할 때, 각 지점에서 이전 지점을 메모이제이션 하여 도착 지점에서 이전 지점을 추적해 나가는 방식으로 경로를 계산했다.

2.3.5. 경로 출력 및 제어 GUI 구현

Xycar가 계산한 경로, 현재 위치, 목적지, 발견한 장애물의 위치를 격자상에 출력하고 사용자가 목적지를 설정할 수 있는 GUI가 필요하다. 이를 구현하기 위해 PyQt5를 이용했다. 또한 PyQt의 이벤트 루프가 필요하기 때문에 새로운 노드에서 구현했다.

GUI에는 경로 정보가 담긴 ROS 메시지를 구독 받아 격자 구조가 출력되게 하고, 사용자가 특정 위치의 격자를 클릭하면 해당 지점을 목적지로 설정하는 ROS 메시지를 발행하도록 했다.

3. 구현 결과

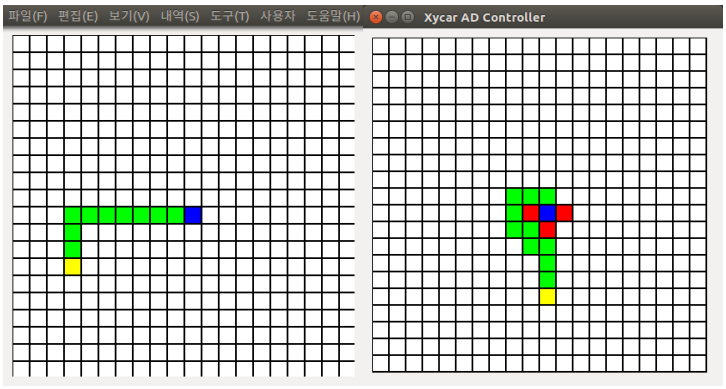
3.1. 표지판 인식 결과

이용한 정지 표지판은 다음과 같다.

YOLOv3를 이용하여 다음과 같이 성공적으로 인식한 결과를 확인했다.

3.2. 경로 계산 및 경로 출력 및 제어 GUI 구현 결과

구현된 경로 출력 및 제어 GUI는 다음과 같다.



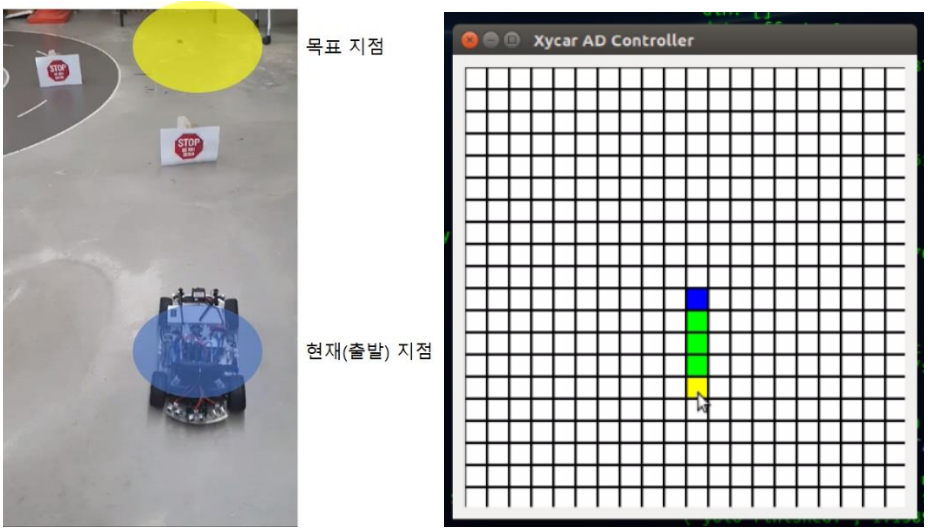
각 격자의 색은 다음을 의미한다.

- 파랑색: Xycar의 현재 지점
- 초록색: 계산한 경로
- 노랑색: 사용자가 설정한 목적지
- 빨간색: Xycar가 탐지한 장애물

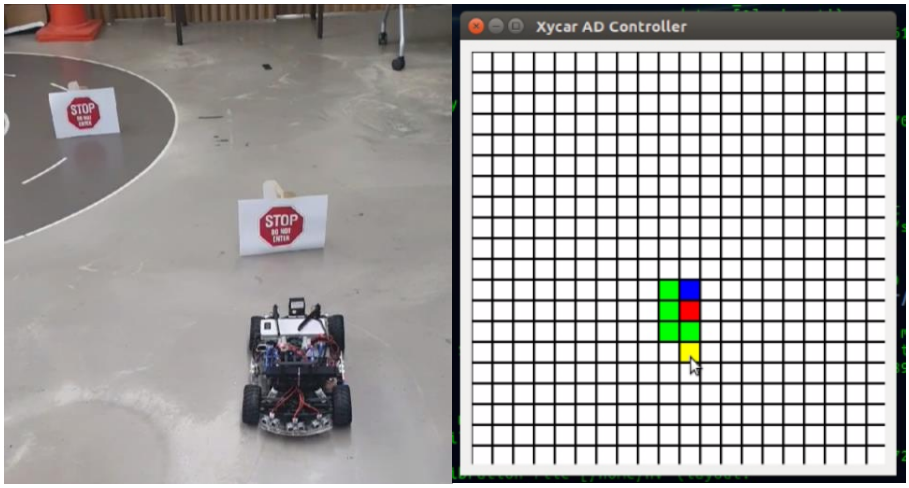
3.3. 주행 테스트 결과

실제 주행 테스트를 진행한 결과, 다음과 같다.

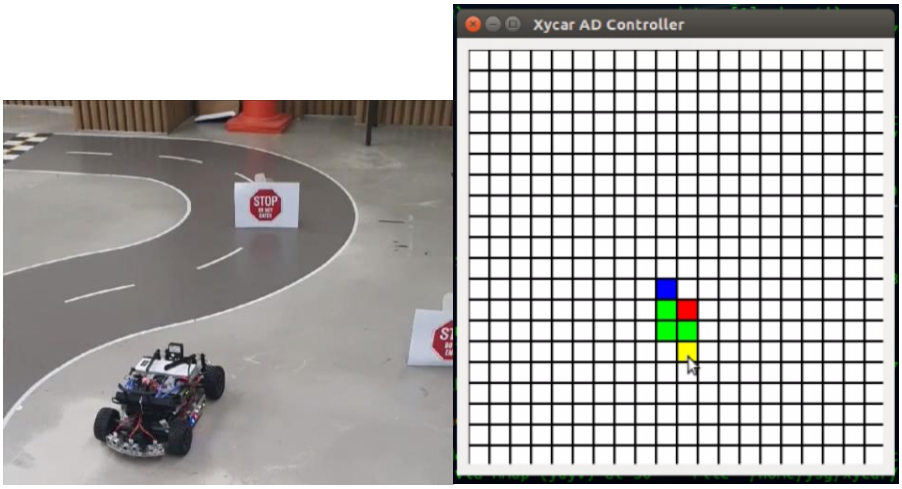
다음은 제어 GUI창에서 설정한 목표 지점(우측)과 실제 위치(왼쪽)이다.



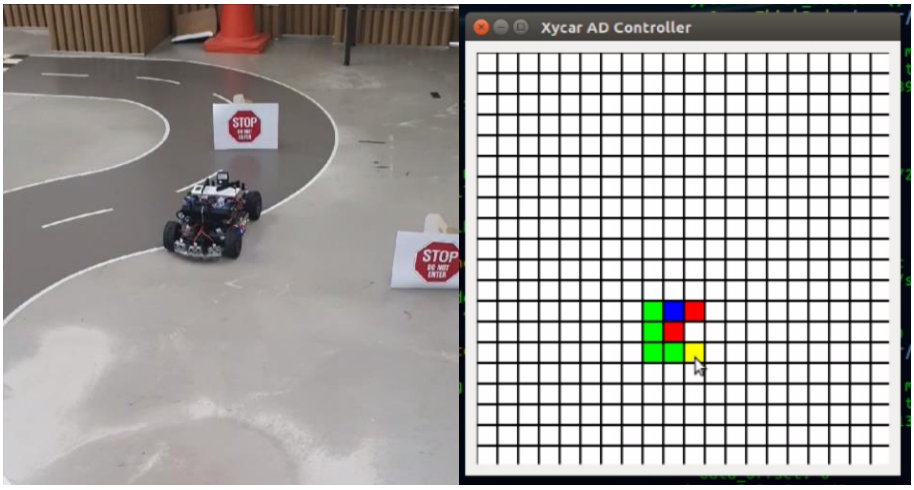
다음에서 표지판을 인식한 모습을 확인할 수 있다.



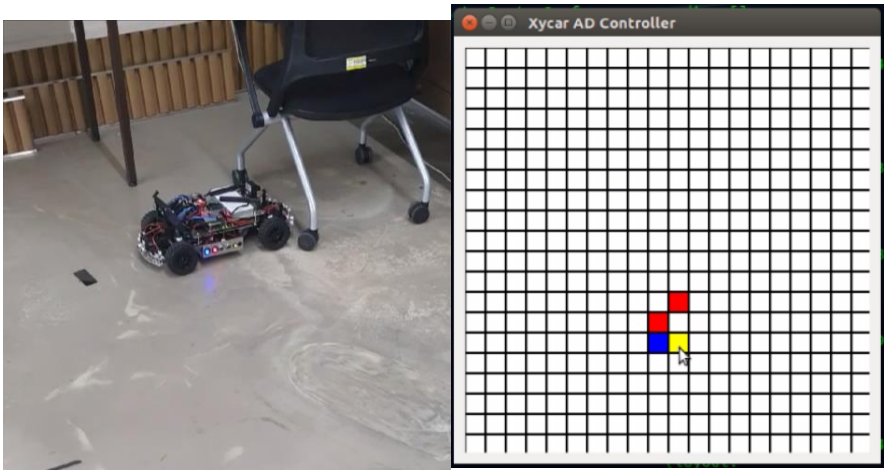
90도 회전 연산 및 전진 연산을 통해 계산된 최단 경로로 이동하는 모습을 볼 수 있다.



최단 경로로 이동하던 중, 새로운 장애물을 발견하고 경로를 수정한 모습이다.



사용자가 설정한 목적지까지 자율적으로 도착한 모습이다.



모든 연산을 수행하여 사용자가 설정한 목적지까지 자율적으로 도달하는 성공적인 모습을 보였다.