AIoT AutoCar Prime 으로 배우는

온디바이스 AI 프로그래밍

3 AIoT AutoCar Prime 제어

AIoT AutoCar Prime 제어

- pop.Pilot 라이브러리의 AutoCar클래스를 이용해 쉽게 제어 가능
  - 차량의 전/후진과 좌/우회전을 제어하는 메소드
  - 카메라를 위/아래로 움직이거나 좌/우로 회전시키는 메소드
  - 6축 센서 값을 읽어오는 메소드

차량 제어

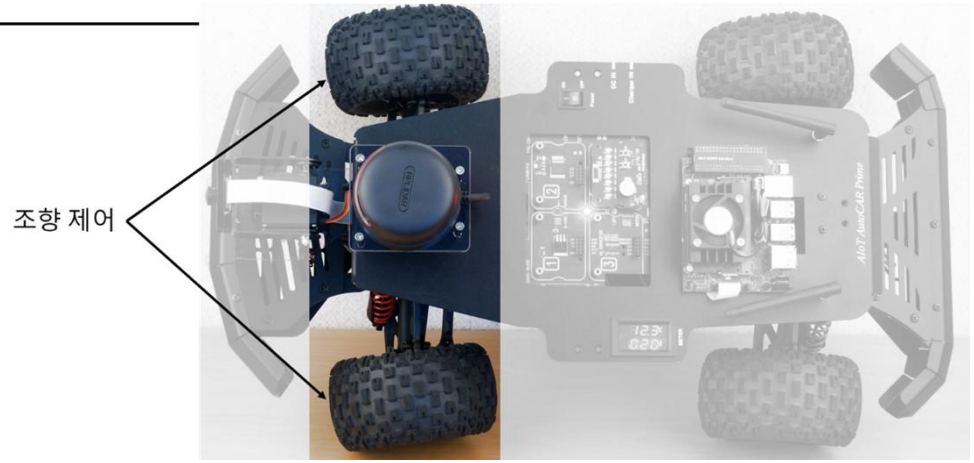
- pop에서 Pilot라이브러리를 import
- AutoCar 클래스 생성
- 코드 진행에 딜레이를 주고 관찰하기 위해 time 모듈 import

```
01:     import time
02:     from pop import Pilot
03:
04:     Car = Pilot.AutoCar()
```

차량 제어 – 조향

- AutoCar의 steering 속성 값을 입력하면 차량을 좌우로 조향 가능

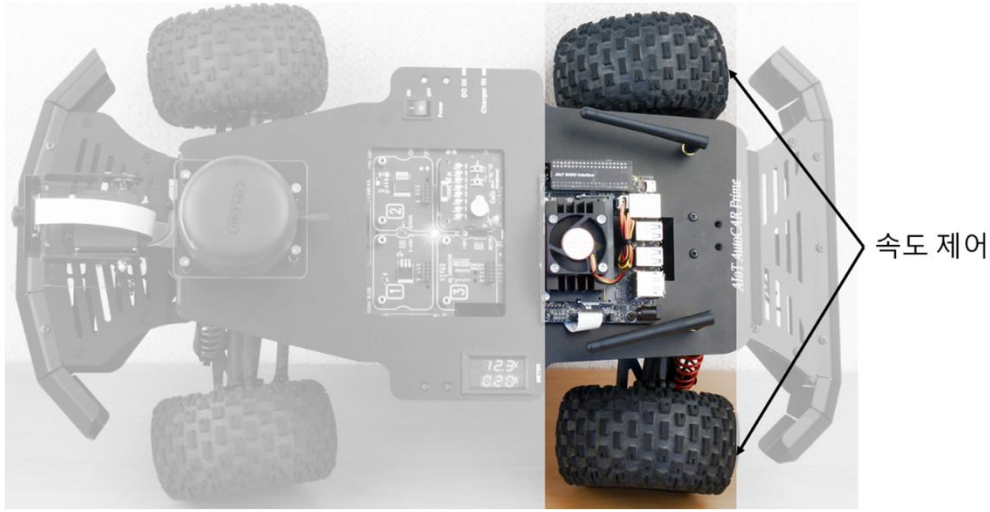
```
05:         Car.steering = -1
06:
07:         time.sleep(1)
08:
09:         Car.steering = 1
```



차량 제어 – 전진,후진

- forward() 메소드 : 차량 전진
- backward() 메소드 : 차량 후진
- stop() 메소드 : 차량 정지

```
10:         Car.forward()
11:
12:         time.sleep(1)
13:
14:         Car.backward()
15:
16:         time.sleep(1)
17:
18:         Car.stop()
```



차량 제어 – 속도 제어

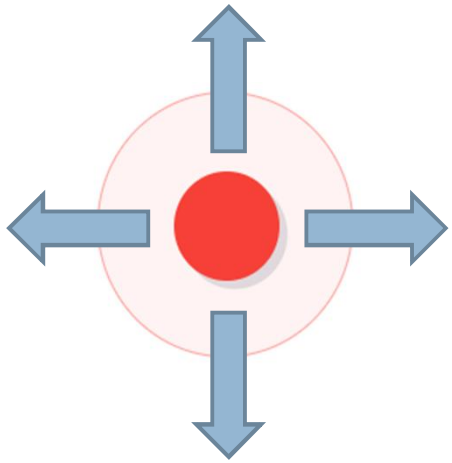
- setSpeed(value) 메소드 : 속도 설정 (0~99)
- forward(), backward() 메소드의 파라미터로 속도를 입력 가능

```
19:         Car.setSpeed(50)
20:         Car.forward()
21:
22:         time.sleep(1)
23:
24:         Car.backward(99)
25:
26:         time.sleep(1)
27:
28:         Car.stop()
```

차량 제어 – 가상 조이스틱

- joystick() 메소드
  - 가상의 조이스틱 표시
  - 앞, 뒤, 좌, 우로 조작하면 차량이 움직임

```
29:          Car.joystick()
```

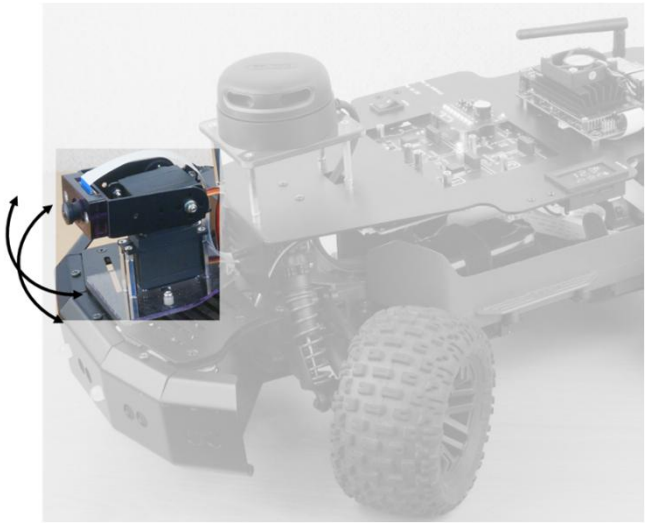


카메라 제어

- camPan() 메소드 : 카메라 좌,우 제어
- camTilt() 메소드 : 카메라 위,아래 제어

```
30:         Car.camPan(0)
31:
32:         time.sleep(1)
33:
34:         Car.camPan(180)
35:
36:         time.sleep(1)
37:
38:         Car.camTilt(180)
39:
40:         time.sleep(1)
41:
42:         Car.camTilt(0)
```

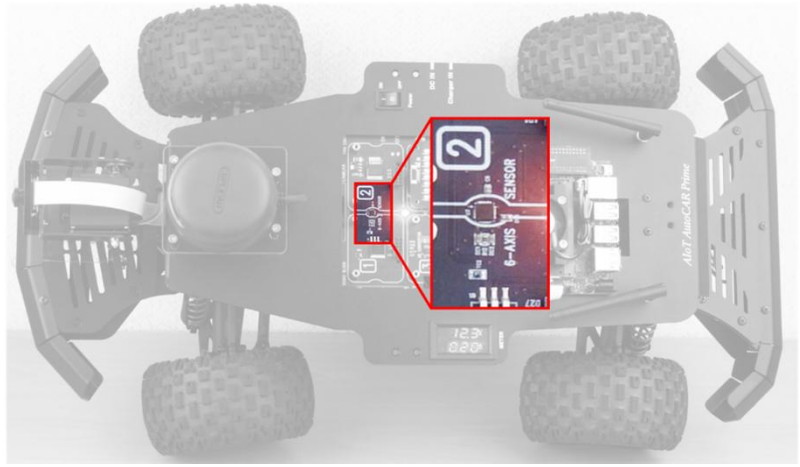
camPan(): 좌 우 제어
camTilt(): 위 아래 제어



6축 센서

- getGyro() 메소드 : 6축 센서의 자이로 값 읽기
  - 파라미터를 입력하지 않으면 모든 축에 대한 값 반환
  - 파라미터로 'x', 'y', 'z'를 입력하면 해당 축의 값 반환

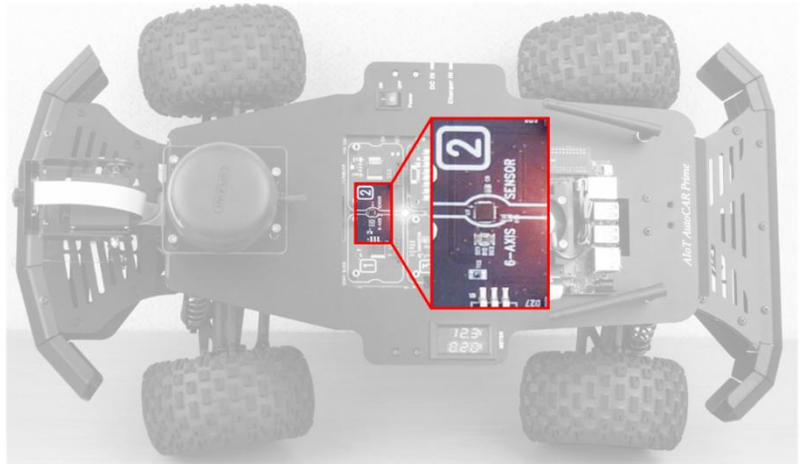
```
43:     value = Car.getGyro()
44:     print(value)
45:
46:     time.sleep(1)
47:
48:     value = Car.getGyro('z')
49:     print(value)
```



6축 센서

- getAccel() 메소드 : 6축 센서의 가속도 값 읽기
  - 파라미터를 입력하지 않으면 모든 축에 대한 값 반환
  - 파라미터로 'x', 'y', 'z'를 입력하면 해당 축의 값 반환

```
50:     value = Car.getAccel()
51:     print(value)
52:
53:     time.sleep(1)
54:
55:     value = Car.getAccel('z')
56:     print(value)
```



LiDAR 센서

- LiDAR : 원거리에서 물체를 탐지하는 장치
  - 360도로 회전하는 LiDAR는 각도와 거리 값 측정 가능
  - AIoT AutoCAR Prime 은 LiDAR 센서 기본 장착

LiDAR 기본 제어

- Pop의 LiDAR라이브러리에서 장비를 제어하는 주요 메소드
  - connect() : LiDAR 연결
  - startMotor() : LiDAR 스캐닝 모터 시작
  - stopMotor() : LiDAR 스캐닝 모터 정지
  - getVertors() : 1회전 동안 스캔한 벡터값 튜플 (Degree, Distance, Quality)들을 각도 기준 오름차순으로 정렬한 리스트 반환
  - getXY() : 1회전 동안 스캔한 좌표값 튜플 (X, Y)들을 각도 기준 오름차순으로 정렬한 리스트 반환

LiDAR 기본 제어

- getMap(size, limit_distance) : 1회전 동안 스캔한 값을 2차원 맵으로 반환
  - size : 반환할 맵의 사이즈. 기본값: (300, 300)
  - limit_distance : 맵에 표현할 최대 거리, 2000으로 설정한다면 2000mm 이하의 값만 맵으로 반영. 기본값 : 12000

LiDAR 기본 제어

- Pop에서 LiDAR라이브러리를 불러오고 Rplidar 클래스를 생성

```
01:     from pop import LiDAR
02:
03:     lidar = LiDAR.Rplidar()
```

- connect() 메소드로 LiDAR와 연결
- startMotor() 메소드로 스캐닝 모터 동작

```
04:     lidar.connect()
05:     lidar.startMotor()
```

LiDAR 기본 제어

- getVectors() 메소드 사용
  - 1회전 동안 스캔된 각도와 거리 데이터를 갖는 벡터 객체 반환
    - (각도, 거리, 데이터 신뢰도)
    - 각도는 Degree, 거리는 mm 단위
    - 데이터 신뢰도는 반사되어 돌아온 빛의 산란도와 굴절도 등으로 산출된 값으로 0~47 사이의 값

```
06:         vectors = lidar.getVetors()
07:
08:         for v in vertors:
09:                         print(v)
```

LiDAR 기본 제어

- getXY() 메소드 사용
  - 1회전 동안 스캔된 X좌표와 Y좌표 데이터를 갖는 좌표 객체 반환
    - (X, Y)
    - 단위는 mm

```
10:         coords = lidar.getXY()
11:
12:         for c in coords:
13:                 print(c)
```

LiDAR 기본 제어

- getMap () 메소드 사용
  - 1회전 동안 스캔된 값을 2차원 배열로 반환
  - 2000mm 이하의 값만 스캔해 imshow() 메소드로 출력해보면 회색조 맵 출력
  - 2차원 배열 값의 범위 : 0~255
    - 해당 좌표에 인식된 값이 많을수록 255에 가까워짐.

```
14:     from pop.Util import imshow
15:
16:     lidar_map = lidar.getMap(limit_distance=2000, size=(300,300))
17:     imshow("map", lidar_map)
```

LiDAR 기본 제어

- getMap() 메소드의 size를 (50,50)으로 조절하면 조금 더 간소화된 맵 출력

```
18:     lidar_map = lidar.getMap(limit_distance=2000, size=(50,50))
19:     imshow("map", lidar_map)
```

LiDAR 응용 제어

- Pop에서 LiDAR라이브러리를 불러오고 Rplidar 클래스를 생성

```
01:     from pop import LiDAR, Pilot
02:
03:     lidar = LiDAR.Rplidar()
04:     car = Pilot.AutoCar()
```

- startMotor() 메소드로 스캐닝 모터 동작
- 주행속도를 99로 차량의 조향을 가운데로 설정

```
05:     lidar.connect()
06:     lidar.startMotor()
07:
08:     car.setSpeed(99)
09:     car.steering = 0
```

LiDAR 응용 제어

- LiDAR에서 백터값을 받아 전방 120도 범위 내에 물체를 감지

```
10:         while True:
11:             no_collision = True
12:
13:             vectors = lidar.getVectors()
14:             for v in vectors:
15:                 degree = v[0]
16:                 distance = v[1]
17:
18:                 if degree <= 60 or degree >= 300:
19:                     if distance <= 700:
20:                         no_collision = False
21:
```

```
22:             if no_collision:
23:                 car.setSpeed(99)
24:                 car.forward()
25:             else:
26:                 if car.steering == 0:
27:                     car.stop()
28:                 car.steering = -1
29:                 car.backward(50)
```

내용 정리

- 차량 제어 주요 메소드
  - steering : -1 ~ 1 값을 지정하면 차량의 앞바퀴를 좌/우로 조향
  - setSpeed(value) : 0~99 값으로 차량의 속력을 제어
  - stop() : 차량을 정지
  - forward() : 차량을 전진
  - backward() : 차량을 후진
  - joystick() : 가상 조이스틱을 이용해 차량을 제어

내용 정리

- 카메라 제어 주요 메소드
  - camPan(value) : 0~180 값으로 카메라를 좌/우로 제어
  - camTilt(value) : -30~200 값으로 카메라를 위/아래로 제어

내용 정리

- 6축 센서 주요 메소드
  - getGyro(axis)
    - ’ x ‘, ’ y’ , ‘z’ 를 입력하면 해당 축의 자이로 값 반환
    - 아무 입력없이 호출시 모든 축에 대한 값을 딕셔너리 반환
  - getAccel(axis)
    - ’ x ‘, ’ y’ , ‘z’ 를 입력하면 해당 축의 가속도 값 반환
    - 아무 입력없이 호출시 모든 축에 대한 값 딕셔너리로 반환

내용 정리

- LiDAR 센서 주요 메소드 – Pop의 LiDAR 라이브러리
  - connect() : LiDAR 연결
  - startMotor() : LiDAR 스캐닝 모터 시작
  - stopMotor() : LiDAR 스캐닝 모터 정지
  - getVertors() : 1회전 동안 스캔한 벡터값 튜플 (Degree, Distance, Quality) 들을 각도 기준 오름차순으로 정렬한 리스트 반환

내용 정리

- getXY() : 1회전 동안 스캔한 좌표값 튜플 (X, Y)들을 각도 기준 오름차순으로 정렬한 리스트 반환
- getMap(size, limit_distance) : 1회전 동안 스캔한 값을 2차원 맵으로 반환