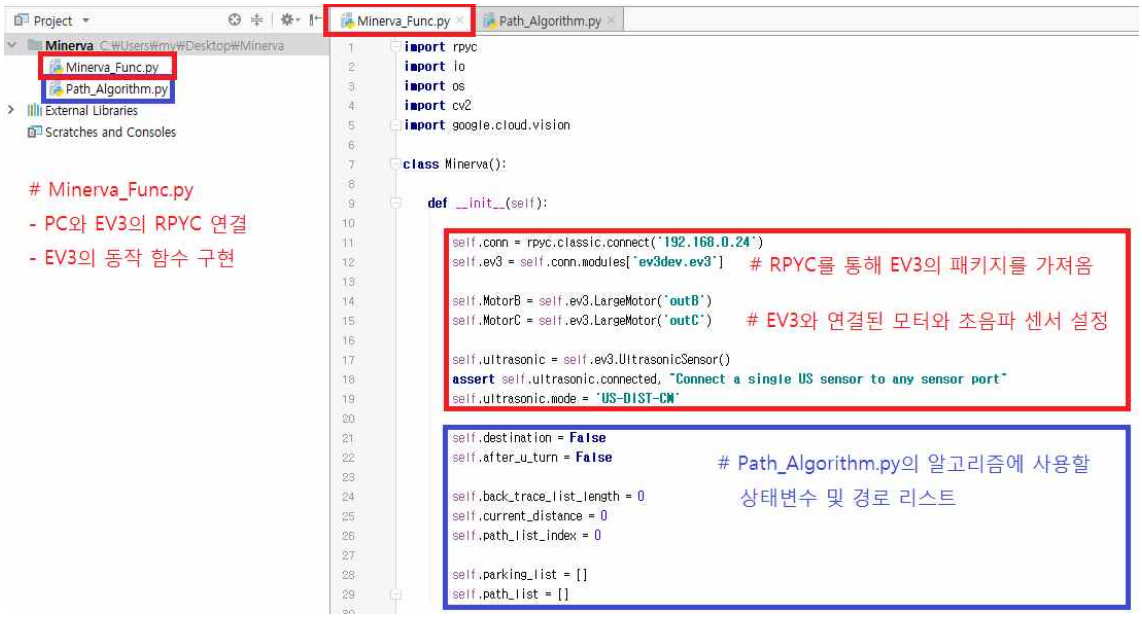
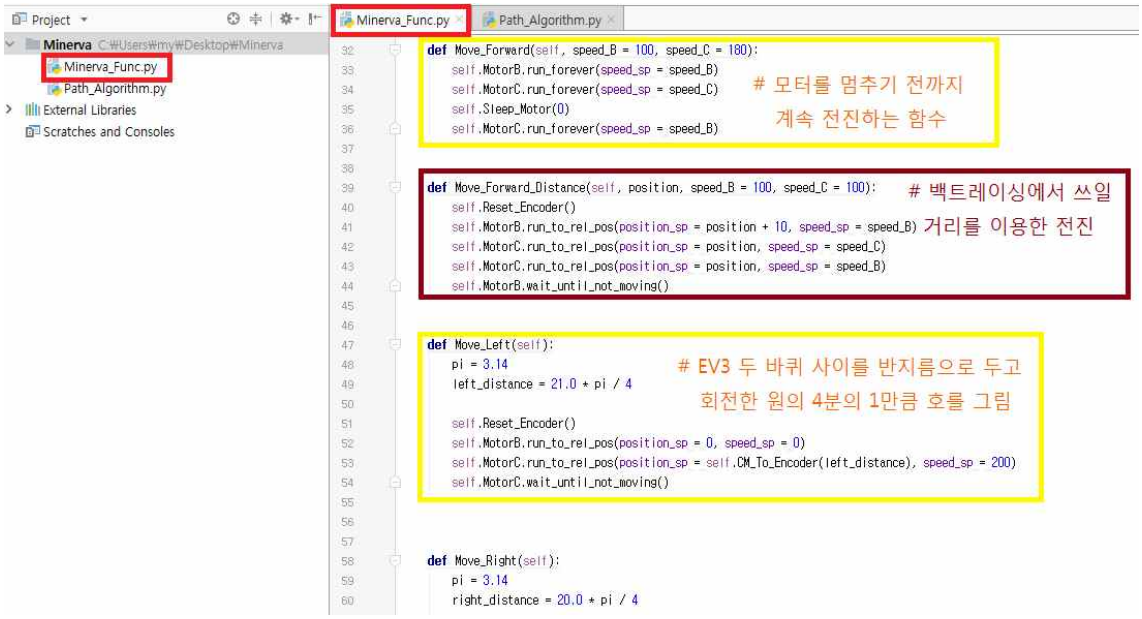
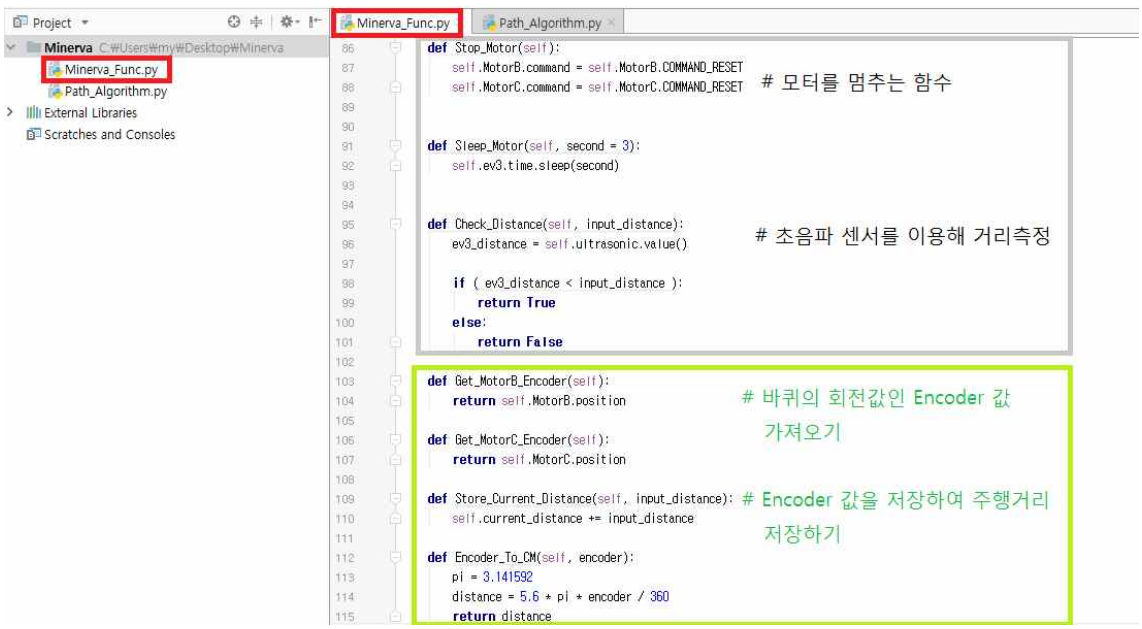
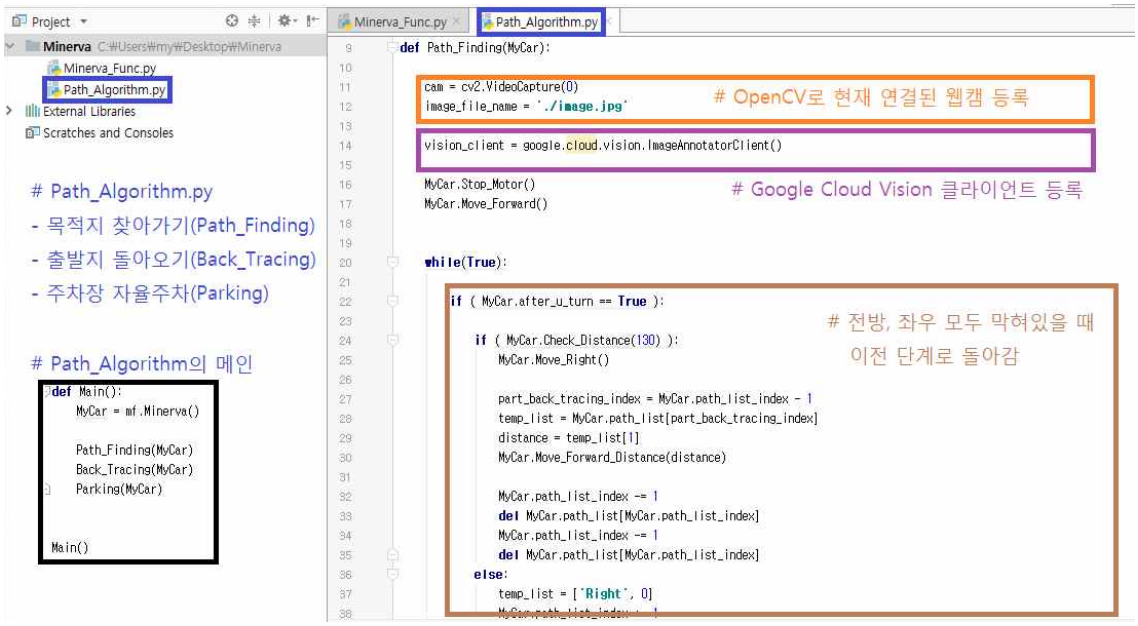
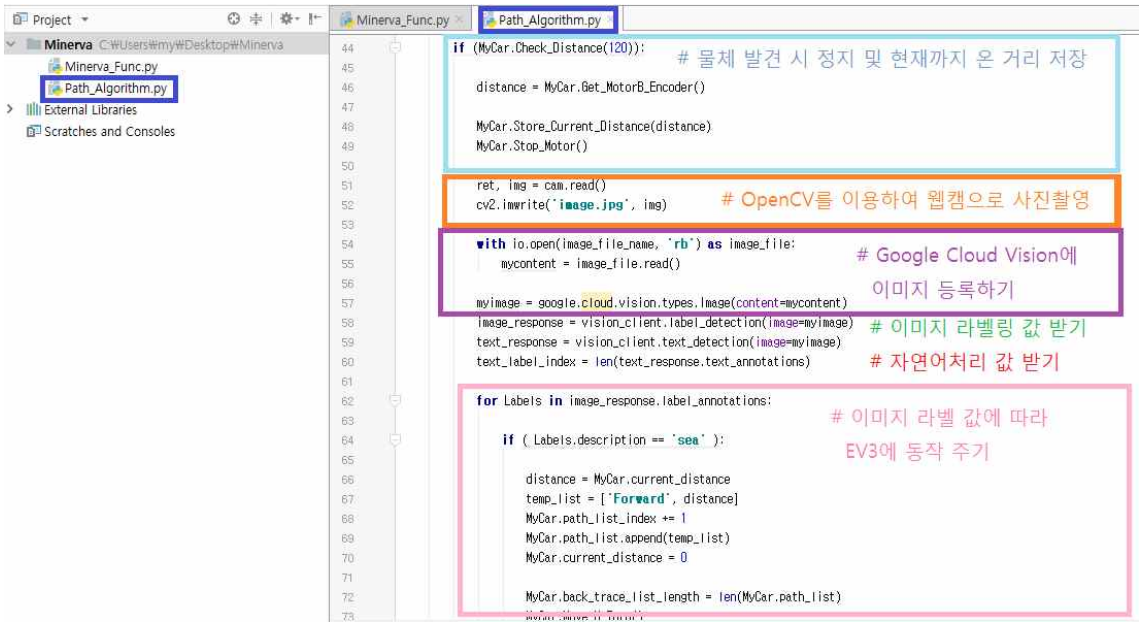
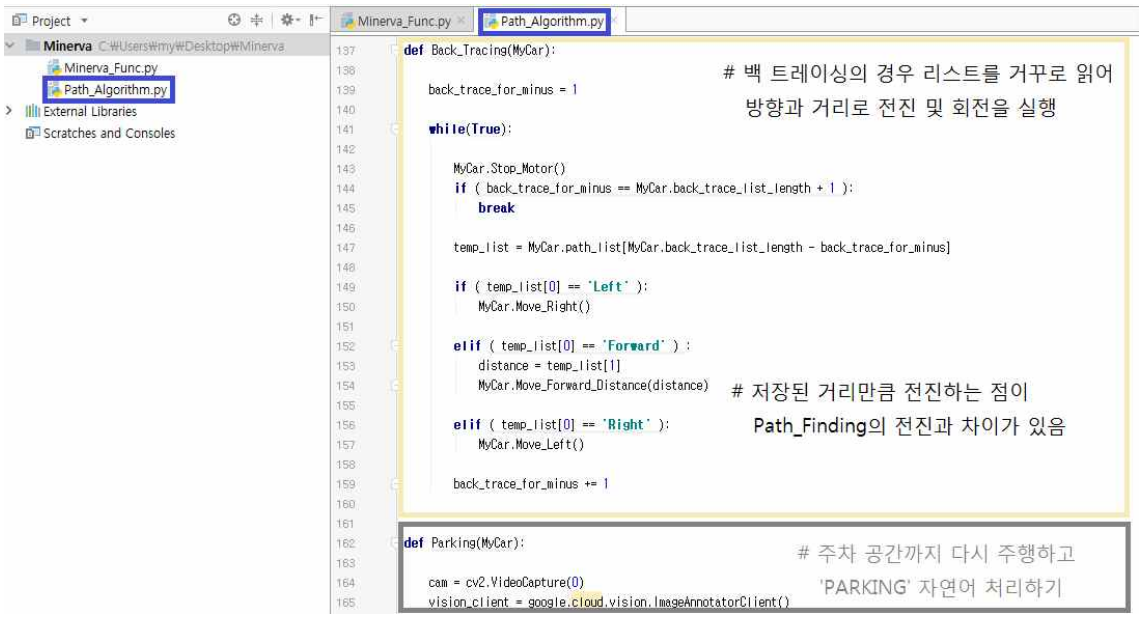

Project
Minerva C:₩Users₩my₩Desktop₩Minerva
Minerva_Func.py
Path_Algorithm.py
External Libraries
Scratches and Consoles
Minerva_Func.py
- PC와 EV3의 RPYC 연결
- EV3의 동작 함수 구현
import rpyc
import io
import os
import cv2
import google.cloud.vision
class Minerva():
def __init__(self):
self.conn = rpyc.classic.connect('192.168.0.24')
self.ev3 = self.conn.modules['ev3dev.ev3']
# RPYC를 통해 EV3의 패키지를 가져옴
self.MotorB = self.ev3.LargeMotor('outB')
self.MotorC = self.ev3.LargeMotor('outC')
# EV3와 연결된 모터와 초음파 센서 설정
self.ultrasonic = self.ev3.UltrasonicSensor()
assert self.ultrasonic.connected, "Connect a single US sensor to any sensor port"
self.ultrasonic.mode = 'US-DIST-CM'
self.destination = False
self.after_u_turn = False
# Path_Algorithm.py의 알고리즘에 사용할
상태변수 및 경로 리스트
self.back_trace_list_length = 0
self.current_distance = 0
self.path_list_index = 0
self.parking_list = []
self.path_list = []


Project
Minerva C:₩Users₩my₩Desktop₩Minerva
Minerva_Func.py
Path_Algorithm.py
External Libraries
Scratches and Consoles
def Move_Forward(self, speed_B = 100, speed_C = 180):
self.MotorB.run_forever(speed_sp = speed_B)
self.MotorC.run_forever(speed_sp = speed_C)
self.Sleep_Motor(0)
self.MotorC.run_forever(speed_sp = speed_B)
# 모터를 멈추기 전까지
계속 전진하는 함수
def Move_Forward_Distance(self, position, speed_B = 100, speed_C = 100):
self.Reset_Encoder()
self.MotorB.run_to_rel_pos(position_sp = position + 10, speed_sp = speed_B)
self.MotorC.run_to_rel_pos(position_sp = position, speed_sp = speed_C)
self.MotorC.run_to_rel_pos(position_sp = position, speed_sp = speed_B)
self.MotorB.wait_until_not_moving()
# 백트레이싱에서 쓰일
거리를 이용한 전진
def Move_Left(self):
pi = 3.14
left_distance = 21.0 * pi / 4
self.Reset_Encoder()
self.MotorB.run_to_rel_pos(position_sp = 0, speed_sp = 0)
self.MotorC.run_to_rel_pos(position_sp = self.CM_To_Encoder(left_distance), speed_sp = 200)
self.MotorC.wait_until_not_moving()
# EV3 두 바퀴 사이를 반지름으로 두고
회전한 원의 4분의 1만큼 호를 그림
def Move_Right(self):
pi = 3.14
right_distance = 20.0 * pi / 4


Project
Minerva C:₩Users₩my₩Desktop₩Minerva
Minerva_Func.py
Path_Algorithm.py
External Libraries
Scratches and Consoles
def Stop_Motor(self):
self.MotorB.command = self.MotorB.COMMAND_RESET
self.MotorC.command = self.MotorC.COMMAND_RESET
# 모터를 멈추는 함수
def Sleep_Motor(self, second = 3):
self.ev3.time.sleep(second)
def Check_Distance(self, input_distance):
ev3_distance = self.ultrasonic.value()
if (ev3_distance < input_distance):
return True
else:
return False
# 초음파 센서를 이용해 거리측정
def Get_MotorB_Encoder(self):
return self.MotorB.position
def Get_MotorC_Encoder(self):
return self.MotorC.position
바퀴의 회전값인 Encoder 값
가져오기
def Store_Current_Distance(self, input_distance):
self.current_distance += input_distance
Encoder 값을 저장하여 주행거리
저장하기
def Encoder_To_CM(self, encoder):
pi = 3.141592
distance = 5.6 * pi * encoder / 360
return distance


Project
Minerva C:₩Users₩my₩Desktop₩Minerva
Minerva_Func.py
Path_Algorithm.py
External Libraries
Scratches and Consoles
Path_Algorithm.py
- 목적지 찾아가기(Path_Finding)
- 출발지 돌아오기(Back_Tracing)
- 주차장 자율주차(Parking)
# Path_Algorithm의 메인
def Main():
MyCar = mf.Minerva()
Path_Finding(MyCar)
Back_Tracing(MyCar)
Parking(MyCar)
Main()
def Path_Finding(MyCar):
cam = cv2.VideoCapture(0)
image_file_name = './image.jpg'
# OpenCV로 현재 연결된 웹캠 등록
vision_client = google.cloud.vision.ImageAnnotatorClient()
# Google Cloud Vision 클라이언트 등록
MyCar.Stop_Motor()
MyCar.Move_Forward()
while(True):
if (MyCar.after_u_turn == True):
if (MyCar.Check_Distance(130)):
MyCar.Move_Right()
part_back_tracing_index = MyCar.path_list_index - 1
temp_list = MyCar.path_list[part_back_tracing_index]
distance = temp_list[1]
MyCar.Move_Forward_Distance(distance)
MyCar.path_list_index -= 1
del MyCar.path_list[MyCar.path_list_index]
MyCar.path_list_index -= 1
del MyCar.path_list[MyCar.path_list_index]
else:
temp_list = ['Right', 0]
# 전방, 좌우 모두 막혀있을 때
이전 단계로 돌아감


Project
Minerva C:₩Users₩my₩Desktop₩Minerva
Minerva_Func.py
Path_Algorithm.py
External Libraries
Scratches and Consoles
if (MyCar.Check_Distance(120)):
# 물체 발견 시 정지 및 현재까지 온 거리 저장
distance = MyCar.Get_MotorB_Encoder()
MyCar.Store_Current_Distance(distance)
MyCar.Stop_Motor()
ret, img = cam.read()
cv2.imwrite('image.jpg', img)
# OpenCV를 이용하여 웹캠으로 사진촬영
with io.open(image_file_name, 'rb') as image_file:
mycontent = image_file.read()
myimage = google.cloud.vision.types.Image(content=mycontent)
image_response = vision_client.label_detection(image=myimage)
text_response = vision_client.text_detection(image=myimage)
text_label_index = len(text_response.text_annotations)
# Google Cloud Vision에
이미지 등록하기
# 이미지 라벨링 값 받기
# 자연어처리 값 받기
for Labels in image_response.label_annotations:
if (Labels.description == 'sea'):
distance = MyCar.current_distance
temp_list = ['Forward', distance]
MyCar.path_list_index += 1
MyCar.path_list.append(temp_list)
MyCar.current_distance = 0
MyCar.back_trace_list_length = len(MyCar.path_list)
# 이미지 라벨 값에 따라
EV3에 동작 주기


Project
Minerva C:₩Users₩my₩Desktop₩Minerva
Minerva_Func.py
Path_Algorithm.py
External Libraries
Scratches and Consoles
def Back_Tracing(MyCar):
back_trace_for_minus = 1
while(True):
MyCar.Stop_Motor()
if (back_trace_for_minus == MyCar.back_trace_list_length + 1):
break
temp_list = MyCar.path_list[MyCar.back_trace_list_length - back_trace_for_minus]
if (temp_list[0] == 'Left'):
MyCar.Move_Right()
elif (temp_list[0] == 'Forward') :
distance = temp_list[1]
MyCar.Move_Forward_Distance(distance)
elif (temp_list[0] == 'Right'):
MyCar.Move_Left()
back_trace_for_minus += 1
백 트레이싱의 경우 리스트를 거꾸로 읽어
방향과 거리로 전진 및 회전을 실행
저장된 거리만큼 전진하는 점이
Path_Finding의 전진과 차이가 있음
def Parking(MyCar):
cam = cv2.VideoCapture(0)
vision_client = google.cloud.vision.ImageAnnotatorClient()
# 주차 공간까지 다시 주행하고
'PARKING' 자연어 처리하기