#!/usr/bin/python

/usr/bin폴더에 있는 python파일을 사용하여 실행한다.

import rospy, math, time

rospy, math, time의 모듈을 가져온다.

from std\_msgs.msg import Int32MultiArray

std\_msgs.msg라는 모듈으로부터 Int32MultiArray 변수를 가져온다.

fr = 0.0

front right 값을 0.0으로 고정 시켜준다

def callback(msg):

callback은 다른 코드의 인수로서 넘겨주는 실행 가능한 코드를 말한다. callback을 넘겨받는 msg는 이callback을 필요에 따라 즉시 실행할 수도 있고, 아니면 나중에 실행할 수도 있다

global fr

fr 변수를 글로벌로 전역선언한다.

fr = msg.data[2]

print(msg)

msg라는 결과물을 출력한다.

rospy.init\_node('guide')

프로세스의 ros노드를 초기화(선언)하는 것으로 rospy에게 사용자 코드가 속한 노드의 이름을 알려주는 것이다. 사용자 노드 이름을 ‘guide’로 설정하였다. Rospy가 이 정보를 얻기전까지 사용자는 ROS Master와 통신하지 못한다

Rospy 프로세스에는 하나의 노드만 있을수 있으므로 rospy.init\_node()를 한번만 호출 할 수 있다.

여기서 중요한 점은 노드 이름이 고유해야한다는 것이다. 동일한 이름의 노드가 여러 개라면 한 개의 노드가 실행될때 다른 노드는 종료된다.

motor\_pub = rospy.Publisher('xycar\_motor\_msg', Int32MultiArray, queue\_size=1)

Int32MultiArray 타입의 message를 사용하는 xycar\_motor\_msg토픽에게 사용자의 노드에서 데이터를 지속적으로 발행한다는 것을 선언한다. queue\_size는 쌓을 큐의 크기를 말하는데 여기서는 초당 1번 발행할 것이다.

ultra\_sub = rospy.Subscriber('ultrasonic', Int32MultiArray, callback)

사용자 노드가 Int32MultiArray형태의 타입의 데이터를 ultrasonic토픽으로부터 구독한다는 선언이다.새로운 데이터를 수신했을 때, callback의 함수에 그 데이터를 첫 번째 매개변수로 불러오게 된다.

xycar\_msg = Int32MultiArray()

xycar\_msg를 Int32MultiArray타입으로 초기화한다.

Angle = 0.0

조향값을 0.0으로 고정시킨다.

speed = 25

자동차의 주행 속도를 25로 고정시킨다.

setPoint = 112

fr센서가 값을 유지할 기준이 되는 목표값을 112로 고정시킨다.

error = 0.0

setPoint에서 fr길이를 뺀 오차값을 0.0으로 고정시킨다.

pTerm = 0.0

P제어를 이용해 도출한 결과를 0.0으로 고정시킨다.

iTerm = 0.0

P제어를 이용해 도출한 결과를 0.0으로 고정시킨다.

dTerm = 0.0

P제어를 이용해 도출한 결과를 0.0으로 고정시킨다.

beforeT = 0.0

afterT = 0.0

error값이 새로 갱신되기까지의 시간을 계산하가 위한 변수를 0.0으로 고정시킨다.

while not rospy.is\_shutdown():

사용자가 ctrl+c를 눌러 작동을 멈추기 전까지는 계속 루프를 돌린다는 의미이다. ctrl+c는 노드를 종료시킨다.

xycar\_msg.data = [angle, speed]

motor\_pub.publish(xycar\_msg)

beforeT = time.time()

beforeT는 time타입으로 초기화시킨다.

data = fr

data는 fr변수로

beforeErr = error

error = setPoint - data

error은 setPoint에서 데이터를 뺀 오차값을 0.0으로 고정시킨다.

kp = 0.05

pTerm = kp \* error

ki = 0.0001

if error != beforeErr:

afterT = time.time()

dt = afterT - beforeT

beforeT = afterT

iTerm += ki \* error \* dt

kd = 0.000000001

dError = setPoint - error

if dt != 0:

dTerm = -kd \* (dError / dt)

#pid = pTerm + iTerm + dTerm

pid = pTerm

angle = -pid \* (180.0 / math.pi)

print("fr: %f, error: %f, pTerm: %f, pid: %f, angle: %f"%(fr, error, pTerm, pid, angle))