**임베디드 시스템 설계**

**아두이노와 라즈베리파이를 이용한 라인트레이서 동작**

컴퓨터과학부

2014920056 최주영

2015920005 김대현

**1. 작성코드**

* Serial.ino

#include <SoftwareSerial.h>

#include <AFMotor.h>

AF\_DCMotor motor\_R(3);

AF\_DCMotor motor\_L(4);

String Speed;

char LorR;

int i, s;

char DataToRead[6]; // 라즈베리로부터의 통신 읽어 저장할 배열

void setup() {

// put your setup code here, to run once:

Serial.begin(9600); // 씨리얼!

// 모터 전부 스탑

motor\_L.setSpeed(0);

motor\_L.run(RELEASE);

motor\_R.setSpeed(0);

motor\_R.run(RELEASE);

}

void loop() {

DataToRead[5] = '\n';

// 마지막을 \n으로

Serial.readBytesUntil(char(13), DataToRead, 5);

// 시리얼 통신 읽어들이기

LorR = DataToRead[0]; // 방향 가져오기

Speed = ""; // 스피드 초기화

for (i = 1; (DataToRead[i] != '\n') && (i < 6); i++) {

Speed += DataToRead[i]; // 스피드 값 읽기

}

s = Speed.toInt(); // 스피드 값 정수로 변환

if (LorR == 'L') { // 왼쪽 모터 움직이기

motor\_L.setSpeed(s);

motor\_R.setSpeed(0);

motor\_L.run(FORWARD);

motor\_R.run(FORWARD);

}

else if (LorR == 'R') { // 오른쪽 모터 움직이기

motor\_L.setSpeed(0);

motor\_R.setSpeed(s);

motor\_L.run(FORWARD);

motor\_R.run(FORWARD);

}

else if (LorR == 'F') { // 직진하기

motor\_L.setSpeed(s);

motor\_R.setSpeed(s);

motor\_L.run(FORWARD);

motor\_R.run(FORWARD);

}

else if (LorR == 'A') { // 왼쪽 모터는 뒤로, 오른쪽 모터는 앞으로 ( 오른쪽 모터 앞으로 가기의 극대화)

motor\_L.setSpeed(s);

motor\_R.setSpeed(s);

motor\_L.run(BACKWARD);

motor\_R.run(FORWARD);

}

else if (LorR == 'B') { // 오른쪽 모터는 뒤로, 왼쪽 모터는 앞으로 ( 왼 모터 앞으로 가기의 극대화)

motor\_L.setSpeed(s);

motor\_R.setSpeed(s);

motor\_L.run(FORWARD);

motor\_R.run(BACKWARD);

}

else if (LorR == 'H') { // Hㅜ진

motor\_L.setSpeed(s);

motor\_R.setSpeed(s);

motor\_L.run(BACKWARD);

motor\_R.run(BACKWARD);

}

delay(100); // 통신이 원할하지 않을 때를 고려하여, 분할적으로 delay를 줌

delay(100);

// 0.2초간 움직임

motor\_L.run(RELEASE);

motor\_R.run(RELEASE);

// 이후 멈춤

Serial.println("AAAAAAAAAAAA");

// 모든 행동이 끝난 뒤 시리얼로 문자 보냄.

}

* Main.py

# -\*- coding: utf-8 -\*-

import socket

import sys

import os

import numpy as np

import pdb

import serial

import cv2

import time

import math

from Image import \*

from Utils import \*

from picamera import PiCamera

from time import sleep

def write\_serial(msg):

print("My cmd is %s" % msg)

ser.write(msg)

ser.flush()

cnt = 0

width = 480 # 사진 가로 크기

height = 360 # 사진 세로 크기

try:

ser = serial.Serial('/dev/ttyUSB0',9600) # USB 포트

except: # 0번이 없을 경우

ser = serial.Serial('/dev/ttyUSB1',9600) # USB 포트

while True:

try:

camera = PiCamera() # PiCamera

camera.resolution = (width, height) # 사진 크기 480 \* 360으로 변경

cnt += 1 # 사진 카운트

camera.capture("/home/pi/Desktop/image/route" + str(cnt) + ".jpeg", use\_video\_port = True) # 사진 저장

camera.close() # 카메라 끄기

font = cv2.FONT\_HERSHEY\_SIMPLEX

direction = 0

#N\_SLICES만큼 이미지를 조각내서 Images[] 배열에 담는다

Images=[]

N\_SLICES = 3

for q in range(N\_SLICES):

Images.append(Image())

img = cv2.imread('/home/pi/Desktop/image/route' + str(cnt) + '.jpeg') # 사진 파일 읽어오기

if img is not None:

#이미지를 조각내서 윤곽선을 표시하게 무게중심 점을 얻는다

Points = SlicePart(img, Images, N\_SLICES)

#N\_SLICES 개의 무게중심 점을 x좌표, y좌표끼리 나눈다

x = []

base = 255

print(str(cnt) + "turn")

for i in Points:

x.append(i[0])

print(i[0])

print("\n")

#

if x[0] == None or x[0] < 20 or x[0] > (width-20): # 만약, x[0]가 읽히지 않거나 양쪽 끝에 치우쳐져 있다면

if x[1] == None or x[1] < 20 or x[1] > (width-20): # 만약, x[1]이 읽히지 않거나 양쪽 끝에 치우쳐져 있다면

if x[2] == None or x[2] < 20 or x[2] > (width-20): # 만약, x[2]가 읽히지 않거나 양쪽 끝에 치우쳐져 있다면

cmd = ("H%d\n" % 200).encode('ascii') # 후진

elif x[2] > (width/2): # 만약, x[2]가 오른쪽이라면

cmd = ("B%d\n" % base).encode('ascii') # 극 우회전

else: # 만약, x[2]가 왼쪽이라면

cmd = ("A%d\n" % base).encode('ascii') # 극 좌회전

elif x[1] > (width/2): # 만약, x[1]가 오른쪽이라면

cmd = ("B%d\n" % base).encode('ascii') # 극 우회전

elif x[1] < (width/2): # 만약, x[2]가 왼쪽이라면

cmd = ("A%d\n" % base).encode('ascii') # 극 좌회전

elif x[0] > (width \* (2/3)) and x[1] > (width \* (2/3)): # 만약, x[0]과 x[1]이 모두 오른쪽 1/3부근이라면

cmd = ("L%d\n" % base).encode('ascii') # 우회전

elif x[0] < (width/3) and x[1] < (width/3): # 만약, x[0]과 x[1]이 모두 왼쪽 1/3부근이라면

cmd = ("R%d\n" % base).encode('ascii') # 좌회전

elif abs(x[0] - x[1]) < (50 \* width / 1920): # 만약, x[0] - x[1]의 절대값이. 일정 값 이하라면,

cmd = ("F%d\n" % 200).encode('ascii') # 직진

else:

gradient = 1.0 / (x[0] - x[1]) # gradient 계산

#print(gradient \* 1000)

if gradient > 0: # x[0]이 x[1]보다 크다면

cmd = ("L%d\n" % base).encode('ascii') # 우회전

else: # x[0]이 x[1]보다 작다면

cmd = ("R%d\n" % base).encode('ascii') # 좌회전

write\_serial(cmd) # 시리얼 통신

ser.readLine() # 대기

else:

print('not even processed')

except:

continue

* Image.py
* Threshold값 5로 수정 ( 테스트 환경 기준 )
* Util.py
* 수정사항 없음.

**2. 동작 주요 원리**

3등분 된 이미지에서 경로의 x값을 리스트에 받는다. 위에서부터 x[0], x[1], x[2]이다.

* 아두이노와 라즈베리파이의 시리얼 통신

write\_serial(cmd) # 시리얼 통신

ser.readLine() # 대기

* 라즈베리파이 통신을 한 번 보내고 나면, 아두이노에서 다시 통신을 보낼 때 까지 대기한다.
* 이미지 축소

camera.resolution = (width, height)

* 사이드 처리

if x[0] == None or x[0] < 5 or x[0] > (width-20):

* 이동 방향 계산

1. 화면에서 읽히는게 없다면, 후진한다.
2. x[2]만 읽힌다면, x[2]의 방향에 맞추어 제자리회전한다.
3. x[0]만 읽히지 않는다면, x[1]의 방향에 맞추어 제자리 회전한다.
4. x[0]과 x[1]이 이미지 내에 같은 방향으로 치우쳐 있다면, 방향에 맞추어 회전한다.
5. x[0]과 x[1]이 어느정도 일직선 상에 존재한다면, 직진한다.
6. 그 외의 경우, x[0]과 x[1]의 차이를 고려해서 회전한다.

**3. 실험 고찰**

* 초기, 아두이노에서 입력받은 값을 그대로 라즈베리 파이로 돌려줄 때 제대로 값을 돌려주지 않는 문제가 발생.

아두이노가 라즈베리로부터 받은 값에는 문제가 없기에 동작에는 문제가 없으나, 아두이노에서 라즈베리로 아무런 값도 돌려주지 않을 경우, 라즈베리에서 아두이노로부터의 전송을 기다리며 wait상태에 빠져있는 문제가 발생.

**이에 대해, 어느정도 손실이 발생하더라도 문제가 없도록 긴 문자열을 전송.**

* 이미지 사이즈가 너무 커서 처리 속도 및 처리에 문제 발생.

약간의 이미지 흐림만 있어도 검은색을 제대로 인식하지 못함.

**이에 대해, 이미지 사이즈를 테스트를 통해 재조정**

**(기존 : 1920 x 1080 -> 최종 : 480 x 360)**

* x[0]과 x[1]의 차가 일정 값 이하면 직진하도록 하였으나, 양쪽 사이드에 몰려있는 경우에 해당 방향으로 회전해야 함에도 직진하는 문제 발생.

**이에 대해, 사이드에 있는 경우는 예외처리 하여 해당 방향으로 회전하도록 변경.**

* 카메라의 그림자가 카메라에 인식되어서 경로 계산에 문제가 생김.

이에 대해, Threshold값을 낮췄으나, 빛의 양에 따라 경로를 인식하지 못하는 문제가 발생.

**이에 대해, 카메라를 아예 종이를 통해 가려서 카메라의 그림자로 인한 문제를 없앰.**

* 경로를 벗어나서 카메라에 경로가 읽히지 않는 경우, 로봇의 동작이 이상해지는 문제 발생.

**이에 대해, 읽히는 경로가 없는 경우 후진하도록 함.**