국민대학교
소프트웨어융합대학
소프트웨어학부

창업연계공학설계

프로젝트 명	창업연계공학설계
팀 명	*9*
문서 제목	보고서

Version	2.0
Date	29

팀원	김세연 (조장) / 20135180
	이윤서 / 20171673
	이원주 / 20171671
지도교수	김상철 교수

문서 정보 / 수정 내역

Filename	미로찾기.doc

국민대학교
소프트웨어학부

창업연계공학설계 보고서

미로찾기

9조

Confidential Restricted Version 1.2
2009-Apr-25

원안작성자	김세연, 이원주, 이윤서
수정작업자	김세연, 이원주, 이윤서

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2017-12-1	이윤서	1.0	최초 작성	
2017-12-2	김세연	1.1	내용 수정	
2017-12-4	이윤서	1.2	내용 수정	향후 추진 계획 수정
2017-12-5	이원주	1.3	코드 수정	변경된 코드 내용 수정
2017-12-8	이원주	1.4	내용 수정	
2017-12-10	김세연	2.0	내용 수정 코드 변경	코드 변경, 내용 수정

목 차

미로찾기

9조

Confidential Restricted Version 1.2
2009-Apr-25

1. 서론

라즈베리카가 미로를 통과해 출발점에서 도착점까지 이동할 수 있게 한다.

미로를 통과하는 여러가지 방법중, 오른쪽을 우선으로 주행하는 우수법 알고리즘을 이용해 라즈베리카가 미로를 의도한 방법으로 통과할 수 있게 했다.

우수법의 특성상, 좌회전, 우회전을 언제 어느 조건에 수행해야 하는 지를 판단해야 한다. 이런 문제를 해결하기 위해 5방향추적센서의 양측 각각 1개의 센서를 통해 좌회전, 우회전 상황을 구별하도록 했다. 또한 구동체의 무개가 주행속도에 큰 영향을 주므로 구동체의 무개를 최대한 줄여 효율적인 주행을 가능하게 했다.

2. 기본 아이디어

라인트레이싱을 할 때 발생할 수 있는 경우를 고려하여 미로통과를 안정적으로 완주 할 수 있게 만들도록 한다. 각 경우(전진,회전 등)는 5방향추적센서가 라인을 인식할 때의 조건으로 나누어 구현하며, 각 모듈들이 조건을 가지고 구동체의 움직임을 다룬다.

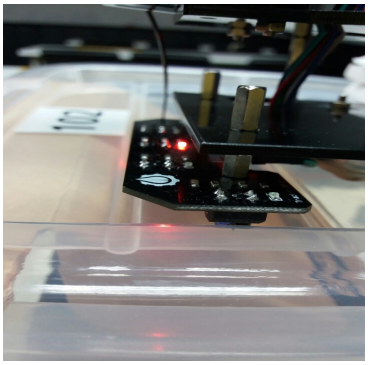
전진,회전 경우를 구분하기 위해5방향추적센서를 사용하는데, 각 센서는 D B A C E로 부른다. 그리고 센서가 라인 위에 있는 경우 off(0), 반대의 경우에는 on(1)한다.

D LED는 좌회전 교차로 판별에 사용하며 E LED는 우회전 교차로 판별에 사용하고 나머지 B A C LED는 라인트레이싱에 사용한다.

2.1 H/W 디자인 방향

후륜 모터가 구동체의 기본적인 전후진과 진행방향을 변경하도록 했다. 후륜 모터의 속도와 파워를 변경하는 방법으로 진행방향을 설정했다.

적외선 5방향 트래킹 센서가 바닥의 라인을 인식해 라인을 따라 우수법을 시행하도록 했다. 이번 과제에는 진행경로에 장애물이 있지 않았으므로 초음파센서는 사용하지 않았다.

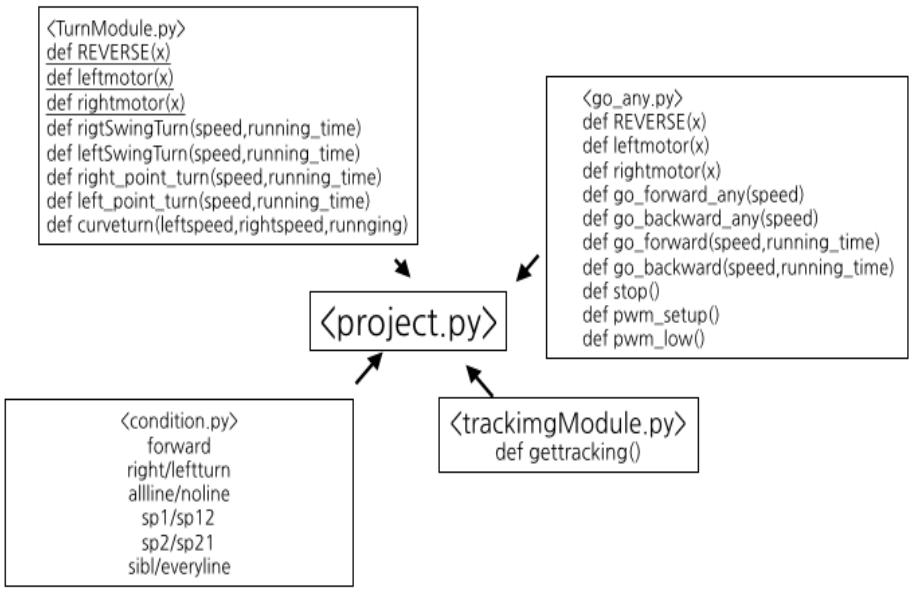


2.2 S/W 디자인 방향

우수법 알고리즘을 적용하여 라인트레이싱을 하다 교차로가 나타나는 경우, 오른쪽을 우선순위로 진행하도록 했다. 만약 더 이상 갈 곳이 없을 경우, 유턴을 하도록 했다.

Go_any.py에서 기본적인 전,후진을, TurnModule.py에서 기본적인 회전을 구현했고 condition.py에서 좌,우회전 조건을 정리했다. 또 trackingModule에서 5방향추적센서의 인식값을 리스트로 반환해 주행을 구현하는데 효율성을 높혔다. 이런 방법으로 각 기능을 모듈화 시켜 encapsulation을 추구했다.

<코드 설계도>



3. 수행 내용

3.1.프로그램 코드

<TurnModule.py>

```
# import GPIO librery
import RPi.GPIO as GPIO
import time

# set up GPIO mode as BOARD
GPIO.setmode(GPIO.BOARD)

# set GPIO warnings as flase
GPIO.setwarnings(False)
```

```
# ========================================================================
# REVERSE function to control the direction of motor in reverse
def REVERSE(x):
  if x == True:
    return False
  elif x == False:
    return True
# ========================================================================


# ========================================================================
# Set the motor's true / false value to go forward.
forward0 = True
forward1 = False
# ========================================================================

# ========================================================================
#Set the motor's true / false value to go opposite.
backward0 = REVERSE(forward0)
backward1 = REVERSE(forward1)
# ========================================================================

# ========================================================================
# declare the pins of 12, 11, 35 in the Rapberry Pi
# as the left motor control pins in order to control left motor
# left motor needs three pins to be controlled
# ========================================================================
```

```
MotorLeft_A = 12
MotorLeft_B = 11
MotorLeft_PWM = 35

# ============================================================================
# declare the pins of 15, 13, 37 in the Rapberry Pi
# as the right motor control pins in order to control right motor
# right motor needs three pins to be controlled
# ============================================================================
MotorRight_A = 15
MotorRight_B = 13
MotorRight_PWM = 37

# ============================================================================
# Control the DC motor to make it rotate clockwise, so the car will
# move forward.
# if you have different direction, you need to change HIGH to LOW
# or LOW to HIGH,in MotorLeft_A
# and LOW to HIGH or HIGH to LOW in MotorLeft_B
# if you have different direction, you need to change HIGH to LOW
# or LOW to HIGH in MotorLeft_A
# and LOW to HIGH or HIGH to LOW in MotorLeft_B
# ============================================================================

def leftmotor(x):
  if x == True:
    GPIO.output(MotorLeft_A, GPIO.HIGH)
    GPIO.output(MotorLeft_B, GPIO.LOW)
```

```
  elif x == False:
    GPIO.output(MotorLeft_A, GPIO.LOW)
    GPIO.output(MotorLeft_B, GPIO.HIGH)
  else:
    print 'Config Error'
def rightmotor(x):
  if x == True:
    GPIO.output(MotorRight_A, GPIO.LOW)
    GPIO.output(MotorRight_B, GPIO.HIGH)
  elif x == False:
    GPIO.output(MotorRight_A, GPIO.HIGH)
    GPIO.output(MotorRight_B, GPIO.LOW)
# ================================================================================
# because the connetions between motors (left motor) and Rapberry Pi has been
# established, the GPIO pins of Rapberry Pi
# such as MotorLeft_A, MotorLeft_B, and MotorLeft_PWM
# should be clearly declared whether their roles of pins
# are output pin or input pin
# ================================================================================


GPIO.setup(MotorLeft_A,GPIO.OUT)
GPIO.setup(MotorLeft_B,GPIO.OUT)
GPIO.setup(MotorLeft_PWM,GPIO.OUT)


# ================================================================================
# because the connetions between motors (right motor) and Rapberry Pi has been
```

```
# established, the GPIO pins of Rapberry Pi
# such as MotorLeft_A, MotorLeft_B, and MotorLeft_PWM
# should be clearly declared whether their roles of pins
# are output pin or input pin
# ======================================================================


GPIO.setup(MotorRight_A,GPIO.OUT)
GPIO.setup(MotorRight_B,GPIO.OUT)
GPIO.setup(MotorRight_PWM,GPIO.OUT)


# ======================================================================
# create left pwm object to control the speed of left motor
# ======================================================================
LeftPwm=GPIO.PWM(MotorLeft_PWM,100)


# ======================================================================
# create right pwm object to control the speed of right motor
# ======================================================================
RightPwm=GPIO.PWM(MotorRight_PWM,100)


# ======================================================================
# perform right swing turn of 90 degree
# ======================================================================
def RightSwingTurn(speed, running_time):
```

```
    # set the left motor to go fowrard
    leftmotor(forward0)
    #leftmotor(forward1)


    # set the left motor pwm to be ready to go forward
    GPIO.output(MotorLeft_PWM,GPIO.HIGH)


    # set the right motor pwm to be ready to stop
    # Turn Off Right PWM
    GPIO.output(MotorRight_PWM,GPIO.LOW)
    # set the speed of the left motor to go fowrard
    LeftPwm.ChangeDutyCycle(speed)
    # set the speed of the right motor to stop
    RightPwm.ChangeDutyCycle(0)
    # set the running time of the left motor to go fowrard
    time.sleep(running_time)


# ======================================================================
# perform left swing turn of 90 degree
# ======================================================================
def leftSwingTurn(speed, running_time):


    # set the left motor pwm to be ready to stop
    # Turn Off Left PWM
    GPIO.output(MotorLeft_PWM,GPIO.LOW)
```

```
    # set the right motor to go fowrard
    rightmotor(forward0)


    # set the right motor pwm to be ready to go forward
    GPIO.output(MotorRight_PWM,GPIO.HIGH)


    # set the speed of the left motor to stop
    LeftPwm.ChangeDutyCycle(0)
    # set the speed of the right motor to go fowrard
    RightPwm.ChangeDutyCycle(speed)
    # set the running time of the right motor to go fowrard
    time.sleep(running_time)



# ======================================================================
# perform right point turn of 90 degree  # student assignment (1)
# ======================================================================


def right_point_turn(speed, running_time):  # student assignment (1)
# set the left motor to go fowrard
   leftmotor(forward0)
    #leftmotor(forward1)
```

```
  # set the left motor pwm to be ready to go forward
  GPIO.output(MotorLeft_PWM,GPIO.HIGH)

  # set the right motor pwm to be ready to stop
  rightmotor(backward0)
  # Turn Off Right PWM
  GPIO.output(MotorRight_PWM,GPIO.HIGH)
  # set the speed of the left motor to go fowrard
  LeftPwm.ChangeDutyCycle(speed)
  # set the speed of the right motor to stop
  RightPwm.ChangeDutyCycle(speed)
  # set the running time of the left motor to go fowrard
  time.sleep(running_time)

#=====================================================================
# perform left point turn of 90 degree   # student assignment (2)
# =====================================================================


def left_point_turn(speed, running_time):  # student assignment (2)
# set the left motor to go fowrard
  leftmotor(backward0)
  #leftmotor(forward1)


  # set the left motor pwm to be ready to go forward
  GPIO.output(MotorLeft_PWM,GPIO.HIGH)
```

```
    # set the right motor pwm to be ready to stop
    rightmotor(forward0)
    # Turn Off Right PWM
    GPIO.output(MotorRight_PWM,GPIO.HIGH)
    # set the speed of the left motor to go fowrard
    LeftPwm.ChangeDutyCycle(speed)
    # set the speed of the right motor to stop
    RightPwm.ChangeDutyCycle(speed)
    # set the running time of the left motor to go fowrard
    time.sleep(running_time)
def curveturn(leftspeed, rightspeed, running):
    leftmotor(forward0)
    GPIO.output(MotorLeft_PWM,GPIO.HIGH)
    rightmotor(forward0)
    GPIO.output(MotorRight_PWM,GPIO.HIGH)


    LeftPwm.ChangeDutyCycle(leftspeed)
    RightPwm.ChangeDutyCycle(rightspeed)


    time.sleep(running)
```

<condition2.py>

```
forward= [
    [1,1,0,1,1],
    [1,0,0,0,1],
    #[1,1,0,0,1],
]
```

```
###
rightturn = [
    [1,1,0,0,0],
    #[1,0,0,0,0],
    [1,0,0,0,0],
]
###
noline = [
    [1,1,1,1,1],
    [1,1,1,1,0],
    [0,1,1,1,1],
]
allline = [
    [0,0,0,0,0],
]
leftturn = [
    [0,0,0,1,1],
    [0,0,1,1,1],
    [0,0,0,0,1],
]
sp1 = [
            [1,0,0,1,1],
            [1,0,1,1,1],
    ]
sp12 = [
    [0,0,1,1,1],
]
sp2 = [
    [1,1,0,0,1],
```

```
    [1,1,1,0,1],
    ]
sp21 = [
    [1,1,1,0,0],
]
everyline = [
        [1,1,0,1,1],
        [1,0,0,0,1],
        [1,0,0,1,1],
        [1,1,0,0,1],
        [1,1,1,0,0],
        [0,0,1,1,1],
        [1,1,1,0,1],
        [1,0,1,1,1],
]
sibl = [1,0,0,0,1]
```

<go_any.py>

```
# import GPIO library

import RPi.GPIO as GPIO

from time import sleep

# set GPIO warnings as flase
GPIO.setwarnings(False)

# set up GPIO mode as BOARD
GPIO.setmode(GPIO.BOARD)

#
==========================================================================
===

# REVERSE function to control the direction of motor in reverse
```

```
#
=============================================================================
===

def REVERSE(x):

    if x == True:

        return False

    elif x == False:

        return True

#
=============================================================================
===
# Set the motor's true / false value to go forward.
#
=============================================================================
===

forward0 = True

forward1 = False

#
=============================================================================
===

# Set the motor's true / false value to go opposite.

#
=============================================================================
===

backward0 = REVERSE(forward0)

backward1 = REVERSE(forward1)

#
=============================================================================
===

# declare the pins of 12, 11, 35 in the Rapberry Pi

# as the left motor control pins in order to control left motor
```

```
# left motor needs three pins to be controlled
#
================================================================

MotorLeft_A = 12

MotorLeft_B = 11

MotorLeft_PWM = 35

#
================================================================
===

# declare the pins of 15, 13, 37 in the Rapberry Pi

# as the right motor control pins in order to control right motor

# right motor needs three pins to be controlled

#
================================================================
===

MotorRight_A = 15

MotorRight_B = 13

MotorRight_PWM = 37

#
================================================================
=======

# Control the DC motor to make it rotate clockwise, so the car will

# move forward.

# if you have different direction, you need to change HIGH to LOW

# or LOW to HIGH,in MotorLeft_A

# and LOW to HIGH or HIGH to LOW in MotorLeft_B

# if you have different direction, you need to change HIGH to LOW

# or LOW to HIGH in MotorLeft_A
```

```
# and LOW to HIGH or HIGH to LOW in MotorLeft_B
#
================================================================================

def leftmotor(x):
    if x == True:
        GPIO.output(MotorLeft_A, GPIO.HIGH)
        GPIO.output(MotorLeft_B, GPIO.LOW)
    elif x == False:
        GPIO.output(MotorLeft_A, GPIO.LOW)
        GPIO.output(MotorLeft_B, GPIO.HIGH)
    else:
        print
        'Config Error'
def rightmotor(x):
    if x == True:
        GPIO.output(MotorLeft_A, GPIO.LOW)
        GPIO.output(MotorLeft_B, GPIO.HIGH)
    elif x == False:
        GPIO.output(MotorLeft_A, GPIO.HIGH)
        GPIO.output(MotorLeft_B, GPIO.LOW)
    else:
        print
        'Config Error'
```

```
# student assignment (3)
#
=============================================================================

# because the connetions between motors (left motor) and Rapberry Pi has been
# established, the GPIO pins of Rapberry Pi
# such as MotorLeft_A, MotorLeft_B, and MotorLeft_PWM
# should be clearly declared whether their roles of pins
# are output pin or input pin
#
=============================================================================

GPIO.setup(MotorLeft_A, GPIO.OUT)
GPIO.setup(MotorLeft_B, GPIO.OUT)
GPIO.setup(MotorLeft_PWM, GPIO.OUT)

#
=============================================================================

# because the connetions between motors (right motor) and Rapberry Pi has been
# established, the GPIO pins of Rapberry Pi
# such as MotorLeft_A, MotorLeft_B, and MotorLeft_PWM
# should be clearly declared whether their roles of pins
# are output pin or input pin
#
=============================================================================

GPIO.setup(MotorRight_A, GPIO.OUT)
```

```
GPIO.setup(MotorRight_B, GPIO.OUT)

GPIO.setup(MotorRight_PWM, GPIO.OUT)

#
==========================================================================
===

# create left pwm object to control the speed of left motor

#
==========================================================================
===

LeftPwm = GPIO.PWM(MotorLeft_PWM, 100)


#
==========================================================================
===

# create right pwm object to control the speed of right motor

#
==========================================================================
===

RightPwm = GPIO.PWM(MotorRight_PWM, 100)

#
==========================================================================
===

#  go_forward_any method has been generated for the three-wheeled moving

#  objec to go forward without any limitation of running_time

#
==========================================================================
===



def go_forward_any(speed):

   leftmotor(forward0)
```

```
    leftmotor(forward1)
    GPIO.output(MotorLeft_PWM,GPIO.HIGH)
    rightmotor(forward0)
    rightmotor(forward1)
    GPIO.output(MotorRight_PWM,GPIO.HIGH)
    LeftPwm.ChangeDutyCycle(speed)
    RightPwm.ChangeDutyCycle(speed)
# student assignment (4)
#
=============================================================================
#  go_backward_any method has been generated for the three-wheeled moving
#  objec to go backward without any limitation of running_time
#
=============================================================================


def go_backward_any(speed):


    leftmotor(backward0)
    leftmotor(backward1)
    GPIO.output(MotorLeft_PWM,GPIO.HIGH)

    rightmotor(backward0)
    rightmotor(backward1)
```

```
    GPIO.output(MotorRight_PWM,GPIO.HIGH)
    LeftPwm.ChangeDutyCycle(speed)
    RightPwm.ChangeDutyCycle(speed)
# student assignment (5)
#
================================================================
===
#  go_forward_any method has been generated for the three-wheeled moving
#  objec to go forward with the limitation of running_time
#
================================================================
===
def go_forward(speed, running_time):
    leftmotor(forward0)
    leftmotor(forward1)
    GPIO.output(MotorLeft_PWM,GPIO.HIGH)
    rightmotor(forward0)
    rightmotor(forward1)

    GPIO.output(MotorRight_PWM,GPIO.HIGH)
    LeftPwm.ChangeDutyCycle(speed)
    RightPwm.ChangeDutyCycle(speed)
    sleep(running_time)
# student assignment (6)
#
================================================================
===
#  go_backward_any method has been generated for the three-wheeled moving
```

```
#  objec to go backward with the limitation of running_time

#
================================================================================

def go_backward(speed, running_time):

    leftmotor(backward0)

    leftmotor(backward1)

    GPIO.output(MotorLeft_PWM,GPIO.HIGH)

    rightmotor(backward0)

    rightmotor(backward1)

    GPIO.output(MotorRight_PWM,GPIO.HIGH)
    LeftPwm.ChangeDutyCycle(speed)
    RightPwm.ChangeDutyCycle(speed)
    sleep(running_time)

# student assignment (7)
#
================================================================================
# define the stop module
#
================================================================================

def stop():

    # the speed of left motor will be set as LOW

    GPIO.output(MotorLeft_PWM, GPIO.LOW)

    # the speed of right motor will be set as LOW

    GPIO.output(MotorRight_PWM, GPIO.LOW)

    # left motor will be stopped with function of ChangeDutyCycle(0)

    LeftPwm.ChangeDutyCycle(0)

    # left motor will be stopped with function of ChangeDutyCycle(0)
```

```
    RightPwm.ChangeDutyCycle(0)

def pwm_setup():
    LeftPwm.start(0)
    RightPwm.start(0)



def pwm_low():
    GPIO.output(MotorLeft_PWM, GPIO.LOW)
    GPIO.output(MotorRight_PWM, GPIO.LOW)
    LeftPwm.ChangeDutyCycle(0)
    RightPwm.ChangeDutyCycle(0)
    GPIO.cleanup()
```

<project.py>

```
# ================================================================

# import GPIO library and time module

# ================================================================

import RPi.GPIO as GPIO

from time import sleep
# ================================================================

#  set GPIO warnings as false

# ================================================================

GPIO.setwarnings(False)

# ================================================================
```

```
# import TurnModule() method

# ====================================================================

from TurnModule import *
from trackingModule import gettracking

# ====================================================================

# rightPointTurn() and leftPointTurn() in TurnModule module

# ====================================================================

# student assignment (1)
# student assignment (2)

# ====================================================================

# import go_forward_any(), go_backward_any(), stop(), LeftPwm(),
# RightPwm(), pwm_setup(), and pwm_low() methods in the module of go_any

# ====================================================================

from go_any import *

from condition2 import *

# implement rightmotor(x)  # student assignment (3)
# implement go_forward_any(speed): # student assignment (4)
# implement go_backward_any(speed): # student assignment (5)
# implement go_forward(speed, running_time)  # student assignment (6)
# implement go_backward(speed, running_time)  # student assignment (7)
# ====================================================================

# setup and initilaize the left motor and right motor

# ====================================================================

pwm_setup()


# ====================================================================

#  define your variables and find out each value of variables

#  to perform the project3 with ultra sensor

#  and swing turn

# ====================================================================

leftSwingTurn(1,0.01)
```

```
con = 0
print('motorInitialize')
try:
    while True:
        # ultra sensor replies the distance back
        ledcondition = gettracking()
        print(ledcondition)
            if ledcondition == sibl:
                stop()
                sleep(2)
                #go_forward(15,0.1)
        if ledcondition in forward:
            go_forward(30,0.3)
        elif ledcondition in sp1:
            curveturn(20,50,0.1)
            elif ledcondition in sp12:
                curveturn(20,50,0.1)
        elif ledcondition in sp2:
            curveturn(50,20,0.1)
            elif ledcondition in sp21:
                curveturn(50,20,0.1)
            elif ledcondition in leftturn:
                go_forward(50,0.5)
                con =1
        if ledcondition in rightturn:
            print("rightturn forward")
            go_forward(50,0.5)
            stop()
            print("RIGHTTURN")
```

```
    stop()
    sleep(1)
    right_point_turn(40,0.5)
    stop()
    sleep(1)
    con =0
if ledcondition in allline:
        con =0
        stop()
    print(ledcondition)
    print('all line')
        sleep(0.5)
        RightSwingTurn(50,0.3)
        stop()
    sleep(0.5)
    if ledcondition in everyline:
                print(ledcondition)
            print("right condition please right turn")
       right_point_turn(40,1)
        else:
                print("there is no line")
                con = 0
if ledcondition in noline:
    if con == 0:
       print("no line")
       right_point_turn(40,0.15)
       stop()
```

```
            sleep(0.1)
        elif con ==1:
            print("no line 2 ")
            left_point_turn(40,0.15)
            stop()
            sleep(0.1)
    # when the Ctrl+C key has been pressed,
    # the moving object will be stopped
except KeyboardInterrupt:
    pwm_low()
```

<trackingModule.py>

```
import RPi.GPIO as GPIO
import time
def gettracking():
# ======================================================================
#  set GPIO warnings as false
# ======================================================================
    GPIO.setwarnings(False)
# ======================================================================
# set up GPIO mode as BOARD
# ======================================================================
    GPIO.setmode(GPIO.BOARD)
# ======================================================================
# declare the pins of 16, 18, 22, 40, 32 in the Rapberry Pi
# as the control pins of 5-way trackinmg sensor in order to
# control direction
#
```

```
# leftmostled    leftlessled    centerled    rightlessled    rightmostled
#      16          18           22           40           32
#
# led turns on (1) : trackinmg sensor led detects white playground
# led turns off(0) : trackinmg sensor led detects black line
# leftmostled off : it means that moving object finds black line
#                   at the position of leftmostled
#                   black line locates below the leftmostled of the moving object
#
# leftlessled off : it means that moving object finds black line
#                   at the position of leftlessled
#                   black line locates below the leftlessled of the moving object
#
# centerled off : it means that moving object finds black line
#                   at the position of centerled
#                   black line locates below the centerled of the moving object
#
# rightlessled off : it means that moving object finds black line
#                   at the position of rightlessled
#                   black line locates below the rightlessled  of the moving object
#
# rightmostled off : it means that moving object finds black line
#                   at the position of rightmostled
#                   black line locates below the rightmostled of the moving object
# ========================================================================
  leftmostled=16
  leftlessled=18
```

```
   centerled=22
   rightlessled=40
   rightmostled=32
# ========================================================================
# because the connetions between 5-way tracking sensor and Rapberry Pi has been
# established, the GPIO pins of Rapberry Pi
# such as leftmostled, leftlessled, centerled, rightlessled, and rightmostled
# should be clearly declared whether their roles of pins
# are output pin or input pin
# since the 5-way tracking sensor data has been detected and
# used as the input data, leftmostled, leftlessled, centerled, rightlessled, and rightmostled
# should be clearly declared as input
#
# ========================================================================
   GPIO.setup(leftmostled, GPIO.IN)
   GPIO.setup(leftlessled, GPIO.IN)
   GPIO.setup(centerled,   GPIO.IN)
   GPIO.setup(rightlessled, GPIO.IN)
   GPIO.setup(rightmostled, GPIO.IN)
# ========================================================================
# GPIO.input(leftmostled) method gives the data obtained from leftmostled
# leftmostled returns (1) : leftmostled detects white playground
# leftmostled returns (0) : leftmostled detects black line
#
#
# GPIO.input(leftlessled) method gives the data obtained from leftlessled
# leftlessled returns (1) : leftlessled detects white playground
```

```
# leftlessled returns (0) : leftlessled detects black line
#
# GPIO.input(centerled) method gives the data obtained from centerled
# centerled returns (1) : centerled detects white playground
# centerled returns (0) : centerled detects black line
#
# GPIO.input(rightlessled) method gives the data obtained from rightlessled
# rightlessled returns (1) : rightlessled detects white playground
# rightlessled returns (0) : rightlessled detects black line
#
# GPIO.input(rightmostled) method gives the data obtained from rightmostled
# rightmostled returns (1) : rightmostled detects white playground
# rightmostled returns (0) : rightmostled detects black line
#
# ======================================================================
  try:
      ledcondition = [GPIO.input(leftmostled),
                      GPIO.input(leftlessled),
                      GPIO.input(centerled),
                      GPIO.input(rightlessled),
                      GPIO.input(rightmostled),
                      ]
      return ledcondition
      time.sleep(0.02)

  except KeyboardInterrupt:
```

GPIO.cleanup()

3.2.코드 설명

<TurnModule.py>

한쪽모터만 이용한 회전방법인leftSwingTurn, rightSwingTurn과 양쪽모터의 구동 방향을 달리한 방법인 leftPointTurn, rightPointTurn을 구현했다.
좌우 구분 없이 양쪽 모터가 돌아가는 속도만 다르게 해 회전한다는 개념을 이용해 CurveTurn을 구현했다.

<project.py>

우수법을 기반으로 프로그래밍 한 이번 과제의 메인 모듈이다. 구동체가 움직이는 동안에 주기적으로 TrackingModule에서 5방향추적센서에서 리턴값을 받았다.
TrackingModule의 리턴값과 condition2 모듈의 라인트래이싱 중 발생할 수 있는 경우들을 비교해 조건문을 이용해 전진,회전을 하도록 했다.

<condition2.py>

TrackignMoudle의 리턴값 형식(list) 으로 라인트레이싱중 발생할 수 있는 경우들을 정리한 모듈이다. E (A B C) D 센서가 라인을 가운데에 두고 신호가 들어오는(1) 경우 forward한다.
rightturn, leftturn의 경우 신호가 우측,좌측에 들어올 경우를 조건으로 두었다. 각각 우회전과 좌회전 발생하는 경우 E나 D의 값은 0이어야 하는게 필수조건이다.
또한 구동체가 라인을 찾아 잘 따라갈 수 있도록 sp1,sp12,sp2,sp21 조건들을 만들어 주었다.

<go_any.py>

구동체의 기본적인 전진,후진을 담당하는 모듈이다.
일정 시간 동안 직진하는 함수와 정해진 시간 없이 직진하는 함수,
일정 시간 동안 후진하는 함수와 정해진 시간 없이 후진하는 함수,
그리고 구동체를 정지하는 함수가 구현되어 있다.

<trackingModule.py>

5방향추적센서의 출력을 리스트형식으로 반환하도록 한 모듈이다. TrackingModule의 리턴값을 기반으로 라인트레이싱을 하도록 한다.

4. 향후 추진계획

4.1.향후 계획의 세부 내용

여러 번의 모의주행결과, 후륜 모터의 파워가 약해 구동체의 속도가 구동체의 무개에도 영향을 크게 받는다는 사실을 알게 되었다. 구동체의 무개를 줄이기 위해 이번 과제에선 사용하지 않는 초음파센서를 탈착한 뒤 주행해보겠다.

개발 도중 10자 교차로와 T자 교차로를 구분하지 못하는 문제점이 있었다. 따라서 교차로를 만났을 때 구동체를 약간 직진 시킨 후 검은색 라인이 있으면 10자 없으면 T자로 판단하도록 설계했다.

5. 애로 및 건의사항

5방향추적센서의 모양이 완벽히 일자가 아니기 때문에 프로그래밍에 난항을 겪었다. 예를 들어 모든 센서가 라인 위에 있는 상황을 다룰 때, 가운데 3개의 센서가 측면의 센서2개보다 약간 앞에 위치해 [0,0,0,0,0] 값이 아닌 [1,0,0,0,1]로 인식할 때가 자주 있다. 이런 부분까지 고려해 프로그래밍 하기가 너무 어려웠다.