


창업연계공학설계입문

| | |
|--------|-----------|
| 프로젝트 명 | 미로 찾기 |
| 팀 명 | 8조였나 9조였나 |
| 문서 제목 | 미로 찾기 보고서 |

| | |
|---------|------------|
| Version | 1.2 |
| Date | 2017-DEC-8 |

| | |
|------|---------|
| 팀원 | 노성환 |
| | 문성찬 |
| | 박건후 |
| 지도교수 | 한광수 교수님 |
| 분반 | 2 분반 |

| | | | |
|---------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  국민대학교 소프트웨어학부 창업연계공학설계입문 | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |


| CONFIDENTIALITY/SECURITY WARNING |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 공학설계입문 수강 학생 중 프로젝트 "미로 찾기"를 수행하는 팀 "8조였나 9조였나"의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 "8조였나 9조였나"의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.</p> |

문서 정보 / 수정 내역

| | |
|-----------------|---------------|
| Filename | 미로 찾기 보고서.pdf |
| 원안작성자 | 노성환, 문성찬, 박건후 |
| 수정작성자 | 노성환, 문성찬, 박건후 |


| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
|-----------|-------|----------|----------|----------------------|
| 2017-12-7 | 노성환 | 1.0 | 최초 작성 | |
| 2017-12-8 | 문성찬 | 1.1 | 내용 수정 | 내용 검토/정리 겸 수정된 내용 추가 |
| 2017-12-9 | 박건후 | 1.2 | 내용 추가 | 회의록 내용 추가 및 내용 정리 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

본 양식은 소프트웨어학부 공학설계입문 과목의 프로젝트 중간보고서 작성을 위한 기본 양식입니다. 문서의 필수 항목을 제시하는 것이니 폰트, 문단 구조 등의 디자인 부분은 자유롭게 설정하기 바랍니다. 양식 내에 붉은 색으로 기술한 부분은 지우고 작성하기 바랍니다.

| | | | |
|---------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  국민대학교 소프트웨어학부 창업연계공학설계입문 | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

목 차

| | | |
|-----|--------------------|----|
| 1 | 서론 | 4 |
| 2 | 기본 아이디어 | 5 |
| 2.1 | H/W 디자인 방향 | 5 |
| 2.2 | S/W 디자인 방향 | 6 |
| 3 | 수행 내용 | 7 |
| 3.1 | 프로그램 코드 | 7 |
| 4 | 향후 추진계획 | 15 |
| 4.1 | 향후 계획의 세부 내용 | 15 |
| 5 | 애로 및 건의사항 | 16 |
| 6 | 회의록 | 17 |

| | | | |
|---------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  국민대학교 소프트웨어학부 창업연계공학설계입문 | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

1 서론

1) 프로젝트 목표 (과제 개요)


- 1-1 5방향 센서의 입력 값을 통해 흰색 아크릴 판에 있는 검은색 테이프 라인을 따라간다.
- 1-2 오른쪽 법칙을 이용한 주행이므로, 주행 도중 오른쪽으로 가는 길이 존재하면 오른쪽으로 간다.
- 1-3 오른쪽으로 갔던 길이 정상 길이 아니면 유턴을 통해 다시 빠져나온다.
- 1-4 오른쪽 법칙을 이용한 주행이므로, 왼쪽과 오른쪽 갈림길이 나올 시, 무조건 오른쪽으로 간다.
- 1-5 오른쪽과 유턴도 아닌 갈림길이 나온다면 왼쪽으로 간다.
- 1-6 위 조건들을 모두 충족시키며 미로를 탈출한다.

2) 고려해야 할 기능/모듈

- 2-1 구동체의 모터의 움직임과 속도를 설정할 모듈
- 2-2 구동체의 기본적인 움직임(직진, 회전)을 담당할 모듈
- 2-3 라인 인식 기능을 담당할 모듈
- 2-4 미로 찾기 때, 필요로 하는 움직임들을 담당할 모듈

3) 고려해야 할 요소들

- 3-1 배터리의 충전 상태
 - 100% / 100% 이후 30분 경과
- 3-2 자동차의 무게
 - LCE 모니터 동행 유무 / 초음파 센서 카메라 동행 유무
- 3-3 바닥의 상태
 - 아크릴판의 매끄러움 정도
- 3-4 방향 센서의 감지
 - 반응성 수치: 0.01 / 0.001 / 또는 그 이하의 값

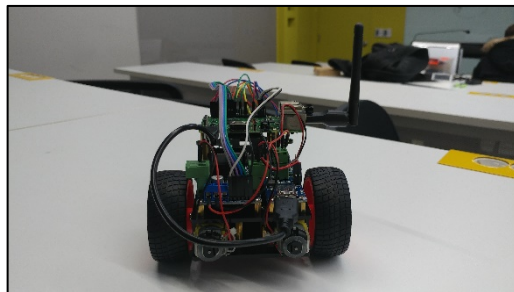
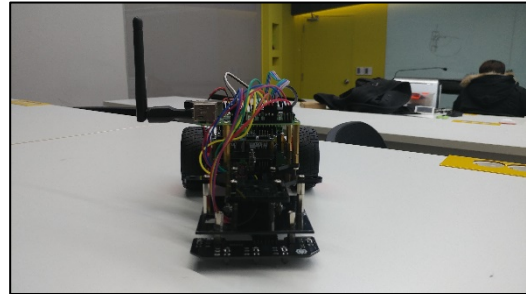
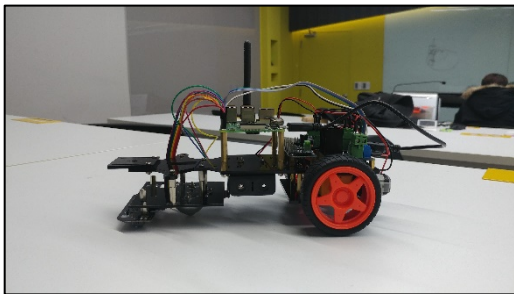
| | | | |
|---------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  국민대학교 소프트웨어학부 창업연계공학설계입문 | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |


2 기본 아이디어

- 구동체가 검은색 라인을 따라가게끔 하면서, 오른쪽 갈림길이 나올 시, 오른쪽으로 가게 한다. 두 갈림길이 나올 시, 마찬가지로 오른쪽으로 가게 하고 한다. 주행하는 길이 정상적인 길이 아니라면 유턴을 통해 빠져나오게 하고, 왼쪽 갈림길만 존재 시 왼쪽으로 가게 한다.

2.1 H/W 디자인 방향


- 초음파 센서 카메라가 이번 과제에서 필요 없다는 것을 알았고, 초음파 센서 카메라를 제거해 줌으로써 구동체의 무게를 줄여, 움직일 때 더욱 간결하고 가볍도록 만들었다.
- 라인을 원활히 따라가고, 턴할 때의 모터 쪽 바퀴의 움직임을 정확히 하기 위해, 뒷바퀴를 완전히 몸통 부분에 밀착시켜 주는 것이 아닌 몸통 사이의 공간을 만듦으로써, 뒷바퀴와 몸통과의 마찰력이 없게끔 디자인하였다.
- 구동체의 라인을 인식하는 센서들의 선이 앞바퀴와의 접촉이 없도록 정리하였다.



| | | | |
|---------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  국민대학교 소프트웨어학부 창업연계공학설계입문 | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

2.2 S/W 디자인 방향

- 위의 서론에서 언급한 대로, 고려해야 할 기능/모듈을 토대로 미로 찾기의 핵심 코드인 Main코드와 함께 4개의 모듈들을 구현한다.
- **Setup 모듈:**
구동체를 움직이기 위해 뒷바퀴의 모터에 대한 설정을 해주는 모듈로, GPIO 라이브러리를 받아오며, 구동체의 뒷바퀴의 전진과 후진을 변수를 이용해 정해주며, 뒷바퀴 모터의 속도 제어도 담당하는 모듈이다.
기본적으로 구동체의 모든 움직임을 위해 내재되어야 하는 기본 설정들을 포함하는 모듈이다.
- **Movement 모듈:**
이번 미로 찾기 과제에서 필요로 하는 기본적인 움직임을 담당하는 모듈로, 라인 트레이싱뿐만 아니라 우측 턴, 좌측 턴, 직진 동작의 움직임들을 설정하는 모듈이다. 그리고 정확한 회전을 위해 스윙턴 대신 포인트턴을 사용하여 설계하였다.
- **Linetracing 모듈:**
5방향 센서가 읽어오는 값에 따라서 구동체가 라인에 어떤 위치에 서있는지 판단해 주어야 하는데, 구동체가 라인에 어떤 상태로 위치하는지 알려주는 기능을 담당하는 모듈이다. 이 모듈 안에선 5방향 센서가 라인을 읽기 위한 기본적인 설정을 정해주었고 다른 코드에서 쉽게 불러올 수 있도록 함수를 만들어 읽어오는 센서 값들을 리스트 형태로 리턴하게끔 설계하였다.
- **Maze 모듈:**
이 모듈을 위에 기본적인 움직임들을 이용해 미로에서의 특정 움직임들을 담당하는 모듈로, 미로 찾기에서 필요한 우회전, 좌회전, 유턴의 동작들을 구현해주고, 해당 동작을 수행할 때의 세기와 시간들도 변수를 지정해 할당해 주도록 설계하였다.
- **Main 코드:**
이 코드에서는 위 4개의 모듈들을 이용해 오른손 법칙을 사용하여 미로를 통과하도록 설계한다. 기본적인 라인 트레이싱과 더불어 오른쪽으로 턴해야 될 때의 센서 상태 값들과 마찬가지로 왼쪽으로 턴할 때, 유턴할 때의 센서 상태 값들을 이중리스트 형태로 저장하고 조건문을 활용해 구동체가 알맞은 움직임을 수행하게끔 설계하였다.

| | | | |
|-----------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  <div> 국민대학교 소프트웨어학부 창업연계공학설계입문 </div> | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

3 수행 내용

3.1 프로그램 코드 (코드 설명)

<Setup 모듈>

```
# GPIO 라이브러리 임포트
import RPi.GPIO as GPIO

# GPIO 모듈 BOARD 모드로 설정
GPIO.setmode(GPIO.BOARD)

# GPIO 경고 메시지를 나타나지 않게 설정
GPIO.setwarnings(False)

# 전진하기 위한 변수 forward와 backward에 각각 True, False값 할당
forward = True
backward = not(forward)

# =====
# 라즈베리파이에는 12, 11, 35번 핀 선언
# 좌측모터를 컨트롤하기 위해 필요한 세개의 핀
# =====
MotorLeft_A = 12
MotorLeft_B = 11
MotorLeft_PWM = 35

# =====
# 라즈베리파이에는 15, 13, 37번 핀 선언
# 우측모터를 컨트롤하기 위해 필요한 세개의 핀
# =====
MotorRight_A = 15
MotorRight_B = 13
MotorRight_PWM = 37

# =====
# GPIO 핀을 통해 라즈베리파이가 OUT으로 설정
# 좌측 모터의 출력 내보냄
# =====

# 좌측 모터 핀을 Output으로 설정
GPIO.setup(MotorLeft_A, GPIO.OUT)
GPIO.setup(MotorLeft_B, GPIO.OUT)
GPIO.setup(MotorLeft_PWM, GPIO.OUT)
```

- GPIO 라이브러리 임포트
- 전진하기 위한 변수 forward와 backward에 각각 True, False값 할당
- GPIO 핀을 통해 라즈베리파이가 OUT으로 설정


```
# 우측 모터 핀을 Output으로 설정
GPIO.setup(MotorRight_A, GPIO.OUT)
GPIO.setup(MotorRight_B, GPIO.OUT)
GPIO.setup(MotorRight_PWM, GPIO.OUT)

# 좌측/우측 모터 속도를 작동시키게 설정
GPIO.output(MotorLeft_PWM, GPIO.HIGH)
GPIO.output(MotorRight_PWM, GPIO.HIGH)

# =====
# 좌측 모터 속도 제어할 변수 생성
# 우측 모터 속도 제어할 변수 생성
# =====

# 속도 제어 (100 제한)
LeftPwm = GPIO.PWM(MotorLeft_PWM, 100)
RightPwm = GPIO.PWM(MotorRight_PWM, 100)
```

- 모터 핀을 Output으로 설정
- 두 모터 속도 제어할 변수 생성

| | | | |
|----------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  <div> <p>국민대학교</p> <p>소프트웨어학부</p> <p>창업연계공학설계입문</p> </div> | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

```
def leftmotor(x):
    # 왼쪽 모터의 Red 전원선이 위에 있을 때 leftmoto
    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):
    # 오른쪽 모터의 Red 전원선이 위에 있을 때 leftmoto
    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)
    else:
        print('Config Error')
```

- forward, backward의 boolean 값에 따라 두 모터의 속도를 정해주는 함수 생성

<Movement 모듈>

```
# GPIO 라이브러리, time, Setup 모듈 임포트
import RPi.GPIO as GPIO

# time 매소드 임포트
import time

# Setup 모듈 임포트
# 구동체의 움직임을 위해 기본적으로 설정해주는 모듈로,
# 구동체의 좌측/우측 모터의 속도 조절과 전진/후진 설정을 해줌
import Setup


def pwm_setup():
    # 좌측/우측 모터 속도 초기값 설정
    Setup.LeftPwm.start(0)
    Setup.RightPwm.start(0)

def go_forward(speed, running_time):
    # 시간 running_time과 속도 speed를 파라미터로 받음
    # 구동체를 특정 시간동안 특정 속도로 전진하게끔 하는 함수임
    Setup.leftmotor(Setup.forward)
    Setup.GPIO.output(Setup.MotorLeft_PWM, Setup.GPIO.HIGH)
    Setup.rightmotor(Setup.forward)
    Setup.GPIO.output(Setup.MotorRight_PWM, Setup.GPIO.HIGH)
    Setup.LeftPwm.ChangeDutyCycle(speed)
    Setup.RightPwm.ChangeDutyCycle(speed)
    time.sleep(running_time)

def stop():
    # 구동체의 정지하게 하는 코드
    # 모터의 속도를 GPIO.LOW로 설정하고 속도를 0으로 설정함으로써 정지하게끔 함
    GPIO.output(Setup.MotorLeft_PWM, GPIO.LOW)
    GPIO.output(Setup.MotorRight_PWM, GPIO.LOW)
    Setup.LeftPwm.ChangeDutyCycle(0)
    Setup.RightPwm.ChangeDutyCycle(0)

def pwm_low():
    # 구동체의 동작을 완전히 멈추는 코드 (대부분 강제 정지 때 사용)
    # 모터의 속도를 GPIO.LOW로 설정하고 속도를 0으로 설정함으로써 정지하게끔 함
    # GPIO에 있는 cleanup() 메서드를 이용해 구동체의 동작을 멈출
    GPIO.output(Setup.MotorLeft_PWM, GPIO.LOW)
    GPIO.output(Setup.MotorRight_PWM, GPIO.LOW)
    Setup.LeftPwm.ChangeDutyCycle(0)
    Setup.RightPwm.ChangeDutyCycle(0)
    GPIO.cleanup()
```

- GPIO 라이브러리, time, Setup 모듈 임포트
- pwm_setup() 함수는 좌측/우측 모터 속도 초기값을 설정
- go_forward() 함수는 시간 running_time과 속도 speed를 파라미터로 받고, 구동체를 특정 시간동안 특정 속도로 전진하게끔 함
- stop() 함수는 구동체를 정지하게끔 함
- pwm_low() 함수는 구동체의 동작을 완전히 멈추게 함

| | | | |
|--------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  <div> 국민대학교 소프트웨어학부 창업연계공학설계입문 </div> | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

```

def rightPointTurn(speed, running_time):
    # speed와 running_time을 파라미터로 가져옴
    # 각 바퀴의 방향을 반대로 해서 던하는 유형
    # 좌측 모터를 전진으로, 우측 모터를 후진으로 설정
    Setup.leftmotor(Setup.forward)
    Setup.rightmotor(Setup.backward)

    Setup.LeftPwm.ChangeDutyCycle(speed)
    Setup.RightPwm.ChangeDutyCycle(speed)
    time.sleep(running_time)


def leftPointTurn(speed, running_time):
    # speed와 running_time을 파라미터로 가져옴
    # 각 바퀴의 방향을 반대로 해서 던하는 유형
    # 우측 모터를 전진으로, 좌측 모터를 후진으로 설정
    Setup.leftmotor(Setup.backward)
    Setup.rightmotor(Setup.forward)

    Setup.LeftPwm.ChangeDutyCycle(speed)
    Setup.RightPwm.ChangeDutyCycle(speed)
    time.sleep(running_time)

def go(leftspeed, rightspeed):
    # 이 함수는 구동체의 오른쪽바퀴와 왼쪽바퀴의 속도를
    # start라는 메서드를 이용해 바퀴를 전진하게 하
    # leftspeed는 좌측 바퀴의 속도를, rightspeed는
    Setup.leftmotor(Setup.forward)
    Setup.rightmotor(Setup.forward)
    Setup.LeftPwm.start(leftspeed)
    Setup.RightPwm.start(rightspeed)

```

- rightPointTurn() 함수는 speed와 running_time을 파라미터로 가져오고, 좌측 모터를 전진으로, 우측 모터를 후진으로 설정하고 던함
- leftPointTurn() 함수는 speed와 running_time을 파라미터로 가져오고, 좌측 모터를 후진으로, 우측 모터를 전진으로 설정하고 던함
- go() 함수는 구동체의 오른쪽바퀴와 왼쪽바퀴의 속도를 일일이 입력값으로 받아와, start라는 메서드를 이용해 바퀴를 전진하게 하게끔 만든 함수임
(+ 전진을 go_any에 내재된 메서드를 이용하지 않음)
(leftspeed는 좌측 바퀴의 속도를, rightspeed는 우측 바퀴의 속도를 받아오는 입력값임)

| | | | |
|--------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  <div> 국민대학교 소프트웨어학부 창업연계공학설계입문 </div> | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

<Linetracing 모듈>

```

# GPIO 라이브러리 импорт
import RPi.GPIO as GPIO

# GPIO 모드를 BOARD 모드로 설정
GPIO.setmode(GPIO.BOARD)

# GPIO 경고 메시지를 나타나지 않게 설정
GPIO.setwarnings(False)

leftmostled = 16    # OTD(16)는 가장 왼쪽에
leftlessled = 18    # OTB(18)는 두 번째 왼쪽에
centerled = 22      # OTA(22)는 중앙에 있는
rightlessled = 40   # OTC(40)는 두 번째 오른쪽에
rightmostled = 32   # OTE(32)는 가장 오른쪽에

# 5방향 센서의 출력 데이터가 라즈베리 파이로 입력
# 라즈베리 파이의 16, 18, 22, 40, 32번 핀은 1
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)


def getLine():
    # 5방향 센서가 라인을 읽을 때 받아오는 값들
    # 그 변수를 하나의 리스트에 순서대로 입력시켜
    # 다른 코드에서 이 함수를 통해 현재 라인 상태
    l1 = (GPIO.input(leftmostled))
    l2 = (GPIO.input(leftlessled))
    c = (GPIO.input(centerled))
    r2 = (GPIO.input(rightlessled))
    r1 = (GPIO.input(rightmostled))

    sensor_list = [l1,l2,c,r2,r1]

    return sensor_list

```

- GPIO 라이브러리 импорт
- 5 방향 각각의 센서와 연결되어있는핀 번호를 값으로, 각각의 센서의 위치와
알맞게 변수 이름을 정해준 후, 할당해줌
- 5 방향 센서의 출력 데이터가 라즈베리 파이로 입력되기 때문에 라즈베리 파이의
16, 18, 22, 40, 32 번 핀은 Input 핀으로 설정함
- getLine() 함수는 5방향 센서가 라인을 읽을 때 받아오는 값들을 각각의 특정 변수이름에
할당하고, 그 변수를 하나의 리스트에 순서대로 입력시켜 리턴시킴
(다른 코드에서 이 함수를 통해 현재 라인 상태를 알 수 있음)

| | | | |
|---------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  국민대학교 소프트웨어학부 창업연계공학설계입문 | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

<Maze 모듈>

```
# Movement 모듈 임포트
# 구동체의 기본적인 움직임을 담당하는 모듈로,
# 전진, 포인트턴, 멈춤, 라인트레이싱 등의 움직임을 메서드를 포함하고
import Movement


# Linetracing 모듈 임포트
# 5방향 센서가 라인을 읽을 때 받아오는 값들을 받아오기 위해 필요한 모듈
# 이 모듈의 메서드를 통해 현재 라인 상태를 알 수 있음
import Linetracing

# 구동체가 특정 움직임을 수행하기 전에 전진함
# 그때의 전진 속도와 시간을 설정
forwardspeed = 40
forwardtime = 0.5

# 구동체가 우회전/좌회전이나 유턴을 행할 때,
# 처음 회전은 큰 회전으로, 회전 세기와 시간을 설정해줌
# 두번째 회전은 세밀한 회전으로, 회전 시간만 설정해줌
turnspeed = 27
turntime_bf = 0.4
turntime_af = 0.1
turntime_ut = 1

def isright():
    # 우회전을 행할 상황에서 수행되는 함수로,
    # 회전 수행 전, 위에서 정해진 속도와 시간으로 전진하고,
    # 우측으로 포인트턴을 수행함, 그리고 5방향 센서 중 가운데 센서가
    # 첫번째 회전은 큰 회전으로, 두번째 회전은 세밀한 회전으로 주행
    Movement.go_forward(forwardspeed, forwardtime)
    Movement.rightPointTurn(turnspeed, turntime_bf)
    while True:
        line = Linetracing.getLine()
        Movement.rightPointTurn(turnspeed, turntime_af)
        print('go!right!')
        if line[2] == 0:
            break
```

- Movement 모듈, Linetracing 모듈 임포트
 - 구동체가 특정 움직임을 수행하기 전에 전진함, 그때의 전진 속도와 시간을 설정
 - 구동체가 우회전/좌회전이나 유턴을 행할 때, 처음 회전은 큰 회전으로, 회전 세기와 시간을 설정해줌
 - 두번째 회전은 세밀한 회전으로, 회전 시간만 설정해줌
 - isright() 함수는 우회전을 행할 상황에서 수행되는 함수로, 회전 수행 전, 위에서 정해진 속도와 시간으로 전진하고, 우측으로 포인트턴을 수행함, 그리고 5방향 센서 중 가운데 센서가 검은색 라인에 올 때까지 진행함.
- (첫번째 회전은 큰 회전으로, 두번째 회전은 세밀한 회전으로 주행 속도를 높임)

| | | | |
|--------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  <div> 국민대학교 소프트웨어학부 창업연계공학설계입문 </div> | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

```

def isleft():
    # 좌회전을 행할 상황에서 수행되는 함수로,
    # 회전 수행 전, 위에서 정해진 속도와 시간으로 전진하고,
    # 좌측으로 포인트턴을 수행함, 그리고 5방향 센서 중 가운데 센서가 검
    # 첫번째 회전은 큰 회전으로, 두번째 회전은 세밀한 회전으로 주행 속도
    Movement.go_forward(forwardspeed, forwardtime)
    Movement.rightPointTurn(turnspeed, turntime_bf)
    while True:
        line = Linetracing.getLine()
        print('go!left!')
        Movement.leftPointTurn(turnspeed, turntime_af)
        if line[2] == 0:
            break

def uturn():
    # 유턴을 행할 상황에서 수행되는 함수로,
    # 회전 수행 전, 위에서 정해진 속도와 시간으로 전진하고,
    # 우측으로 포인트턴을 수행함, 그리고 5방향 센서 중 가운데 센서가 검
    # 첫번째 회전은 큰 회전으로, 두번째 회전은 세밀한 회전으로 주행 속도
    Movement.go_forward(forwardspeed, forwardtime)
    Movement.rightPointTurn(turnspeed, turntime_ut)
    while True:
        line = Linetracing.getLine()
        print('go!uturn!')
        Movement.rightPointTurn(turnspeed + 3, turntime_af)
        if line[2] == 0:
            break


```

- isleft() 함수는 좌회전을 행할 상황에서 수행되는 함수로, 회전 수행 전, 위에서 정해진 속도와 시간으로 전진하고, 좌측으로 포인트턴을 수행함, 그리고 5방향 센서 중 가운데 센서가 검은색 라인에 올 때까지 진행함

(첫번째 회전은 큰 회전으로, 두번째 회전은 세밀한 회전으로 주행 속도를 높임)

- uturn() 함수는 유턴을 행할 상황에서 수행되는 함수로, 회전 수행 전, 위에서 정해진 속도와 시간으로 전진하고, 우측으로 포인트턴을 수행함, 그리고 5방향 센서 중 가운데 센서가 검은색 라인에 올 때까지 진행함

(첫번째 회전은 큰 회전으로, 두번째 회전은 세밀한 회전으로 주행 속도를 높임)

| | | | |
|---------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  국민대학교 소프트웨어학부 창업연계공학설계입문 | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

<Main 코드>

```
# Movement 모듈 임포트
# 구동체의 기본적인 움직임을 담당하는 모듈로,
# 전진, 포인트턴, 멈춤, 라인트레이싱 등의 움직임의 메서드를 포함하고 있음
import Movement

# Linetracing 모듈 임포트
# 5방향 센서가 라인을 읽을 때 받아오는 값들을 받아오기 위해 필요함
# 이 모듈의 메서드를 통해 현재 라인 상태를 알 수 있음
import Linetracing


# Maze 모듈 임포트
# 구동체가 미로 속에서 주행할 때, 수행해야할 움직임을 담당하는 모듈로,
# 우회전, 좌회전, 유턴 움직임의 메서드를 포함하고 있음
import Maze

# 속도를 6개의 단계로 나누어 변수에 지정해줌
# 여기서 veryhighspeed는 구동체가 직진할 때의 가장 보편적인 속도임
veryhighspeed = 40
highspeed = 28
midspeed = 18
lowspeed = 10
verylowspeed = 3
zero = 0

# 이중리스트 형태이며, 첫번째 인덱스에 해당되는 부분은 5개의 센서가 받아오는
# 두번째 인덱스에 해당되는 부분은 이 코드에 자체적으로 생성한 go()함수에
# 세번째 인덱스에 해당되는 부분은 이 코드에 자체적으로 생성한 go()함수에
linetracing_list = [
    [[0,1,1,1,1], verylow, veryhigh],
    [[0,0,1,1,1], low, veryhigh],
    [[1,0,1,1,1], mid, veryhigh],
    [[1,0,0,1,1], high, veryhigh],
    [[1,1,1,1,0], veryhigh, verylow],
    [[1,1,1,0,0], veryhigh, low],
    [[1,1,1,0,1], veryhigh, mid],
    [[1,1,0,0,1], veryhigh, high],
    [[1,1,0,1,1], veryhigh, veryhigh],
    [[1,0,0,0,1], veryhigh, veryhigh]
]

# 오른쪽/왼쪽 그리고 유턴으로 포인트턴할때의 센서 상태를 이중리스트 형태로 만들어 줌
right_list = [[0,0,0,0,0],[1,1,0,0,0],[1,0,0,0,0]]
left_list = [[0,0,0,1,1]]
uturn_list = [[1,1,1,1,1]]
```

- Movement 모듈, Linetracing 모듈, Maze 모듈 임포트
- 속도를 6개의 단계로 나누어 변수에 지정해줌
- veryhighspeed 는 구동체가 직진할 때의 가장 보편적인 속도를 입력함
- 이중리스트 형태이며, 첫번째 인덱스에 해당되는 부분은 5 개의 센서가 받아오는 센서의 결과값을 예측한 부분임
- 두번째 인덱스에 해당되는 부분은 이 코드에 자체적으로 생성한 go()함수에 좌측 바퀴 속도로 입력하는 값임
- 세번째 인덱스에 해당되는 부분은 이 코드에 자체적으로 생성한 go()함수에 우측 바퀴 속도로 입력하는 값임
- 오른쪽/왼쪽 그리고 유턴으로 포인트턴할때의 센서 상태를 이중리스트 형태로 만들어 줌

| | | | |
|----------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  <div> <p>국민대학교</p> <p>소프트웨어학부</p> <p>창업연계공학설계입문</p> </div> | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

```

if __name__ == "__main__":
    try:
        while True:
            # Linetracing 모듈에 getLine() 메서드를 이용해서,
            # 현재 라인 상태를 리스트 형태로 받아옴
            line = Linetracing.getLine()

            # 위에 선언해준 linetracing_list의 길이만큼 for문을 반복함
            # 이중 리스트인 linetracing_list의 첫번째 인덱스 값이 5방향 센서의 데이터와 같
            # 두번째 인덱스와 세번째 인덱스 값을 Movement 모듈에 있는 go()함수의 파라미터로
            for i in range(len(linetracing_list)):
                if line == linetracing_list[i][0]:
                    Movement.go(linetracing_list[i][2], linetracing_list[i][1])
                    break

            # 5방향 센서에서 읽어오는 데이터가 오른쪽으로 턴할때의 센서 상태를 저장한
            # 이중리스트 안에 해당 값이 있으면 Maze 모듈에 있는 isright() 메서드를 실행시킴
            if line in right_list:
                print('right')
                Maze.isright()


            # 5방향 센서에서 읽어오는 데이터가 왼쪽으로 턴할때의 센서 상태를 저장한
            # 이중리스트 안에 해당 값이 있으면 Maze 모듈에 있는 isleft() 메서드를 실행시킴
            elif line in left_list:
                print('left')
                Maze.isleft()

            # 5방향 센서에서 읽어오는 데이터가 유턴으로 턴할때의 센서 상태를 저장한
            # 이중리스트 안에 해당 값이 있으면 Maze 모듈에 있는 uturn() 메서드를 실행시킴
            elif line in uturn_list:
                print('uturn')
                Maze.uturn()

        except KeyboardInterrupt:
            # Ctrl+C키를 누를시,
            # 움직이는 구동체 작동 멈춤
            # 이 경우 Movement 모듈에 있는 pwm_low() 메서드 실행시킴
            Movement.pwm_low()

```

- 5 방향 센서가 라인을 계속 인식해야 하므로 while Ture 문을 이용해서 반복시킴
- Linetracing 모듈에 getLine() 메서드를 이용해서, 라인 상태를 리스트 형태로 받아옴
- 위에 선언해준 linetracing_list의 길이만큼 for문을 반복함
- 이중 리스트인 linetracing_list의 첫번째 인덱스 값이 5방향 센서의 데이터와 같으면, 두번째 인덱스와 세번째 인덱스 값을 Movement 모듈에 있는 go()함수의 파라미터로 입력시켜 go()함수를 구동시킴
- 5방향 센서에서 읽어오는 데이터가 오른쪽으로 턴할때의 센서 상태를 저장한 이중리스트 안에 해당 값이 있으면 Maze 모듈에 있는 isright() 메서드를 실행시킴
- 5방향 센서에서 읽어오는 데이터가 왼쪽으로 턴할때의 센서 상태를 저장한 이중리스트 안에 해당 값이 있으면 Maze 모듈에 있는 isleft() 메서드를 실행시킴
- 5방향 센서에서 읽어오는 데이터가 유턴으로 턴할때의 센서 상태를 저장한 이중리스트 안에 해당 값이 있으면 Maze 모듈에 있는 uturn() 메서드를 실행시킴
- Ctrl+C키를 누를시, 움직이는 구동체 작동 멈춤
(이 경우 Movement 모듈에 있는 pwm_low() 메서드 이용함)

| | | | |
|---------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  국민대학교 소프트웨어학부 창업연계공학설계입문 | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

4 향후 추진계획

4.1 향후 계획의 세부 내용

- 배터리:

배터리를 충전한 상태에서 시험을 봤는데 회전할 때 힘이 약해 완전히 돌지 못하고 직진할 때 속도가 느려지는 경향이 보였다. 이 부분에서 우리는 구동체의 배터리의 수명이 짧아졌거나 아크릴판 트랙의 문제라고 생각하였다. 그래서 배터리를 완전히 충전한 상태에서 구동체의 동작을 시험해서 배터리의 수명이 정말로 단축되었는지 확인할 것이다.

그리고 계속 된 시행을 통해 우리는 보다 더 일정한 배터리 상태를 위해 배터리를 완전히 충전한 상태로 구동체를 작동시킬 때의 배터리 사용 시간이 아닌, 주행 동작을 구동체가 몇 번 행했는지 판단하는 것이 보다 배터리의 전력사용을 정확하게 계산할 수 있지 않을까 생각하고 향후 시행때는 구동체의 시행 횟수를 체크하면서 작동시킬 것이다.

- 바닥 상태:


위에서 언급한 대로, 시험 때 구동체가 예상과 다르게 잘 작동하지 않았다. 이번 시험에서 가장 중요한 턴 구간에서 구동체가 힘없이 도는 등, 배터리가 거의 바닥인 상태인 것처럼 구동체가 작동했다. 하지만, 우리 조는 시험보기 전에 충분한 시간동안 배터리를 충전해 두었고, 다른 트랙에서의 시행에서는 구동체가 우리의 예상대로 잘 움직여 주었다. 여기서 우리는 시험을 봤을 때의 트랙의 상태에 대해 의심하지 않을 수 없었다.

우리의 예상은 시험 당시 아크릴판의 매끄러움 자체가 다른 트랙에 비해 다소 낮았다고 생각했었고 우리 조는 트랙일 때는 포인트턴의 회전양을 늘림으로써 해결할 수 있다고 판단했다. 포인트턴의 회전양을 늘리기 위해서 생각한 결과, 포인트턴을 할 때의 모터의 속도가 아닌 시간을 늘려주는 것이 더욱 효율적으로 회전양을 늘릴 수 있다고 결론을 내렸다.

- 더 빠른 완주:

지난 과제와 마찬가지로 이번 과제도 더 빨리 완주하는 것이 좋다.

그러기 위해서 우리 조가 생각한 결론은 회전을 할 때의 구동체의 몸체가 꼭지점까지 전진을 하고 회전하는 형식이 아닌, 회전 전에 전진 할 필요없이 약 45~60도 정도로 구동체를 대각선으로 회전시키고 구동체가 진행 라인을 라인 트레이싱하게끔 만든다면 더욱 빠른 완주를 만들 수 있다.


| | | | |
|---------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  국민대학교 소프트웨어학부 창업연계공학설계입문 | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

5 애로 및 건의사항

이번 과제는 한 번 미로를 통과하기 위해서는 전 과제들에 비하면 많은 시간이 걸렸다. 그래서인지 한 번 실행할 때마다 구동체의 움직임이 많이 느려진 것으로 보아 배터리의 양이 급격히 준 것으로 판단되었다. 그 결과 연습을 하고 시험을 보려고 하면 이미 배터리의 상태가 부족해서 구동체가 느리게 가고 회전이 잘 되지 않는 모습들을 보였다. 하지만 교수님께서 이런 부분들은 감안해 주셔서 모든 학생들이 시험을 편하게 잘 볼 수 있었던 것 같다. 이번 과제 역시 저번 과제와 마찬가지로 구동체의 속도가 너무 빠르면 라인 트레이싱을 제대로 하지못해 라인을 벗어나도 인지를 잘 하지 못하고 회전할 때 회전을 하지 않는 오류들이 발생하였다. 그리고 시험을 본 아크릴 판과 연습을 한 아크릴 판의 차이도 좀 있어서 적응하는데도 시간이 필요했다. 그 중 판의 크기가 달라서 이번 미로 탈출에서 중요한 우회전, 좌회전, 유턴 등을 할 때 구동체가 판을 벗어나는 바람에 다시 손으로 직접 라인위에 올려놓아야 하는 불편함들이 있었다.

이번 과제는 2학기 창업연계 공학설계입문의 마지막 과제인 미로 찾기를 수행하는 것이었는데 마지막답게 이 과제를 해내는데 많은 시행착오들을 겪었다. 하지만 다시 한번 막상 코딩을 하는 것이 아니라 처음부터 같이 프로그램 설계를 하고 계획을 세우면서 여러 번 시도도 해보고, 노력한 결과 마지막 과제를 무사히 끝마쳐서 뿌듯하고 안심이 되었다.

이번 수업으로 프로그램을 하드웨어에 연결시켜서 실험하는 새로운 방식의 수업이라 재미있었다. 그리고 후반부에는 조원들과 같이 과제를 해내는 과정들을 보면서 공동작업에도 흥미를 느끼게 되었다.

| | | | |
|---------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------|------------|
|  국민대학교 소프트웨어학부 창업연계공학설계입문 | 5차 과제 보고서 | | |
| | 프로젝트 명 | 미로 찾기 | |
| | 팀 명 | 8조였나 9조였나 | |
| | Confidential Restricted | Version 1.2 | 2017-DEC-9 |

6 회의록

| | | | |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|-------|
| 일 시 | 11/30, 12/2, 12/3 | 차수 | 1,2,3 |
| 장 소 | 국민대학교 무한상상실 | | |
| 참석자 | 노성환, 문성찬, 박건후 | | |
| 불참자 | 없음 | | |
| 안 건 | 구동체 구동을 위한 효율적인 코드 작성과 개선 방안 | | |
| 회의내용 | <ul style="list-style-type: none"> - 교수님이 원하는 결과물이 무엇인가? - 코드는 어떻게 작성하는 것이 좋을까? - 스켈레톤 코드를 기반으로 무엇을 고려해야할까? - 배터리 문제는 어떻게 해결할까? - 우선, 과제의 목표부터 해결하도록 하자. - 코드를 짤 때 특정 상황을 포괄할 수 있는 방법을 고려해보자. - 역할 분담은 어떻게 해볼까? - 보고서 작성시 필요한 사진을 찍어야 하는데 언제 찍어야할까? - python 라이브러리를 분석해보는 시간을 갖아보도록 하자. | | |
| 결과물 | <ul style="list-style-type: none"> - 코드를 짤 때 특정 상황을 포괄할 수 있는 방법을 고려해보자. 2차원 배열을 사용하여 left/right/uturn을 판단해보도록 하자 - 역할 분담은 어떻게? 코드 작성은 다같이 검토하도록 하는데, 메인코드는 성찬이가, 부수적인 부분은 성환이가, 개선 방안과 오류 처리는 건후가 하도록 했다. - python 라이브러리를 분석해보는 시간을 갖아보도록 하자. 생각보다 쉬울 줄 알았는데 어려웠고, 시간이 많이 걸리니까 패스 - 교수님이 원하는 결과물이 무엇인가? 빠르고 정확한 판단을 하는 라즈베리카, 오류 없는 코드와 시간이 적게 걸리는 미로 찾기 | | |