임베디드응용및실습 과제

8주차. 과제 보고서

과 목 명	임베디드응용및실습
학 번	2020161082
이 름	이강산
제 출 일	2025년 11월 5일

과제

- lecture 7 lab 강의 자료 맨 마지막 페이지의 과제를 수행
- 작성 코드와 실행 장면을 동영상으로 짧게 촬영
- 자신의 github의 폴더를 생성하여 보고서(소스 코드)와 동영상을 업로드하고 URL만 과제 란에 제출

https://github.com/20thgam/git_2025_week8.git

1. Bluetooth통신으로 움직이는 자동차 만들기

1) 지난 시간에 작성한 "스위치로 움직이는 자동차" 코드를 참조하여 스마트폰의 BT app.에서 명령을 내려 움직이는 자동차 만들기
2) 명령을 위한 "go, back, left, right, stop" 함수 구현
3) BT app.에서 버튼과 명령을 mapping
4) 버튼이 눌리면 명령(ex. Right)가 자동차로 전송, serial_thread 함수에서 받아 gData에 할당
5) Main 함수에서 gData 값에 따라 적절한 함수를 호출하여 자동차 구동

- 코드

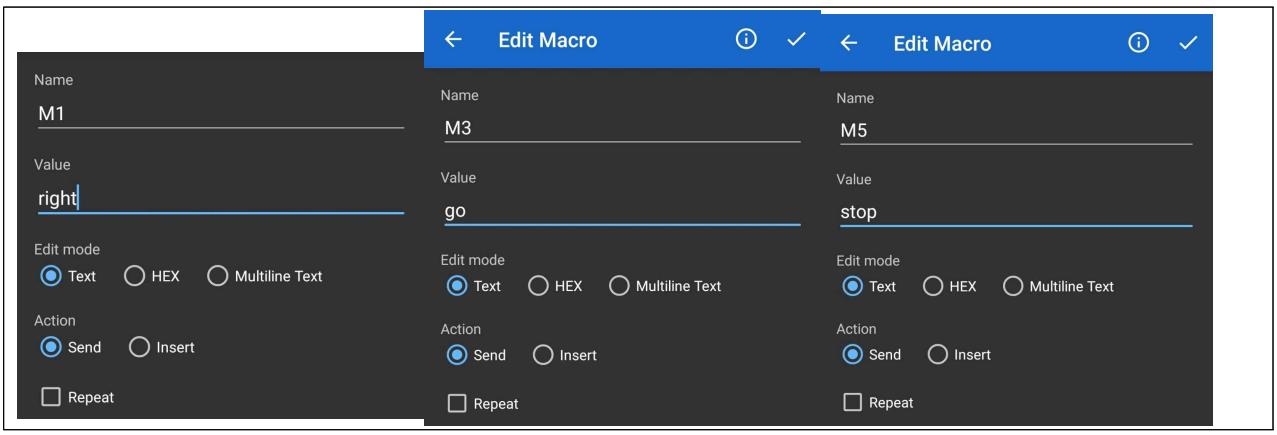
```
import RPi.GPIO as GPIO
import threading
import serial
import time

# ==========
# GPIO 핀 설정 (L298N 모터 드라이버 기준 예시)
# 사용자의 모터 드라이버에 맞게 핀 번호를 수정해야 합니다.
# ==========
MOTOR_A_EN = 18  # 왼쪽 모터 Enable
MOTOR_A_IN1 = 22
MOTOR_A_IN2 = 27

MOTOR_B_EN = 23  # 오른쪽 모터 Enable
MOTOR_B_IN1 = 25
MOTOR_B_IN2 = 24

# ==========
# 블루투스 시리얼 설정
# ==========
bleSerial = serial.Serial("/dev/ttyS0", baudrate=9600, timeout=1.0)

# ==========
# 전역 변수: 블루투스 데이터를 저장할 공간
# ==========
gData = ""
```

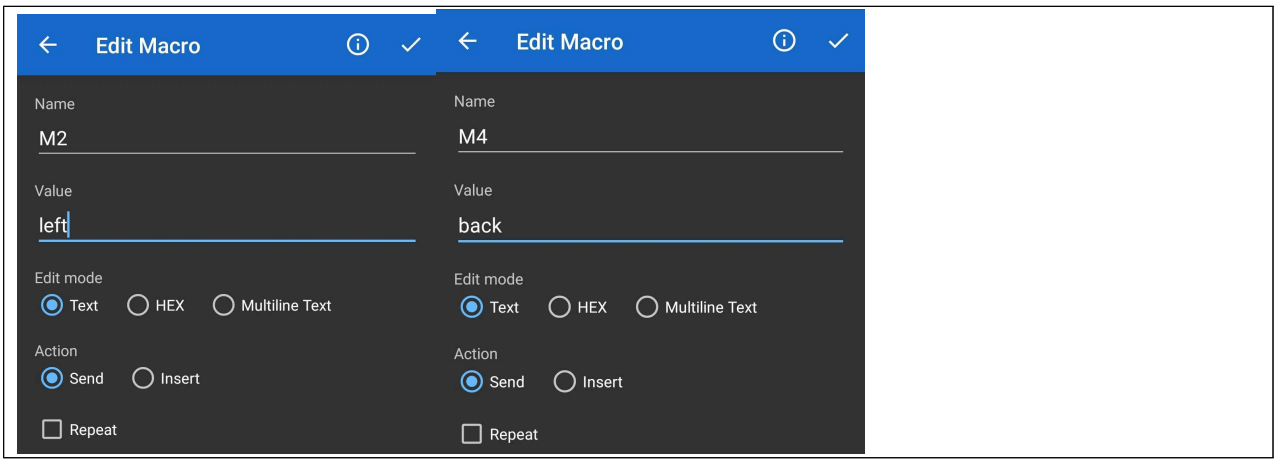
```
# ==========
# 모터 제어 함수
# ==========
def back():
    print("motor: back")
    GPIO.output(MOTOR_A_IN1, GPIO.HIGH)
    GPIO.output(MOTOR_A_IN2, GPIO.LOW)
    GPIO.output(MOTOR_B_IN1, GPIO.HIGH)
    GPIO.output(MOTOR_B_IN2, GPIO.LOW)

def go():
    print("motor: go")
    GPIO.output(MOTOR_A_IN1, GPIO.LOW)
    GPIO.output(MOTOR_A_IN2, GPIO.HIGH)
    GPIO.output(MOTOR_B_IN1, GPIO.LOW)
    GPIO.output(MOTOR_B_IN2, GPIO.HIGH)

def right():
    print("motor: right")
    GPIO.output(MOTOR_A_IN1, GPIO.LOW)
    GPIO.output(MOTOR_A_IN2, GPIO.HIGH)
    GPIO.output(MOTOR_B_IN1, GPIO.HIGH)
    GPIO.output(MOTOR_B_IN2, GPIO.LOW)

def left():
    print("motor: left")
    GPIO.output(MOTOR_A_IN1, GPIO.HIGH)
    GPIO.output(MOTOR_A_IN2, GPIO.LOW)
    GPIO.output(MOTOR_B_IN1, GPIO.LOW)
    GPIO.output(MOTOR_B_IN2, GPIO.HIGH)

def stop():
    print("motor: stop")
    GPIO.output(MOTOR_A_IN1, GPIO.LOW)
    GPIO.output(MOTOR_A_IN2, GPIO.LOW)
    GPIO.output(MOTOR_B_IN1, GPIO.LOW)
    GPIO.output(MOTOR_B_IN2, GPIO.LOW)

# ==========
# 블루투스 통신을 처리하는 쓰레드 함수
# ==========
def serial_thread():
```

```
    global gData
    while True:
        # 블루투스로부터 데이터 한 줄을 읽어옴
        data = bleSerial.readline()
        # byte 형태의 데이터를 문자열(string)으로 변환
        data = data.decode('utf-8').strip()

        # 데이터가 있는 경우에만 gData에 저장
        if data:
            print(f"Received from BT: {data}")
            gData = data

# ==========
# 메인 로직 함수: gData 값을 확인하고 자동차를 제어
# ==========
def main():
    global gData
    try:
        while True:
            if "go" in gData:
                gData = ""  # 명령을 처리했으므로 변수 초기화
                go()
            elif "back" in gData:
                gData = ""
                back()
            elif "left" in gData:
                gData = ""
                left()
            elif "right" in gData:
                gData = ""
                right()
            elif "stop" in gData:
                gData = ""
                stop()

            time.sleep(0.1) # CPU 사용량을 줄이기 위해 잠시 대기

    except KeyboardInterrupt:
        pass

# ==========
# 프로그램 시작점
```

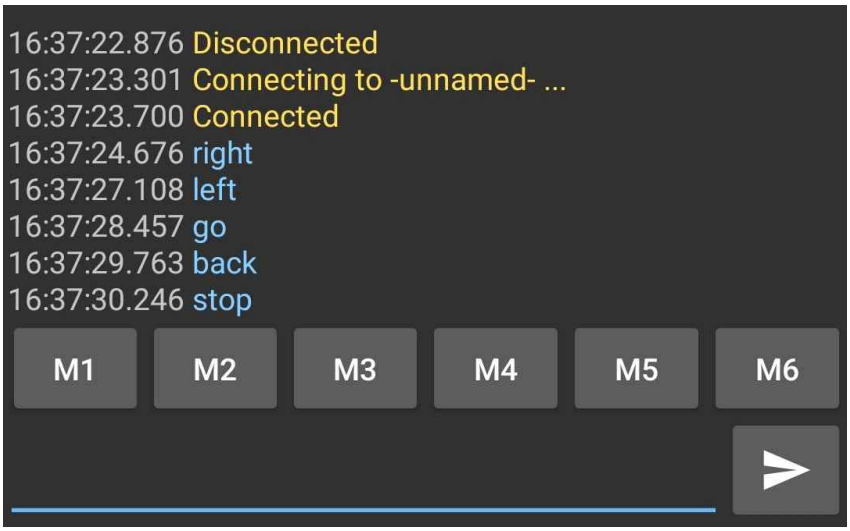
```
# ==========
if __name__ == '__main__':
    # GPIO 설정
    GPIO.setmode(GPIO.BCM)
    GPIO.setup([MOTOR_A_EN, MOTOR_A_IN1, MOTOR_A_IN2, MOTOR_B_EN, MOTOR_B_IN1, MOTOR_B_IN2], GPIO.OUT)
    # 모터 Enable 핀 활성화
    GPIO.output(MOTOR_A_EN, GPIO.HIGH)
    GPIO.output(MOTOR_B_EN, GPIO.HIGH)

    # 블루투스 통신 쓰레드 생성 및 시작
    task1 = threading.Thread(target=serial_thread)
    task1.start()

    print("Bluetooth Car Control Start!")

    # 메인 함수 실행
    main()

    # 프로그램 종료 시 정리
    bleSerial.close()
    GPIO.cleanup()
    print("Program exit.")
```

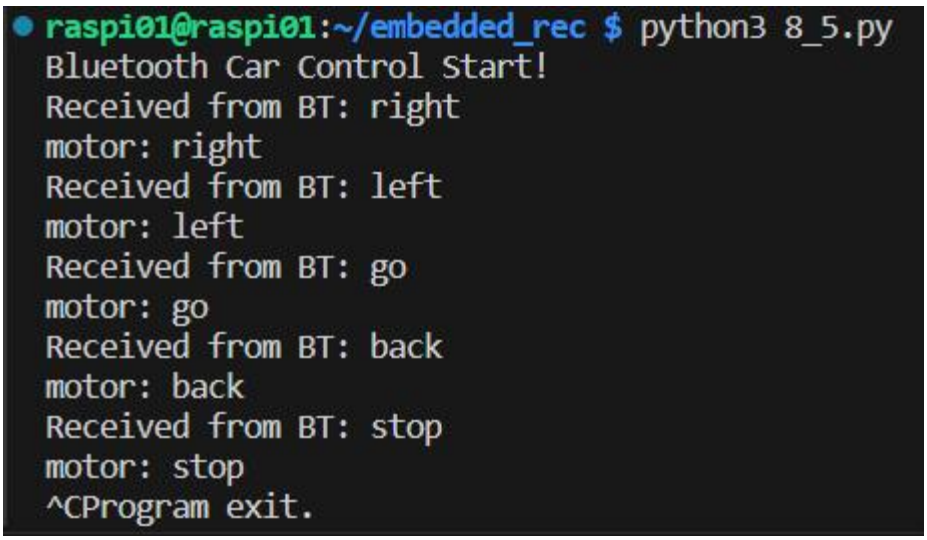
- 버튼 매핑 화면





- 실행 결과





- 고찰

① 초기화: GPIO 핀을 설정하고 모터가 움직일 준비

② 쓰레드 생성: 블루투스 신호를 받을 serial_thread 생성 및 실행 ①은 상태 유지 중
③ 신호 입력 대기: 블루투스를 통해 연결된 휴대전화를 통해 수신 대기
④ 신호 입력: 신호를 받으면 쓰레드는 전역 변수 gData에 그 값을 기록. ③은 계속 유지 중
⑤ 명령 수행: gData에 해당하는 명령어를 호출하여 수행(go, back, ...)
⑥ 초기화: 명령 수행 직후 수행한 명령에 대해서 재동작을 방지하기 위해 초기화
⑦ 입력 대기(0.1s): 0.1초 대기를 통해 0.1초마다 신호를 수신 확인

모바일 -(블루투스)-> 라즈베리파이의 연결을 통해 모바일로 자동차를 조작한다.
스레드를 통해서 두 가지의 동작을 동시에 실행하게 하였습니다. 스레드를 통해 블루투스로부터의 수신을 처리하여 모바일에서 라즈베리파이로 신호를 수신하도록 만들었습니다. 모터제어를 메인 스레드를 통해 처리하여 수신과 자동차 움직임을 동시에 수행하도록 만들었습니다. 두 개의 스레드이다 보니 gData의 전역 변수를 사용하여 명령을 공유하도록 만들었습니다. 어디서든 전역 변수라면 접근할 수 있기 때문입니다. 사용한 전역 변수에 대해서는 초기화를 바로 진행해 의도 밖의 움직임을 하지 않는지 확인했습니다.
미비한 부분에 대해서는 속도 제어가 가능은 하나 해당 코드에서는 구현되지 않았습니다. PWM을 통해 모터 자체의 속도를 조작할 수 있어 이를 활용한다면 더 다양한 움직임을 구현할 수 있습니다.