임베디드 응용 및 실습 과제

7주차. 과제 보고서

과 목 명	임베디드응용및실습
학 번	2021161104
이 름	이주호
제 출 일	2025년 10월 17일

1-1 실습문제

[버튼 입력 받기 구현_10p]

(1) 코드를 추가하여 스위치를 눌렸을 때만 화면에 "click"이 표기되도록 변경

(2) 몇번 스위치가 눌렸는지 확인이 가능하도록 "click x" 등으로 화면 출력

(3) 스위치를 눌렀을 때 0->1, 눌렀다 떼었을 때 1->0으로 값이 변경되므로 0->1인 경우만 동작되도록 변경

(4) 4개의 스위치 입력을 받도록 해보자. 화면에 아래와 같이 출력되도록 한다. 단, 리스트를 최대한 활용하여 GPIO 전/후 값을 저장한다.

1-2. 소스코드

```
# 1
import RPi.GPIO as GPIO
import time
SW1 =5  # 스위치 핀 번호
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(SW1, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  # 풀다운 설정
try:
    while True:
        sw1Value = GPIO.input(SW1)  # 스위치 상태 읽기
        if sw1Value ==1:  # 눌린 경우(0→1)
            print("click!")
        time.sleep(0.1)  # 디바운스 및 CPU 부하 방지
except KeyboardInterrupt:
    pass  # Ctrl+C 종료 시 예외 처리
GPIO.cleanup()  # GPIO 리소스 해제


# 2
import RPi.GPIO as GPIO
import time
# 스위치 핀 번호 지정
SW1, SW2, SW3, SW4 =5, 6, 13, 19
# 스위치 이름 매핑
switches = {
    SW1: "sw1",
    SW2: "sw2",
    SW3: "sw3",
    SW4: "sw4"
}
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
# 모든 스위치를 풀다운 입력으로 설정
for pin in switches:
    GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
try:
    while True:
        for pin in switches:
            # 눌렸을 때만 출력
            if GPIO.input(pin) ==1:
                print("click "+ switches[pin])
                time.sleep(0.3)  # 중복 감지 방지 (디바운스)
        time.sleep(0.01)
```

```
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()


# 3
import RPi.GPIO as GPIO
import time
SW1 =5
switches = {SW1: "sw1"}
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
for pin in switches:
    GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
# 이전 입력 상태 저장 (0=안눌림, 1=눌림)
prev_state = {pin: 0 for pin in switches}
try:
    while True:
        for pin in switches:
            current = GPIO.input(pin)
            # 이전 0 → 현재 1 인 순간만 동작
            if prev_state[pin] ==0 and current ==1:
                print("click "+ switches[pin])
            prev_state[pin] = current  # 상태 갱신
        time.sleep(0.01)
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()


# 4
import RPi.GPIO as GPIO
import time
SW = [5, 6, 13, 19]  # 스위치 핀 리스트
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
# 입력 설정 (풀다운)
for pin in SW:
    GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
# 이전 상태 및 클릭 횟수 초기화
prev_state = [0, 0, 0, 0]
click_count = [0, 0, 0, 0]
try:
    while True:
        for i in range(4):
            current = GPIO.input(SW[i])
            # 눌린 순간(0→1) 감지
            if current ==1 and prev_state[i] ==0:
                click_count[i] +=1
                print(('SW{} click'.format(i+1), click_count[i]))
            prev_state[i] = current  # 상태 갱신
        time.sleep(0.1)
except KeyboardInterrupt:
    pass
```

```
finally:
    GPIO.cleanup()
```

1-3. 실행결과

문제 1번

```
^Cjuho@juho:~/Desktop/embedded_2025/lec6/homework_1 $ python 1.py
click!
click!
click!
```

문제 2번

```
^Cjuho@juho:~/Desktop/embedded_2025/lec6/homework_1 $ python 2.py
click sw2
click sw1
click sw4
click sw3
```

문제 3번

```
^Cjuho@juho:~/Desktop/embedded_2025/lec6/homework_1 $ python 3.py
click sw2
click sw4
click sw3
click sw1
click sw1
click sw3
```

문제 4번

```
^Cjuho@juho:~/Desktop/embedded_2025/lec6/homework_1 $ python 4.py
('SW2 click', 1)
('SW2 click', 2)
('SW2 click', 3)
('SW1 click', 1)
('SW1 click', 2)
('SW1 click', 3)
('SW3 click', 1)
('SW3 click', 2)
('SW4 click', 1)
```

2-1 실습문제

[부저 음계 출력 구현_14p]

(1) "도레미파솔라시도" 음계를 출력
(2) 나만의 경적 소리 구현
(3) 스위치를 한번 누르면 경적 소리가 나도록 구현
(4) 스위치 4개를 사용하여 나만의 음악을 연주

2-2. 소스코드

```
# 1.py
import RPi.GPIO as GPIO  # 라즈베리파이 GPIO 제어 모듈
import time  # 시간 지연을 위한 모듈
BUZZER =12  # 부저 핀 번호
GPIO.setwarnings(False)  # 경고 메시지 비활성화
GPIO.setmode(GPIO.BCM)  # BCM 핀 번호 체계 사용
GPIO.setup(BUZZER, GPIO.OUT)  # 부저를 출력 모드로 설정
p = GPIO.PWM(BUZZER, 1)  # 부저에 PWM 신호 생성 (초기 주파수 1Hz)
p.start(0)  # 듀티비 0으로 PWM 시작 (무음 상태)
scale = [262, 294, 330, 349, 392, 440, 494, 523]  # 도레미파솔라시도 주파수(Hz)
# 각 음을 0.5초씩 재생
for freq in scale:
    p.ChangeFrequency(freq)  # 현재 음의 주파수 설정
    p.ChangeDutyCycle(50)  # 소리 발생 (듀티비 50%)
    time.sleep(0.5)  # 0.5초 동안 유지
p.ChangeDutyCycle(0)  # 소리 끄기
p.stop()  # PWM 중지
GPIO.cleanup()  # GPIO 설정 초기화

# 2.py
import RPi.GPIO as GPIO
import time
BUZZER =12  # 부저 핀 번호
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUZZER, GPIO.OUT)
p = GPIO.PWM(BUZZER, 1)  # 부저 PWM 초기화
p.start(50)  # 초기 듀티비 50%로 시작 (소리 출력 상태)
# 멜로디 음정(Hz)과 각 음의 지속 시간(초)
melody = [440, 440, 440, 392, 440, 440, 392, 392, 330]
duration = [0.2, 0.2, 0.4, 0.2, 0.2, 0.4, 0.2, 0.4, 0.6]
# 각 음을 순서대로 재생
for i in range(len(melody)):
```

```
        p.ChangeFrequency(melody[i])  # 해당 음 주파수로 변경
        time.sleep(duration[i])  # 음 지속 시간만큼 재생
p.ChangeDutyCycle(0)  # 소리 끄기
p.stop()  # PWM 중지
GPIO.cleanup()  # 핀 설정 초기화

# 3.py
import RPi.GPIO as GPIO
import time
BUZZER =12  # 부저 핀 번호
SW =5  # 스위치 핀 번호
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUZZER, GPIO.OUT)  # 부저 출력 설정
GPIO.setup(SW, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)  # 스위치 입력 설정(풀다운)
p = GPIO.PWM(BUZZER, 1)  # PWM 객체 생성
p.start(0)  # 무음 상태로 시작
try:
    while True:
        if GPIO.input(SW) ==1:  # 스위치 눌림 감지
            p.ChangeFrequency(440)  # 440Hz(라) 음 발생
            p.ChangeDutyCycle(50)  # 소리 출력
            time.sleep(0.3)  # 0.3초 유지
            p.ChangeDutyCycle(0)  # 소리 끄기
        time.sleep(0.1)  # 빠른 입력 중복 방지
except KeyboardInterrupt:  # Ctrl+C 시 종료
    pass
p.stop()  # PWM 중지
GPIO.cleanup()  # GPIO 초기화

# 4.py
import RPi.GPIO as GPIO
import time
BUZZER =12  # 부저 핀 번호
SW = [5, 6, 13, 19]  # 4개의 스위치 핀 번호
FREQ = [262, 330, 392, 523]  # 도, 미, 솔, 높은 도 주파수
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUZZER, GPIO.OUT)  # 부저 출력 설정
# 스위치 입력 설정(풀다운)
for pin in SW:
    GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
```

```
p = GPIO.PWM(BUZZER, 1)  # PWM 객체 생성
p.start(0)  # 무음 상태로 시작
try:
    while True:
        for i in range(4):  # 4개의 스위치 순차 확인
            if GPIO.input(SW[i]) ==1:  # 해당 스위치 눌림 시
                p.ChangeFrequency(FREQ[i])  # 대응되는 음 주파수 출력
                p.ChangeDutyCycle(50)  # 소리 출력
                time.sleep(0.3)  # 0.3초 유지
                p.ChangeDutyCycle(0)  # 소리 끄기
        time.sleep(0.05)  # CPU 부하 방지용 짧은 지연
except KeyboardInterrupt:
    pass
p.stop()  # PWM 중지
GPIO.cleanup()  # GPIO 설정 초기화
```

2-3. 실행결과

-동영상 참고-

3-1 실습문제

[자동차 움직이기 구현_17p]

1. 오른쪽 모터부분의 코드를 추가하여 정방향으로 50%로 동작->정지->동작->정지

2. 스위치를 입력 받아 자동차 조종하기
   SW1 : 앞
   SW2 : 오른쪽
   SW3 : 왼쪽
   SW4 : 뒤
   print문을 사용하여 어느 스위치가 눌렸는지 출력

3-2. 소스코드

```
# 3_4.py (1번 문제)
import RPi.GPIO as GPIO  # 라즈베리파이 GPIO 제어 모듈
import time  # 시간 제어용 모듈
# === 핀 번호 설정 ===
PWMA =18
AIN1 =22
AIN2 =27
PWMB =23
BIN1 =25
BIN2 =24
# === GPIO 기본 설정 ===
GPIO.setwarnings(False)  # 경고 메시지 비활성화
GPIO.setmode(GPIO.BCM)  # BCM 핀 번호 체계 사용
GPIO.setup(PWMA, GPIO.OUT)
GPIO.setup(AIN1, GPIO.OUT)
GPIO.setup(AIN2, GPIO.OUT)
GPIO.setup(PWMB, GPIO.OUT)
GPIO.setup(BIN1, GPIO.OUT)
GPIO.setup(BIN2, GPIO.OUT)
# === PWM 객체 생성 ===
L_Motor = GPIO.PWM(PWMA, 500)  # 오른쪽 모터
R_Motor = GPIO.PWM(PWMB, 500)  # 왼쪽 모터
L_Motor.start(0)  # 초기 정지 상태
R_Motor.start(0)
try:
    while True:
        # 정방향 회전
        GPIO.output(AIN1, 0)
        GPIO.output(AIN2, 1)
```

```
            L_Motor.ChangeDutyCycle(100)
            GPIO.output(BIN1, 0)
            GPIO.output(BIN2, 1)
            R_Motor.ChangeDutyCycle(100)
            time.sleep(1.0)  # 1초 동안 전진
            # 정지
            L_Motor.ChangeDutyCycle(0)
            R_Motor.ChangeDutyCycle(0)
            time.sleep(1.0)
            # 동일 동작 반복 (두 번째 주기)
            GPIO.output(AIN1, 0)
            GPIO.output(AIN2, 1)
            L_Motor.ChangeDutyCycle(100)
            GPIO.output(BIN1, 0)
            GPIO.output(BIN2, 1)
            R_Motor.ChangeDutyCycle(100)
            time.sleep(1.0)
            L_Motor.ChangeDutyCycle(0)
            R_Motor.ChangeDutyCycle(0)
            time.sleep(1.0)
except KeyboardInterrupt:  # Ctrl+C 시 종료
    pass
GPIO.cleanup()  # GPIO 설정 초기화
# ============================
# 3_4_2.py (2번 문제)
# ============================
import RPi.GPIO as GPIO
import time
# === 핀 번호 정의 ===
BTN_FWD =5       # 전진 버튼
BTN_RIGHT =6     # 우회전 버튼
BTN_LEFT =13     # 좌회전 버튼
BTN_BACK =19     # 후진 버튼
PWM_RIGHT =18    # 오른쪽 모터 PWM 핀
PWM_LEFT =23     # 왼쪽 모터 PWM 핀
AIN1, AIN2 =22, 27  # 오른쪽 모터 방향 제어
BIN1, BIN2 =25, 24  # 왼쪽 모터 방향 제어
# === GPIO 설정 ===
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
# 버튼 핀 입력 설정 (풀다운 저항)
for btn in [BTN_FWD, BTN_RIGHT, BTN_LEFT, BTN_BACK]:
```

```
    GPIO.setup(btn, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
# 모터 제어 핀 출력 설정
for pin in [PWM_RIGHT, PWM_LEFT, AIN1, AIN2, BIN1, BIN2]:
    GPIO.setup(pin, GPIO.OUT)
# === PWM 초기화 ===
right_motor = GPIO.PWM(PWM_RIGHT, 250)  # 오른쪽 모터 (250Hz)
left_motor = GPIO.PWM(PWM_LEFT, 250)    # 왼쪽 모터 (250Hz)
right_motor.start(0)  # 초기 정지
left_motor.start(0)
last_pressed =None  # 마지막으로 눌린 버튼 저장
# === 모터 제어 함수 ===
def drive_motor(a1, a2, b1, b2, duty_a, duty_b):
    GPIO.output(AIN1, a1)
    GPIO.output(AIN2, a2)
    GPIO.output(BIN1, b1)
    GPIO.output(BIN2, b2)
    right_motor.ChangeDutyCycle(duty_a)
    left_motor.ChangeDutyCycle(duty_b)
try:
    while True:
        if GPIO.input(BTN_FWD):  # 전진 버튼
            if last_pressed !='fwd':
                print("sw1")  # 전진 출력 표시
                last_pressed ='fwd'
            drive_motor(0, 1, 0, 1, 50, 50)  # 양쪽 모터 전진
        elif GPIO.input(BTN_RIGHT):  # 우회전 버튼
            if last_pressed !='right':
                print("sw2")
                last_pressed ='right'
            drive_motor(0, 1, 0, 1, 50, 25)  # 오른쪽 강하게, 왼쪽 약하게
        elif GPIO.input(BTN_LEFT):  # 좌회전 버튼
            if last_pressed !='left':
                print("sw3")
                last_pressed ='left'
            drive_motor(0, 1, 0, 1, 25, 50)  # 왼쪽 강하게, 오른쪽 약하게
        elif GPIO.input(BTN_BACK):  # 후진 버튼
            if last_pressed !='back':
                print("sw4")
                last_pressed ='back'
            drive_motor(1, 0, 1, 0, 50, 50)  # 양쪽 모터 반대 방향
        else:  # 모든 버튼이 눌리지 않았을 때
            if last_pressed is not None:
```

```
                drive_motor(0, 0, 0, 0, 0, 0)  # 정지
                last_pressed =None
        time.sleep(0.02)  # 빠른 반복 방지를 위한 짧은 대기
except KeyboardInterrupt:
    pass
GPIO.cleanup()  # GPIO 설정 초기화
```

3-3. 실행결과

문제 1번 -동영상 참고-	문제 2번  print문으로 어느 스위치가 눌렸는지 출력