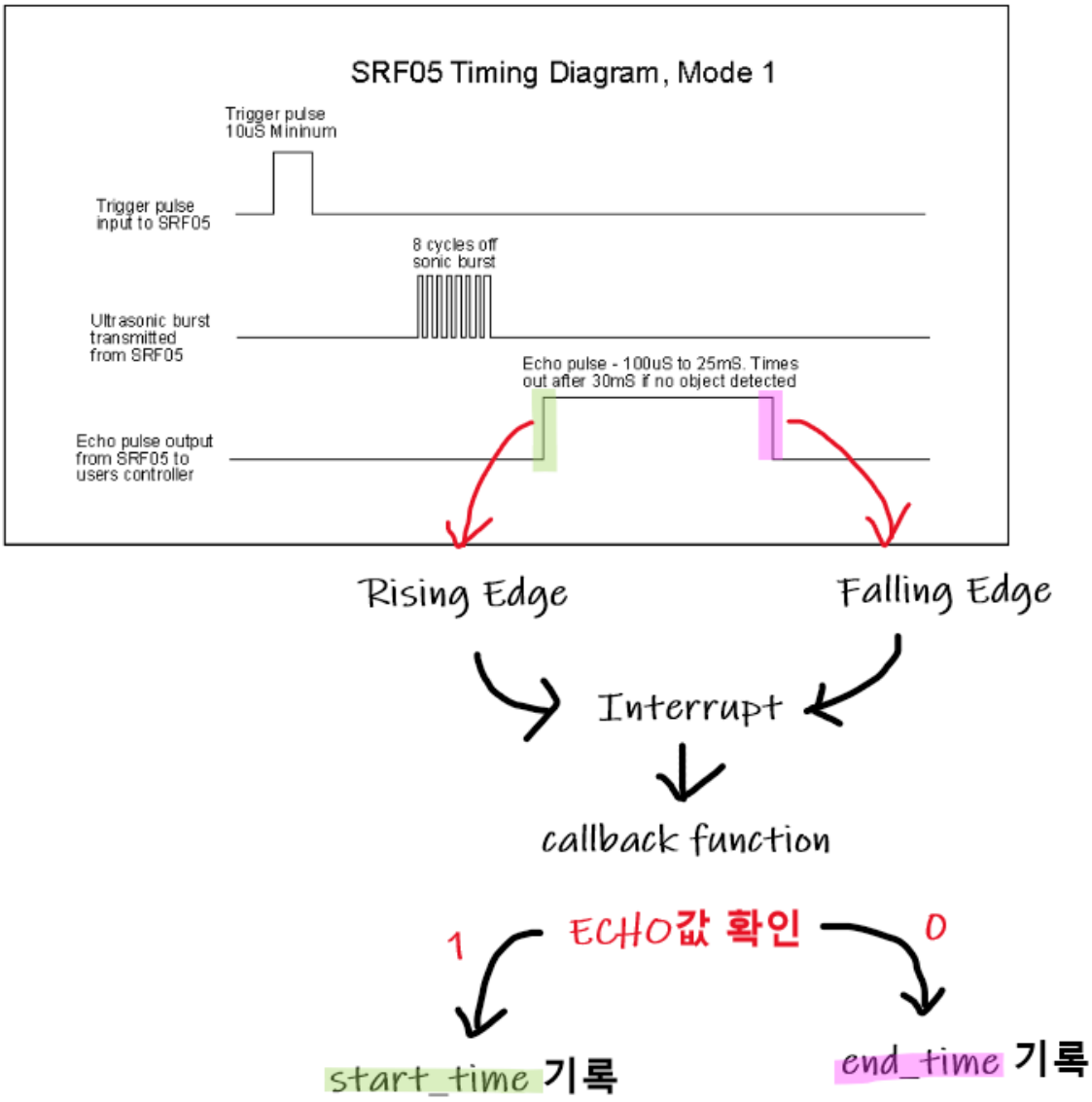
[ESD 2019-2] 도전과제#1 - 20142697 권민수

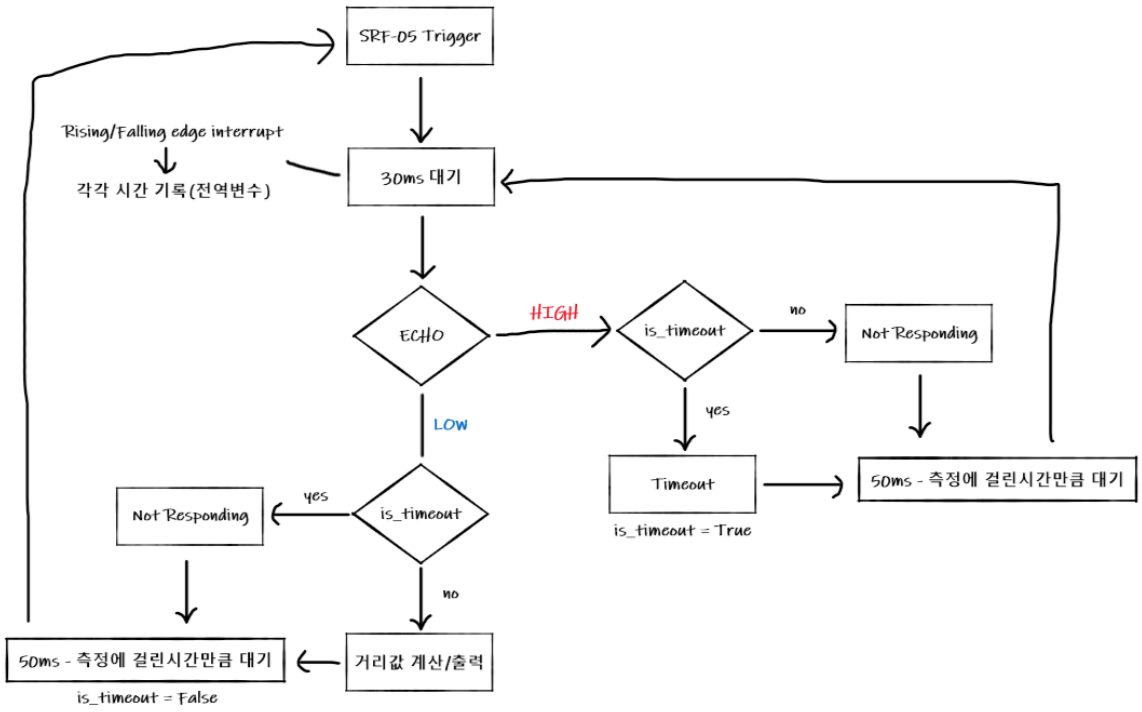
구현방식

인터럽트



- Echo pulse의 **Rising Edge**
  - start_time 기록 후 return
- Echo pulse의 **Falling Edge**
  - end_time 기록 후 return

거리측정 Flowchart



- 위 과정을 전체제한시간동안 반복

wiringpi

microsecond 단위 시간 측정을 위해 wiringpi 라이브러리 사용

- `micros()`
  - 최초로 `wiringPiSetup()` 한 뒤 흐른 시간을 microsecond 단위로 리턴
  - 이를 이용해 microsecond 단위로 시간이 얼마나 흘렀는지 계산 가능
- `delayMicroseconds(uint howLong)`
  - 프로그램의 실행흐름을 `howLong` microsecond만큼 중단
  - 트리거할 때, 다음 트리거까지 기다려야할 때 사용

소스코드

```
import RPi.GPIO as GPIO
import wiringpi

TRIG = 17
ECHO = 18
SECONDS = 60
start_time, end_time, measure_start_time, count, dist = 0,0,0,0,0

def record_edge_time(channel):
    global start_time, end_time

    # Record rising edge
    if (GPIO.input(ECHO)):
        start_time = wiringpi.micros()
        return
    # Record falling edge
    else:
        end_time = wiringpi.micros()
```

```
        return

# GPIO setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

# wiringpi setup
if (wiringpi.wiringPiSetup() == -1):
    exit(1)

# register interrupt handler
GPIO.add_event_detect(ECHO, GPIO.BOTH, callback = record_edge_time)

# record prgoram start time
program_start_time = wiringpi.micros()

# record measure start time
measure_start_time = wiringpi.micros()

# trigger
GPIO.output(TRIG, 0)
wiringpi.delayMicroseconds(20)
GPIO.output(TRIG, 1)
wiringpi.delayMicroseconds(10)
GPIO.output(TRIG, 0)

is_timeout = False
end = False

while (not end):
    # wait 30 ms
    wiringpi.delayMicroseconds(30000)

    # time over -> break
    if (wiringpi.micros() - program_start_time > SECONDS * 1000000):
        break

    count = count + 1

    # if echo still HIGH
    if (GPIO.input(ECHO)):
        # if it's first time, it's timeout
        if (not is_timeout):
            print count, '\t', round(dist, 2), '\tcm TO'
            is_timeout = True
        # if already timeout, it's not responding
        else:
            print count, '\t', round(dist, 2), '\tcm NR'

        measure_time = wiringpi.micros() - measure_start_time
        # make sure the sampling rate is 50 ms
        if (measure_time < 50000):
            wiringpi.delayMicroseconds(50000-(measure_time))

        measure_start_time = wiringpi.micros()

    else:
```

```
        elapsed_time = end_time - measure_start_time

        # if it was timeout, not correctly measured
        if (is_timeout):
            print count, '\t', round(dist, 2), '\tcm NR'
        else:
            # correctly measured, distance value update
            dist = elapsed_time*340/2*100*0.000001
            print count, '\t', round(dist, 2), '\tcm'

        measure_time = wiringpi.micros() - measure_start_time
        if (measure_time < 50000):
            wiringpi.delayMicroseconds(50000-(measure_time)-125)

        is_timeout = False
        # check for time over
        end = True if (wiringpi.micros() - program_start_time > SECONDS *
1000000) else False

        measure_start_time = wiringpi.micros()

        GPIO.output(TRIG, 0)
        wiringpi.delayMicroseconds(20)
        GPIO.output(TRIG, 1)
        wiringpi.delayMicroseconds(10)
        GPIO.output(TRIG, 0)

GPIO.cleanup()
```