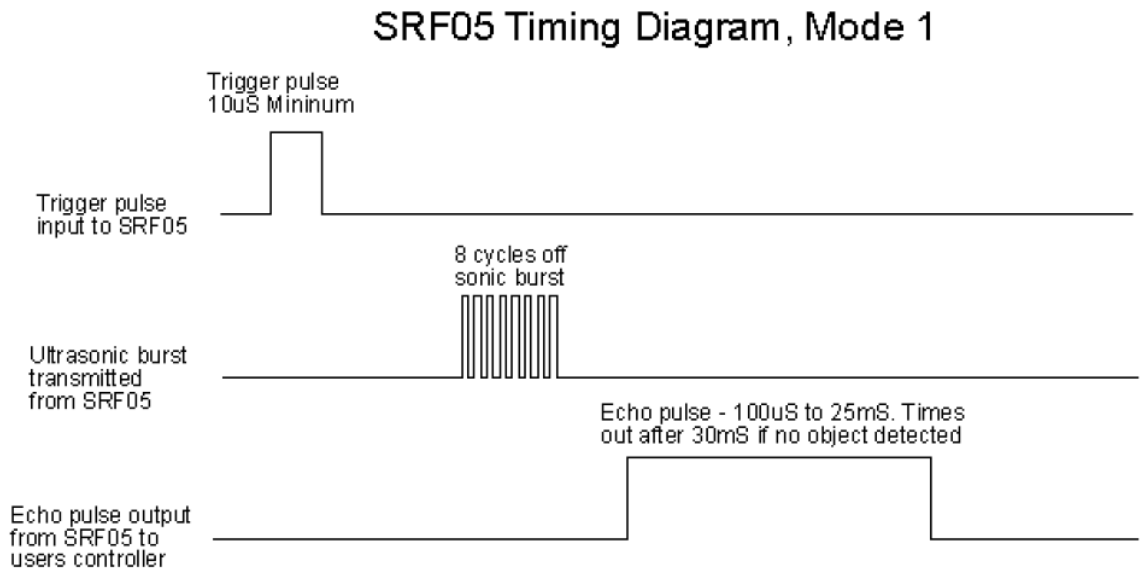
[ESD 2019-2] 도전과제#1 - 20142697 권민수

구현방식

[SRF-05 Timing Diagram]



인터럽트

- Echo pulse의 **Rising Edge**
  - 트리거에 반응하여 **거리측정이 시작**됐다는 뜻
  - 시작 시간을 기록한 후 종료
- Echo pulse의 **Falling Edge**
  - **거리측정이 끝**났다는 뜻
  - 종료 시간을 기록, 시작 시간과의 비교를 통해 얼마동안의 시간이 흘렀는지 계산
    - 해당 시간 값으로 거리 계산 및 출력
  - 다음 측정을 위해 트리거

시간 문제

- SRF-05의 데이터시트에 의하면, 50ms에 한번 트리거 가능
  - 트리거 했을때 시간을 기록하고, 측정이 끝났을 때 시간과 비교
  - 50ms의 시간이 흐르지 않았으면, (**50ms - 측정에 걸린시간**)만큼 추가로 대기한 후 트리거

거리 측정 오류 문제

- SRF-05의 데이터시트에 의하면, SRF-05가 echo 라인을 1로 올린 뒤,
  30ms 동안 아무것도 탐지되지 않으면 echo가 자동으로 0으로 돌아오게 됨
  - 이를 이용해 **echo가 1에서 0이 될 때까지의 시간이 30ms 이상 흘렀다면**, 이는 Timeout에 해당
  - 거리값을 갱신하지 않고, 이전에 측정한 거리값과 오류 메세지 출력

wiringpi

보다 정확한 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

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

# ECHO의 RISING/FALLING EDGE에서 호출될 콜백함수
def distance(channel):
    # global 변수로 사용
    global start_time, end_time, measure_start_time, count, dist

    # RISING EDGE일시(ECHO의 input값이 1)
    if (GPIO.input(ECHO)):
        # 현재 시간을 기록한 후 리턴
        start_time = wiringpi.micros()
        return

    # FALLING EDGE일시 (ECHO의 input값이 0)
    else:
        end_time = wiringpi.micros()

        count = count + 1
        elapsed_time = end_time - start_time

        # ECHO가 1에서 0이되는 시간이 30ms 이상인 경우
        if (elapsed_time >= 30000):
            # 이전 거리값 출력, TimeOut
            print (dist, 'cm, TO(', elapsed_time*0.001, ') ms')

        # 정상적으로 거리가 측정된 경우
        else:
            # 거리값 새로 저장, 출력
            dist = (elapsed_time)*340/2*100*0.000001
            print dist, 'cm'

        # 새로 측정을 시도하기 전, ECHO값 재확인
        if (GPIO.input(ECHO)):
            # ECHO값이 1일 시 이전 거리값 출력, Sensor Not Responding
```

```
            print dist, 'cm, NR'
            return

        # 최근 트리거 이후 50ms 지났는지 확인, 안 지났으면 해당 시간만큼 추가로 흘려보냄
        measure_time = wiringpi.micros()-measure_start_time
        # 125는 경험적으로 얻은값 : 20초마다 측정한번이 밀림
        #                          -> 50000/400 = 125만큼 매번 덜 delay
        if (measure_time < 50000-125):
            wiringpi.delayMicroseconds(50000-(measure_time)-125)

        # 다음 측정을 위해 trigger
        GPIO.output(TRIG, 1)
        wiringpi.delayMicroseconds(10)
        GPIO.output(TRIG, 0)
        # trigger한 시간을 기록
        measure_start_time = wiringpi.micros()
        return

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

GPIO.add_event_detect(ECHO, GPIO.BOTH, callback = distance)

# 최초 trigger
GPIO.output(TRIG, 1)
wiringpi.delayMicroseconds(10)
GPIO.output(TRIG, 0)
# trigger 시간 기록
measure_start_time = wiringpi.micros()

# 지정한 시간만큼 측정대기 (sleep)
wiringpi.delayMicroseconds(SECONDS * 1000000)
print 'measure count :', count, 'in', SECONDS, 'seconds'

GPIO.cleanup()
```

```
pi@raspberrypi:~ $ python uss_chal.py
```

.

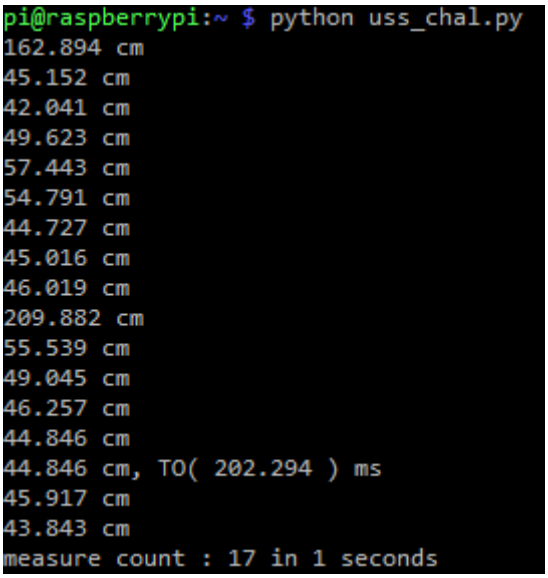
.

.

```
156.91 cm
157.386 cm
157.097 cm
156.689 cm
157.794 cm
158.525 cm
157.386 cm
157.403 cm
157.318 cm
156.859 cm
156.366 cm
measure count : 1200 in 60 seconds
pi@raspberrypi:~ $
```

문제점 / 더 생각해봐야 할 점

- 타임아웃 발생시, 30 ms가 아니라 훨씬 더 길게 시간이 낭비되는 문제가 있다



  - 해당 제품만의 문제인지, 전체 모델의 문제인지, 또는 프로그램의 문제인지 확인 필요
- 최근 트리거 시간과 현재시간을 비교해 정확히 50ms 만큼 트리거 간격을 유지하려 했지만, 20초마다 1번의 측정이 밀리는 문제 발생
  - 매번 125 microsecond 만큼 덜 delay함으로써 문제를 해결했지만, 임시적인 대처일뿐이다