[ESD 2019-2] 도전과제 #1

20171661 이다은

1. 제출 코드 및 결과

1) 제출 코드

```
#!/usr/bin/env python

import RPi.GPIO as GPIO
import time
from threading import Timer,Thread,Event

TRIG = 11
ECHO = 12
echo_start = 0
pre_dis = 0
case_cnt = 0

def setup():
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(TRIG, GPIO.OUT)
    GPIO.setup(ECHO, GPIO.IN)

def my_interrupt(ECHO):
    if GPIO.input(ECHO):  # ECHO rising
        global echo_start
        echo_start  = time.time()
    else: # ECHO falling
        echo_during = time.time() - echo_start
        if echo_during <= 0.03: # get distance
            global pre_dis
            dis =  echo_during * 340 / 2 * 100
            pre_dis = dis
            print "count:", case_cnt, ", ", "  ", ", distance:", dis
        else: # time out
            print "count:", case_cnt, ", ", "TO", ", distance:", pre_dis

def trigger():
    GPIO.output(TRIG, 0)
    time.sleep(0.000002)

    if GPIO.input(ECHO):  # still ECHO = 1
        print "count:", case_cnt, ", ", "NR", ", distance:", pre_dis
    else: # ECHO = 0
        GPIO.output(TRIG, 1)
        time.sleep(0.00001)
        GPIO.output(TRIG, 0)
```

```
def loop():
    loop_start = time.time()
    while time.time() - loop_start <= 60:
        distance_start = time.time()
        global case_cnt
        case_cnt += 1
        trigger()
        distance_during = 0.05 - (time.time() - distance_start)
        time.sleep(distance_during)

def destroy():
    GPIO.cleanup()

if __name__ == "__main__":
    setup()
    GPIO.add_event_detect(ECHO, GPIO.BOTH, callback=my_interrupt)
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

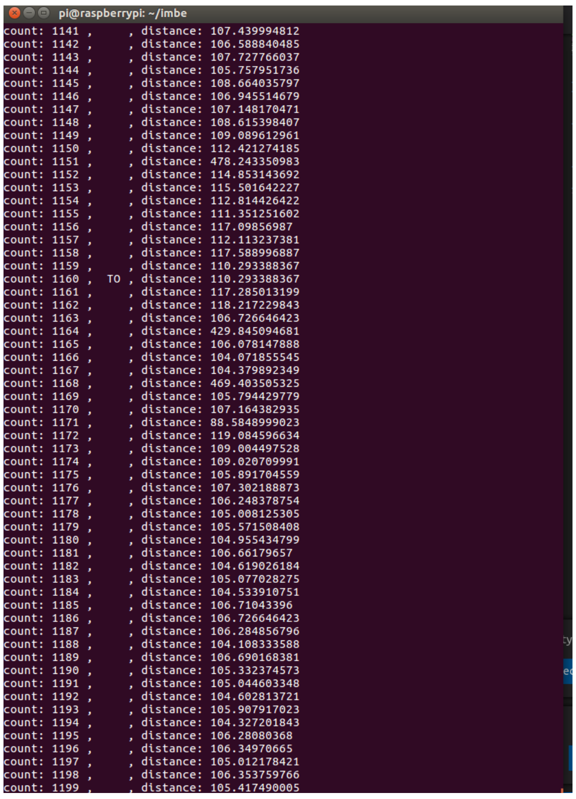
2)결과

- interrupt 방식

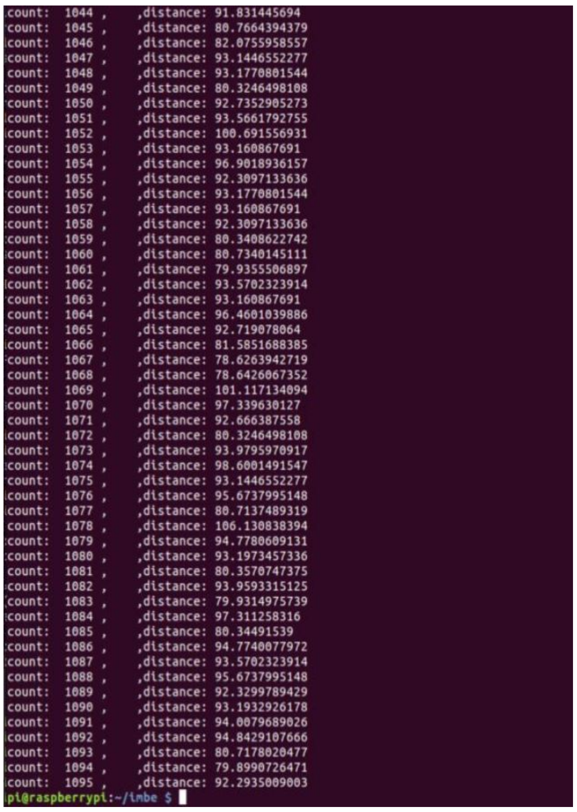
  총 count: 1199 (장애물 1m 기준)

- polling 방식

  총 count: 1095 (장애물 1m 기준)



```
pi@raspberrypi: ~/imbe
count: 1141 ,      , distance: 107.439994812
count: 1142 ,      , distance: 106.588840485
count: 1143 ,      , distance: 107.727766037
count: 1144 ,      , distance: 105.757951736
count: 1145 ,      , distance: 108.664035797
count: 1146 ,      , distance: 106.945514679
count: 1147 ,      , distance: 107.148170471
count: 1148 ,      , distance: 108.615398407
count: 1149 ,      , distance: 109.089612961
count: 1150 ,      , distance: 112.421274185
count: 1151 ,      , distance: 478.243350983
count: 1152 ,      , distance: 114.853143692
count: 1153 ,      , distance: 115.501642227
count: 1154 ,      , distance: 112.814426422
count: 1155 ,      , distance: 111.351251602
count: 1156 ,      , distance: 117.09856987
count: 1157 ,      , distance: 112.113237381
count: 1158 ,      , distance: 117.588996887
count: 1159 ,      , distance: 110.293388367
count: 1160 ,   TO , distance: 110.293388367
count: 1161 ,      , distance: 117.285013199
count: 1162 ,      , distance: 118.217229843
count: 1163 ,      , distance: 106.726646423
count: 1164 ,      , distance: 429.845094681
count: 1165 ,      , distance: 106.078147888
count: 1166 ,      , distance: 104.071855545
count: 1167 ,      , distance: 104.379892349
count: 1168 ,      , distance: 469.403505325
count: 1169 ,      , distance: 105.794429779
count: 1170 ,      , distance: 107.164382935
count: 1171 ,      , distance: 88.5848999023
count: 1172 ,      , distance: 119.084596634
count: 1173 ,      , distance: 109.004497528
count: 1174 ,      , distance: 109.020709991
count: 1175 ,      , distance: 105.891704559
count: 1176 ,      , distance: 107.302188873
count: 1177 ,      , distance: 106.248378754
count: 1178 ,      , distance: 105.008125305
count: 1179 ,      , distance: 105.571508408
count: 1180 ,      , distance: 104.955434799
count: 1181 ,      , distance: 106.66179657
count: 1182 ,      , distance: 104.619026184
count: 1183 ,      , distance: 105.077028275
count: 1184 ,      , distance: 104.533910751
count: 1185 ,      , distance: 106.71043396
count: 1186 ,      , distance: 106.726646423
count: 1187 ,      , distance: 106.284856796
count: 1188 ,      , distance: 104.108333588
count: 1189 ,      , distance: 106.690168381
count: 1190 ,      , distance: 105.332374573
count: 1191 ,      , distance: 105.044603348
count: 1192 ,      , distance: 104.602813721
count: 1193 ,      , distance: 105.907917023
count: 1194 ,      , distance: 104.327201843
count: 1195 ,      , distance: 106.28080368
count: 1196 ,      , distance: 106.34970665
count: 1197 ,      , distance: 105.012178421
count: 1198 ,      , distance: 106.353759766
count: 1199 ,      , distance: 105.417490005
```



```
count:  1044 ,     ,distance: 91.831445694
count:  1045 ,     ,distance: 80.7664394379
count:  1046 ,     ,distance: 82.0755958557
count:  1047 ,     ,distance: 93.1446552277
count:  1048 ,     ,distance: 93.1770801544
count:  1049 ,     ,distance: 80.3246498108
count:  1050 ,     ,distance: 92.7352905273
count:  1051 ,     ,distance: 93.5661792755
count:  1052 ,     ,distance: 100.691556931
count:  1053 ,     ,distance: 93.160867691
count:  1054 ,     ,distance: 96.9018936157
count:  1055 ,     ,distance: 92.3097133636
count:  1056 ,     ,distance: 93.1770801544
count:  1057 ,     ,distance: 93.160867691
count:  1058 ,     ,distance: 92.3097133636
count:  1059 ,     ,distance: 80.3408622742
count:  1060 ,     ,distance: 80.7340145111
count:  1061 ,     ,distance: 79.9355506897
count:  1062 ,     ,distance: 93.5702323914
count:  1063 ,     ,distance: 93.160867691
count:  1064 ,     ,distance: 96.4601039886
count:  1065 ,     ,distance: 92.719078064
count:  1066 ,     ,distance: 81.5851688385
count:  1067 ,     ,distance: 78.6263942719
count:  1068 ,     ,distance: 78.6426067352
count:  1069 ,     ,distance: 101.117134094
count:  1070 ,     ,distance: 97.339630127
count:  1071 ,     ,distance: 92.666387558
count:  1072 ,     ,distance: 80.3246498108
count:  1073 ,     ,distance: 93.9795970917
count:  1074 ,     ,distance: 98.6001491547
count:  1075 ,     ,distance: 93.1446552277
count:  1076 ,     ,distance: 95.6737995148
count:  1077 ,     ,distance: 80.7137489319
count:  1078 ,     ,distance: 106.130838394
count:  1079 ,     ,distance: 94.7780609131
count:  1080 ,     ,distance: 93.1973457336
count:  1081 ,     ,distance: 80.3570747375
count:  1082 ,     ,distance: 93.9593315125
count:  1083 ,     ,distance: 79.9314975739
count:  1084 ,     ,distance: 97.311258316
count:  1085 ,     ,distance: 80.34491539
count:  1086 ,     ,distance: 94.7740077972
count:  1087 ,     ,distance: 93.5702323914
count:  1088 ,     ,distance: 95.6737995148
count:  1089 ,     ,distance: 92.3299789429
count:  1090 ,     ,distance: 93.1932926178
count:  1091 ,     ,distance: 94.0079689026
count:  1092 ,     ,distance: 94.8429107666
count:  1093 ,     ,distance: 80.7178020477
count:  1094 ,     ,distance: 79.8990726471
count:  1095 ,     ,distance: 92.2935009003
pi@raspberrypi:~/imbe $ 
```

동일한 환경(장애물 전방 1m 존재)에서 1분간 20Hz로 거리측정 프로그램을 실행 하였을 때, interrupt 방식이 polling 방식보다 측정 count 수가 약 100개 많습니다.

2. 코드 설명

1) my_interrupt(pin), add_event_detect(pin, edge detect option, callback=)

```
def my_interrupt(ECHO):
    if GPIO.input(ECHO):  # ECHO rising
        global echo_start
        echo_start  = time.time()
    else: # ECHO falling
        echo_during = time.time() - echo_start
        if echo_during <= 0.03: # get distance
            global pre_dis
            dis =  echo_during * 340 / 2 * 100
            pre_dis = dis
            print "count:", case_cnt, ", ", "  ", ", distance:", dis
        else: # time out
            print "count:", case_cnt, ", ", "TO", ", distance:", pre_dis

if __name__ == "__main__":
    setup()
    GPIO.add_event_detect(ECHO, GPIO.BOTH, callback=my_interrupt)
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

  GPIO 라이브러리의 add_event_detect()과 콜백함수를 활용하여 ISR(interrupt service routine)을 구현하였습니다. ECHO(pin 12)의 edge가 rising할 때 interrupt가 발생하면 time 측정을 시작합니다. ECHO(pin 11)의 edge가 falling할 때 interrupt가 발생하면, 측정한 plus 시간의 임계치(30ms)에 따라 분기합니다. 임계치를 넘지 않으면 거리를 갱신하여 출력하고, 넘는다면 time out과 거리의 이전값을 출력합니다.

2) trigger()

```
def trigger():
    GPIO.output(TRIG, 0)
    time.sleep(0.000002)

    if GPIO.input(ECHO):  # still ECHO = 1
        print "count:", case_cnt, ", ", "NR", ", distance:", pre_dis
    else: # ECHO = 0
        GPIO.output(TRIG, 1)
        time.sleep(0.00001)
        GPIO.output(TRIG, 0)
```

ECHO(pin 12)의 값이 여전히 1이라면, 거리 측정 중에 있다는 의미 이므로 sensor not responding을 출력합니다.

3) loop()

```
def loop():
    loop_start = time.time()
    while time.time() - loop_start <= 60:
        distance_start = time.time()
        global case_cnt
        case_cnt += 1
        trigger()
        distance_during = 0.05 - (time.time() - distance_start)
        time.sleep(distance_during)
```

본 프로그램의 sample rate은 20Hz이기 때문에 50ms마다 TRIG(pin 11)를 rising 시켜 초음파를 내보내게 합니다. Thread를 사용하여 정확히 매 50ms마다 TRIG(pin 11)를 rising 시키고자 노력했으나, 결론적으로 함수사용에 대한 시간을 합하여 50ms가 되도록 하였습니다.