**네트워크 프로토콜 설계 및 실습 1주차 과제**

2016-10769 이권형

**1. blink-hello.c의 작동 순서 및 동작 원리에 대한 분석**

1) hello\_world\_process

hello\_world\_process의 경우, 가장 처음에 etimer\_set 함수를 통해 timer를 CLOCK\_CONF\_SECOND \* 4 로 설정하고, 이에 timer는 CLOCK\_CONF\_SECOND \* 4의 시간 이후에 PROCESS\_EVENT\_TIMER에 해당하는 event를 발생시킨다.

이후 while loop에서, PROCESS\_WAIT\_EVENT() 함수는 event가 발생할 때까지 process를 대기시키고, event가 발생하면 진행한다. if문을 통해 event가 PROCESS\_EVENT\_TIMER일 경우, printf 함수를 통해 정해진 message와 count를 출력하고, count를 1 증가시킨다. 이후 etimer\_reset()함수를 통해 timer를 다시 CLOCK\_CONF\_SECOND \* 4로 설정한다.

While문 내부에서 위와 같은 과정이 반복된다.

2) blink\_process

Blink\_process의 경우, hello\_world\_process와 유사하지만 PROCESS\_WAIT\_EVENT()함수 대신 PROCESS\_WAIT\_EVENT\_UNTIL()함수를 사용한다는 점에서 차이가 있다. PROCESS\_WAIT\_EVENT()함수의 경우 주어진 ev가 발생할 때까지 process를 대기하도록 한다.

Hello\_world\_process의 경우, PROCESS\_WAIT\_EVENT()함수를 사용했기 때문에 event의 종류가 PROCESS\_EVENT\_TIMER가 아니어도 계속하여 process를 진행시키기 때문에 if block 뒤의 부분이 실행된다. 따라서 다른 process에 의해 event가 전달되었을 경우, ev가 PRCESS\_EVENT\_TIMER가 아니기 때문에 if block후에 code가 있었다면 이것이 실행되었을 것이다. 이에 반해 blink\_process의 경우 PROCESS\_WAIT\_EVENT\_UNTIL()을 사용하여, ev가 PROCESS\_EVENT\_TIMER일 때만 process를 재개한다. 따라서 timer가 event를 발생시키기 전까지는 process가 진행하지 않는다.

**2. Sensing Application에서 인위적으로 조도를 조절하여 조도 값의 변화를 시간에 따른 그래프로 나타내기**

Sensing app의 Source code는 다음과 같다.

#include "contiki.h"

#include "dev/leds.h"

#include "dev/light-sensor.h"

#include <stdio.h>

/\*---------------------------------------------------------------------------\*/

PROCESS(light\_process, "light sensor process");

AUTOSTART\_PROCESSES(&light\_process);

/\*---------------------------------------------------------------------------\*/

/\*

\* This function return calculated sensor value (lux)

\* maybe unit is incorrect..cannot find reference

\*/

static int get\_light(void) {

// Return calculated value

return 10 \* light\_sensor.value(LIGHT\_SENSOR\_PHOTOSYNTHETIC) / 7;

}

/\*---------------------------------------------------------------------------\*/

/\*

\* This process get value from sensor every 1 second

\*/

PROCESS\_THREAD(light\_process, ev, data)

{

static struct etimer timer;

static int count = 0;

int light = 0;

static char buf[100];

PROCESS\_BEGIN();

// First, Activate sensor

SENSORS\_ACTIVATE(light\_sensor);

while(1) {

// Set interval as 1 second

etimer\_set(&timer, CLOCK\_CONF\_SECOND);

// Wait etimer

PROCESS\_WAIT\_EVENT\_UNTIL(ev == PROCESS\_EVENT\_TIMER);

    light = get\_light();

// print second and light

    printf("%d %d\n", count, light);

count ++;

}

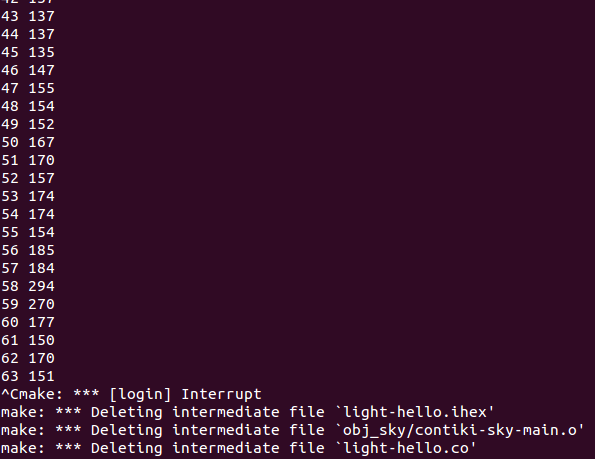
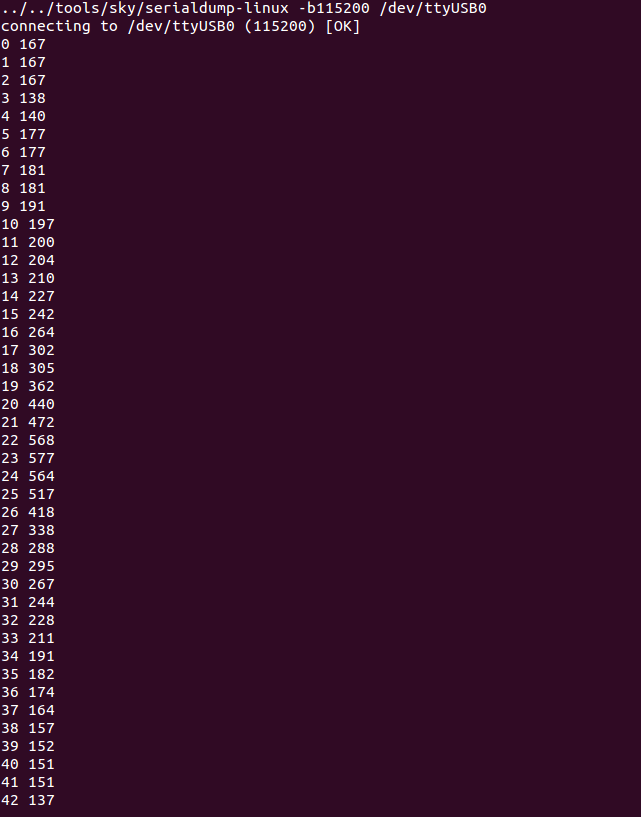
PROCESS\_END();

}

/\*---------------------------------------------------------------------------\*/

**그림1. light-hello.c의 code**

출력 화면은 다음과 같다.



**그림2. Sensing application의 실행 결과**

왼쪽의 숫자는 시간(s)을 나타내고, 오른 쪽의 숫자는 조도를 나타낸다. 이를 file로 만들어 excel을 이용해 그래프를 작성하였다.

그래프는 다음과 같다.

**그림3. 조도 측정 그래프**

위의 그래프를 바탕으로 결과를 해석할 수 있다. 측정시에, 처음에는 적당한 장소에서 손을 대지 않고 시작하였다. 이후 손을 대서 조도를 낮춘 후, 광원을 향해 서서히 가까이 하면서 조도를 높여보았다. 다시 광원으로부터 멀리 떨어지면서 조도를 낮추고, 이와 같은 과정을 반복하여 위와 같은 그래프를 얻을 수 있었다.

k-mote의 경우 sensor에서 value를 Voltage로 return하기 때문에, 이를 lux로 바꾸어 주기 위해 10을 곱하고 7로 나누었다.

과제를 통하여 주기적으로 조도를 측정하는 process를 성공적으로 작성할 수 있었다.