강의명: 임베디드 시스템
숙제 번호: 7
숙제 제목: Real time programming(실시간 프로그래밍)
학생 이름: 한규현(팀장), 전세환

1. 프로그램 multi-threading

1.1 프로그램 코드 쓰기

```
#include "mbed.h"

Serial pc (USBTX , USBRX , 115200 ); // baud rate 115200
Thread thread1, thread2, thread3; // Three threads

void thread_1 ()
{
   static int t =0 ;
   while (true ) {
     printf ("\r\n [%d]: ", t ++);ThisThread ::sleep_for (1000 );
   }
}

void thread_2 ()
{
   while (true ) {
     ThisThread ::sleep_for (2000 );
     printf ("thread_2 ");
   }
}

void thread_3 ()
{
   while (true ) {
     ThisThread ::sleep_for (3000 );
     printf ("thread_3 ");
   }
}

int main ()
{
```

```
    thread1 .start (thread_1);
    thread_sleep_for (10 );
    thread2 .start (thread_2);
    thread_sleep_for (10 );
    thread3 .start (thread_3);
    thread_sleep_for (20 *1000 );
    thread3 .terminate (); thread2 .terminate (); thread1 .terminate ();

    while (true );
}
```

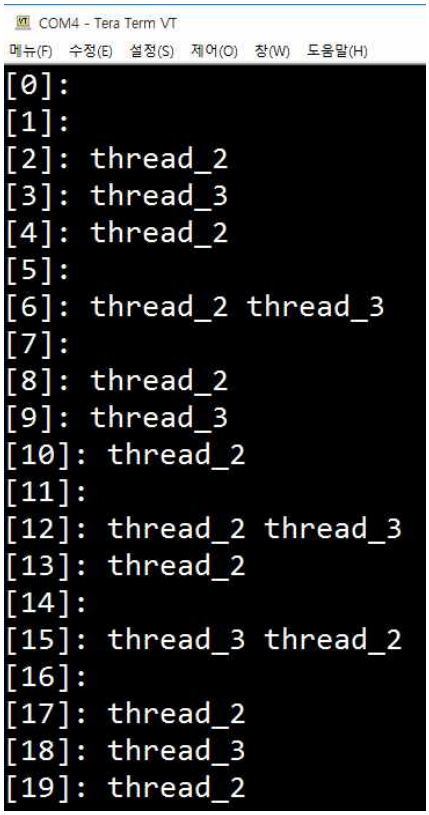
1.2 프로그램 작성 아이디어 혹은 이유 설명 쓰기

thread_1의 경우 1초마다 [%t]:, t는 시간을 출력한다. thread_2는 2초마다, thread_3를 thread_3는 3초마다 출력한다. 각 thread를 sleep하기 위해 ThisThread::sleep_for()을 사용했다. main에서는 약간에 시간을 두고 thread_1~3를 생성했고, 20초 뒤, thread_1~3를 terminate를 사용해서 종료했다.

1.3 하드웨어 구성 사진 첨부하기

1.4 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기

https://youtu.be/q3jj74KjUXs



2. 프로그램 button-isr

2.1 프로그램 코드 쓰기

```
#include "mbed.h"
#include "C12832.h"

C12832 lcd (D11 , D13 , D12 , D7 , D10 ); // lcd = (MOSI, SCK, RESET, A0,
nCS)
PwmOut led_red (D5 ); // led_r = (Red LED)
PwmOut led_green (D9 ); // led_g = (Red LED)

InterruptIn sw2 (SW2 ); // sw2 = SW2
InterruptIn sw3 (SW3 ); // sw3 = SW3

int count_sw2 =0 ; int count_sw3 =0 ;

void ISR_sw2 () {
   led_red =0 ; // on led_red
   led_green =1 ; // off led_green
```

```
   count_sw2 ++; // increment count_sw2
}

void ISR_sw3 () {
   led_red =1 ; // on led_red
   led_green =0 ; // off led_green
   count_sw3 ++; // increment count_sw2
}

int main ()
{

sw2 .rise (&ISR_sw2); // attach ISR_sw2 to sw2
led_red =0 ;
sw3 .fall (&ISR_sw3); // attach ISR_sw3 to sw3
led_green =0 ;

int count =0 ;

while (true ){
   count ++;
   lcd .cls ();
   lcd .locate (0 , 6 );
   lcd .printf ("Button ISRs!");
   lcd .locate (0 , 16 );
   lcd .printf ("Loop=%d, SW2=%d, SW3=%d", count, count_sw2, count_sw3);
   ThisThread ::sleep_for (100 );
   }

}
```

2.2 프로그램 작성 아이디어 혹은 이유 설명 쓰기

C12832 lcd()로 LCD를 설정하고, PwmOut led_red(), PwmOut led_green()으로 각 핀에 연결된 LED를 제어한다. SW2, SW3 버튼에 대한 인터럽트를 처리하는 코드로 InterruptIn sw()을 작성했다. ISR_sw2와 ISR_sw3는 SW2와 SW3버튼의 인터럽트 핸들러로 인터럽트가 생성되면 LED와 카운트 변수를 변경한다. sw2 버튼이 rise가 되면 led_red=0으로 하여 Red LED를 제어하고, sw3 버튼이 fall이 되면 led_green=0으로 하여 Green LED를 제어한다. while에서 loop 값으로 사용할 count 변수를 초기화 한다.

무한 루프에서는 lcd.cls()를 통해 LCD를 클리어 하고, 0,6에 Button ISRs!를 출력하고 0,16에 Loop 값과 SW2, SW3가 눌린 횟수를 100ms 마다 출력하게 했다.

2.3 하드웨어 구성 사진 첨부하기

2.4 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기

https://youtu.be/Sx-Sr2LNWBc

3. 프로그램 digital-clock

3.1 프로그램 코드 쓰기

```
#include "mbed.h"
#include "C12832.h"

C12832 lcd (D11 , D13 , D12 , D7 , D10 );
PwmOut led_r (D5 );
PwmOut led_g (D9 );
InterruptIn sw2 (SW2 ); // sw2 = SW2
InterruptIn sw3 (SW3 );
InterruptIn up (A2 ); // up = A1
InterruptIn down (A3 );
InterruptIn left (A4 );
InterruptIn rite (A5 );
InterruptIn center (D4 );
Timer timer; // timer
int start =0 ;
long offset =0 ;

void ISR_sw2 () {
   if (start ==0 ) { // if not started
     timer .start (); // start timer
     led_g =0 ;
     start =1 ;
```

```
    } else { // else
      timer .stop ();
      led_g =1.0 ;
      start =0 ;
    }
}

void ISR_sw3 (){
    timer .reset ();
    offset =0 ;
}

void ISR_up (){
    offset = offset + (60*1000 );
}

void ISR_down (){
    offset = offset - (60*1000 );
}

void ISR_left (){
    offset = offset - (60*60*1000 );
}

void ISR_rite (){
    offset = offset + (60*60*1000 );
}

void ISR_center (){
    offset = offset - ((timer .read_ms ()+offset) %(1000*60 ));
}

int main ()
{
    long time;
    unsigned char h,m,s,ms;

    led_r = led_g =1.0 ;
    sw2 .fall (&ISR_sw2);
```

```
    sw3 .fall (&ISR_sw3);

    up .rise (&ISR_up);
    down .rise (&ISR_down);
    left .rise (&ISR_left);
    rite .rise (&ISR_rite);
    center .rise (&ISR_center);

    lcd .cls ();
    lcd .locate (0 ,6 );
    lcd .printf ("Digital Clock!");

    while (true ){
      time =timer .read_ms ();
      if ((time + offset) <0 ) offset =-time;
      time = time + offset;

      ms = time %60 ;
      s = (time /1000 ) %60 ;
      m = ((time /1000 ) /60 ) %60 ;
      h = ((time /1000 ) / (60*60 )) %24 ;

      lcd .locate (0 , 16 );
      lcd .printf ("Current Time: %02d:%02d:%02d.%02d", h, m, s, ms);

      thread_sleep_for (100 );
    }

}
```

3.2 프로그램 작성 아이디어 혹은 이유 설명 쓰기

sw2,3을 InterruptIn을 사용하여 attach한다. up,down..center까지 InterruptIn을 사용해 attach한다. sw2의 interrupt가 들어오면 timer를 키게 한다. 만일 start가 0이면 시계가 작동중이지 않으므로 led_g는 0으로 키고 start를 1로 하고 시계를 start한다. 나머지 else일때 stop하고 led_g를 꺼 놓는다. up,down각각 시간 조절을 offset로 조작한다. 예를 들어 left면 1시간을 줄여야 하므로 60*60*1000을 뺀다. main에 h(시간)m(분)s(초)ms를 선언하여 시간을 저장한다. 만일 시간을 조정하다 시간이 음수면 -1을 곱해서 다시 양수로 만든다. h,m,s는 offset을 사용하여 조정해 저장한 후 디스플레이에 출력한다.

3.3 하드웨어 구성 사진 첨부하기

3.4 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기
https://youtu.be/v-4pC0OcJQo

4. 프로그램 mutex-no/mutex-yes

4.1 프로그램 코드 쓰기

mutex-no

```
#include "mbed.h"

Thread thread1, thread2, thread3; // threads


void thread_1 () {
   while (true ) {
     for (int i =0 ; i <50 ; i ++) printf ("1");
     printf ("\r\n ");
   }
}

void thread_2 () {
   while (true ) {
     for (int i =0 ; i <50 ; i ++) printf ("2");
     printf ("\r\n ");
   }
}
```

```
void thread_3 () {
   while (true ) {
     for (int i =0 ; i <50 ; i ++) printf ("3");
     printf ("\r\n ");
   }
}

int main (){
   thread1 .start (thread_1); thread2 .start (thread_2);
   thread3 .start (thread_3);
   while (true ) ;

}
```

mutex-yes

```
#include "mbed.h"

Thread thread1, thread2, thread3; // threads
Mutex mutex; // mutex

void thread_1 () {
   while (true ) {
     mutex .lock ();
     for (int i =0 ; i <50 ; i ++) printf ("1");
     printf ("\r\n ");
     mutex .unlock ();
   }
}

void thread_2 () {
   while (true ) {
     mutex .lock ();
     for (int i =0 ; i <50 ; i ++) printf ("2");
     printf ("\r\n ");
     mutex .unlock ();
   }
}

void thread_3 () {
```

```
    while (true ) {
      mutex .lock ();
      for (int i =0 ; i <50 ; i ++) printf ("3");
      printf ("\r\n ");
      mutex .unlock ();
    }
}

int main (){
    thread1 .start (thread_1); thread2 .start (thread_2);
    thread3 .start (thread_3);
    while (true ) ;

}
```

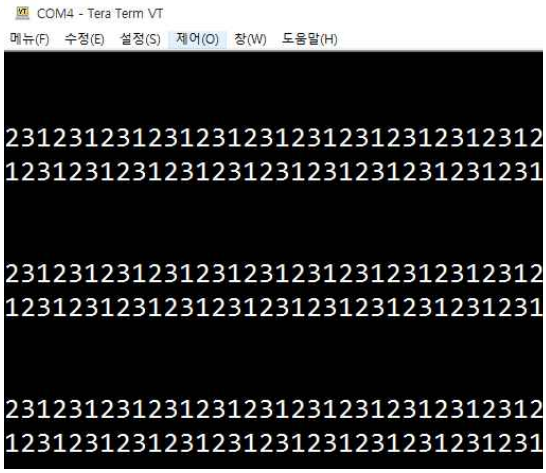
4.2 프로그램 작성 아이디어 혹은 이유 설명 쓰기

화면이라는 공유 자원을 사용하고 있으므로 출력할때 mutex의 lock,unlock으로 화면 자원을 동기화 해준다. lock,unlock을 사용하지 않은 mutex-no는 화면에 출력이 랜덤하게 되지만 mutex-yes는 화면에 한 스레드가 사용 중일때는 다른 스레드가 사용하지 못하게 동기화 하여 일정하게 출력한다. 이를 가능하게 하는 것이 mutex의 사용 여부이다.

4.3 하드웨어 구성 사진 첨부하기

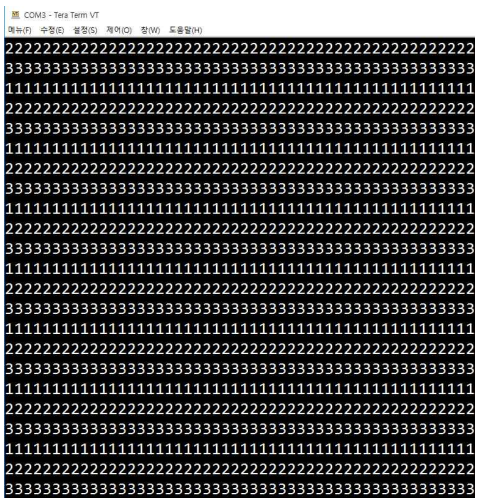
4.4 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기

mutex-no



https://youtu.be/O_w8-bVbVig

mutex-yes



https://youtu.be/dycoN9b8xMs

끝.