강의명: 임베디드 시스템
숙제 번호: 7
숙제 제목:
학생 이름: 오원묵,황대은,정우성
학번: 201810881, 201810897, 201810890

1. 프로그램 multi-threading

1.1 multi-threading 프로그램 코드 쓰기

```
#include "mbed.h"

// https://os.mbed.com/docs/mbed-os/v5.15/tools/creating-a-new-program.html

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()
{
ThisThread::sleep_for(1000);
while (true) {
      printf("thread_2 ");
      ThisThread::sleep_for(2000);
}
}

void thread_3()
{
ThisThread::sleep_for(2000);
while (true) {
      printf("thread_3 ");
      ThisThread::sleep_for(3000);
```

```
}
}

int main()
{
thread1.start(thread_1);
thread_sleep_for(500);
thread2.start(thread_2);
thread_sleep_for(500);
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 는 총 3 개 존재하며, 1 은 순서와 줄바꿈을 1 초당 수행하고, 2 는 thread2 를 2 초, 3 은 thread3 을 3 초 마다 출력합니다.
이 프로그램을 구현하기위해 각각의 thread 를 Thread 로 객체 생성하고, 함수를 만들었습니다. 함수 안에서 ThisThread::sleep_for 이라는 함수를 사용해서 thread_1, thread_2, thread_3 로 각각 1000, 2000, 3000 을 인자로 전달해 알맞은 1 초,2 초,3 초의 시간동안 정지시켰습니다. 그 후 start 라는 멤버함수를 사용해 각 thread 로 함수를 실행했습니다. 각각 thread 를 start 하기 전에 지연을 줘서 시간을 조정했습니다. thread_sleep_for(20 * 1000);에 의해 20 초 동안 각 thread 가 multi-threading 하게 됩니다. 20 초 이 후 terminate 라는 멤버함수에 의해 thread 가 종료됩니다.

1.3 하드웨어 구성 사진 첨부하기

없음

1.4 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기

https://youtube.com/shorts/0E822S6d3KE

2 프로그램 button-isr

2.1 button-isr 프로그램 코드 쓰기

```
#include "mbed.h"
#include "C12832.h"
C12832 lcd(D11, D13, D12, D7, D10);
InterruptIn sw2(SW2);
InterruptIn sw3(SW3);
PwmOut led_R(D5);
PwmOut led_G(D9);
int Loop=0;
int cnt_sw2 = 0;
int cnt_sw3 = 0;
void ISR_sw2() {
    led_R = 0;
    led_G = 1;
    cnt_sw2++;
}
void ISR_sw3() {
    led_R = 1;
    led_G = 0;
    cnt_sw3++;
}
int main()
{
    sw2.rise(&ISR_sw2);
    sw3.fall(&ISR_sw3);
    led_G = 1;
    led_R = 1;
    lcd.cls();
    lcd.locate(0, 6);
    lcd.printf("Button ISRs!");
    while(true){
        lcd.locate(0, 16);
        lcd.printf("Loop=%3d, SW2=%3d, SW3=%3d",Loop,cnt_sw2,cnt_sw3);
        Loop++;
        thread_sleep_for(100);
    }
}
```

2.2 프로그램 작성 아이디어 혹은 이유 설명 쓰기

sw2 와 sw3 를 누르면 interrup 가 발생하고 count 된다. 이것을 LCD 에 출력하게 된다. 이 때 handler 는 ISR_sw2, ISR_sw3 이고, 각각 cnt_sw2, cnt_sw3 라는 전역변수에 1 증가와 led 교체를 한다. rise, fall 이라는 InterrupIn class 에 멤버함수를 통해 handler 로 지정한다. 그 후 C12832 이라는 class 에 멤버함수들을 통해 (0.6)좌표에 "Button ISRs!"라는 문자열을 출력한다. lcd.locate 으로 (0, 16)을 출력 좌표로 지정하고, Loop, cnt_sw2, cnt_sw3 를 출력한다. 이 때 Loop 는 0.1sec 마다 카운터되고, count_sw2, count_sw3 등은 각각 SW2 누름 횟수, SW3 누름 횟수이다.

2.3 하드웨어 구성 사진 첨부하기

2.4 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기

https://youtube.com/shorts/nwqf-flUZCY

3. 프로그램 digital-clock

3.1 digital-clock 프로그램 코드 쓰기

```
#include "mbed.h"
#include "C12832.h"
PwmOut led_G(D9);
PwmOut led_R(D5);
InterruptIn sw2(SW2);
InterruptIn sw3(SW3);
InterruptIn up(A2);
InterruptIn down(A3);
InterruptIn left(A4);
InterruptIn rite(A5);
InterruptIn center(D4);
C12832 lcd(D11, D13, D12, D7, D10);
Timer timer;
int hour=0, mint=0, sec=0, ms=0;
int start = 0;
void ISR_sw2() {
    if (start == 0) {
        timer.start();
        led_G = 0;
        led_R = 1;
        start = 1;
    } else {
        timer.stop();
        led_G = 1;
```

```
            led_R = 0;
            start = 0;
        }
}
void ISR_sw3() {
        timer.reset();
        sec=0;
        ms=0;
        led_G = 0;
        led_R = 1;
        start = 1;
}
void ISR_up() {
        mint++;
        hour=(hour+mint/60)%24;
        mint=mint%60;
}
void ISR_down() {
        mint--;
        hour=(hour-(mint < 0)+24)%24;
        mint=(mint+60)%60;
}
void ISR_left() {
        hour--;
        hour=(hour+24)%24;
}
void ISR_rite() {
        hour++;
        hour=hour%24;
}
void ISR_center() {
        sec=0;
        ms=0;
        timer.reset();
}
int main()
{
        sw2.fall(&ISR_sw2);
        sw3.rise(&ISR_sw3);
        up.rise(&ISR_up);
        down.rise(&ISR_down);
        left.rise(&ISR_left);
        rite.rise(&ISR_rite);
```

```
    center.rise(&ISR_center);
    led_G = 1;
    led_R = 1;
    lcd.cls();
    lcd.locate(0, 6);
    lcd.printf("Digital Clock!");
    while(true){
        ms=timer.read_ms()/10;
        if(ms>=100){
            ms=ms%100;
            sec++;
            mint+=(sec/60);
            hour+=(mint/60);
            hour=hour%24;
            mint=mint%60;
            sec=sec%60;
            timer.reset();
            start = 1;
        }
        lcd.locate(0, 16);
        lcd.printf("Current Time: %2d:%2d:%2d.%2d",hour,mint,sec,ms);
    }
}
4
```

3.2 프로그램 작성 아이디어 혹은 이유 설명 쓰기

Timer 라는 class 를 활용해 msec 마다 1 씩 증가하는 카운터를 얻고, 이 카운터의 값이 1000 을 넘으면 hour(시간), mint(분), sec(초) 변수에 1 이 증가됩니다. 그러면 while 문에서 LCD 에 이 위 변수들에 ms_10(ms/10)을 추가해 출력과 제어를 반복합니다.. Interrupt 는 방향은 UP, DOWN, LEFT, RIGHT, CENTER 5 가지가 존재합니다. UP 은 분 1 증가, DOWN 은 분 1 감소, LEFT 는 시간 1 감소, RIGHT 는 시간 1 증가, CENTER 는 분과 ms/10 을 0 으로 초기화 등 각각에 interrupt 가 발생하면, 이 기능을 수행하는 handler 를 넣었습니다. ENTER 는 분과 ms/10 을 0 으로 초기화하고, Timer 를 reset()멤버함수로 초기화했습니다. 그리고 UP, DOWN 은 범위를 넘어가면, hour(시간)을 1 증가 or 1 감소하도록 작성했고, LEFT, RIGHT 는 범위를 넘으면 23 에서 0 으로, 0 에서 23 으로 넘어가게 만들었습니다. 그리고 sw2 는 Timer 시작과 정지기능, sw3 은 초 초기화 기능을 넣었습니다.

Timer 는 Timer 라는 class 를 활용하여 time 를 적용했습니다. 이 객체에 멤버함수로는 read_ms 사용하여 ms 라는 변수를 함수 반환 값으로 초기화 했다. Timer 에 초기화가 필요하면 reset(), 가동하려면 start(), 정지하려면 stop()등에 멤버함수를 사용했고, 시계에 주기적 증가를 Timer 를 통해 구현했습니다.

while 문은 ms 이 100 을 넘으면, 즉 1 초 되면 sec 는 1 증가, mint 상황에 따라 1 증가, hour 도 상황에 따라 1 증가를 하는 if(조건문)문을 넣었습니다. 이것으로 시각을 변수에 담게 됩니다.

3.3 하드웨어 구성 사진 첨부하기

3.4 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기

https://youtube.com/shorts/a1X60WvWm3o?feature=share

4. 프로그램 mutex-no/yes

4.1 mutex-no 프로그램 코드 쓰기

```
#include "mbed.h"
Serial pc(USBTX, USBRX, 115200);
Thread thread1, thread2, thread3;
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) ;
}
```

4.2 mutex-yes 프로그램 코드 쓰기

```
#include "mbed.h"
Serial pc(USBTX, USBRX, 115200);
Thread thread1, thread2, thread3;
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

mutex-no

화면에 1 을 50 번 출력하고 줄바꿈 하는 작업, 2 를 50 번 출력하고 줄바꿈 하는 작업, 3 을 50 번 출력하고 줄바꿈 하는 작업을 반복적으로 수행하는 프로그램을 3 개의 thread 를 통해 구현하기 위해 우선 thread1, thread2, thread3 3 개의 thread 를 정의했습니다. 이후 1 을 50 번 출력하고 줄바꿈 하는 작업을 반복 수행하는 thread_1 함수, 2 를 50 번 출력하고 줄바꿈 하는 작업을 반복 수행하는 thread_2 함수, 3 을 50 번 출력하고 줄바꿈 하는 작업을 반복 수행하는 thread_3 함수를 반복문을 통해 작성했습니다. 이때 mutex-no 프로그램은 수행 중인 thread 의 작업이 끝나야 다른 thread 가 수행하는 프로그램이 아니므로 thread 동기화를 위한 별도의 함수를 사용하지 않았습니다. 이후 메인 함수를 작성할 때 start 함수를 통해 3 개의 thread 가 수행되도록 했습니다. 이때 start 함수의 인수로 미리 정의한 3 개의 함수 thread_1, thread_2, thread_3 을 사용했습니다. 이를 통해 3 개의 함수가 각각 thread1, thread2, thread3 3 개의 thread 로써 동작하도록 프로그램을 작성했습니다.

mutex-yes 프로그램은 mutex-no 프로그램과 달리 수행 중인 thread 의 작업이 끝나야 다른 thread 가 수행될 수 있도록 프로그램을 작성해야 합니다. 즉, 화면이라는 공유자원을 수행 중인 thread 가 독점적으로 사용할 수 있게 해야 합니다. 이를 위해 thread 동기화가 필요하며 이때 mutex 를 활용할 수 있습니다. 따라서 mutex-yes 프로그램을 작성할 때 Mutex 라는 클래스를 활용했습니다. 이후 1 을 50 번 출력하고 줄바꿈 하는 작업을 반복 수행하는 thread_1 함수를 작성했습니다. 이때 lock() 함수를 사용하여 thread 가 작업을 수행할 때 화면을 독점적으로 사용할 수 있게 했으며, 이후 unlock() 함수를 통해 다른 thread 가 화면을 사용할 수 있도록 작성했습니다. 2 를 출력하는 thread_2 함수, 3 을 출력하는 thread_3 함수도 thread_1 함수처럼 lock(), unlock() 함수를 사용하여 작성했습니다.

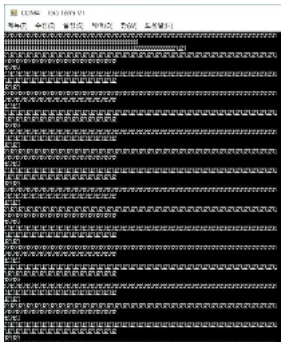
이후 메인 함수를 작성할 때 start 함수를 사용하여 thread_1, thread_2, thread_3 함수가 thread 로써 수행을 시작하도록 했습니다.

4.3 하드웨어 구성 사진 첨부하기

없음

4.4 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기

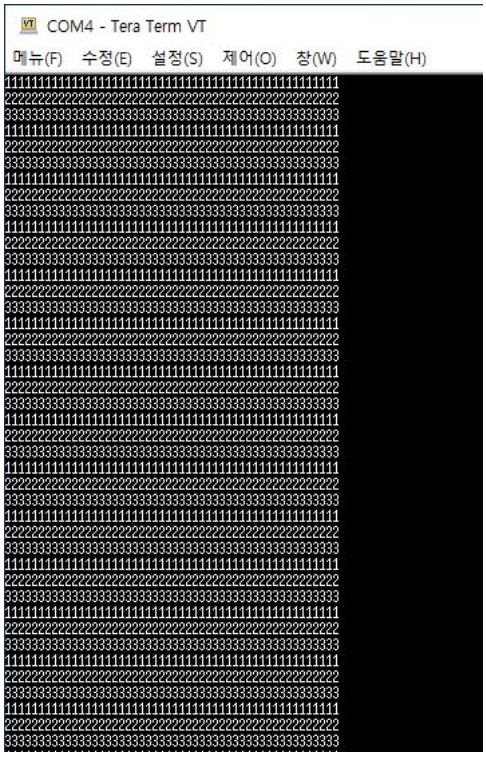
mutex-no



링크

https://youtube.com/shorts/oK1xBy22U7U?feature=share

mutex-yes



링크

https://youtube.com/shorts/LE7ELBKEBEM?feature=share

끝.