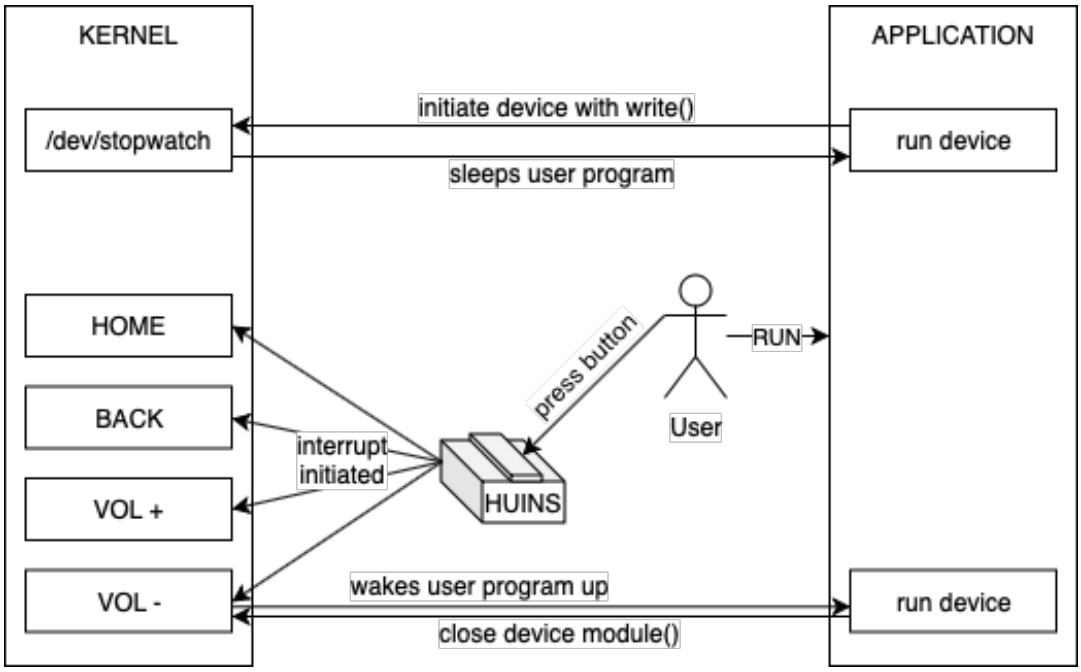
Embedded System Software
HW #3

20151623
한상구

1. 프로젝트 목표

Module programming, interrupt handler, timer, wait queue, device driver 구현 등 실습 시간 때 배운 내용을 활용하여 프로그램을 작성한다.

2. 설계



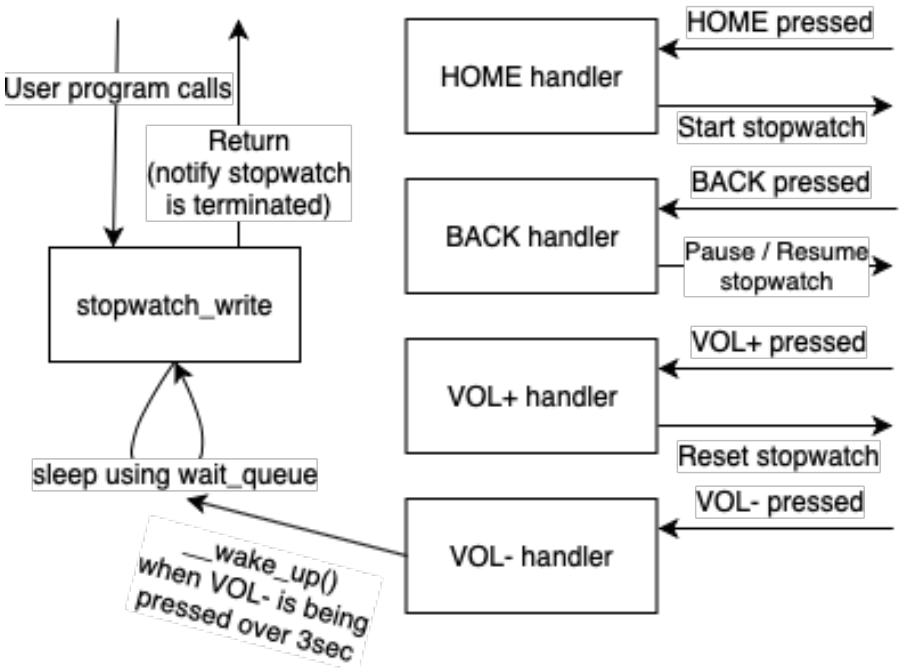
<그림 1 - 전체적인 프로젝트 구조도>

위 그림 1 과 같이 프로젝트를 설계, 구현하였습니다.

Run user program → User program initiate (and initialize) stopwatch via write() → Run stopwatch via press buttons on HUINS board 의 흐름이며, 세부 구현 내용은 아래 [3. 기능 구현] 에서 서술하도록 하겠습니다.

3. 기능 구현

a. Module (Device driver, Timer)



<그림 2 - device driver 의 전체적인 흐름도>

전체적인 흐름은 위 그림 2 와 같으며, 2 개의 타이머와 1 개의 `WAIT_QUEUE` 를 이용하여 구현하였습니다.

user program 이 `stopwatch_write` 을 호출하면 `interruptible_sleep_on` 를 이용해 user program 을 sleep 시키고, 아래 두 타이머를 이용해 stopwatch 가 종료될 때 까지 return 하지 않고 sleep 상태로 유지시켰습니다.

타이머 하나는 시간 측정을 위해 사용하였고 (`watch_timer`), 하나는 VOL- 버튼의 눌려진 시간을 측정하기 위해 사용하셨습니다 (`term_timer`).

Pause/Resume 의 구현은 `watch_timer` 를 timer_list 에서 `timer_pending()`를 이용해 현재 상태를 확인하고, `add_timer()`, `del_timer()` 를 이용하여 꺼내고 집어 넣는 방식으로 구현했습니다.
timer_list 에서 꺼낼 때 (Pause 되었을 때)의 시간을 전역변수로 저장해 두고 (`paused_at = watch_timer.timer.expires - get_jiffies_64();`), Resume 할 때 이용하여 (`watch_timer.timer.expires = get_jiffies_64() + HZ - paused_at;`) expires_at 을 설정했습니다.

VOL- 버튼이 눌리게 되면 `term_timer` 를 이용해 stopwatch 를 종료할 것인지 확인했습니다. timer 가 pending 되어있는 상태라면 `mod_timer` 를 이용하여 다시 3 초 후의 버튼 상태를 체크하도록 업데이트하고, 없다면 3 초후의 버튼 상태를 체크하도록 timer_list 에 `term_timer` 를 추가합니다.

`term_timer` 가 pending 된 상태에서 버튼이 release 되는 경우엔 `del_timer` 를 이용하여 timer_list 에서 `term_timer` 를 제거하도록 하여 `term_run` 이 실행되는 경우는 3 초동안 버튼이 release 되지 않은 상태가 되도록 구현했습니다.

`term_run` 에서 `__wake_up` 을 이용해 sleep 되어 있던 user program 을 깨우고, 정상적으로 종료할 수 있도록 구현했습니다.

b. Application

위에서 서술한 device driver 를 이용하여 stopwatch 를 제어할 수 있도록 작성한 응용 프로그램입니다.

device driver open 을 통해 stopwatch 를 initialize 하고, write 을 불러 stopwatch 가 종료될 때 까지 sleep 하게 됩니다.

VOL- 버튼을 3 초이상 눌러 stopwatch 가 종료되면, device file 을 close 하고 종료합니다.