Embedded System Software 과제 1

(과제 수행 결과 보고서)

컴파일 법 -> make push : make와 push를 함께해줍니다

과목명: [CSE4116] 임베디드시스템소프트웨어

담당교수: 서강대학교 컴퓨터공학과 박 성 용

학번 및 이름: 20171677, 이진훈

개발기간: 2020. 04. 27. -2020. 05. 01.

최 종 보 고 서

1. 개발 목표  
   - 각 과제마다 주어지는 주제를 바탕으로 본 과제에서 추구하는 개발 목표를 설정하고 그 내용을 기술할 것
   1. 인풋을 스위치나 이벤트로 받아서 fpga에 출력할 수 있다.
      1. 이를 위해, 싱크로나이제이션 등의 기법을 사용한다.
      2. 커널 모듈을 심고 그를 이용한다. 그를 위해 디바이스 파일을 연다
      3. mmap을 사용하여, 디바이스를 이용한다.

II. 개발 범위 및 내용  
- 자신들이 설계한 개발 목표를 달성하기 위하여 어떠한 내용의 개발을 수행할 지 그 범위와 개발 내용을 기술할 것.

가. 개발 범위

* 1)  Clock - 모드 1   
  스위치, FND, LED
* 2)  Counter – 모드 2   
  FND, LED, 간단한 수학
* 3)  Text editor - 모드3   
  FND, LCD, 스위치, Matrix
* 4)  Draw board -모드4   
  FND, 스위치, Matrix

나. 개발 내용

* 1)  Clock - 모드 1
  + 스위치로 입력을 받을 수 있게 구현해야한다.
  + FND로 시간 출력을 할 수 있어야한다.
  + 현재 시간을 디바이스에서 받아올 수 있어야한다.
  + LED를 사용해서 지정된 행동을 해야한다.
* 2)  Counter – 모드 2
  + FND로 total숫자를 출력할 수 있어야한다.
  + LED로 진수에 따라 불빛이 바껴야한다.
  + 간단한 수학을 이용해야한다.
* 3)  Text editor - 모드3
  + FND로 total 카운트를 출력할 수 있어야한다.
  + LCD로 텍스트를 출력할 수 있어야한다.
  + 스위치로 영어입력을 할 수 있어야한다.
  + Matrix로 현재 모드를 출력해야한다.
* 4)  Draw board -모드4
  + FND로 total 카운트를 출력할 수 있어야한다.
  + 스위치로 입력으로 다양한 행위를 한다.
  + Matrix로 현재 상태를 출력해야한다.

III. 추진 일정 및 개발 방법  
- 자신들이 설정한 개발 목표를 달성하기 위한 개발 일정을 설정하고, 각 요소 문제를 해결하기 위해서 어떤 방법 을 사용할 지 기술할 것.

가. 추진 일정  
2020. 4. 27 구상 2020. 4. 28 -2020. 5. 1 구현

나. 개발 방법  
로컬 개발환경으로 코딩하고 git을 이용해서, 타겟에 넣는다.

0. shm

기본적으로 공유메모리로

input -> main 으로 버튼의 입력을 넘기고

main에서 이로 적절한 행동을 한뒤

main -> output으로 주요 변수들 (FND, matrix 등) 의 값들을 넘긴다.

**int** shmid = shmget((key\_t) 0x10, **sizeof**(**struct** in\_packet), IPC\_CREAT|0644);

**struct** in\_packet\*shmaddr = (**struct** in\_packet\*)shmat(shmid, **NULL**, 0);

이것이 그 코드인데, 처음에는 ftok를 썼다가 잘 열리지않아서 ID를 직접 할당하였다. 패킷이라는 구조체를 만들었다. 패킷 == 공유메모리라서 공유메모리를 레퍼런스하여 값을 얻을 수 있다.

ev[0].type = shmaddr->type;

ev[0].value = shmaddr->value;

ev[0].code = shmaddr->code;

위와 같이 shmaddr를 레퍼런스하여 버튼의 값을 얻는다.

1. action

action은 명세서를 그대로 구현하였다.

**if** (mode == 0) {

**if** (push\_sw\_buff[0] == 1) {

위처럼 모드와 버튼을 조건으로 사용하여 행위를 구분한다.

1. 디바이스와 mmap

LED를 제외한 모든 디바이스는 커널모듈을 사용하고, LED는 mmap을 사용한다.

dev = open(FPGA\_TEXT\_LCD\_DEVICE, O\_WRONLY);

**if** (dev<0) {

printf("Device open error : %s\n", FPGA\_TEXT\_LCD\_DEVICE);

exit(1);

}

write(dev, str, 8);

디바이스 모듈은 위 처럼 오픈을 하고, 거기에 write을 한다.

하지만 mmap은 아래와 같다.

fd = open("/dev/mem", O\_RDWR | O\_SYNC);

**if** (fd < 0) {

perror("/dev/mem open error");

exit(1);

}

fpga\_addr = (**unsigned** **long** \*)mmap(**NULL**, 4096, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, FPGA\_BASE\_ADDRESS);

일단 fd를 할당한다. 그 fd로 addr를 얻는다.

led\_addr=(**unsigned** **char**\*)((**void**\*)fpga\_addr+LED\_ADDR);

그리고 오프셋을 더한다.

그러면 디바이스를 가상메모리주소로 접근할 수 있다. 사용은 아래와 같다.

\*led\_addr=data;

mmap은 디바이스를 가상메모리주소로 끌고와서 사용하는 방식인데 이는 cpu의 io pin과 대응된다.

page table의 인덱스가 가상메모리 주소고, 그 데이터가 디바이스의 위치 되는 것이다.

IV. 연구 결과  
- 최종 연구 개발 결과를 자유롭게 기술할 것.

모든 액션을 구현하고 실행되는 것을 확인하였다.

V. 기타  
양이 많은데, 시간이 부족해서 아쉽다. 개발자는 계획이 중요하다는 것을 깨달았다.