**Embedded System Software HW#1**

**(설계 프로젝트 수행 결과)**

과목 명 : [CSE4116] 임베디드 시스템 소프트웨어

담당 교수 : 박성용 교수님

개발 기간 : 2019. 04. 07 ~ 2019. 04. 13

**프로젝트 제목 : Embedded System Software HW#1**

**제출일 : 2019년 4월 14일**

**참여 조원 : 황기덕**

1. **개발 목표**

* 디바이스 컨트롤과 IPC를 이용하여 주어진 Clock, Counter,Text editor, Draw board 를 구현한다.

1. **개발 범위 및 내용**
   1. **개발 범위, 내용**

* **input, main, output process**

본 프로그램은 디바이스로부터 입력을 받고, 그에 따라 연산을 한 후 그것을 출력해 주어야 한다. 여러 가지 일을 동시에 해야 하므로 관리가 쉽게 하기 위해 세 개의 프로세스를 만들어 입력받는 부분 따로, 연산하는 부분 따로, 출력해주는 부분을 따로 만들어 실행해준다.

* **IPC**

프로세스들이 독립적으로 움직이긴 하지만 input process에서 받은 입력을 main process로 넘겨주고, 연산을 한 후 그 결과를 output process로 넘겨주는 등의 정보는 서로서로 이동해야 한다. 그러므로 ipc. 이 프로젝트에서는 message queue를 이용하여 프로세스간 통신을 해 주기로 한다. input -> main, main -> output 단방향으로만 메세지가 오가고, 두 개의 메세지 큐는 서로 다른 메세지를 사용하게끔 만들어 주었다.

* **Mode**

이 프로젝트에선 총 다섯개의 모드가 돌아간다. 각 모드를 효율적으로 관리하기 위하여 각각을 따로 만들고, 비슷한 틀로 구현하였다. main process에서 이 mode들을 부르게 될 것이며, 그 mode들 안에서 실행에 필요한 연산들을 실질적으로 하게 될 것이다.

* **user define mode**

추가구현 부분에서는 piano tile 이라는 리듬게임을 제작하였다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

|  |  |
| --- | --- |
| **일정** |  |
| **4.07** | **요구사항 분석** |
| **4.08~09** | **input. main, output process 구현** |
| **4.10~4.11** | **각 mode들 구현** |
| **4.12** | **프로그램 검증, 정리** |
| **4.13** | **보고서 작성** |

* 1. **개발 방법**
* **input, main, output process**

20141602.c에서 fork() 를 해 주어서 총 세 개의 process를 만들어 주었다. 메인 process를 main process로 두고, 두 개의 자식 프로세스를 input, output process로 만들어 주었다. 20141602.c 에서 main\_process.c, input.c output.c 를 불러 주어서 각각의 process가 동작하도록 한다. input에선 사용자로부터 입력을 받는 것에 집중하여, 어떠한 입력이 들어오면 적당한 연산을 한 후 main process 로 넘겨준다. main process에선 input.c로부터 받은 입력을 각 모드에 맞게 연산하여 그 결과를 output.c 로 전달해준다. output.c에서는 main process에서 전달받은 값으로 사용자에게 적당한 출력을 보여준다.

* **IPC**

각각의 process들끼리 데이터를 주고받아야 한다. message queue 방식으로 그걸 구현하였다. 20141602.c에서 message queue를 초기화 후 만들어 주었다. 각각의 key number를 input process, main process, output process를 부를 때 넘겨 주었다. 그 후 msgsnd, msgrcv 를 이용하여 각각의 데이터를 주고받았다.

* **Mode**

program\_mode/ 에 총 다섯 개, clock.c counter.c text.c draw.c user.c를 구현하였다. 각각의 파일에는 그 모드를 구현하기 위해 필요한 정보들이 들어있다. 각 모드들에 들어갈 때 construct mode를 해 주고, 그 모드에서 빠져나올 때 destroy mode를 해 주었다. 모드 내에서 thread가 필요할 경우에 ( ex : led의 깜빡임, dot matirx의 깜빡임...) thread를 만들어 주었고 그 안에서 mutex lock를 이용해 서로간의 synchronization을 맞춰 주었다.

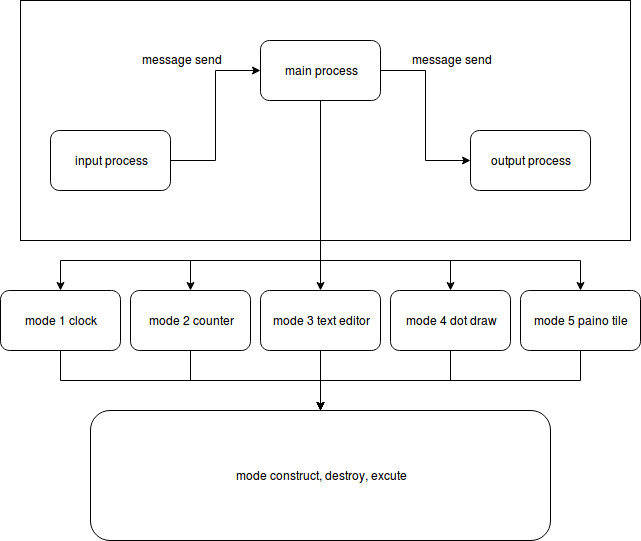
* **user define mode**

리듬게임을 제작하기 위해 dot 부분을 노트가 내려오는 부분으로 설정하였고, switch 부분을 각 노트를 누르는 것으로, led는 현재 속도, fnd는 남은 life를 나타내었고 text lcd에는 간단한 정보를 나타내었다. 그리고 게임을 플레이하다 잘못 누르거나 누르지 못하면 buzzer가 울리도록 구현해 주었다.

* 1. **연구원 역할 분담**
* 개인 프로젝트이다.

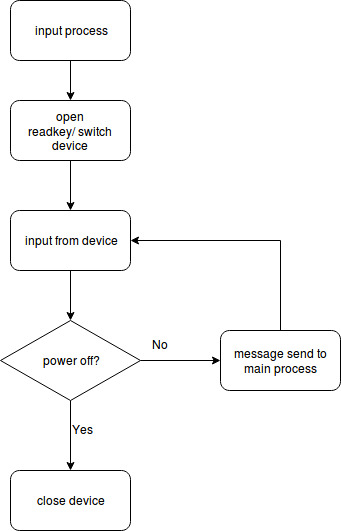
1. **연구 결과**
   1. **합성 내용**

* **all**

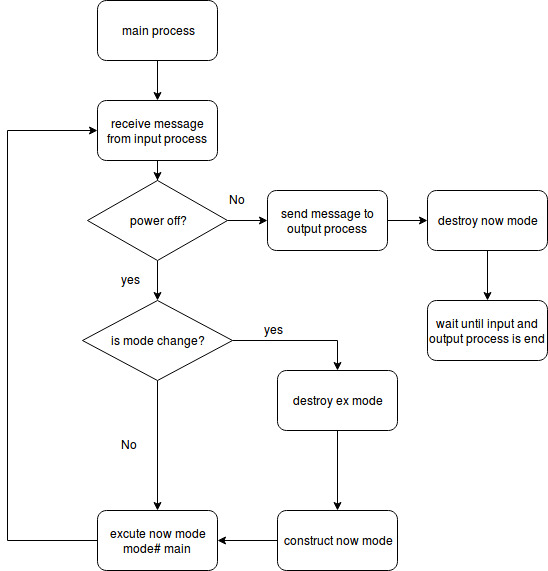
****

* **input, main, output process**

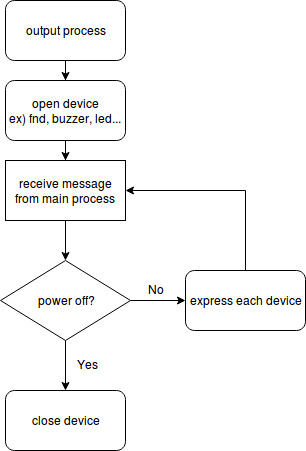
**input process :**

****

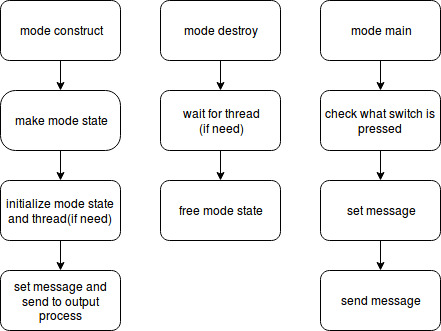
**main process :**

****

**output process :**

****

* **each mode**

****

* 1. **제작 내용**
* **input, main, output process**

Input process 에서는 먼저 readkey, switch device를 오픈한 후 루프를 돌면서 입력이 받아지게끔 해 주었다. 두 개의 device 가 연결되어있지 않고 서로 다른 디바이스로 간주되기 때문에, 하나를 못 받았다고 그 키 입력을 받을 때까지 계속 기다리면 안 된다. 그러므로 open 할 때 non blocking 으로 열어서 입력받지 못했더라도 기다리지 않고 바로 다음으로 넘어갈 수 있게 해 주었다.

switch 부분에서는 동시 입력이 가능해야 하고, 꾹 누르고 있어도 한 번의 입력으로 간주해야 하기 때문에, prev buffer 를 만들어 주어 이전 입력과 계속 같은 입력이 들어오면 무시하게 만들어 주었다. readkey 부분에서는 버튼이 눌러졌을 때, 즉 key press 상태일 때 입력을 받아 주었다.

이렇게 readkey와 switch의 상태를 받은 후 io\_msgbuf에 넣어서 main process로 보내 주었다. 만약 poweroff 를 입력받았다면 message를 main process로 보내준 이후 자신은 끝나버린다.

main process에서는 먼저 기본 모드인 mode1, clock을 construct 해 주고 루프에 들어간다. 그 후 input process로부터 message를 전달받은 후, 먼저 모드가 변경되었는지, 아니면 power off버튼이 눌려졌는지를 확인하다. 만약 mode가 변경되었다면 현재 모드를 변한 모드로 변경시켜 주고, 이전 mode를 destroy 해 준다. 그 후 각각의 mode main 함수를 불러서 그 모드에 대한 코드가 실행되게 해 준다. poweroff 버튼이 눌러졌다면 현재 해당하는 mode를 destroy 해 주고 루프를 빠져나간 후, 자식 process들, 즉 input process와 output process가 끝날 때까지 기다렸다가 끝난다.

output process에서는 실제로 출력을 해 주어야 하기 때문에. 각 출력을 담당할 dev들을 open 해주고, led에 한해서는 mmap을 사용할 것이기 때문에 mapping시켜 준다. 그 후 루프에 들어가게 된다. main process에서 온 message를 읽고, 각 데이터를 output해주는 함수에 필요한 인자들과 함께 넘겨 (ex : output\_dot, output\_text\_lcd ...) 출력을 해 준다. 그러다가 poweroff 가 들어온다면 바로 루프를 종료시키고, 열었던 디바이스 드라이버들을 close 해 주고 unmap 해 준 다음 끝난다.

* **IPC**

각각의 process들이 통신하기 위해서 message queue를 썻는데. 이건 20141602.c 에서 msgget을 해 주어서 선언하였다. input - main process에서 이용할 message queue의 key는 key\_io 에 저장해두고, main - output process에서 이용할 queue는 key\_mo에 저장해 두고 각각의 process로 나뉘어질 때 key를 같이 넘겨주어서 서로서로 통신할 수 있게끔 만들어 주었다. message를 통해 정보를 받아야 무언가 연산을 하거나 출력을 할 수 있으므로 message는 blocking으로 만들어 주었다.

* **Mode**

program\_mode/ 에 총 다섯개의 모드를 만들어 놓았다. 각각에 모드 파일 안에는 construct, destroy, mode main, set msg의 함수가 기본적으로 구현되어 있고, 필요에 따라 background function 과 각각 모드에서 필요한 다른 함수들이 포함되어 있다. 처음으로 construct가 불리고 그 다음부터는 mode main 함수가 돌아가며, 모드가 변경 혹은 디바이스가 사용자에 의해 종료되었을 때 destroy 함수가 불려진다. construct 에서는 mode state를 할당해주고 각종 변수들을 초기화해준다. 필요하면 thread를 이곳에서 만들고 mutex lock을 초기화해 주기도 한다.

mode main에서는 input에서 어떠한 switch number가 왔는지 확인하고 그 번호에 알맞은 행동을 하게끔 도와준다. 그 후 결과를 state 변수에 저장한 후 msg를 set한 후 msgsnd 함수를 호출해 output에 출력해야 할 것을 넘겨준다.

msg set 함수에서는 현재 mode의 state를 보고 그에 알맞게 output으로 넘겨줄 message를 만들어준다.

destroy 함수에서는 현재 모드에서 사용하던 모든 것들을 종료하고, mode state를 free, thread를 사용하였다면 thread join, mutet destroy를 해 준다.

background function에서는 mode main함수를 support해 주기적으로 해 주어야 하는 무언가를 할 때 background function 에서 실행해준다. 엉킬 위험이 있으니 synchronization은 mutex lock을 통해 맞추어준다.

* **mode 1 clock**

1번 스위치를 누르면 시간을 변경 가능하다. 이 상태에서 3번을 누르면 시간을 증가시킬 수 있고, 4번을 누르면 분을 증가시킬 수 있다. 이 상태에서 1을 누르면 설정한 시간으로 변화하고 어느 때든 2를 누르면 다시 초기 시간으로 돌아온다.

* **mode 2 counter**

1번 스위치를 누르면 진법 변환이 가능하다. 2번을 누르면 3번째 자리, 3번을 누르면 2번째 자리, 4번을 누르면 1번째 자리수가 올라간다. 초기 진법은 10진법이다.

* **mode 3 text editor**

각각의 스위치에 알파벳이 저장되어있다. 5,6 스위치를 동시에 누르면 숫자 <->영어 입력을 서로 전환할 수 있다. 8,9번 스위치를 동시에 누르면 한 칸 띄울 수 있고, 2, 3번 스위치를 동시에 누르면 text lcd에 표시된 모든 문자들을 지울 수 있다. 최대 입력 길이를 벗어나면 맨 처음 썻던 글자가 밀려나고, 마지막 위치에 새로운 글씨가 작성된다.

* **mode 4 draw board**

2,4,6,8번 스위치로 상하좌우 움직일 수 있다. 5번 스위치를 누르면 그 위치에 불이 켜진다. 3번 스위치는 현재 커서가 어디 있는지 나타내는 버튼인데, 기본으로는 ON 되어있어서 현재 위치의 커서가 깜빡거린다. 3번을 끄면 현재 커서 위치가 표시되지 않는다. 9번을 누르면 칠해진건 꺼지고, 꺼진건 칠해지게 된다. 1번을 누르면 초기 상태로 reset되고 7번을 누르면 현재 그려져있는 그림이 모두 지워진다.

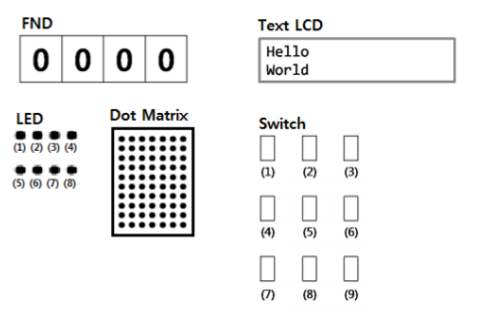
* **mode 5 user define mode**

리듬게임을 제작하였다. program\_mode/user.c 에 구현되어 있고, 이 파일 역시 다른 mode들과 비슷하게 construct, destroy, main. set msg 함수가 존재한다. 추가로 thread를 사용하기 때문에 background 함수를 만들어 주었고, 알맞은 타이밍에 버튼을 눌렀는지 확인하기 위한 check 함수, text lcd에 표시하기 위해 만들어 둔 mode5\_strcat 함수를 추가로 구현해 주었다.

맨 처음으로 construct 함수가 호출되면 mode5\_state를 생성하고 초기 설정을 넣어준다. 그리고 thread를 새롭게 만들어 준 후 background 함수로 넘겨준다. mode main 함수가 호출된 후 게임이 시작된다. 크게 세 가지의 상태가 있는데, before game, running, finish 상태이다. before game 상태에서는 스위치 1, 2, 3번으로 난이도를 조작할 수 있고 5번 스위치로 게임을 시작할 수 있다. running 상태에서는 실제 게임이 이루어지는데, 4, 7, 8, 9, 6번 버튼을 사용하여 dot matrix를 보고 게임을 하게 된다. 이 때 스위치가 눌러지면 check 함수를 호출해 제때 맞게 눌렀는지를 확인하게 된다. 맞게 눌렀으면 combo와 score가 올라간다. 틀리게 누르거나 놓치면 life가 줄어들게 된다.

life가 다 줄어들게 되면 게임이 종료되고 text lcd에 score와 max combo가 표시된다. 이 상태에서 5번 스위치를 다시 누르면 mode5\_destroy, mode5\_construct가 차례로 호출되고(reset) 다시 시작하게 된다.

background function에서는 dot에 그려진 노트들을 주기적으로 밑으로 내려주며, swi 입력이 받아지지 않았을 경우에는 그 노트를 치지 못한 것이므로 life를 감소시킨다. 일정 시간이 지나갈수록 점점 속도가 빨라지게 해 주고, 빨라졌다는 표시를 led로 나타내 주었다.



위의 그림에서 보면 led에 speed 상태가 나와있다 (숫자가 커질수록 속도가 빠르다)

fnd에는 현재 남은 life가 표시된다.

text lcd에는 현재 상태가 표시된다.

switch:

-> before running :

sw1 = easy mode

sw2 = normal mode

sw3 = hard mode

sw5 = game start

-> running :

sw4: dot matrix에서 두번째 칸

sw7: dot matrix에서 세번째 칸

sw8: dot matrix에서 네번째 칸

sw9: dot matrix에서 다섯번째 칸

sw6: dot matrix에서 여섯번째 칸

-> finish:

sw5: retry

1. **기타**
   1. **연구 조원 기여도**

* 개인 프로젝트이다.
  1. **소감**
* 처음으로 cross compile을 통해 target device에서 돌아가는 프로그램을 작성해 보았다. 많이 헤매었지만 차츰 해 볼수록 익숙해져서 이전 학기에 프로젝트를 했던 것처럼 진행할 수 있어서 괜찮았다. 하지만 하다가 뭔가 실수하면 아예 껏다가 다시 켜야 하는 불편함이 있었다.