**Embedded 1st HW. Calculator on Board with IPC**

**(설계 프로젝트 수행 결과)**

**과목명: 임베디드 시스템 프로그래밍**

**담당교수: 서강대학교 컴퓨터공학과 박 성 용**

**학번 및 이름: 20141500, 권태국**

**개발기간: 2017. 04. 05. -2017. 04. 07.**

**최 종 보 고 서**

**I. 개발 목표**

이 프로젝트에서 목표하는 개발 목표는 크게 3가지 이다. 첫째, 디바이스드라이버를 이용하여 디바이스를 조작하는 것. 둘째, IPC를 이용하여 여러 프로세스 간의 통신을 하는 것. 셋째, 유저의 switch, readkey를 통한 입력을 처리하는 동시에 background에서 해야하는 처리를 동시에 하는 멀티스레드 프로그래밍이다. 나는 이러한 3가지 개발 목표를 가지고 개발을 하였다.

**II. 개발 범위 및 내용**

**가. 개발 범위 및 개발 내용**

**1. 사용자로부터 입력을 받아 main process에게 전달하는 input process**

**2. input process로부터 메시지를 받아 처리하고, output process에게 device에 결과를 출력하는 메시지를 전달하는 main process**

**3. main process로부터 message를 받아서 해당하는 디바이스에 규약에 맞춰 결과를 출력해주는 output process**

**4. 위 3개의 process가 서로 통신하기 위한 체계 (IPC)**

**5. 4가지 기본 구현 모드 (Clock, Counter, Text editor, Draw board)**

**6. 추가 구현 모드 (두더지 잡기 게임)**

**7. main process에서 몇 가지 모드를 구현하기 위해 사용되는 multi-thread programming.**

**III. 추진 일정 및 개발 방법**

- 자신들이 설정한 개발 목표를 달성하기 위한 개발 일정을 설정하고, 각 요소 문제를 해결하기 위해서 어떤 방법을 사용할 지 기술할 것. 또한 각 연구원의 역할을 분명히 기술할 것.

**가. 추진 일정**

**4/5 : 프로젝트의 전체 설계 및 프로세스간 통신 구현**

**4/6 : 기본 구현 모드 4가지 구현.**

**4/7 : 추가 구현 모드 (두더지 잡기 게임) 구현.**

**나. 개발 방법**

위 개발 내용에 해당하는 프로그램을 개발하기 위해서 여러가지 방법론이 필요하다.

1. IPC

나는 process 간 통신을 위해 pipe를 사용하였다.

1. non-blocking read

input process에서 readky device를 polling하기 위해서는 read가 non-blocking이어야 한다. 나는 device를 open할 때 O\_NONBLOCK 을 함께 사용하여 해결하였다.

1. atomic

프로그램 내에서 thread간의 atomic operation을 위해 gcc extension인 atomic을 사용하였다. 원래 C11의 atomic을 사용하고 싶었으나, 실습실 컴퓨터의 gcc version 4.8.\* 이라서 C11 기능은 사용하지 못했다.

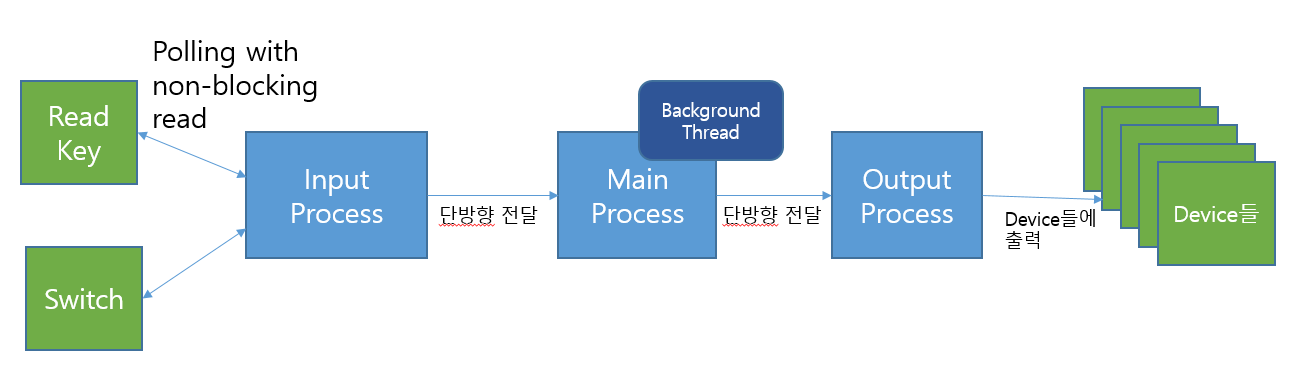
1. pthread

thread를 사용하기 위해 pthread 라이브러리를 사용하였다.

**IV. 연구 결과**

- 최종 연구 개발 결과를 자유롭게 기술할 것.

**1. 합성 내용:**

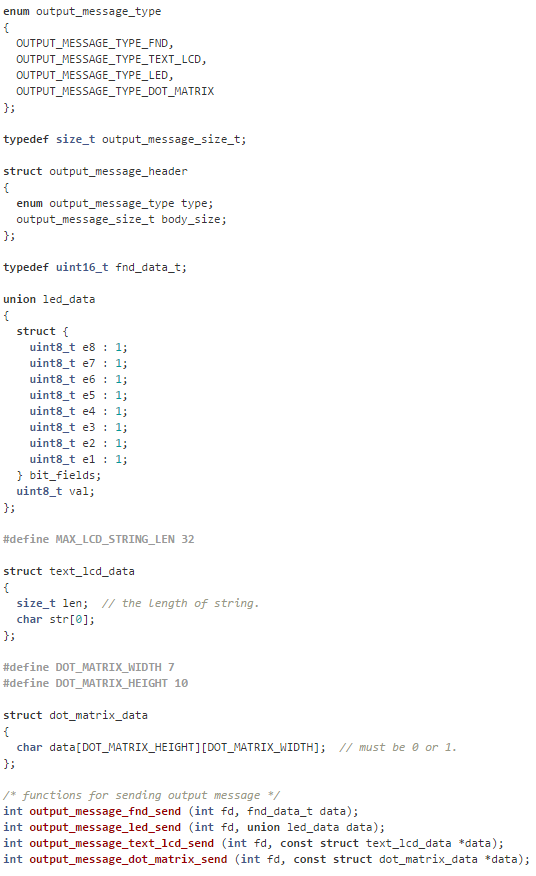
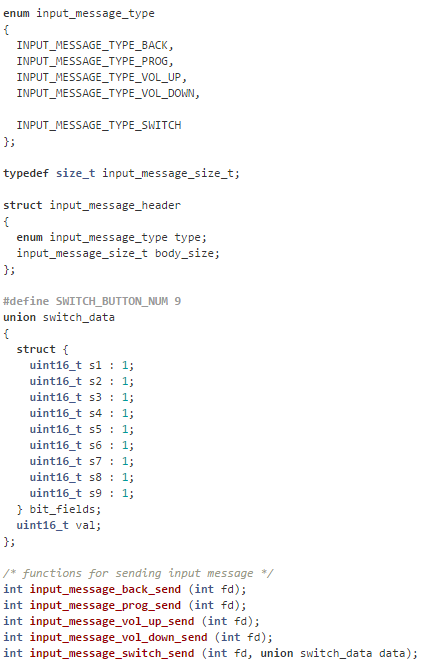


전체 프로그램의 흐름도는 위와 같다;.

**2. 제작 내용:**

1. process간 통신을 IPC

나는 input process, main process, output process 들끼리 통신을 하기 위한 체계를 만들었다. 나는 기본적으로 pipe를 기반으로 통신하는데, 서로 간의 통신은 메시지라는 추상화된 layer를 바탕으로 일어난다. pipe는 단순히 메시지를 송수신하기 위한 창구고, 나는 message를 통해 통신한다.



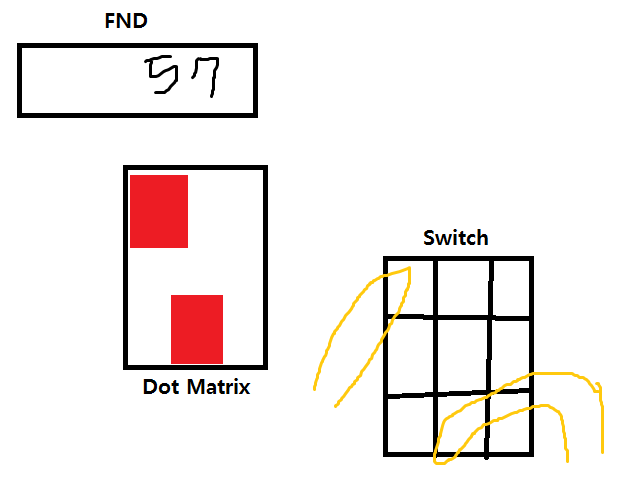
메시지의 구조는 위와 같다. 나는 이러한 구조를 바탕으로 프로세스간 통신을 하였다.

2. 기본 구현 모드

프로젝트 설명서에 적혀 있는데로 성실하게 구현하였다. 어차피 나는 프로젝트 설명서에 적혀 있는데로 구현했을 뿐이므로 자세한 설명은 생략하도록 하겠다.

3. 추가 구현 모드 (두더지 잡기 게임)

나는 추가 구현으로서 두더지 잡기 게임을 구현하였다.



그림으로 그리자면 위와 같다.

DOT Matrix에 두더리를 의미하는 사각형들이 등장하고, 사용자는 그 사각형(두더지)들이 사라지기 전에 빠르게 두더지를 잡아야한다. 두더지를 잡는 방법은 해당하는 위치를 switch에서 누르면 된다. 그림만 봐도 직관적으로 이해가 갈 것이다.

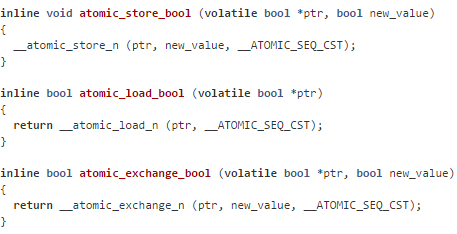
FND에는 점수가 출력된다. 두더지를 잡으면 1점이 추가되고, 만약 두더지를 잡지 못하면 1점이 감소한다. 만약 두더지가 없는 위치에 해당하는 switch를 누를 때에도 점수가 1점 감소한다.

나는 게임의 재미를 위해 점수에 따라서 두더지가 나오는 속도를 빠르게 하였다. 점수가 낮으면 두더지가 천천히 나오고, 점수가 높으면 두더지가 빨리 나온다. 내 경험상 100점을 넘기 힘들다.

4. multi-thread 프로그래밍

나는 Clock모드, Draw board모드, Extra(두더지 잡기 게임)모드를 위해서 멀티스레드 프로그래밍을 하였다. clock 모드에서 led가 깜박이는 기능, draw board 모드에서 cursor가 깜박이는 기능, 두더지잡기게임모드에서 두더지가 계속 등장하는 기능은 모드 사용자의 input과 상관없이 계속 진행되어야한다. 이를 구현하기 위한 방법은 싱글스레드 기반에서 ioctl을 이용해 pipe를 non-blocking mode로 설정해놓은 뒤, pipe에 읽을 데이터가 없을 경우 (사용자의 인풋이 없을 경우) 동시적으로 실행되어야하는 기능들을 실행하는 그런 방법이 있다. 이 방법을 사용해도 좋지만, 나는 좀 더 구현의 편의성을 위해 thread를 따로 하나 만들었다.

mutli-thread 프로그래밍에서는 thread간의 동기화가 중요하다. 특히 공유 변수에 대한 접근은 철저히 동기화되어야한다. 나는 이를 위해 GCC extension중 하나인 atomic 기능을 사용하였다.



나는 GCC extension의 atomic 기능들을 wrapping하는 함수를 만들어 사용하였다.

그리고 program의 mode가 바뀔 때마다 각 mode에 해당하는 resource를 해제해줘야하는데 여기에는 thread도 포함된다. 즉, 원할 때 thread를 종료시킬 수 있는 그런 방법이 필요한데, 나는 그것을 위해 bool 변수를 하나 두고 thread가 그 변수가 true일 동안만 작업을 수행하고, false가 되는 순간 빠져나오도록 하였다. 이 bool 변수는 엄밀하게 동기화되야하므로 위에서 말한 atomic 을 사용하였다.

**3. 시험 및 평가 내용:**

평가방법은 프로젝트 설명서에 나와있는 흐름대로 평가하면 된다. 프로젝트 설명서에 없는 내용이 추가구현모드인 두더지잡기게임인데, 이 것 같은 경우 내가 위에서 충분히 설명을 하였고, 직접 게임을 진행해보면서 평가하면 된다.

여러 가지 테스트를 진행하였고, 또한 프로젝트 자체의 설계 및 코드 퀄리티에 신경을 썼기 때문에 내 프로그램은 보건 및 안정, 내구성이 뛰어나다

**V. 기타**

**1.** **연구 조원 기여도**:

나 혼자 했다.

권태국 : 100%

**2.** 기타 본 설계 프로젝트를 수행하면서 느낀 점을 요약하여 기술하라. 내용은 어떤 것이든 상관이 없으며, 본 프로젝트에 대한 문제점 제시 및 제안을 포함하여 자유롭게 기술할 것.

참 재미있었다. 다만 프로젝트 설명서에 설명이 약간 부실하여 실제 구현상에서 어떻게 해야할 지 헷갈리는 것들이 많았다. 따라서 나는 설명이 엄밀히 되있지 않은 부분은 알아서 잘 하라는 뜻으로 해석하여 상식 선에서 구현을 하였다. 다만, 프로젝트 설명서가 좀 더 엄밀했으면 좋겠다는 생각이 든다.