

E M B E D D E D   S Y S T E M

# 목숨을 건 구구단 게임

2017152049 정하림  
2016156005 김주하  
2017152000 이길형  
2017156037 정수경

**BTS**  
BEST TEAM OF SANGIDAE

# CONTENTS

## 01

### 문제 정의

- 문제정의
- 설계과제 목표

## 02

### 요구사항 분석

- 디바이스
- 요구사항 분석
- 시나리오

## 03

### 시스템 설계

- 소프트웨어 설계도
- 스레드 설계

## 04

### 구현 및 데모

- 시스템데모
- 소스코드
- 현실적 제한요소

## 05

### 마무리

- SCRUM  
FRAMWORK
- 팀원간역할분담
- 결론 및 총평

**BTS**

BEST TEAM OF SANGIDAE

## 문제 정의

“ 5개 이상의 디바이스와 스레드를 사용한 병행 처리 응용프로그램 설계 ”

## 설계 목표

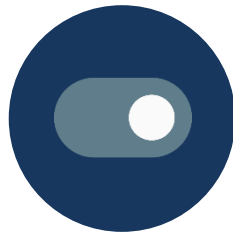
1. 각종 디바이스 드라이버의 구조를 파악하며 응용프로그램을 작성
2. Synchronization tool을 사용한 스레드간 순서 조정, 경쟁상태 방지
3. 메모리 사용량, 성능 최적화
4. 시큐어 코딩 기법을 활용하여 프로그래밍

## 02

## “ 디바이스 ”



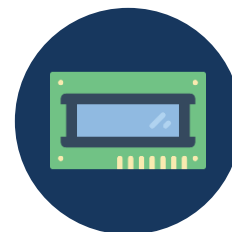
제한횟수



출정답



입력 O/X 표시



문제



출력카운트다운

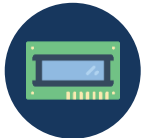
## 02

1. FND를 30:00 부터 카운트를 시작한다.
2. LCD에 문제를 출력한다.
3. 문제를 보고, 스위치를 이용해 정답을 입력한다.
  - 1) 정답이면 Dot Matrix 에 O를 표시, 다음문제를 출력하고
  - 2) 정답이 아니면 Dot Matrix 에 X를 표시, LED를 하나씩 점멸한다.
4. FND가 00:00 초가 되거나 LED 8개가 모두 점멸 되면 게임을 종료한다.

## 02



- 8개부터 시작하여, 오답일 경우 하나씩 점멸한다.



- 10문제의 구구단 문제를 출력하되, Ex)  $8 \times ? = 56$   
사용자가 정답을 맞추면 다음 문제를 출력한다.



- 1 부터 9까지의 문제의 정답을 입력한다.



- 정답 여부를 O와 X 로 표시한다.

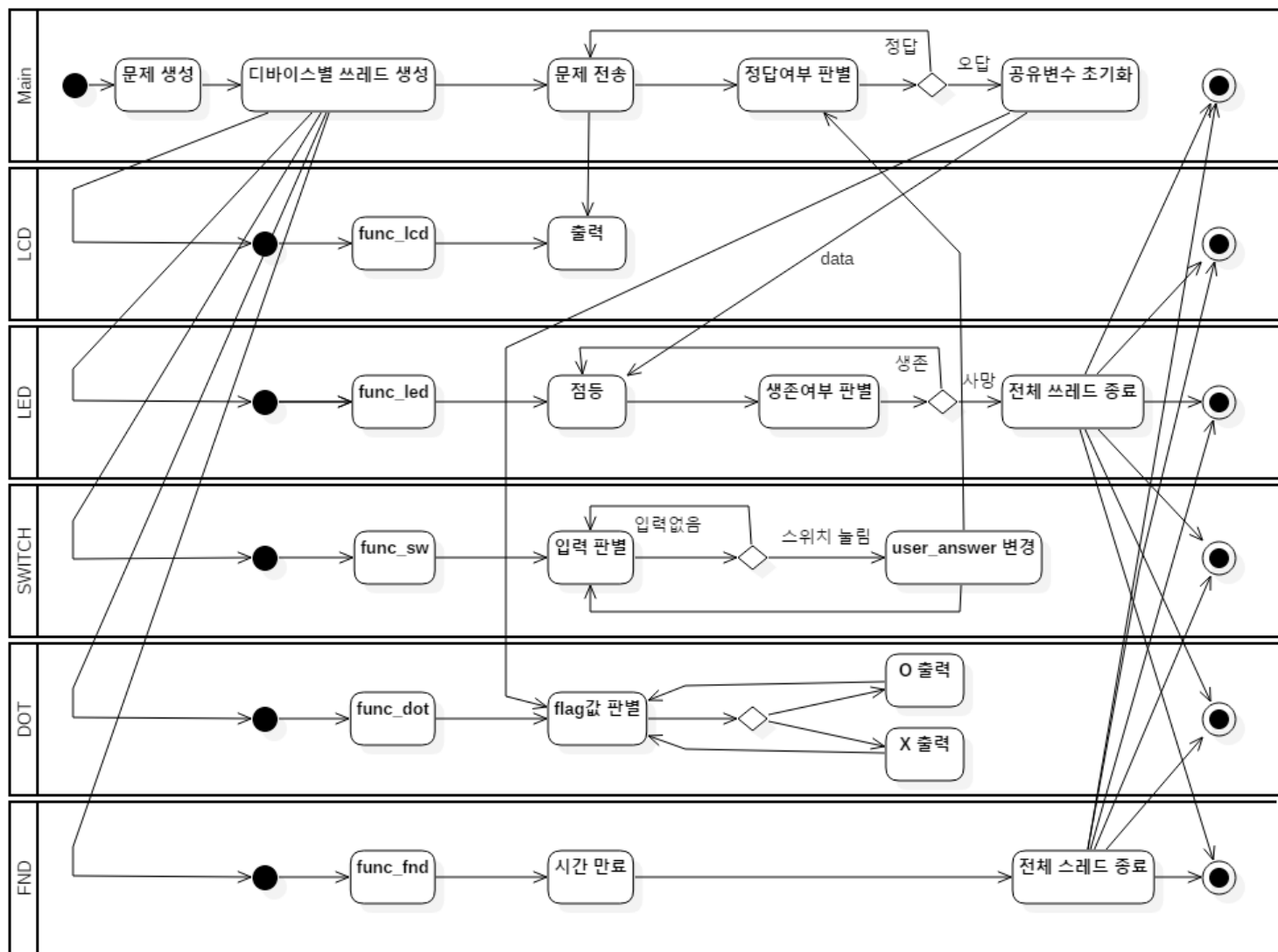


- 30초부터 시작하여 시간을 카운트한다.



- 30초부터 시작하여 FND 가 00:00이 되거나, 목숨을 모두 소모하여  
LED 8개가 모두 점멸하면 프로그램을 종료한다.

## 03



## 03

## 공유 변수

user_answer	is_pushed	m_id (pthread_mutex_t)
<p>사용자가 입력한 답을 저장하는 변수. 스위치를 누르면 입력한 스위치 값으로 변경. main에서는 정답여부 판별 후, user_answer 값을 0으로 초기화.</p>	<p>스위치의 눌림 여부를 표시하며 초깃값은 0. 스위치를 누르면 is_pushed 값을 1로 변경. main에서는 정답여부 판별 후, is_pushed 값을 다시 0으로 변경.</p>	<p>user_answer와 is_pushed는 main과 switch 스레드 에서 write 하는 변수이기 때문에 경쟁상태가 발생 할 수 있어 m_id 로 critical section 설정.</p>

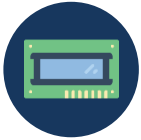
## 전역 변수

Flag	led_data	quiz_number
<p>Dot Matrix 에서 사용. main에서 변경한 flag 값에 따라 O/X 출력</p>	<p>Led 에서 사용. main에서 변경한 led_data값을 LED 모듈에 출력</p>	<p>LCD에서 사용. main에서 변경한 quiz_number 를 이용해 문제배열 접근.</p>



## 03

## 함수



- void\* func\_lcd();
- main에서 정답 판별 후 변경한 quiz\_number 를 이용해 문제 [quiz\_number] 를 lcd에 출력.



- void\* func\_led();
- main에서 정답 판별 후 변경한 data 값에 따라 led 점등



- void\* func\_switch();
- 스위치 입력 시, 공유변수 user\_answer의 값 변경, 공유변수 is\_pushed 를 1로 변경.



- void\* func\_dot();
- main에서 정답 판별 후 변경한 flag 값에 따라 O/X 출력



- void\* func\_fnd();
- 30초부터 0초까지 카운트다운.

## 04



main

```
for (int i = 0; i < 10; i++) {
```

```
// 전송
```

```
    while (1) {
```

```
        if ((right_answer == user_answer) && is_pushed == 1) {
```

```
            printf("correct! right answer: %d\n", right_answer);
```

```
            printf("user_answer :%d\n\n", user_answer);
```

```
            quiz_number++;
```

```
            flag = 1;
```

```
            pthread_mutex_lock(&m_id);
```

```
            user_answer = -1;
```

```
            is_pushed = 0;
```

```
            pthread_mutex_unlock(&m_id);
```

```
            break;
```

```
        }
```

## 04

```
else if ((right_answer != user_answer) && is_pushed == 1) {
```

```
    flag = 0;
```

```
    led_data >>= 1;
```

```
    pthread_mutex_lock(&m_id);
```

```
    is_pushed = 0;
```

```
    pthread_mutex_unlock(&m_id);
```

```
    if (led_data == 0x00) {
```

```
        printf("Bomb!!!!!!!!!!\nYou Die...\n");
```

```
        printf("Exit 3 sec later...\n");
```

```
        for (int j = 0; j < 3; j++) {
```

```
            usleep(10000);
```

```
        }
```

```
        // 스레드 종료
```

```
        pthread_exit(&p_switch); pthread_exit(&p_led);
```

```
        pthread_exit(&p_fnd); pthread_exit(&p_lcd);
```

```
        pthread_exit(&p_dot);
```

```
    }
```

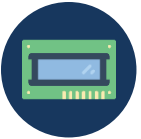
```
}
```

```
}
```

## 04



```
while (1) {  
    if (write(dev, &led_data, 1) < 0) {  
        puts("led write error!\n");  
    }  
    sleep(1);  
}
```



```
while (1) {  
    i = quiz_number;  
  
    right_answer = question[i].rear_number;  
  
    // 변수 i 를 사용한 문제배열 question 접근, 문제를 string으로 변환과정 진행  
  
    memset(string + str_size, '\0', MAX_BUFF - str_size);  
  
    write(dev, string, MAX_BUFF);  
}
```

## 04



```
while (1) {  
    usleep(400000);  
    read(sw_dev, &push_sw_buff, buff_size);  
  
    for (i = 0; i < MAX_BUTTON; i++) {  
        if (push_sw_buff[i] == 1) {  
            // printf("%d button pushed!\n", i);  
            pthread_mutex_lock(&m_id);  
            user_answer = i + 1;  
            is_pushed = 1;  
            pthread_mutex_unlock(&m_id);  
        }  
    }  
}
```

## 04



```
while (1) {  
    if (flag == 1) { //answer is true...  
        str_size = sizeof(fpga_number[0]); //0  
        write(dev, fpga_number[1], str_size);  
    }  
  
    if (flag == 0) { //answer is false...  
        str_size = sizeof(fpga_number[1]); //X  
        write(dev, fpga_number[1], str_size);  
    }  
}
```

FND 시간카운트 기능 작성  
응용 프로그램 작성  
보고서 작성

정하림

김주하

스위치 입력 기능 작성  
응용 프로그램 작성  
PPT 작성  
프로그램 통합

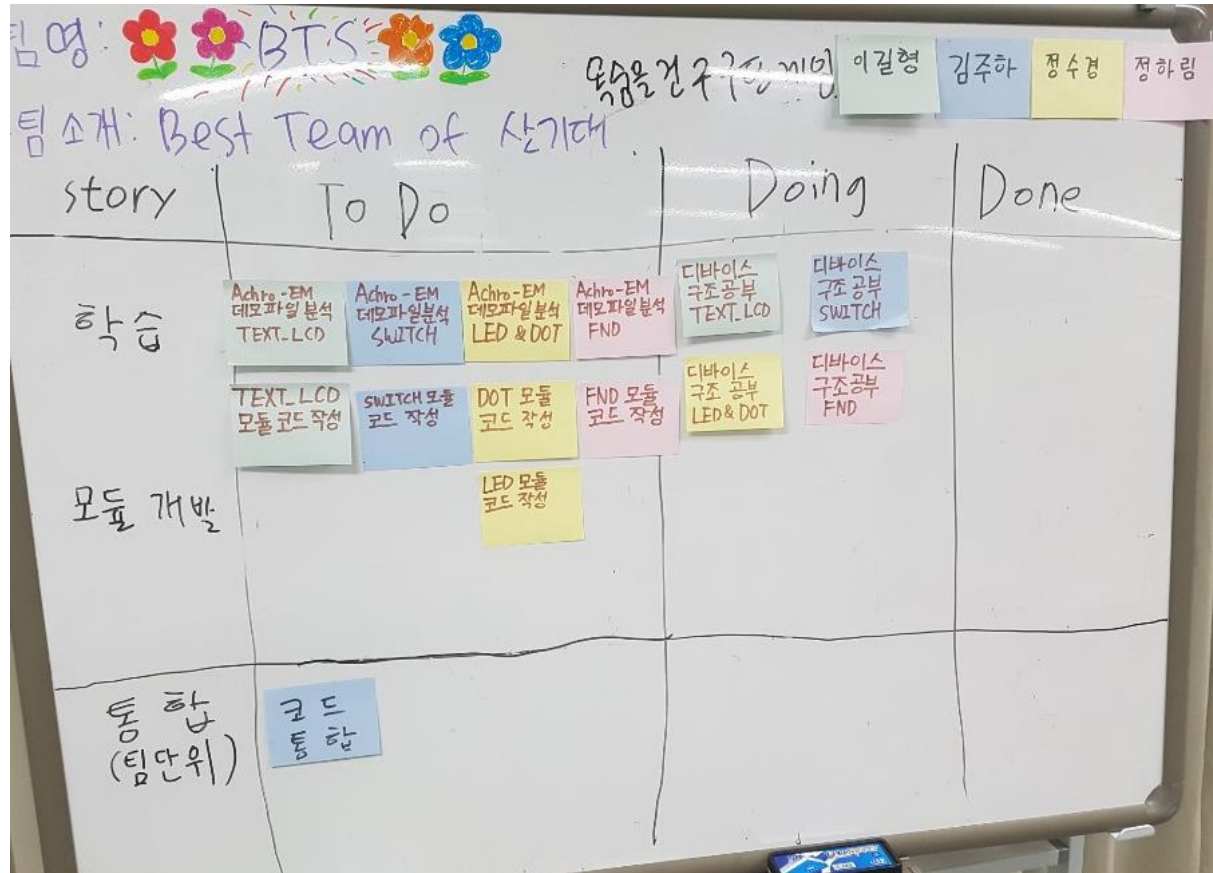
LCD 구구단 문제 출력  
응용 프로그램 작성  
보고서 작성

이길형

정수경

LED 점등/점멸 기능 작성  
Dot Matrix O/X 표시 작성  
응용 프로그램 작성  
보고서 작성



## 1주차



- 디바이스 드라이버의 기본구조 공부
- 각자 맡은 디바이스의 소스 분석



## 2주차





팀명:  BTS  독립운동기념관 기념 이길형 김주하 정수경 정하림


팀 소개: Best Team of 상기대


story	To Do	Doing	Done
학습	Secure 기법 코딩	공유자원 설정 SWITCH 모듈 코드 작성	디바이스 구조공부 TEXT_LCD 디바이스 구조공부 SWITCH 디바이스 구조공부 LED & DOT 디바이스 구조공부 FND
모듈 개발		LED 모듈 코드 작성 DOT 모듈 코드 작성	Achro-EM 데모파일 분석 TEXT_LCD Achro-EM 데모파일 분석 SWITCH Achro-EM 데모파일 분석 LED & DOT Achro-EM 데모파일 분석 FND
통합 (팀단위)			TEXT_LCD 모듈코드작성 FND 모듈 코드 작성

- 각자 맡은 디바이스 예제 실행
- 예제를 바탕으로 한 응용 프로그램 작성
- 스레드를 활용한 프로그램 통합 시작, 메인 프로그램 작성

## 3주차

팀명:   BTS  

팀 소개: Best Team of  살기대

담당:  김주하

이길형   김주하   정수경   정하

story	To Do	Doing	Done
학습	Secure 기법 코딩		<div>디바이스 구조 공부 TEXT_LCD</div> <div>디바이스 구조 공부 SWITCH</div> <div>디바이스 구조 공부 LED &amp; DOT</div> <div>디바이스 구조 공부 FND</div>
모듈 개발			<div>Acho-EM 데모파일 분석 TEXT_LCD</div> <div>Acho-EM 데모파일 분석 SWITCH</div> <div>Acho-EM 데모파일 분석 LED &amp; DOT</div> <div>Acho-EM 데모파일 분석 FND</div>
통합 (팀단위)			<div>TEXT_LCD 모듈 코드 작성</div> <div>SWITCH 모듈 코드 작성</div> <div>LED 모듈 코드 작성</div> <div>FND 모듈 코드 작성</div> <div>공유자원 설정</div> <div>DOT 모듈 코드 작성</div>

- 메인 프로그램 작성 완료
- 프로그램 통합 진행
- 디바이스들을 하나씩 합치면서 발생하는 오류 해결

## 결론

디바이스 드라이버 사용 전에는  
insmod()로 모듈을 커널에 적재하고,  
mknod()를 실행해 장치파일을 생성한다.  
또한 사용은 파일을 다루는 법과 같으며,  
open() 으로 드라이버를 열고 read() 와 write() 를  
사용해 디바이스 드라이버에 데이터를 쓰거나  
데이터를 읽어온다.  
사용 후에는 close() 를 사용해  
드라이버를 닫는다.

## 총평

방금 전까지 잘 작동하던 보드가  
코드를 수정하지 않았음에도 불구하고  
시나리오 대로 작동하지 않아서 어려웠다.  
또한 각각 실행하면 잘 되지만, 통합 시 제대로  
작동하지 않은 적도 많아서  
원인을 찾느라 매우 힘들었다.  
생각하는 대로 보드가 작동하지 않아  
그 점이 매우 어려웠다.

**THANK  
YOU**