

# REPORT

## 제목 : IoT 프로그래밍 프로젝트

제출일	2024. 06. 18	전 공	소프트웨어융합전공
과 목	IoT프로그래밍	학 번	2019250027, 2019250036, 2019250041, 2020740067
담당교수	신정호	조 원	안병한, 이상민, 이정민, 장민석

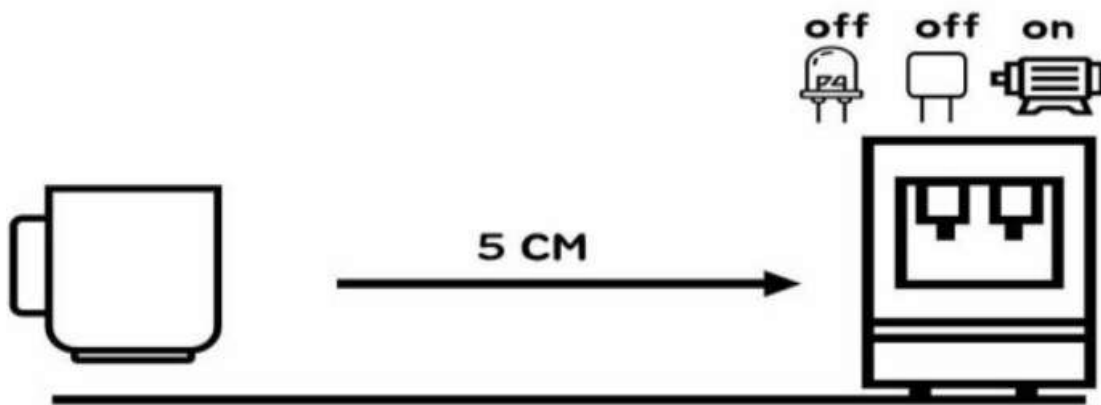
# 목 차

I. 개	요	3
1. 기존 프로젝트 조사		3
2. 프로젝트 선정		4
3. Chat GPT 사용		5
II. 구성	요소	6
1. 주요 파일 디스크립터		6
2. 주요 데이터 구조		6
3. 주요 함수		7
III. 동작	설명	8
1. 난이도 설정 및 초기화		8
2. 게임 루프		10
3. 프로그램 종료 및 정리		13
IV. 동작	확인	14
V. 요약 및 역할분담		16
참 고 자 료		17

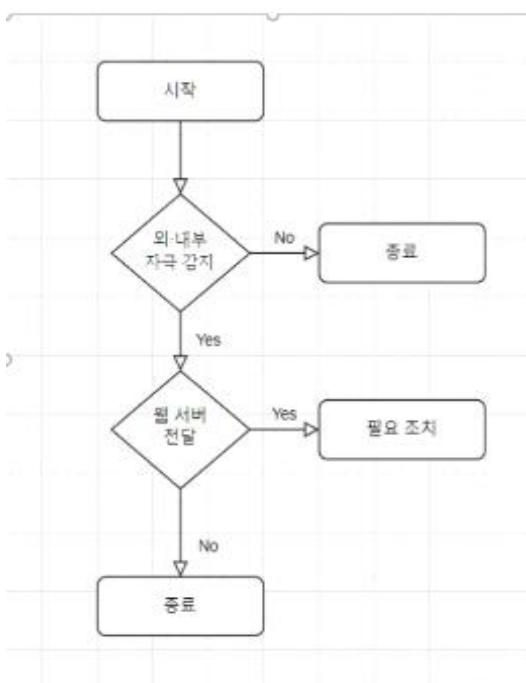
# I . 개 요

## 1. 기존 프로젝트 조사

첫 번째로 IoT센서를 이용한 시각 장애인용 자동화 정수기를 조사하였음. 이의 목적은 시각장애인들의 일상생활에서 겪는 불편함을 해소하고자 자동화된 정수기를 개발하는 것을 목적으로 함. 필요한 하드웨어 구성은 초음파 센서, 부저, LED, 서보모터 등으로 이를 통해 컵의 위치를 인식하고 거리를 측정하며 안내, 물 공급의 자동화를 구현하도록 함. 소프트웨어는 센서 데이터를 기반으로 부저를 이용하는 것으로 계획하였음.

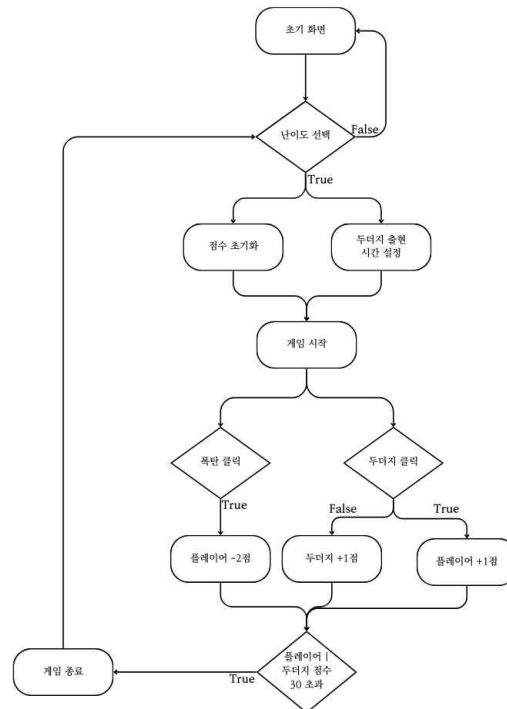


두 번째로 계획한 것은 건강 모니터링 시스템임. 다양한 IoT 센서를 활용하여 환자의 건강 상태와 주변 환경을 종합적으로 모니터링하고, 급격한 변화가 있을 경우 미리 대응할 수 있게 하는 시스템임. 진동, 자이로스코프, 가속도, 온도, 가스, 먼지, 적외선, 초음파 센서 등 다양한 센서를 활용하여 환자의 움직임, 체온, 주변 환경 등을 감지할 수 있게 함. 소프트웨어로는 센서 데이터를 웹 서버로 전송하여 모니터링하고, 필요 시 의료진에게 경고 및 알림을 통해 정보를 전달함.



## 2. 프로젝트 선정

기존의 프로젝트를 조사하며 현실적으로 구현이 가능한지, 그리고 참고할 수 있는 자료가 존재하는 지에 대한 여부를 찾아보며 두더지 게임을 최종적으로 프로젝트 주제로 선정하였음. 두더지 구멍에서 두더지가 튀어나오면 이를 제때 잡는 일반적인 형식을 기반으로 함. 이는 추가하거나 변경할 수 있는 요소의 존재 뿐 아니라 게임의 룰 자체가 어렵지 않기 때문에 선정하게 됨. 우선적으로 Tact Switch는 두더지 구멍 역할을 하는 스위치로 활용하도록 하고, 해당 위치의 스위치를 눌러 두더지를 잡는 방식을 채택함. Dot Matrix는 게임 화면을 표시하는 디스플레이의 역할임. 8x8 크기의 디스플레이에 두더지 구멍과 두더지를 표현할 수 있다고 판단함. Character LCD는 게임 정보 (점수, 남은 시간 등)을 표시하는 디스플레이의 역할을 하도록 계획함.



기존에 존재하는 두더지 잡기와 차별을 둔 부분은 라운드 선택 시스템과 다양한 변수 요소임. 참고했었던 두더지 잡기 게임에서는 한 라운드를 끝내면 다음 라운드로 자동적으로 넘어가는 시스템이었다면, 새롭게 계획한 시스템은 자신이 원하는 라운드를 선택하여 자유롭게 난이도를 변경할 수 있도록 하게하는 것임. 다양한 변수 요소로는 아이템이나 폭탄과 같은 요소가 있음. 아이템을 통해 점수를 빠르게 얻거나 폭탄으로 인해 점수를 잃는 형태로 게임에 흥미를 높일 수 있다고 판단함.

### 3. Chat GPT 사용

이 이게 처음에 누르고 다음부터 matrix에 불이 안들어오네

아, 그 부분은 제가 실수로 누락했네요. 코드를 수정하여 다시 보내드리겠습니다.

```
c Copy

// ...

int main() {
    // ...

    while(1)
    {
        // 다음 단계로 넘어간다던
        if (isNext == 1)
        {
            isNext = 0;
            startmole = 1;
            stage++;
        }
    }
}
```

이 unsigned char bomb\_mole[9][8] = {  
 {0, 36, 36, 0, 0, 0, 0, 0}, //1번폭탄  
 {0, 9, 9, 0, 0, 0, 0, 0}, //2번폭탄  
 {0, 66, 66, 0, 0, 0, 0, 0}, //3번폭탄  
 {0, 0, 0, 36, 36, 0, 0, 0}, //4번폭탄  
 {0, 0, 0, 9, 9, 0, 0, 0}, //5번폭탄  
 {0, 0, 0, 66, 66, 0, 0, 0}, //6번폭탄  
 {0, 0, 0, 0, 36, 36, 0, 0}, //7번폭탄  
 {0, 0, 0, 0, 9, 9, 0, 0}, //8번폭탄  
 {0, 0, 0, 0, 66, 66, 0, 0}, //9번폭탄  
};  
이렇게 하면 matrix에 어떻게 나올까

위의 bomb\_mole 배열은 9개의 폭탄 두더지에 대한 도트 매트릭스 패턴을 정의하고 있습니다.  
각 폭탄 두더지는 8x1 크기의 도트 패턴으로 표현됩니다.

이 배열을 사용하여 도트 매트릭스에 폭탄 두더지를 표시하면 다음과 같이 나타날 것입니다:

Copy

1번 폭탄:	2번 폭탄:	3번 폭탄:
.. .	...	.. .
.. .	...	.. .

4번 폭탄:	5번 폭탄:	6번 폭탄:
.. .	...	.. .
.. .	...	.. .

이 ALSA install했는데 위치가 어디야

ALSA 라이브러리를 설치한 후, 라이브러리 파일과 헤더 파일의 위치는 시스템마다 조금씩 다를 수 있습니다. 일반적으로 다음과 같은 위치에 설치됩니다:

- 라이브러리 파일: `/usr/lib/` 또는 `/usr/local/lib/`
- 헤더 파일: `/usr/include/` 또는 `/usr/local/include/`

ALSA 라이브러리의 정확한 위치를 찾으려면 다음 명령어를 사용할 수 있습니다:

1. 라이브러리 파일 위치 찾기:

```
find /usr/lib -name "libasound.so*"
find /usr/local/lib -name "libasound.so"
```

2. 헤더 파일 위치 찾기:

```
find /usr/include -name "asoundlib.h"
find /usr/local/include -name "asoundlib.h"
```

위 명령어를 터미널에서 실행하면 해당 파일들의 경로를 확인할 수 있습니다.

만약 라이브러리 파일이나 헤더 파일을 찾을 수 없다면, 다음 명령어를 사용하여 ALSA 개발 패키지를 설치해야 할 수 있습니다:

```
sudo apt-get install libasound2-dev
```

위 기록과 같이 개발과정에서 Chat GPT를 이용하여 프로그래밍을 진행하였음.

## II. 구성 요소

### 1. 주요 파일 디스크립터

주요 파일 디스크립터는 이러함. 'dot\_d'인 점 매트릭스 디바이스("/dev/dot")는 LED 매트릭스를 제어하기 위한 장치임. 본 프로그램을 통해 두더지, 또는 폭탄 두더지를 표시할 수 있음. 'tact\_d'인 택트 스위치 디바이스("/dev/tactsw")는 사용자의 버튼 입력을 감지하는 장치임. 'clcd\_d'인 문자 LCD 디바이스("/dev/clcd")는 게임 점수를 표시하기 위한 장치로 쓰임. 프로그램을 통해 본 파일을 열어 점수를 LCD에 출력할 수 있음. 'led\_d'인 LED 디바이스("/dev/led")는 난이도를 나타내기 위한 장치임. DIP 스위치의 상태에 따라 LED가 꺼지거나 켜질 수 있음. 'dip\_d'인 DIP 스위치 디바이스("/dev/dipsw")는 난이도를 설정하기 위한 장치임. 각 파일 디스크립터는 특정 하드웨어 장치와의 통신을 담당하며, 프로그램은 이를 통해 사용자 입력을 처리하고, 게임 상태를 표시함.

### 2. 주요 데이터 구조

#### 두더지 매트릭스 (mole[9][8])

이 배열은 9개의 두더지 각각의 점 매트릭스 표시를 정의함. 각 두더지는 8개의 바이트로 구성된 배열로 나타내며, 각각의 바이트는 점 매트릭스의 한 줄을 나타냄.

```
unsigned char mole[9][8] = {
    {0, 96, 96, 0, 0, 0, 0, 0}, //1번두더지
    {0, 24, 24, 0, 0, 0, 0, 0}, //2번두더지
    {0, 6, 6, 0, 0, 0, 0, 0},   //3번두더지
    {0, 0, 0, 96, 96, 0, 0, 0}, //4번두더지
    {0, 0, 0, 24, 24, 0, 0, 0}, //5번두더지
    {0, 0, 0, 6, 6, 0, 0, 0},   //6번두더지
    {0, 0, 0, 0, 0, 96, 96, 0}, //7번두더지
    {0, 0, 0, 0, 0, 24, 24, 0}, //8번두더지
    {0, 0, 0, 0, 0, 6, 6, 0}   //9번두더지
};
```

#### 폭탄 두더지 매트릭스 (bomb\_mole[9][8])

폭탄 두더지의 매트릭스 배열임. 이 배열도 9개의 폭탄 두더지 각각의 점 매트릭스 표시를 정의함. 두더지 배열과 유사하게 각 폭탄 두더지는 8개의 바이트로 구성됨.

```
unsigned char bomb_mole[9][8] = {
    {0, 64, 32, 0, 0, 0, 0, 0},
    {0, 16, 8, 0, 0, 0, 0, 0},
    {0, 4, 2, 0, 0, 0, 0, 0},
    {0, 0, 0, 64, 32, 0, 0, 0},
    {0, 0, 0, 16, 8, 0, 0, 0},
    {0, 0, 0, 4, 2, 0, 0, 0},
    {0, 0, 0, 0, 0, 64, 32, 0},
    {0, 0, 0, 0, 0, 16, 8, 0},
    {0, 0, 0, 0, 0, 4, 2, 0}
};
```

#### 현재 표시 매트릭스 (printmatrix[8])

현재 화면에 표시할 점 매트릭스 데이터를 저장하는 배열임. 이 배열은 두더지 또는 폭탄 두더지의 매트릭스를 복사하여 표시할 때 사용됨. 8개의 바이트로 구성되어 있으며, 각 바이트는 점 매트릭스의 한 줄을 나타냄.

```
unsigned char printmatrix[8] = {0};
```

### 3. 주요 함수

#### 함수 정의

##### clcd\_input2(int score\_player, int score\_mole)

해당 함수는 게임의 현재 점수를 문자 LCD에 표시하는 역할을 함. 사용자와 두더지의 점수를 각각 받아서 문자열로 변환한 후, 이를 합쳐서 문자 LCD에 출력함.

```
void clcd_input2(int score_player, int score_mole) {
    int clcd;
    char player_score[5];
    char mole_score[5];
    char clcd_text1[30] = "Player_Score: ";
    char clcd_text2[30] = "Mole_Score: ";

    clcd = open(clcd_d, O_RDWR);
    if (clcd < 0) {
        printf("clcd error\n");
        return;
    }
    sprintf(player_score, "%d ", score_player);
    sprintf(mole_score, "%d", score_mole);
    strcat(clcd_text1, player_score);
    strcat(clcd_text1, clcd_text2);
    strcat(clcd_text1, mole_score);

    write(clcd, clcd_text1, strlen(clcd_text1));
    close(clcd);
}
```

#### 함수 설명

##### 파일 디스크립터 열기

문자 LCD 디바이스 파일을 열어 읽기/쓰기가 가능하도록 함.

```
clcd = open(clcd_d, O_RDWR);
if (clcd < 0) {
    printf("clcd error\n");
    return;
}
```

##### 점수를 문자열로 변환

sprintf 함수를 사용하여 정수형 점수를 문자열로 변환함.

```
sprintf(player_score, "%d ", score_player);
sprintf(mole_score, "%d", score_mole);
```

##### 문자열 결합

strcat 함수를 사용하여 점수 문자열을 기본 텍스트에 추가함.

```
strcat(clcd_text1, player_score);
strcat(clcd_text1, clcd_text2);
strcat(clcd_text1, mole_score);
```

##### 문자 LCD에 쓰기

결합된 문자열을 문자 LCD에 씀.

```
write(clcd, clcd_text1, strlen(clcd_text1));
```

##### 파일 디스크립터 닫기

문자 LCD 디바이스 파일을 닫음.

```
close(clcd);
```

### III. 동작 설명

#### 1. 난이도 설정 및 초기화

전체적으로 난이도 설정은 DIP 스위치를 사용하여 이루어지며, 이에 따라 게임의 난이도가 조정됨. 또한 LED를 통해 현재 난이도를 시각적으로 표시함.

##### DIP 스위치 값 읽기

먼저, DIP 스위치 디바이스 파일을 열고 DIP 스위치의 값을 읽음. DIP 스위치는 8비트 값으로 표현되며, 각 비트는 개별 스위치의 상태를 나타냄.

```
dip = open(dip_d, O_RDONLY);
if (dip < 0) {
    printf("dip error\n");
    return 1;
}
```

##### 난이도 설정

DIP 스위치의 값에 따라 stagetime 변수를 설정함. 각 비트가 켜져 있는지 확인하고, 이에 따라 적절한 난이도를 설정함.

```
if (dipval & 0x01) {
    stagetime = 40; // DIP 스위치 1번 ON: 난이도 1 (가장 쉬움)
    led_data = 0x01; // DIP 스위치 1번 LED 켜기
} else if (dipval & 0x02) {
    stagetime = 35; // DIP 스위치 2번 ON: 난이도 2
    led_data = 0x02; // DIP 스위치 2번 LED 켜기
} else if (dipval & 0x04) {
    stagetime = 30; // DIP 스위치 3번 ON: 난이도 3
    led_data = 0x04; // DIP 스위치 3번 LED 켜기
} else if (dipval & 0x08) {
    stagetime = 25; // DIP 스위치 4번 ON: 난이도 4
    led_data = 0x08; // DIP 스위치 4번 LED 켜기
} else if (dipval & 0x10) {
    stagetime = 20; // DIP 스위치 5번 ON: 난이도 5
    led_data = 0x10; // DIP 스위치 5번 LED 켜기
} else if (dipval & 0x20) {
    stagetime = 15; // DIP 스위치 6번 ON: 난이도 6
    led_data = 0x20; // DIP 스위치 6번 LED 켜기
} else if (dipval & 0x40) {
    stagetime = 10; // DIP 스위치 7번 ON: 난이도 7
    led_data = 0x40; // DIP 스위치 7번 LED 켜기
} else if (dipval & 0x80) {
    stagetime = 5; // DIP 스위치 8번 ON: 난이도 8 (가장 어려움)
    led_data = 0x80; // DIP 스위치 8번 LED 켜기
}
```



## LED 표시

DIP 스위치의 값에 따라 설정된 난이도를 사용자에게 표시하기 위해 LED를 켜. LED는 DIP 스위치의 각 비트에 해당하는 위치에서 켜지도록 설정됨.

```
write(led_dev, &led_data, sizeof(unsigned char)); // 해당 DIP 스위치 LED 켜기
```

## 게임 초기화

난이도가 설정되면 게임의 상태를 초기화함. 초기화 과정에는 점수 초기화, 게임 루프 진입 준비 등이 포함됨.

```
int count = 0;
int score_mole = 0;
int score_player = 0;
int startmole = 1;
```

## 게임 루프

게임 루프는 DIP 스위치 값이 설정된 후에 시작됨. 게임 루프에서는 두더지가 나타나고, 사용자가 두더지를 잡는 과정을 반복함.

```
while (1) {
    // 두더지 생성
    if (startmole == 1) {
        startmole = 0;
        random_mole = (rand() % 9) + 1;
        timeleft = (rand() % 10);

        // 폭탄 생성 확률에 따라 폭탄 두더지 설정
    }
}
```

## DIP 스위치가 꺼진 경우

DIP 스위치가 꺼진 경우 LED를 끄고 대기 상태로 전환함. DIP 스위치 값이 0이면 게임을 종료함.

```
if (dipval == 0) {
    printf("게임을 종료합니다.\n");
    break;
}
```

## 요약

난이도 설정 및 초기화 과정은 게임의 시작 부분에서 이루어지며, DIP 스위치를 통해 난이도를 설정하고 LED를 통해 이를 시각적으로 표시함. 설정된 난이도에 따라 게임의 흐름과 난이도가 조정되며, 게임이 원활하게 진행될 수 있도록 초기 상태를 설정함.

## 2. 게임 루프

게임 루프는 두더지 게임의 핵심으로, 두더지를 생성하고 표시하며 사용자의 입력을 처리하는 일련의 과정을 반복적으로 수행함. 여기서는 게임 루프의 주요 단계인 두더지 생성, 두더지 표시 및 입력 처리, 두더지 맞추기 및 점수 계산, 시간 초과 시 두더지 사라짐, 그리고 게임 종료 조건에 대해 설명함.

### 두더지 생성

두더지는 일정 간격으로 랜덤한 위치에 생성됨. 두더지가 생성될 위치와 시간이 설정되고, 일정 확률로 폭탄 두더지가 생성됨.

```
if (startmole == 1) {
    startmole = 0;
    random_mole = (rand() % 9) + 1;
    timeleft = (rand() % 10);

    // 폭탄 생성 확률에 따라 폭탄 두더지 설정
    if (rand() % bomb_probability == 0)
        bomb_mole_index = random_mole - 1;
    else
        bomb_mole_index = -1;

    int p;
    for (p = 0; p < 8; p++) {
        if (bomb_mole_index == random_mole - 1)
            printmatrix[p] = bomb_mole[bomb_mole_index][p];
        else
            printmatrix[p] = mole[random_mole - 1][p];
    }
}
```

### 두더지 표시 및 입력 처리

두더지가 화면에 표시되며, 사용자의 입력을 감지하여 처리함. 텍스트 스위치의 입력을 읽어 사용자가 누른 버튼을 확인함.

```
dot = open(dot_d, O_RDWR);
if (dot < 0) {
    printf("dot error\n");
    exit(0);
}
write(dot, &printmatrix, sizeof(printmatrix));
usleep(100000);
close(dot);

tact = open(tact_d, O_RDWR);
read(tact, &c, sizeof(c));
close(tact);
switch (c) {
    case 1: num = 1; usleep(100000); break;
    case 2: num = 2; usleep(100000); break;
    case 3: num = 3; usleep(100000); break;
    case 4: num = 4; usleep(100000); break;
    case 5: num = 5; usleep(100000); break;
    case 6: num = 6; usleep(100000); break;
    case 7: num = 7; usleep(100000); break;
    case 8: num = 8; usleep(100000); break;
    case 9: num = 9; usleep(100000); break;
    case 12: isStop = 1; break;
}
```

## 두더지 맞추기 및 점수 계산

사용자가 두더지를 맞췄는지 검사하고, 맞췄을 경우 점수를 계산함. 폭탄 두더지를 맞추면 점수가 감소함.

```
// 두더지 맞췄는지 검사
if (num == random_mole) {
    // 폭탄 두더지를 클릭한 경우
    if (bomb_mole_index == random_mole - 1) {
        is_bomb = 1;
        score_player -= 2; // 점수 2점 차감
        printf("폭탄을 클릭했습니다! 점수: %d\n", score_player);
    } else {
        is_bomb = 0;
        count++;
        score_player += 1;
        printf("점수: %d\n", score_player);
    }

    clcd_input2(score_player, score_mole);

    // 매트릭스에서 맞은 두더지 제거
    int q2;
    for (q2 = 0; q2 < 8; q2++) {
        printmatrix[q2] = 0;
    }

    // 두더지 번호 재지정, 남은시간 재지정
    timeleft = rand() % 10;
    num = 0;
    random_mole = rand() % 9 + 1;

    // 폭탄 생성 확률에 따라 폭탄 두더지 설정
    if (rand() % bomb_probability == 0)
        bomb_mole_index = random_mole - 1;
    else
        bomb_mole_index = -1;

    // 새로운 두더지 표시
    int p;
    for (p = 0; p < 8; p++) {
        if (bomb_mole_index == random_mole - 1)
            printmatrix[p] = bomb_mole[bomb_mole_index][p];
        else
            printmatrix[p] = mole[random_mole - 1][p];
    }
}
```

## 시간 초과 시 두더지 사라짐

두더지를 제시 시간에 맞추지 못한 경우, 두더지가 사라지며 두더지 점수가 증가함.

```
// 두더지 못잡았을 경우
timeleft++;
if (timeleft > stagetime) {
    // 폭탄이 아닌 경우에만 두더지 점수 증가
    if (bomb_mole_index != random_mole - 1) {
        score_mole += 1;
        clcd_input2(score_player, score_mole);
        printf("두더지점수: %d \n", score_mole);
    }

    // 못잡은 두더지 매트릭스에서 제거
    int q2;
    for (q2 = 0; q2 < 8; q2++) {
        printmatrix[q2] = 0;
    }

    timeleft = 0;
    random_mole = rand() % 9 + 1;

    // 폭탄 생성 확률에 따라 폭탄 두더지 설정
    if (rand() % bomb_probability == 0)
        bomb_mole_index = random_mole - 1;
    else
        bomb_mole_index = -1;

    // 새로운 두더지 표시
    int p;
    for (p = 0; p < 8; p++) {
        if (bomb_mole_index == random_mole - 1)
            printmatrix[p] = bomb_mole[bomb_mole_index][p];
        else
            printmatrix[p] = mole[random_mole - 1][p];
    }
}
```

## 게임 종료 조건

게임 종료 조건은 사용자나 두더지의 점수가 30점 이상일 때임. 게임 종료 시 결과를 출력하고, 게임을 재시작할 수 있도록 60초 동안 대기함.

```
if (score_mole > 30 || score_player > 30) {
    if (score_mole >= 30) {
        printf("두더지 승리! 난이도를 다시 선택해주세요!\n");
        clcd_input2(score_player, score_mole);
    }

    if (score_player >= 30) {
        printf("플레이어 승리! 난이도를 다시 선택해주세요!\n");
        clcd_input2(score_player, score_mole);
    }

    sleep(60);
    break;
}
```

## 요약

게임 루프는 이와 같은 과정을 반복하여 두더지가 나타나고 사라지며, 사용자는 두더지를 잡기 위해 버튼을 누름. 이 과정에서 점수가 실시간으로 업데이트되고, 게임 종료 조건에 도달할 때까지 게임이 계속 진행됨.

### 3. 프로그램 종료 및 정리

게임이 종료되거나 사용자가 프로그램을 종료하려고 할 때, 프로그램은 파일 디스크립터를 정리하고, 필요한 경우 추가적인 종료 절차를 수행해야 함. 여기서는 DIP 스위치가 꺼질 때의 처리와 프로그램의 종료 절차에 대해 자세히 설명함.

#### DIP 스위치 값 읽기

게임 루프가 실행되는 동안, 주기적으로 DIP 스위치의 값을 읽어 상태를 확인함.

```
read(dip, &dipval, sizeof(dipval));
```

#### DIP 스위치가 꺼진 경우의 처리

DIP 스위치의 값이 0이면 모든 DIP 스위치가 꺼진 상태이므로, 프로그램을 종료함.

```
if (dipval == 0) {  
    printf("게임을 종료합니다.\n");  
    break;  
}
```

#### LED 끄기

LED 디바이스에 0x00 값을 써서 모든 LED를 끄. 이는 DIP 스위치가 꺼진 상태에서 LED가 여전히 켜져 있는 것을 방지함.

```
led_data = 0x00;  
write(led_dev, &led_data, sizeof(unsigned char)); // 모든 DIP 스위치 LED 끄기
```

#### 파일 디스크립터 닫기

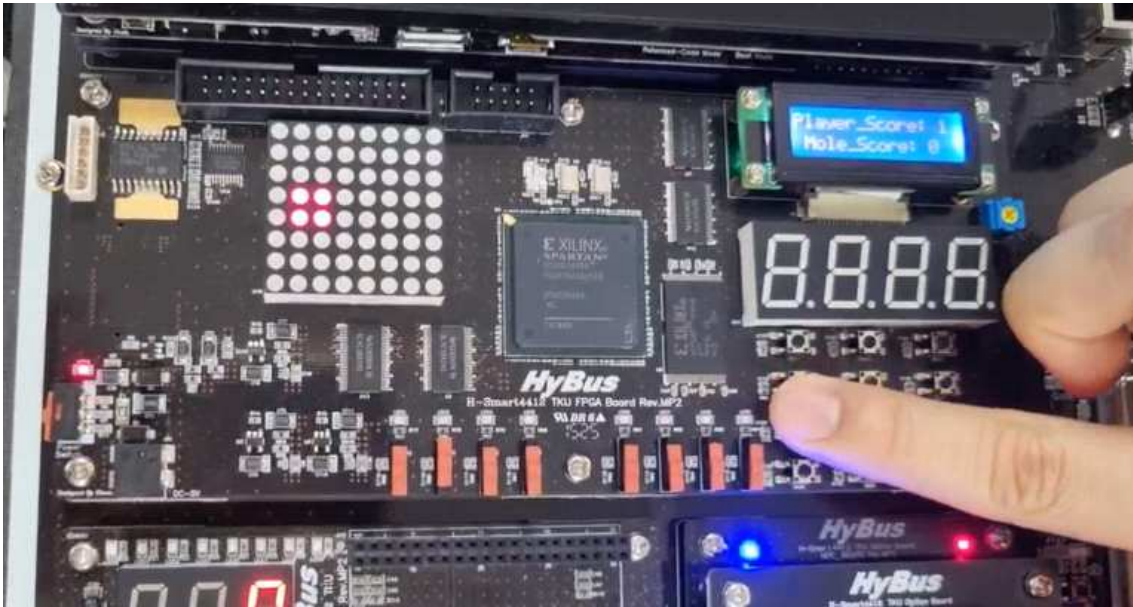
게임이 종료되면, 열려 있는 모든 디바이스 파일을 닫음. 여기에는 DIP 스위치, LED, 점 매트릭스, 텍스트 스위치, 문자 LCD 디바이스 파일이 포함됨.

```
close(dip);  
close(led_dev);
```

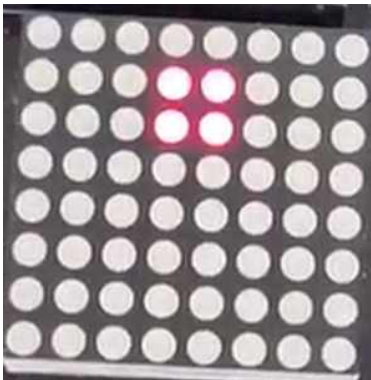
#### 요약

프로그램 종료 및 정리 절차는 게임이 종료되거나 사용자가 프로그램을 종료할 때 시스템 리소스를 적절히 해제하고, 디바이스 파일을 닫아 다른 프로그램이 사용할 수 있도록 하는 중요한 단계임. DIP 스위치의 값을 계속해서 읽어 상태를 확인하고, 모든 DIP 스위치가 꺼지면 게임을 종료하고 필요한 정리 작업을 수행함. 이러한 절차를 통해 프로그램이 안정적으로 종료될 수 있음.

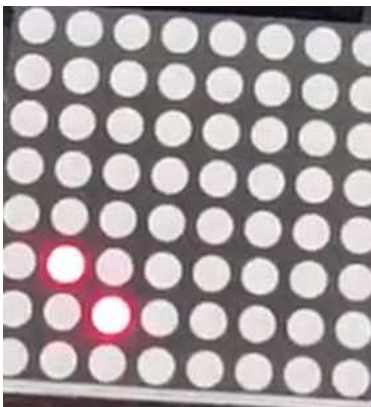
#### IV. 동작 확인



정상적으로 게임이 작동되는 것을 확인할 수 있었음.



해당 경우, 두더지를 나타냄. 잡을 경우 1점을 얻음.



해당 경우, 폭탄 두더지를 나타냄. 잡을 경우 2점이 깎임.





## V. 요약 및 역할분담

두더지 게임 프로그램은 임베디드 시스템 환경에서 작동하는 게임으로, 사용자가 랜덤하게 나타나는 두더지를 텍스트 스위치를 사용해 잡는 재미를 제공함. 이 프로그램은 다양한 하드웨어 장치와 상호작용하며, 게임의 많은 측면을 효과적으로 관리하는 방법을 보여줌. 프로그램은 DIP 스위치를 통해 난이도를 설정하고, LED를 사용해 현재 설정된 난이도를 시각적으로 표시함. 또한, 점 매트릭스 디스플레이를 통해 두더지와 폭탄 두더지를 화면에 표시하고, 문자 LCD를 통해 실시간으로 점수를 출력함. 또한 프로그램의 시작 단계에서 난이도를 설정하는 과정은 사용자 경험을 개인화할 수 있는 중요한 부분임. DIP 스위치의 상태를 읽어 난이도를 결정하고, 이를 LED를 통해 사용자에게 명확하게 전달하며, 다양한 난이도를 제공함으로써 게임은 초보자부터 숙련자까지 모두 즐길 수 있는 유연성을 갖추게 됨. 난이도에 따라 두더지가 화면에 머무는 시간이 조절되며, 이는 게임의 도전성을 조절하는 핵심 요소임. 게임 루프는 프로그램의 중심으로, 두더지가 나타나고 사라지며, 사용자의 입력을 처리하는 일련의 과정을 반복함. 게임 루프 내에서 두더지의 생성, 표시, 사용자 입력 처리, 점수 계산, 시간 초과 처리 등이 이루어지며, 두더지가 랜덤하게 나타나고 사라지기 때문에 게임은 항상 새롭고 흥미롭게 유지됨. 또한, 폭탄 두더지의 추가로 인해 게임은 더욱 긴장감 넘치는 경험을 제공함. 사용자가 실수로 폭탄 두더지를 누를 경우 점수가 감소하기 때문에 주의 깊게 플레이해야 함을 강조할 수 있었음. 또한 사용자는 DIP 스위치를 통해 난이도를 손쉽게 설정하고, LED와 문자 LCD를 통해 현재 상태와 점수를 직관적으로 확인할 수 있음. 게임의 목표는 단순하지만, 빠른 반응 속도와 집중력이 요구되기 때문에 짧은 시간 안에 재미와 도전을 동시에 제공함. 폭탄 두더지의 추가로 인해 게임은 단순히 두더지를 누르는 것 이상의 전략적 요소를 포함하게 되며, 이는 게임의 깊이를 더해줌. 또한 본 프로젝트는 팀플레이와 협업을 통해 성공적으로 완성할 수 있었음. 각자 다양한 역할을 분담하여 각 팀원이 자신의 강점을 최대한 발휘할 수 있도록 하였음. 또한 효과적인 의사소통을 통해 팀 프로젝트를 성공시켰으며, 정기적인 회의 및 문제 해결을 위한 의견 표출 활성화하여 협업의 효율성을 높일 수 있었음. GitHub를 활용하여 프로젝트 진행 상황을 실시간으로 공유하였으며, 지속적으로 코드를 통합하고 테스트하였음. 결론적으로 본 프로젝트 주제인 두더지 게임 프로그램은 단순한 게임이지만, 다양한 기술적 요소가 결합된 프로젝트임. 임베디드 시스템 환경에서 실행되며, 다양한 하드웨어 장치와의 상호작용을 통해 사용자는 게임을 즐길 수 있도록 난이도 설정, 두더지 생성 및 표시, 사용자 입력 처리, 점수 계산 등 게임의 많은 측면이 통합되어 있음. 본 2달간의 과정은 통해 팀원 간의 협업능력을 향상 시키고 프로그래밍에 대한 깊은 이해도를 가질 수 있는 기회였다고 판단됨.

맡은 역할은 아래와 같음.

2019250027 안병한: 두더지 게임 순서도 작성, 폭탄 기능 코드 작성

2019250036 이상민: 하드웨어 소프트웨어 통합 및 테스트, 시스템 통합 및 연동, 코드 최적화 및 디버깅, 임베디드 시스템 관련 코드 설계 구현

2019250041 이정민: 자료 조사, 두더지 게임 기본 룰 구상

2020740067 장민석: 보고서 작성, 발표, ppt 작성, 자료 조사



## 참 고 자 료

- [1] 임베디드 개발환경 - ⑤ 타겟시스템 리눅스 설치 . (2022). <https://takethat.tistory.com/21>.
- [2] H-Smart4412TKU 사용법 전자정보공학과 이 종 복. . (2019). <https://slidesplayer.org/slide/14109337/>.
- [3] [전공-임베디드] 임베디드 015 - 개발키트 내의 디바이스를 활용한 간단한 게임 애플리케이션 제작 - 두더지 잡기 . (2016). [https://m.blog.naver.com/Post\\_View.naver?isHttpsRedirect=true&blogId=bieemiho92&logNo=220744722833](https://m.blog.naver.com/Post_View.naver?isHttpsRedirect=true&blogId=bieemiho92&logNo=220744722833).
- [4] 2022HKNUiotprogrammingTeam1 . (2022). <https://github.com/2022HKNUiotprogrammingTeam1/project>.