LG 부트캠프 10기

프로젝트 결과보고서

SOLE snow

밸런스 보드를 활용한 2D 스키 게임 구현

Linux System 반 5팀

이승엽, 예종호, 남주형, 김재욱

목차구성 CONTENTS COMPOSITION

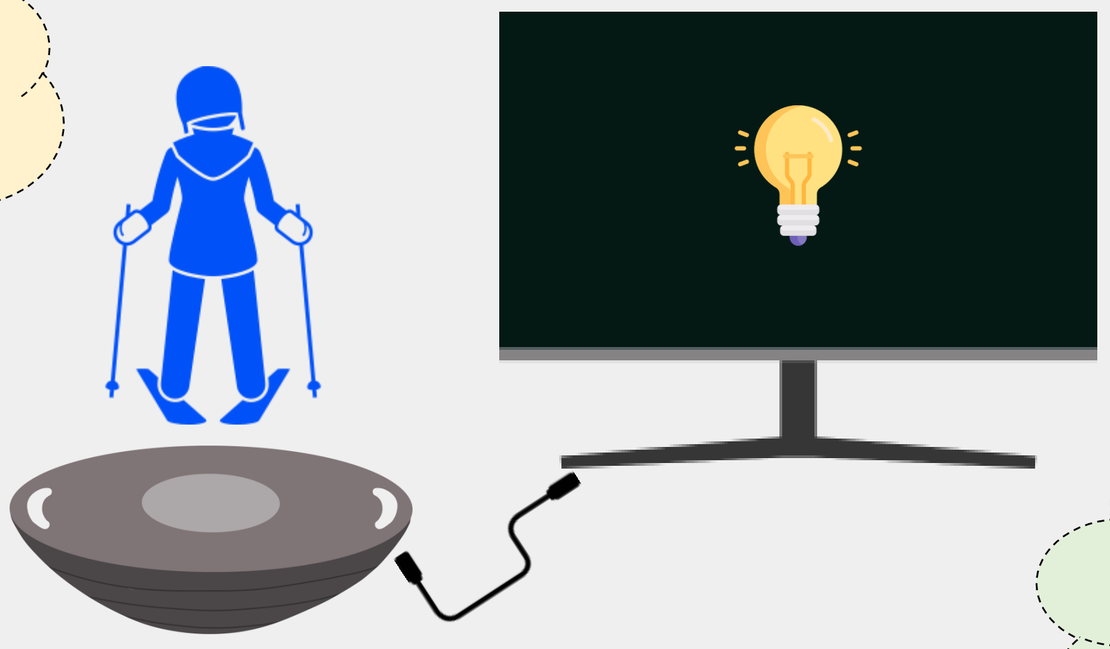
1. 주제 및 개발 아이디어
2. 게임 소개 및 조작 방법
3. 핵심 기술
4. 시연 영상

1. 주제 및 개발 아이디어

더운 여름에도 스노우 보드의 재미를 느낄 수 있을까?

밸런스 보드 움직임 활용한 스노우 보드 액션 게임을 구상

보드를 타는 듯한
역동적인 재미



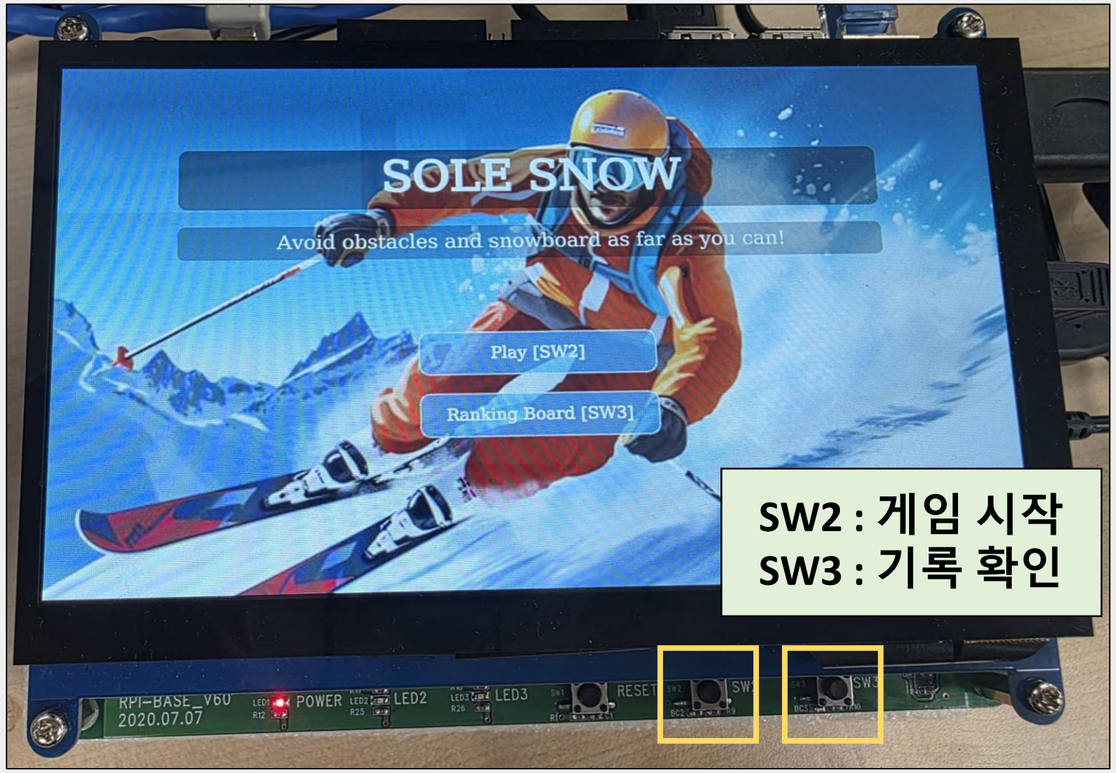
쏠쏠한
코어 근육 운동

열심히 배운 임베디드 지식
활용을 통한 게임 개발

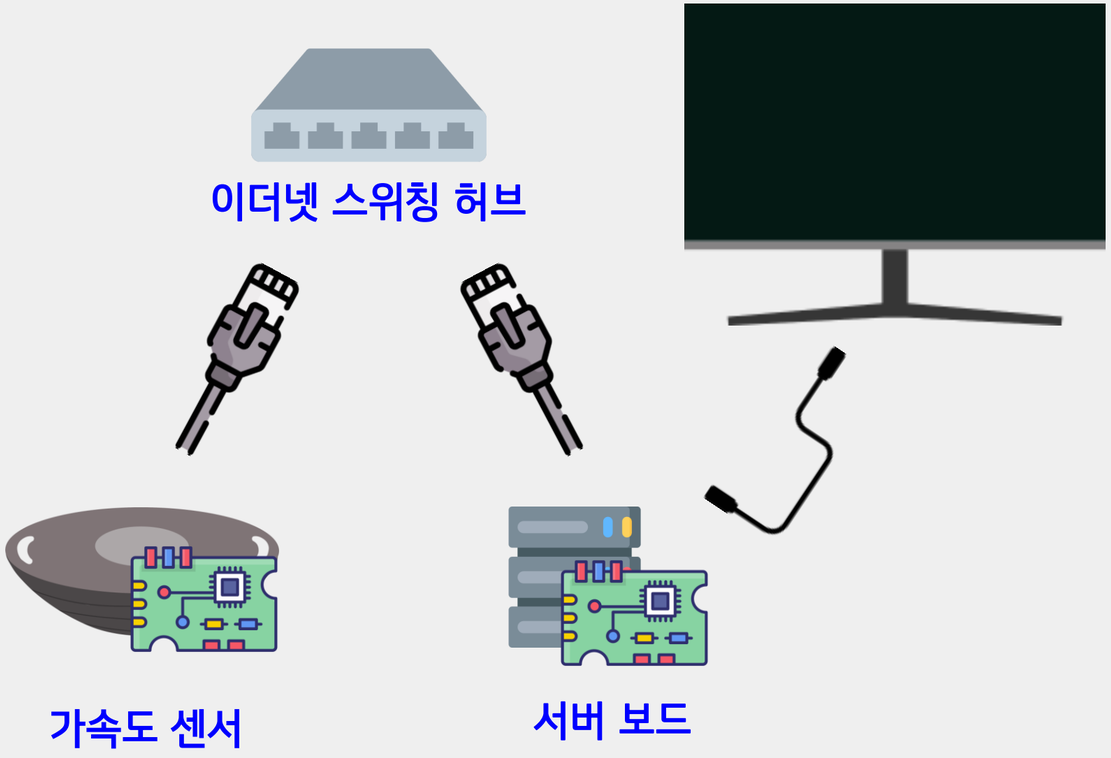
2. 게임 소개 및 조작 방법 (메인 화면 및 HW)

게임은 누구나 쉽게 접할 수 있으면서도 모두가 재미있게 해야 하는 것

남녀노소 누구나 손쉽게 조작할 수 있는 직관적인 UI를 표현



게임 메인 화면

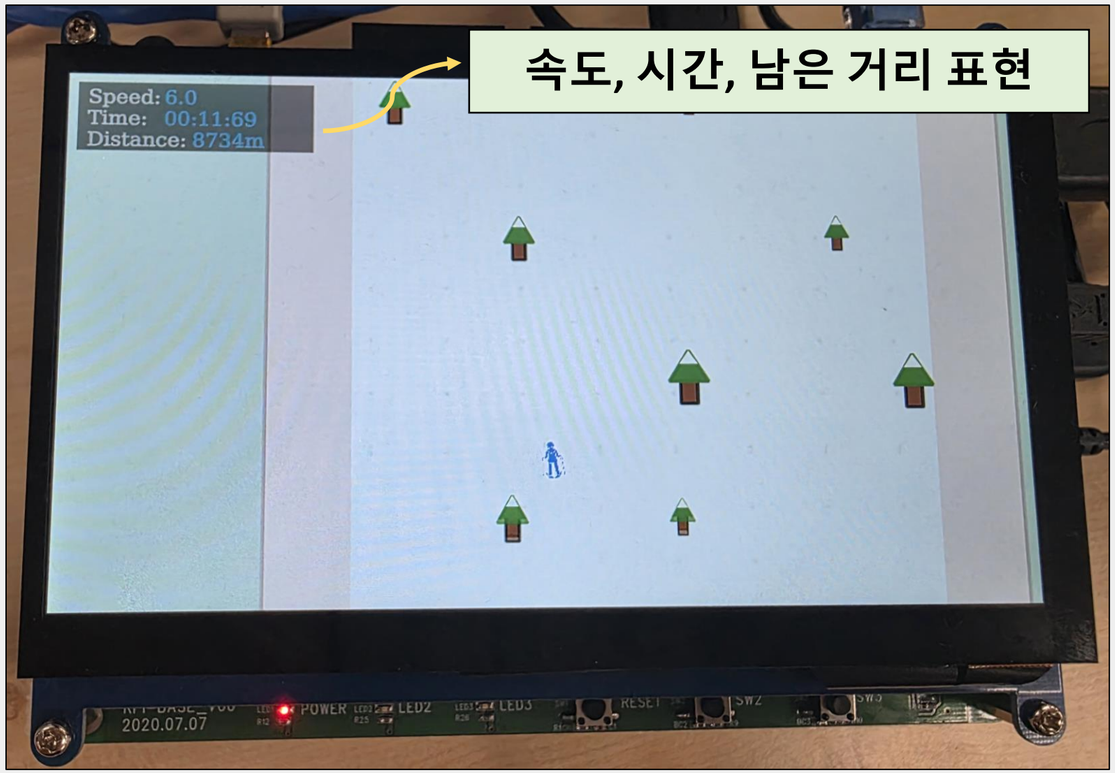


HW 설계 구조

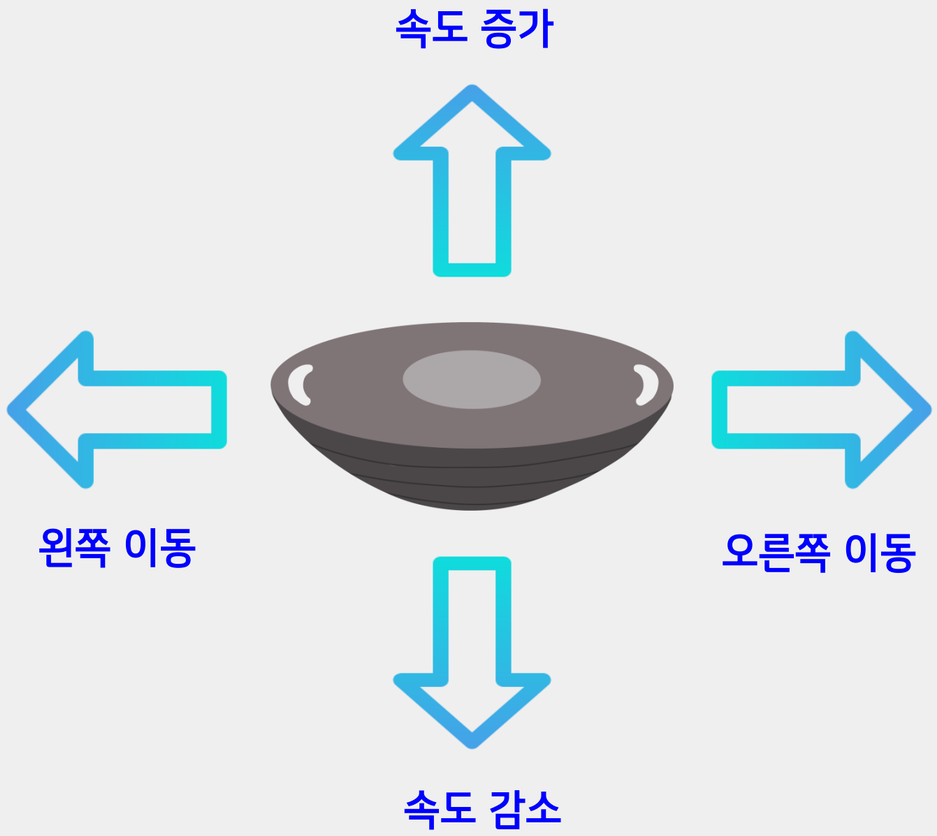
2. 게임 소개 및 조작 방법 (게임 플레이)

게임은 누구나 쉽게 접할 수 있으면서도 모두가 재미있게 해야 하는 것

남녀노소 누구나 손쉽게 조작할 수 있는 직관적인 UI를 표현



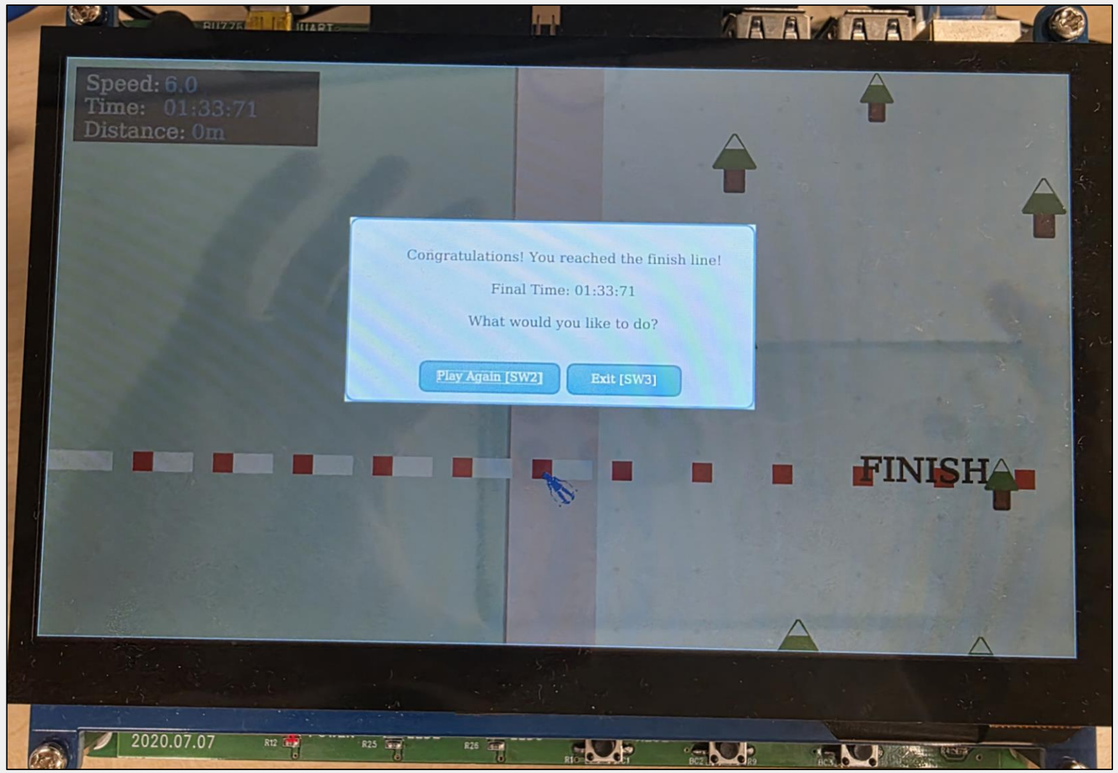
게임 플레이 화면



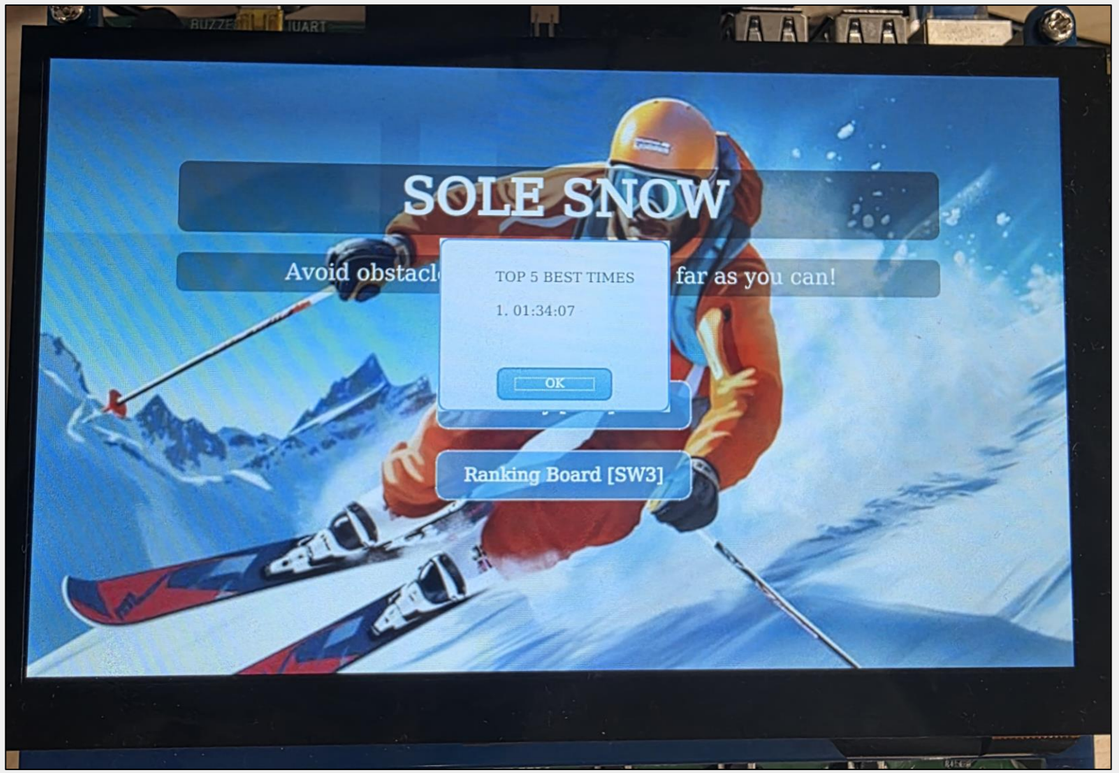
2. 게임 소개 및 조작 방법 (기록 확인)

결승선 완주할 경우, 완주 시간을 랭킹 보드에 기록

완주 기록은 서버에서 상위 5개 레코드까지 저장하고 있다가 Ranking Board 버튼을 누를 시 보여줌



결승선 완주 화면



Ranking Board

3. 핵심 기술 (커널)

커널 포팅 및 개발 환경 구성

VirtualBox 환경 구성

ubuntu

Ubuntu 24.04 LTS

Ubuntu 24.04 LTS

Visual Studio Code

Visual Studio Code 1.101

GitHub Copilot

Copilot 1.341.0

Ubuntu upgrade 후 최신버전 vscode 설치를 통해 copilot 사용을 위한 개발 환경 세팅

최신 Linux 커널 포팅

Linux 6.12.35 LTS 커널

BCM2837(라즈베리파이3) 보드 최적화

BCM2837(BCM2835 계열) 보드에 맞는 커널 빌드, 하드웨어 호환성 보장

실시간 센서 데이터 처리

실시간 커널 프리엠션, 고해상도 타이머로 센서 데이터 신속 처리

다양한 인터페이스 지원

이더넷, USB, 시리얼 등 다양한 통신 인터페이스 지원 (센서 데이터 송수신 및 보드 간 통신)

루트 파일 시스템 구성

Glibc(GCC Linaro Toolchain)

ubuntu 16.04에서는 7.5-2019.12이 최신
ubuntu 24.04에서는 14.0.0-2023.06 사용 가능

ALSA

최신 버전의 alsa-lib-1.2.14,
최신 버전의 alsa-utils-1.2.14 적용

부팅 설정

부팅 시 바로 모든 설정이 완료되고
게임 실행되도록 설정

3. 핵심 기술 (디바이스 드라이버)

가속도 센서 드라이버 포팅 및 각도 환산 알고리즘 설계

가속도 센서 최신 Linux 커널에 포팅

* Device Drivers → Misc devices

```
    Texas Instruments shared transport line discipline  --->
<*> STMicroeletronics LIS3LV02Dx three-axis digital accelerometer (SPI)
< > STMicroeletronics LIS3LV02Dx three-axis digital accelerometer (I2C)
< > Altera FPGA firmware download module
```

* Device Drivers → SPI Support

```
<*>   BCM2835 SPI controller
<*>   BCM2835 SPI auxiliary controller
<*>   Broadcom BSPI and MSPI controller support
```

Linux 6.12.35 LTS 커널 menuconfig 설정

```
spi: spi@7e204000 {
    compatible = "brcm,bcm2835-spi";
    reg = <0x7e204000 0x200>;
    interrupts = <2 22>;
    clocks = <&clocks BCM2835_CLOCK_VPU>;
    #address-cells = <1>;
    #size-cells = <0>;
    status = "okay";
};
```

spi 노드 설정
status 활성화

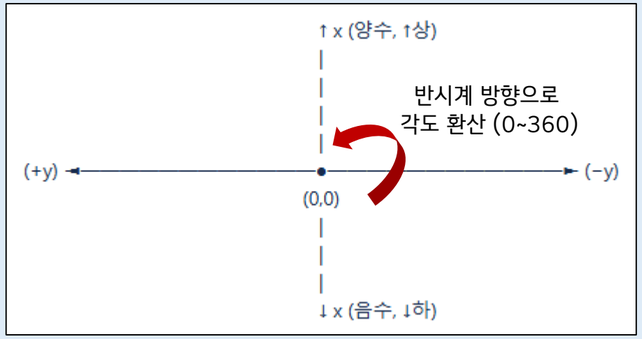
arch/arm/boot/dts/Broadcom/bcm283x.dtsi

```
lis302@0 {
    compatible = "st,lis302dl-spi";
    reg = <0>;
    spi-max-frequency = <10000>;
    interrupt-parent = <&gpio>;
    interrupts = <16 0>;
```

가속도 센서 노드를
디바이스 트리에 추가

arch/arm/boot/dts/Broadcom/wt2837.dts

가속도 센서 각도 환산 알고리즘 설계



가속도 센서 값에 따른 좌표평면 (x 상하 / y 좌우)

축	.code	.value (-900 ~ +900)		방향
x	0	+	0~100 (기우는 정도)	상
x	0	-	0~100 (기우는 정도)	하
y	1	+	0~100 (기우는 정도)	좌
y	1	-	0~100 (기우는 정도)	우

tan 역함수를 활용하여 좌표 평면에 대한 값을 각도로 환산

.value 값의 범위를 0~100으로 정규화하여 기우는 정도를 표현

```
(27, 9) [3] 108 degree
(24, 9) [3] 110 degree
(24, 11) [3] 114 degree
(24, 11) [3] 114 degree
(24, -3) [3] 82 degree
(12, -3) [1] 75 degree
(12, 7) [2] 120 degree
(12, 9) [2] 126 degree
(12, 11) [2] 132 degree
(12, 11) [2] 132 degree
(12, 8) [2] 123 degree
(12, 8) [2] 123 degree
(12, -36) [4] 18 degree
(12, -36) [4] 18 degree
(12, 0) [1] 90 degree
(12, 0) [1] 90 degree
(72, 0) [8] 90 degree
(72, -36) [9] 63 degree
(72, -36) [9] 63 degree
(126, -36) [15] 74 degree
(126, -45) [15] 70 degree
(126, -45) [15] 70 degree
(126, -47) [15] 69 degree
(90, -47) [11] 62 degree
(99, -47) [12] 64 degree
(99, -44) [12] 66 degree
```

가속도 센서 각도 환산 로그

3. 핵심 기술 (디바이스 드라이버)

GPIO KEY 인터럽트 구현 및 응용

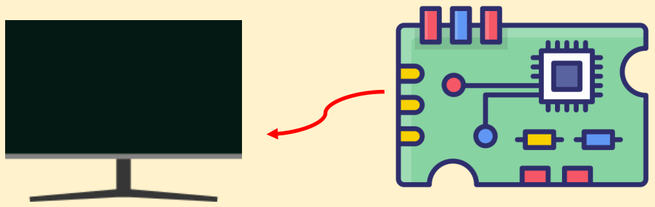
GPIO KEY 인터럽트 구현

커널 버전	GPIO PIN 번호
Linux 6.1.63	17번
Linux 6.12.35	529번 (17+512)

Linux 커널 버전에 따른 GPIO PIN 번호 변동 사항

버튼	용도
SW2	게임 시작 (Play 버튼)
SW3	기록 확인 (Ranking Board 버튼)

bcm2837 보드 내 SW2, SW3 버튼 용도 정리

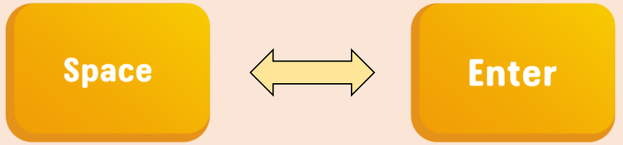


모니터 화면에 게임 실행 화면을 보여줌 (터치 기능 사용 불가)
보드 기본 버튼 조작을 통해 세부적인 메뉴를 실행

게임 앱 내에서 GPIO 컨트롤 및 관련 디바이스 탐색 로직 설계

GPIO PIN	가상 키보드 활용
GPIO 529	KEY_SPACE 매핑
GPIO 530	KEY_ENTER 매핑

앱에서 이벤트 발생을 컨트롤 하기 가상 키보드 활용



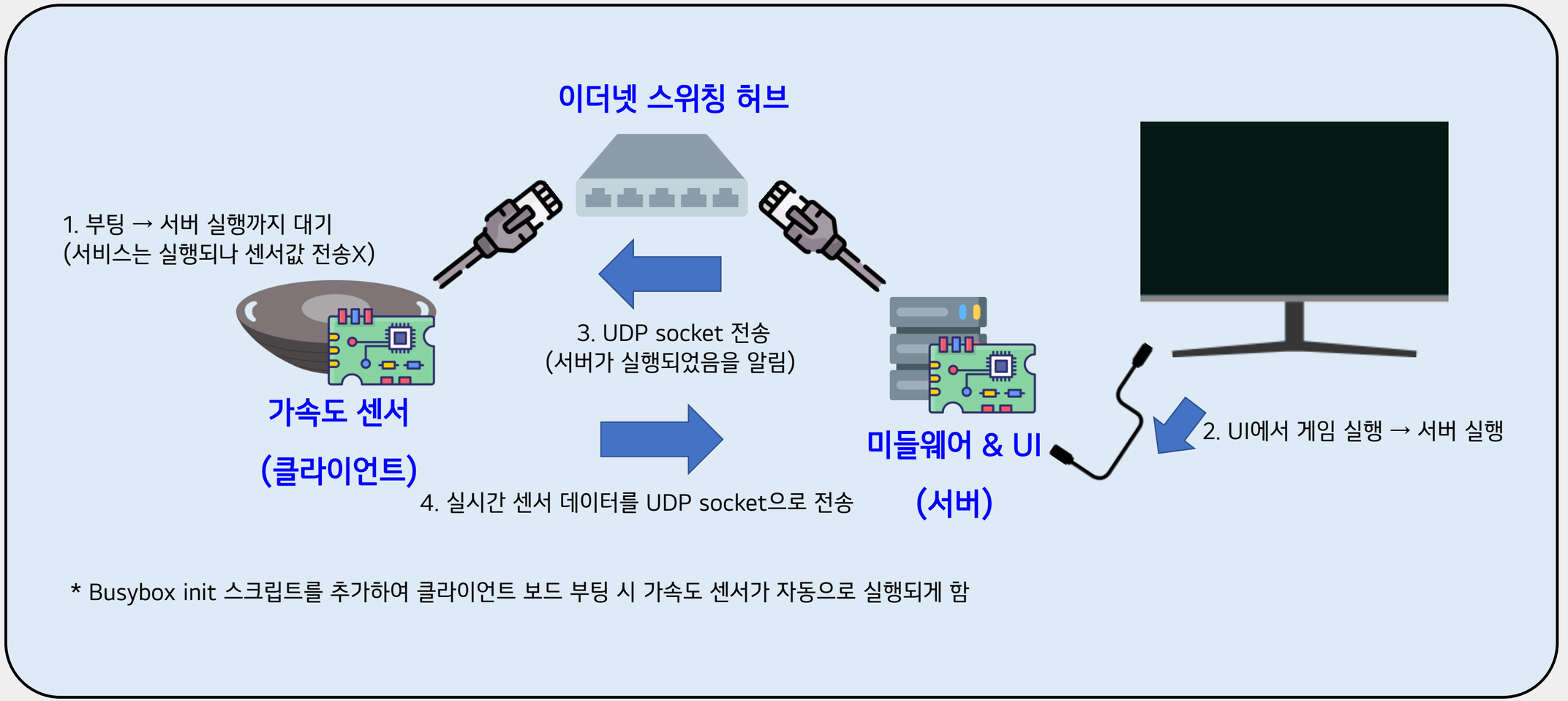
각 버튼에 대한 이벤트 실행 시 다른 버튼 비활성화

```
root@HT2837:/mnt/nfs# ./snow_board
Attribute Qt::AA_EnableHighDpiScaling must be set before QCoreApplication is created.
Found device by direct check: "/dev/input/event0"
Found device by direct check: "/dev/input/event1"
Found device by direct check: "/dev/input/event2"
Found input device: "/dev/input/event0" Name: "ST LIS3LV02DL Accelerometer"
Found input device: "/dev/input/event1" Name: "深▼市全▼▼子技▼有限公司 ByQDtech ▼控USB盘▼"
Found input device: "/dev/input/event2" Name: "GPIO Key Input"
Found GPIO device: "/dev/input/event2"
```

게임 앱 실행 시 자동으로 연결된 디바이스를 탐색하여 GPIO 관련 디바이스를 찾는 로직 추가

3. 핵심 기술 (어플리케이션)

실시간 데이터 전송을 위한 비연결성 프로토콜(UDP) 활용

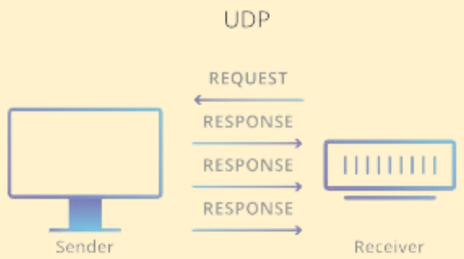


* Busybox init 스크립트를 추가하여 클라이언트 보드 부팅 시 가속도 센서가 자동으로 실행되게 함

3. 핵심 기술 (어플리케이션)

네트워크 환경 구축

UDP 통신



실시간 센서 데이터 처리
↓
지연시간 단축

이더넷 스위칭 허브

서로 다른 IP 간 통신
&
여러 기기 연결 가능

Telnet

Telnet

두 보드에 동시 원격 접속 가능
↓
실시간 송수신 데이터 확인 용이

3. 핵심 기술 (어플리케이션)

UI

QPainter 기반 2D 그래픽 렌더링

QPainter를 활용해 스키어, 나무, 배경, 정보 UI 등
모든 게임 오브젝트 표현
스키어의 좌우 회전과 중심 기준 구현

충돌 판정 영역 커스터마이징

스키어와 나무 각각 실체 크기와 모양에 맞춰
충돌 영역을 별도로 계산하여,
현실감 있는 충돌 판정 구현

월드 좌표계와 화면 좌표계 변환

월드 좌표계를 화면 좌표계로 변환하면서 맵 스크롤 효과 구현

월드 좌표계 => 맵 기준 게임 오브젝트의 고정 위치
화면 좌표계 => 실제 화면에 표현할 위치

```
for (const auto& tree : m_trees) {
    QPointF treePos = tree.position();
    int screenY = m_mapOffset - treePos.y(); // 월드 y → 화면 y
    int screenX = width()/2 + treePos.x() - m_cameraOffsetX; // 월드 x → 화면 x

    // 실제 그리기
    // painter.drawRect(screenX + ..., screenY + ..., ...);
}
```