



리눅스 디바이스 드라이 스마트 ~~버~~지털 시계

유종민 | 임정민 | 허진경



목차보기

01
프로젝트 개요

02
프로젝트 목표

03
HW

04
S/W

05
요소 기술

06
Trouble Shooting



프로젝트 개요



프로젝트 개요



- DS1302 RTC 모듈을 이용한 시간 정보 제공
- DHT11 센서를 이용한 온습도 정보 제공
- 로터리엔코더 인터럽트를 활용한 직관적인 시간 수정
- OLED 디스플레이를 통한 실시간 날짜/시간 /온습도 정보 시각화



프로젝트 목표



프로젝트 목표



핵심 목표

- OLED 디스플레이를 통한 실시간 날짜/시간 /온습도 정보 시각화

구현 범위

- Device Driver 구현

RTC: 레지스터 주소로부터 시간 데이터를 가져와 BCD값을 2진수로 변환

온습도 모듈: DHT11 핸드셰이크 프로토콜 구현

Rotary Encoder: GPIO 인터럽트 처리, 회전 방향 판별, 디바운싱 적용

- User Application 구현

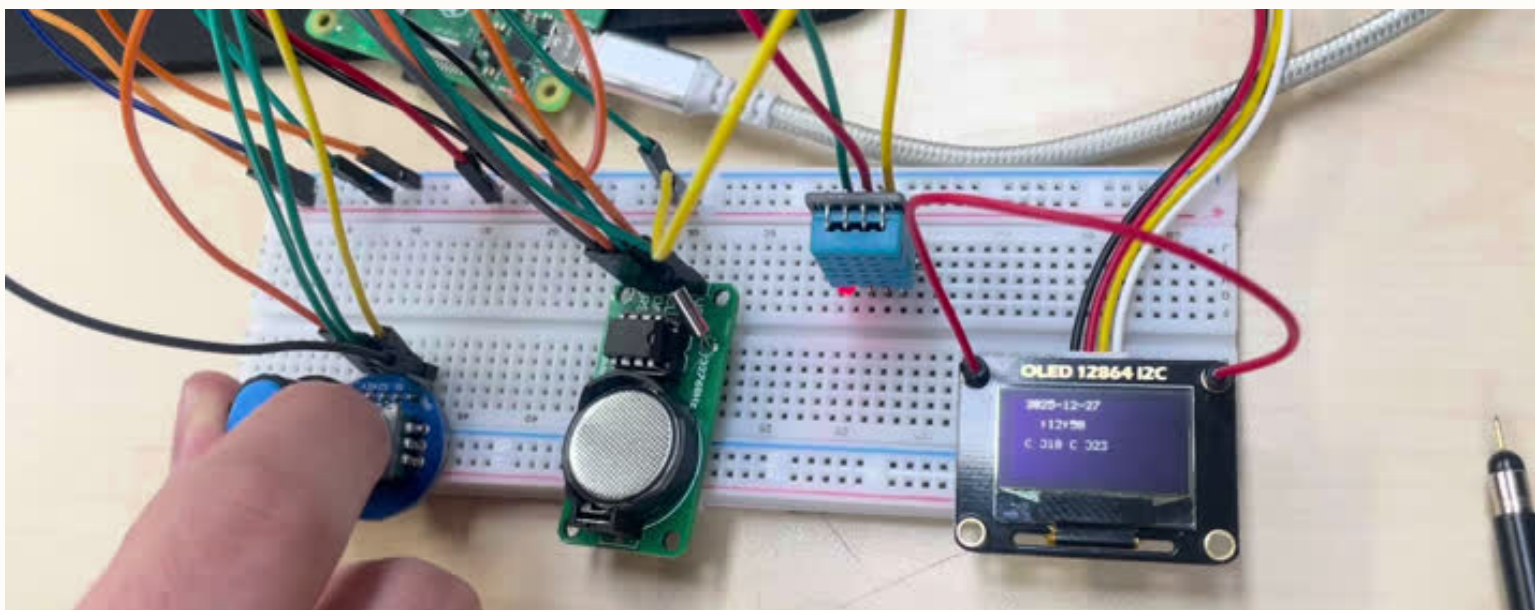
I2C 프로토콜을 이용하여 OLED로 데이터 전송

시계 모드 ↔ 수정 모드 상태 전이 로직 구현

HW



프로젝트 사진



HW

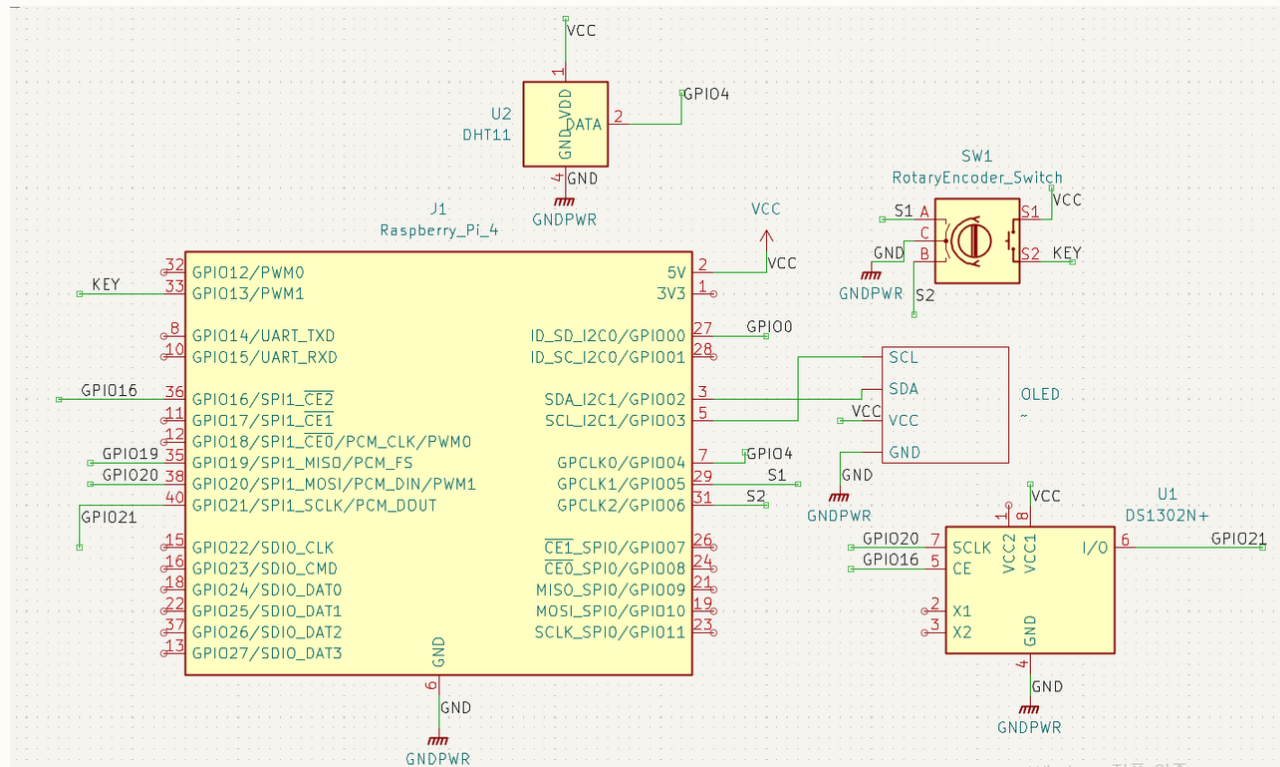


BOM (Bill Of Material)

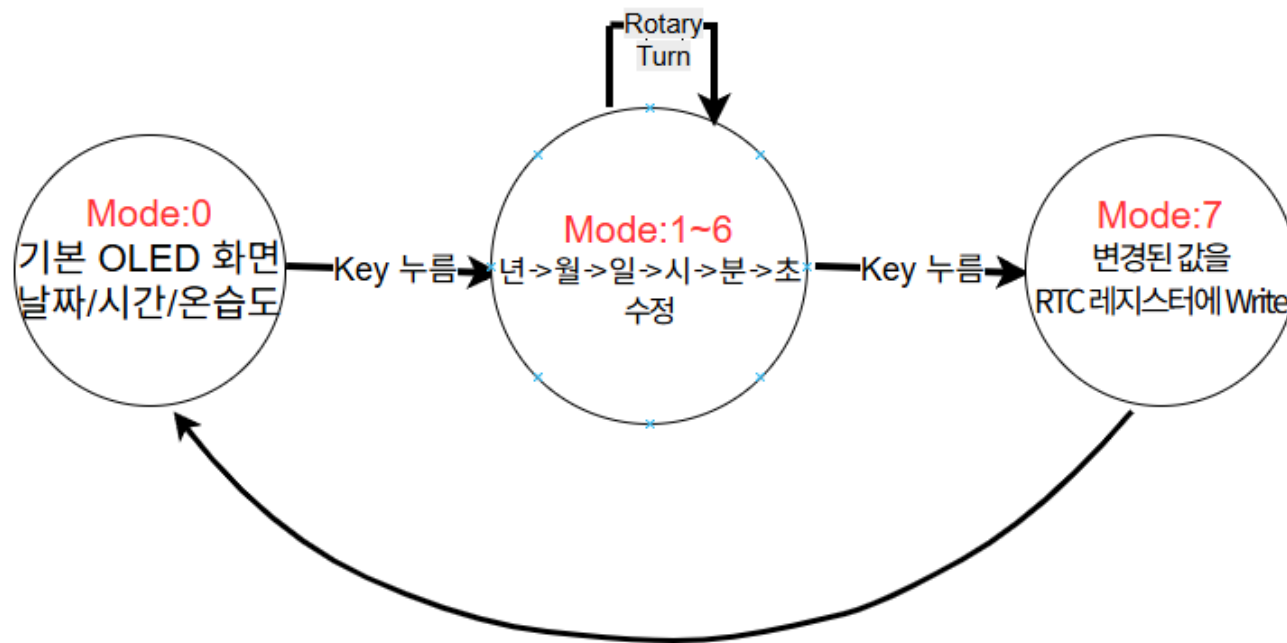
부품명	모델명/규격	용도
SBC	Raspberry Pi 4	메인 컨트롤러
Display	0.96" OLED (I2C)	시간/날짜 출력
Input	Rotary Encoder	UI 조작 (회전/클릭)
RTC	DS1302	Real Time Clock
Sensor	DHT11	온도/습도 측정
Etc	Jumper Wires, Breadboard	회로 구성

HW

회로도



상세 설계 (FSM: Finite State Machine)





S/W



Source Code Review (핵심 코드 설명) - DHT11 온습도 센서

- **START 신호(센서 깨우기)**

- mdelay(20): 데이터핀을 **LOW**로 18ms 이상 유지

- udelay(30): 잠깐 **HIGH**로 전환 후 입력 모드로 변경

- **무한 대기 방지(안전장치)**

- wait_for_level()+ TIMEOUT_US

- 원하는 HIGH/LOW가 안 오면 타임아웃으로 실패 처리

- **데이터(40비트) 읽는 핵심 원리**

- 각 비트는 **HIGH**가 유지되는 시간으로 0/1 판단

- high_len > 40us이면 1, 아니면 0

- **데이터가 맞는지 검증**

- 체크섬 확인: 앞 4바이트 합이 마지막 바이트와 같아야 함

- 아니면 데이터 오류로 실패 처리



요소 기술



I2C (Inter-Integrated Circuit) – OLED 제어용

개요

- 근거리용 저속동기식 직렬 통신 규격
- 2선(SCL, SDA)으로 구성됨

동작 방식

- Master 가 SCL 을 생성하고, SDA 로 데이터를 장치에 전송
- 각 장치는 I2C 주소로 구분됨

특징

- 배선이 단순해서 회로 구성이 쉬움
- 여러 장치를 같은 버스에 병렬로 추가가능
- SPI통신에 비해 속도는 느림
- SCL, SDA 모두 Open drain 방식으로 Pull-up 저항 필요



요소 기술



문자 디바이스드라이버 & GPIO 인터럽트

개요

- 리눅스 커널이 하드웨어를 파일처럼 다룰 수 있게 해주는 인터페이스

동작 방식

- /dev/ds1302, /dev/rotary 장치 파일을 생성.
- Application이 read()를 호출하면 커널이 하드웨어 상태를 읽어 반환
- Polling 방식 대신 Interrupt 방식을 사용하여 버튼이 눌렸을 때만 이벤트 발생.

장점

- 시스템 리소스 효율적 사용, 하드웨어 제어 코드의 모듈화 및 재 사용성 증대.

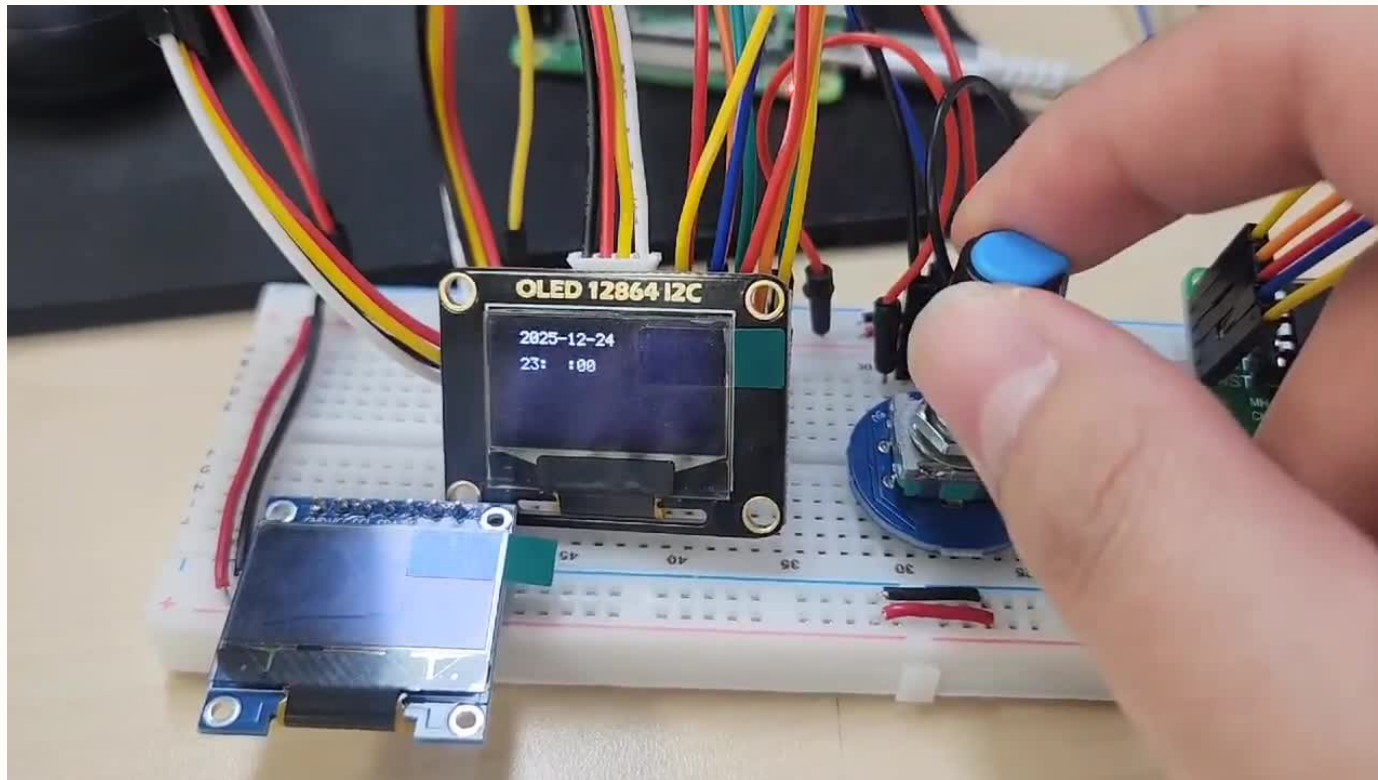


Trouble Shooting



현상 (Symptom)	원인 (Cause)	조치 (Solution)
OLED 화면 깨짐 / 노이즈	OLED 초기화 시 메모리 주소 지정 모드 (Addressing Mode) 설정 오류	초기화 코드에 0x20(Set Memory Mode), 0x02(Page Addressing Mode) 명령어 추가하여 해결
시간 저장 안 됨	DS1302의 쓰기 방지(WP) 비트 활성화 및 Clock Halt(CH) 비트 설정	Write 함수에서 0x8E(WP) 레지스터를 0x00으로 해제하고, 초(Sec) 기록 시 0x80 비트(CH)를 0으로 클리어함
버튼 중복 입력 (체터링)	기계적 스위치 접점의 미세한 떨림으로 인해 인터럽트가 수십 번 발생	커널 드라이버: jiffies를 이용해 250ms 이내 재입력 무시 앱: usleep 및 이벤트 버퍼 비우기(flush) 로직 추가
"SAVED" 글자 안 보임	폰트 배열(Font Array)에 알파벳 데이터가 0x00(공백)으로 비어있었음	전체 ASCII 폰트 데이터를 배열에 추가하여 문자 출력 정상화

동작 동영상





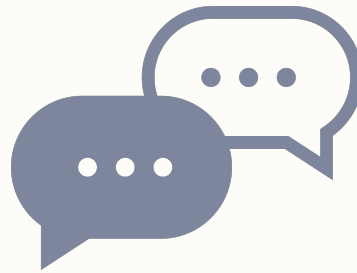
결론 및 시사점



- 리눅스 디바이스 드라이버 구조를 직접 구현하며 User ↔ Kernel ↔ Hardware 흐름을 이해
- 폴링방식이 아닌 인터럽트 기반 설계로 CPU 효율 향상
- 단순 제어가 아닌 확장 가능한 드라이버 구조 설계 경험



질문과 답변



궁금하신 점이 있으신가요?
자유롭게 질문해 주세요.



감사합니다.