Washing Machine Project

With ATmega128A

(Telechips) AI 시스템반도체 SW 개발자

박성호 이경주

TABLE OF CONTENTS

01 프로젝트 개요

프로젝트 목적

- AVR의 ATmega128A 마이크로컨트롤러에 대한 이해 및 응용
- LED, BUTTON, FND, PWM, MOTOR, BUZZER를 프로젝트에 활용

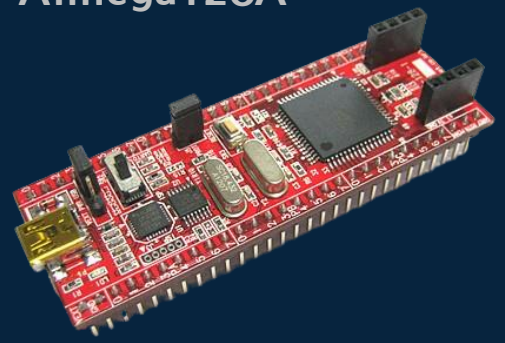
프로젝트 내용

- 세탁기의 동작 구현 (세탁, 헹굼, 탈수)
- 세탁기 기능의 가시적 효과 표현 (LED, FND)

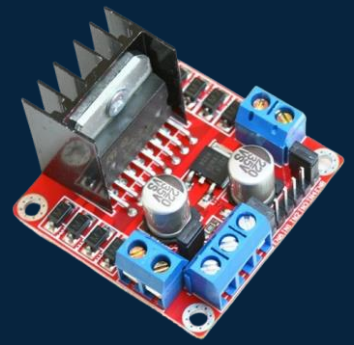
01 프로젝트 개요

주요 부품

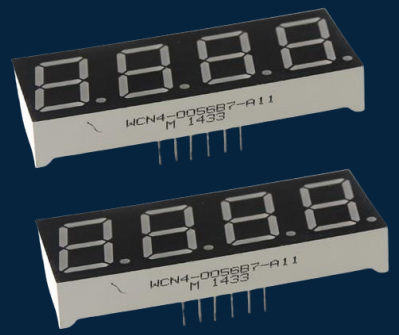
ATmega128A



L298N Motor Drive

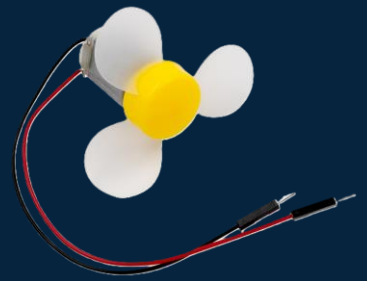


FND

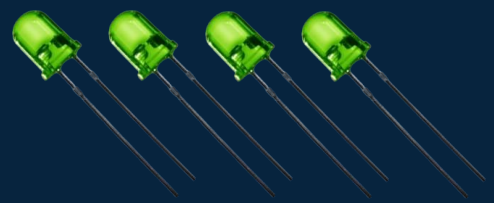


BUTTON

MOTOR

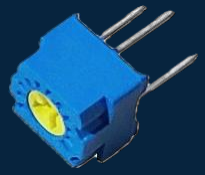


LED



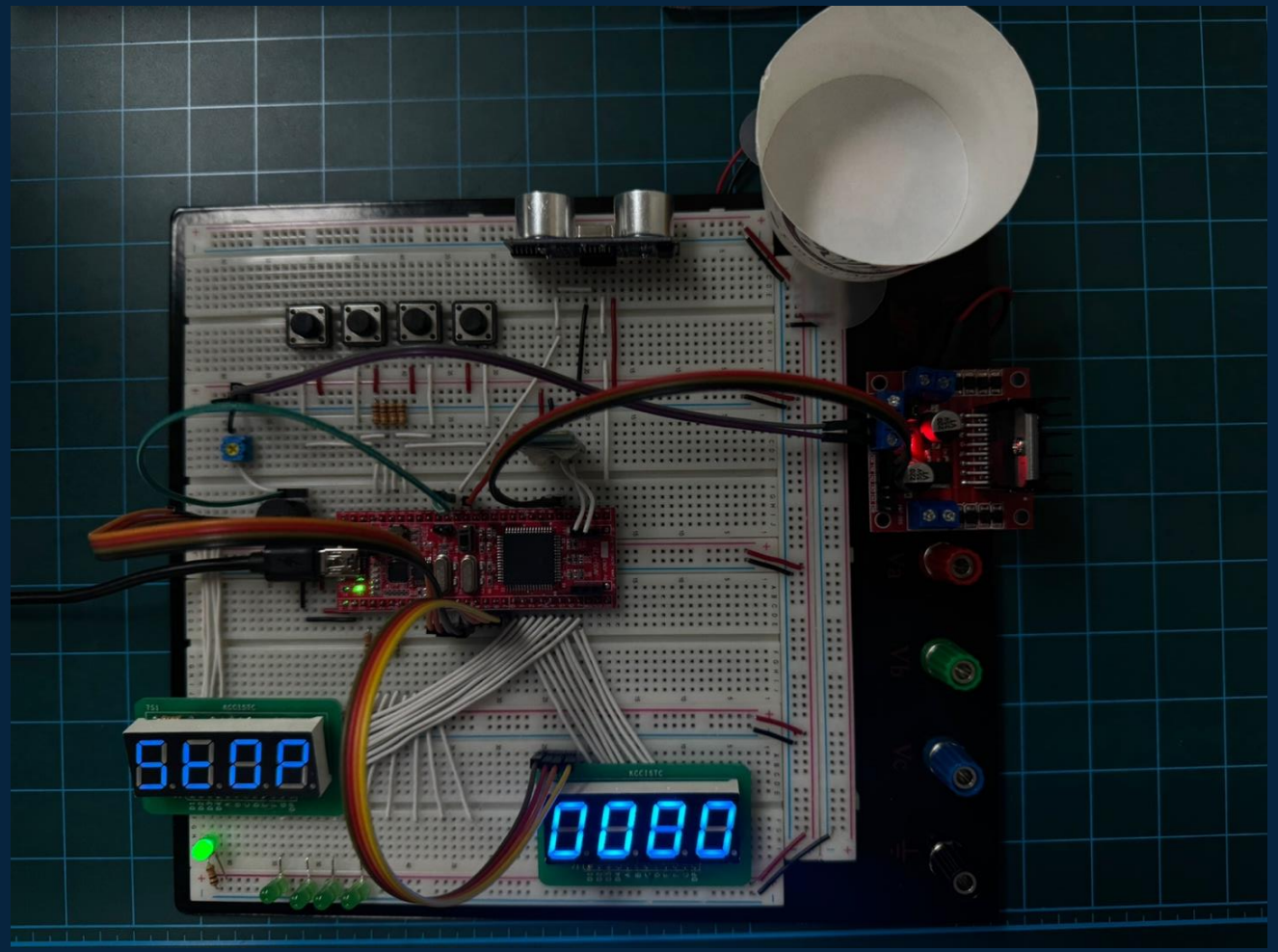
BUZZER

Single Turn Trimmer



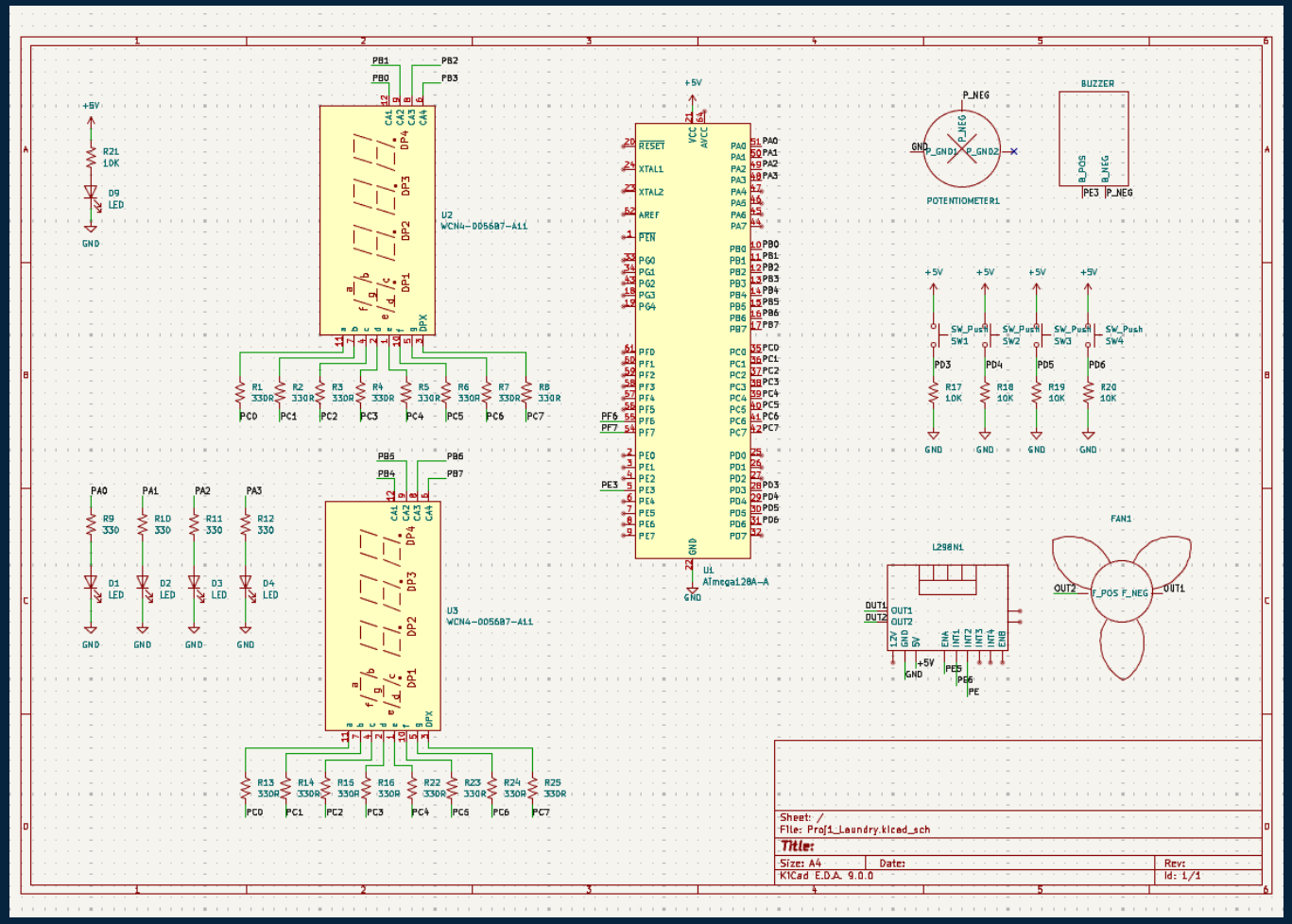
02 Schematic & FSM

Schematic (회로도)



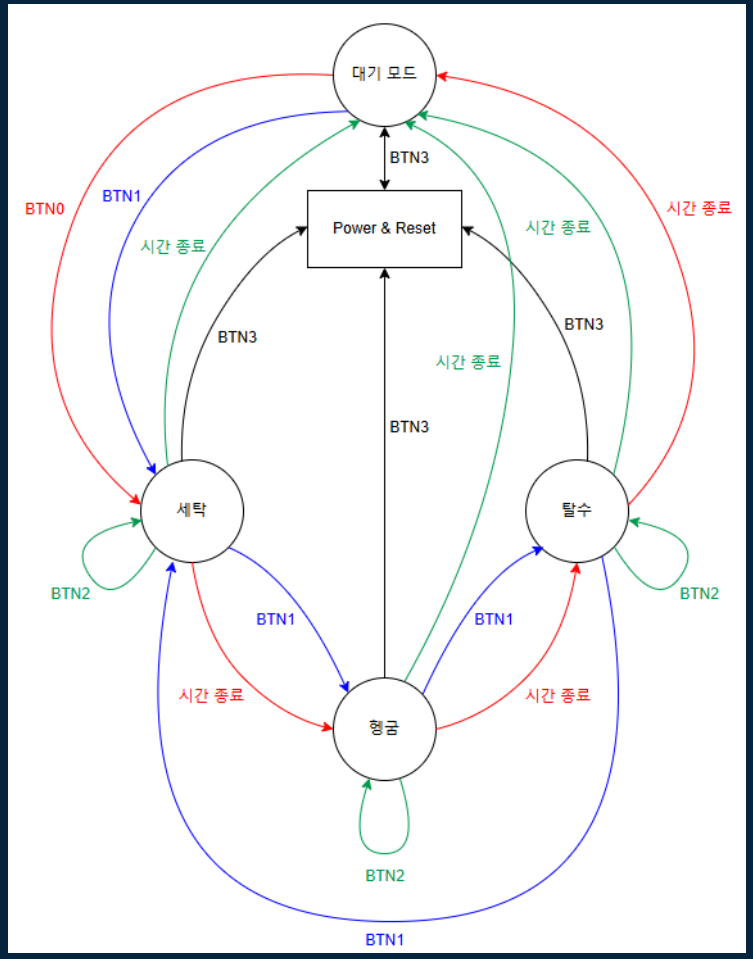
02 Schematic & FSM

Schematic (회로도)



02 Schematic & FSM

FSM (상태천이도)



03 세부 구현 기능

기능 설명

- 버튼3 - 세탁기 전체 전원 버튼 (전원 off 시 모든 기능 정지, 세탁기 동작 중 긴급 정지 가능)
- 버튼0 - 자동 모드 (세탁, 헹굼, 탈수 순서로 자동 동작)
- 버튼1 - 수동 모드 (세탁(1), 헹굼(2), 탈수(3) 선택 가능)
- 버튼2 - 선택한 모드 시작 버튼
- FND1 - 세탁기 남은 동작 시간 표시
- FND2 - 세탁기 동작 멈추면 STOP 표시
  - 세탁기 동작 중 모터 회전 방향에 따른 애니메이션 표시
- LED - 세탁 중에는 LED0 점등
  - 헹굼 중에는 LED1 점등
  - 탈수 중에는 LED2 점등
- BUZZER - 버튼 클릭 시 효과음 출력
  - 세탁, 헹굼, 탈수 동작 종료 시 효과음 출력

04 코드 설명 (pwm.c)

```c
void init_timer3(void)
{
    DDRE |= (1 << 3) | (1 << 5);
    TCCR3A |= 1 << WGM30;
    TCCR3B |= 1 << WGM32;
    TCCR3A |= 1 << COM3C1;
    TCCR3B |= (1 << CS31) | (1 << CS30);
    OCR3C = 0;
}
```

타이머3 초기화 : PWM 신호 생성용

- PE3, PE5 핀을 출력으로 설정
- OC3C 핀에서 비반전 PWM 출력
- 타이머 분주비를 64로 설정하여 속도 조절
- PWM 출력값을 0으로 초기화하여 모터를 처음에는 정지

```c
void init_L298N(void)
{
    DDRF |= (1 << 6) | (1 << 7);
    PORTF &= ~((1 << 6) | (1 << 7));
    PORTF |= (1 << 6);
}
```

L298N 모터 드라이버 초기화
: 회전 방향 제어용 핀 설정

- PORTF의 6번과 7번 핀을 출력으로 설정
- 초기 상태에서 두 핀 모두 LOW로 설정
- 기본 회전 방향을 정회전으로 설정

04 코드 설명 (pwm.c)

```c
void L298N_pwm_fan_fnd_control_main(void)
{
    uint8_t forward = 1;
    unsigned int phase_start_sec = 0;

    init_button();
    init_timer3();
    init_L298N();
    init_buzzer();


    DDRA |= (1 << 0) | (1 << 1) | (1 << 2);
```

- 모터의 회전 방향을 나타내는 변수 (forward) : 1은 정회전, 0은 역회전
- 현재 phase가 시작된 시점을 기록 (초 단위: sec2_count 사용)
- 각종 하드웨어 모듈 초기화
- LED 출력 핀 초기화

```c
    while(1)
    {

        if(get_button(BUTTON3, BUTTON3PIN)) {
            beep(100);
            system_on = !system_on;
            _delay_ms(200);
        }


        if(system_on == 0) {
            OCR3C = 0;
            fnd_stop_display();
            PORTA &= ~((1 << 0) | (1 << 1) | (1 << 2));
            phase = 0;
            selected_phase = 0;
            continue;
        }
```

버튼3: 전체 전원 토글

- 버튼3이 눌리면 부저를 울리고 system_on 상태를 반전

전체 전원이 OFF인 경우

- 모든 모듈을 OFF 처리
- 나머지 코드는 건너뜀

04 코드 설명 (pwm.c)

```
if(phase == 0) {

    OCR3C = 0;
    display_time = selected_phase;
    fnd_stop_display();
    PORTA &= ~((1 << 0) | (1 << 1) | (1 << 2));


    if(get_button(BUTTON0, BUTTON0PIN)) {
        beep(100);
        phase = 1;
        manual_mode = 0;
        phase_start_sec = sec2_count;
        display_time = PRESET_TIME;
        forward = 1;
        PORTF &= ~((1 << 6) | (1 << 7));
        PORTF |= (1 << 6);
        OCR3C = 250;
        PORTA |= (1 << 0);
        _delay_us(100);
    }
```

Phase 0 : 사용자가 phase를 선택할 수 있는 상태

- 모터 OFF, LED 모두 끔
- FND1에 선택된 phase 번호(또는 0)를 표시

버튼0 : 기본 실행 모드 선택
(phase1부터 순차 실행)

- 현재 시간을 phase 시작 시간으로 기록
- 동작 시간을 디스플레이에 표시
- 정회전으로 설정
- PWM 듀티 사이클 설정 (모터 작동)
- LED0 켜기 (Phase1 동작 중 표시)

```
    if(get_button(BUTTON1, BUTTON1PIN)) {
        beep(100);
        selected_phase++;
        if(selected_phase > 3) selected_phase = 1;
        _delay_us(100);
    }
```

버튼1 : 수동 선택 모드에서 실행할 phase를 선택

- 선택 phase 번호 증가
- 3 초과 시 1로 순환

```
    if(get_button(BUTTON2, BUTTON2PIN)) {
        beep(100);
        if(selected_phase != 0) {
            phase = selected_phase;
            manual_mode = 1;
            phase_start_sec = sec2_count;
            display_time = PRESET_TIME;
            if(phase == 1) {
                forward = 1;
                PORTF &= ~((1 << 6) | (1 << 7));
                PORTF |= (1 << 6);
                OCR3C = 250;
                PORTA |= (1 << 0);
                } else if(phase == 2) {
                forward = 1;
                PORTF &= ~((1 << 6) | (1 << 7));
                PORTF |= (1 << 6);
                OCR3C = 250;
                PORTA |= (1 << 1);
                } else if(phase == 3) {
                forward = 0;
                PORTF &= ~((1 << 6) | (1 << 7));
                PORTF |= (1 << 7);
                OCR3C = 250;
                PORTA |= (1 << 2);
            }
            _delay_us(100);
        }
    }
}
```

버튼2 : 수동 선택 모드에서 선택한 phase 실행

- 선택한 phase로 전환
- 수동 모드 활성화
- phase 시작 시간 기록
- 동작 시간 설정
- 선택한 phase에 따라 모터 방향, PWM, LED 제어 설정

04 코드 설명 (pwm.c)

```
    else if(phase == 1) {
        int elapsed = sec2_count - phase_start_sec;
        int remaining = PRESET_TIME - elapsed;
        if(remaining > 0) {
            display_time = remaining;
            } else {
            display_time = 0;
            fnd_stop_display();
            OCR3C = 0;
            PORTA &= ~(1 << 0);
            beep(100);
            _delay_us(100);

            if(manual_mode == 0) {
                phase = 2;
                phase_start_sec = sec2_count;
                display_time = PRESET_TIME;
                forward = 1;
                PORTF &= ~((1 << 6) | (1 << 7));
                PORTF |= (1 << 6);
                OCR3C = 250;
                PORTA |= (1 << 1);
                } else {
                phase = 0;
                selected_phase = 0;
            }
        }
        fnd_big_circle_forward();
    }
```

Phase1 : PRESET_TIME 동안 연속 정회전 수행

- 현재 phase 경과 시간 계산
- 남은 시간 계산
- 남은 시간을 FND에 표시
- 자동 모드인 경우 : 자동으로 Phase2로 전환
- 수동 모드인 경우: phase 종료 후 idle 상태로 복귀
- FND2에 정방향 애니메이션 표시
  (모터의 회전 방향 시각화)

04 코드 설명 (pwm.c)

```
else if(phase == 2) {
    int elapsed = sec2_count - phase_start_sec;
    int remaining = PRESET_TIME - elapsed;
    if(remaining > 0) {
        display_time = remaining;
        int subphase = (elapsed / SUBPHASE_TIME) % 2;
        if(subphase == 0) {
            if(forward != 1) {
                forward = 1;
                PORTF &= ~((1 << 6) | (1 << 7));
                PORTF |= (1 << 6);
            }
            } else {
            if(forward != 0) {
                forward = 0;
                PORTF &= ~((1 << 6) | (1 << 7));
                PORTF |= (1 << 7);
            }
        }
        OCR3C = 250;
        if(forward)
        fnd_big_circle_forward();
        else
        fnd_big_circle_backward();
        } else {
        display_time = 0;
        fnd_stop_display();
        OCR3C = 0;
        PORTA &= ~(1 << 1);
        beep(100);
        _delay_us(100);
        if(manual_mode == 0) {
            phase = 3;
            phase_start_sec = sec2_count;
            display_time = PRESET_TIME;
            forward = 0;
            PORTF &= ~((1 << 6) | (1 << 7));
            PORTF |= (1 << 7);
            OCR3C = 250;
            PORTA |= (1 << 2);
            } else {
            phase = 0;
            selected_phase = 0;
        }
    }
}
```

Phase2: 동작 시간 동안 일정 간격마다 회전 방향 전환

- 현재 phase 경과 시간 계산
- 남은 시간 계산
- 남은 시간을 FND에 표시
- 일정 시간마다 회전 방향을 전환하기 위한 subphase 계산
- 현재 회전 방향에 따라 FND2에 정/역 애니메이션 표시
- 자동 모드인 경우 : 자동으로 Phase3로 전환
- 수동 모드인 경우 : idle 상태로 복귀

04 코드 설명 (pwm.c)

```
        else if(phase == 3) {
            int elapsed = sec2_count - phase_start_sec;
            int remaining = PRESET_TIME - elapsed;
            if(remaining > 0) {
                display_time = remaining;
                } else {
                display_time = 0;
                fnd_stop_display();
                OCR3C = 0;
                PORTA &= ~(1 << 2);
                beep(100);
                _delay_us(100);
                phase = 0;
                selected_phase = 0;
            }
            fnd_big_circle_backward();
        }

        fnd_display();
        _delay_us(500);
    }
}
```

Phase3 : 동작 시간 동안 연속 역회전 수행

- 현재 phase 경과 시간 계산
- 남은 시간 계산
- 남은 시간을 FND에 표시
- Phase3 종료 후, 자동 모드와 수동 모드 모두 idle 상태로 복귀
- FND2에 역방향 애니메이션 표시
- FND1 업데이트 : 동작 중이면 남은 시간 카운트
  idle 상태면 선택한 phase 번호 표시

04 코드 설명 (fnd.c)

```c
void fnd_display(void)
{
    uint8_t fnd_font[] = {0xc0, 0xf9, 0xA4, 0xb0, 0x99, 0x92, 0x82, 0xd8, 0x80, 0x90, 0x7f};

    static int digit_select = 0;

    int value = display_time;

    switch(digit_select)
    {
        case 0:
        FND_DIGIT_PORT = 0x80;
        FND_DATA_PORT = fnd_font[value % 10];
        break;
        case 1:
        FND_DIGIT_PORT = 0x40;
        FND_DATA_PORT = fnd_font[(value / 10) % 10];
        break;
        case 2:
        FND_DIGIT_PORT = 0x20;
        FND_DATA_PORT = fnd_font[0];
        break;
        case 3:
        FND_DIGIT_PORT = 0x10;
        FND_DATA_PORT = fnd_font[0];
        break;
    }
    digit_select++;
    digit_select %= 4;
}
```

- 전역 변수 display_time에 저장된 값을 FND에 숫자 형태로 출력
- 4자리 FND에 대해 각각 일의 자리, 십의 자리, 백의 자리, 천의 자리 순으로 갱신
- digit_select 변수를 통해 순환적으로 자릿수 선택

04 코드 설명 (fnd.c)

```c
void fnd_big_circle_forward(void)
{
    uint8_t fnd_font0[] = {0xf7, 0xf7, 0xf7, 0xf7, 0xf7, 0xf7, 0xf7, 0xf7, 0xf7, 0xf6, 0xf4, 0xf0};
    uint8_t fnd_font1[] = {0xff, 0xf7, 0xf7, 0xf7, 0xf7, 0xf7, 0xf7, 0xf7, 0xf6, 0xf6, 0xf6, 0xf6};
    uint8_t fnd_font2[] = {0xff, 0xff, 0xf7, 0xf7, 0xf7, 0xf7, 0xf7, 0xf6, 0xf6, 0xf6, 0xf6, 0xf6};
    uint8_t fnd_font3[] = {0xff, 0xff, 0xff, 0xf7, 0xe7, 0xc7, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6, 0xc6};

    static int digit_select = 0;

    int frame = (msec2_count / 50) % 12;

    switch(digit_select)
    {
        case 0:
        FND_DIGIT_PORT = 0x08;
        FND_DATA_PORT = fnd_font0[frame];
        break;
        case 1:
        FND_DIGIT_PORT = 0x04;
        FND_DATA_PORT = fnd_font1[frame];
        break;
        case 2:
        FND_DIGIT_PORT = 0x02;
        FND_DATA_PORT = fnd_font2[frame];
        break;
        case 3:
        FND_DIGIT_PORT = 0x01;
        FND_DATA_PORT = fnd_font3[frame];
        break;
    }
    digit_select++;
    digit_select %= 4;
}
```

- 정방향 애니메이션을 위해 4개의 배열에 저장된 프레임 값을 사용
- msec2_count를 기준으로 애니메이션 프레임을 결정, 자릿수별로 다른 배열을 사용

04 코드 설명 (fnd.c)

```c
void fnd_big_circle_backward(void)
{
    uint8_t fnd_font0[] = {0xfb, 0xf9, 0xf8, 0xf8, 0xf8, 0xf8, 0xf8, 0xf8, 0xf8, 0xf8, 0xf8, 0xf0};
    uint8_t fnd_font1[] = {0xff, 0xff, 0xff, 0xfe, 0xfe, 0xfe, 0xfe, 0xfe, 0xfe, 0xfe, 0xf6, 0xf6};
    uint8_t fnd_font2[] = {0xff, 0xff, 0xff, 0xff, 0xfe, 0xfe, 0xfe, 0xfe, 0xfe, 0xf6, 0xf6, 0xf6};
    uint8_t fnd_font3[] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xde, 0xce, 0xc6, 0xc6, 0xc6, 0xc6};

    static int digit_select = 0;

    int frame = (msec2_count / 50) % 12;

    switch(digit_select)
    {
        case 0:
        FND_DIGIT_PORT = 0x08;
        FND_DATA_PORT = fnd_font0[frame];
        break;
        case 1:
        FND_DIGIT_PORT = 0x04;
        FND_DATA_PORT = fnd_font1[frame];
        break;
        case 2:
        FND_DIGIT_PORT = 0x02;
        FND_DATA_PORT = fnd_font2[frame];
        break;
        case 3:
        FND_DIGIT_PORT = 0x01;
        FND_DATA_PORT = fnd_font3[frame];
        break;
    }
    digit_select++;
    digit_select %= 4;S
}
```

- fnd_big_circle_forward()와 유사하게 역방향 애니메이션을 표현
- 다른 프레임 배열을 사용해 반대 방향의 효과 나타냄

04 코드 설명 (fnd.c)

```c
void fnd_stop_display(void)
{
    uint8_t fnd_font0 = 0x8c;
    uint8_t fnd_font1 = 0xc0;
    uint8_t fnd_font2 = 0x87;
    uint8_t fnd_font3 = 0x92;

    static int digit_select = 0;

    switch(digit_select)
    {
        case 0:
        FND_DIGIT_PORT = 0x08;
        FND_DATA_PORT = fnd_font0;
        break;
        case 1:
        FND_DIGIT_PORT = 0x04;
        FND_DATA_PORT = fnd_font1;
        break;
        case 2:
        FND_DIGIT_PORT = 0x02;
        FND_DATA_PORT = fnd_font2;
        break;
        case 3:
        FND_DIGIT_PORT = 0x01;
        FND_DATA_PORT = fnd_font3;
        break;
    }
    digit_select++;
    digit_select %= 4;

    msec2_count = temp;
}
```

- “STOP” 메시지 또는 특정 패턴을 FND에 출력
- 각 자릿수에 미리 정의된 7 세그먼트 제어 값들을 출력하여 “STOP”을 표시
- msec2_count를 temp 값으로 재설정하여 필요에 따라 애니메이션 카운터 초기화

04 코드 설명 (speaker.c)

```c
void Music_Player(int *tone, int *Beats)
{
    while(*tone != '/0')
    {
        OCR3A = *tone;
        delay_ms(*Beats);
        tone++;
        Beats++;
        OCR3A = 0;
        _delay_ms(10);
    }
    return;
}
```

```c
void init_speaker(void)
{
    DDRE |= 0x08;
    TCCR3A = (1<<COM3A0);
    TCCR3B = (1<<WGM32) | (1<<CS31);
    TCCR3C = 0;
    OCR3A = 0;

    return;
}
```

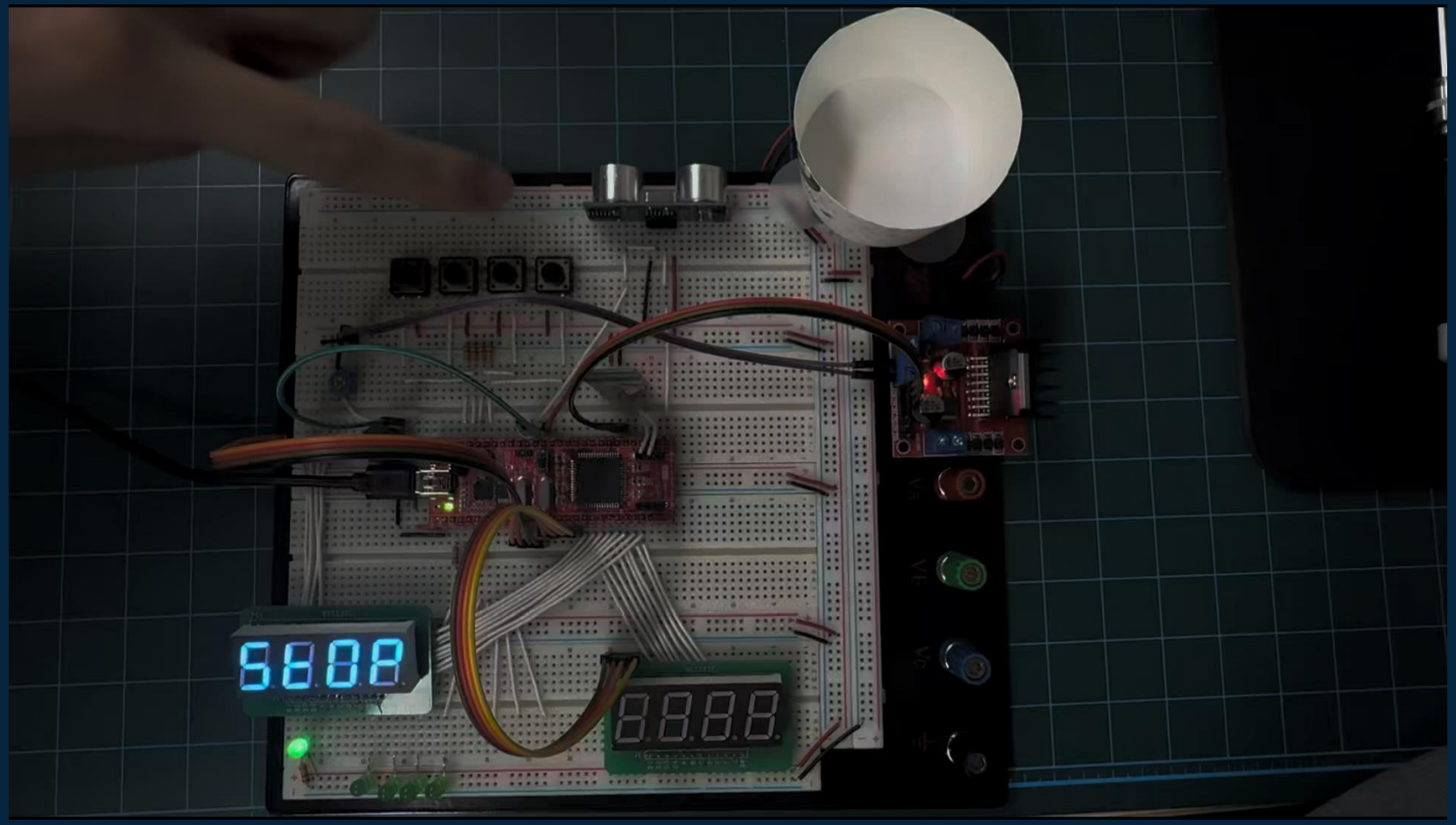
```c
int laundry[] = {DO_02_H, FA_02_H, FA_02, RE_02_H, DO_02_H, LA_01_H,
                 TI_01, DO_02_H, RE_02_H, SO_01, LA_01_H, TI_01, LA_01_H, DO_02_H,
                 DO_02_H, FA_02_H, FA_02, RE_02_H, DO_02_H, FA_02_H,
                 FA_02_H, SO_02_H, FA_01_H, FA_02, RE_02_H, FA_02, FA_02_H, '\0'};

const int laundry_beats[] = {BEAT_1_4, BEAT_1_8, BEAT_1_8, BEAT_1_8, BEAT_1_4, BEAT_1_4,
                             BEAT_1_8, BEAT_1_8, BEAT_1_8, BEAT_1_8, BEAT_1_8, BEAT_1_8, BEAT_1_4, BEAT_1_4,
                             BEAT_1_4, BEAT_1_8, BEAT_1_8, BEAT_1_8, BEAT_1_4, BEAT_1_4,
                             BEAT_1_8, BEAT_1_8, BEAT_1_8, BEAT_1_8, BEAT_1_8, BEAT_1_8, BEAT_1_2};
```

- Music_Player(laundry, laundry_beats);
- 세탁기 종료음 멜로디 출력 코드
- 블로킹 방식으로 인해 FND와의 동작에 문제 발생
- 타이머 인터럽트 기반으로 구현하면 해결 가능할 것으로 보임

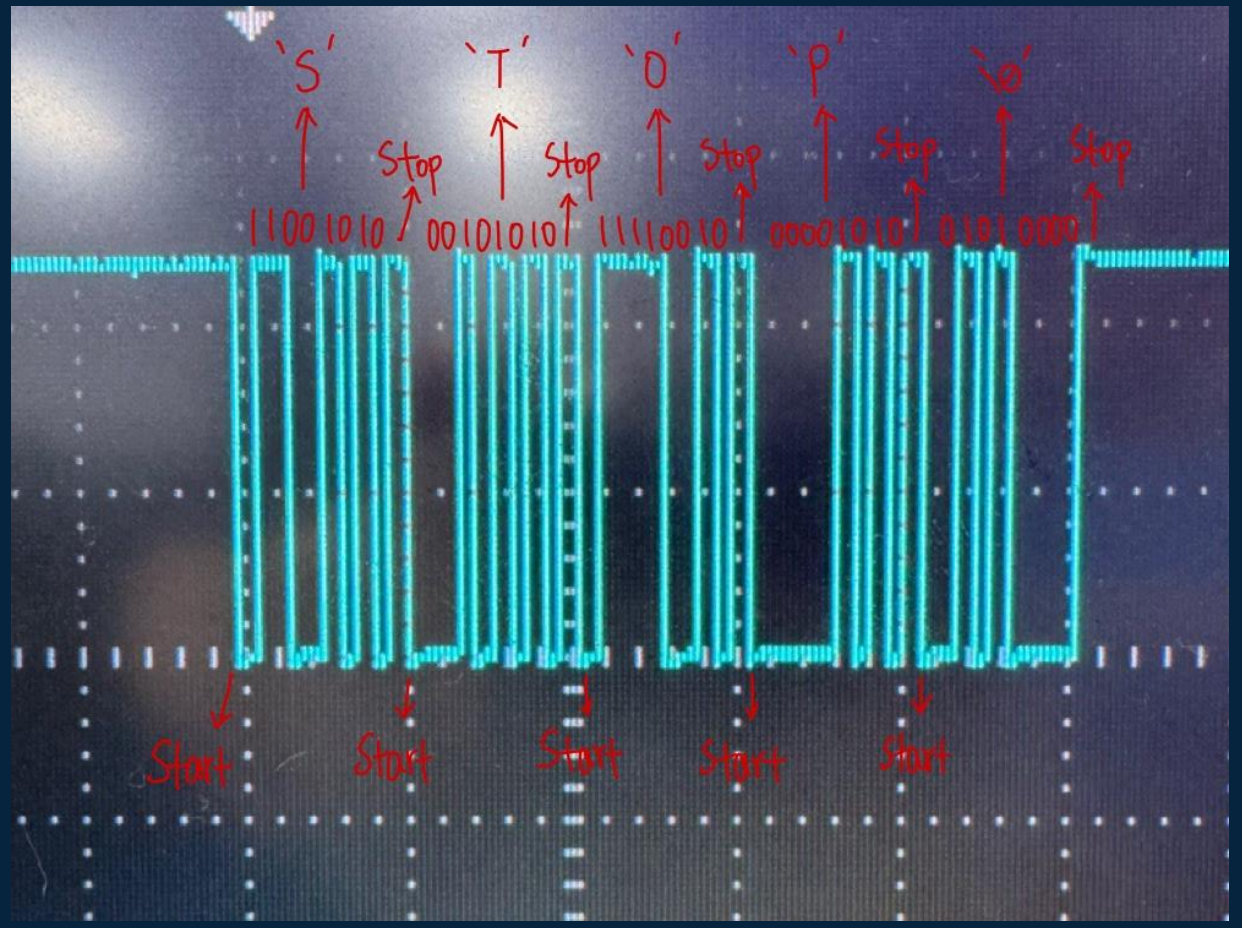
05 결과 및 개선점

동작 영상 (https://youtube.com/shorts/i7M0yNZkVg0)



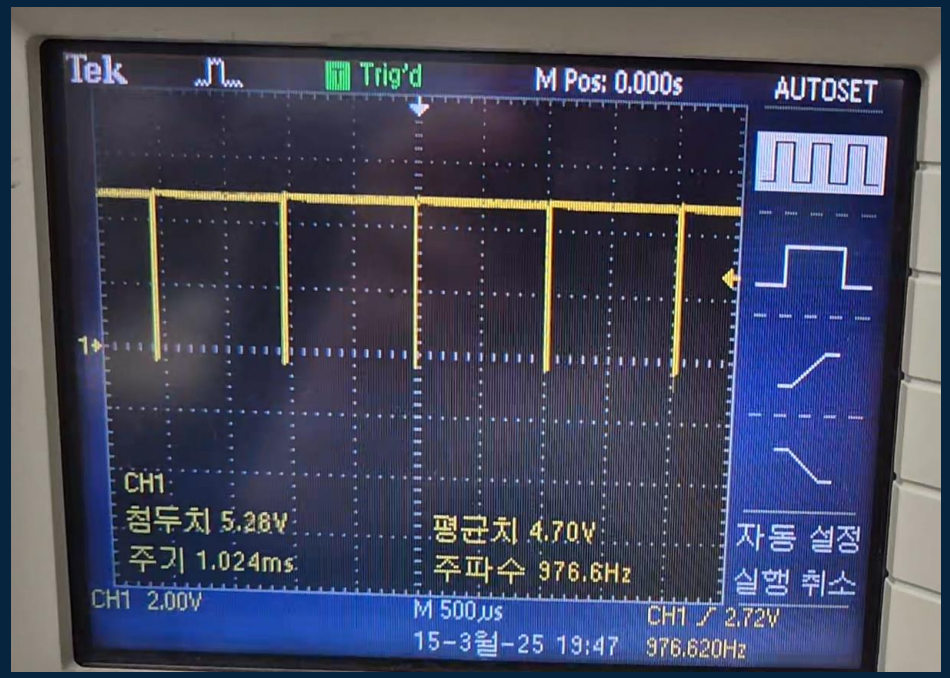
05 결과 및 개선점

UART 오실로스코프 검증 (세탁기 동작 정지 시 'STOP' 출력)



05 결과 및 개선점

PWM 파형 검증 (모터 ON / OFF)

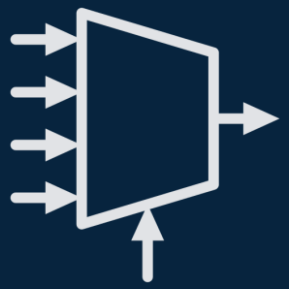


모터 ON

모터 OFF

05 결과 및 개선점

문제점



delay 같은 블로킹 방식에 의한 문제

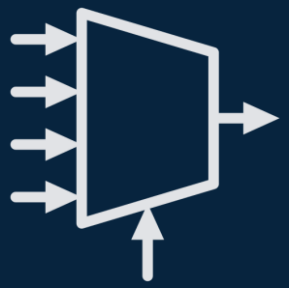
- FND 다중화 문제 및 낮은 밝기
  - 블로킹 delay를 사용하면 각 자릿수가 실제로 켜지는 시간이 짧아져
    평균적으로 FND에 공급되는 전류가 줄어들고 그 결과 밝기가 낮게 출력

- BUZZER와 FND의 동시 동작 문제
  - BUZZER 출력과 FND 갱신 타이밍과 충돌

- BUTTON 처리 로직 내 추가 작업
  - BUTTON을 누를 때 BUZZER를 울리거나 다른 상태 전환 작업을 처리하는 동안에도
    delay 함수가 호출되어 FND 업데이트가 지연되어 깜빡임 현상 나타남

05 결과 및 개선점

개선점



delay 같은 블로킹 방식에 의한 문제

- 타이머 인터럽트 기반 구현
  - FND 다중화와 BUZZER 타이밍 제어를 타이머 인터럽트를 통해 처리하면 블로킹 delay 없이도 각 기능을 정밀하게 제어 가능

모터 속도 제어 기능 추가

모터 동작 시간 제어 기능 추가

블루투스, 초음파 센서 같은 모듈 추가

Thank You

(Telechips) AI 시스템반도체 SW 개발자

박성호 이경주