



페달 오조작 방지 보조 시스템

실시간 운영체제를 활용한 차량 제어 시스템

3조 텔레강스
송지은, 이정현, 황서현, 최우영, 김종우, 임수빈

Section 1

프로젝트 개요

1. 프로젝트 개요

기획 배경

24년 7월 1일 시청역 차량 돌진 사고
국과수 감정 결과...

“페달 오조작”

출처: 헤럴드 경제



페달 오조작으로 인한 사고 **지속적으로 발생 중**

최신기사

급발진 신고 321건 분석해보니, 전부 ‘페달 오조작’ 사고였다

2024.09.11 11:40

급가속 사고, 매월 160건 발생...페달 오조작 대책 마련 시급

보험저널 최지호 기자 | 입력 2024.10.16 15:05 | 댓글 0



출처: 보험저널

국과수 “급발진 88%는 페달 오조작, 나머지 12%는...”

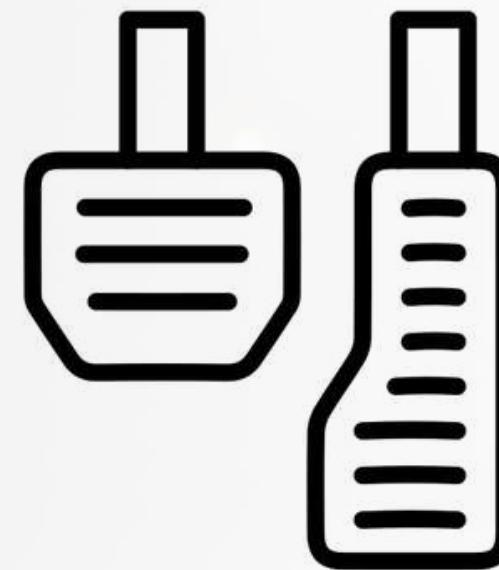
입력 2024.09.10 (14:29)

출처: KBS뉴스

1. 프로젝트 개요

기획 배경

이를 해결하고자 페달 오조작 방지 보조 시스템을 만들기로 기획



1. 프로젝트 개요

개발 일정



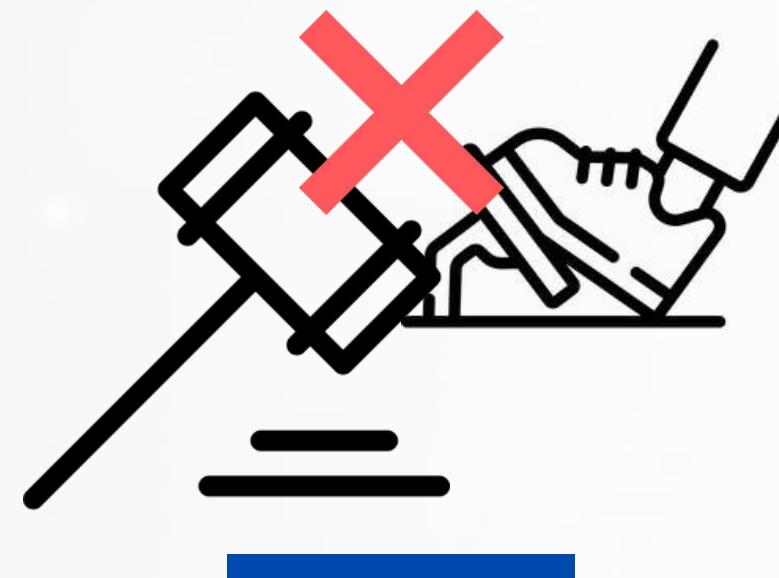
Section 2

요구사항 분석 및 정의

2. 요구사항 분석 및 정의

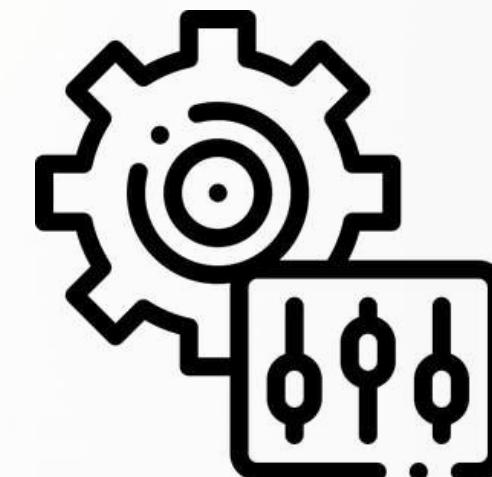
주요 기능 요약

오조작 판단



정의된 오조작 상황에 따라
오조작 판단

모터 제어



조이스틱을 통해 가속 및 브레이크
페달의 압력값을 측정하여
오조작 상황 판단 후 모터 제어

2. 요구사항 분석 및 정의

오조작 상황 정의

1) 오토 홀드 비활성화 시,

- ① 정차 후 출발 시 7CM 이내 장애물이 있는 상황에서 운전자가 0.25초 이내 가속 페달을 100% 밟는 경우

2) 오토 홀드 활성화 시,

- ① 7CM 이내 장애물이 있는 상황에서 가속 페달을 밟는 경우
- ② 전방에 적색 신호등이 점등 되어 있는 상황에서 가속 페달을 밟는 경우

오토 홀드: 차량이 완전히 정차한 후, 브레이크를 밟고 있지 않아도 차량의 정지 상태를 유지 시켜주는 기능

오토 홀드 활성화 상태: 오토 홀드 모드가 켜져있고, 브레이크를 밟아 차량이 완전히 정차한 상태

2. 요구사항 분석 및 정의 오조작 상황 정의



2. 요구사항 분석 및 정의 오조작 상황 정의

사회

행인 4명 들이받은 전기차 돌진...'페달 오조작' 무게

© 1분 이내 입력 2024.11.22 19:58

운전자는 "주행모드(D) 상태에서 **오토홀드**(정차 시 제동 기능)를 누른 뒤 차를 세워놓고 운전석에서 신발을 신는데 갑자기 차가 움직였다"는 취지로 경찰에 진술했습니다.

하지만 경찰이 CCTV 등을 분석한 결과 사고 차량이 돌진할 때 브레이크 등은 들어오지 않은 걸로 조사됐습니다.

이 때문에 운전자가 당황해서 **브레이크 대신 가속 페달을 잘못 밟았을 가능성에 무게를 두고 있습니다.**

사고 원인: 오토홀드 활성화 상태에서 신발을 신으려다 가속 페달을 건드림

기존 캐스퍼 일렉트릭 PMSA



**오토홀드 상태에서의
오조작 방지 보조**

2. 요구사항 분석 및 정의

요구사항 정의서

요구 사항 정의서

1. 기능적 요구 사항					
기능 분류	기능 그룹	세부 기능	ID	설명	우선 순위
판단	오조작 감지	급가속 검출(전후방)	FR1.1.1	장애물이 정의한 사고위험거리 내에 있으면서, 가속페달을 0.25초이내 최대로 밟는 경우	상
		오토팔드(신호등O)	FR1.1.2	오토팔드 상태에서 앞에 차가 없으면서(지정 거리 이내 차X), 빨간불을 대기하는 상황에서 가속페달을 밟았을 경우	상
		오토팔드(신호등X)	FR1.1.3	오토팔드 상태에서 앞에 차가 있으면서(지정 거리 이내 차O), 가속페달을 밟았을 경우	상
	신호등 감지	적색등 감지/판단	FR1.2.1	카메라로 적색등 상태의 신호등 인식	상
데이터 출력	상태 시/청각 알림	주행 모드 출력	FR2.1.1	P/R/D -> 빨/노/초 LED로 색상표시	하
		오토팔드 사용 여부 출력	FR2.1.2	LED로 오토팔드 on/off 표시	하
		오조작 알림 출력	FR2.1.3	오조작 감지시 부저로 경고음 알림	상
		가속 페달이 눌린 정도 출력	FR2.1.4	조이 스틱(가속 페달)에 가해진 압력을 퍼센트로 표시	하
I/O 제어	주행 모드 설정	오토팔드 ON	FR3.1.1	오토팔드 스위치 on일 때, 브레이크 밟는 동안 속도가 0이 되면 브레이크 밟지 않아도 모터 속도 증가 X	중
		오토팔드 OFF	FR3.1.2	오토팔드 스위치 off일 때, 가 안밟아도 차량 속도 증가 -> D모드일 때 모터 기본 RPM 정해야 함	중
		P(Parking)	FR3.1.3	주행모드 스위치 P일 때, 모터 전압off, 차량 정지	중
		R(Reverse)	FR3.1.4	주행모드 스위치 R일 때, 모터 +,- 극성 전환, 차량 후방 이동	중
		D(Drive)	FR3.1.5	주행모드 스위치 D일 때, 엑셀의 입력 비례 PWM 방식 모터 RPM 증가, 차량 전방 이동	중
	모터 제어	가속	FR3.2.1	조이 스틱(가속 페달)에 압력이 가해질 경우, 정도에 따라 모터 속도 증가	중
		브레이크	FR3.2.2	조이 스틱(브레이크 페달)에 압력이 가해질 경우, 정도에 따라 모터 속도 감소	중
측정	실시간 추적 측정	장애물 거리 측정	FR4.1.1	초음파를 통해 전/후방 인근에 장애물이 있는지 판단, 거리 측정	상
		가속 페달 압력 측정	FR4.1.2	조이 스틱(가속 페달)의 압력을 측정하여 가속 정도를 계산, 오조작 방지 기능과 연동	상

Section 3

컨셉 아키텍처

3. 컨셉 아키텍처

아키텍처 선정

Da 디지털데일리 + 구독

이장규 텔레칩스 대표 "車, 움직이는 고성능 컴퓨터...칩 다변화 대응→중앙제어 필수"

입력 2024.11.18. 오전 10:18 기사원문

고성현 기자 TALK

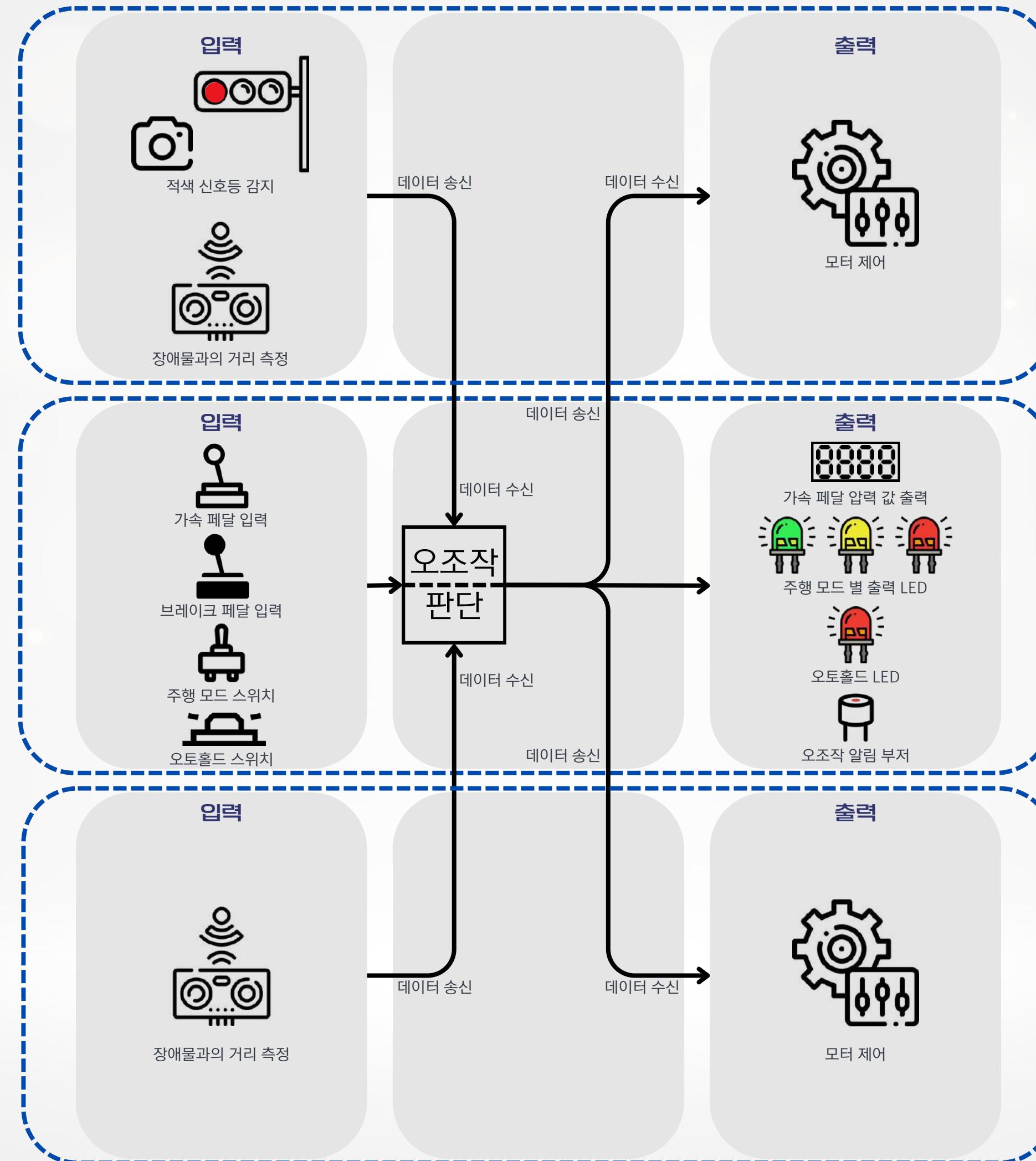
SDV를 구현하기 위해서는 제각기 움직이는 ECU 등을 통합해 하나의 중앙집권화 SoC로 제어하는 '조널 아키텍처(Zonal Architecture)' 플랫폼으로의 전환이 필요하다. 중앙 SoC를 통해 차량 각 기능 요소 업데이트가 적용되려면 이들 부품 간 상호연결이 필수불가결해서다. 스마트폰 OS 업데이트를 통해 카메라, 음성 녹음 등 앱을 강화하는 것과 같은 원리다.

Flat
Architecture

Domain
Architecture

Zonal
Architecture

3. 컨셉 아키텍처 컨셉 개요



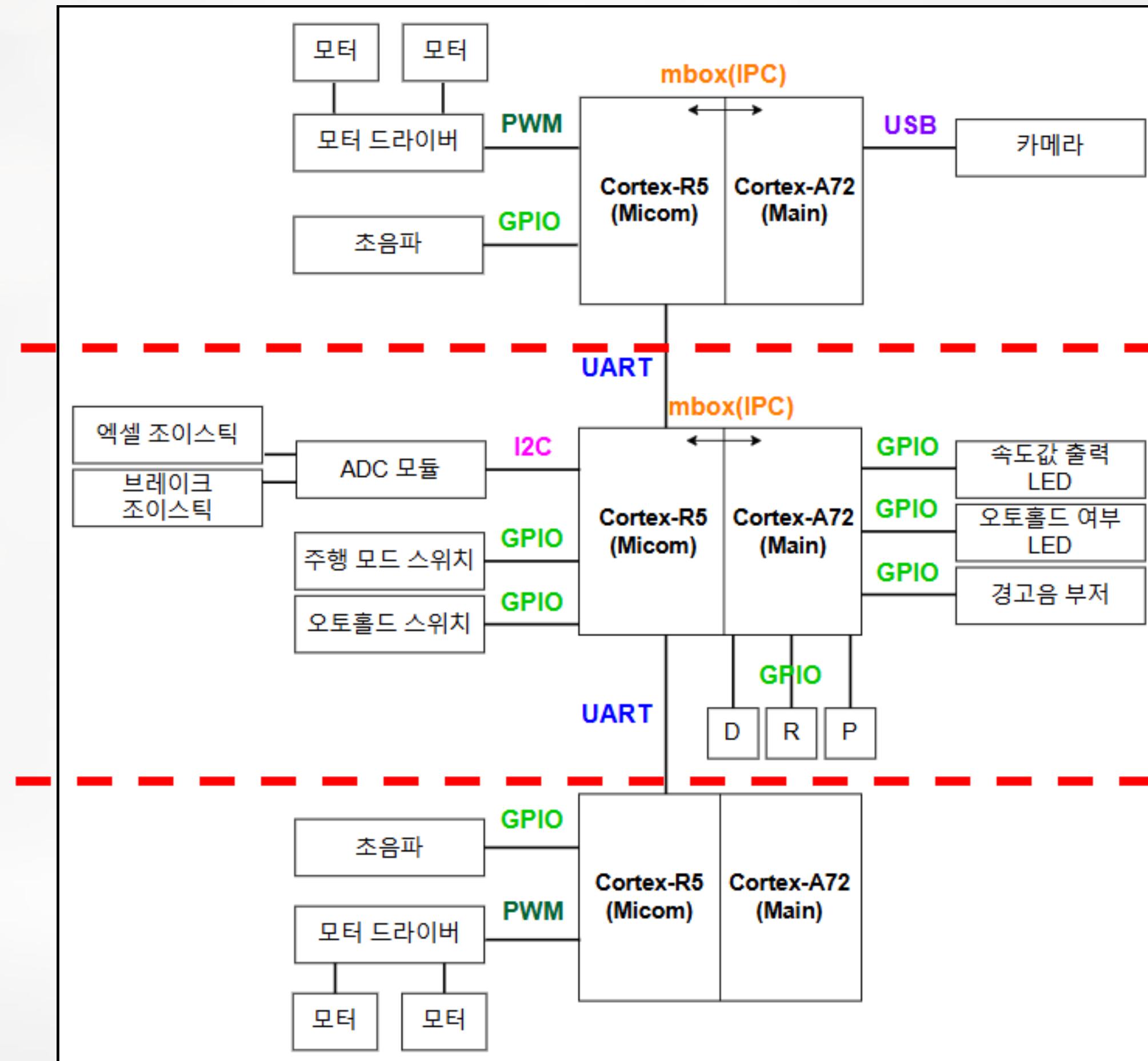
전방 보드

중앙 보드

후방 보드

3. 컨셉 아키텍처

최종 아키텍처



전방 보드

중앙 보드

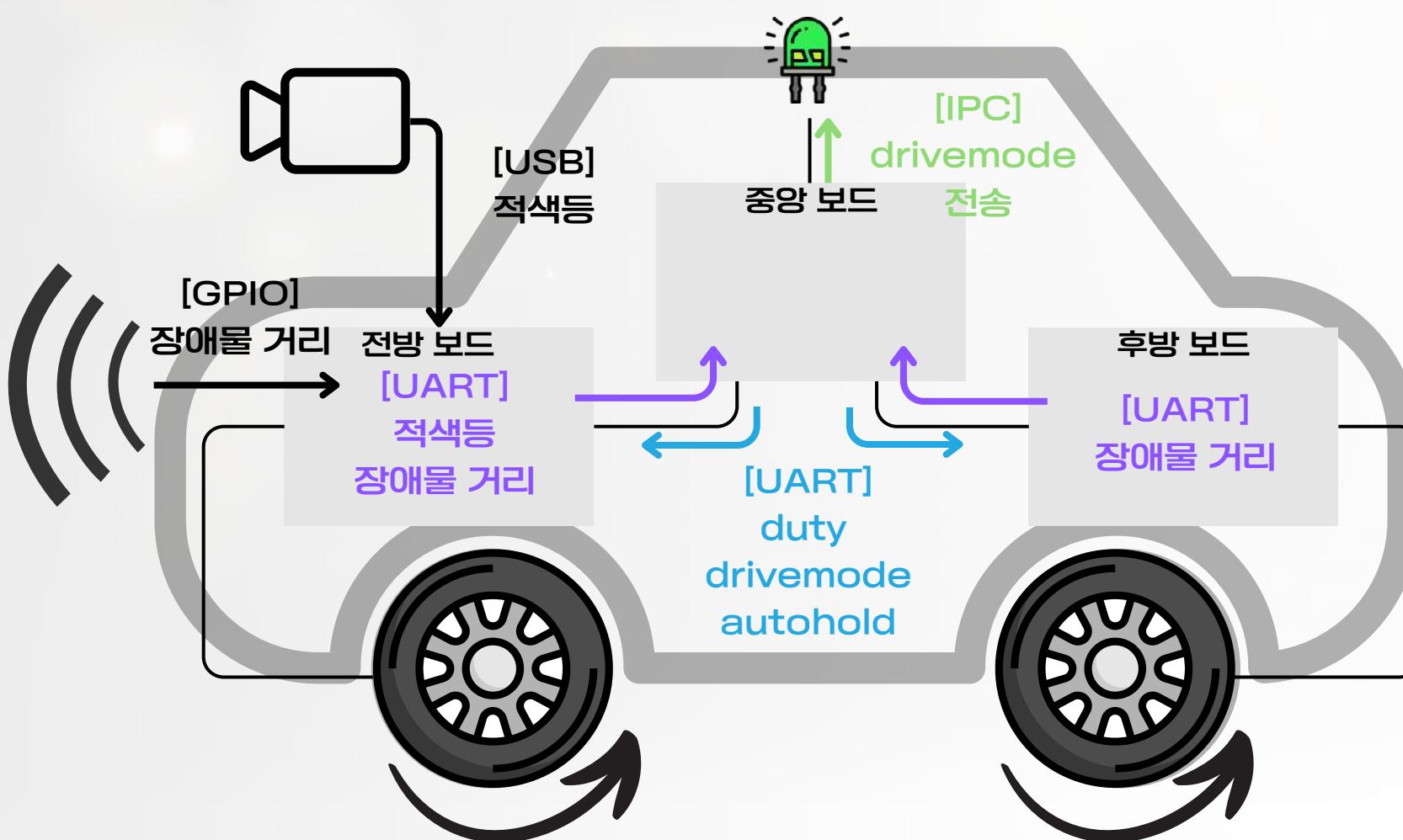
후방 보드

Section 4

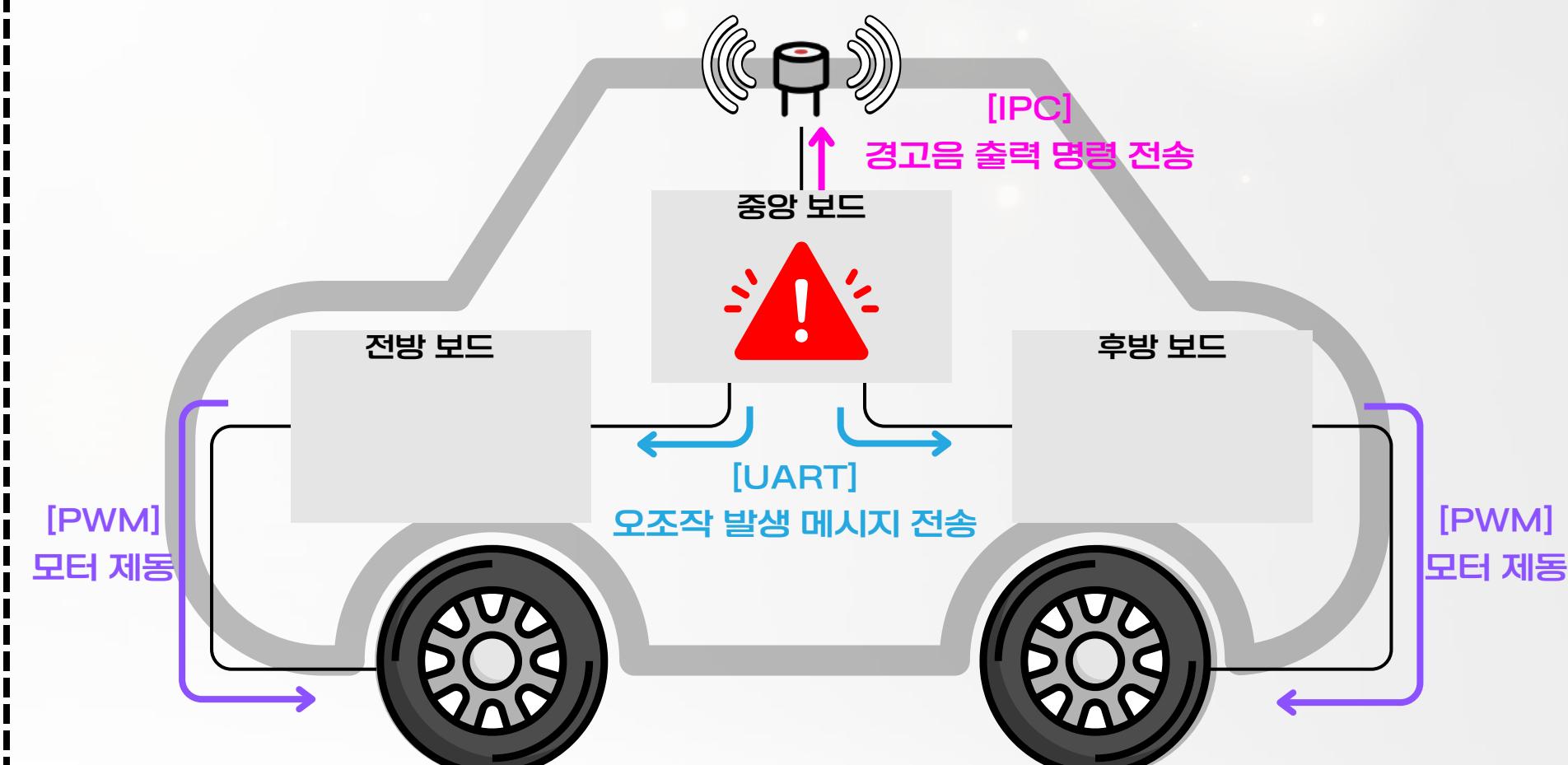
개발 및 설계

4. 개발 및 설계 주요 로직

평상시 (D Mode)



오조작 발생 시



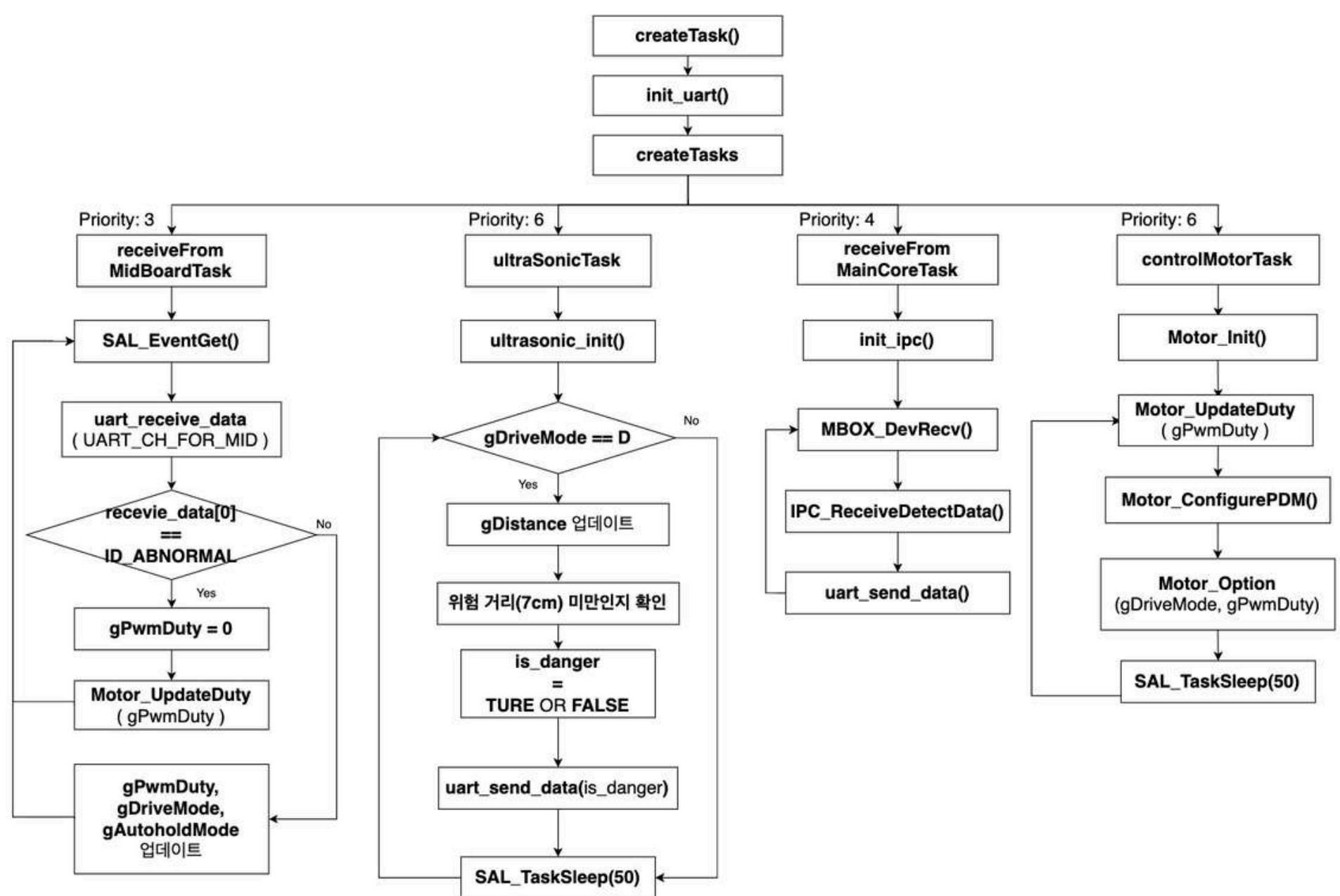
4. 개발 및 설계

보드 별 주요 로직

전방 보드

MICOM

MAIN

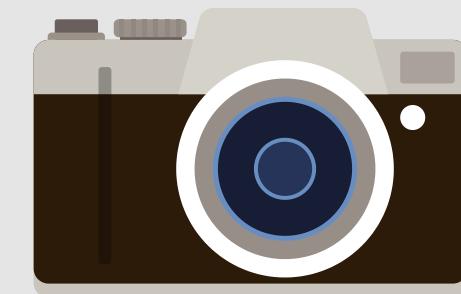


중앙 보드로부터
데이터 수신

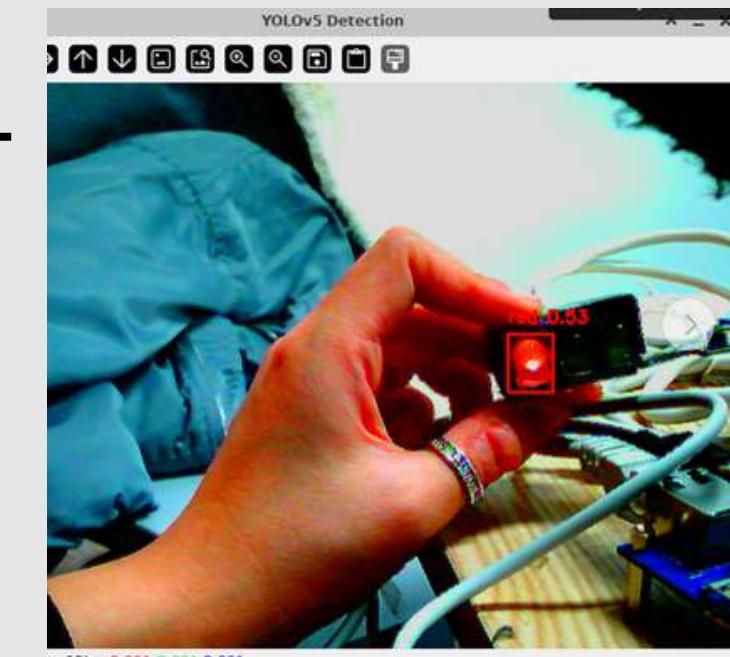
전방의 장애물 거리
측정 및 전송

적색 신호등
감지 결과 수신

모터 제어



IPC

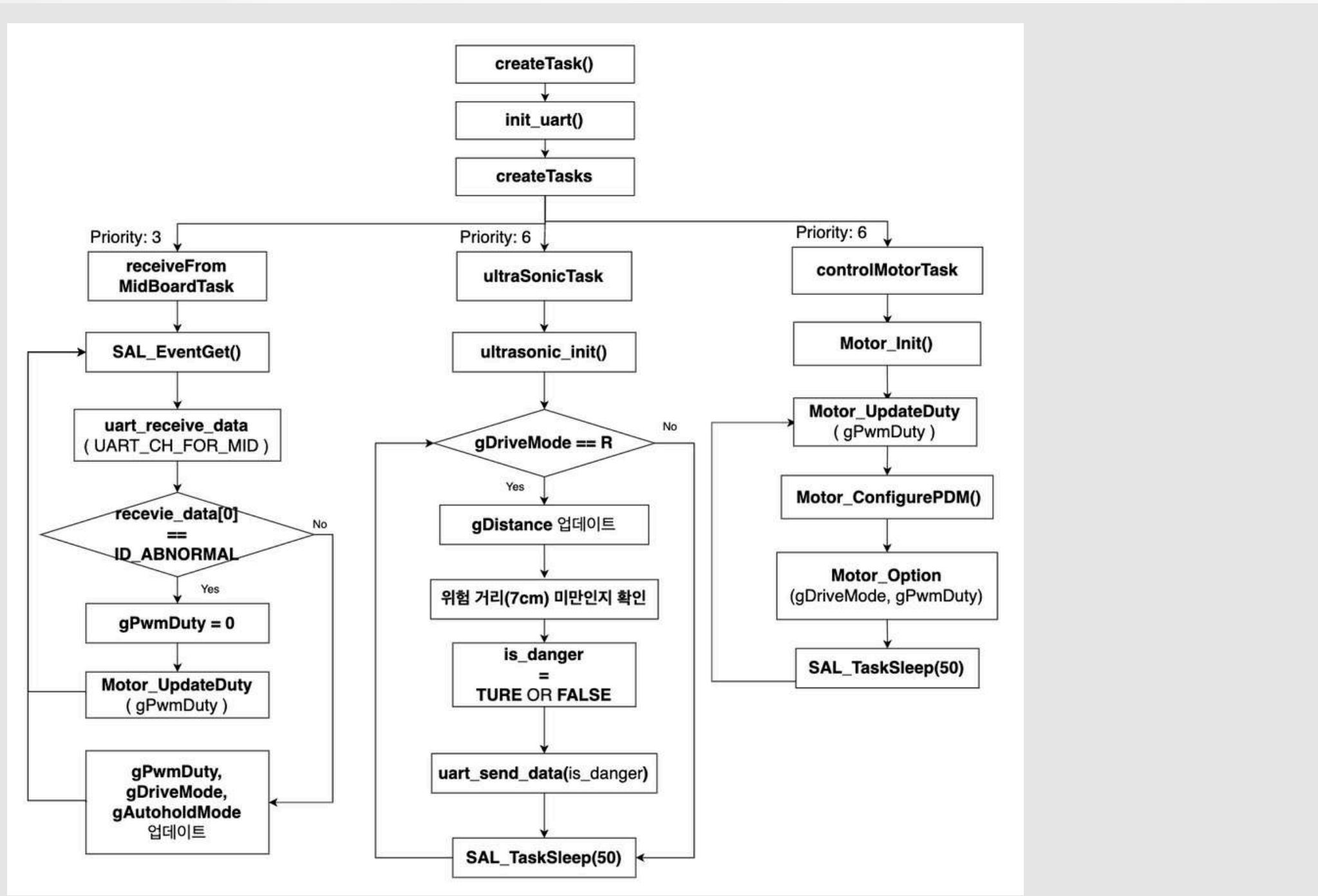


4. 개발 및 설계

보드 별 주요 로직

후방 보드

MICOM



중앙 보드로부터
데이터 수신

후방의 장애물 거리
측정 및 전송

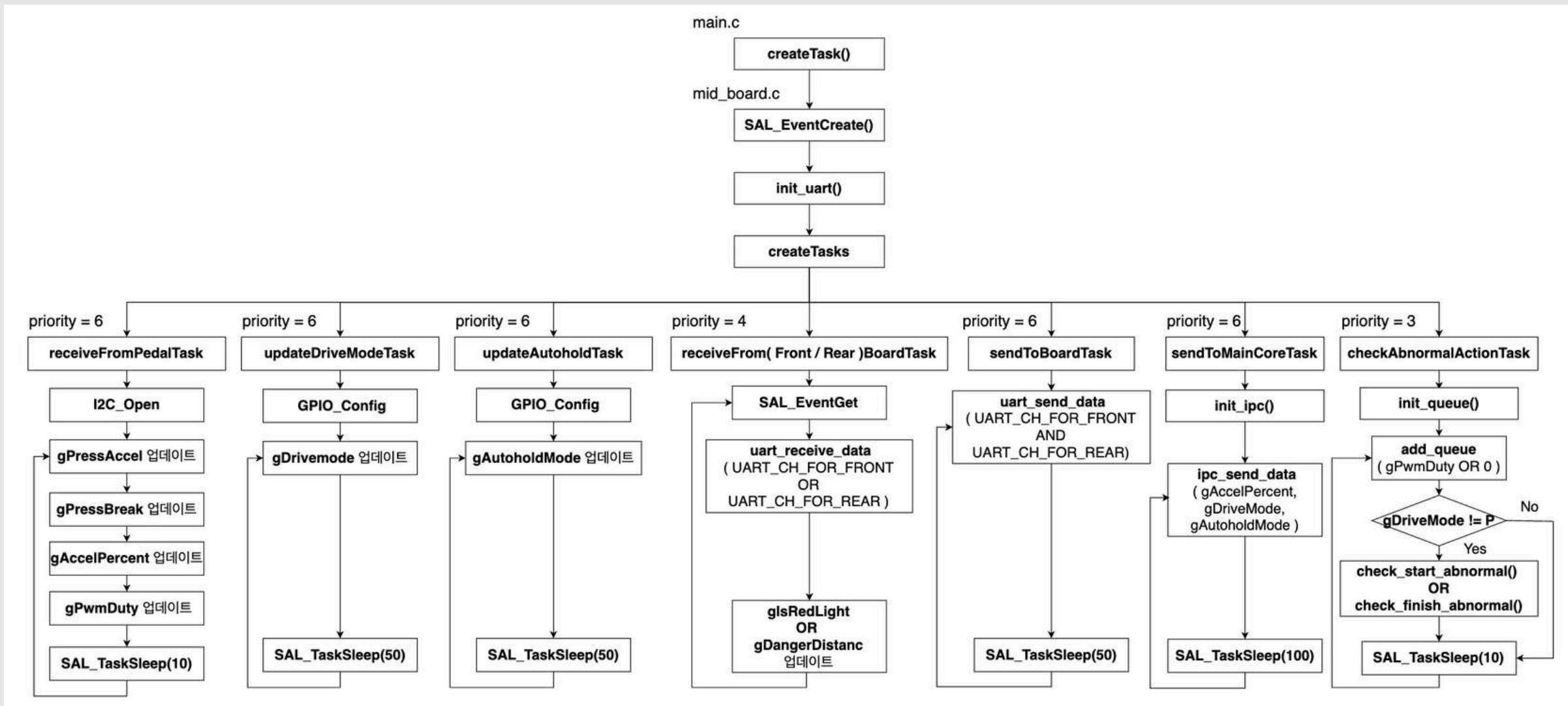
모터 제어

4. 개발 및 설계

보드 별 주요 로직

중앙 보드

MICOM



페달 입력값
업데이트

드라이브 모드
업데이트

오토홀드 모드
업데이트

전후방
보드로부터
데이터 수신

전후방 보드로
데이터 전송

메인 코어로
데이터 전송
오조작 판단

MAIN



가속 페달 압력값

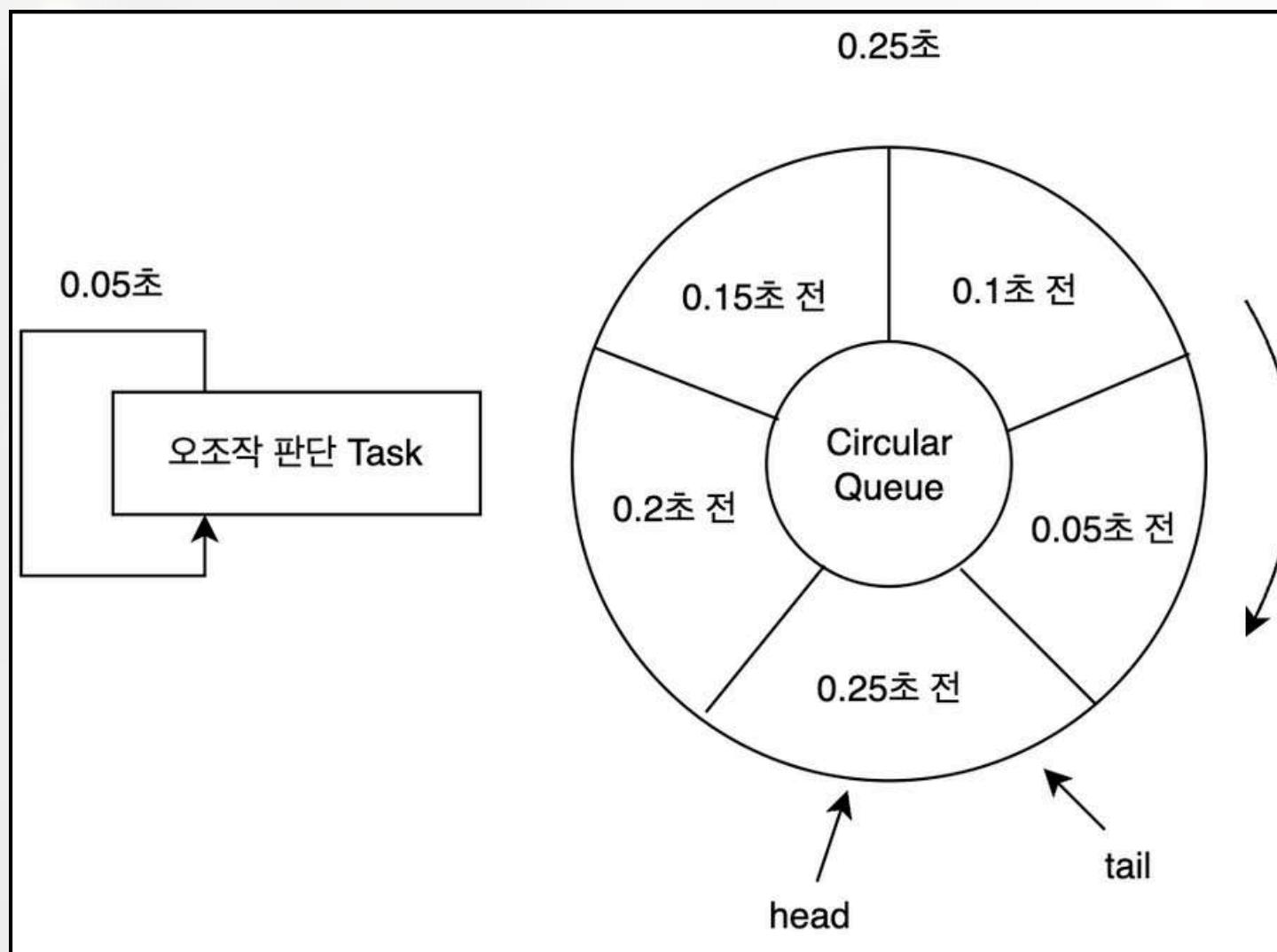


4. 개발 및 설계 주요 로직

오조작 판단 알고리즘

오조작 판단을 위해 일정 시간 동안 차량의 속도 저장 필수

- 원형 큐(Circular Queue)를 이용하여 구현
- 태스크 실행 주기에 맞추어 버퍼의 크기를 정의



```
#define ABNORMAL_BASE_TIME_MS      (250)  
#define ABNORMAL_DELAY_MS          (50)  
  
#define BUFFER_SIZE                (ABNORMAL_BASE_TIME / ABNORMAL_DELAY)  
  
typedef struct CircularQueue {  
    unsigned char buffer[BUFFER_SIZE]; // 데이터를 저장하는 배열  
    int head;                         // 가장 오래된 데이터의 인덱스  
    int tail;                          // 새로운 데이터를 추가할 인덱스  
    int size;                          // 현재 저장된 데이터의 개수  
} CircularQueue;
```

4. 개발 및 설계

주요 로직

오조작 판단 알고리즘

오조작 상황 종료

- 경고음 출력 종료
- 오조작 종료 flag 설정

오조작 상황 발생

- PWM duty 0
- 오조작 발생 메시지 전송
- 경고음 출력 시작
- 오조작 발생 flag 설정

```

WHILE (true)
    // 현재 DRIVE_MODE가 Parking Mode가 아닌 경우에만 오조작 상황 발생 여부 판단
    IF (DRIVE_MODE != PARKING_MODE) THEN

        // 오조작 상황(ABNORMAL_ACTION)이 이전에 발생한 경우
        IF (ABNORMAL_ACTION == 1) THEN
            // 오조작 상황이 종료되었는지 확인
            IF (check_finish_abnormal()) THEN
                WARNING_SOUND_OFF()           // 경고음 출력 종료
                ABNORMAL_ACTION = 0          // 오조작 상태 해제
            END IF

        // 오조작 상황(ABNORMAL_ACTION)이 발생하지 않은 경우
        ELSE
            // 오조작 상황이 시작되었는지 확인
            IF (check_start_abnormal()) THEN
                PWM_DUTY = 0                  // 모터의 속도를 제어하는 PWM duty를 0으로 설정
                SEND_TO_EXTERNAL_BOARD(IS_ABNORMAL) // 전후방 보드가 모터를 멈출 수 있도록 데이터 전송
                WARNING_SOUND_ON()           // 경고음 출력
                ABNORMAL_ACTION = 1          // 오조작 상태 활성화
            END IF
        END IF

    END IF

    // ABNORMAL_DELAY 시간(ms)동안 대기
    SLEEP(ABNORMAL_DELAY)

END WHILE

```

오조작 상황 발생 후,
브레이크를 세게 밟아야 오조작 상황이 종료

4. 개발 및 설계 구현 기능

가속 및 브레이크 페달 (I2C)

가속 페달 입력 브레이크 페달 입력



아날로그 신호



ADC to I2C 모듈

디지털 신호

TOPST D3
(MICOM)

```
#define I2C_PCF8591_CH          (I2C_CH_MASTER1)
#define I2C_PCF8591_SCL           (GPIO_GPC(20UL))
#define I2C_PCF8591_SDA           (GPIO_GPC(21UL))

#define CONTROL_DATA              ((uint8)0x44)
#define I2C_CLK_RATE_100           (100)
```

→ADC to I2C 모듈에서 채널 2개를 사용하기 위한 명령

```
static SALRetCode_t receiveFromPedalTask(void *pArg)
{
    (void)pArg;

    SALRetCode_t ret;
    uint8 control[1] = {CONTROL_DATA};
    uint8 input_data[2] = {0};
    uint8 isAutohold = 0;

    ret = SAL_RET_SUCCESS;

    // I2C set
    ret = I2C_Open(I2C_PCF8591_CH, I2C_PCF8591_SCL, I2C_PCF8591_SDA, I2C_CLK_RATE_100, NULL, NULL);
    if(ret != SAL_RET_SUCCESS){
        mcu_printf("[DEBUG] I2C OPEN FAIL!!!!!!\n");
        return ret;
    }
}
```

```
while(1){
    ret = PCF8591Read(&input_data[0], (uint32)2, control[0]);
    if(ret != SAL_RET_SUCCESS){
        mcu_printf("[DEBUG] PCF8591 READ FAIL!!!!!!\n");
        return ret;
    }
}
```

```
// accle_data 오크파이프 처리
gPressAccel = (int)input_data[0] - ACCEL_OFFSET;
if(gPressAccel < I2C_NOISE_RANGE ){
    gPressAccel = INPUT_MIN;
}
else if(gPressAccel > INPUT_MAX){
    gPressAccel = INPUT_MAX;
}
```

→gPressAccel 값 업데이트

```
// break_press 오크파이프 처리
gPressBreak = (int)input_data[1] - BREAK_OFFSET;
if(gPressBreak < I2C_NOISE_RANGE ){
    gPressBreak = INPUT_MIN;
}
else if(gPressBreak > INPUT_MAX){
    gPressBreak = INPUT_MAX;
}
```

→gPressbreak 값 업데이트

4. 개발 및 설계

구현 기능

주행모드, 오토홀드(GPIO), 7-segment

```
static SALRetCode_t checkAbnormalActionTask(void *pArg)
{
    // 버튼이 이전에 눌려있지 않았고, 현재 눌려있다면 상승 예지 감지
    if (current_pin_state == 1 && previous_pin_state == 0)
    {
        gAutoholdMode ^= (1); // autohold 값 반전
    }

    // 버튼 상태 업데이트
    previous_pin_state = current_pin_state;
}
```

오토홀드(입력)

```
void autohold_led(char is_on)
{
    if(is_on) {
        project_gpio_set(AUTOHOLD_PIN, 1);
    }
    else {
        project_gpio_set(AUTOHOLD_PIN, 0);
    }
}
```

오토홀드(출력)

```
static SALRetCode_t updateDriveModeTask(void *pArg)
{
    if (D_state)
    {
        gDriveMode = 1; // 1로 세팅
    }
    else if (R_state)
    {
        gDriveMode = 2;
    }
    else
    {
        gDriveMode = 0;
    }
}
```

주행 모드(입력)

```
void drive_mode_led(char drive_mode)
case MODE_D:
    project_gpio_set(MODE_D_PIN, 1);
    project_gpio_set(MODE_R_PIN, 0);
    project_gpio_set(MODE_P_PIN, 0);

    break;

case MODE_R:
    project_gpio_set(MODE_D_PIN, 0);
    project_gpio_set(MODE_R_PIN, 1);
    project_gpio_set(MODE_P_PIN, 0);

    break;

case MODE_P:
    project_gpio_set(MODE_D_PIN, 0);
    project_gpio_set(MODE_R_PIN, 0);
    project_gpio_set(MODE_P_PIN, 1);

    break;
```

주행 모드(출력)

4. 개발 및 설계

구현 기능

모터 제어 (PWM)

```

static void controlMotorTask(void *pArg)
{
    (void)pArg;
    mcu_printf("Motor Task Start!!!\n");
    {
        // 모터 초기화
        Motor_Init(&ModeConfigInfo);

        while (1) {
            // 상태 및 속도 적용
            Motor_Option(gDriveMode, &gPwmDuty);
            {
                // PDM 뉴티 업데이트
                Motor_UpdateDuty(&ModeConfigInfo, gPwmDuty);

                // PDM 설정 적용
                Motor_ConfigurePDM(&ModeConfigInfo);

                SAL_TaskSleep(10);
            }
        }
    }
}

```

1) 전달받은 PwmDuty값에 따라 속도 제어

```

void Motor_UpdateDuty(PDMModeConfig_t *ModeConfigInfo, uint32 speed) { // DutyCycle, Period 설정
    // uint32 duty_percentage;

    ModeConfigInfo->mcDutyNanoSec1 = (speed * (1000UL * 1000UL)) / 255; // x값 0 ~ 255 입력에 따라 PDM값 조절
    ModeConfigInfo->mcPeriodNanoSec1 = 1000UL * 1000UL;

    // duty_percentage = (ModeConfigInfo->mcDutyNanoSec1 * 100UL) / ModeConfigInfo->mcPeriodNanoSec1;

    // mcu_printf("\n == Duty : %d%%\n", duty_percentage);
}

```

2) DriveMode 값에 따라 모터 방향 제어

```

void Motor_Option(uint8 gDriveMode, uint32 *speed) {
    static uint8 prev_mode = 0; // 내부적으로 이전 상태를 관리

    // 방향 전환 감지 및 속도 초기화
    if (gDriveMode != prev_mode) {
        *speed = 0; // 방향 전환 시 속도 초기화
    }

    // 상태에 따른 GPIO 설정
    switch (gDriveMode) {
        case 0: // 정지 상태 (Parking)
            GPIO_Set(IN_2_4, 0UL);
            GPIO_Set(IN_1_3, 0UL);
            *speed = 0; // 정지 시 속도 고정
            break;

        case 1: // 정회전 (Drive)
            GPIO_Set(IN_2_4, 1UL);
            GPIO_Set(IN_1_3, 0UL);
            break;

        case 2: // 역회전 (Reverse)
            GPIO_Set(IN_2_4, 0UL);
            GPIO_Set(IN_1_3, 1UL);
            break;

        default: // 잘못된 입력 처리
            break;
    }

    // 이전 모드 업데이트
    prev_mode = gDriveMode;
}

```

4. 개발 및 설계

구현 기능

적색 신호등 감지 (YOLOv5)

1) 주요 초기화 및 설정

- YOLOv5 모델 로드

```
# YOLOv5 모델 로드
path = '/home/root/best2.pt' # 모델 경로
model = torch.hub.load('ultralytics/yolov5', 'custom', path=path)
classes = ['red'] # 클래스 이름 정의
```

PyTorch를 사용하여 YOLOv5 모델을 로드하고,
"red"라는 클래스만 처리하도록 설정

- IPC 장치 초기화

```
# IPC 디바이스 파일
IPC_DEVICE = "/dev/tcc_ipc_micom"
fcntl.ioctl(fd, IOCTL_IPC_DETECT_DATA, packed_value)
```

IPC 통신을 설정

fcntl.ioctl을 사용해 R5로 데이터를 전송

- 스레드 설정

```
capture_thread = threading.Thread(target=capture_frames)
process_thread = threading.Thread(target=process_frames)
```

캡처 스레드와 추론 스레드를 각각 생성

2) 메인 루프

```
frame, results = result_queue.get() 1. result_queue에서 결과를 가져와
detections = results.pandas().xyxy[0] # Pandas DataFrame으로 결과 가져오기
if classes[cls] == 'red': 현재 프레임의 "빨간불 감지" 여부 확인
    red_detected_in_frame = True # 빨간 불 감지

# 최근 감지된 True의 개수에 따라 현재 상태 결정 (노이즈 필터링 로직)
recent_detections = recent_detections.count(True) >= DETECTION_THRESHOLD
# 노이즈로 인해 일시적으로 True/False로 변경되지 않도록 최소 감지 개수를 조건으로 설정
```

2. 최근 프레임 히스토리(DETECTION_HISTORY : 10) 중
최소 감지 임계값(DETECTION_THRESHOLD : 7) 이상일 경우,
빨간 불 감지(True)로 판단

```
if red_detected != previous_red_detected:
    send_to_ipc(fd, red_detected)
```

3. 이전 상태와 비교하여 상태가 변경되었을 때만 IPC로 데이터 전송

```
# 'q' 키로 종료
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

4. 'q' 키 입력 시 종료

4. 개발 및 설계 구현 기능

보드 내 코어 간 통신 (MBOX)

MICOM

```
typedef struct IPCMboxMsgReg
{
    uint32 mmReg [MAX_MBOX_CMD_FIFO_CNT];
    uint32 mmDat [MAX_MBOX_DATA_FIFO_CNT];
} IPCMboxMsgReg_t;
```

cr5-bsp/sources/app.drivers/ ipc/common/ ipc_cmd.h

```
typedef enum {
    CTL_CMD = 0x0000U,
    WRITE_CMD,
    DATA_CMD,
    MAX_CMD_TYPE
} IPCCmdType_t;
```

cr5-bsp/sources/app.drivers
/ipc/common/ ipc_typedef.h

```
typedef enum IPCCmdId
{
    /* control command */
    IPC_OPEN = 0x0001U,
    IPC_CLOSE,
    IPC_SEND_PING,
    IPC_WRITE,
    IPC_ACK,
    IPC_NACK,
    START_CTL,
    SOUND_ON, // 오조작 상황 시작
    SOUND_OFF, // 오조작 상황 종료
    END_CTL,
    MAX_CMD_ID,
} IPCCmdId_t;
```

Main Core

```
struct tcc_mbox_data {
    uint32_t cmd [MBOX_CMD_FIFO_SIZE];
    uint32_t data [MBOX_DATA_FIFO_SIZE];
    uint32_t data_len;
};
```

kernel-source/include/linux/mailbox/tcc805x_multi_mailbox
/tcc805x_multi_mbox.h

```
typedef enum {
    CTL_CMD = 0x0000U,
    WRITE_CMD,
    DATA_CMD,
    MAX_CMD_TYPE
} IpcCmdType;
```

kernel-source/drivers/char
/tcc_ipc/tcc_ipc_typedef.h

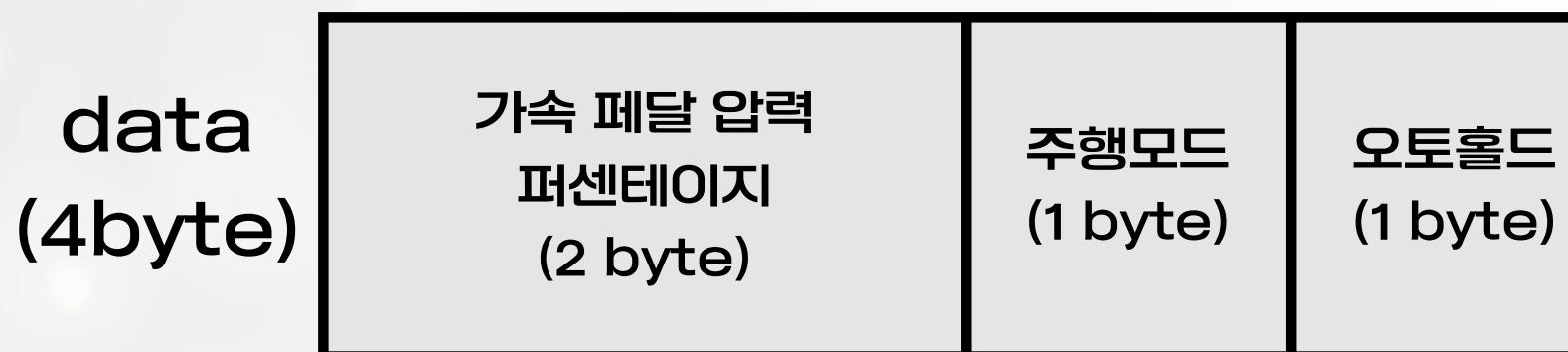
```
typedef enum {
    /* control command */
    IPC_OPEN = 0x0001U,
    IPC_CLOSE,
    IPC_SEND_PING,
    IPC_WRITE,
    IPC_ACK,
    IPC_NACK,
    START_CTL,
    SOUND_ON, // 오조작 상황 시작
    SOUND_OFF, // 오조작 상황 종료
    END_CTL,
    MAC_CMD_ID
} IpcCmdID;
```

4. 개발 및 설계

구현 기능

보드 내 코어 간 통신 (MBOX)

1) MICOM → Main Core



```

int32 IPC_SendData
(
    IPCSvCh_t           siCh,
    uint32               data
)
{
    int32   ret;

    sendMsg[siCh].mmReg[0] = IPC_GetSequentialId(siCh);
    sendMsg[siCh].mmReg[1] = ((uint32) DATA_CMD) << 16U;
    sendMsg[siCh].mmReg[2] = sizeof(data);
    sendMsg[siCh].mmReg[3] = 0U;

    sendMsg[siCh].mmDat[0] = data;
    sendMsg[siCh].mmDat[1] = 0U;

    ret = MBOX_DevSend(dev[siCh].mbDev, sendMsg[siCh].mmReg, sendMsg[siCh].mmDat, sizeof(data));

    return ret;
}

```

MICOM

```

ret = ipc_receive_queue_init(
    &ipc_handle->receiveQueue[DATA_CMD],
    &ipc_receive_datacmd,
    NULL,
    "tc_ipc_data_recevie_handler");

```

```

if ((pMsg != NULL) &&
    (ipc_dev != NULL) &&
    (ipc_dev->ipc_handler.ipcStatus == IPC_READY)) {
    accel_press = ((pMsg->data[0] & (IPC_UINT32)DATA_ACCEL_MASK)
                   >> (IPC_UINT32)16);
    drive_mode = ((pMsg->data[0] & (IPC_UINT32)DATA_DRIVE_MODE_MASK)
                  >> (IPC_UINT32)8);
    autohold = (pMsg->data[0] & (IPC_UINT32)DATA_AUTOHOLD_MASK);

    display_segment(accel_press);

    if (flag_drive_mode != drive_mode)
    {
        drive_mode_led(drive_mode);
        flag_drive_mode = drive_mode;
    }

    if(flag_autohold != autohold)
    {
        autohold_led(autohold);
        flag_autohold = autohold;
    }
}

```

Main Core

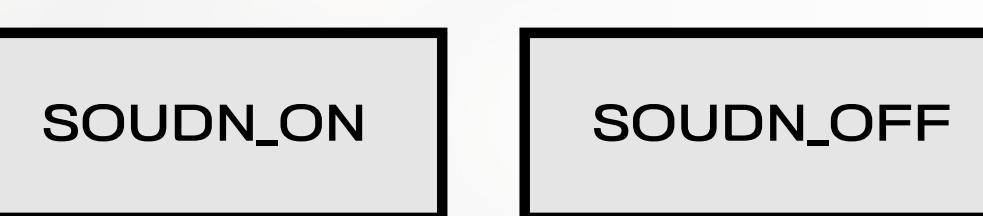
4. 개발 및 설계

구현 기능

보드 내 코어 간 통신 (MBOX)

1) MICOM → Main Core

uiCmdID



```
int32 IPC_SendCtl
(
    IPCSvCh_t          siCh,
    uint32              uiCmdID
)
{
    int32 ret = 1;

    sendMsg[siCh].mmReg[0] = TPC_GetSequentialId(siCh);
    sendMsg[siCh].mmReg[1] = (((uint32) CTL_CMD) << 16U) | ((uint32) uiCmdID);
    sendMsg[siCh].mmReg[2] = 0U;
    sendMsg[siCh].mmReg[3] = 0U;
    sendMsg[siCh].mmReg[4] = 0U;

    IPC_CmdWakePreSet(siCh, CTL_CMD, sendMsg[siCh].mmReg[0]);

    MBOX_DevSendCmd(dev[siCh].mbDev, sendMsg[siCh].mmReg);

    return ret;
}
```

MICOM

```
ret = ipc_receive_queue_init(
    &ipc_handle->receiveQueue[CTL_CMD],
    &ipc_receive_ctlcmd,
    NULL,
    "tc_ipc_ctl_recevie_handler");
```

```
case SOUND_ON:
    sound_on();
    break;

case SOUND_OFF:
    sound_off();
    break;
```

Main Core

4. 개발 및 설계 구현 기능

보드 내 코어 간 통신 (MBOX)

2) Main Core → MICOM

적색 신호등 점등 여부를 판단하는 파이썬 코드

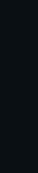
```
# IPC 디바이스 파일
IPC_DEVICE = "/dev/tcc_ipc_micom"

# TCC_IPC_MAGIC과 IOCTL_IPC_DETECT_DATA 정의
TCC_IPC_MAGIC = 'I'
IOCTL_IPC_DETECT_DATA = (ord(TCC_IPC_MAGIC) << 8) | 6
```

:

```
def send_to_ipc(fd, value):
    try:
        packed_value = struct.pack('i', int(value))
        fcntl.ioctl(fd, IOCTL_IPC_DETECT_DATA, packed_value)
        print(f"Sent detect data to IPC: {value}")
    except Exception as e:
        print(f"Error sending to IPC: {e}")
```

적색등 감지 여부 데이터



```
#define TCC_IPC_MAGIC ('I')

#define IOCTL_IPC_SET_PARAM          (_IO(TCC_IPC_MAGIC, 1))
#define IOCTL_IPC_GET_PARAM          (_IO(TCC_IPC_MAGIC, 2))
#define IOCTL_IPC_PING_TEST          (_IO(TCC_IPC_MAGIC, 3))
#define IOCTL_IPC_FLUSH              (_IO(TCC_IPC_MAGIC, 4))
#define IOCTL_TPC_TSREADY           (_IO(TCC_TPC_MAGIC, 5))
#define IOCTL_IPC_DETECT_DATA        (_IO(TCC_IPC_MAGIC, 6))
```

source_kernel/include/uapi/linux/tcc_ipc.h

```
case IOCTL_IPC_DETECT_DATA:
{
    IPC_INT32 input_data;

    if (copy_from_user(&input_data, (const void *)arg, sizeof(int)) == 0)
    {
        // IPC를 통해 데이터 전송
        ret = ipc_send_detect_data(ipc_dev, &input_data);
    }
    else
    {
        ret = -EINVAL; // 에러: 잘못된 인수
    }
}
break;
```

tcc_ipc_ioctl() 함수

4. 개발 및 설계 구현 기능

보드 내 코어 간 통신 (MBOX)

2) Main Core → MICOM

```

case DATA_CMD:
{
    IPC_ReceiveDetectData(IPC_SVC_CH_CA72_NONSECURE, (const IPCMboxMsgReg_t *) &recv[MBOX_CH_CA72MP_NONSECURE]);
    break;
}

if(pMsg != NULL_PTR)
{
    if(pMsg->mmDat[0] != 0 && pMsg->mmDat[0] != 1)
    {
        return;
    }

    if(gAutoholdMode){
        send_data[1] = pMsg->mmDat[0];
        (void)uart_send_data(UART_CH_FOR_MID, send_data, MID_TX_BUFFER_SIZE);
    }
}

```

MICOM

ipc_send_detect_data() 함수

```

(void)memset(&sendMsg, 0x00, sizeof(struct tcc_mbox_data));

sendMsg.cmd[0] = get_sequential_ID(ipc_dev);
sendMsg.cmd[1] = ((IPC_UINT32)DATA_CMD << (IPC_UINT32)16);
sendMsg.cmd[2] = sizeof(data);

(void)memcpy((void *)&sendMsg.data,
             (const void *)data,
             (size_t)sizeof(data));

sendMsg.data_len = sizeof(data);
ipc_cmd_wake_preset(ipc_dev, DATA_CMD, sendMsg.cmd[0]);

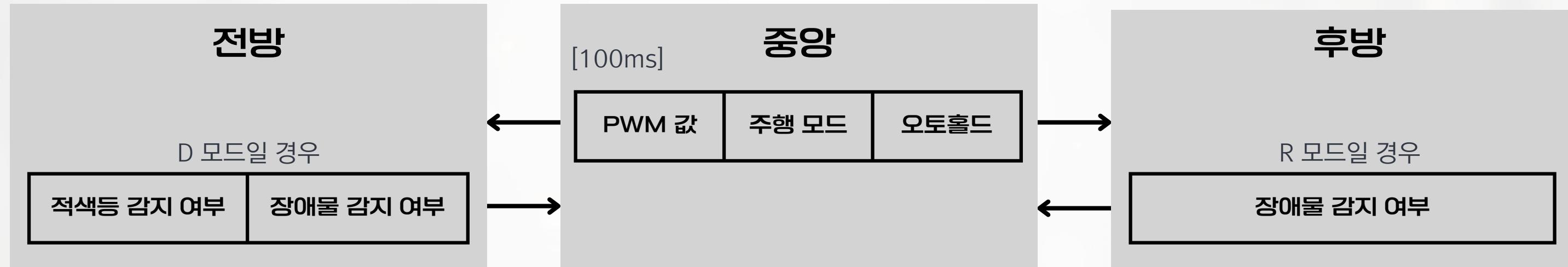
ret = ipc_mailbox_send(ipc_dev, &sendMsg);

```

Main Core

4. 개발 및 설계 구현 기능

보드 간 통신 (UART) - 기존 방식



문제 발생

- 수신 데이터 처리 지연
- 데이터 밀림

해결 방법

```
// UART Interrupt FIFO Level status register Fields
#define UART_IFLS_RXIFLSEL          (7UL << 3)
#define UART_IFLS_TXIFLSEL          (7UL << 0)

if((ucCh == UART_CH1) || (ucCh == UART_CH2)){
    uint32 ifsl;

    ifsl = UART_RegRead(ucCh, UART_REG_IFLS);
    ifsl &= ~(UART_IFLS_RXIFLSEL|UART_IFLS_TXIFLSEL);
    UART_RegWrite(ucCh, UART_REG_IFLS, ifsl);
}
```

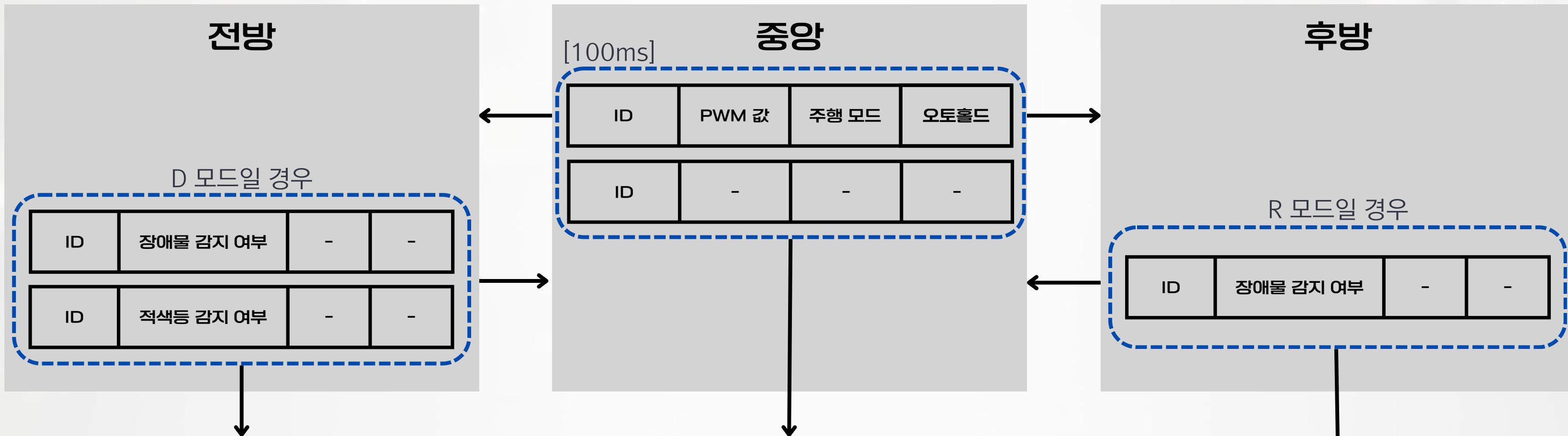
IFLS 레지스터 값 설정

- Default (0x12)
 - TXIFLSEL: 0x2
 - 인터럽트 FIFO level: 1/2 full
 - RXIFLSEL: 0x2
 - 인터럽트 FIFO level: 1/2 full
- 설정 값 (0x00)
 - TXIFLSEL: 0x0
 - 인터럽트 FIFO level: 1/8 full
 - RXIFLSEL: 0x0
 - 인터럽트 FIFO level: 1/8 full

데이터 사이즈 변경: 4byte

4. 개발 및 설계 구현 기능

보드 간 통신 (UART) - 최종 방식



- 범위 내 장애물 감지 여부가 변경된 경우
 - ID: ID_ULTRA SONIC(0x02)
 - DATA: 장애물 감지 여부
- 적색등 감지 여부가 변경된 경우
 - ID: ID_RED(0x03)
 - DATA: 적색등 감지 여부
- 오조작이 감지되지 않은 경우
 - ID: ID_DATA(0x00)
 - DATA: PWM 값, 주행모드, 오토홀드
- 오조작이 감지된 경우
 - ID: ID_ANORMAL(0x01)

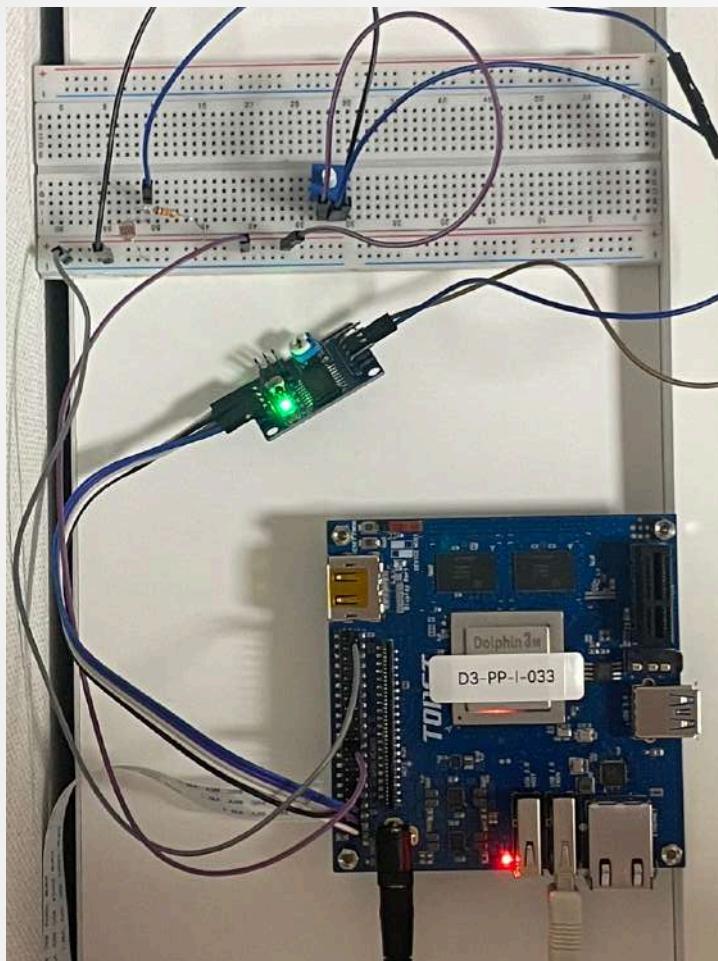
Section 5

테스팅 및 결과

5. 테스팅 및 결과 단위 테스트

페달, ADC to I2C 모듈 테스트

- ADC to I2C 모듈의 입력 채널 2개를 사용하여 값을 받는 테스트



```
PutTY (inactive)
[13000] AIN0: 113 AIN1: 255
[13250] AIN0: 164 AIN1: 255
[13500] AIN0: 162 AIN1: 255
[13750] AIN0: 161 AIN1: 255
[14000] AIN0: 160 AIN1: 255
[14250] AIN0: 158 AIN1: 255
[14500] AIN0: 159 AIN1: 255
[14750] AIN0: 160 AIN1: 255
[15000] AIN0: 161 AIN1: 255
[15250] AIN0: 139 AIN1: 192
[15500] AIN0: 131 AIN1: 150
[15750] AIN0: 102 AIN1: 142
[16000] AIN0: 92 AIN1: 143
[16250] AIN0: 77 AIN1: 143
[16500] AIN0: 78 AIN1: 143
[16750] AIN0: 78 AIN1: 144
[17000] AIN0: 77 AIN1: 180
[17250] AIN0: 76 AIN1: 228
[17500] AIN0: 77 AIN1: 226
[17750] AIN0: 77 AIN1: 228
[18000] AIN0: 76 AIN1: 236
[18250] AIN0: 76 AIN1: 236
```

오토홀드 톤글 테스트

```
while (1)
{
    // 현재 버튼 상태 읽기
    uint32_t current_pin_state = GPIO_Get(GPIO_GPG(6UL));

    // 버튼이 이전에 눌려있지 않았고, 현재 눌려있다면 상승 에지 감지
    if (previous_pin_state == 0 && current_pin_state == 1)
    {
        mcu_printf("Button pressed.\n");
        autohold = !autohold; // autohold 값 반전
        mcu_printf("autohold is now: %d\n", autohold);
    }

    // 버튼 상태 업데이트
    previous_pin_state = current_pin_state;

    // 10ms 간격으로 상태 확인
    (void)SAL_TaskSleep(10);
}
```

```
-----TCC_EVM_BD_VERSION == TCC8050_BD_VER_0_1-----
Start checking autohold
Configuration succeeded.
cnt: 0
KEY_AppIPCHandleForMainAP
KEY_AppIPCHandleForSubAP
Button pressed.
autohold is now: 1
Button pressed.
autohold is now: 0
Button pressed.
autohold is now: 1
Button pressed.
autohold is now: 0
Button pressed.
autohold is now: 1
Button pressed.
autohold is now: 0
Button pressed.
autohold is now: 1
Button pressed.
autohold is now: 0
Button pressed.
```

5. 테스팅 및 결과

단위 테스트

모터 테스트

```

== Duty : 7%
Speed (x): 20, gDriveMode: 1, Increasing: 0

-- Duty : 5%
Speed (x): 15, gDriveMode: 1, Increasing: 0

== Duty : 3%
Speed (x): 10, gDriveMode: 1, Increasing: 0

== Duty : 1%
Speed (x): 5, gDriveMode: 1, Increasing: 0
Direction changed: gDriveMode

== Duty : 0%
Speed (x): 0, gDriveMode: 1, Increasing: 0

== Duty : 1%
Speed (x): 4, gDriveMode: 1, Increasing: 0

== Duty : 3%
Speed (x): 8, gDriveMode: 1, Increasing: 0

```



초음파 거리 측정 테스트

```

static void receiveFromTrasonicTask(void *pArg)
{
    (void)pArg;
    mcu_printf("ultra test start\n");

    ultrasonic_init();

    while (1) {
        if()
            gDistance = get_distance();
            mcu_printf("Distance : %dcm\n",gDistance);
            SHL_TASKSLEEP(10),
    }
}

```

```

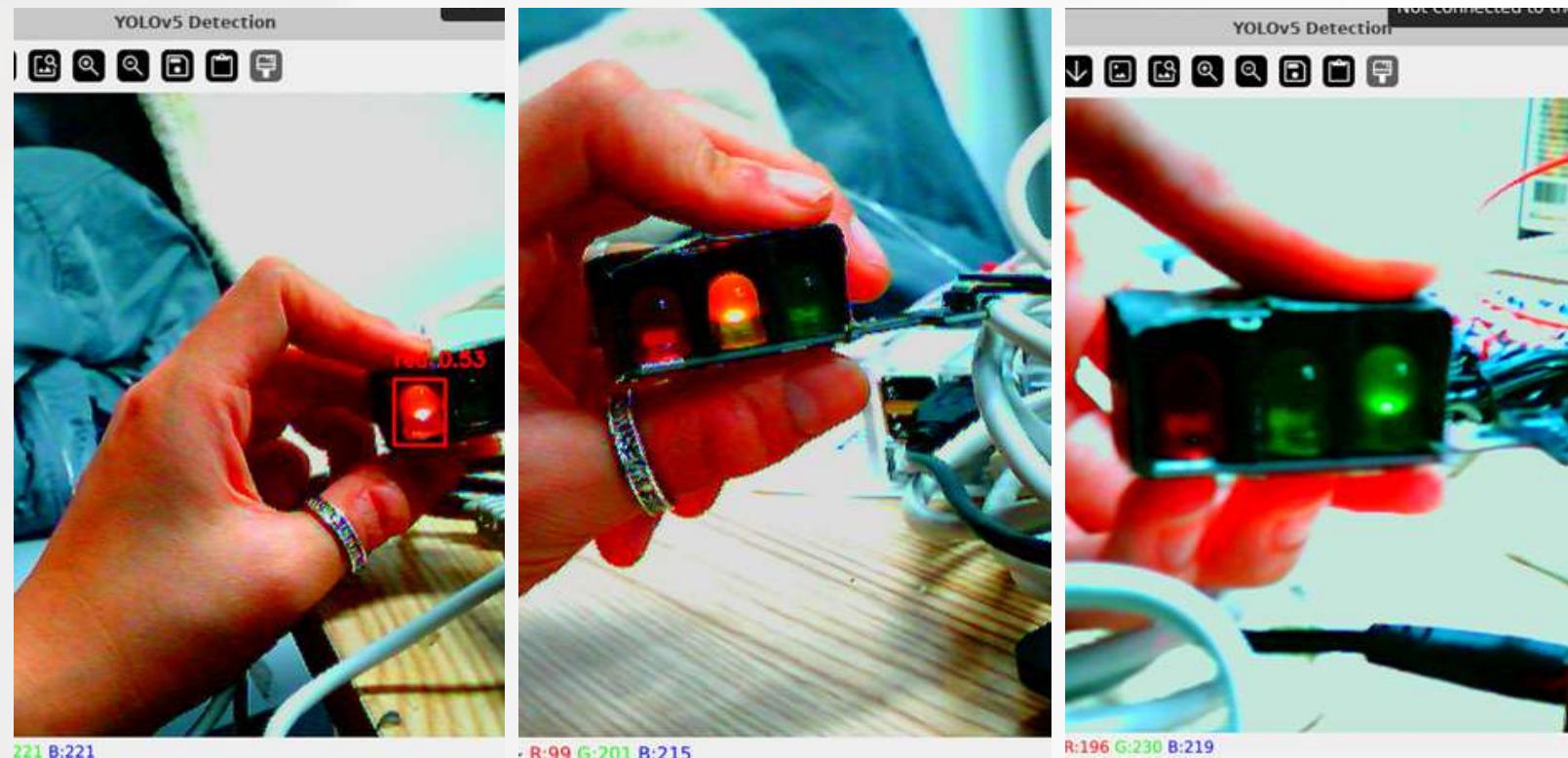
Distance : 198 cm
Distance : 198 cm
Distance : 59 cm
Distance : 61 cm
Distance : 51 cm
Distance : 49 cm
Distance : 47 cm
Distance : 37 cm
Distance : 32 cm
Distance : 27 cm
Distance : 21 cm
Distance : 17 cm
Distance : 12 cm
Distance : 8 cm
Distance : 3 cm
Distance : 200 cm
Distance : 11 cm
Distance : 806 cm
Distance : 11 cm
Distance : 11 cm

```

5. 테스팅 및 결과

단위 테스트

카메라 테스트



빨간 불일 때

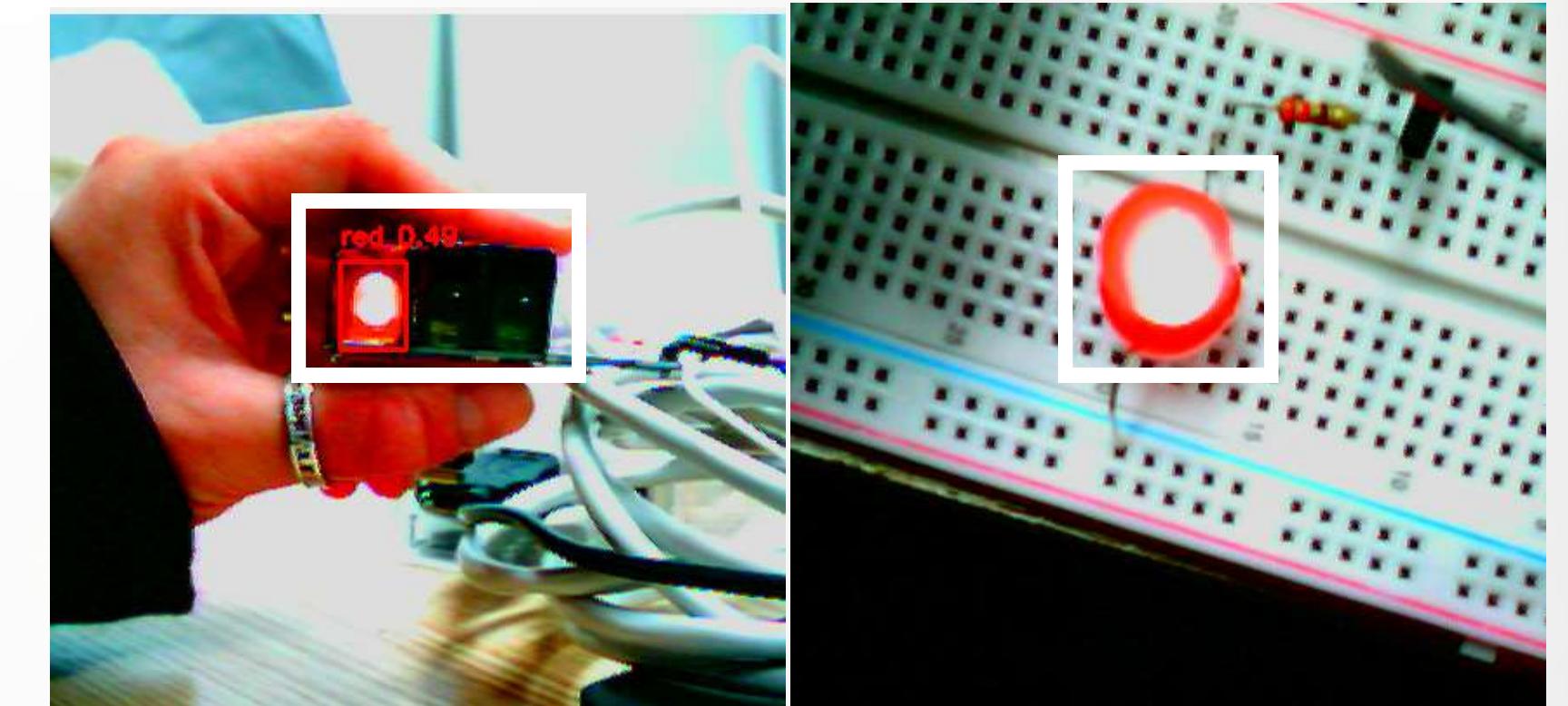
```
tect data to IPC: True  
ht detected. Sending True
```

노란 불일 때

```
ect data to IPC: False  
ight detected. Sending False
```

초록 불일 때

```
ect data to IPC: False  
ight detected. Sending False
```



**‘적색 신호등’ 만을 인식
일반 불빛일 때는 인식 X**

5. 테스팅 및 결과

보드 별 통합 테스트

- [중앙] 입력에 따른 출력 테스트

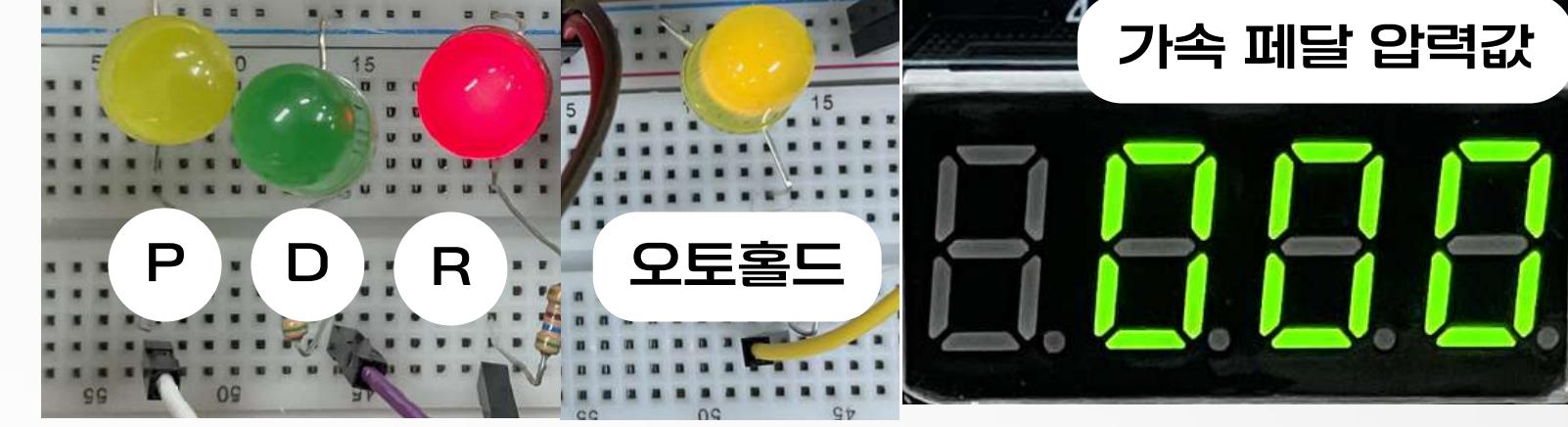
입력

```
gAccelPercent: 59
gDriveMode: 0 (0: P mode, 1: D mode, 2: R mode)
gAccelPercent: 59
gDriveMode: 0 (0: P mode, 1: D mode, 2: R mode)
gAccelPercent: 59
gDriveMode: 0 (0: P mode, 1: D mode, 2: R mode)
gAccelPercent: 59
gDriveMode: 0 (0: P mode, 1: D mode, 2: R mode)
```

출력



```
gAccelPercent: 0
gDriveMode: 2 (0: P mode, 1: D mode, 2: R mode)
gAccelPercent: 0
gDriveMode: 2 (0: P mode, 1: D mode, 2: R mode)
gAccelPercent: 0
gDriveMode: 2 (0: P mode, 1: D mode, 2: R mode)
gAccelPercent: 0
Autoholde: 1
```



```
gAccelPercent: 0
gDriveMode: 1 (0: P mode, 1: D mode, 2: R mode)
gAccelPercent: 0
gDriveMode: 1 (0: P mode, 1: D mode, 2: R mode)
gAccelPercent: 0
gDriveMode: 1 (0: P mode, 1: D mode, 2: R mode)
gAccelPercent: 0
gDriveMode: 1 (0: P mode, 1: D mode, 2: R mode)
```



5. 테스팅 및 결과

시연 영상

Section 6

협업과정

6. 협업과정

보드별 빌드 방법

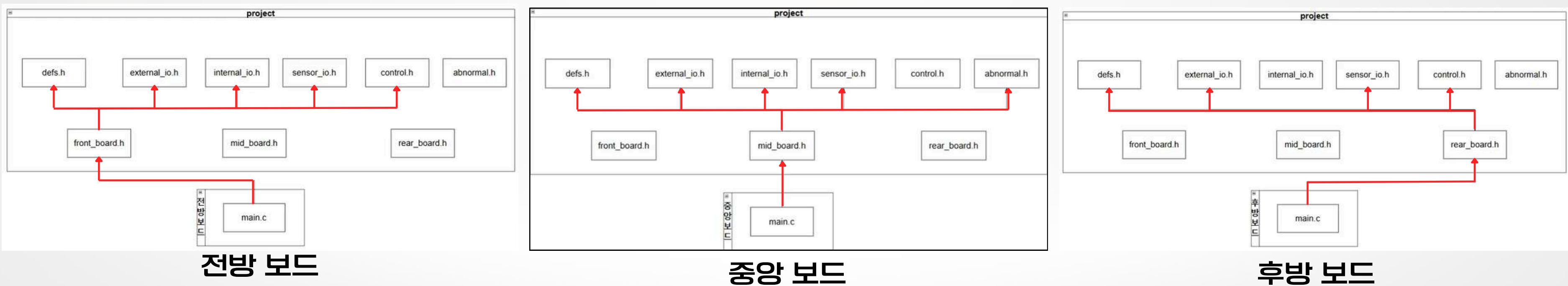
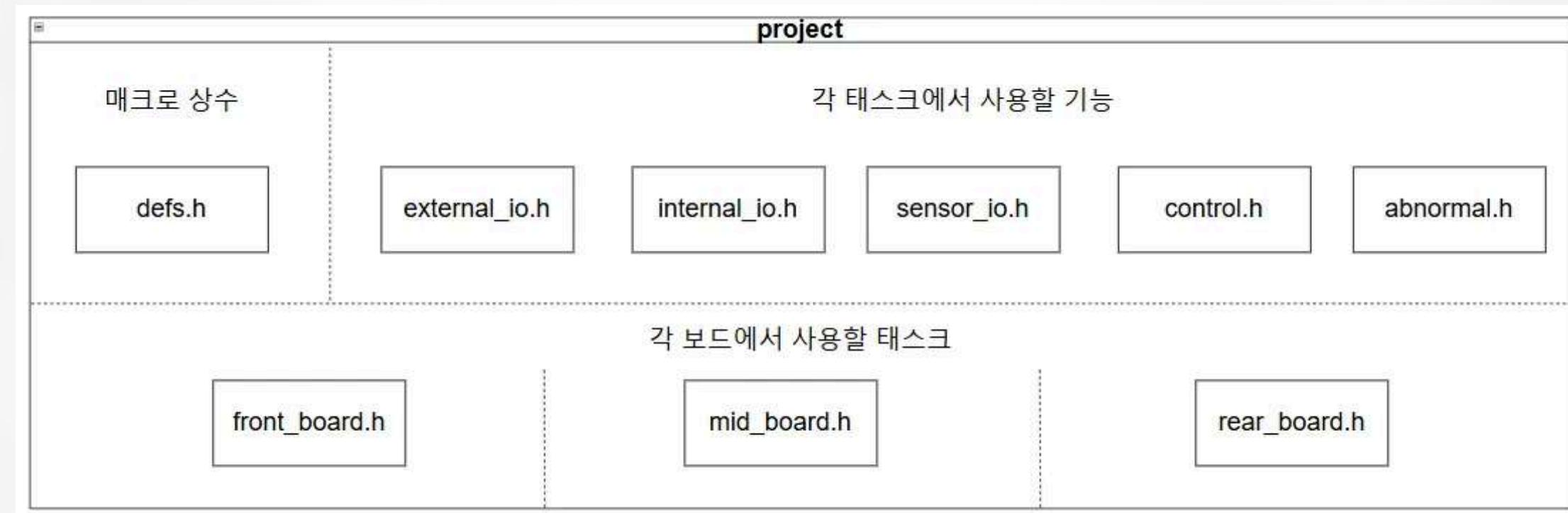


하나의 개발 환경에서 세 가지 보드(전방, 중앙, 후방)에 대한 빌드가 모두 가능하도록 소프트웨어를 관리하자!

- 3개의 보드에서 모터 제어, 보드 간 통신 등 공통적으로 사용하는 기능 존재
 - 각 팀원들은 보드 기준이 아닌, 구현해야 하는 기능을 기준으로 역할을 분담
- 팀원들은 하나의 보드가 아닌 여러 보드에서 작업을 해야 할 가능성이 높음

6. 협업과정

보드별 빌드 방법



6. 협업과정

보드별 빌드 방법

main.c와 rules.mk를 수정함으로써 모든 보드에 대해 빌드 가능

예시) 중앙 보드에 대한 빌드를 하는 방법

```
// ----- 보드에 따라 아래의 값 중 하나만 주석 제거
// #define CONFIG_FRONT // 전방 보드일 때
#define CONFIG_MID // 중앙 보드일 때
// #define CONFIG_REAR // 후방 보드일 때
// -----



#ifndef CONFIG_FRONT
#include "front_board.h"
#endif

#ifndef CONFIG_MID
#include "mid_board.h"
#endif

#ifndef CONFIG_REAR
#include "rear_board.h"
#endif
```

main.c

```
# ----- 보드에 맞게 설정 -----
# 보드(전방(CONFIG_FRONT) | 중앙(CONFIG_MID) | 후방(CONFIG_REAR)) = 1
CONFIG_MID= 1
# -----
```

```
# Sources
ifdef CONFIG_FRONT
SRCS += front_board.c \
control.c \
external_io.c \
internal_io.c \
ultra_sonic.c
endif

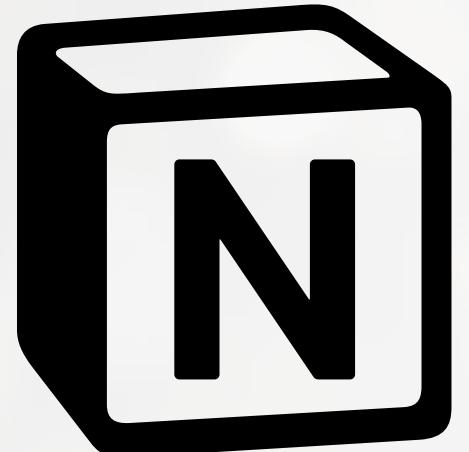
ifdef CONFIG_MID
SRCS += mid_board.c \
abnormal.c \
external_io.c \
internal_io.c \
sensor_io.c
endif

ifdef CONFIG_REAR
SRCS += rear_board.c \
control.c \
external_io.c \
ultra_sonic.c
endif
```

rules.mk

6. 협업과정

협업 툴



Notion



텔레강스

| 프로젝트 기간 2024.11.01 ~ 2024.12.17



공유 파일

To Do List

안내 사항

분석 내용 정리

SW

기타

회의록

멘토링

제출 파일

멘토링

⋮ List

[7주차] 최종 멘토링

2024/12/13

[6주차] 보드 간 통신 및 최종 발표 준비

2024/12/06

[5주차] CAN 및 라이브러리화 질문

2024/11/29

[4주차] 중간 발표

2024/11/22

[3주차] 중간발표 초안 피드백

2024/11/15

[2주차] 주제 컨펌

2024/11/08

[1주차] 멘토 첫 만남 및 프로젝트 방향성 조언

2024/11/01

분석 내용 정리

⋮ All ⋮ TOPSTD3 ⋮ GPIO ⋮ I2C ⋮ CAN ⋮ IPC ⋮ SPI ⋮ Camera

Camera detect

camera source code A72 ● 완료 ●

PWM - MOTOR

Motor source code R5 ● 완료 ● 수

GPIO - Autohold

GPIO manual R5 ● 완료 ● 우

main → micom

IPC manual A72 ● 완료 ●

CAN 데모 통신

CAN analysis R5 ● 완료 ● 중도

micom에서 main으로 데이터 보내기

R5 IPC manual ● 완료 ●

GPIO - UltraSonic

Ultra Sonic source code R5 ● 완료 ● 수

[GPIO] 7-segment

A72 LED analyze manual ● 완료 ● J

6. 협업과정

협업 툴



GitHub

The screenshot shows a GitHub repository for a "Pedal Abnormal Detection System".

README:

[텔레강스] 페달 오조작 감지 보조 시스템 🚗

텔레강스 팀의 협업 공간입니다.

File List:

File	Status
abnormal.c	complete
abnormal.h	complete
control.c	modify ready
control.h	modify receive from external board task at rear and front
defs.h	complete exclude camera
external_io.c	complete exclude camera
external_io.h	complete exclude camera
front_board.c	complete exclude camera
front_board.h	complete exclude camera

Commit History:

- o- Commits on Dec 17, 2024
 - edit control, rear_board
subin committed 3 days ago
- o- Commits on Dec 10, 2024
 - Merge branch 'main' of https://github.com/junghyun21/topst_d3
UyeongChoe committed last week
 - merged motor control, ultra sonic, uart at front and rear board
UyeongChoe committed last week
- o- Commits on Dec 9, 2024
 - Edit mid_board.h
JieunSong99 committed last week
 - Edit mid_board
JieunSong99 committed last week
 - add check abnormal action
junghyun21 committed last week

6. 협업과정

역할 및 담당업무



팀장 송지은



팀원 이정현



팀원 황서현



팀원 최우영



팀원 김종우



팀원 임수빈

- 보드 간 통신
- 7-segment 출력
- ADC to I₂C
- 코드 통합

- SW 아키텍처 설계
- 코어 간 IPC 통신
- 오조작 알고리즘 설계
- 코드 통합

- 영상 편집
- 카메라
- Yolov5 모델 커스텀데이터 학습
- 실시간 객체 인식

- 주행 모드 / 오토홀드 스위치 및 LED
- 보드 간 통신
- 결선도 작성
- 코드 통합

- 발표
- 보드 간 통신
- HW 아키텍처

- 영상
- 모터 제어
- 초음파 거리 측정

Section 7

정리 및 회고

7. 정리 및 회고

향후 개선점

1. 보드 간 통신에 CAN 통신 적용

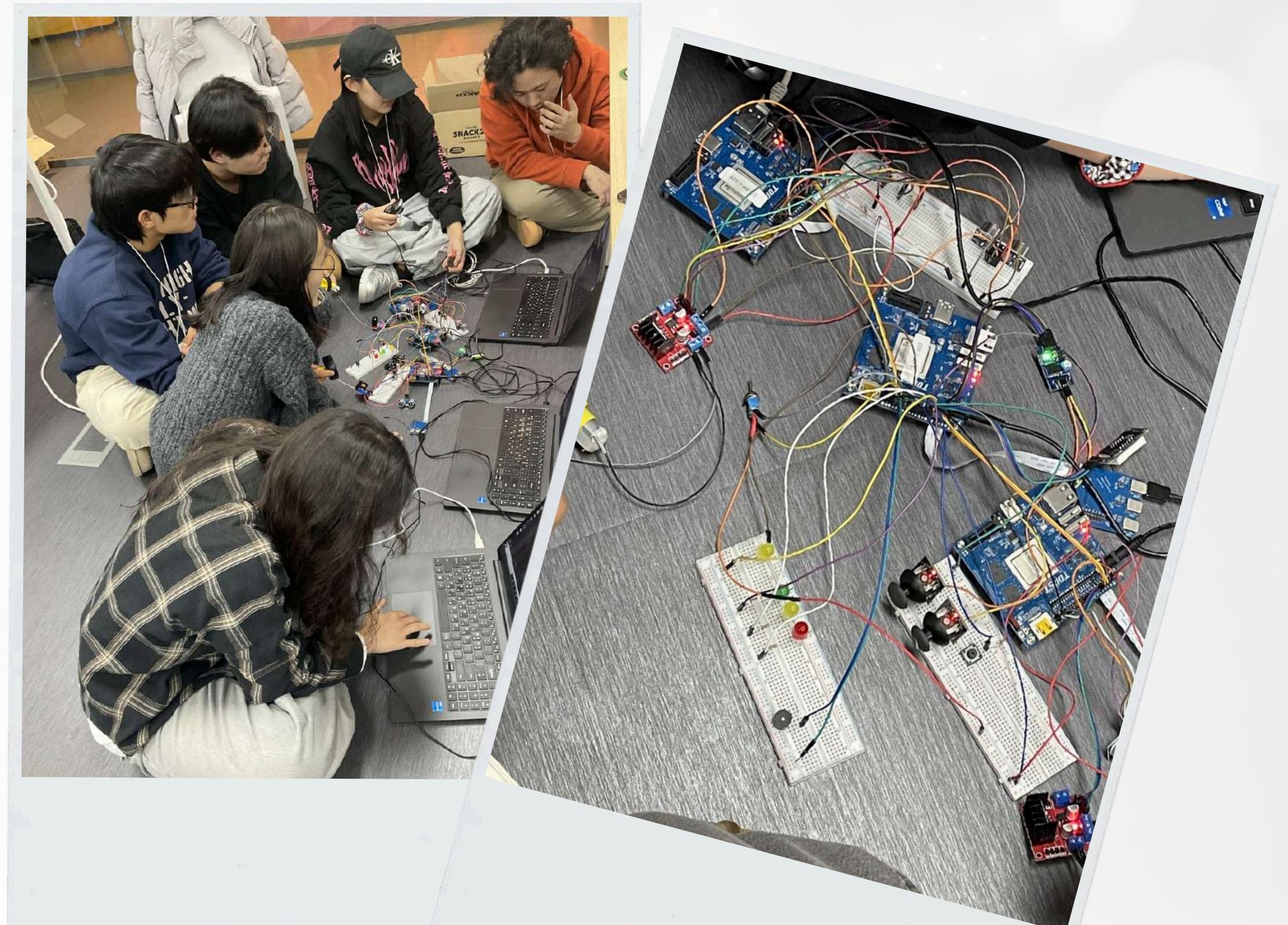
2. 카메라 구동 시, 전력 및 발열 최적화

3. 추가적인 오조작 케이스 모색

7. 정리 및 회고 협업의 가치

“성장은 혼자 하는 것이 아니고,
모두가 같이 할 때,
이루어지는 결과다”

[James Cash Penney]



THANK YOU

3조

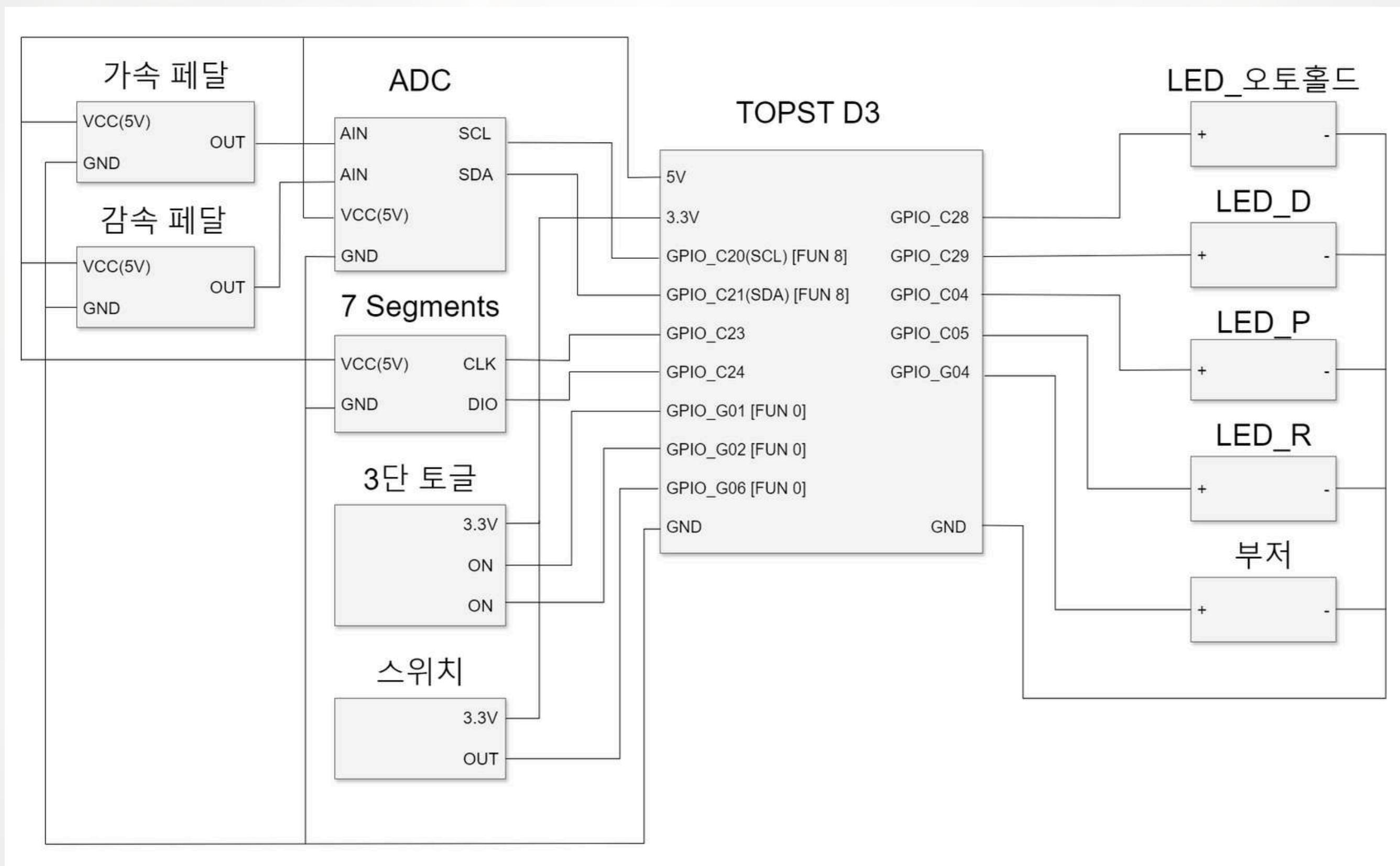
도움 주신 분들: 문호선 강사님, 박동일 멘토님, 심윤석 멘토님

51

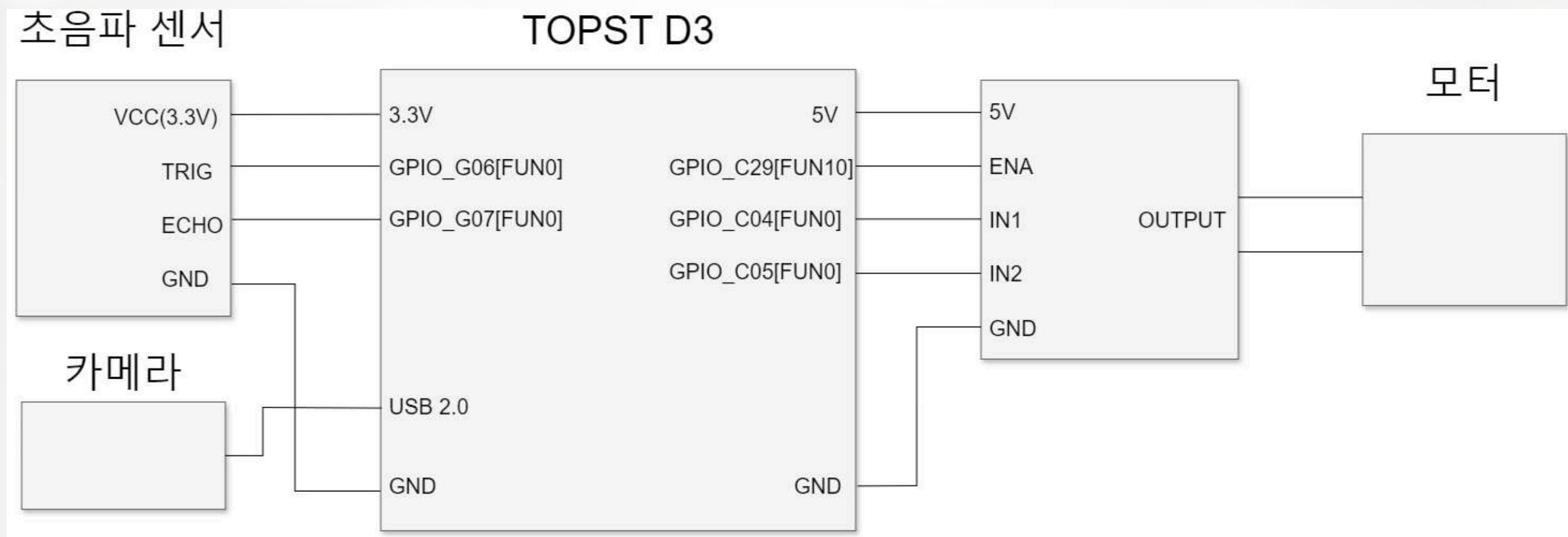
뒷장 부록

APPENDIX

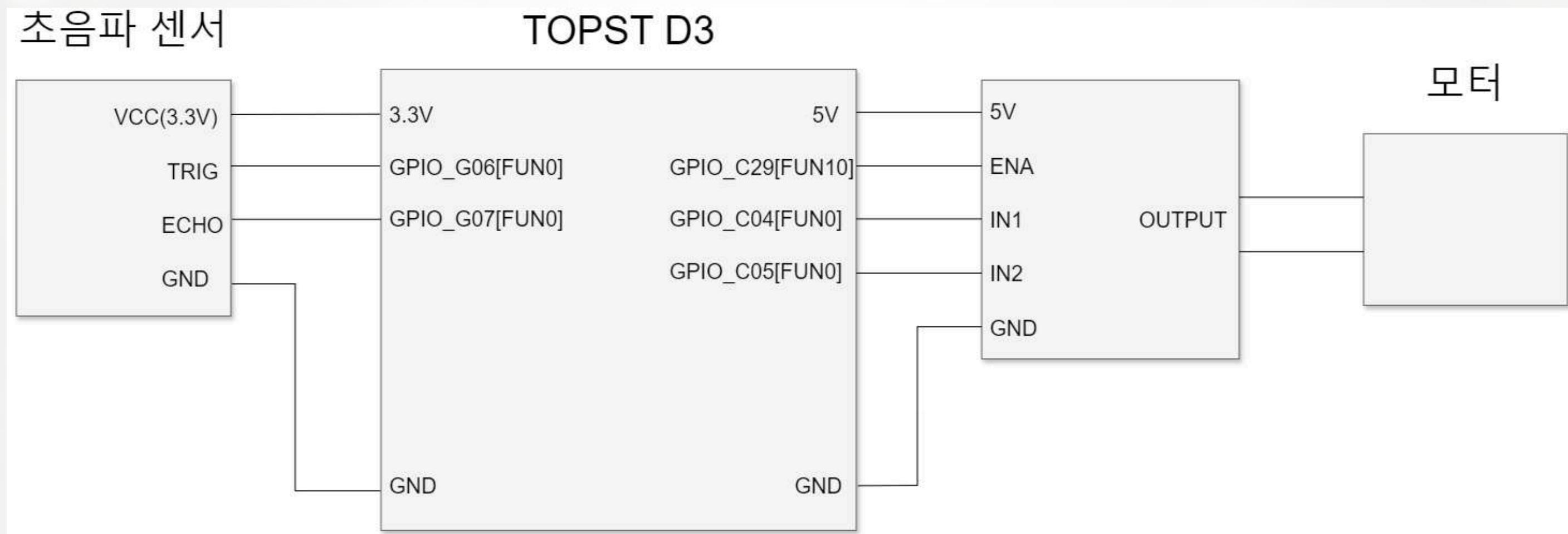
중앙보드 하드웨어 핀 결선도 구성



전방보드 하드웨어 핀 결선도 구성



후방보드 하드웨어 핀 결선도 구성



시연 시나리오

모드에 따른 동작



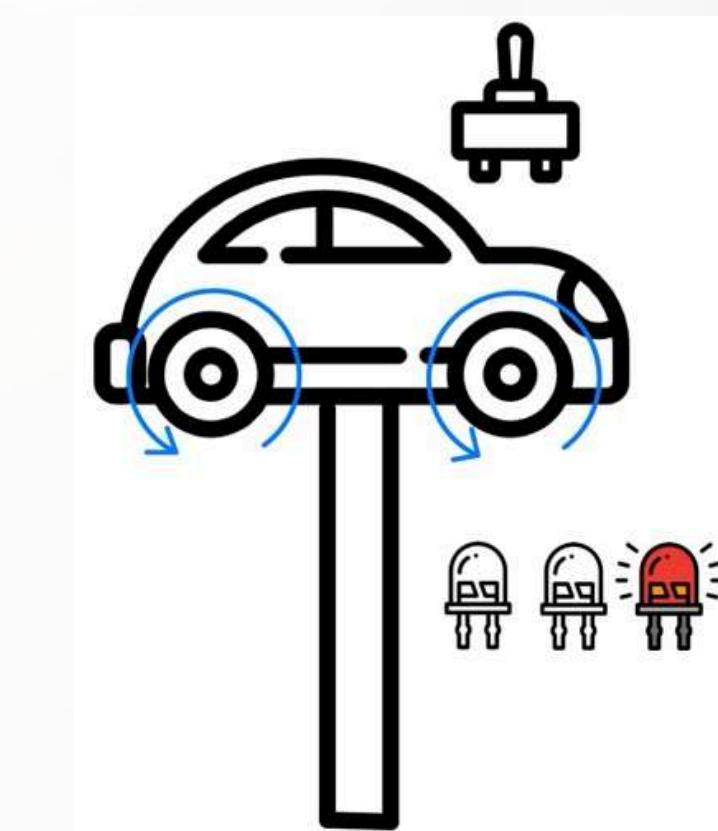
D Mode

- 입력
 - 토글 스위치 입력(D)
- 출력
 - 모터 정회전
 - 모드 별 LED ON(D)



P Mode

- 입력
 - 톤 스위치 입력(P)
- 출력
 - 모터 정지
 - 모드 별 LED ON(P)

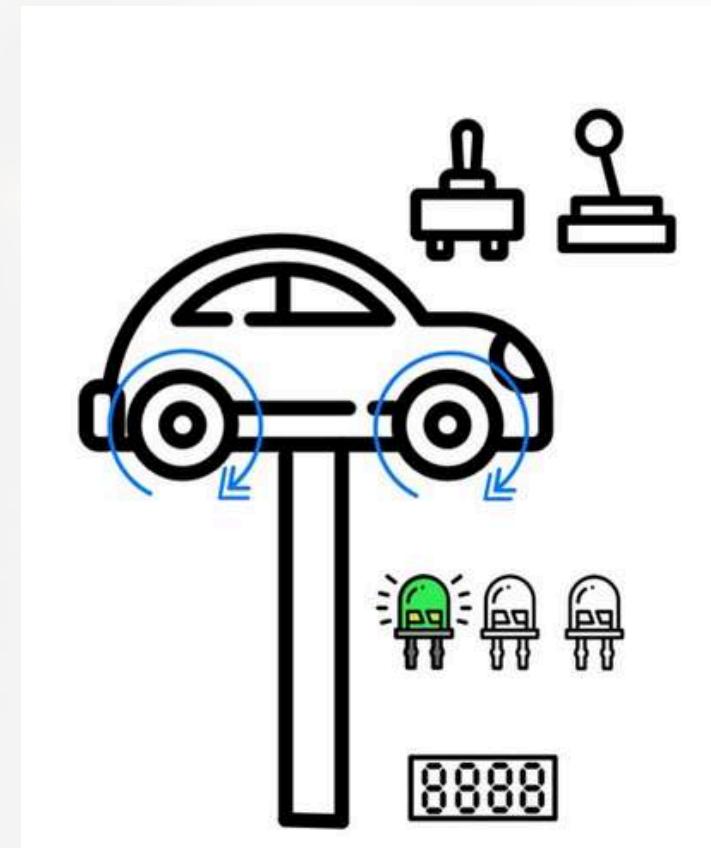


R Mode

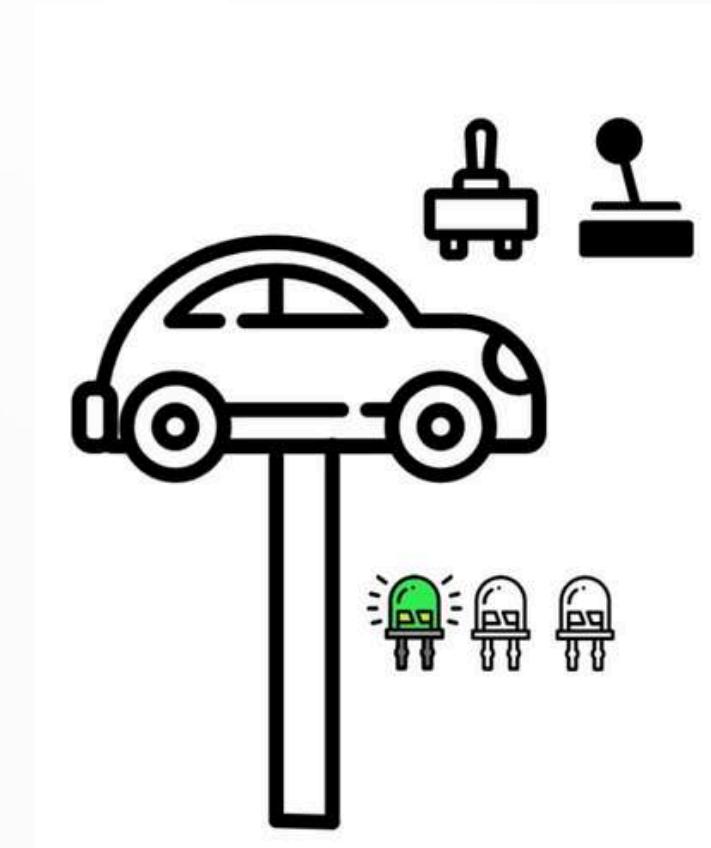
- 입력
 - 톤 스위치 입력(R)
- 출력
 - 모터 역회전
 - 모드 별 LED ON(R)

시연 시나리오

페달 압력에 따른 동작



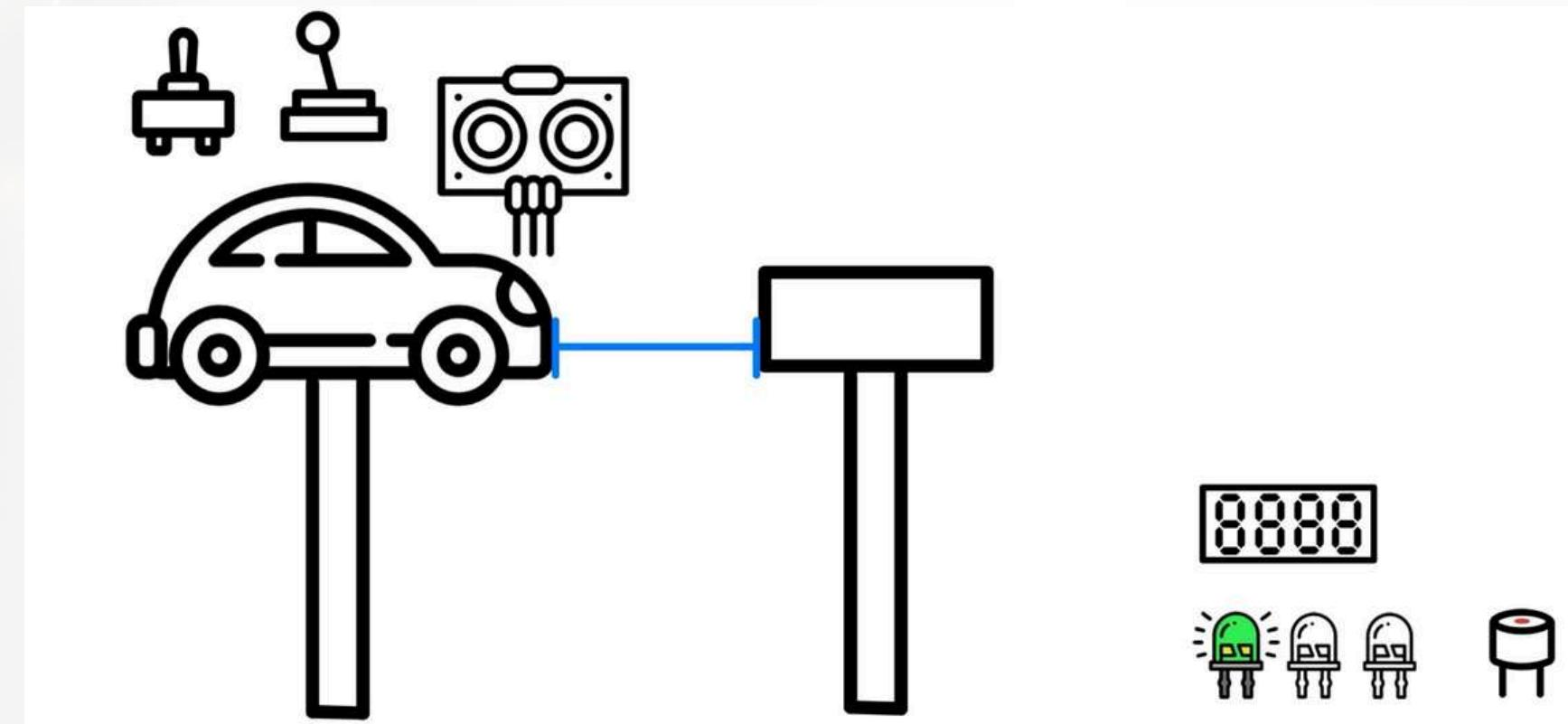
- 입력
 - 토글 스위치 입력(D or R)
 - 조이 스틱(가속 페달)
- 출력
 - 모터 속도 증가
 - 가속 페달 압력 비율 출력
 - 모드 별 LED ON(D or R)



- 입력
 - 토글 스위치 입력(D or R)
 - 조이 스틱(브레이크 페달)
- 출력
 - 모터 속도 감소
 - 모드 별 LED ON(D or R)

시연 시나리오

급가속 검출



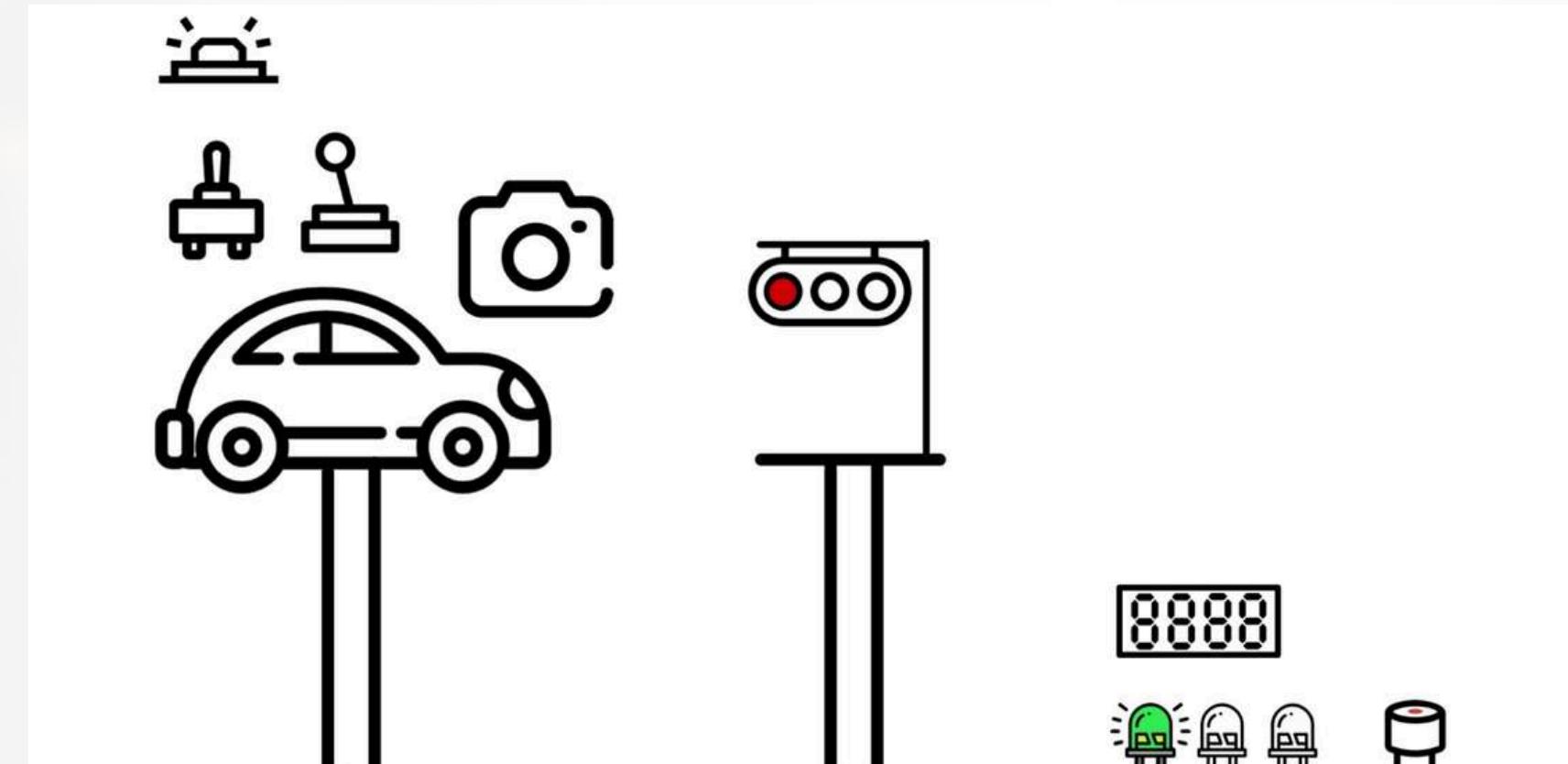
D/R Mode

- 입력
 - 토글 스위치 입력(D or R)
 - 오토홀드 스위치 입력(ON)
 - 조이 스틱(가속 페달)
 - 거리 측정

- 출력
 - 모터 제어
 - 가속 페달 압력 비율 출력
 - 모드 별 LED ON(D or R)
 - 부저 ON

시연 시나리오

오토홀드가 켜져있고 적색등이 감지된 상황에서 가속페달을 밟았을 경우



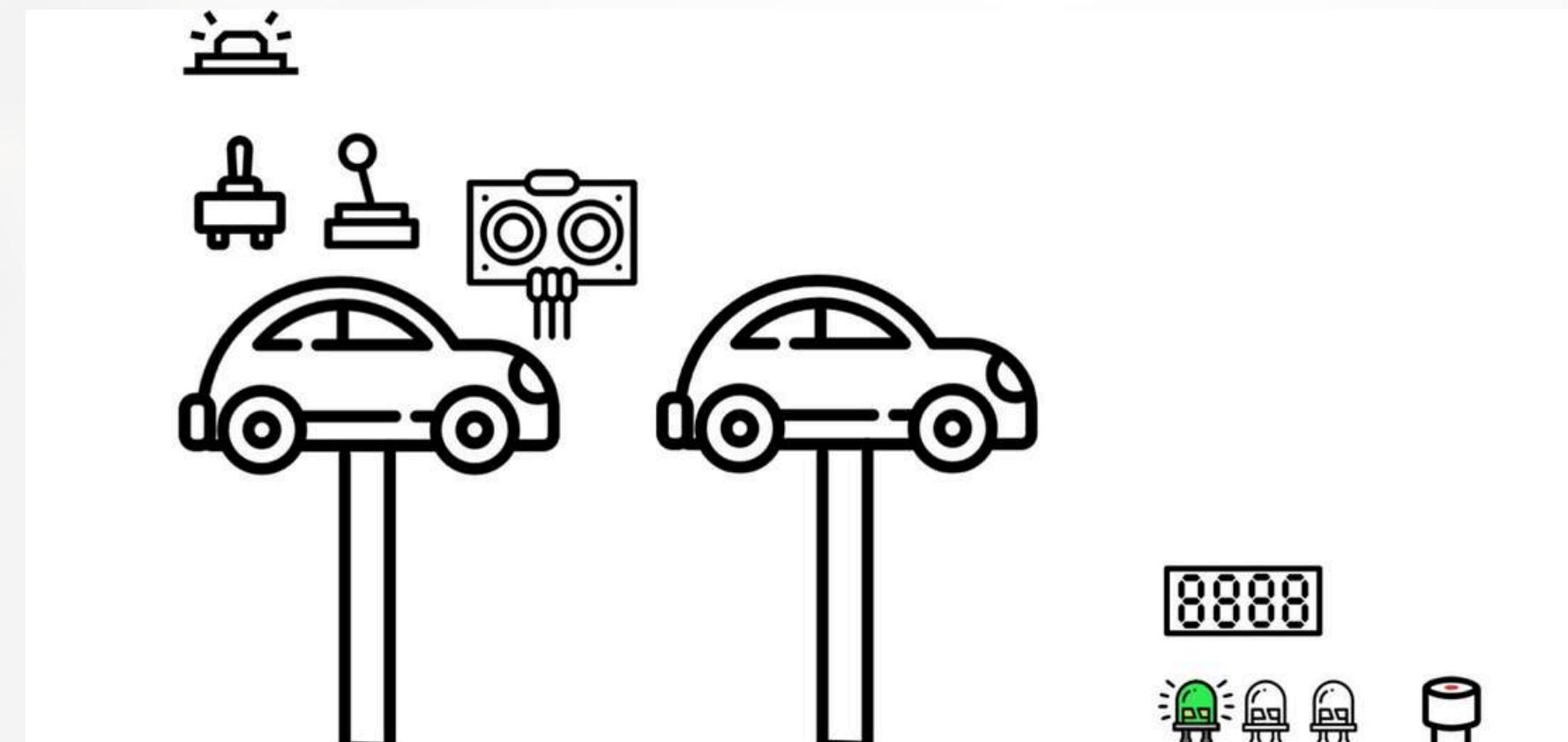
D Mode

- 입력
 - 토클 스위치 입력(D)
 - 오토홀드 스위치 입력(ON)
 - 조이 스틱(가속 페달)
 - 신호등 감지

- 출력
 - 모터 제어
 - 가속 페달 압력 비율 출력
 - 모드 별 LED ON(D)
 - 부저 ON

시연 시나리오

오토홀드가 켜져있고 지정 거리 이내에 차가 감지된 상황에서 가속페달을 밟았을 경우

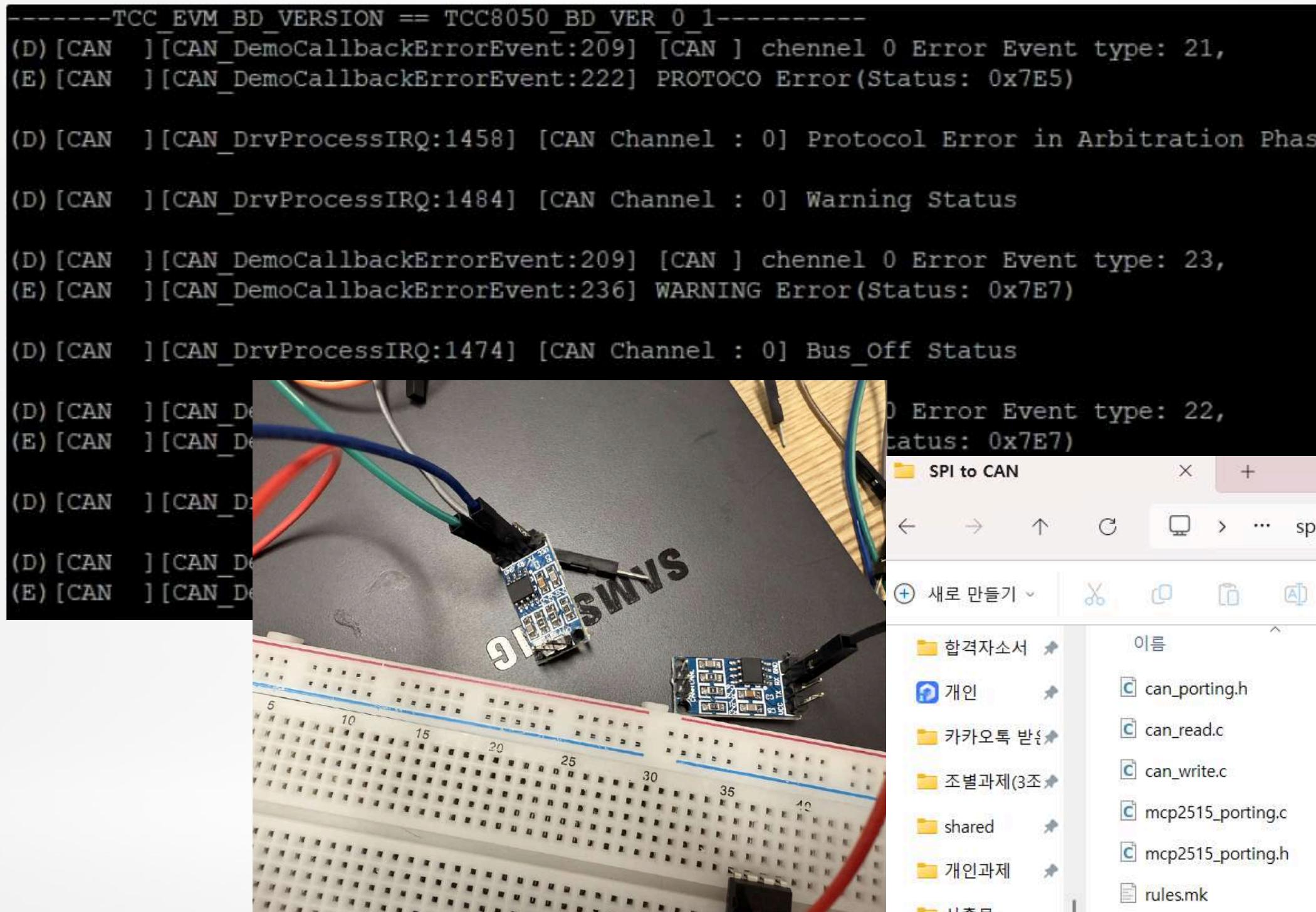


D/R Mode

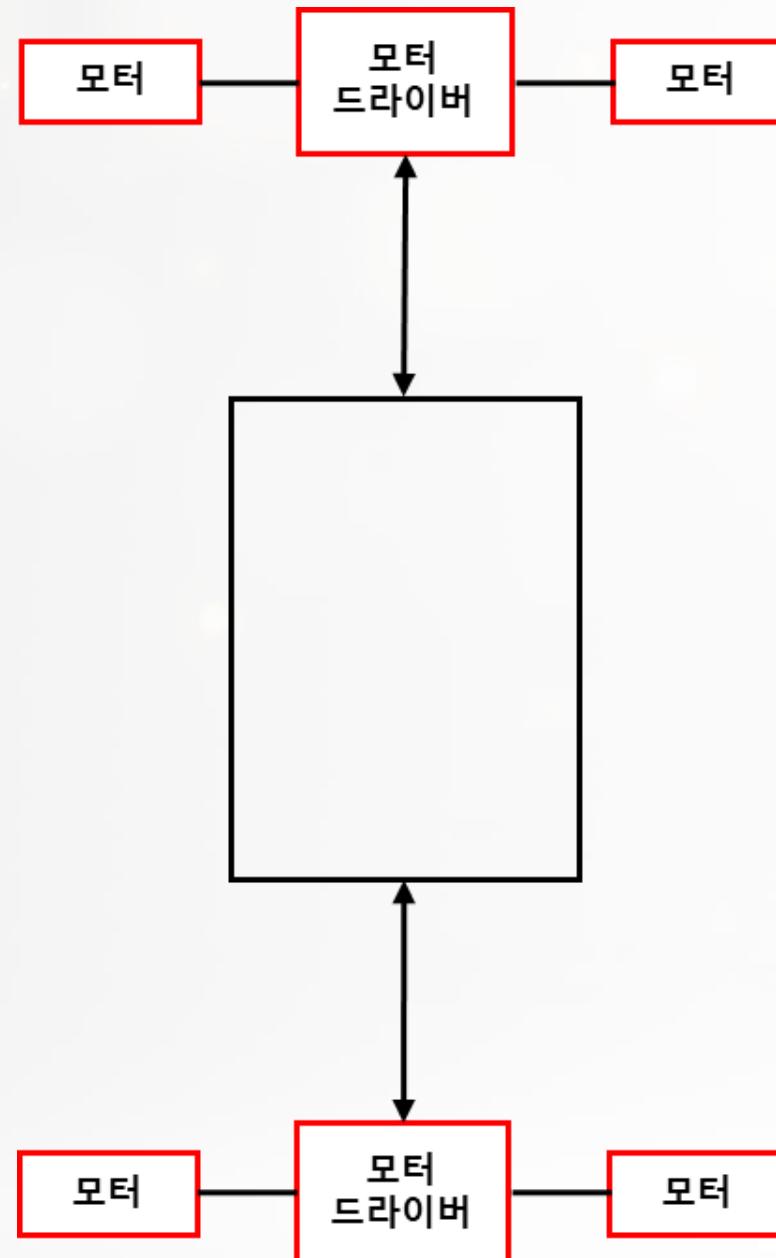
- 입력
 - 토클 스위치 입력(D or R)
 - 오토홀드 스위치 입력(ON)
 - 조이 스틱(가속 페달)
 - 신호등 감지

- 출력
 - 모터 제어
 - 가속 페달 압력 비율 출력
 - 모드 별 LED ON(D or R)
 - 부저 ON

CAN 통신 시도 자료(데모 테스트, TJA1050, MCP2515 모듈 코드 포팅 등)



4륜 구동 방식 선정 이유



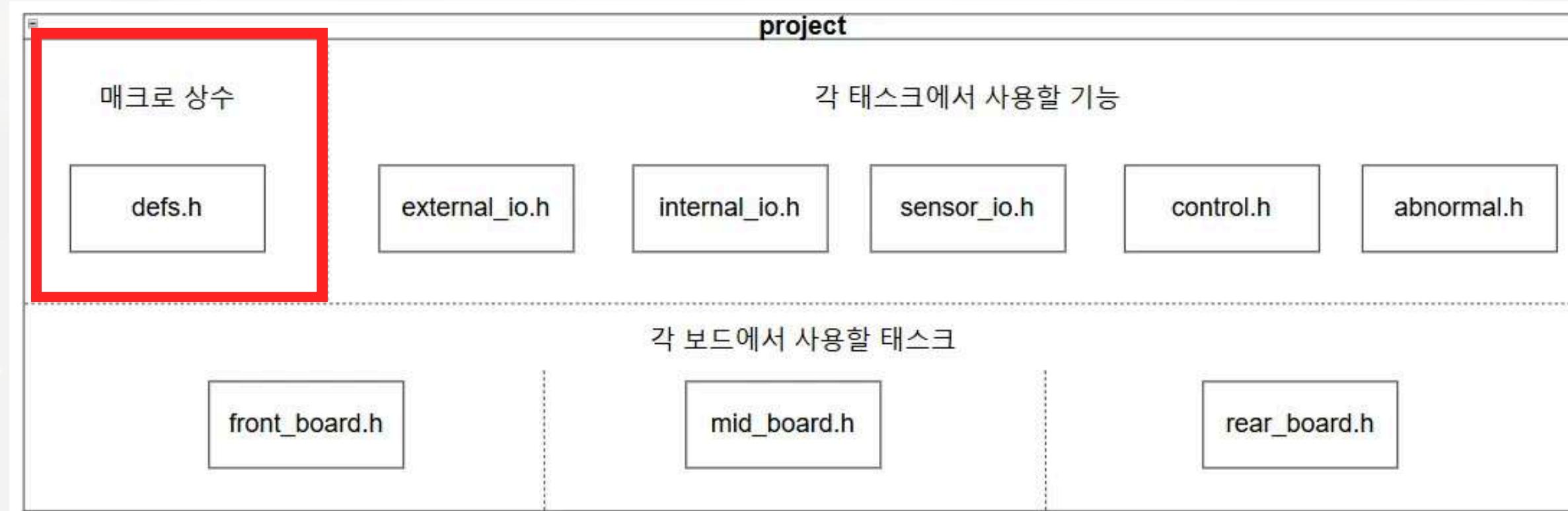
기술적 한계

유압식 브레이크를 구현하는 것에 대한
애로사항이 있음

유압식 브레이크 대체

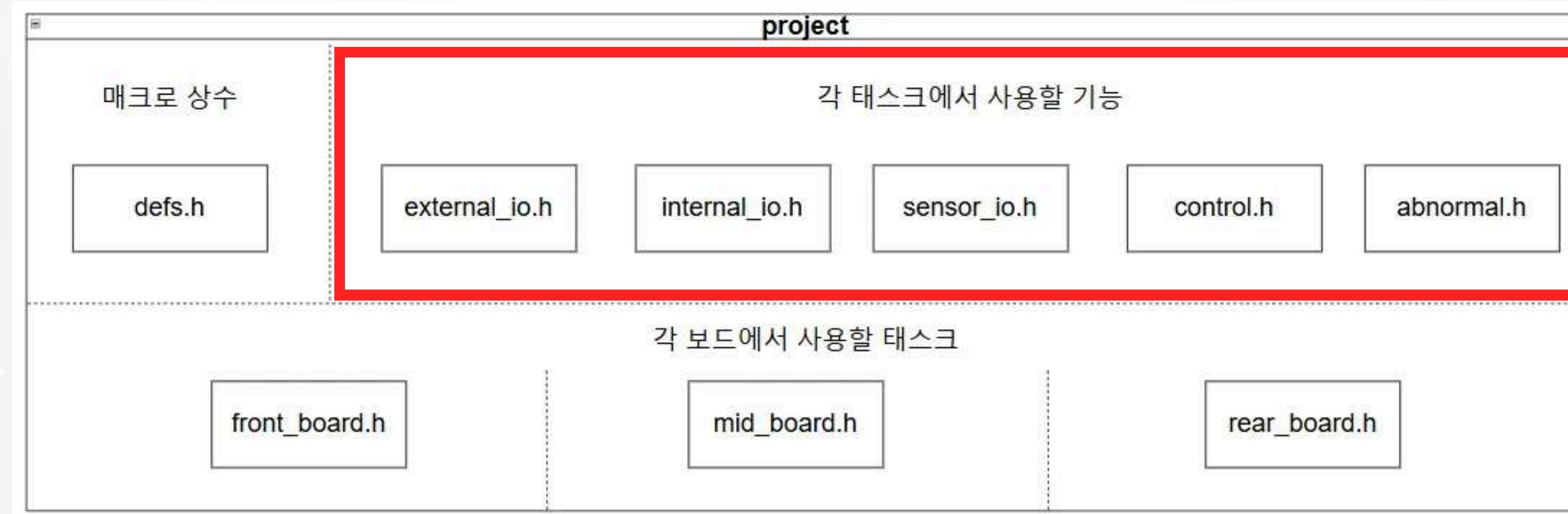
4륜 구동 방식을 채택 함으로써 모터의 속도를 줄여서 모든 바퀴의 구동을 제어 할 수 있음

보드별 빌드 방법



- **defs.h**: 모든 보드에서 사용하는 매크로 상수 정의

보드별 빌드 방법



- **external_io.h**: 보드 간 통신(외부 통신)과 관련된 파일
- **internal_io.h**: 코어 간 통신(내부 통신)과 관련된 파일
- **sensor_io.h**: 센서와 관련된 파일
- **control.h**: 모터 제어와 관련된 파일
- **abnormal.h**: 오조작 판단과 관련된 파일

보드별 빌드 방법



- **front_board.h**: 전방 보드에서 실행할 태스크 관련 파일
- **mid_board.h**: 중앙 보드에서 실행할 태스크 관련 파일
- **rear_board.h**: 후방 보드에서 실행할 태스크 관련 파일

오조작 판단 알고리즘



오토 홀드 OFF



Drive Mode | Reverse Mode



- 정지 후 차량과 전방의 장애물 간의 거리가 7cm 미만일 때, 0.25초 내 가속 페달이 최대로 밟힌 경우



오토 홀드 ON



Drive Mode

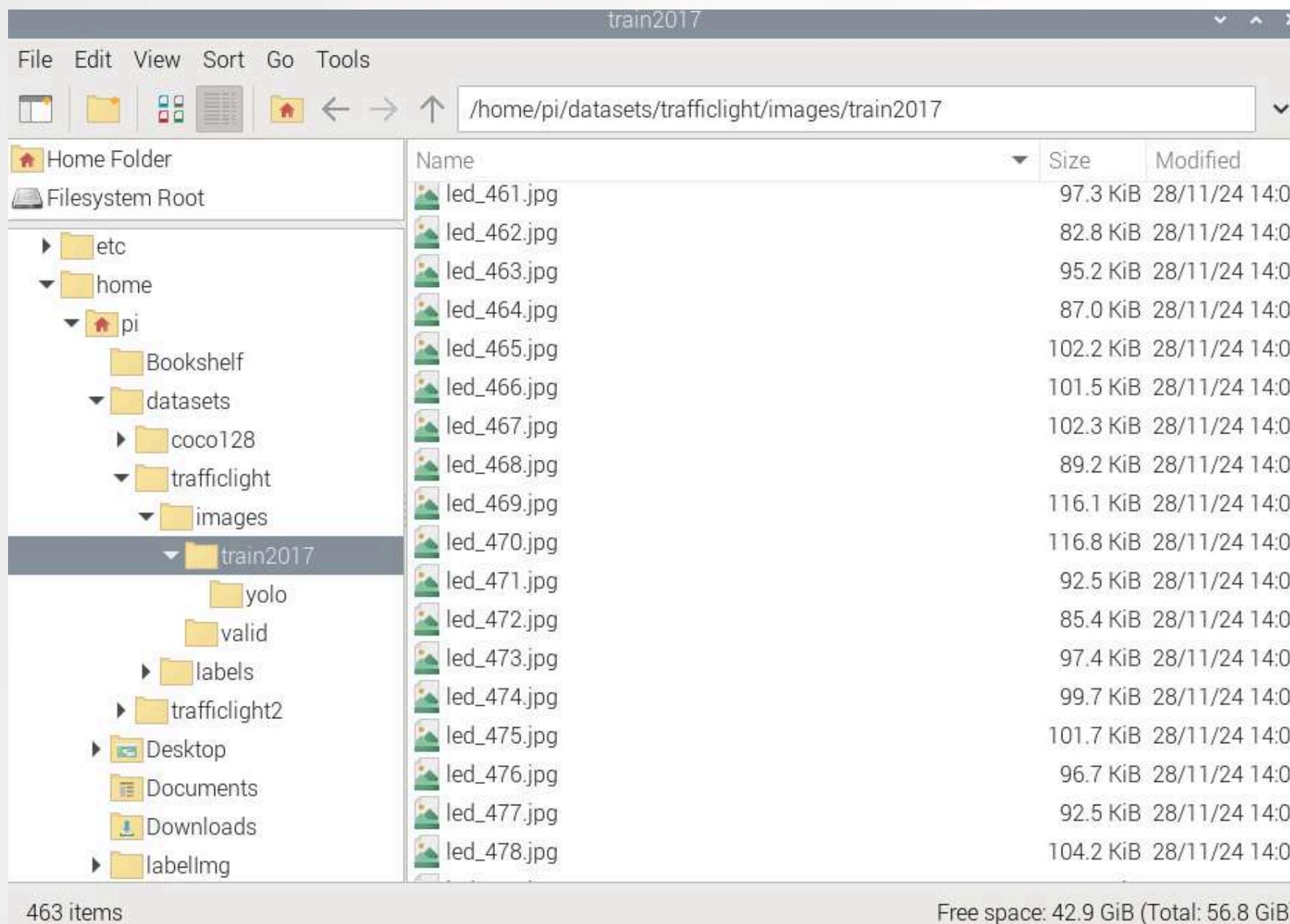
- 정지 후 차량과 전방의 장애물 간의 거리가 7cm 미만이거나 적색 신호 등이 점등되었을 때, 가속 페달이 조금이라도 밟히거나 경우



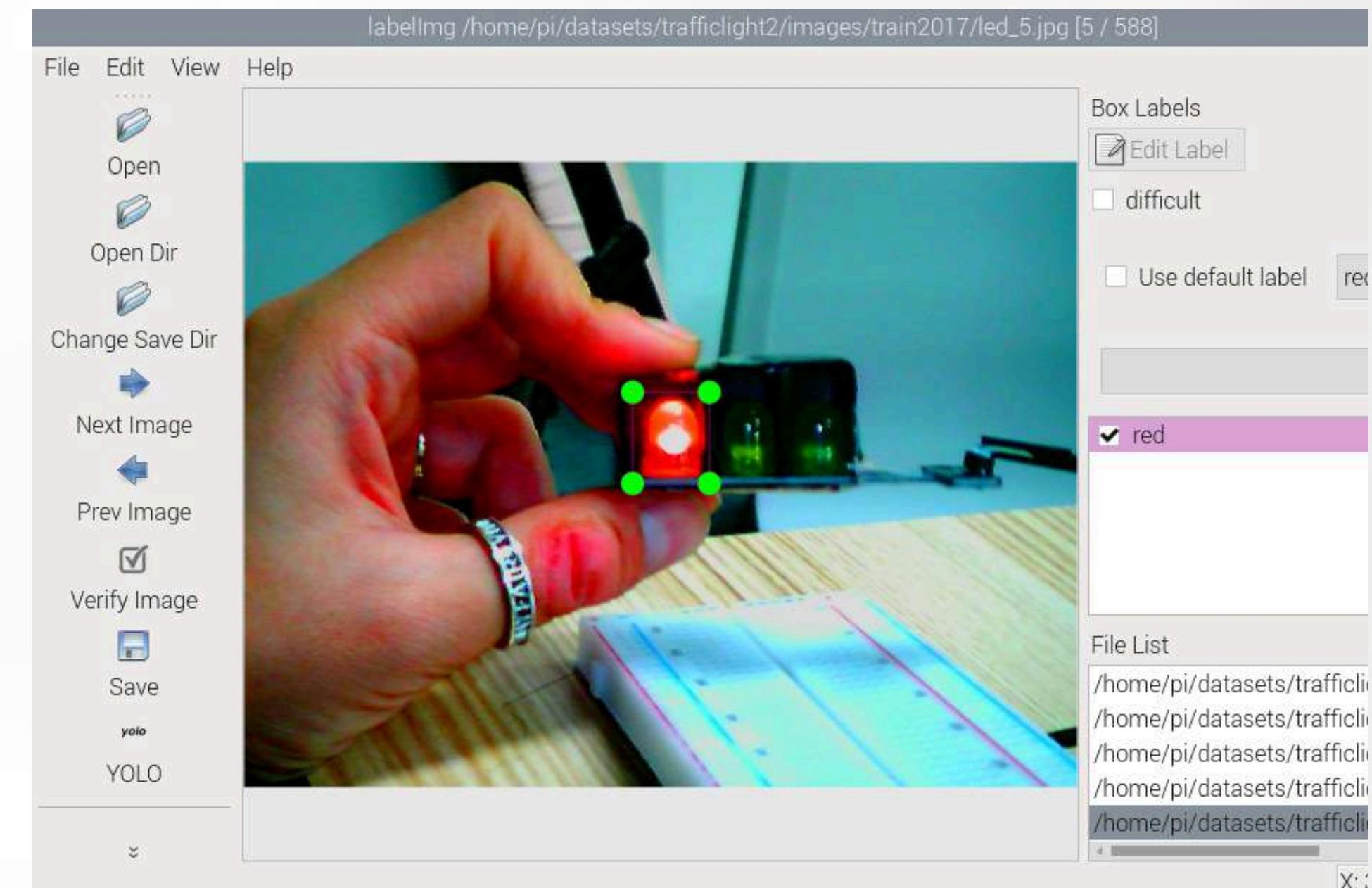
Reverse Mode

- 정지 후 차량과 후방의 장애물 간의 거리가 7cm 미만일 때, 가속 페달이 조금이라도 밟힌 경우

카메라 - yolov5 커스텀 데이터 학습



데이터 수집



데이터 라벨링

카메라 - 핵심 로직

```

# 메인 루프
while True:
    if not result_queue.empty():
        frame, results = result_queue.get()
        detections = results.pandas().xyxy[0] # Pandas DataFrame으로 결과 가져오기

        red_detected_in_frame = False # 현재 프레임에서의 상태 초기화

        for _, row in detections.iterrows():
            x1, y1, x2, y2 = int(row['xmin']), int(row['ymin']), int(row['xmax']), int(row['ymax'])
            conf, cls = row['confidence'], int(row['class'])
            label = f"{classes[cls]} {conf:.2f}"

            if classes[cls] == 'red':
                red_detected_in_frame = True # 빨간 불 감지
                # 바운딩 박스 및 텍스트 표시
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2) # 사각형 빨간색
                cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

        # 최근 상태 업데이트 (노이즈 필터링 적용)
        recent_detections.append(red_detected_in_frame) # 현재 프레임 감지 상태 추가
        if len(recent_detections) > DETECTION_HISTORY: # 최대 히스토리 크기 초과 시 제거
            recent_detections.pop(0)

        # 최근 감지된 True의 개수에 따라 현재 상태 결정 (노이즈 필터링 로직)
        red_detected = recent_detections.count(True) >= DETECTION_THRESHOLD
        # 노이즈로 인해 일시적으로 True/False로 변경되지 않도록 최소 감지 개수를 조건으로 설정

        # 상태가 변경된 경우에만 IPC로 데이터 전송
        if red_detected != previous_red_detected:
            send_to_ipc(fd, red_detected)
            if red_detected:
                print(f"Red light detected. Sending True to IPC.")
            else:
                print(f"No red light detected. Sending False to IPC.")

        # 이전 상태 업데이트
        previous_red_detected = red_detected

```

카메라 성능개선

```
Terminal - root@jammy: ~
Help
  1 user,  load average: 2.31, 1.21, 0.53
  running, 163 sleeping, 1 stopped, 0 zombie
  0.0 ni, 24.5 id, 0.0 wa, 0.0 hi, 0.2 si, 0.0 st
  590.6 free, 744.8 used, 798.4 buff/cache
  0.0 free, 0.0 used. 1151.4 avail Mem

      VIRT   RES   SHR S %CPU %MEM   TIME+ COMMAND
  1828800 529544 168532 R 294.7 24.2 2:07.24 python3
          0     0     0 I  +.0  0.0  0:04.04 kworker/
  248484 105320 51040 S  3.0  4.8  0:23.32 Xorgerv+
          0     0     0 S  0.3  0.0  0:00.01 pvr_dev+
  1198992 94936 73404 S  0.3  4.3  0:03.15 xfwm4
  456868 39604 30472 S  0.3  1.8  0:01.15 panel-8+
  459936 37248 27640 S  0.3  1.7  0:02.13 xfce4-t+

```

yolov5s 모델

```
Terminal - root@jammy: ~
Tabs Help
  1 user,  load average: 2.38, 0.84, 0.35
  running, 163 sleeping, 1 stopped, 0 zombie
  0.0 ni, 32.8 id, 0.0 wa, 0.0 hi, 0.2 si, 0.0 st
  633.3 free, 702.1 used, 798.3 buff/cache
  0.0 free, 0.0 used. 1194.1 avail Mem

      NI      VIRT   RES   SHR S %CPU %MEM   TIME+ COMMAND
      0  1794224 484564 168960 S 255.0 22.2 3:18.84 python3
      0  248484 105320 51040 S  7.8  4.8 0:18.45 Xorgerv+
      0  459936 37232 27640 S  2.3  1.7 0:01.40 xfce4-t+
      20     0     0     0 I  1.6  0.0 0:02.67 kworker/
      0  1198992 94936 73404 S  1.6  4.3 0:02.59 xfwm4
      0     0     0     0 I  0.8  0.0 0:00.10 kworker/

```

yolov5n 모델

단위 테스트

코어 간 통신 테스트

- 보드간 통신 데이터를 임의로 설정하여 코어 간 통신 테스트

```
static uint32 gAccelPercent = 75;      // 액셀 페달 값  
static uint8 gDriveMode = 2;  
static uint8 gAutoholdMode = 1;        // 오토홀드 모드
```

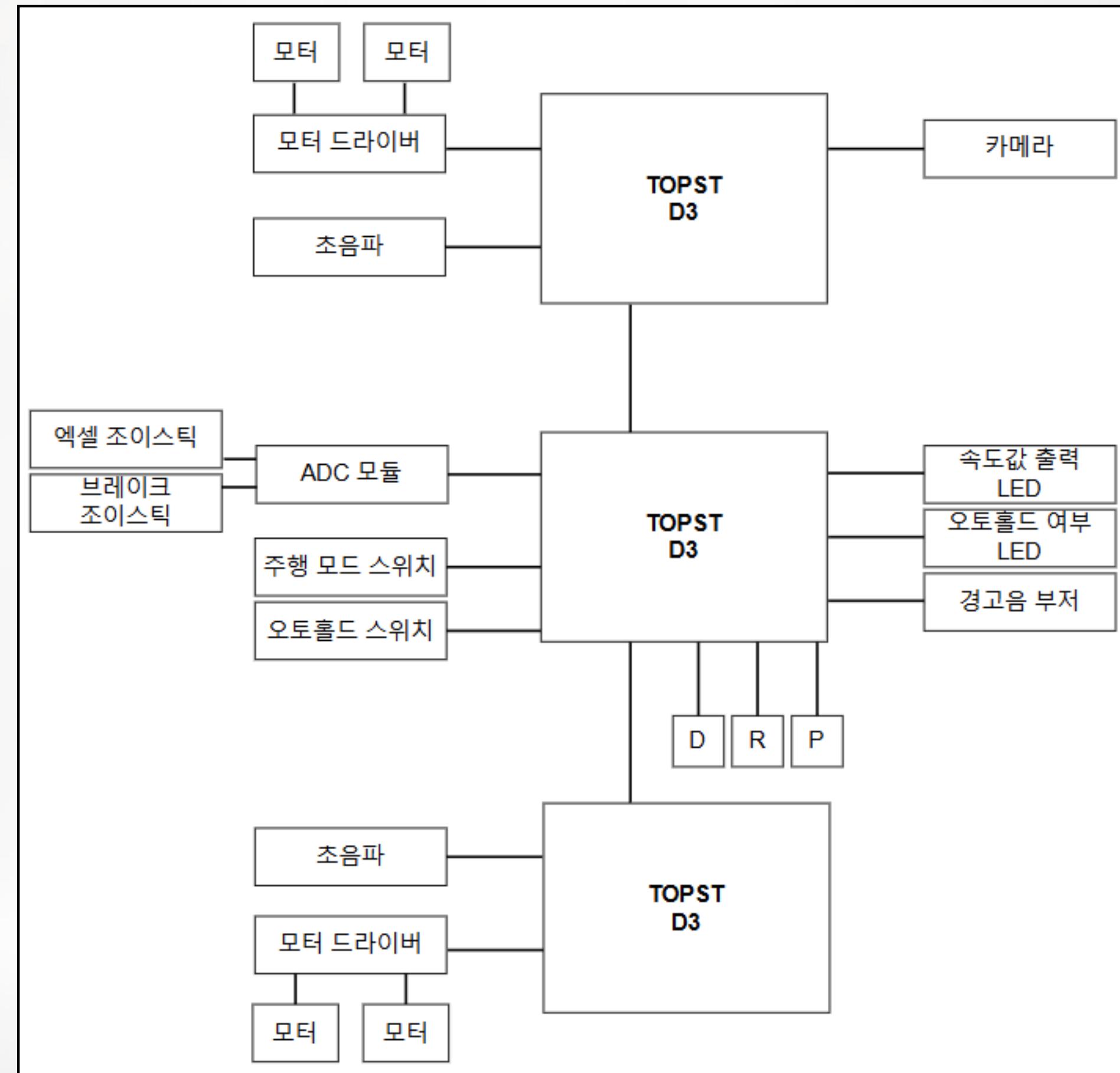
- MICOM → Main Core 데이터 수신 확인

```
[MBOX_TEST] accel_press(int type): 75  
[MBOX_TEST] drive_mode(char type): 2 (0: P mode, 1: D mode, 2: R mode)  
[MBOX_TEST] autohold: 1 (0: off, 1: on)  
[MBOX_TEST] accel_press(int type): 75  
[MBOX_TEST] drive_mode(char type): 2 (0: P mode, 1: D mode, 2: R mode)  
[MBOX_TEST] autohold: 1 (0: off, 1: on)  
[MBOX_TEST] accel_press(int type): 75  
[MBOX_TEST] drive_mode(char type): 2 (0: P mode, 1: D mode, 2: R mode)  
[MBOX_TEST] autohold: 1 (0: off, 1: on)
```

APPENDIX

HW 아키텍처

HW 구조와 필요 모듈 구상



오조작 판단 알고리즘 상세

```
IF (check_start_abnormal()) THEN
    PWM_DUTY = 0                      // 모터의 속도를 제어하는 PWM duty를 0으로 설정
    SEND_TO_EXTERNAL_BOARD(IS_ABNORMAL) // 전후방 보드가 모터를 멈출 수 있도록 데이터 전송
    WARNING_SOUND_ON()                 // 경고음 출력
    ABNORMAL_ACTION = 1                // 오조작 상태 활성화
END IF
```

오조작 상황 발생 여부 확인 수도 코드

```
FUNCTION check_start_abnormal()
    // 오토홀드가 활성화된 상태에서 오조작 발생 여부 확인
    IF (isAutohold == true) THEN
        RETURN check_start_abnormal_with_autohold()
    // 오토홀드가 비활성화된 상태에서 오조작 발생 여부 확인
    ELSE
        RETURN check_start_abnormal_without_autohold()
    END IF
END FUNCTION
```

오토 홀드 스위치가 놀림
&&

차량이 멈춘 상태에서 브레이크를 밟음

오토 홀드 기능 on/off 여부에 따라
오조작 상황 발생 여부 판단

오조작 판단 알고리즘 상세

[오토홀드 OFF] 가속 페달을 세게 밟은 경우 탐지

```

FUNCTION check_start_abnormal_without_autohold()
    // 장애물과 거리가 일정 범위 내에 있을 때, 현재 가속 페달이 최대로 밟혔는지 확인
    IF (isInDangerZone == TRUE && pressAccel >= MAX_PRESS) THEN

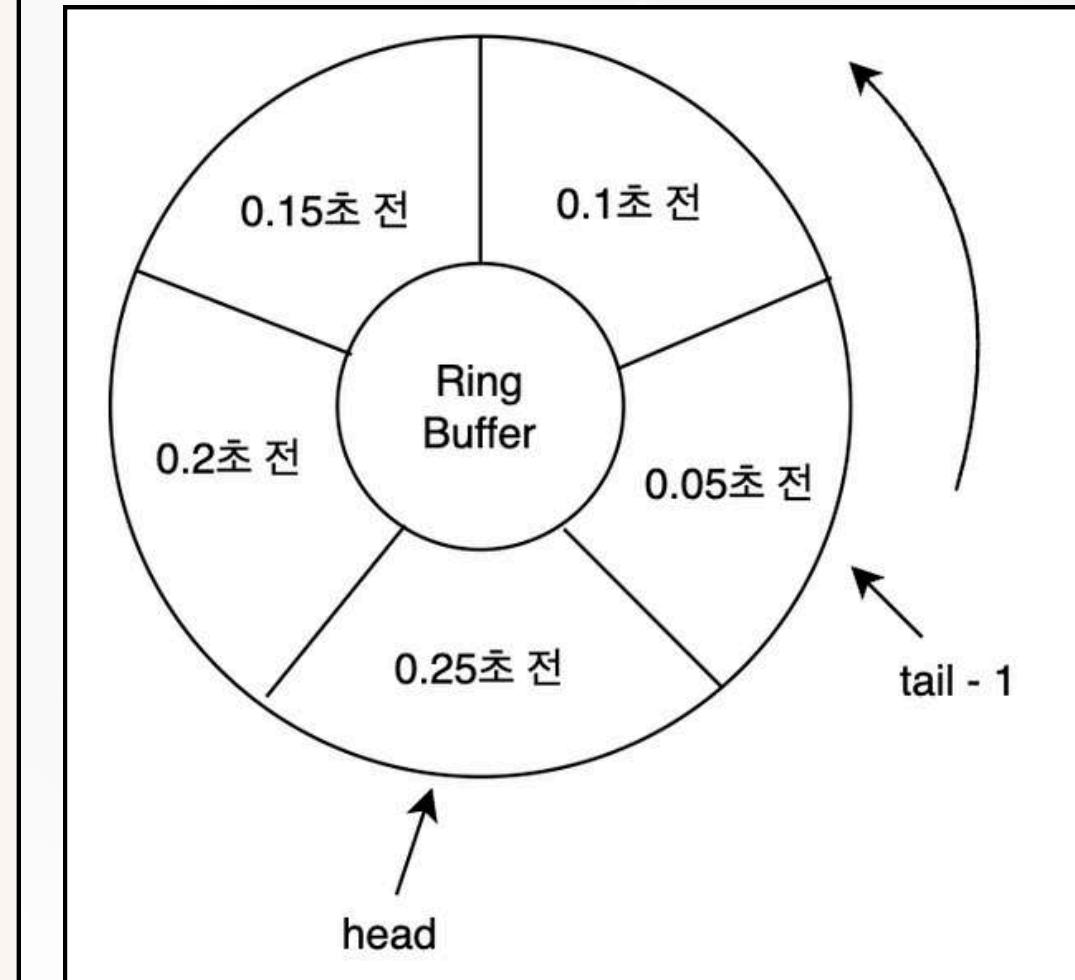
        // 0.25초 내에 차량 속도가 IDLE_SPEED보다 느린 적이 있는지 확인
        IF (is_pwm_below_idle_threshold(IDLE_SPEED)) THEN
            RETURN TRUE
        END IF

        버퍼를 탐색하여 IDLE_SPEED 이하의 값이
        발견되면 TRUE 리턴

    END IF

    RETURN FALSE // 오조작 시작 조건 충족하지 않음
END FUNCTION

```



오조작 판단 알고리즘 상세

[오토홀드 ON] 가속 페달이 살짝이라도 밟힌 경우 탐지

```

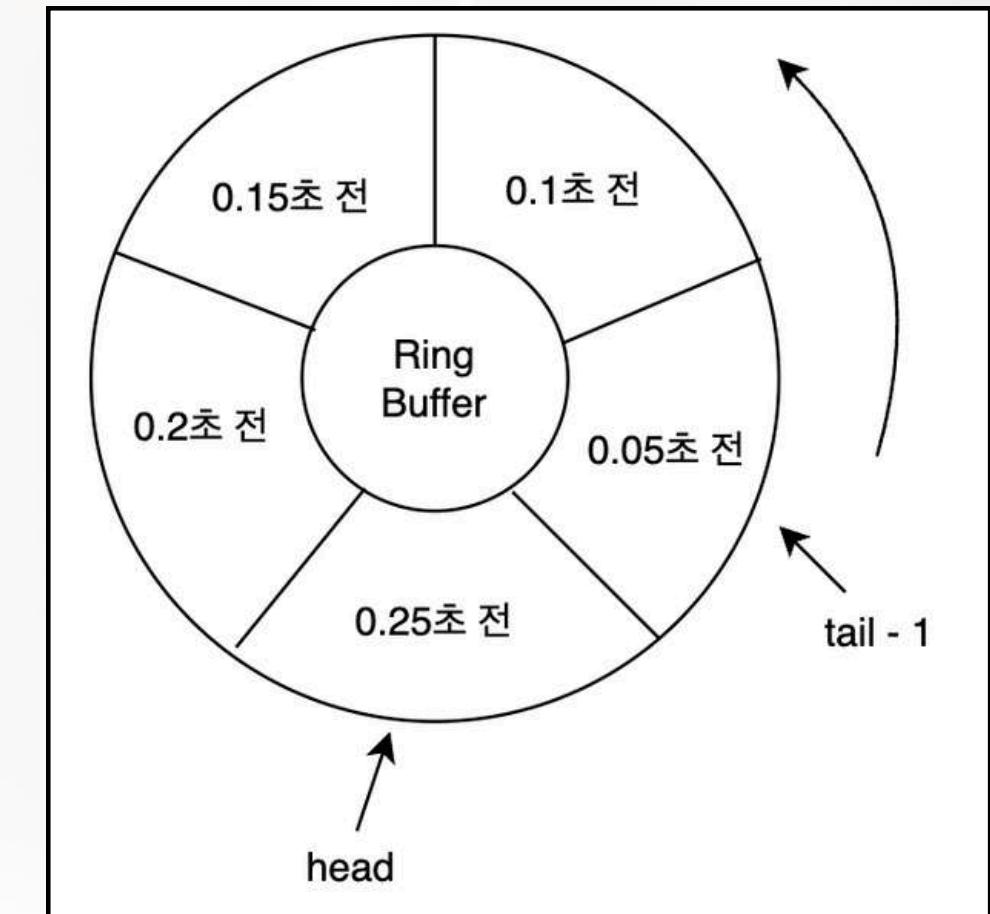
FUNCTION check_start_abnormal_with_autohold()
    // 장애물과 거리가 일정 범위 내에 있거나
    // 주행 모드가 Drive Mode이며, 전방에서 적색 신호등이 점등되어 있을 때
    IF (isInDangerZone == TRUE || (driveMode == D && isTurnRedLight == TRUE)) THEN

        // 0.25초 내에 차량이 정차한 적이 있는지 확인
        IF (is_pwm_below_idle_threshold(0)) THEN
            RETURN TRUE
        END IF
        버퍼를 탐색하여 OOI 발견되면 TRUE 리턴

    END IF

    RETURN FALSE // 오조작 시작 조건 충족하지 않음
END FUNCTION

```



APPENDIX

오조작 판단 알고리즘 상세

```
IF (check_finish_abnormal()) THEN
    WARNING_SOUND_OFF()          // 경고음 출력 종료
    ABNORMAL_ACTION = 0          // 오조작 상태 해제
END IF
```

오조작 상황 종료 여부 확인 수도 코드

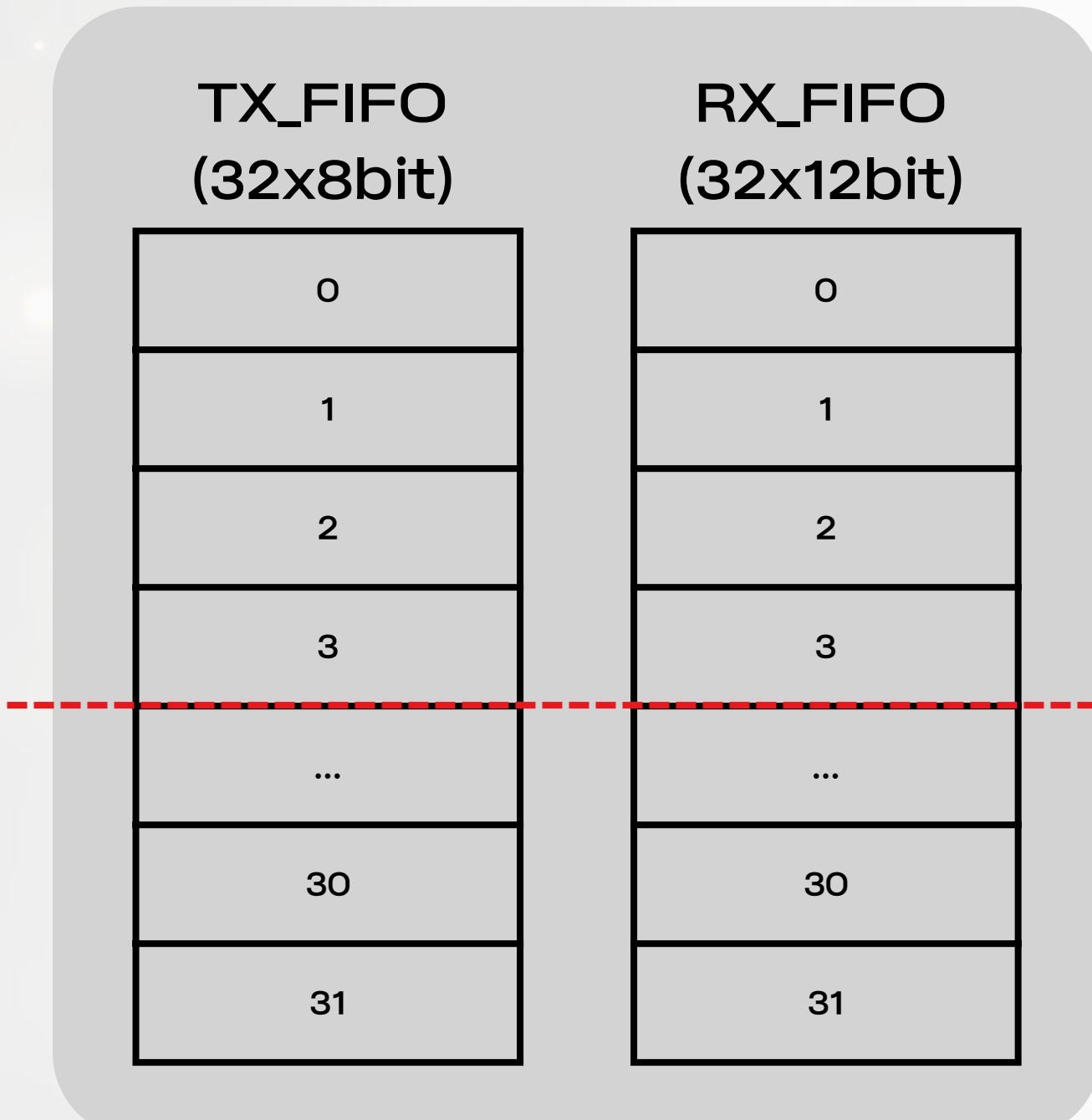
```
FUNCTION check_finish_abnormal()
    // 브레이크(pressBreak)가 최대(MAX_PRESS)로 밟혔을 때 오조작 상황 해제
    IF (pressBreak >= MAX_PRESS) THEN
        RETURN true      // 오조작 상황 종료 조건 충족
    ELSE
        RETURN false     // 오조작 상황 종료 조건 미충족
    END IF
END FUNCTION
```

오조작 상황 발생 후,
브레이크를 세게 밟아야 오조작 상황이 종료

오조작 종료 방법

APPENDIX

보드 간 통신 (UART)



```
// UART Interrupt FIFO Level status register Fields
#define UART_IFLS_RXIFLSEL          (7UL << 3)
#define UART_IFLS_TXIFLSEL          (7UL << 0)

if((ucCh == UART_CH1) || (ucCh == UART_CH2)){
    uint32 ifsl;

    ifsl = UART_RegRead(ucCh, UART_REG_IFLS);
    ifsl &= ~(UART_IFLS_RXIFLSEL|UART_IFLS_TXIFLSEL);
    UART_RegWrite(ucCh, UART_REG_IFLS, ifsl);
}
```

IFLS 레지스터 값

[8] 0012 → [8] 0000

Default

- TXIFLSEL: 0x2
 - 인터럽트 FIFO level:
1/2 full
- RXIFLSEL: 0x2
 - 인터럽트 FIFO level:
1/2 full

설정 값

- TXIFLSEL: 0x0
 - 인터럽트 FIFO level:
1/8 full
- RXIFLSEL: 0x0
 - 인터럽트 FIFO level:
1/8 full

4. 개발 및 설계

구현 기능

입출력 장치 (GPIO) - 7 Segment

```
static void tm1637_delay(int delay)
{
    volatile int temp = delay;

    while(temp > 0)
    {
        --temp;
    }
}
```

```
static void tm1637_write_raw(const char *data)
{
    int i;

    // COMM1(data set) 명령 전송
    tm1637_start();
    tm1637_write_byte(TM1637_COMM1);
    tm1637_stop();

    // COMM2(IP address) 명령 전송
    tm1637_start();
    tm1637_write_byte(TM1637_COMM2);

    // 4개의 데이터 전송
    for (i = 0; i < 4; ++i)
    {
        tm1637_write_byte(data[i]);
    }
    tm1637_stop();

    // COMM3(display control) 명령 전송
    tm1637_start();
    tm1637_write_byte(TM1637_COMM3);
    tm1637_stop();
}
```

```
static void tm1637_write_byte(char data)
{
    int i;
    char ack;

    // 8비트 데이터를 순차적으로 전송
    for (i = 0; i < 8; ++i)
    {
        project_gpio_set(TM1637_CLK_PIN, 0);
        tm1637_delay_us(TM1637_BIT_DELAY);

        project_gpio_set(TM1637_DIO_PIN, data & 0x01);
        tm1637_delay_us(TM1637_BIT_DELAY);

        project_gpio_set(TM1637_CLK_PIN, 1);
        tm1637_delay_us(TM1637_BIT_DELAY);

        data = data >> 1;
    }
}
```

```
project_gpio_set(TM1637_CLK_PIN, 0);
project_gpio_set(TM1637_DIO_PIN, 1);
tm1637_delay_us(TM1637_BIT_DELAY);

project_gpio_set(TM1637_CLK_PIN, 1);
tm1637_delay_us(TM1637_BIT_DELAY);

ack = project_gpio_get(TM1637_DIO_PIN);
if(!ack) {
    project_gpio_set(TM1637_DIO_PIN, 0);
    tm1637_delay_us(TM1637_BIT_DELAY);
}

project_gpio_set(TM1637_CLK_PIN, 0);
tm1637_delay_us(TM1637_BIT_DELAY);
```

```
static void tm1637_start(void)
{
    project_gpio_set(TM1637_DIO_PIN, 0);
    tm1637_delay_us(TM1637_BIT_DELAY);
}

static void tm1637_stop(void)
{
    project_gpio_set(TM1637_DIO_PIN, 0);
    tm1637_delay_us(TM1637_BIT_DELAY);

    project_gpio_set(TM1637_CLK_PIN, 1);
    tm1637_delay_us(TM1637_BIT_DELAY);

    project_gpio_set(TM1637_DIO_PIN, 1);
    tm1637_delay_us(TM1637_BIT_DELAY);
}
```