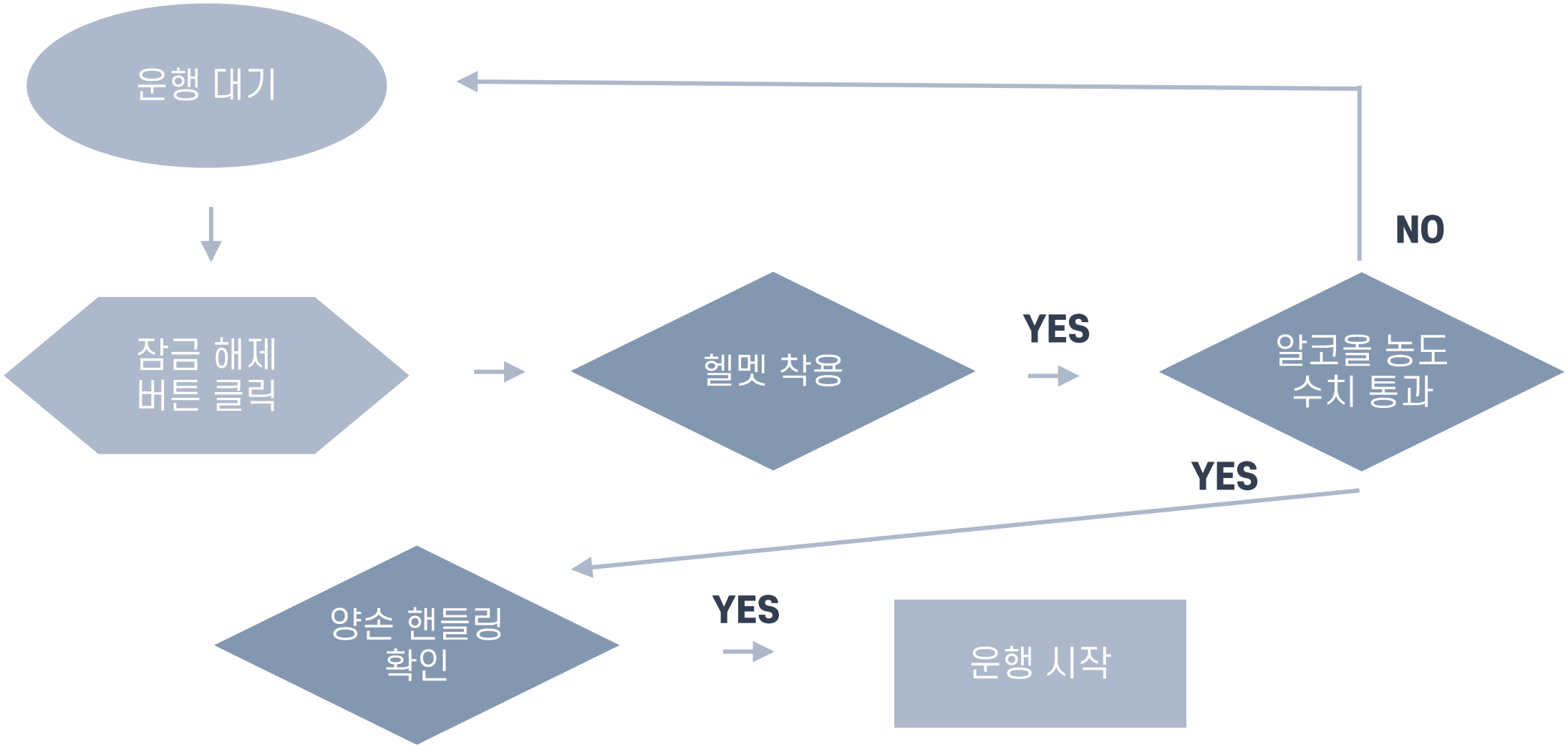
안전한 전동 킥보드

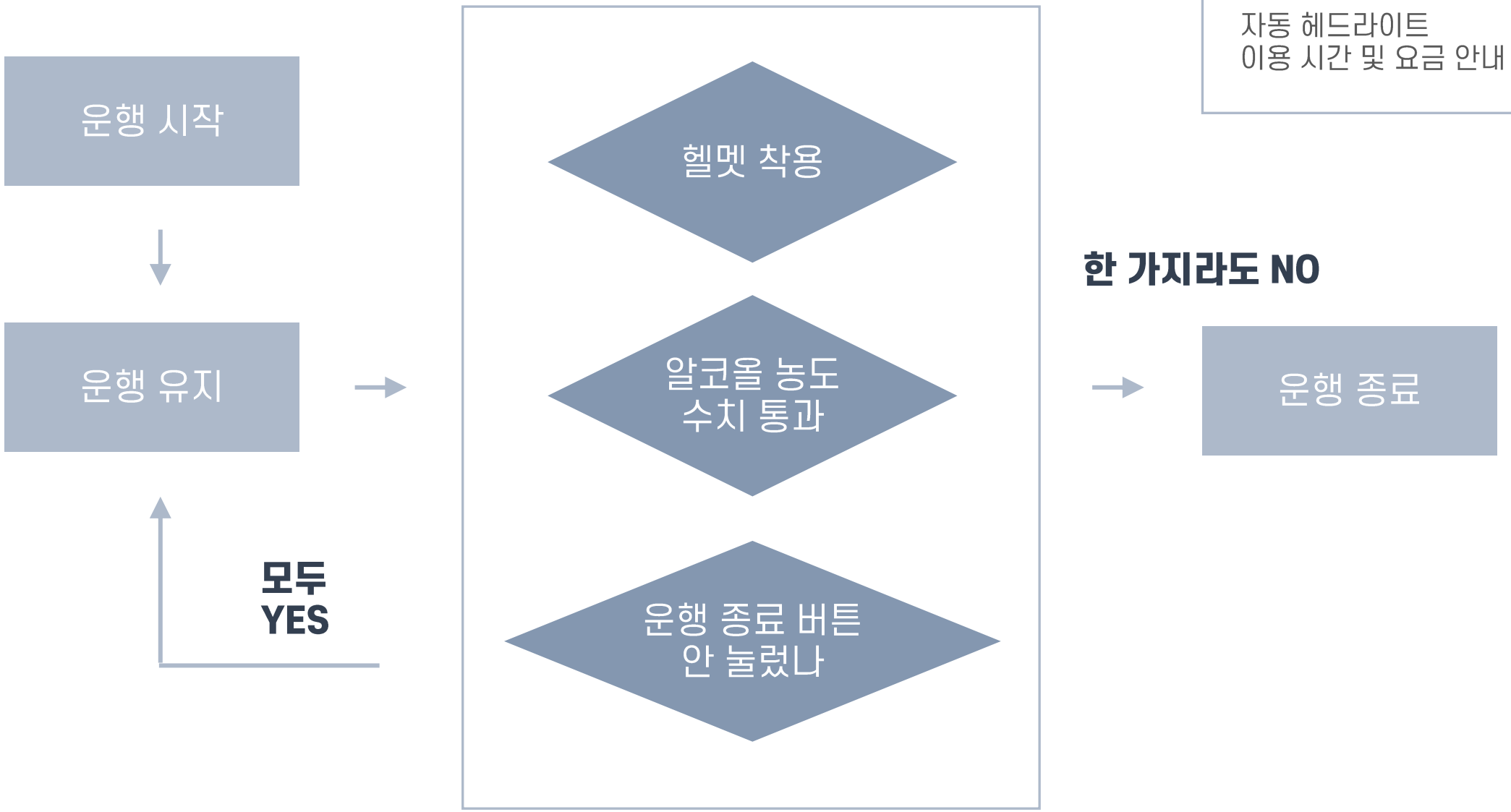
#음주측정 #헬멧필수착용

안하민 조승아 황시은

작품 기능 설명 – BLOCKDIAGRAM



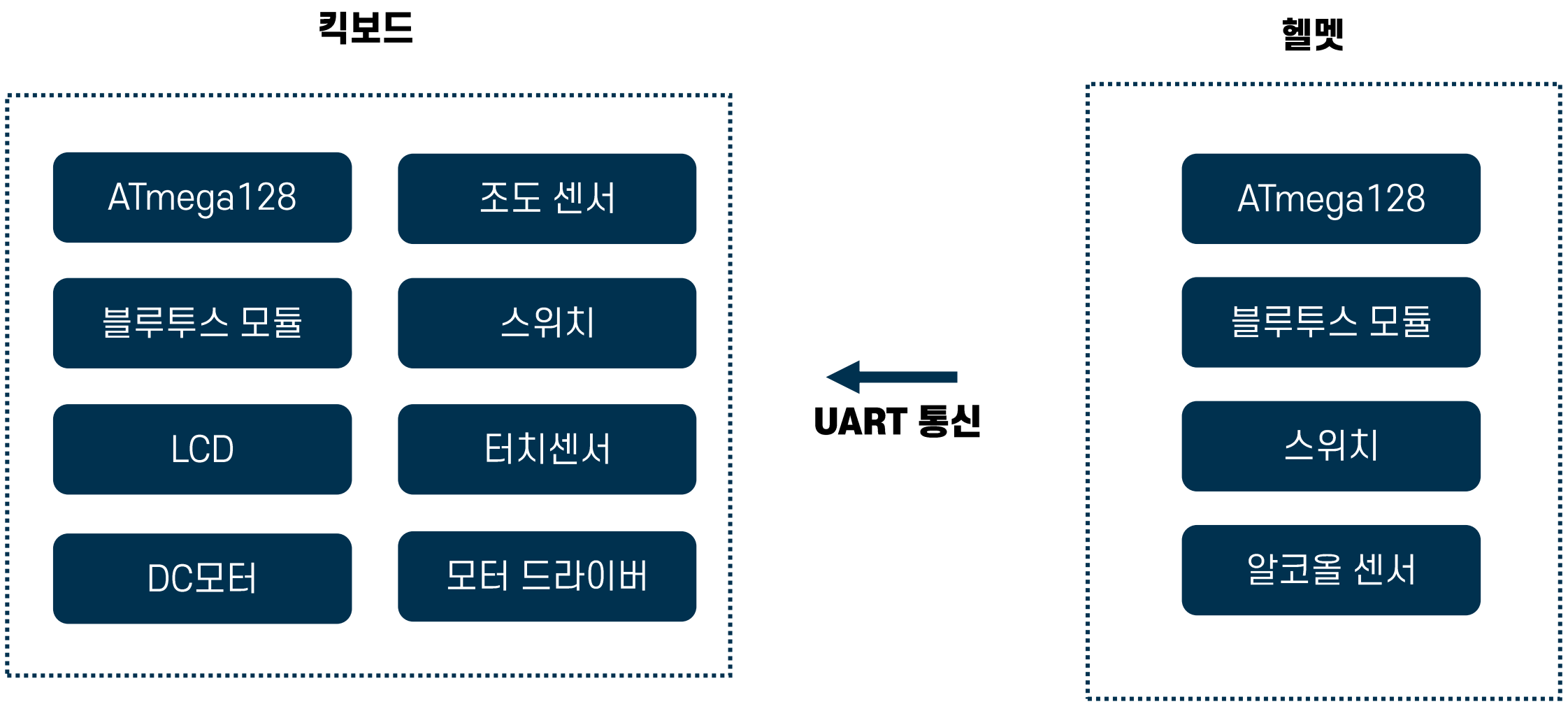
작품 기능 설명 – BLOCKDIAGRAM



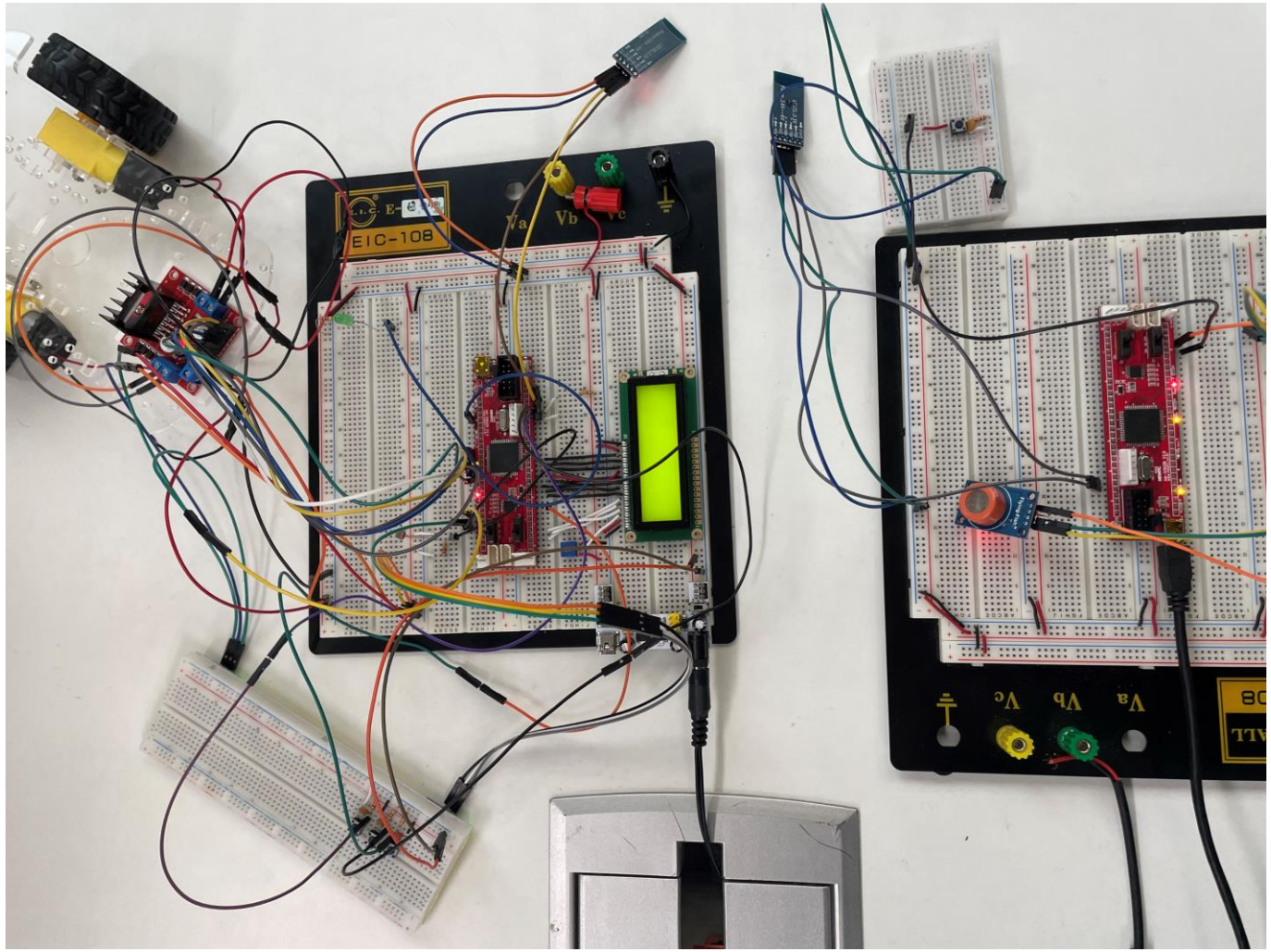
운행 중 부가기능

자동 헤드라이트
이용 시간 및 요금 안내

작품 기능 설명 – 회로 설계

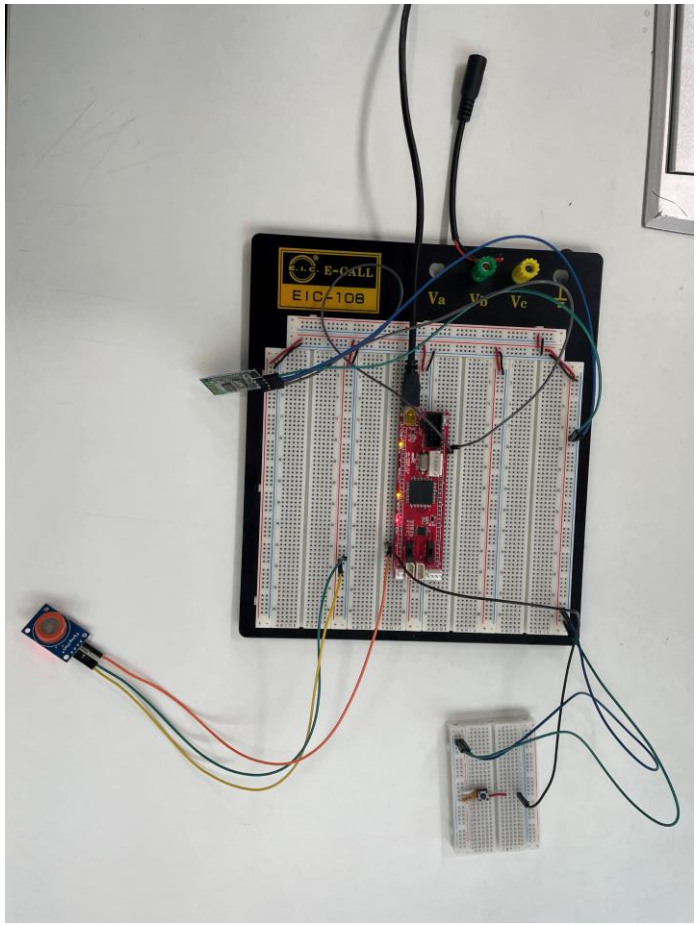

키보드
ATmega128
조도 센서
블루투스 모듈
스위치
LCD
터치센서
DC모터
모터 드라이버
UART 통신
헬멧
ATmega128
블루투스 모듈
스위치
알코올 센서

작품 부분 회로 설명

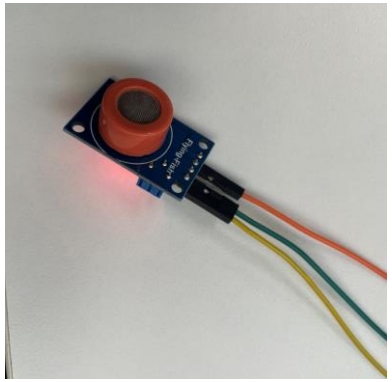


작품 부분 설명

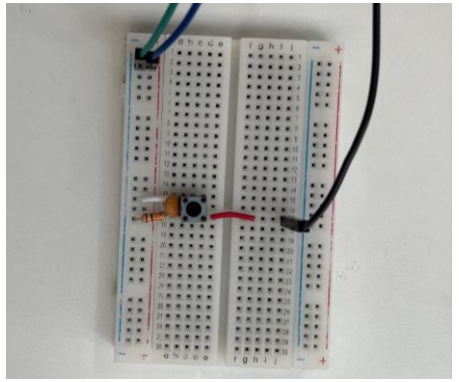
헬멧부 ATMEGA128

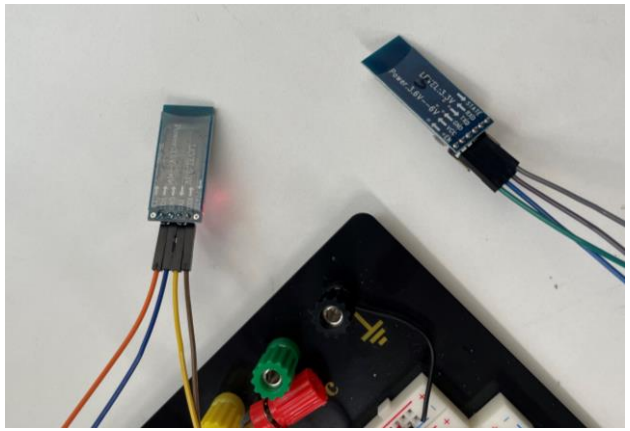


알코올 농도 측정 센서



헬멧에 달린 스위치

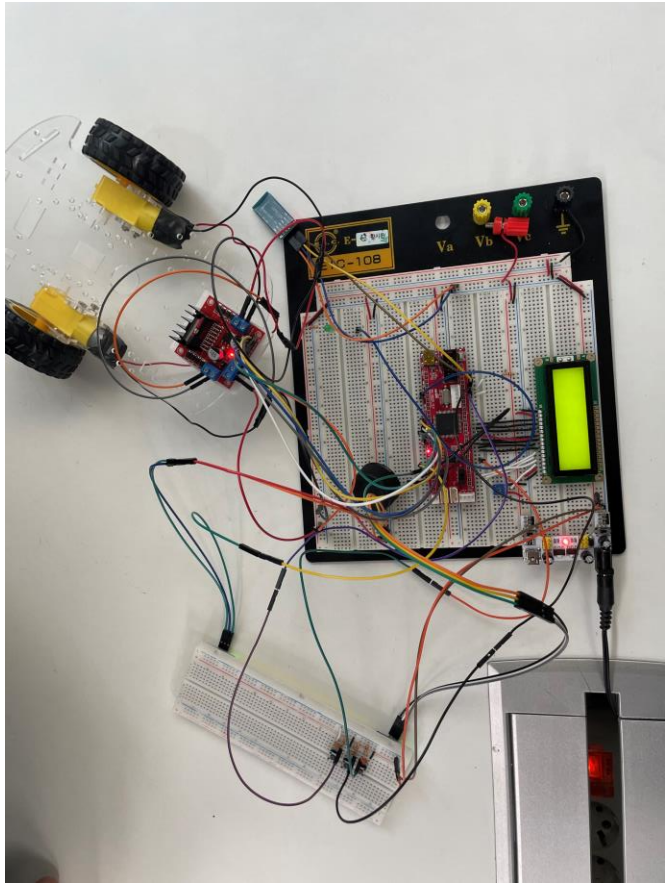




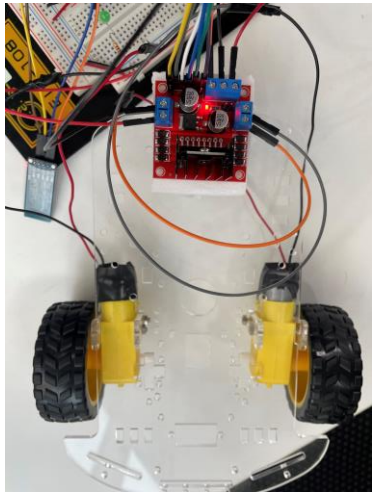
블루투스 모듈 : 헬멧부, 킥보드부

작품 부분 설명

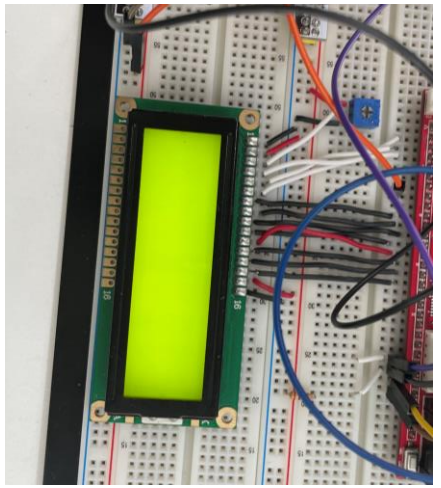
킥보드부 ATMEGA128



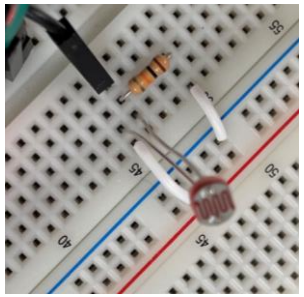
DC모터 (킥보드)

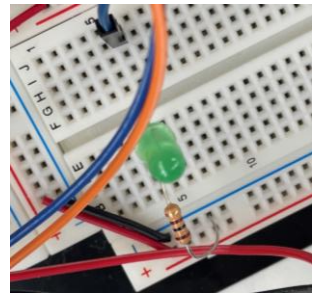


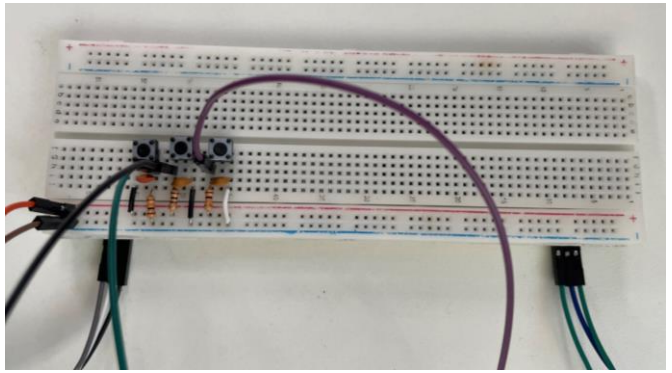
CLCD



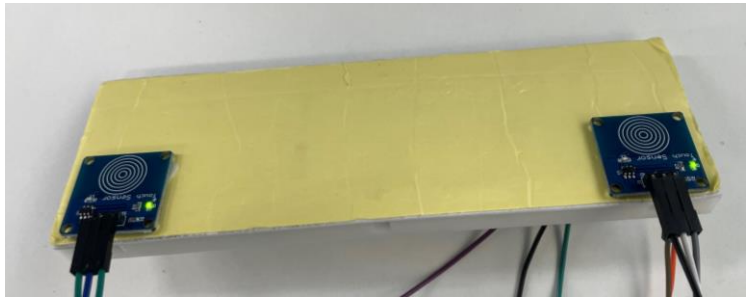
조도 센서 및 LED (헤드라이트)







스위치 1, 2, 3



터치센서

CODE - 헬멧부 ATMEGA128

- main 함수 – 레지스터 설정

```
#include <avr/io.h>
#include <avr/delay.h>
#include <avr/interrupt.h>
#include <util/delay.h>

unsigned int alcohol_Result;
unsigned char alcohol_LOW, alcohol_HIGH;

int main(void) {
    cli();
    DDRA = 0xFF; //A0~A7 출력으로 사용
    DDRC = 0xFF; //C0~C7 출력으로 사용
    DDRD = 0x08; //D3 출력으로 사용
    DDRF = 0x00; //F0~F7 입력으로 사용

    TIMSK = (0 << OCIE2) | (0 << TOIE2) | (0 << TICIE1) | (0 << OCIE1A) | (0 << OCIE1B) | (1 << TOIE1) | (0 << OCIE0) | (0 << TOIE0); //
    TCCR1A = 0x00;
    TCCR1B = 0x05;
    TCNT1H = 0xFF;
    TCNT1L = 0xFE;

    ADMUX = (0 << REFS1) | (1 << REFS0) | (0 << ADLAR) | (0 << MUX4) | (0 << MUX3) | (0 << MUX2) | (0 << MUX1) | (0 << MUX0);  //ADC
    ADCSRA = 0x87;

    UCSR1C = 0x07; //parity x, 8
    UCSR1B = (1 << RXCIE1) | (0 << TXCIE1) | (0 << UDRIE1) | (1 << RXEN1) | (1 << TXEN1) | (0 << UCSZ12) | (0 << RXB81) | (0 << TXB81); //Rx, Tx
    UBRR1L = 0x19; //38400 baud rate

    sei();

    do {

    } while (1);
}
```

CODE - 헬멧부 ATMEGA128

- 타이머 인터럽트 / 킥보드부로 보낼 데이터 계산 / 알코올 농도 수치 측정 함수

```
ISR(TIMER1_OVF_vect) {
    cli();
    TCNT1H = 0xC2;
    TCNT1L = 0xF6;

    while (!(UCSR1A & 0x20)); //데이터가 있을 때(1) 넘어갈 수 있음
    UDR1 = calc_uart(); //킥보드부로 신호 전송
    sei();
}

int calc_uart() {
    int uart;
    int a = measure_alcohol();
    int b = testhelmet();
    uart = (a << 4) | b; // 앞 4bit = 알코올 , 뒤 4bit = 헬멧 정보를 TX로 전송
    return uart;
}

int measure_alcohol() {
    ADMUX = 0x40;
    ADCSRA |= 0x40; // ADC start

    while ((ADCSRA & 0x40)) {} //ADC complete stand by
    alcohol_LOW = ADCL;
    alcohol_HIGH = ADCH;
    alcohol_Result = (alcohol_HIGH << 8) | alcohol_LOW;

    return alcohol_Result / 100;

}
```

CODE - 킥보드부 ATMEGA128

- main 함수 – 레지스터 설정 / LCD 함수 / UART 인터럽트

```
#include <avr/io.h>
#include <avr/delay.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#define ENABLE_PORTE |= 0x01;
#define DISABLE_PORTE &=~0x01;

#define RS_SET_PORTE |= 0x04;
#define RS_CLI_PORTE &=~0x04;

#define RW_SET_PORTE |= 0x02;
#define RW_CLI_PORTE &= ~0x02;

#define MAX_LCD_CNT 7

volatile int usage_time = 0;
volatile int cnt = 0;
unsigned int ADC_Result, inten;
unsigned char ADC_LOW, ADC_HIGH;
unsigned int alcohol[8];
volatile int name_pass[6] = { 'P', 'A', 'S', 'S' };
volatile int name_nonpass[7] = { 'N','O','N','P', 'A', 'S', 'S' };
volatile int fromhelmet;

int main(void) {
    cli();
    DDRA = 0xFF; // RC카 출력
    DDRB = 0xFF; // CLCD 출력
    DDRE = 0x07; // CLCD 출력 + TX 출력
    DDRD = 0x08; // TX 출력
    DDRC = 0xFF; // 조도센서 LED 출력
    DDRF = 0x00; // 터치센서 입력

    EICRA = (0 << ISC31) | (0 << ISC30) | (0 << ISC21) | (0 << ISC20) | (1 << ISC11) | (1 << ISC10) | (1 << ISC01) | (1 << ISC00); //
    EICRB = (0 << ISC71) | (0 << ISC70) | (1 << ISC61) | (1 << ISC60) | (0 << ISC51) | (0 << ISC50) | (0 << ISC41) | (0 << ISC40); //
    EIMSK = (0 << INT7) | (1 << INT6) | (0 << INT5) | (0 << INT4) | (0 << INT3) | (0 << INT2) | (1 << INT1) | (1 << INT0); //

    TIMSK = (0 << OCIE2) | (0 << TOIE2) | (0 << TICIE1) | (0 << OCIE1A) | (0 << OCIE1B) | (1 << TOIE1) | (0 << OCIE0) | (0 << TOIE0); //
    TCCR1A = 0x00;
    TCCR1B = 0x05;

    TCNT1H = 0xFF;
    TCNT1L = 0xFE;

    ADMUX = (0 << REFS1) | (1 << REFS0) | (0 << ADLAR) | (0 << MUX4) | (0 << MUX3) | (0 << MUX2) | (0 << MUX1) | (1 << MUX0); //
    ADCSRA = (1 << ADEN) | (0 << ADSC) | (0 << ADFR) | (0 << ADIF) | (0 << ADIE) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); //
    UCSR1C = 0x07; //parity x, 8bit
    UCSR1B = (1 << RXCIE1) | (0 << TXCIE1) | (0 << UDRIE1) | (1 << RXEN1) | (1 << TXEN1) | (0 << UCSZ12) | (0 << RXB81) | (0 << TXB81); //Rx, Tx
    UBRR1L = 0x67; //16 메가clock , 9600 baud rate
    _delay_ms(100);
    init_LCD();

    sei();
    do {

    } while (1);

}
```

```
/*-------------LCD-------------------------*/
void write_instruction(unsigned char data) {

    RS_CLI
        RW_CLI
        ENABLE
        _delay_us(1);
    PORTB = data;
    _delay_us(1);
    DISABLE
}

void write_data(unsigned char data) {

    RS_SET
        RW_CLI
        ENABLE
        PORTB = data;
    _delay_us(1);
    DISABLE
        _delay_us(1);
}

void init_LCD(void) {

    _delay_ms(75);
    write_instruction(0x30);
    _delay_ms(25);
    write_instruction(0x30);
    _delay_ms(5);
    write_instruction(0x30);
    _delay_ms(5);
    write_instruction(0x3c);
    _delay_ms(5);
    write_instruction(0x08);
    _delay_ms(5);
    write_instruction(0x01);
    _delay_ms(5);
    write_instruction(0x06);
    _delay_ms(5);
    write_instruction(0x0c);
    _delay_ms(5);

}
```

```
void clear(void) {
    write_instruction(0x80 | 0x46); _delay_ms(10);
    write_data(' '); _delay_us(1000);
    write_instruction(0x80 | 0x47); _delay_ms(10);
    write_data(' '); _delay_us(1000);
    write_instruction(0x80 | 0x48); _delay_ms(10);
    write_data(' '); _delay_us(1000);
    write_instruction(0x80 | 0x49); _delay_ms(10);
    write_data(' '); _delay_us(1000);
    write_instruction(0x80 | 0x4A); _delay_ms(10);
    write_data(' '); _delay_us(1000);
    write_instruction(0x80 | 0x4B); _delay_ms(10);
    write_data(' '); _delay_us(1000);
    write_instruction(0x80 | 0x4C); _delay_ms(10);
    write_data(' '); _delay_us(1000);
    write_instruction(0x80 | 0x4D); _delay_ms(10);
    write_data(' '); _delay_us(1000);
    write_instruction(0x80 | 0x4E); _delay_ms(10);
    write_data(' '); _delay_us(1000);
    write_instruction(0x80 | 0x4F); _delay_ms(10);
    write_data(' '); _delay_us(1000);
}
```

```
ISR(SIG_UART1_RECV) { // UART 인터럽트
    cli();

    while (!(UCSR1A & 0x20)); // 1일때만 허용
    fromhelmet = UDR1; // 변수 저장

    sei();
}
```

CODE - 킥보드부 ATMEGA128

- 킥보드 운행 제어 인터럽트

```
ISR(INT0_vect) {
    cli();
    if (cnt == 0) cnt++; // 준비 상태에서 INT0 발생시 cnt 증가 -> 안전검사 진행
    else if ((cnt >= 4) && (cnt < MAX_LCD_CNT)) cnt = 0; // 주행 중 INT0 발생 -> 운행종료
    sei();
}

// lcd
ISR(INT1_vect) { // 주행 중 INT1 발생 시 LCD 화면 변경
    cli();
    if (cnt >= 4) {

        cnt++;
        init_LCD();
        if (cnt == MAX_LCD_CNT) cnt = 4;
    }
    sei();
}
```

```
ISR(TIMER1_OVF_vect) {
    cli();

    TCNT1H = 0xC2; //1sec
    TCNT1L = 0xF6;



    if ((cnt > 3) && (cnt < MAX_LCD_CNT)) { // 운행 중(짐금해제 후)
      usage_time++; // 시간 증가
      jodo(); // 조도센서 함수
      if(fromhelmet >= 0x40){ // 알코올 넘으면 종료
         init_LCD();
           _delay_us(100);
           write_data('1');
         _delay_ms(1000);
         cnt = 0;
         }
      else if(((fromhelmet & 0x01) == 0)){ // 헬멧 벗으면 종료
         init_LCD();
           _delay_us(100);
           write_data('2');
         _delay_ms(1000);
         cnt = 0;
         }
      }

    if (cnt == 0) off(); // 0 : 준비상태, 1~3 : 안전검사, 4~ : 주행
    else if (cnt == 1) testhelmet();
    else if (cnt == 2) testalcohol();
    else if (cnt == 3) testtouch();
    else if (cnt == 4) LCD_usagetime();
    else if (cnt == 5) LCD_jodo();
   else if (cnt == 6) LCD_money();
    _delay_ms(100);

    sei();
}
```

CODE - 킥보드부 ATMEGA128

- 킥보드 운행 제어 함수 (1)

```
void off() {
    usage_time = 0; // 운행 시간 초기화
    init_LCD(); // LCD 초기화
    fromhelmet = 0; // RX 변수 초기화
   RC_stop();// 킥보드 정지
   PORTC=0x00; // 조명 OFF
}

void testhelmet() {
    int name_testhelmet[6] = { 'H', 'E', 'L', 'M', 'E', 'T' };

    RS_CLI;
    RW_CLI;
    ENABLE;
    _delay_us(1);
    init_LCD();
    for (int i = 0; i < 6; i++) {
        _delay_us(100);
        write_data(name_testhelmet[i]);
    }
    _delay_us(100); write_data(' ');

    if ((fromhelmet & 0x01) == 1) { //헬멧 착용하면


        for (int i = 0; i < 4; i++) {
            _delay_us(100);
            write_data(name_pass[i]);
        }

        _delay_ms(1000);
        cnt++; // cnt 증가 -> 알코올검사 진행
        init_LCD();
    }
}
```

```
void testalcohol() {
    int name_alcohol[8] = { 'A', 'L', 'C', 'O', 'H', 'O', 'L', ':' };
    RS_CLI;
    RW_CLI;
    ENABLE;
    _delay_us(1);
    init_LCD();
    for (int i = 0; i < 7; i++) {
        _delay_us(100);
        write_data(name_alcohol[i]);
    }
    _delay_ms(3000);

    write_data(' ');
    if (fromhelmet >= 0x40) { // 알코올 값 기준 이상이면
        for (int i = 0; i < 7; i++) {
            _delay_us(100);
            write_data(name_nonpass[i]); // 논패스
        }
        _delay_ms(1500);
        init_LCD();

        _delay_us(100); write_data('0');
        _delay_us(100); write_data('F');
        _delay_us(100); write_data('F');
        _delay_ms(1500);

        cnt = 0; // 논패스 -> 종료

    }
    else { // 기준 이하 알코올 값이면
        for (int i = 0; i < 4; i++) {
            _delay_us(100);
            write_data(name_pass[i]);
        }
        _delay_ms(1000);
        cnt++;// 다음 단계 -> 양손파지
    }
}
```

CODE - 킥보드부 ATMEGA128

- 킥보드 운행 제어 함수 (2) / 조도 센서 함수

```
void testtouch() {
    init_LCD();
    int name_testhandle[6] = { 'H', 'A', 'N', 'D', 'L', 'E' };

    RS_CLI;
    RW_CLI;
    ENABLE;
    _delay_us(1);

    init_LCD();

    for (int i = 0; i < 6; i++) {
        _delay_us(100);
        write_data(name_testhandle[i]);
    }
    _delay_us(100); write_data(' ');

    while (((PINF & 0x08) == 0) || ((PINF & 0x04) == 0)){}//양손파지확인


    for (int i = 0; i < 4; i++) {
        _delay_us(100);
        write_data(name_pass[i]);
    }
    _delay_ms(1000);

    cnt++; // 다음단계 -> 주행

}
```

```
void jodo(void) {
    ADCSRA |= 0x40; // ADC start

    while ((ADCSRA & 0x40)) {} //ADC complete stand by
    ADC_LOW = ADCL;
    ADC_HIGH = ADCH;
    ADC_Result = (ADC_HIGH << 8) | ADC_LOW;// 조도센서 값
    if (ADC_Result == 0) { inten = 0; }
    else if (ADC_Result < 102) { inten = 1; }
    else if (ADC_Result < 204) { inten = 2; }
    else if (ADC_Result < 306) { inten = 3; }
    else if (ADC_Result < 408) { inten = 4; }
    else if (ADC_Result < 510) { inten = 5; }
    else if (ADC_Result < 612) { inten = 6; }
    else if (ADC_Result < 814) { inten = 7; }
    else if (ADC_Result < 916) { inten = 8; }
    else { inten = 9; } // 조도센서 값 분류

    if (inten <= 5) PORTC = 0xFF; // 조도센서 값이 50이하면 조명 ON
    else PORTC = 0x00;
}
```

CODE - 킥보드부 ATMEGA128

- LCD 출력 함수- 조도, 사용 시간, 요금

```
void LCD_jodo(void) {
    int name_light[6] = { 'L', 'I', 'G', 'H', 'T',':'};
   int name_on[2] = { 'O', 'N'};
   int name_off[3] = { 'O', 'F', 'F'};
    RS_CLI;
    RW_CLI;
    ENABLE;
    _delay_us(1);
    init_LCD();

    write_instruction(0x80 | 0x00); _delay_ms(10);

    for (int i = 0; i < 6; i++) {
        _delay_us(100);
        write_data(name_light[i]);
    }
    _delay_us(100); write_data(inten + '0');
   _delay_us(100); write_data(' ');
   if(inten <= 5){
      for (int i = 0; i < 2; i++) {
           _delay_us(100);
           write_data(name_on[i]);
       }
   }
   else{
      for (int i = 0; i < 3; i++) {
           _delay_us(100);
           write_data(name_off[i]);
       }
   }
    _delay_ms(1000);
}
```

```
void LCD_usagetime(void) { // 사용시간 표시함수
    int name_usagetime[10] = { 'U', 'S', 'A', 'G', 'E', ' ', 'T', 'I', 'M', 'E' };
    int whereIcd[] = { 0x80 | 0x00, 0x80 | 0x01, 0x80 | 0x02, 0x80 | 0x03, 0x80 | 0x04 };
    int usage_time2 = usage_time;
    int usagetime_sec1, usagetime_sec2;
    int usagetime_min1, usagetime_min2;
    int usagetime_hour1, usagetime_hour2;

    usagetime_hour1 = (usage_time2 / 3600) / 10; // 시간
    usagetime_hour2 = (usage_time2 / 3600) % 10;
    usage_time2 = usage_time % 3600;

    usagetime_min1 = (usage_time2 / 60) / 10; //분
    usagetime_min2 = (usage_time2 / 60) % 10;
    usage_time2 = usage_time % 60;

    usagetime_sec1 = usage_time2 / 10; //초
    usagetime_sec2 = usage_time2 % 10;

    RS_CLI;
    RW_CLI;
    ENABLE;
    _delay_us(1);

    write_instruction(0x80 | 0x00); _delay_ms(10);
    for (int i = 0; i < 10; i++) {
        _delay_us(50);
        write_data(name_usagetime[i]);
    }
    _delay_us(50);
    write_instruction(0x80 | 0x40);
    _delay_us(50);
    write_data(usagetime_hour1 + '0'); _delay_us(100);
    write_data(usagetime_hour2 + '0'); _delay_us(100);
    write_data(':'); _delay_us(100);
    write_data(usagetime_min1 + '0'); _delay_us(100);
    write_data(usagetime_min2 + '0'); _delay_us(100);
    write_data(':'); _delay_us(100);
    write_data(usagetime_sec1 + '0'); _delay_us(100);
    write_data(usagetime_sec2 + '0'); _delay_us(100);

}
```

```
void LCD_money(void){
    int money;
    int price[5];
    int name_won[3] = {'₩','O','N'};
    RS_CLI;
     RW_CLI;
     ENABLE;
     _delay_us(1);
     init_LCD();
     write_instruction(0x80 | 0x00); _delay_ms(10);
    money = usage_time*10;
    price[4] = money/10000;
    money %= 10000;
    price[3] = money/1000;
    money %= 1000;
    price[2] = money/100;
    money %= 100;
    price[1] = money/10;
    money %= 10;
    price[0] = money;
    for (int i = 4; i >=0; i--) {
            write_data(price[i]+'0');
          _delay_us(100);
        }
    write_data(' '); _delay_us(100);
    for (int i = 0; i < 3; i++) {
            write_data(name_won[i]);
          _delay_us(100);
        }

}
```

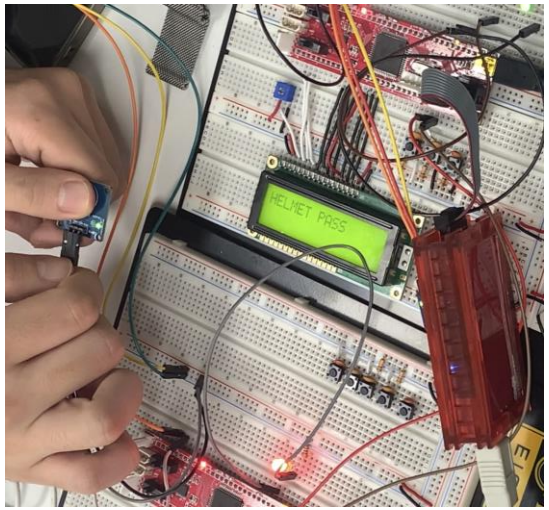
CODE - 킥보드부 ATMEGA128

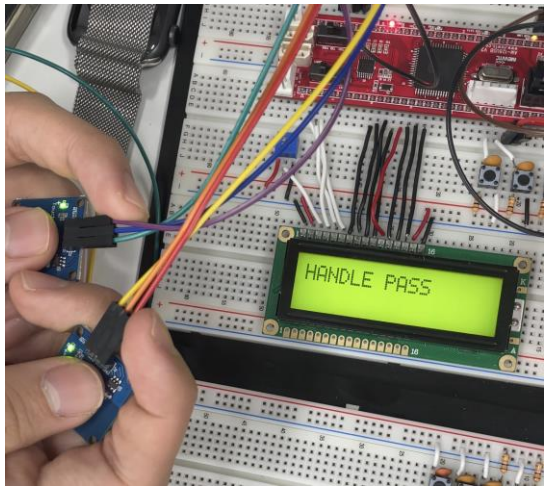
- 킥보드(RC카) 운행 함수

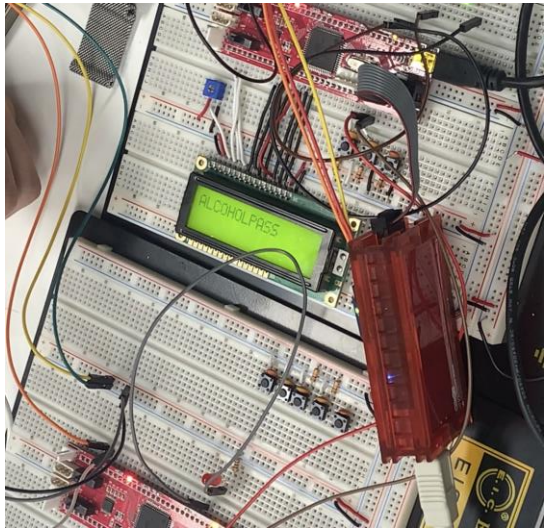
```
void RC_start() {
    PORTA = 0x3A;
}

void RC_stop() {
    PORTA = 0x00;
}
```

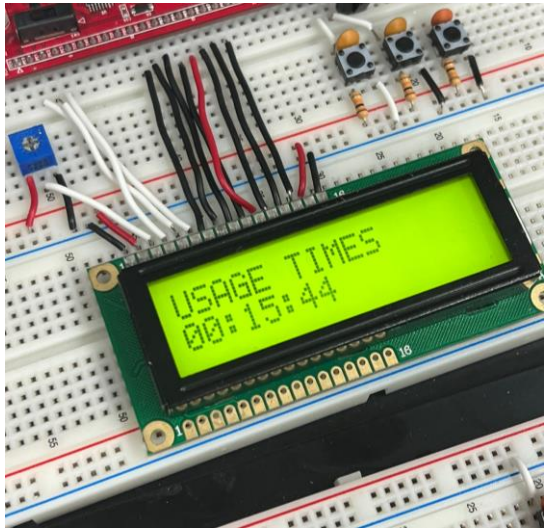
```
ISR(INT6_vect) { // 주행 버튼
    cli();
    if ((cnt > 3) && (cnt < MAX_LCD_CNT)) {
        if (EICRB == 0x30) { //라이징 엣지라면 -> 폴링엣지로
            RC_start();
            EICRB = (0 << ISC71) | (0 << ISC70) | (1 << ISC61) | (0 << ISC60) | (0 << ISC51) | (0 << ISC50) | (0 << ISC41) | (0 << ISC40); //set falling edge
        }

        else if (EICRB == 0x20) {//폴링 엣지라면 -> 라이징 엣지로
            RC_stop();
            EICRB = (0 << ISC71) | (0 << ISC70) | (1 << ISC61) | (1 << ISC60) | (0 << ISC51) | (0 << ISC50) | (0 << ISC41) | (0 << ISC40);
        }
    }
    sei();
}
```

작품 시연









Q&A

감사합니다