마이크로프로세서설계및실습

최종 보고서

차량 후방 감지기

서울시립대학교

컴퓨터과학부

2017920021 류영준

목차

1. 이름 및 학번 소개

이름: 류영준
학번: 2017920021

2. 텀프로젝트 주제 및 기능 소개

(1) 주제

- 차량 후방 감지기
- 저속 주행 및 후진 시에 차량과 장애물 사이의 거리를 실시간으로 감시하는 차량 후방 감지기

(2) 기능

- 장애물과 차량 간의 거리별로 구별된 경고음 알림
  - 거리가 1m 이내면 0.5 초간 "삐~" 경고음 -> 0.3 초간 묵음 반복
  - 거리가 60cm 이내면 0.1 초간 "삐~" 경고음 -> 0.1 초간 묵음 반복
  - 거리가 30cm 이내면 연속적으로 "삐~" 경고음 지속
- 장애물과 차량 간의 거리를 LCD 에 디스플레이 표시

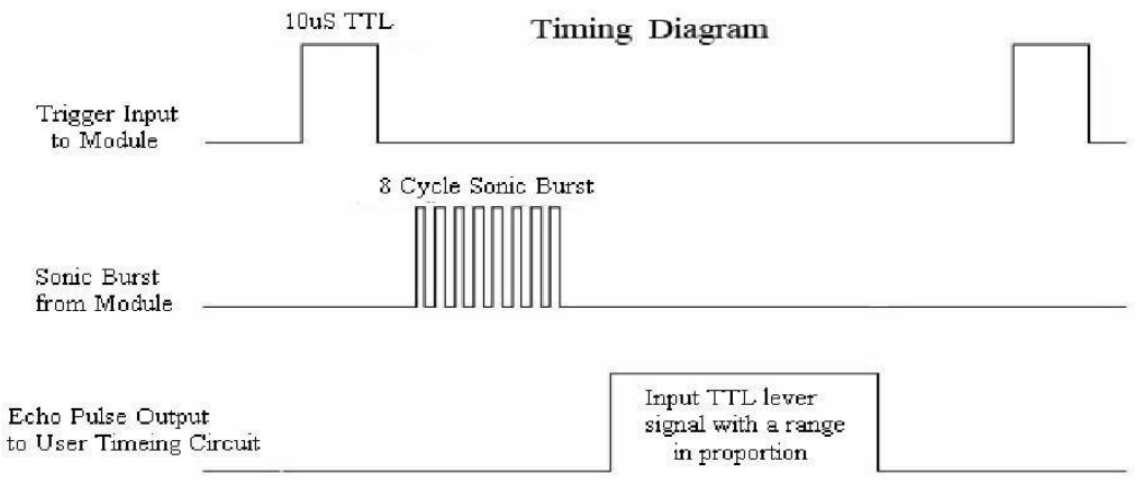
(3) 세부 내용

① HC-SR04 초음파 거리센서 분석
차량과 장애물 사이의 거리를 측정하기 위해서 HC-SR04 초음파 거리 센서를 사용하였다. 기능 구현에 앞서 해당 모듈의 Data Sheet 를 보며 부품의 중요 특징과 작동 조건 그리고 최대 정격을 파악하였다.

거리 센서란 장애물까지의 거리를 측정하는 센서로, 초음파 발신기인(transmitter)과 초음파 수신기(receiver)를 내장하고 있다. 거리를 측정하기 위해 센서는 초음파를 발신하고 장애물에 반사되어 수신된 초음파를 감지한다. 그 후 발신한 시간과 수신된 시간 차이를 측정함으로써 거리를 계산한다.

이 프로젝트에서 사용한 HC-SRO4 기본 규격은 5V DC 전원을 사용하며, VCC 와 GND 포트로 제어한다. 전방 초음파 감지 각도는 좌우 15 도가 최적이고, 최대 좌우 약 30 도까지 측정 가능하다. 거리 측정 범위는 약 2cm ~ 약 4m 까지 가능하다. 외부

인터페이스 신호로는 Trig 와 Echo 를 가진다. Trig 는 초음파 출력을 시작하는 입력 신호이고, Echo 는 초음파가 발사된 직후에 HIGH 로 된 후 초음파가 물체에 부딪혀 반사되서 다시 돌아온 것이 확인되면 LOW 가 되는 출력 신호이다. 즉, Echo 신호의 HIGH 유지 시간은 초음파가 물체까지 왕복하는데 걸리는 시간이 된다. 그러나 여기서 주의할 점은 초음파가 발사된 후 38ms 동안 돌아오지 않는다면 LOW 가 된다는 것이다.



HC-SR04 Data Sheet 에 포함되어있는 Timing Diagram 은 위와 같다. Timing Diagram 을 시간에 따른 해석 결과는 다음과 같다.

1. 모듈의 Trigger Input 에 10us TTL High pulse 를 준다.
2. 40kHz 의 8 개 초음파 cycle sonic burst 가 발생한다.
3. Echo 는 초음파가 발사된 직후 HIGH Level 로 토글되고, 초음파의 반사를 감지하면 LOW Level 로 다시 토글된다. (Echo Pulse 는 초음파가 장애물을 만나고 다시 Echo 로 되돌아올 때까지의 왕복 시간)
4. 거리 = Echo High Pulse 시간 X 소리의 속도(340m/s) / 2 (Echo High Pulse 시간은 왕복 시간이기 때문에 2 로 나누어주어야 차량과 장애물과의 거리가 계산된다.)

② 초음파 거리 센서 및 버저 출력 구현 방법 정의
초음파 거리 센서의 Trig 와 Echo 신호는 Atmega128 의 일반 GPIO 포트와 연결하고, 버저는 JKIT-128-1 키트에 내장된 버저 모듈을 사용하였다.
차량과 장애물과의 거리 측정과 버저 출력 알고리즘 순서도는 다음과 같다.

1. Trig 신호 생성
2. 10us 동안 HIGH 유지

3. Echo 신호가 HIGH 인 것을 감지
4. Echo 신호가 HIGH 가 된다면 타이머/카운터 활성화
5. Echo 신호가 LOW 가 될 때까지 대기
6. Echo 신호가 LOW 가 된다면 타이머/카운터 비활성화
7. 타이머/카운터 값을 읽어 시간과 거리 계산
8. 측정된 거리에 대응되는 버저 신호 출력

③ ATMEGA128 Timer/Counter1 제어

위에서 초음파 센서가 초음파가 발사된 후 38ms 동안 돌아오지 않는다면 LOW 가 된다는 주의점이 있었다. Timer/Counter 는 이에 대하여 38ms 보다 긴 시간동안 측정이 가능하게 하여 측정 실패를 대비해야한다. 따라서 16 비트 Timer/Counter 인 ATMEGA128 의 TCNT1 를 사용하였다. TCCR1B 는 Timer/Counter1 를 제어하는 레지스터 중 하나이다.

측정 시간을 제어하기 위해 프리스케일러값을 `TCCR1B = 0x03;` 로하여 64 분주로 동작하게끔 설정하였다.

④ 차량과 장애물 간의 거리 측정

차량과 장애물 간의 거리 측정 코드는 다음과 같다.

```
distance = (unsigned int)(SOUND_VELOCITY * (TCNT1 * 4 / 2) / 1000);
```

거리를 구하는 공식은 거리 = 속도 X 시간이다. SOUND_VELOCITY 는 소리 속도의 상수값 변수이다. 소리 속도는 340m/s 로 설정하였다. 시간은 64 분주이므로 TCNT1 값이 4us 마다 1 씩 증가한다. 따라서 TCNT1 값에 4 를 곱한 값과 속도를 곱하면 차량과 장애물 간의 왕복 거리가 계산된다. 차량과 장애물 간의 거리를 계산하기 위해 2 로 나누고 마지막으로 1000 으로 나누게 되면, mm 단위의 측정된 거리값을 얻을 수 있다.

⑤ 1602LCD 디스플레이 출력

초음파 센서와 ATMEGA128 를 활용하여 거리 측정과 거리에 대응되는 버저 신호를 출력하였다. 이에 더하여 거리를 1602LCD 모듈에 디스플레이 출력하였다.

출력을 하기에 앞서 1602LCD 모듈을 초기화 하는 작업이 필요하다. 해당 모듈의 Data Sheet 를 확인해보니 아래와 같은 초기화 과정표가 포함되어있었다.

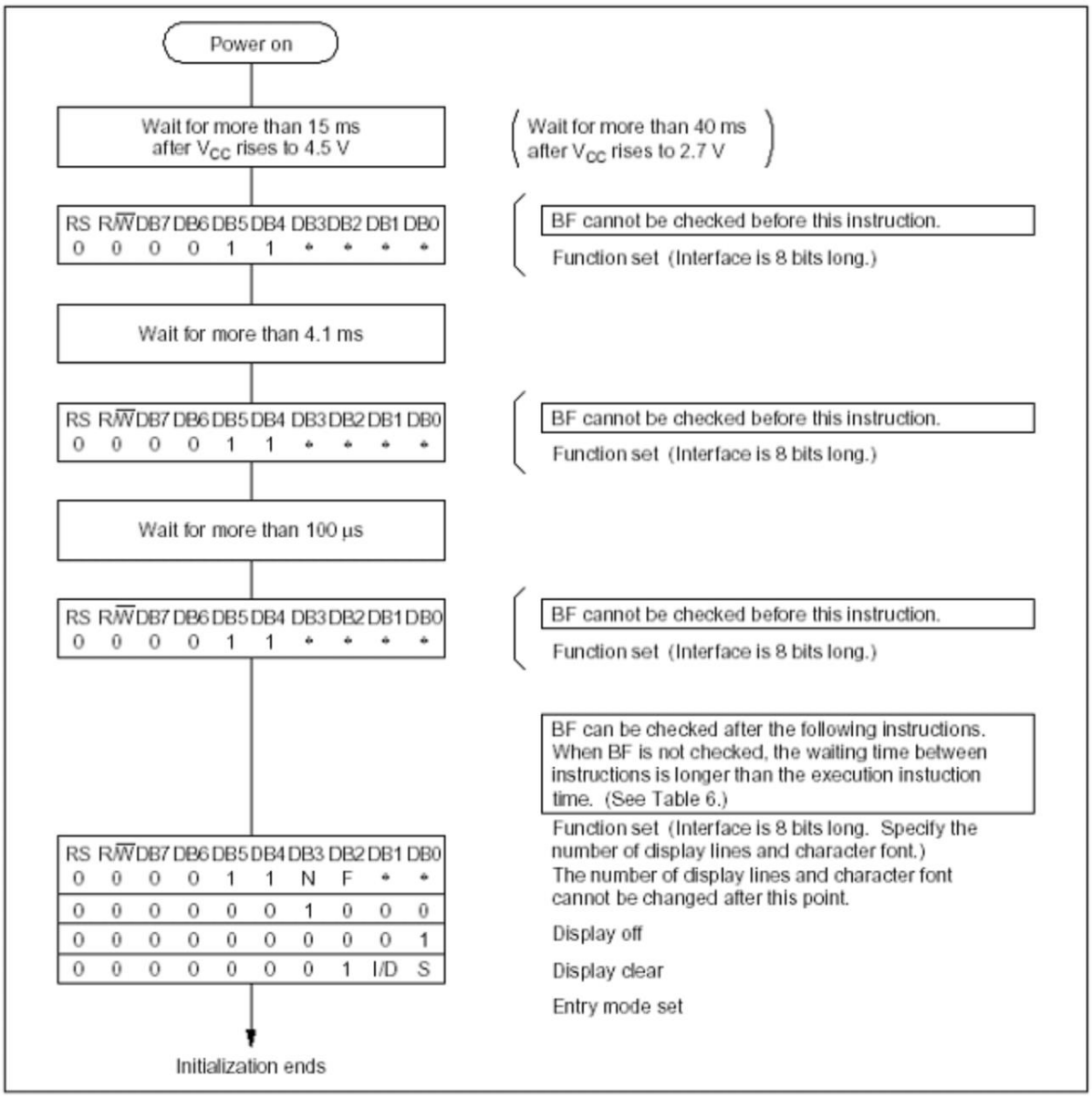


Figure 23 8-Bit Interface

위 Data Sheet 의 초기화 과정을 참고하여 아래와 같은 순서로 코드를 넣어 초기화를 해주었다. 또한 과정마다 지연 시간를 추가하여 초기화 과정이 무리없이 동작하도록 하였다.

```
void LCD_Init(void){
    _delay_ms(100);

    // 비지 플래그를 체크하지 않는 Function Set
    LCD_wCommand(0x38);
    _delay_ms(10);
```

```
    // 비지 플래그를 체크하지 않는 Function Set
    LCD_wCommand(0x38);
    _delay_us(200);

    // 비지 플래그를 체크하지 않는 Function Set
    LCD_wCommand(0x38);
    _delay_us(200);

    // 비지 플래그를 체크하는 Function Set
    LCD_wBCommand(0x38);

    // 비지 플래그를 체크하는 Display On/Off Control
    LCD_wBCommand(0x0c);

    // 비지 플래그를 체크하는 Clear Display
    LCD_wBCommand(0x01);
}
```

위 LCD 를 초기화 하는 함수 뿐만 아니라 LCD 로부터 명령을 읽는 함수, LCD 의 Busy Flag 상태를 확인하는 함수, LCD 에 명령을 출력하는 함수, LCDdp 1Byte 데이터를 출력하는 함수를 하나의 헤더 파일 내에 구성하여 LCD 를 편리하게 제어하였다.

3. 텀프로젝트 진행 내용 요약

11/16 – 텀프로젝트 기획 및 목표 수립
11/17 – 프로젝트 제안서 작성
12/2 – 부품 수령(LCD 모듈 제외)
12/5 – LCD 모듈 부품 수령
12/7 – 회로도 작성
12/8 – 초음파 센서 연결 및 구현
12/9 – 거리에 따른 부저 제어 구현, 가변 저항 구매, 회로도 수정
12/13 – 가변 저항 연결, LCD 연결 및 구현

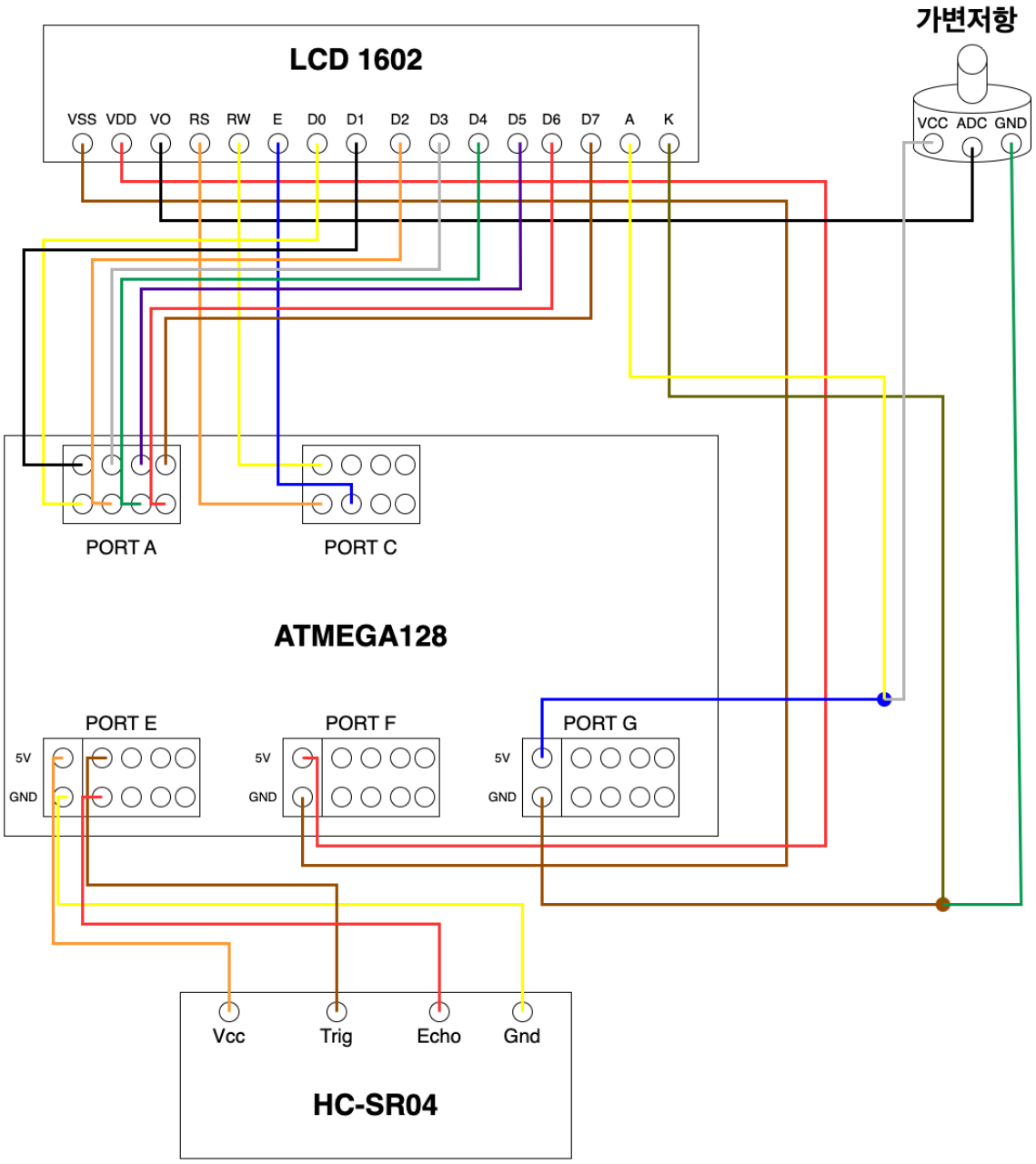
12/14 – 최종 구현 완료 및 테스트
12/19 – 최종 보고서 작성

4. 추가 사용 부품 내역 및 회로도

(1) 추가 사용 부품 내역

상품코드	수량	상품명	단가	부가세 포함단가	금액	소계
1076851	2	[OEM] 초음파 거리센서 모듈 HC-SR04 [SZH-EK004]	₩1,300	₩1,430	₩2,860	
1327456	1	[SMG] 아두이노 I2C 1602 LCD 모듈 [SZH-EK101]	₩3,600	₩3,960	₩3,960	
1322408	1	[SMG] 브레드보드 830 핀 MB-102 [SZH-BBAD-002]	₩1,500	₩1,650	₩1,650	
1328409	1	[KEYES] 테스트[CH254]소켓 점퍼 케이블 40P (칼라) (M/M) 10cm	₩700	₩770	₩770	
						₩9,240

(2) 회로도



5. 소스코드

<main.c> 소스코드

```
#define F_CPU 16000000UL
#include <avr/io.h> // ATmega128 의 레지스터 등이 정의되어 있음
#include <util/delay.h> // _delay_ms() 함수 등이 정의되어 있음

#include <stdio.h>
```

```c
#include <string.h>

#define TRIG 6      //Trigger 신호 (출력 = PE6)
#define ECHO 7      //Echo 신호 (입력 = PE7)
#define SOUND_VELOCITY 340UL //소리 속도 (m/sec)

#include "lcd.h"

void display_distance_lcd(unsigned int distance);
void display_ok_lcd();

// C 언어의 주 실행 함수
int main(void){

    DDRA = 0B11111111; // 포트 A 의 방향 설정, 0 : 입력, 1 : 출력
    DDRC = 0B11111111; // 포트 C 의 방향 설정, 0 : 입력, 1 : 출력

    LCD_Init(); // 텍스트 LCD 초기화 - 함수 호출

    unsigned int distance; // 거리 변수
    int i;

    DDRB = 0x10; // 버저 출력
    DDRE = ((DDRE|(1<<TRIG)) & ~(1<<ECHO)); // TRIG = 출력 , ECHO = 입력
setting

    while (1) {
        TCCR1B = 0x03; // Timer/Counter1 클록 4us(64 분주)
        PORTE &= ~(1 << TRIG); // Trig=LOW 상태
        _delay_us(10); // 10us 동안 유지

        PORTE |= (1 << TRIG); // Trig=HIGH -> 거리 측정 명령 시작
        _delay_us(10); // 10us 동안 유지

        PORTE &= ~(1 << TRIG); // Trig=LOW -> 거리 측정 명령 끝

        while (!(PINE & (1 << ECHO))); // Echo=HIGH 가 될 때까지 대기

        TCNT1 = 0x0000; // Timer/Counter1 값 초기화

        while (PINE & (1 << ECHO)); // Echo=LOW 가 될 때까지 대기

        TCCR1B = 0x00; // Timer/Counter1 클록 정지(클록 입력
차단,CS11~CS10=000)

        distance = (unsigned int)(SOUND_VELOCITY * (TCNT1 * 4 / 2) / 1000);
// 거리=속도 x 시간, 거리 단위는 1mm
```

```
        if (distance < 300) { //30cm 이내 장애물
            display_distance_lcd(distance);
            for (i = 0; i < 5; i++) { // 연속하여 "삐~" 지속
                PORTB=0x10;
                _delay_ms(1);

                PORTB=0x00;
                _delay_ms(1);
            }
        } else if (distance < 600) { // 60cm 이내 장애물
            display_distance_lcd(distance);
            for (i = 0; i < 50; i++) { // 0.1 초 동안 "삐~"
                PORTB=0x10;
                _delay_ms(1);

                PORTB=0x00;
                _delay_ms(1);
            }
            _delay_ms(100); // 0.1 초 동안 묵음
        } else if (distance < 1000) { // 1m 이내 장애물
            display_distance_lcd(distance);
            for (i = 0; i < 250; i++) { // 0.5 초동안 "삐~"
                PORTB=0x10;
                _delay_ms(1);

                PORTB=0x00;
                _delay_ms(1);
            }
            _delay_ms(300); // 0.3 초 동안 묵음
        } else { // 1m 이내 장애물 없을 시 버저 울리지 않음
            display_ok_lcd();
      }
    }

    return 0; // 함수의 형태와 같이 정수형(int)의 값을 반환함
}

void display_distance_lcd(unsigned int distance) {
    unsigned int out = distance / 10;
    char cm[4] = " CM";
    char s1[30];
    sprintf(s1, "%d", out);
    strcat(s1, cm);

    LCD_wBCommand(0x01);
    _delay_ms(1.64);

    LCD_wBCommand(0x80 | 0x00);  // DDRAM Address = 0 설정
    LCD_wString("DISTANCE IS");  // 텍스트 LCD 문자열 출력
```

```
    LCD_wBCommand(0x80 | 0x40); // DDRAM Address = 0x40 설정
    LCD_wString(s1); // 거리 출력
}

void display_ok_lcd() {
    LCD_wBCommand(0x01);
    _delay_ms(1.64);

    LCD_wBCommand(0x80 | 0x00);  // DDRAM Address = 0 설정
    LCD_wString("IT'S OK!");  // 텍스트 LCD 문자열 출력
}
```

<lcd.h> 소스코드

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

#define sbi(x, y) (x |= (1 << y))  // x 의 y 비트를 설정(1)
#define cbi(x, y) (x &= ~(1 << y)) // x 의 y 비트를 클리어(0)

// CON 포트는 포트 C 와 연결됨을 정의
#define LCD_CON PORTC
// DATA 포트는 포트 A 와 연결됨을 정의
#define LCD_DATA PORTA
// DATA 포트의 출력 방향 설정 매크로를 정의
#define LCD_DATA_DIR DDRA
// DATA 포트의 입력 방향 설정 매크로를 정의
#define LCD_DATA_IN  PINA
// RS 신호의 비트 번호 정의
#define LCD_RS 0
// RW 신호의 비트 번호 정의
#define LCD_RW 1
// E 신호의 비트 번호 정의
#define LCD_E 2

// 텍스트 LCD 로 부터 상태(명령)를 읽는 함수
unsigned char LCD_rCommand(void){
    unsigned char temp = 1;

    LCD_DATA_DIR = 0X00;

    cbi(LCD_CON, LCD_RS); // 0 번 비트 클리어, RS = 0, 명령
    sbi(LCD_CON, LCD_RW); // 1 번 비트 설정, RW = 1, 읽기
    sbi(LCD_CON, LCD_E); // 2 번 비트 설정, E = 1
```

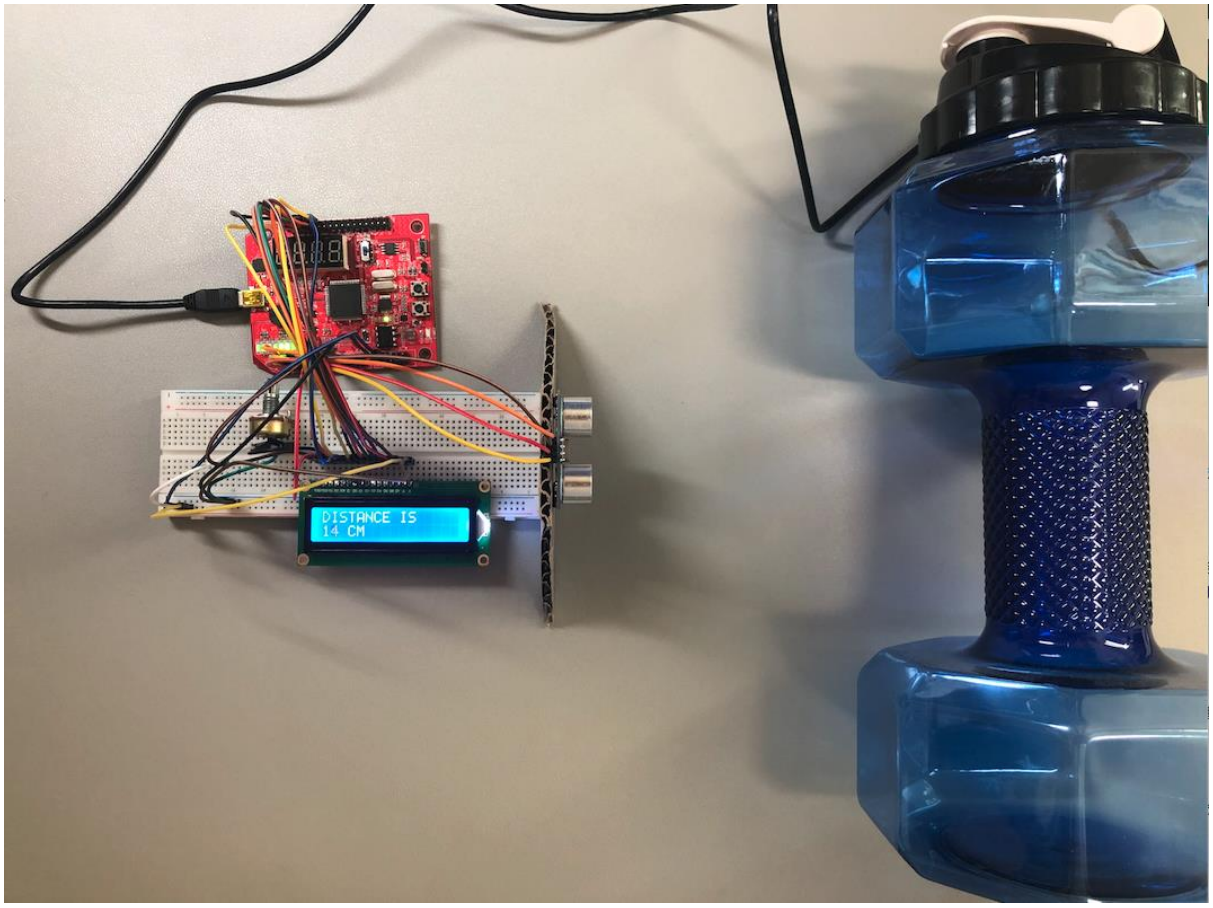
```
    _delay_us(1);

    temp = LCD_DATA_IN; // 명령 읽기
    _delay_us(1);

    cbi(LCD_CON, LCD_E); // 명령 읽기 동작 끝

    LCD_DATA_DIR = 0XFF;
    _delay_us(1);

    return temp;
}

// 텍스트 LCD 의 비지 플래그 상태를 확인하는 함수
char LCD_BusyCheck(unsigned char temp){
    if (temp & 0x80) {
        return 1;
    } else {
        return 0;
    }
}

// 텍스트 LCD 에 명령을 출력하는 함수 - 단, 비지플래그 체크하지 않음
void LCD_wCommand(char cmd){
    cbi(LCD_CON, LCD_RS); // 0 번 비트 클리어, RS = 0, 명령
    cbi(LCD_CON, LCD_RW); // 1 번 비트 클리어, RW = 0, 쓰기
    sbi(LCD_CON, LCD_E);  // 2 번 비트 설정, E = 1

    LCD_DATA = cmd; // 명령 출력
    _delay_us(1);
    cbi(LCD_CON, LCD_E); // 명령 쓰기 동작 끝

    _delay_us(1);
}

// 텍스트 LCD 에 명령을 출력하는 함수 - 단, 비지플래그 체크함
void LCD_wBCommand(char cmd){
    while (LCD_BusyCheck(LCD_rCommand()))
    _delay_us(1);

    cbi(LCD_CON, LCD_RS); // 0 번 비트 클리어, RS = 0, 명령
    cbi(LCD_CON, LCD_RW); // 1 번 비트 클리어, RW = 0, 쓰기
    sbi(LCD_CON, LCD_E); // 2 번 비트 설정, E = 1

    LCD_DATA = cmd; // 명령 출력
    _delay_us(1);
    cbi(LCD_CON, LCD_E); // 명령 쓰기 동작 끝

    _delay_us(1);
}
```

```c
// 텍스트 LCD 를 초기화하는 함수
void LCD_Init(void){
    _delay_ms(100);

    // 비지 플래그를 체크하지 않는 Function Set
    LCD_wCommand(0x38);
    _delay_ms(10);

    // 비지 플래그를 체크하지 않는 Function Set
    LCD_wCommand(0x38);
    _delay_us(200);

    // 비지 플래그를 체크하지 않는 Function Set
    LCD_wCommand(0x38);
    _delay_us(200);

    // 비지 플래그를 체크하는 Function Set
    LCD_wBCommand(0x38);

    // 비지 플래그를 체크하는 Display On/Off Control
    LCD_wBCommand(0x0c);

    // 비지 플래그를 체크하는 Clear Display
    LCD_wBCommand(0x01);
}

// 텍스트 LCD 에 1 바이트 데이터를 출력하는 함수
void LCD_wData(char dat){
    while (LCD_BusyCheck(LCD_rCommand()))
    _delay_us(1);

    sbi(LCD_CON, LCD_RS); // 0 번 비트 설정, RS = 1, 데이터
    cbi(LCD_CON, LCD_RW); // 1 번 비트 클리어, RW = 0, 쓰기
    sbi(LCD_CON, LCD_E); // 2 번 비트 설정, E = 1

    LCD_DATA = dat; // 데이터 출력
    _delay_us(1);
    cbi(LCD_CON, LCD_E);  // 데이터 쓰기 동작 끝

    _delay_us(1);
}

// 텍스트 LCD 에 문자열을 출력하는 함수
void LCD_wString(char *str){
    while (*str) {
        LCD_wData(*str++);
    }
}
```

6. 결과 및 고찰

(1) 결과

- 차량 후방 감지기 거리 감지



- 차량과 장애물 간의 거리가 0cm ~ 1m 이내일 때 출력 화면

- 차량과 장애물 간의 거리가 1m 초과일 때 출력 화면

(2) 고찰

① 결과에 대한 확인
- 계획한 차량 후방 감지기의 기능은 장애물과 차량 간의 거리별로 구별된 경고음 알림과 LCD 에 거리를 표시하는 것이였다.
- 수업에서 제공 받은 JKT-128-1 과 주변 기기(HC-SR04, 1602LCD)를 활용하여 목표했던 기능을 100% 구현하였다.

② 수강한 전공 과목의 컴퓨터 과학 지식 활용
- 1 학년 2 학기 때 수강한 'C 언어 및 실습' 과목에서 배운 내용을 응용하여 임베디드 프로그래밍 코드를 작성하였다.

- 하드웨어에 제어에 최적화된 C 언어를 실제로 프로젝트에 제작 언어로 사용함으로써 low level 에서 어떻게 동작하는지에 대해 이해할 수 있었다.
- 2 학년 2 학기 때 수강한 '논리회로 및 실습' 과목에서 배운 내용을 응용하여 회로 설계를 하고 하드웨어를 구성하였다.
- 배운 이론을 바탕으로 마이크로프로세서인 JKIT-128-1 과 초음파 거리 센서 모듈, LCD 모듈와의 정보 전달 방식을 파악할 수 있었고 하드웨어 실무 능력을 키울 수 있었다.

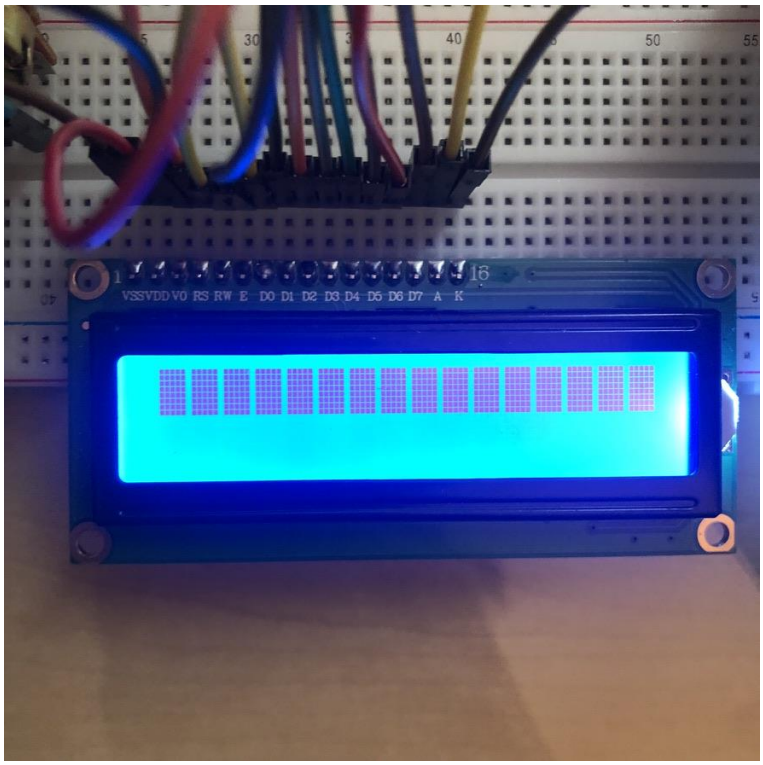
③ Data Sheet 의 중요성 인지
- 프로젝트를 계획하고 개발 초기에 하드웨어에 대한 접근이 미숙하였다.
- 모듈의 Data Sheet 를 기반으로 하드웨어를 빠르게 적응하였다. 모듈의 중요한 기능과 절대 최대 정격을 파악하고 소자들을 어떤 포트에 어떠한 방식으로 연결되어야 하는지 확인하였다.
- 이를 통하여 회로도를 작성함으로써 ATMEGA128 과 주변 모듈들을 어떻게 구성해야 하는지 이해할 수 있었다.

④ 문제 해결: 메인 소스 파일 가독성 저하 문제
- 코드 작성 초기에는 모든 소스코드들을 하나의 메인 소스 파일에서 작성하였다. 컴파일 시에는 문제가 없었지만, 코드 개발이 복잡해져 가독성이 크게 저하되었다.
- 이를 해결하기 위해 모듈에 대한 제어 정의 함수는 헤더 파일로 따로 작성하여 가독성 저하 문제를 해결하였다.

⑤ 문제 해결: HC-SR04 초음파 센서 정밀도 문제
- HC-SR04 초음파 센서를 사용하여 차량과 장애물 간의 거리를 측정하였다. Data Sheet 에 명시되어있는 spec 대로라면 최소 거리가 약 2cm 가 측정되어야 하는데, 최소 6cm 밖에 되지 않았다.
- 이를 해결하기 위해 HC-SR04 의 Data Sheet 에 명시되어있는 전방 초음파 감지 최소, 최대 감지 각도와 거리 측정 범위를 파악하고 정밀도를 검사해보았다.
- 기기를 고정시키고 자를 설치하여 장애물과의 거리를 검사해본 결과, 초음파의 방향과 장애물의 방향이 연직이어야 정밀하게 측정되었고, 연직이 아닌 경우 초음파의 난반사로 인하여 측정 거리가 약 5mm 미만의 오차가 발생하여 측정되었다.
- 따라서 HC-SR04 초음파 센서를 지면과 수평한 상태로 고정시켜 최대한 연직이 되도록 하여 문제를 해결하였다.

⑥ 문제 해결: 1602LCD 출력한 글자가 보이지 않는 문제

- 수령받은 1602LCD 모듈의 뒷면 볼륨 가변 저항이 불량으로 저항을 조절할 수 없었다. 아래 사진과 같이 출력한 글자가 보이지 않고 정사각형만 출력되었다.



- 이를 해결하기 위해 가변 저항을 추가 구매하여 브레드 보드에 설치하였다.
브레드 보드의 특성을 활용하여 맨 아래 칸에서 ATMEGA128 의 GND 과 VCC 를 하나의 케이블로 연결하였다. 가변 저항의 손잡이를 돌리면 저항값이 변하게 되는데, 적절한 저항을 줌으로써 1602LCD 의 글자 대비 값을 조정하였다.