

최종 과제. 초음파 센서를 이용한 리듬 게임

전자공학과 21611591 김 난 희

실험 목적

1. 타이머/카운터 1,3을 사용해본다.
2. 타이머/카운터의 사용해보지 않은 Mode를 실험해본다.
3. 스위치를 대신할 입력 장치로 초음파 센서를 사용해본다.
4. 게임에 따른 정답/오답으로 적절한 LED를 ON/OFF 한다.

실험 동기

어떤 것을 실험해볼까 많이 고민해보았다. 다음의 여러 생각에 의해 아이디어가 나온 작품이다.

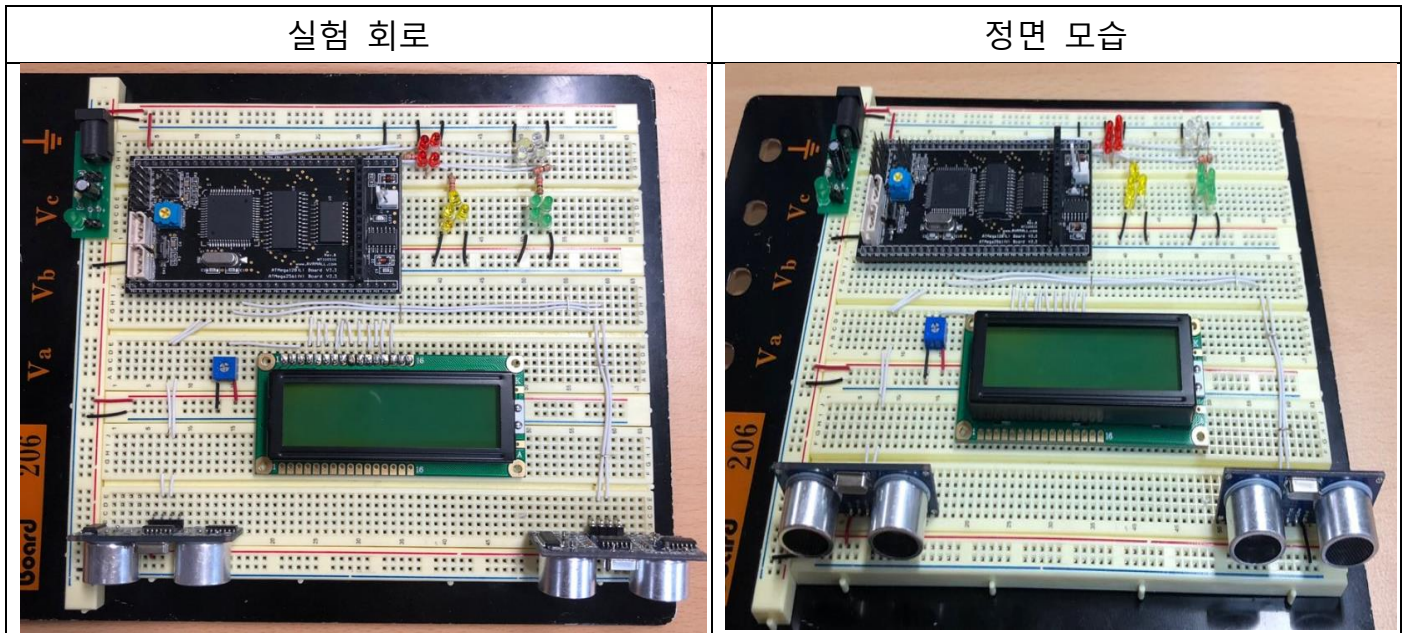
1. 학기동안 실험을 해보면서, 스위치는 채터링 현상과 잘 인식이 되지 않는 현상이 있으므로 사용이 불편하므로, 사용을 덜하자. 채터링 현상을 줄일 수는 있지만, 손대지 않아도 할 수 있는 무엇가를 만들고 싶었다. 그렇게 초음파 센서나 적외선 센서 둘 중 하나를 사용해보고 싶었다.
2. 타이머나 ADC 둘 중 하나는 꼭 사용하자. 타이머/인터럽트는 학기 중 실험에도 많이 사용해보았지만, 타이머/카운터 1은 사용을 해본적이 없어 한번 사용해보고 싶었다. CTC모드도 그러하다.
3. 만들고나면 즐거움과 배운점이 있으면 좋겠다. 처음에는 LED를 피아노 연주처럼 게임을 만들어보자, LCD로 점프게임을 해보자, 여러 재밌는 실험 아이디어가 나왔다. 하지만, 피아노 연주를 하기에는 가지고 있는 LED 색상이 많이 부족하였고, MCU를 통해 많은 LED를 켜기 위해서는 Sink제어가 필요할 것 같았다. 회로 구성 또한 복잡할 것 같았다. LCD로 하는 점프게임은, 학기 중 실험을 통해 LCD가 생각보다 튼튼하지 않음을 느꼈다.

위와 같은 여러가지 이유로 최종 과제로 초음파 센서를 이용하여, LCD로 문제가 주어지고, LED로 정답을 볼 수 있는 간단한 리듬 게임을 만들게 되었다.





1. 본 실험전 초음파의 거리 테스트

본 실험에 앞서 두 개의 초음파가 같은 초음파이지만, 동일한 스펙과 프로그래밍에도 동일한 측정값을 가지는지 테스트해보아야 한다. 또한 두 개의 초음파를 거리식에 따라 거리를 계산해보았을 때, 스펙에 따른 정확한 측정값을 가지는 지 확인해보아야한다. 거리가 정확하지 않게 나올 경우 실험을 통해 수정해주는 작업이 필요하다.

1-1. 회로 분석과 구동

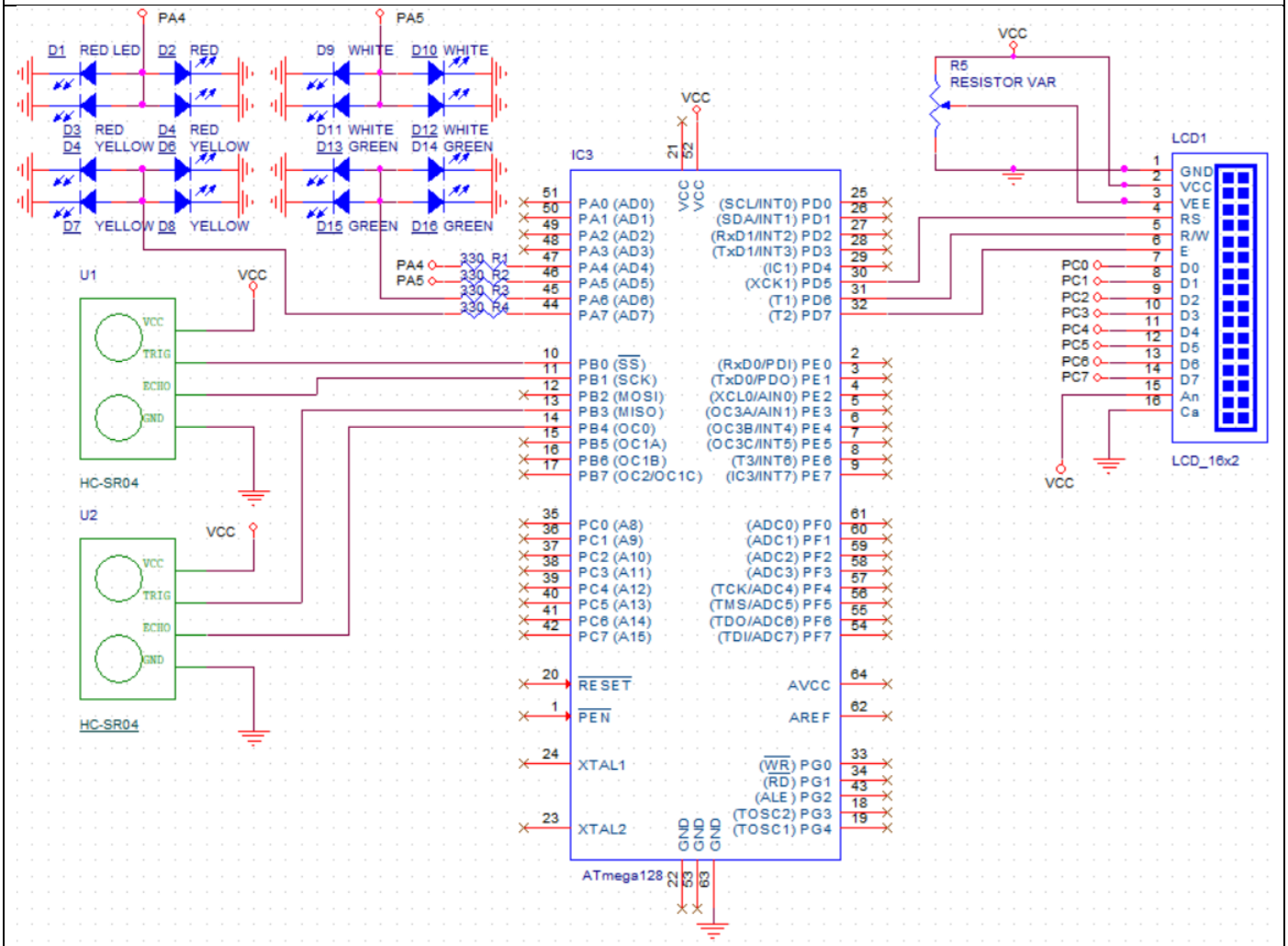


회로는 사진과 같이, LCD, 초음파, 가변저항, LED(붉은색, 흰색, 초록색, 노란색)으로 구성된다. 초음파를 이용하여 앞에 장애물이 있을 때, 거리를 계산한 후 10cm, 20cm, 30cm, 40cm에 따라 LCD에 "*****"으로 표시한다.

10cm 이내일 경우	20cm 이내일 경우
<div>*****</div>  <p><표시되는 LCD와 LED 모습></p>	<div>*****</div>  <p>< 표시되는 LCD와 LED 모습></p>
30cm 이내일 경우	30cm 이상일 경우
<div>*****</div>  <p>< 표시되는 LCD와 LED 모습></p>	<div>*****</div>  <p>< 표시되는 LCD와 LED 모습></p>

회로를 더 보기 쉽게 아래의 OrCAD로 그린 회로도들 직접 작성하여 첨부한다. (회로도는 전문가가 작성하는 방식과는 틀릴 수 있습니다.)

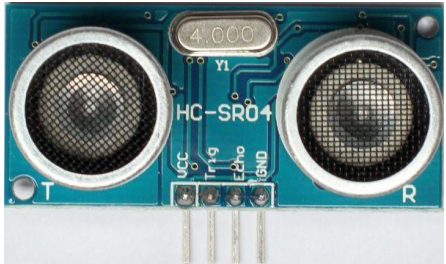
(사용한 프로그램)OrCAD Capture CIS – (파일명)design_PaperRhythm



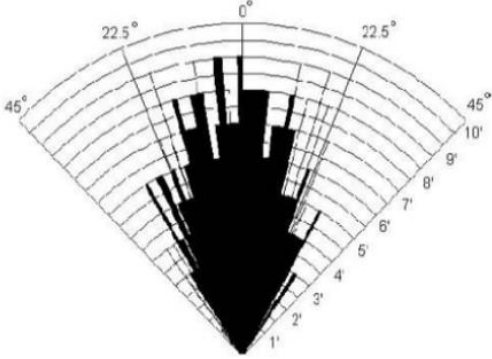

초음파는 사람이 듣지 못하는 음파 영역대이고, 20kHz를 넘으면 사람이 듣지 못한다. 아래 사진은 사용한 초음파의 사진이고 Data sheet는 다음과 같다. 초음파 센서 모듈은 송신부(Trigger), 수신부(Echo)와 제어 회로로 구성된다. 간단하게 어떤식으로 구동하는 지 알아보면,

1. TRIGGER 핀을 통해 10us 길이의 펄스를 입력
2. 센서에서 전방을 향해 초음파를 발사
3. 물체에 부딪혀 반사되는 초음파가 돌아오는 시간의 길이를 계산
4. ECHO 핀에서 시간의 길이에 비례하는 펄스를 출력한다.

자세하게는 뒤에서 다시 설명할 것이다.

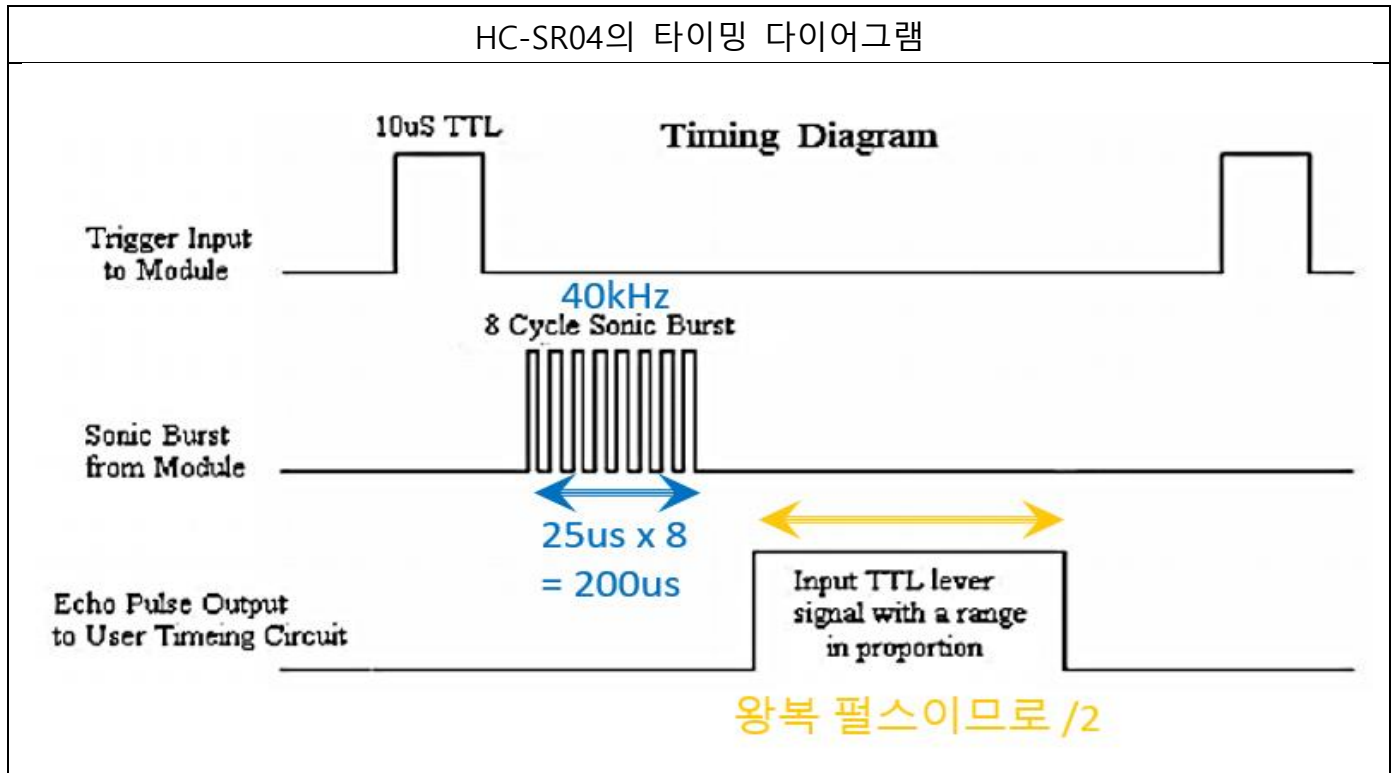
사용한 초음파	Data sheet																	
	Electric Parameter																	
	<table> <tr> <td>Working Voltage</td><td>DC 5 V</td></tr> <tr> <td>Working Current</td><td>15mA</td></tr> <tr> <td>Working Frequency</td><td>40Hz 40kHz</td></tr> <tr> <td>Max Range</td><td>4m</td></tr> <tr> <td>Min Range</td><td>2cm</td></tr> <tr> <td>MeasuringAngle</td><td>15 degree</td></tr> <tr> <td>Trigger Input Signal</td><td>10uS TTL pulse</td></tr> <tr> <td>Echo Output Signal</td><td>Input TTL lever signal and the range in proportion</td></tr> <tr> <td>Dimension</td><td>45*20*15mm</td></tr> </table>	Working Voltage	DC 5 V	Working Current	15mA	Working Frequency	40Hz 40kHz	Max Range	4m	Min Range	2cm	MeasuringAngle	15 degree	Trigger Input Signal	10uS TTL pulse	Echo Output Signal	Input TTL lever signal and the range in proportion	Dimension
Working Voltage	DC 5 V																	
Working Current	15mA																	
Working Frequency	40Hz 40kHz																	
Max Range	4m																	
Min Range	2cm																	
MeasuringAngle	15 degree																	
Trigger Input Signal	10uS TTL pulse																	
Echo Output Signal	Input TTL lever signal and the range in proportion																	
Dimension	45*20*15mm																	

입력 전압은 DC 5V, 동작 전류는 15mA,
 동작 주파수는 40Hz로 초음파 센서이므로 40Hz의 초음파를 사용한다는 의미이다. 하지만, 사람은 20kHz이상은 듣지 못하므로, 데이터시트가 잘 못된 것임을 알 수 있다. 40kHz로 정정해야한다.
 최대 측정 거리는 4m로 나온다. 실제 실험을 해보면 4m까지는 나오지 않는다. 초음파는 음파이므로 주변의 여러 잡음에도 영향을 미칠 것이다.
 최소 측정 거리는 2cm이다.
 측정 각은 15도로 되어있는데 실제로 다음의 Datasheet의 사진과 매칭이 되지않아 잘 이해가 되지 않았다. 공부를 해보니, 아래 첫 번째 그림 처럼 최대 30도 정도의 각도로 측정하지만 , 15도는 다음 두 번째 그림과 같이 이해할 수 있다. 타겟의 15도 정도로 측정이 가능한 것이다. HC-SR04의 데이터 시트가 여러 개라, 둘 다 정확한 그림인지, 둘 중하나가 잘못된 그림인지는 모르겠다.

Data sheet	이해한바
 <p>Practical test of performance, Best in 30 degree angle</p>	 <p>HC-SR04</p>

Trigger input signal은 초음파에서 보내는 신호로, 10us 길이의 구형파 형태의 펄스를 내보낸다.
 Ehco output signal은 출력되는 에코 펄스 신호로, 측정 거리에 따라 펄스의 너비가 달라짐을 의미한다.
 다. 마지막으로 dimension은 치수로, 45*20*15mm는 가로, 세로, 높이를 의미한다.

위의 Datasheet를 이해하는 것도 중요하지만, 초음파의 원리가 담겨 있는 타이밍 다이어그램을 해석하는 것도 프로그램을 짜는데 큰 역할을 한다. 사용한 초음파 센서의 타이밍 다이어그램은 다음과 같다.



맨 위에서부터 순서대로 설명한다.

1. 모듈의 트리거 인풋에서 10μs의 HIGH 펄스를 내보낸다.
2. 40kHz의 초음파 버스트가 발생한다.

초음파 Datasheet에서 초음파는 40kHz로 내보낸다고 하였으므로,

$$f = \frac{1}{T} = \frac{1}{40k} = 25\mu s$$

8 사이클을 내보내므로, $25\mu s \times 8 = 200\mu s$ 가 된다.

3. 에코는 초음파를 보낸 직후 High이고, 앞의 장애물을 감지하면 Low가 된다. 신호는 갔다가 오는 것 까지 포함되므로, 왕복 시간을 의미한다. 왕복이므로 2를 나누어주어 계산한다.

$$(\text{거리}) = (\text{에코의 High pulse time 왕복시간}) \times \text{소리의 속도}(340m/s) / 2$$

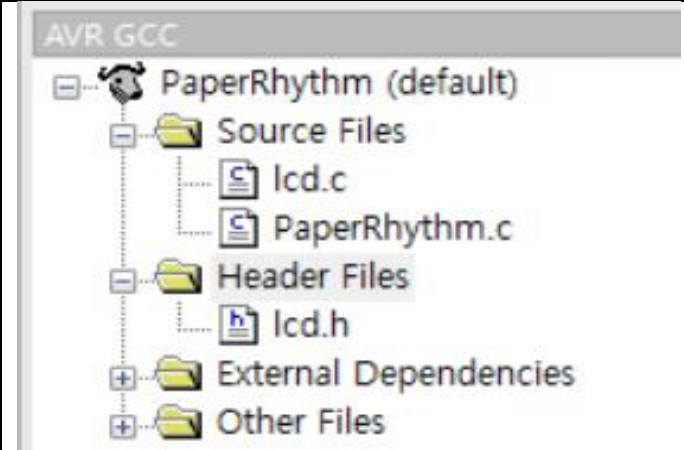
예를 들어 1cm를 가는데 걸리는 시간을 계산해준다.

$$\begin{aligned} \text{(에코의 High pulse time 왕복 시간)} &= \frac{(\text{거리}) \times 2}{0.034 \text{ cm/us}} \\ &= \frac{1 \text{ cm} \times 2}{0.034 \text{ cm/us}} = 58.8235 \text{ us} \end{aligned}$$

즉, 1cm 가는데 걸리는 시간은 왕복으로 58.8235us가 걸린다.

사용한 초음파의 최소 측정 거리는 2cm이므로, $58.8235 \text{ us} \times 2 = 117.6 \text{ us}$ 가 된다. 다음을 참고하여 프로그램을 작성하였다.

1-2. 프로그램 분석

소스 프로그램 구성	
	<p>저번 실험(LCD 실험, USART 실험)때 사용하였던 lcd.c파일과 lcd.h파일이 동일하다. 여기서 본 메인 함수가 들어가는 파일 PaperRhythm.c 파일로 전체 프로그램이 구성된다.</p> <p>파일의 이름을 PaperRhythm으로 설정한 이유는, 손바닥으로하는 리듬 게임이기 때문이다.</p>

프로그램은 초음파가 Trigger를 쏘면, High pulse를 기다린다. High pulse가 감지되면, 타이머1의 CTC 모드로 시간을 세아린다. Low Pulse가 감지되면, 타이머를 정지하고 거리를 계산한다. 거리를 이용하여 어떻게 동작될지를 결정한다. 타이머1과 CTC 모드를 사용하였지만, 굳이 사용한 특별한 역할이 없다. 다른 타이머0나 다른 모드를 사용하여도 된다.

프로그램에서 초음파 2개에 대해 다음의 구문으로 거리를 계산해주었는데, 의미는 다음과 같다.

```
distance1=TCNT1/116; // cm로 변경
distance2=TCNT3/116; // cm로 변경
```

위에서 1cm를 가는데 걸리는 시간이 58.8235us가 걸린 것으로, 사용한 초음파의 최소 측정 거리는 2cm이므로, $58.8235 \text{ us} \times 2 = 117.6 \text{ us}$ 가 되는 점을 참고하였다. High level일 때 CTC 모드로 시간을 측정하기 시작하여, Low level이 되면 타이머를 정지하도록 하였다. 8개의 40kHz 초음파 버스트를 발생시켰으므로 8분주를 하여 사용하여, 그때의 TCNT를 117로 나누어주면 거리를 계산된다. 실험에서는 대략적으로 계산하다보니 116을 넣어 실험하게 되었다.

PaperRhythm.c (메인 c파일)

```
#include <stdio.h> // standard input output 헤더파일

#include <avr/io.h> // avr의 input output 헤더파일

#include <util/delay.h> // delay 구문 사용하기 위함

#include <avr/interrupt.h> // 오버플로우인터럽트와 컴페어매치인터럽트를 사용안하면 사용안해도 됨

본 프로그램은 초음파의 HIGH PULSE 감지 후 타이머를 정지하는 형태로 작성하였으므로 생략해도 프로그램이 돌아갈 것이다.

#include "lcd.h" // lcd.c의 함수들을 사용하기 위한 함수들이 정의된 헤더파일 선언

#define RED PA4 //빨간색 LED

#define WHITE PA5 //흰색 LED

#define GREEN PA6 //초록색 LED

#define YELLOW PA7 //노란색 LED

char lcd_string[2][MAX_LCD_STRING]; //lcd에 입력할 문자열을 저장하는 변수

unsigned int distance1; //초음파 1에서 계산할 거리

unsigned int distance2; //초음파 2에서 계산할 거리

void PORT_init() //포트 초기화

{

    //PA4~7 : LED를 출력으로 설정한다.

    DDRA = 0xF0;

    //PB0~1 : 초음파 1 , PB3~4 : 초음파 2 // Trigger를 출력으로, Echo를 입력으로 설정

    DDRB = 0x09; //PORTB = 0x12;

}
```

```
void MODE_1_2() // 초음파 1, 2의 왕복 시간을 계산할 타이머 모드 선택
```

```
{
```

```
TCCR1A=0x00; TCCR1B=0x08; // CTC mode
```

```
TCCR3A=0x00; TCCR3B=0x08; // CTC mode
```

```
}
```

타이머 0와 2가 같고, 타이머 1과 3이 같다. 정확히 회로나 모든 부분에서 같은 것은 아니고, 대략적인 설정부분에서 같다.

Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter3 Control Register A – TCCR3A

Bit	7	6	5	4	3	2	1	0	
	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	TCCR3A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR1A와 TCCR3A를 모두 0x00으로 설정했기 때문에 OCRnA/OCRnB/OCRnC 모두 사용하지 않음을 의미한다.

Timer/Counter1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter3 Control Register B – TCCR3B

Bit	7	6	5	4	3	2	1	0	
	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30	TCCR3B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR1B와 TCCR3B를 모두 0x08으로 설정했는 것은 WGM12과 WGM32을 High로 SET한 것과 같다. 즉 TCCR1A 레지스터와 TCCR3에 레지스터와 합쳐서 WGM비트를 보면 다음과 같다.

타이머 1	WGM13	WGM12	WGM11	WGM10
-------	-------	-------	-------	-------

	0	1	0	0
타이머 3	WGM33	WGM32	WGM31	WGM30
	0	1	0	0

즉, 아래 표에서 CTC모드로 설정한 것이다.

Table 61. Waveform Generation Mode Bit Description

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation ⁽¹⁾	TOP	Update of OCRnX at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

Note: 1. The CTCn and PWMn1:0 bit definition names are obsolete. Use the WGMn2:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

다음은 초음파 트리거 신호를 보내는 곳이다. 초음파 1과 초음파 2를 같은 트리거 함수에 작성하여 포트도 통합하여 제어해주면 동작이 되지않았다. 합쳐서하는 것은 좀 더 테스트를 해봐야할 것 같고, 실험은 트리거 신호를 분리하여 쏘는 것으로 작성되었다.

```
void Trigger1() //초음파1 트리거 신호
```

```
{
```


Table 62. Clock Select Bit Description

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	clk _{IO} /1 (No prescaling)
0	1	0	clk _{IO} /8 (From prescaler)
0	1	1	clk _{IO} /64 (From prescaler)
1	0	0	clk _{IO} /256 (From prescaler)
1	0	1	clk _{IO} /1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

아래에서 TCCR1B=0x02로 설정한 것은 8분주를 의미한다. TCCR3B도 마찬가지이다.

```
void Echo1() //초음파센서1 값 // Ehco : PB1
{
    while(!(PINB&0x02)); // high가 될때까지 대기 //PULSE 시작
    TCNT1=0x00; TCCR1B=0x02; // 카운터 시작, 8분주
    while(PINB&0x02); // low가 될때까지 대기 //PULSE 종료
    TCCR1B=0x08; // 카운터 정지의 목적
    distance1=TCNT1/116; // cm로 변경 // 프로그램 코드 전에 설명 참조
}

void Echo2() //초음파센서2 값 // Ehco : PB4
{
    while(!(PINB&0x10)); // high가 될때까지 대기 //PULSE 시작
    TCNT3=0x00; TCCR3B=0x02; // 카운터 시작, 8분주
    while(PINB&0x10); // low가 될때까지 대기 //PULSE 종료
    TCCR3B=0x08; // 카운터 정지의 목적
    distance2=TCNT3/116; // cm로 변경 // 프로그램 코드 전에 설명 참조
}
```

```
void LED(int led) //LED ON/OFF 함수
```

```
{  
  
    //RED : PA4 , WHITE : PA5 , GREEN : PA6 , YELLOW : PA7  // #define으로 선언해놓음  
  
    // 나머지는 다 끄고, 빨강색 LED만 ON  
  
    if(led==1) {PORTA = ~((1<< WHITE)|(1<< GREEN)|(1<< YELLOW)); PORTA |= 1<< RED; }  
  
    // 나머지는 다 끄고, 흰색 LED만 ON  
  
    else if(led==2){PORTA &= ~((1<< RED)|(1<< GREEN)|(1<< YELLOW)); PORTA |= 1<< WHITE; }  
  
    // 나머지는 다 끄고, 초록색 LED만 ON  
  
    else if(led==3){PORTA &= ~((1<< RED)|(1<< WHITE)|(1<< YELLOW)); PORTA |= 1<< GREEN; }  
  
    // 나머지는 다 끄고, 초록색 LED만 ON  
  
    else if(led==4){PORTA &= ~((1<< RED)|(1<< WHITE)|(1<< GREEN)); PORTA |= 1<< YELLOW; }  
  
    //else { PORTA &= ~((1<< RED)|(1<< WHITE)|(1<< GREEN)|(1<< RED)); } 안해도 됨  
  
}
```

```
void Question(int position) // LCD에 "*****"를 띄우는 위치를 받아 LCD에 출력
```

```
{  
  
    sprintf(lcd_string[0], "*****"); //LCD에 출력할 문자열 저장  
  
    LCD_command(0x01); //LCD 화면 Clear  
  
  
    if(position==1) LCD_str_write(0, 0, lcd_string[0]); //첫 번째 위치  
  
    else if(position==2) LCD_str_write(0, 8, lcd_string[0]); //두 번째 위치  
  
    else if(position==3) LCD_str_write(1, 0, lcd_string[0]); //세 번째 위치  
  
    else if(position==4) LCD_str_write(1, 8, lcd_string[0]); //네 번째 위치  
  
}
```

```
int    main(void)  //MAIN 문
{

    LCD_init(); // LCD 초기화 함수

    PORT_init(); // 포트 초기화 함수

    MODE_1_2(); // 타이머1, 3 모드 선택 초기화 함수

    while(1){

        Trigger1(); Echo1(); //초음파1의 Trigger 신호를 보내고 받아 거리를 계산

        Trigger2(); Echo2(); //초음파2의 Trigger 신호를 보내고 받아 거리를 계산


        //거리가 5cm 미만 : RED LED ON, LCD 첫 번째 위치 “*****” 출력

        if (distance1<5 && distance2< 5){ LED(1); Question(1); }


        //거리가 10cm 미만 : RED LED ON, LCD 첫 번째 위치 “*****” 출력

        else if(distance1<10 && distance2<10){ LED(1); Question(1); }


        //거리가 15cm 미만 : WHITE LED ON, LCD 두 번째 위치 “*****” 출력

        else if(distance1<15 && distance2<15){ LED(2); Question(2); }


        //거리가 20cm 미만 : WHITE LED ON, LCD 두 번째 위치 “*****” 출력

        else if(distance1<20 && distance2<20){ LED(2); Question(2); }


        //거리가 25cm 미만 : GREEN LED ON, LCD 세 번째 위치 “*****” 출력

        else if(distance1<25 && distance2<25){ LED(3); Question(3); }
```

```

//거리가 25cm 미만 : GREEN LED ON, LCD 세 번째 위치 "*****" 출력

else if(distance1<30 && distance2<30){ LED(3); Question(3); }

//거리가 35cm 미만 : YELLOW LED ON, LCD 네 번째 위치 "*****" 출력

else if(distance1<35 && distance2<35){ LED(4); Question(4); }

//거리가 40cm 미만 : YELLOW LED ON, LCD 네 번째 위치 "*****" 출력

else if(distance1<40 && distance2<40){ LED(4); Question(4); }

//거리가 40cm 이상 : YELLOW LED ON, LCD 네 번째 위치 "*****" 출력

else { LED(4); Question(4); }

}

return 0; // 종료하며 마침
}

```

lcd.c

```

#include <avr/io.h> // avr의 input output 헤더파일
#include "lcd.h" // lcd.c의 함수들을 사용하기 위한 함수들이 정의된 헤더파일 선언

#define RS PD5 // LCD 문자디스플레이에 연결된 포트D 의 핀번호
// RS는 명령인지 DATA 인지 결정
#define RW PD6 // 포트D의 6번 핀에 RW 핀 연결
#define E PD7 // Enable 연결

void gen_E_strobe(void) // 사용할 수 있게 Enable 해주는 함수
{
    volatile int i; // volatile 변수로, 사이즈 최적화를 피함.

    PORTD |= 1<<E; // E 신호를 High로
    for(i=0; i<10; i++); // E 스트로브 신호를 일정기간 High로 유지
}

```



```

PORTD &= ~(1<<E);    // E 신호를 Low로
}

void wait_BusyFlag(void) // busy flag를 읽어 0이 될 때까지 기다림
{
    volatile int      i; // volatile 변수로, 사이즈 최적화를 피함.
    unsigned char      bf; // buffer를 의미하는 unsigned 형 변수 선언

    DDRC = 0x0;          // 포트C를 입력핀으로 설정
    PORTD = (PORTD & ~(1<<RS)) | 1<<RW; // RS <- Low, RW <- High
    do{ // 먼저 실행 후 while()의 조건을 보고 후 판단
        PORTD |= 1<<E;      // E 신호를 High로
        for(i=0; i<10; i++); // E 스트로브 신호를 일정기간 High로 유지
        bf = PINC & 1<<PC7; // busy flag 읽어 냄
        PORTD &= ~(1<<E);   // E 신호를 Low로
    }while( bf );          // bf 값이 0이 아니면 busy, 0 일 때까지 반복
}

void LCD_command(unsigned char data) // lcd에 주는 명령 함수, 명령어 보냄
{
    wait_BusyFlag();        // busy flag가 0될 때까지 대기
    DDRC = 0xFF;           // 포트C를 출력핀으로 설정
    PORTC = data;          // data 출력
    PORTD &= ~(1<<RS | 1<<RW); // RS <- 0, RW <-0
    gen_E_strobe();        // E 스트로브 신호 만들기
}

void LCD_data_write(unsigned char data) // 여기 data는 아스키 코드가 들어갈 것.
{
    wait_BusyFlag(); // busy 체크
    DDRC = 0xFF; // 출력으로 설정
    PORTC = data; // data의 값이 여기 들어옴
    PORTD = (PORTD | 1<<RS) & ~(1<<RW); // RS <- 1, RW <-0
    // 데이터 내보냄 //명령어 x
    gen_E_strobe(); // E 스트로브 신호 만들기
}

void LCD_init(void) // lcd 초기화 함수
{
    DDRD |= 1<<RS | 1<<RW | 1<<E; // RS, RW, E 핀을 출력핀으로 설정

    PORTD &= ~(1<<RS | 1<<E | 1<<RW); // 초기에 RS, E, RW <- 0
}

```

```

LCD_command(0x3C); // 인터페이스, 디스플레이 설정
LCD_command(0x02); // cursor 초기화
LCD_command(0x01); // clear display
LCD_command(0x06); // entry 모드
LCD_command(0x0F); // display on/off
}

```

LCD 사용 순서는 다음과 같이 진행된다. 위의 LCD_init(void) 와도 매칭해보며 비교해보면 된다.

LCD 초기화 순서

1. 초기화 전 30msec 를 기다림
2. Function set 명령(0011xx00B)을 내보냄
3. Display On/Off control 명령(00001xxxB) 을 내보냄
4. Entry Mode set 명령(000001xxB)을 내보냄
5. DD RAM 어드레스를 내보냄
6. 표시할 문자데이터를 내보냄
7. 5,6과정을 반복

Function set에는 다음과 같은 것들이 초기에 설정된다.

인터페이스/ 디스플레이 설정	DL	1 : 8비트 인터페이스 0 : 4비트 인터페이스
	N	1 : 두 줄 표시 0 : 한 줄 표시
	F	1 : 문자 5×10 도트 0 : 문자 5×7 도트

font를 보면 우리는 5x10 도트를 사용한다.

이런 것들이 어떻게 디스플레이되는 지 살펴보면, 다음 그림과 같다.

A Custom 5x8 Pixel Character:

Image Coding:

	1	2	3	4	5
1					
2					
3					
4					
5					
6					
7					
8					

Binary Coding:

0b000	00000
0b000	01010
0b000	01010
0b000	01010
0b000	00000
0b000	10001
0b000	01110
0b000	00000

1 = Black, 0 = White

위 그림은 5x8 도트를 예시로 나타낸 것이다. 검은색으로 디스플레이할 모양을 다음과 같이 1로 설정해주어 보아게 한다.

```
void set_cursor(unsigned int row, unsigned int col) // 커서 위치 설정
```

```
{
    LCD_command(0x80 + (row % 2) * 0x40 + (col % 0x40));
} // 0x80은 DDRAM 주소이고 row를 설정해주는 부분과 column을 설정하는 부분으로 되어있다.
```

만약 위의 함수에 set_cursor(0,0)을 넣어주면, 1000 0000이 들어가, 0x80 command를 실행한다. 만약 (1,2),를 넣어주면, 0x80 + 1x(0x40) + 2 이므로,

```
= 1000 0000
   0100 0000
   0000 0010
```

1100 0010 = C2 가 된다.

노란색으로 표시해 놓은 부분은 DDRAM 주소 부분이다.

결국 아래의 함수를 통해 위 set_cursor 내부의 Command 함수는 data 주소를 날리는 부분이라 이해할 수 있겠다.

직관적으로 소스를 해석하자면, (row % 2)*0x40은 LCD의 한줄이 0x3F임을 의미한다. 16글자를 표시할 수 있다. row 값에는 0 아니면 1이 들어가기 때문에, 0이 들어가면 전체 값((row % 2)*0x40)은 0이 되므로 첫째줄부터 디스플레이 함을 알려준다. col 값에는

// 함수 정의 : row, col 위치에서 문자열 str 을 LCD에 출력시킨다.

```
void LCD_str_write(unsigned int row, unsigned int col, char *str)
```

```
{
    int i;
```

```

set_cursor(row, col); // 위 함수 인자로부터 받은 곳에 Cursor 위치 조정, DDRAM 주소
for(i=0; (i+col < MAX_LCD_STRING) && (str[i] != '\0'); i++) /\0은 문자 끝을 말함
    LCD_data_write(str[i]); // data를 날림
}

```

lcd.h

```

#ifndef __LCD_H__
#define __LCD_H__

#define MAX_LCD_STRING    0x40 // 한 줄에 64글자 최대 디스플레이

// 아래는 외부에서도 함수를 쓸 수 있게 extern 으로 선언해 놓은 부분.
extern void    gen_E_strobe(void);
extern void    wait_BusyFlag(void);
extern void    LCD_command(unsigned char data);
extern void    LCD_data_write(unsigned char data);
extern void    LCD_init(void);
extern void    set_cursor(unsigned int row, unsigned int col);
extern void    LCD_str_write(unsigned int row, unsigned int col, char *str);
#endif

```

1-3. 실험 결과 및 분석

휴지곽으로 거의 평평하게 하여 거리를 측정하려 노력했다. 아래의 자는 30cm이다.

실험 사진을 보면,

10cm 미만일 경우 **빨간색 LED**가 ON되고, LCD에는 **첫** 번째 자리에 "*****"이 표시된다.

20cm 미만일 경우 **흰색 LED**가 ON되고, LCD에는 **두** 번째 자리에 "*****"이 표시된다.

30cm 미만일 경우 **초록색 LED**가 ON되고, LCD에는 **세** 번째 자리에 "*****"이 표시된다.

30cm 이상일 경우 **노란색 LED**가 ON되고, LCD에는 **네** 번째 자리에 "*****"이 표시된다.

휴지곽을 각 10cm, 20cm, 30cm에 대하여 최대 임계값까지 두었을 때, 사진을 찍었다. 10cm까지는 매우 정확했다. 그 후로 20cm, 30cm는 경계가 아니고, 조금 더 작은 약 19cm, 28.3cm가 경계로 측정되었다. 물론 초음파가 나란하게 휴지곽과 평행으로 위치하고, 손 떨림이 있음 등 오차가 발생할 수 있다.

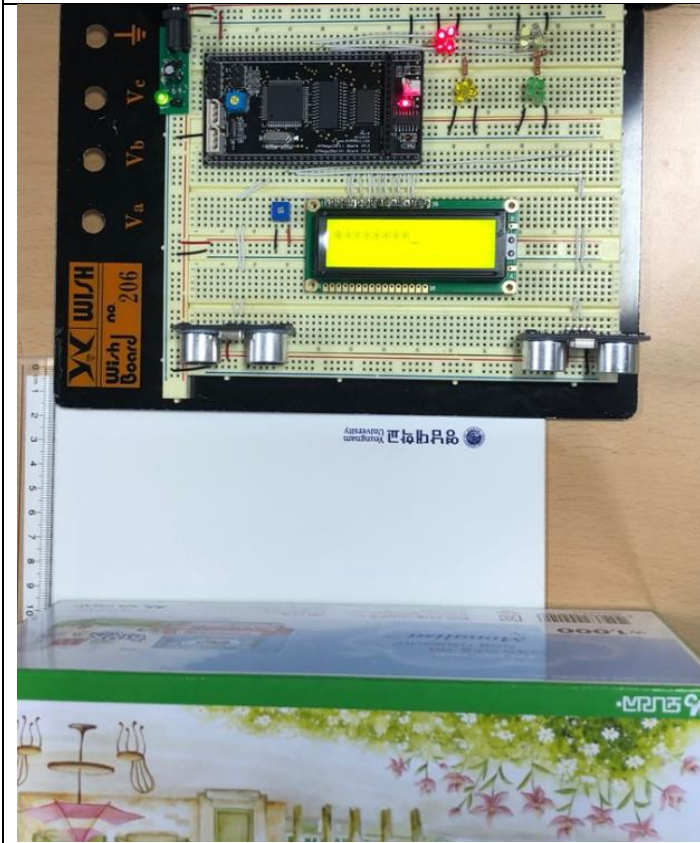
뒤의 본 실험에서는 10cm 전후로만 이용하여 코딩을 할 것이기 때문에 큰 문제가 없어 더 이상 테스트를 종료했다. 만약 더욱 정확한 거리를 얻고 싶다면, Echo1(), Echo2() 함수의 다음 부분의 숫자를 수정해주면 될 것이다. 정확한 테스트 결과는 동영상으로 확인할 수 있다.

```

distance1=TCNT1/116; // cm로 변경
distance2=TCNT3/116; // cm로 변경

```

10cm 미만



20cm 미만



30cm 미만



30cm 이상

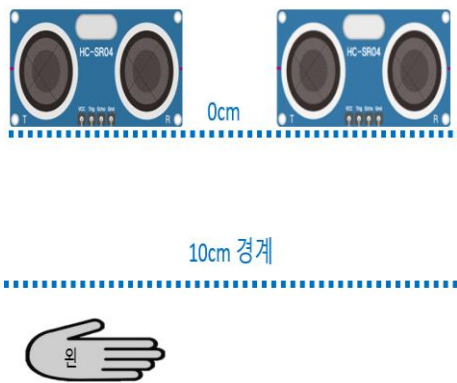
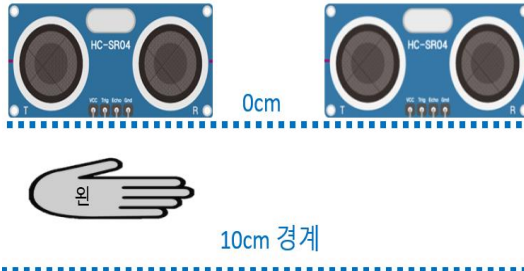


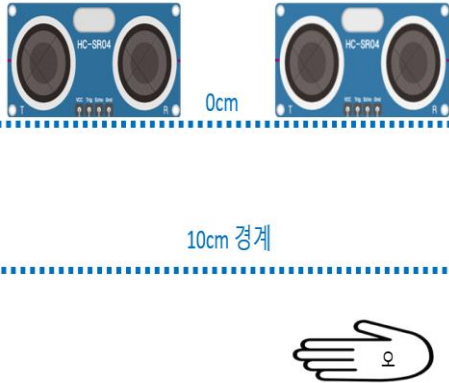

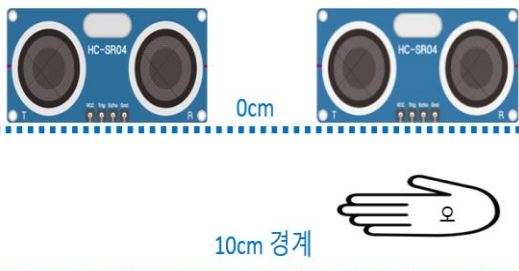

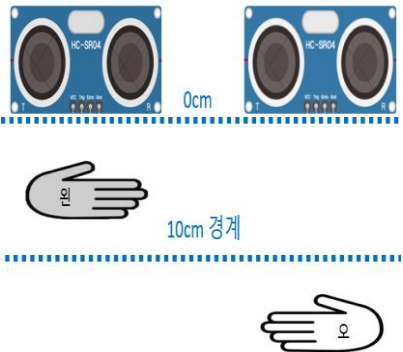
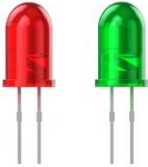

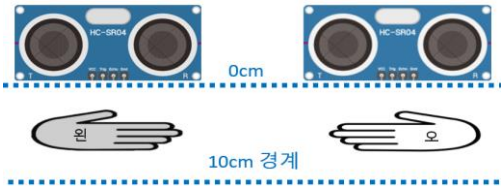

2. 본 실험. 초음파 센서를 이용한 리듬 게임

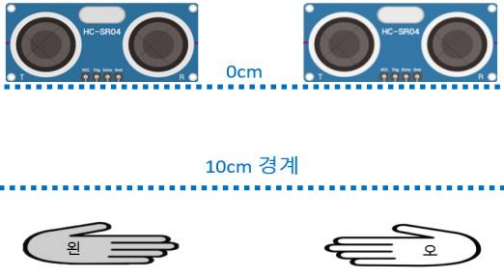
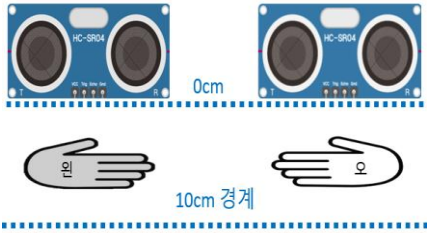
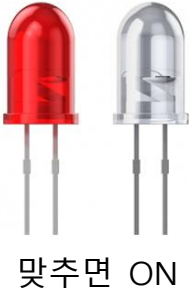
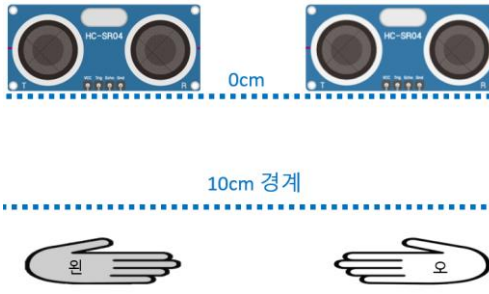
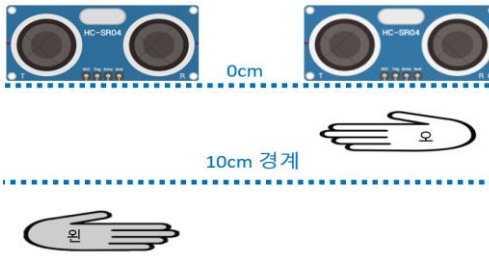
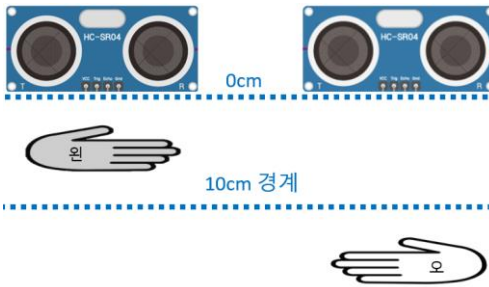

<리듬 게임의 규칙>

1. LCD에서 문제가 제시되고, 초음파로부터 손바닥을 반대로 위치하면 정답이고, 똑같이 위치하면 오답이다.
2. level 1에서는 표시되는 쪽의 손동작만 반대로 하면 되고, level 2에서는 양쪽(오른쪽 왼쪽)의 손동작 모두 반대로 해야 정답이다.
3. 정답일 경우 같은 위치에 있는 LED가 ON되고, 틀릴 경우 빨간색 LED가 3번 깜빡인다.

아래는 문제와 오답의 예시이다. (이해가 잘되지않는다면 동영상을 보시면 이해가 잘 될 것입니다.)

<p>제시된 문제 예시 level 1-(1)</p>	<p>*****</p> <p><LCD에 제시되는 문제의 모습></p>		
<p>정답</p>  <p>맞추면 ON</p>	<p>오답</p>  <p>3번 ON/OFF</p>		<p>제시된 문제 예시 level 1-(2)</p> <p>*****</p> <p>LCD에 제시되는 문제의 모습</p>
<p>정답</p>	<p>오답</p>		

	 맞추면 ON		 3번 ON/OFF
제시된 문제 예시 level 2-(3)	<div>*****</div> <div>*****</div> <div>LCD에 제시되는 문제의 모습</div>		
정답		오답	
	 맞추면 ON	<div>(1)</div>  <div>(2)</div>  <div>(3)</div>	 3번 ON/OFF

		 <p>(4)</p>	
<p>제시된 문제 예시 level 2-(4)</p>	<p>*****</p> <p>LCD에 제시되는 문제의 모습</p>		
정답		오답	
	 <p>맞추면 ON</p>	<p>(1)</p>  <p>(2)</p>  <p>(3)</p> 	 <p>3번 ON/OFF</p>

2-1. 실험 회로

실험 회로는 초음파 거리 테스트 실험과 동일합니다.

2-2. 프로그램 분석 및 결과

다음의 메인 C파일을 제외하고, lcd.h와 lcd.c는 동일합니다. 전체 프로그램은 lcd.h, lcd.c, PaperRhythm.c로 구성되어 있습니다. 앞선 실험, 초음파 거리 테스트의 메인 c파일 소스와 다른 부분은 붉은 색으로 표기하였습니다. (겹치는 부분의 자세한 설명은 앞선 실험에서 확인하시길 바랍니다.)

PaperRhythm.c (메인 c파일)	
#include	<stdio.h> // standard input output 헤더파일
#include	<avr/io.h> // avr의 input output 헤더파일
#include	<util/delay.h> // delay 구문 사용하기 위함
#include	<avr/interrupt.h> // 오버플로우인터럽트와 컴페어매치인터럽트를 사용안하면 사용안해도 됨
본 프로그램은 초음파의 HIGH PULSE 감지 후 타이머를 정지하는 형태로 작성하였으므로 생략해도 프로그램이 돌아갈 것이다.	
#include	"lcd.h" // lcd.c의 함수들을 사용하기 위한 함수들이 정의된 헤더파일 선언
#define	RED PA4 //빨간색 LED
#define	WHITE PA5 //흰색 LED
#define	GREEN PA6 //초록색 LED
#define	YELLOW PA7 //노란색 LED
char	lcd_string[2][MAX_LCD_STRING]; //lcd에 입력할 문자열을 저장하는 변수
#define	boundary_dist10 10 //10cm 경계, 전처리기로 선언
unsigned int	distance1; //초음파 1에서 계산할 거리

```

unsigned int distance2; //초음파 2에서 계산할 거리

unsigned int Q; // Q == 1~4 level 1, 2 문제 번호 해당

unsigned int QQ; // QQ == 0: level 1 문제 번호 해당, QQ == 1~4 : level 2 문제 번호 해당

#define Q1 1 //문제 1
#define Q2 2 //문제 2
#define Q3 3 //문제 3
#define Q4 4 //문제 4

void PORT_init() //포트 초기화
{
    //PA4~7 : LED를 출력으로 설정한다.

    DDRA = 0xF0;

    //PB0~1 : 초음파 1 , PB3~4 : 초음파 2 // Trigger를 출력으로, Echo를 입력으로 설정

    DDRB = 0x09; //PORTB = 0x12;
}

void MODE_1_2() // 초음파 1, 2의 왕복 시간을 계산할 타이머 모드 선택
{
    TCCR1A=0x00; TCCR1B=0x08; // CTC mode
    TCCR3A=0x00; TCCR3B=0x08; // CTC mode
}

void Trigger1() //초음파1 트리거 신호
{

```

```
//_delay_ms(1000); //1초후 Trigger 신호 보냄
```

신호를 더 빨리 받기 위해, 최소한으로 delay를 사용하도록 이 부분에서는 delay를 없앴다.

```
PORTB = 0x01; //PB0 : 트리거
```

```
_delay_ms(10);
```

```
PORTB = 0x00; //0.01초후 신호 끝
```

```
}
```

```
void Trigger2() //초음파2 트리거 신호
```

```
{
```

```
//_delay_ms(1000); //1초후 Trigger 신호 보냄
```

```
PORTB = 0x08; //PB3 : 트리거
```

```
_delay_ms(10);
```

```
PORTB = 0x00; //0.01초후 신호 끝
```

```
}
```

```
void Echo1() //초음파센서1 값 // Ehco : PB1
```

```
{
```

```
while(!(PINB&0x02)); // high가 될때까지 대기 //PULSE 시작
```

```
TCNT1=0x00; TCCR1B=0x02; // 카운터 시작, 8분주
```

```
while(PINB&0x02); // low가 될때까지 대기 //PULSE 종료
```

```
TCCR1B=0x08; // 카운터 정지의 목적
```

```
distance1=TCNT1/116; // cm로 변경 // 프로그램 코드 전에 설명 참조
```

```
}
```

```
void Echo2() //초음파센서2 값 // Ehco : PB4
```

```
{
```

```
while(!(PINB&0x10)); // high가 될때까지 대기 //PULSE 시작
```

```

    TCNT3=0x00; TCCR3B=0x02; // 카운터 시작, 8분주

    while(PINB&0x10); // low가 될때까지 대기 //PULSE 종료

    TCCR3B=0x08; // 카운터 정지의 목적

    distance2=TCNT3/116; // cm로 변경 // 프로그램 코드 전에 설명 참조
}

void LED(int led1, int led2) //LED를 동시에 2개 색깔을 ON 시키기 위한 함수
{

    //RED : PA4 , WHITE : PA5 , GREEN : PA6 , YELLOW : PA7

    if(led2==0){ //led2 변수가 0일 경우 한 가지 색깔만 ON

        // led1 == 1, led2 == 0 : 나머지는 다 끄고, 빨간색 LED만 ON

        if(led1==1) {PORTA = ~((1<< WHITE)|(1<< GREEN)|(1<< YELLOW)); PORTA = 1<< RED; }

        // led1 == 2, led2 == 0 : 나머지는 다 끄고, 흰색 LED만 ON

        else if(led1==2){PORTA &= ~((1<< RED)|(1<< GREEN)|(1<< YELLOW)); PORTA |= 1<< WHITE; }

        // led1 == 3, led2 == 0 : 나머지는 다 끄고, 초록색 LED만 ON

        else if(led1==3){PORTA &= ~((1<< RED)|(1<< WHITE)|(1<< YELLOW)); PORTA |= 1<< GREEN; }

        // led1 == 4, led2 == 0 : 나머지는 다 끄고, 노란색 LED만 ON

        else if(led1==4){PORTA &= ~((1<< RED)|(1<< WHITE)|(1<< GREEN)); PORTA |= 1<< YELLOW; }

        // led1 == 5, led2 == 0 : 전체 led 모두 OFF

        else { PORTA &= ~((1<< RED)|(1<< WHITE)|(1<< GREEN)|(1<< YELLOW)); }
    }
}

```



```

    }

    else{ //led2 변수가 0이 아닐 경우 두 가지 색깔의 LED ON

        // led1 == 1, led2 == 2 : 빨간색 LED, 흰색 LED ON

        if(led1==1 && led2==2) {PORTA = ~((1<< GREEN)|(1<< YELLOW)); PORTA = (1<< RED) | (1<<
WHITE); }

        // led1 == 1, led2 == 3 : 빨간색 LED, 초록색 LED ON

        else if(led1==1 && led2==3){PORTA = ~((1<< WHITE)|(1<< YELLOW)); PORTA = (1<< RED) | (1<<
GREEN); }

        // led1 == 4, led2 == 2 : 노란색 LED, 흰색 LED ON

        else if(led1==4 && led2==2){PORTA = ~((1<< GREEN)|(1<< RED)); PORTA = (1<< YELLOW) | (1<<
WHITE); }

        // led1 == 4, led2 == 3 : 초록색 LED, 노란색 LED ON

        else if(led1==4 && led2==3){PORTA = ~((1<< RED) | (1<< WHITE)); PORTA = (1<< GREEN) | (1<<
YELLOW); }

        //else { PORTA &= ~((1<< RED)|(1<< WHITE)|(1<< GREEN)|(1<< YELLOW)); } : 안해줘도 됨

    }

}

void START_SIGN(int level) //LCD에 시작신호를 알린다. // level은 1과 2가 있다.

{

    _delay_ms(3000); LED(5,0); //3초 뒤 다음 level의 시작 신호를 알리기 위한 기다림, LED 모두 OFF

    LCD_command(0x01); // LCD Clear


    if(level == 1) sprintf(lcd_string[0], " LEVEL1 START "); //level 1일 경우 level 1 시작을 알림

```

```

else sprintf(lcd_string[0], " LEVEL2 START "); //level 2일 경우 level2 시작을 알림

LCD_str_write(0, 0, lcd_string[0]); _delay_ms(3000); //lcd에 출력
}

void Question(int position) // LCD에 "*****"를 띄우는 위치를 받아 LCD에 출력
{
    sprintf(lcd_string[0], "*****"); //LCD에 출력할 문자열 저장

    // LCD_command(0x01); //LCD 화면 Clear는 START_SIGN 함수에서 해줌

    if(position==1) LCD_str_write(0, 0, lcd_string[0]); //첫 번째 위치
    else if(position==2) LCD_str_write(0, 8, lcd_string[0]); //두 번째 위치
    else if(position==3) LCD_str_write(1, 0, lcd_string[0]); //세 번째 위치
    else if(position==4) LCD_str_write(1, 8, lcd_string[0]); //네 번째 위치
}

void Rhythm(int Q, int QQ, int speed)
{ // QQ==0인 경우 level 1, QQ==0을 제외한 나머지 숫자인 경우 Level 2

    // speed는 게임의 속도를 조절함. 문제가 제시된 후 speed에 따라 손동작이 얼마나 빨라야하는가가 결정됨

    LCD_command(0x01); // LCD Clear

    if(QQ == 0){ //lcd에 제시되는 문제가 1개일 경우 : "*****"가 한 개만 표시

        Question(Q); _delay_ms(speed); //lcd에 위치(Q가 1,2,3,4 중 하나)에 맞게 LCD 디스플레이

        Trigger1(); Echo1(); //초음파 1의 Trigger 신호를 보내, Echo로 받은 후 거리 계산 완료

        Trigger2(); Echo2(); //초음파 2의 Trigger 신호를 보내, Echo로 받은 후 거리 계산 완료

        if(Q==Q1) //QQ==0, Q==1인 경우 Level 1-(1) 왼손 문제
    }

```

```

{

    //경계 10cm보다 뒤에 왼손 바닥이 위치하면 정답이므로,

    //손바닥이 위치하는 LED ON -> YELLOW ON

    if(distance1 > boundary_dist10) LED(4,0); //왼손 문제이므로 초음파 1과 관련

    //오답일 경우 붉은색 LED를 3번 ON/OFF

    else{ for(int i=0;i<3;i++){ LED(1,0); _delay_ms(100); LED(5,0); _delay_ms(100); } }

}

else if(Q==Q2) //QQ==0, Q==2인 경우 Level 1-(2) 오른손 문제
{

    //경계 10cm보다 뒤에 오른손 바닥이 위치하면 정답이므로,

    //손바닥이 위치하는 LED ON -> GREEN ON

    if(distance2 > boundary_dist10) LED(3,0); //오른손 문제이므로 초음파 2과 관련

    //오답일 경우 붉은색 LED를 3번 ON/OFF

    else{ for(int i=0;i<3;i++){ LED(1,0); _delay_ms(100); LED(5,0); _delay_ms(100); } }

}

else if(Q==Q3) //QQ==0, Q==3인 경우 Level 1-(3) 왼손 문제
{

    //경계 10cm보다 앞에 왼손 바닥이 위치하면 정답이므로,

    //손바닥이 위치하는 LED ON -> RED ON

    if(distance1 < boundary_dist10) LED(1,0);

```

```

        //오답일 경우 붉은색 LED를 3번 ON/OFF

        else{ for(int i=0;i<3;i++){ LED(1,0); _delay_ms(100); LED(5,0); _delay_ms(100); } }

    }

    else if(Q==Q4) //QQ==0, Q==4인 경우 Level 1-(4) 오른손 문제
    {

        //경계 10cm보다 앞에 오른손 바닥이 위치하면 정답이므로,

        //손바닥이 위치하는 LED ON -> WHITE ON

        if(distance2 < boundary_dist10) LED(2,0);

        //오답일 경우 붉은색 LED를 3번 ON/OFF

        else{ for(int i=0;i<3;i++){ LED(1,0); _delay_ms(100); LED(5,0); _delay_ms(100); } }

    }

}

else{ //lcd에 제시되는 문제가 2개일 경우 : “*****”가 2 개 표시

    Question(Q); Question(QQ); _delay_ms(speed); //LCD에 Q와 QQ에 해당하는 문제 “*****” 2개
디스플레이, speed로 문제 빠르기 조절 //Q : 1 or 3(왼손 문제) + QQ= 2 or 4(오른손 문제)

    Trigger1(); Echo1(); //초음파 1의 Trigger 신호를 보내, Echo로 받은 후 거리 계산 완료

    Trigger2(); Echo2(); //초음파 2의 Trigger 신호를 보내, Echo로 받은 후 거리 계산 완료

    if(Q==Q1 && QQ==Q2) // Q==1, QQ==2인 경우 Level 2-(1) 왼손 + 오른손 문제
    {

        //경계 10cm보다 뒤에 왼손 바닥 + 10cm보다 뒤에 오른손 바닥이 위치하면 정답,

        //손바닥이 위치하는 LED ON -> YELLOW, GREEN ON

```

```

        if(distance1 > boundary_dist10 && distance2 > boundary_dist10){ LED(4,3); }

        //오답일 경우 붉은색 LED를 3번 ON/OFF
        else{ for(int i=0;i<3;i++){ LED(1,0); _delay_ms(100); LED(5,0); _delay_ms(100); } }
    }

    else if(Q==Q1 && QQ==Q4) // Q==1, QQ==4인 경우 Level 2-(2) 왼손 + 오른손 문제
    {

        //경계 10cm보다 뒤에 왼손 바닥 + 10cm보다 앞에 오른손 바닥이 위치하면 정답,
        //손바닥이 위치하는 LED ON -> YELLOW, WHITE ON
        if(distance1 > boundary_dist10 && distance2 < boundary_dist10){ LED(4,2); }

        //오답일 경우 붉은색 LED를 3번 ON/OFF
        else{ for(int i=0;i<3;i++){ LED(1,0); _delay_ms(100); LED(5,0); _delay_ms(100); } }
    }

    else if(Q==Q3 && QQ==Q2) // Q==3, QQ==2인 경우 Level 2-(3) 왼손 + 오른손 문제
    {

        //경계 10cm보다 앞에 왼손 바닥 + 10cm보다 뒤에 오른손 바닥이 위치하면 정답,
        //손바닥이 위치하는 LED ON -> RED, GREEN ON
        if(distance1 < boundary_dist10 && distance2 > boundary_dist10){ LED(1,3); }

        //오답일 경우 붉은색 LED를 3번 ON/OFF
        else{ for(int i=0;i<3;i++){ LED(1,0); _delay_ms(100); LED(5,0); _delay_ms(100); } }
    }
}

```

```

else if(Q==Q3 && QQ==Q4) // Q==3, QQ==4인 경우 Level 2-(4) 왼손 + 오른손 문제
{
    //경계 10cm보다 앞에 왼손 바닥 + 10cm보다 앞에 오른손 바닥이 위치하면 정답.
    //손바닥이 위치하는 LED ON -> YELLOW, GREEN ON

    if(distance1 < boundary_dist10 && distance2 < boundary_dist10){ LED(1,2); }

    //오답일 경우 붉은색 LED를 3번 ON/OFF

    else{ for(int i=0;i<3;i++){ LED(1,0); _delay_ms(100); LED(5,0); _delay_ms(100); } }

}

}

int main(void) //MAIN 문
{
    LCD_init(); // LCD 초기화 함수
    PORT_init(); // 포트 초기화 함수
    MODE_1_2(); // 타이머1, 3 모드 선택 초기화 함수

    while(1){

        int speed = 1000; // 문제 제시 후 주어지는 시간 1초 // 게임 스피드 조절

        START_SIGN(1); // level 1 게임 시작 알림, LCD에 디스플레이

```



```
Rhythm(1, 0, speed); // level 1-(1) 문제

Rhythm(2, 0, speed); // level 1-(2) 문제

Rhythm(3, 0, speed); // level 1-(3) 문제

Rhythm(4, 0, speed); // level 1-(4) 문제


START_SIGN(2); // level 2 게임 시작 알림, LCD에 디스플레이


Rhythm(1, 2, speed); // level 2-(1) 문제

Rhythm(1, 4, speed); // level 2-(2) 문제

Rhythm(3, 2, speed); // level 2-(3) 문제

Rhythm(3, 4, speed); // level 2-(4) 문제

}

return 0; // 종료하며 마침

}
```

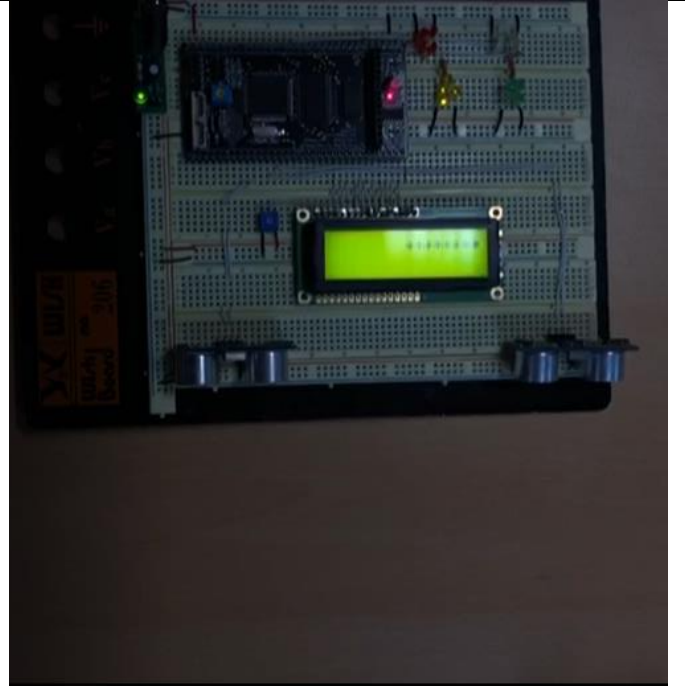
다음 페이지에는 동영상 결과를 캡처하여 결과 사진을 넣었다. 10cm 경계 이상에 손을 가져다 놓는 것은 캡처 사진에 손이 찍히지 않을 수도 있다. 10cm 경계 안에 들어오는 것이면 사진에 손이 찍혀있다.

정확한 결과는 동영상을 통해 확인이 가능하다.

LEVEL 1 시작



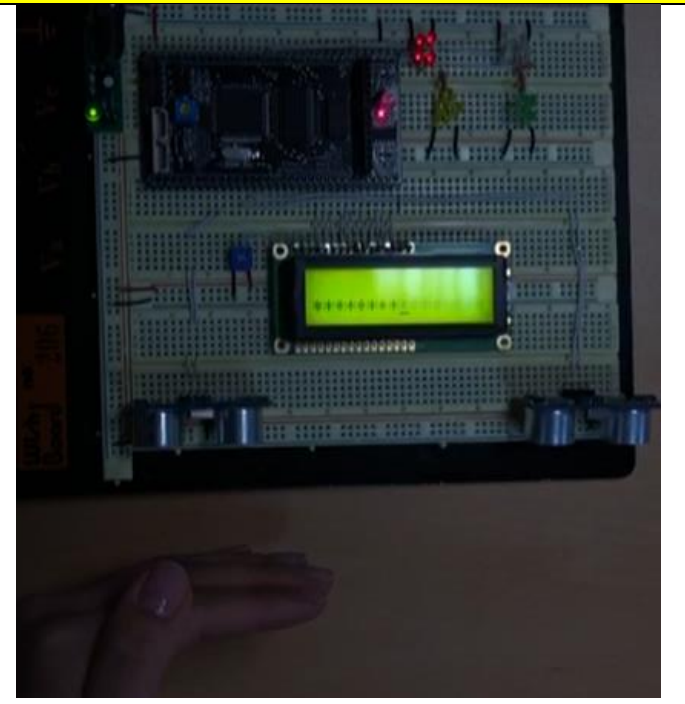
LEVEL 1-(1) 정답 후
→ 노란색 LED ON하면서
다음 문제가 제시된 모습



LEVEL 1-(2) 정답 후
→ 초록색 LED ON 하면서
다음 문제가 제시된 모습

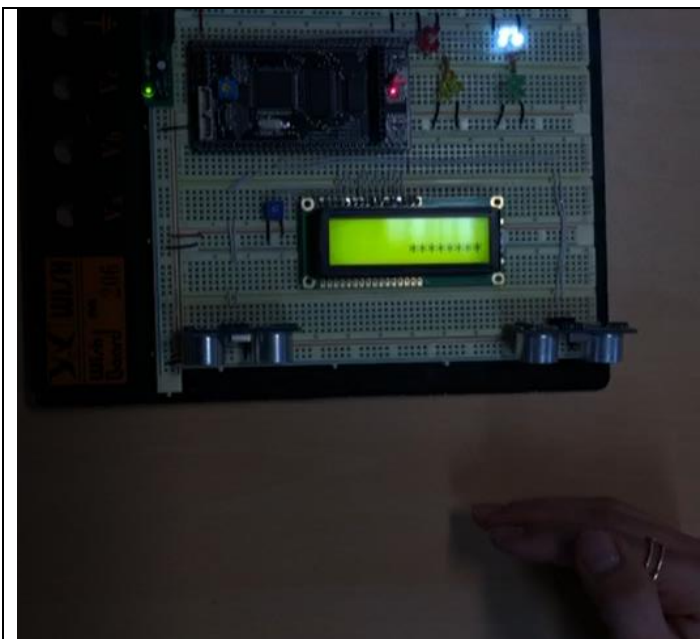


LEVEL 1-(3) 정답 → 빨간색 LED ON



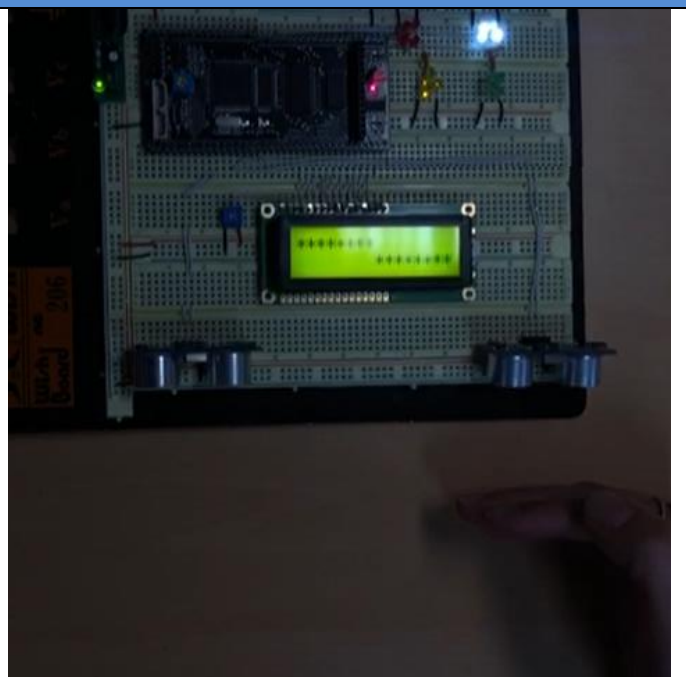
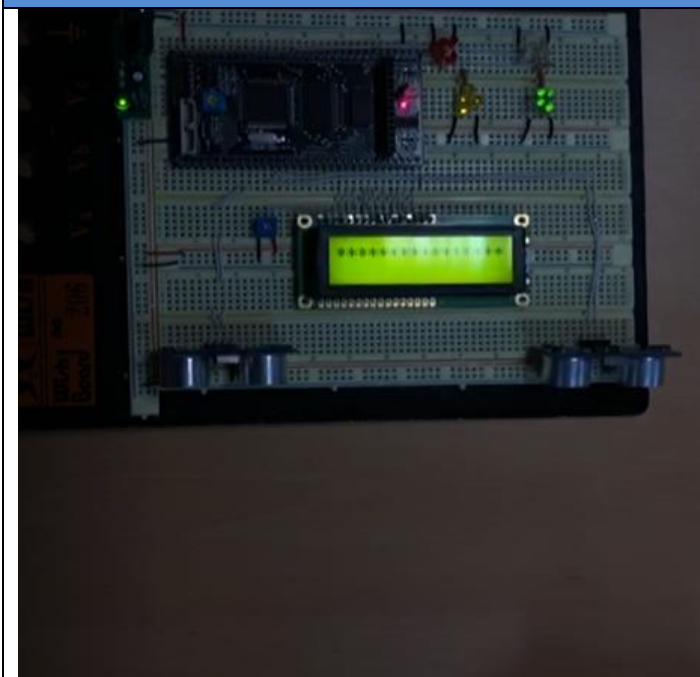
LEVEL 1-(4) 정답 → 흰색 LED ON

LEVEL 2 시작



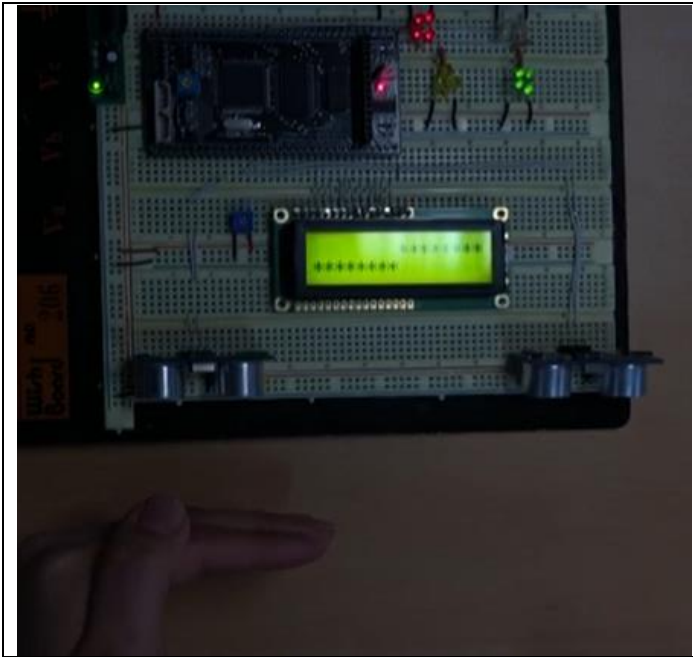
LEVEL 2-(1) 정답
→ 노란색 & 초록색 LED ON

LEVEL 2-(2) 정답
→ 흰색 & 노란색 LED ON

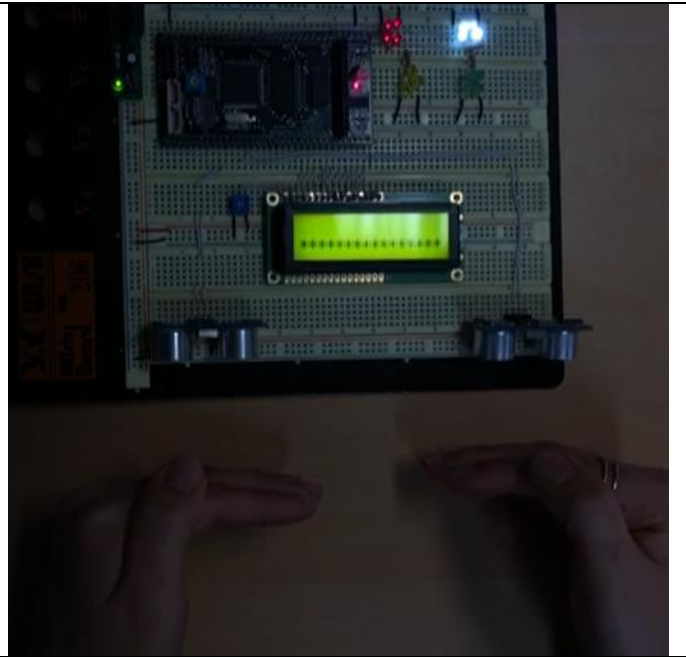


LEVEL 2-(3) 정답
→ 빨간색 & 초록색 LED ON

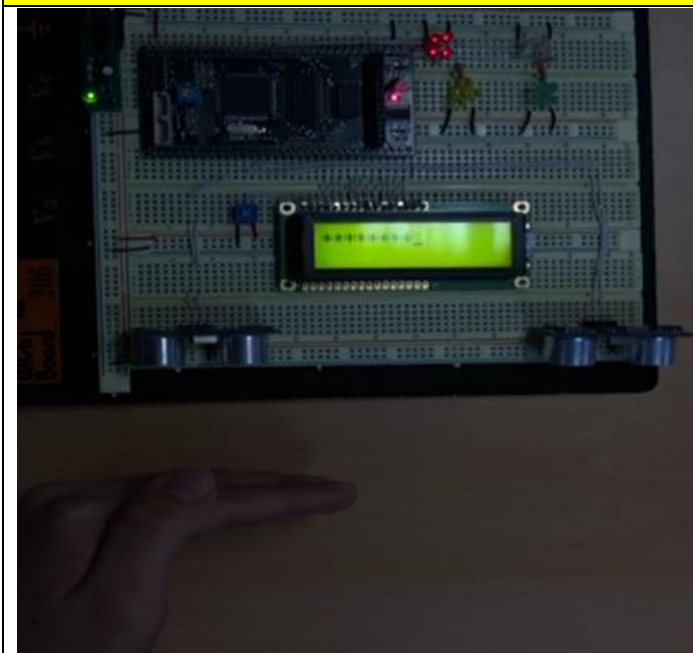
LEVEL 2-(4) 정답
→ 빨간색 & 흰색 LED ON



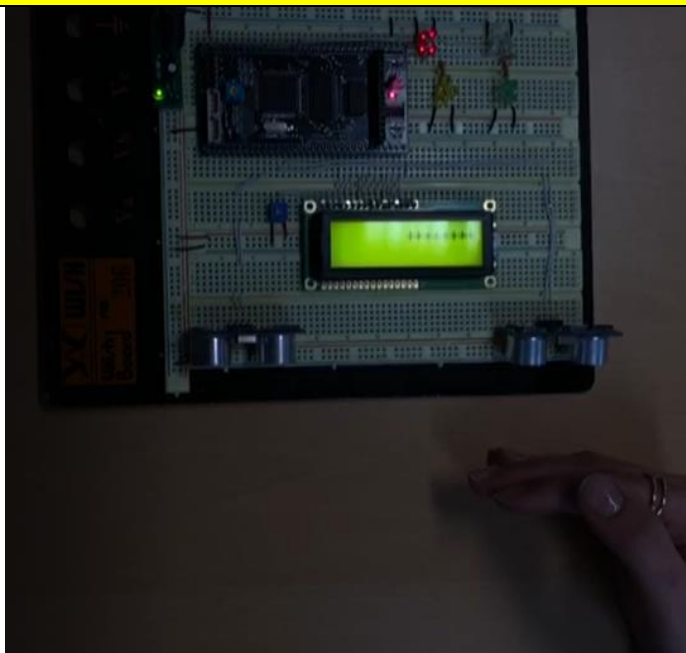
LEVEL 1-(1) 오답
→ 빨간색 LED ON/OFF 3번



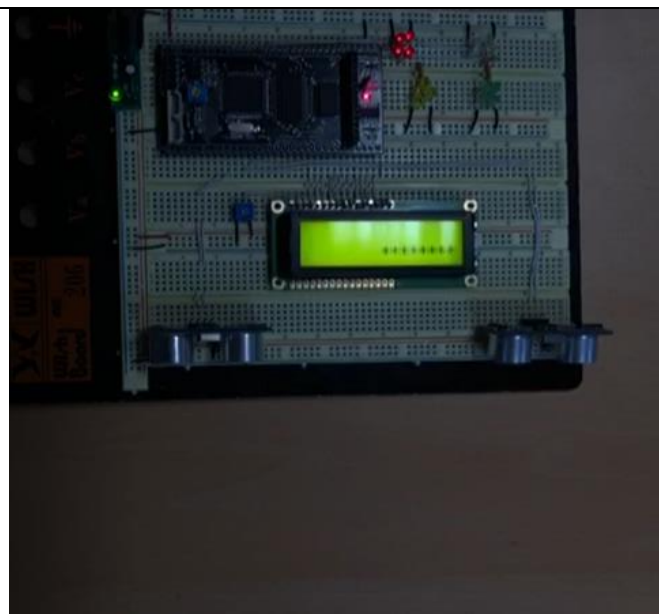
LEVEL 1-(2) 오답
→ 빨간색 LED ON/OFF 3번



LEVEL 1-(3) 오답
→ 빨간색 LED ON/OFF 3번

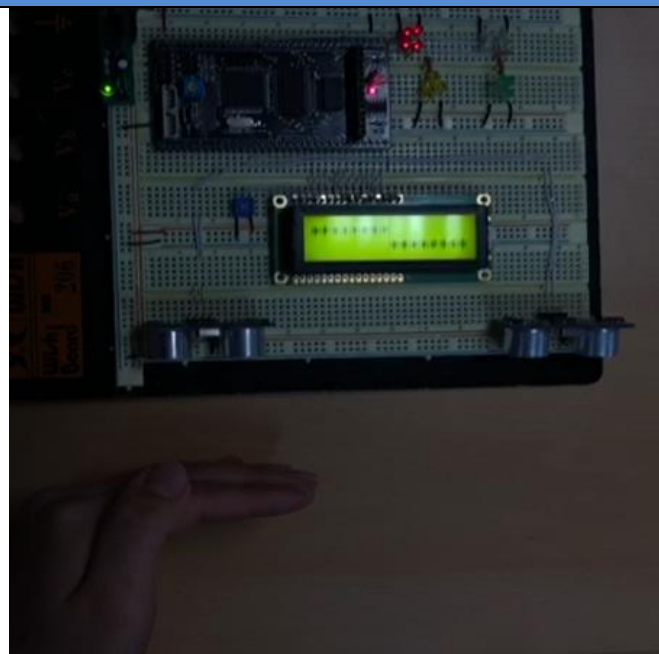
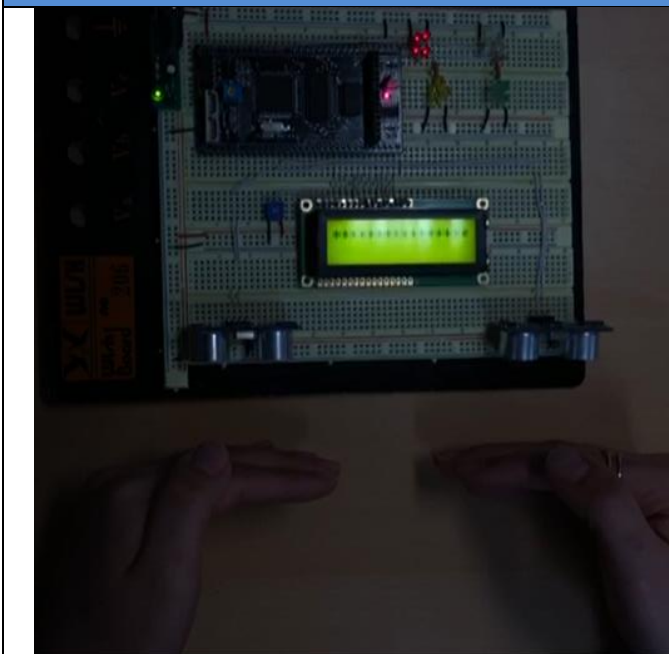


LEVEL 1-(4) 오답
→ 빨간색 LED ON/OFF 3번



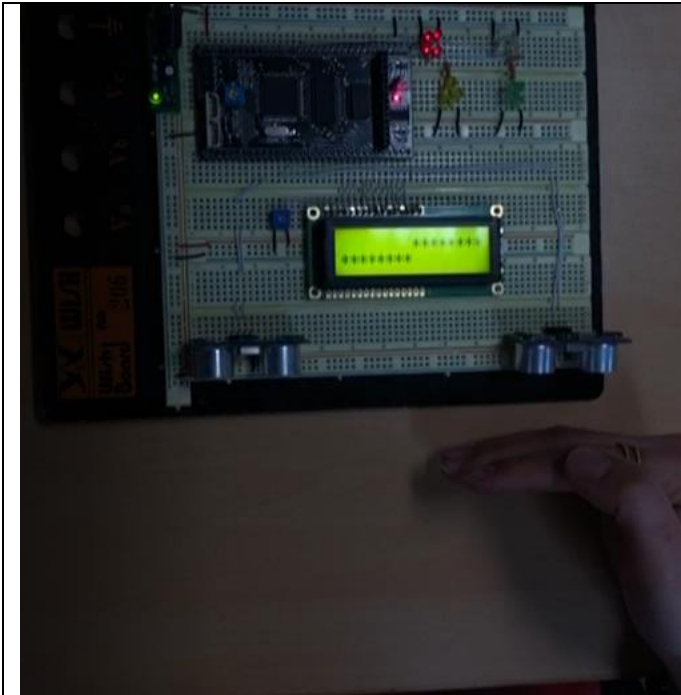
LEVEL 2-(1) 오답
→ 빨간색 LED ON/OFF 3번

LEVEL 2-(2) 오답
→ 빨간색 LED ON/OFF 3번



LEVEL 2-(3) 오답
→ 빨간색 LED ON/OFF 3번

LEVEL 2-(4) 오답
→ 빨간색 LED ON/OFF 3번



마지막 경우는 동영상에서
정답으로 해서 오답 사진이 없습니다.