마이크로프로세서
프로젝트 결과보고서

2020.6

참여학생

20171588 전재문

20171550 양석현

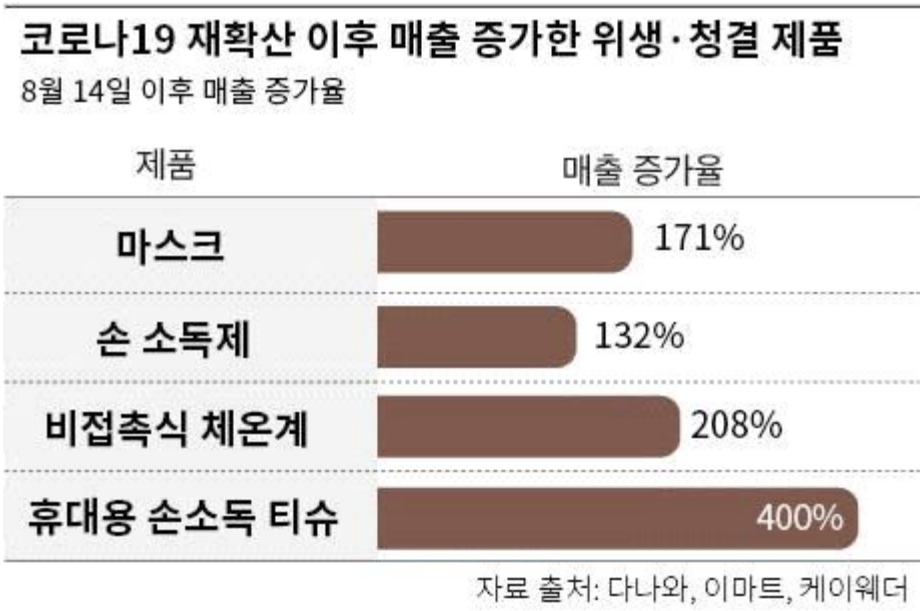
20171578 이현민

20171599 조현준

컴퓨터공학부(IoT트랙)

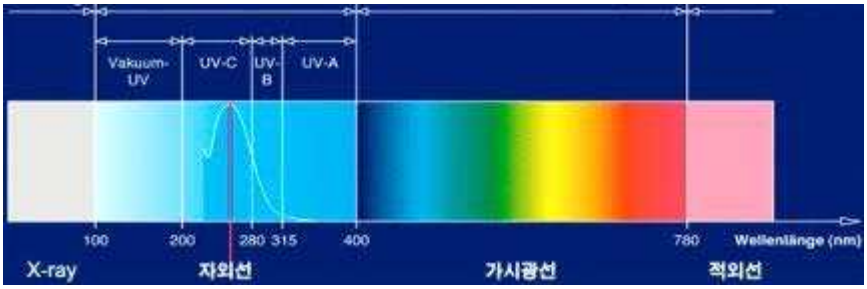
1) 프로젝트 필요성

코로나 19 확산 이후 많은 사람이 위생과 청결에 대한 관심이 높아졌습니다. 따라서 마스크와 손 세정제 등 위생 및 청결 제품의 매출이 증가하게 되었습니다. 아래는 가격비교 사이트 에누리닷컴의 8월 14일 기준 전후 3주간 매출 분석 결과입니다.



마스크는 171%, 손 소독제는 132%, 비접촉식 체온계는 208%, 휴대용 손소독 티슈는 400% 증가했습니다. 위 자료와 같이 국민의 건강과 위생의 관심이 높아졌지만 가까이 있음에도 불구하고 등잔 밑이 어둡다는 말이 있는 것처럼 위생 관리가 되지 않고 있는 부분이 있었습니다. 바로 핸드폰입니다. 세계보건기구(WHO) 발표에 따르면 코로나 19 바이러스는 짧게는 4시간, 길게는 7일 가량 생존이 가능합니다. 구리 표면은 4시간, 마분지는 24시간, 스테인리스와 플라스틱은 72시간, 마스크는 최장 7일간 생존할 수 있다고 합니다. 핸드폰에는 코로나 바이러스가 평균 3~4일가량 생존할 수 있다고 합니다. 연구팀은 핸드폰에 있는 바이러스를 제거하기 위해서는 "핸드폰을 항상 자주 닦아주는 것이 좋다" 고 권고했습니다. 하지만 바쁜 현대인들에게 작은 기기를 매일 닦아주는 것보다 더 나은 방법이 필요했습니다.

그중 하나가 UV 살균 방식입니다. 먼저 자외선(Ultraviolet Light)이란 태양광선 중 가시광선의 가장 짧은 파장인 보라색 안쪽으로 더 파장이 짧아 눈에 보이지 않는 광선을 의미합니다. 자외선은 파장이 짧아 투과력은 약하지만 강력한 에너지를 가지고 있어 화학반응을 촉진하고 유기물을 산화시키며 미생물들에게는 살균작용을 일으킵니다. 양지에 계속 세워둔 자동차가 변색되고 이불을 일광소독하는 것도 이러한 자외선의 특성이 작용하여 가능한 일입니다. 자외선은 아래처럼 분류됩니다.



UV-A Light (315nm - 400nm)

Black Light 라고도 하며 실내에서 선탠 (Sun Tan)을 하거나 푸른 조명을 할 때 사용됩니다.

UV-B Light (280nm - 315nm)

Dorno선이라고 부르며 비타민D를 형성하거나 피부에 홍반현상을 일으킵니다.

UV-C Light (200nm - 280nm)

Germicidal(살균)선이라고 하여 살균 작용이외에 산소를 오존으로 바꾸기도 합니다.

254nm - 살균전용파장, 오존파괴작용

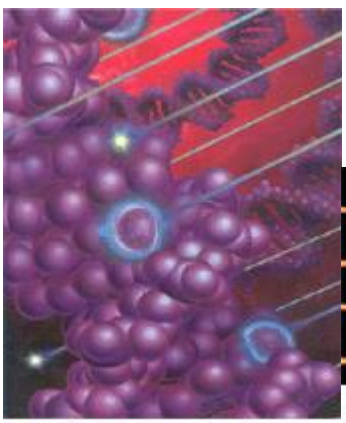
진공자외선

파장이 짧아 투과력이 극히 약하며 우주공간에서 존재하는 자외선입니다.

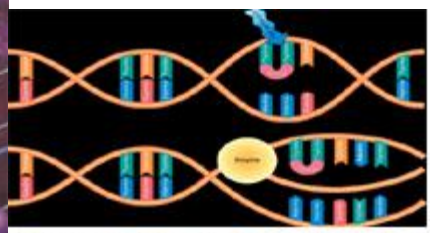
185nm - 오존생성파장 (TOC 제거용)

자외선이 DNA에 조사되면 DNA의 염기 중 티민의 분자구조가 집중적으로 파괴됩니다. 자외선을 흡수한 티민은 이웃한 티민이나 시토신과 눌러붙게 됩니다. 이와 같이 티민이 중합되면 DNA의 복제가 제대로 이루어질 수 없기 때문에 생명체로서의 기능이 정지되는 것입니다.

세균들마다 자외선에 대한 민감성이 차이가 나는 것은 DNA속에 포함된 티민의 양이 다르기 때문입니다. 이외에도 자외선은 세포막을 이루는 인지질과 단백질을 산화시켜 세균들의 생명활동이 연장되지 못하도록 합니다.

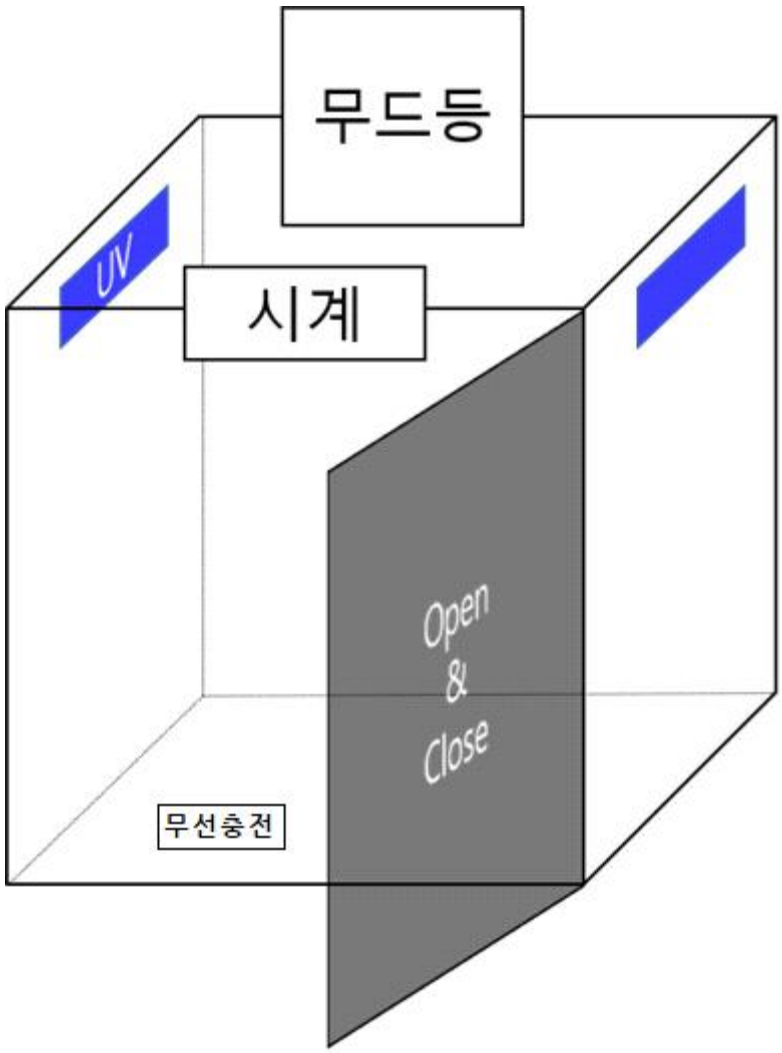


자외선에의한 DNA파괴

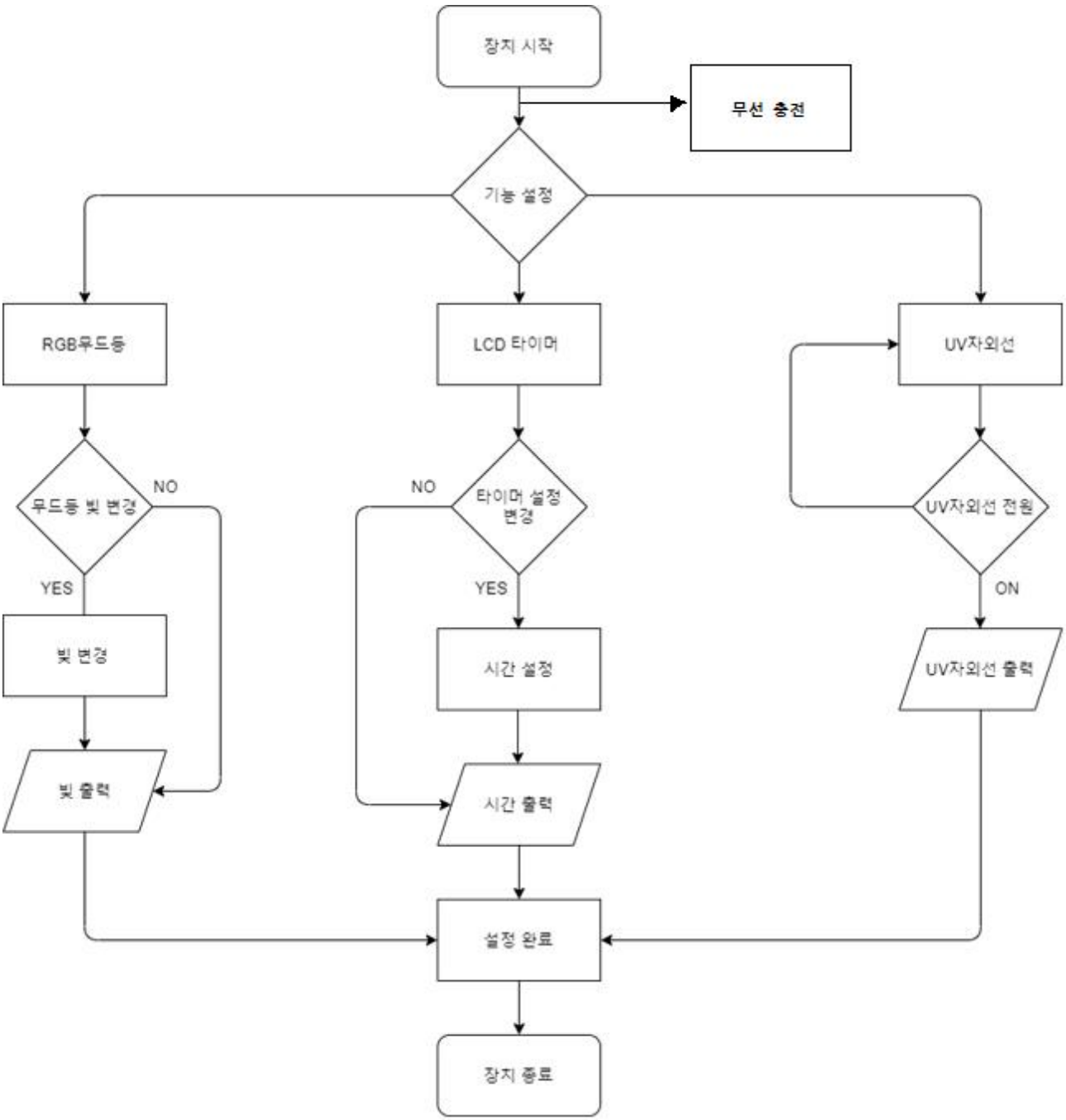


자외선에 의해 파괴된 DNA에서의 티민염기 변화

결론적으로 바이러스로부터 안전하게 하고 건강과 위생을 지키기 위해 핸드폰을 UV 살균하는 방식을 택하게 되었습니다. 그리고 단순하게 살균이라는 기능으로 제품이 만들어진다면 친근하게 다가오지 않는다고 생각하여 무선 충전과 시계, 그리고 무드등의 기능을 수행하는 제품을 고안하게 되었습니다.



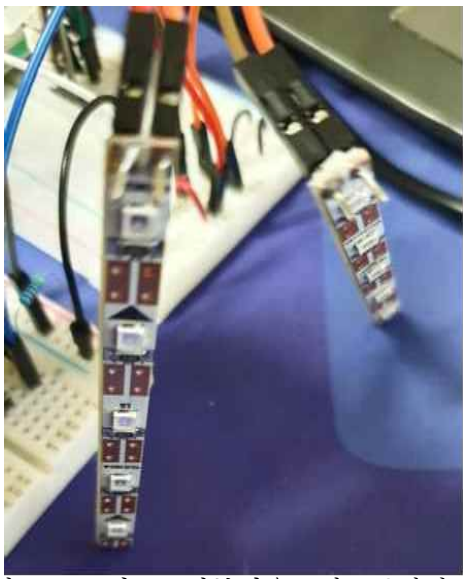
2) 동작순서도


장치 시작
무선 충전
기능 설정
RGB무드등
LCD 타이머
UV자외선
무드등 빛 변경
NO
YES
빛 변경
빛 출력
NO
타이머 설정 변경
YES
시간 설정
시간 출력
UV자외선 전원
ON
UV자외선 출력
설정 완료
장치 종료

3) 준비 부품

센서모듈	센서명	역할
8 BIT 선형 네오픽셀 LED	WS2812B-8	다양한 색을 표현하는 무드등
UV LED바	[그린맥스] 5V UV LED바	UV 살균을 위한 LED
LCD	LC1621-TRNS6 캐릭터 LCD 16x2	시계구현을 위한 LCD
택트 스위치	Tactile Switches	각종 모듈의 제어 스위치
무선 충전 모듈	XJW-W13C	무선 충전

8 BIT 선형 네오픽셀 LED입니다. 네오픽셀은 LED의 한 종류입니다. 네오픽셀은 전원과 제어용 데이터 핀만으로 많은 LED를 제어할 수 있는 특징이 있습니다. 네오픽셀은 모양에 따라 여러 가지 타입의 네오픽셀이 존재하는데 그 중 선형 타입을 선택하여 사용하였습니다.



UV 항균, UV건조, 피부미용, 세균증식억제 등의 효과가 있는 UV-a 파장대의 자외선 LED바입니다. 5cm 단위로 원하는 길이만큼 자유롭게 재단할 수 있습니다.

캐릭터 LCD 16x2입니다. 16x2 LCD 디스플레이는 각각 줄에 16개씩 32개의 문자를 표현할 수 있습니다. 0 ~ 255사이의 ASCII코드의 문자라면 전부 표현 할 수 있습니다.

영어 알파벳의 대소문자와 숫자 그리고 특수 문자를 표현 가능합니다.

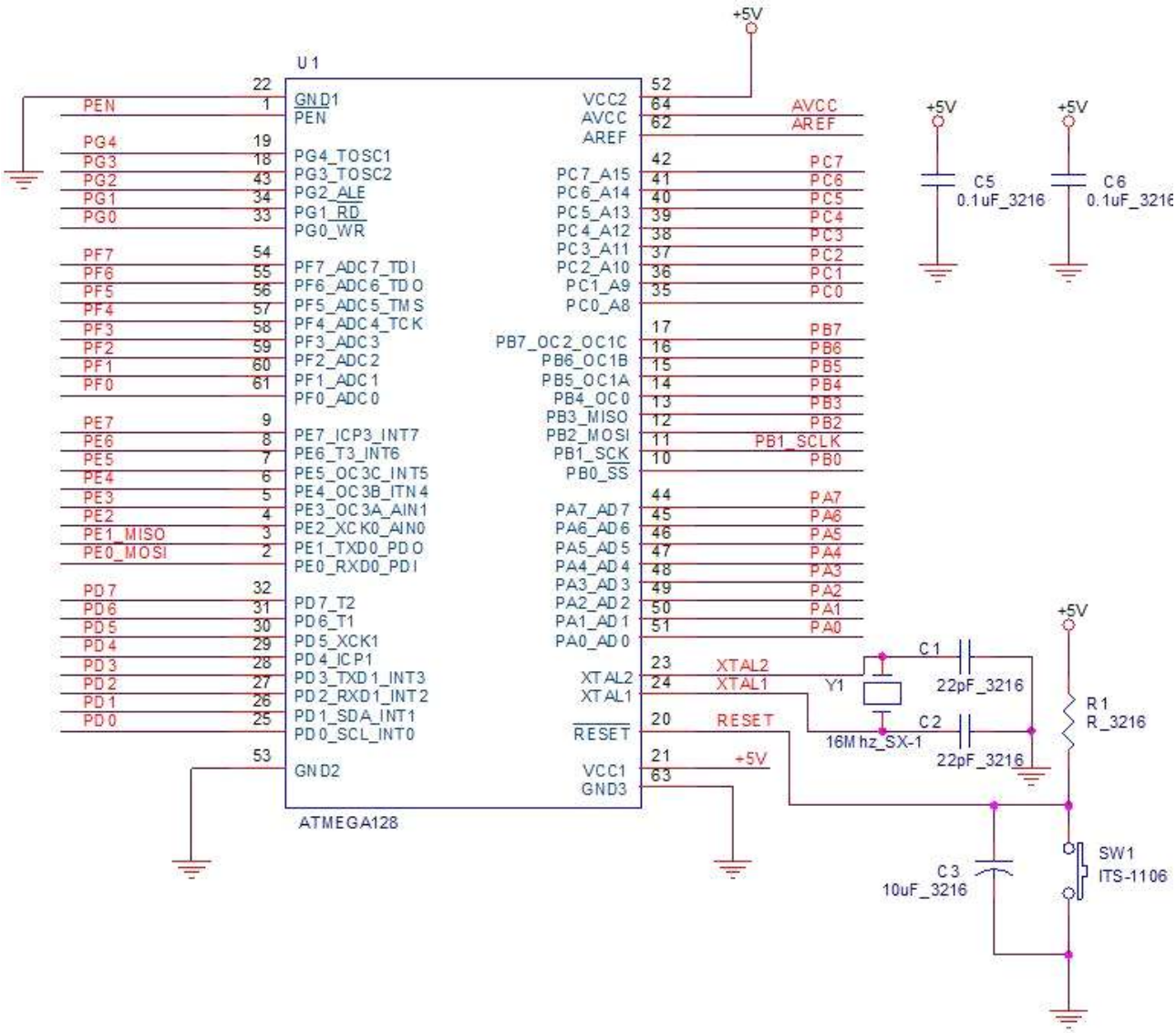
4핀 택트 스위치입니다. 스위치는 거의 모든 전자제품에 필수적으로 들어가는 부품입니다. 전원을 ON/OFF 하거나 사용하고자 하는 기능의 ON/OFF에 주로 쓰입니다. 택트 스위치는 손으로 누르면 ON되고 떼면 OFF 되는 가장 일반적인 택트 스위치입니다.

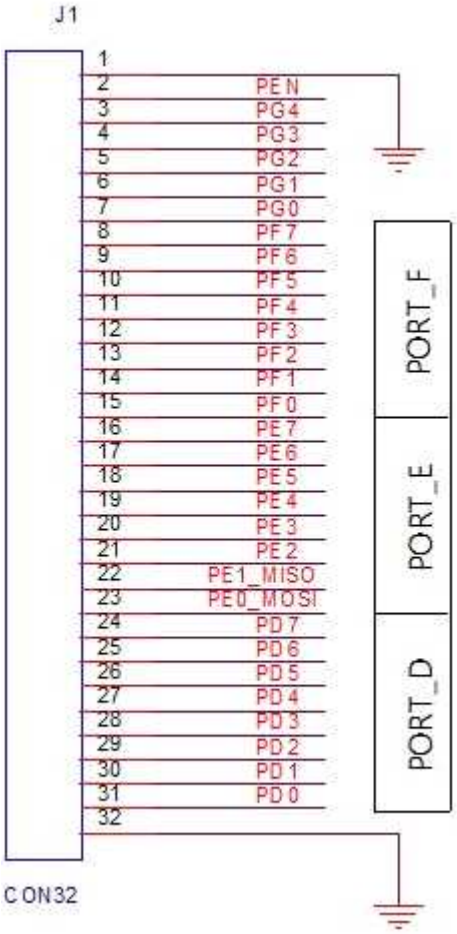
무선 충전 모듈은 다음 페이지 설명 참고

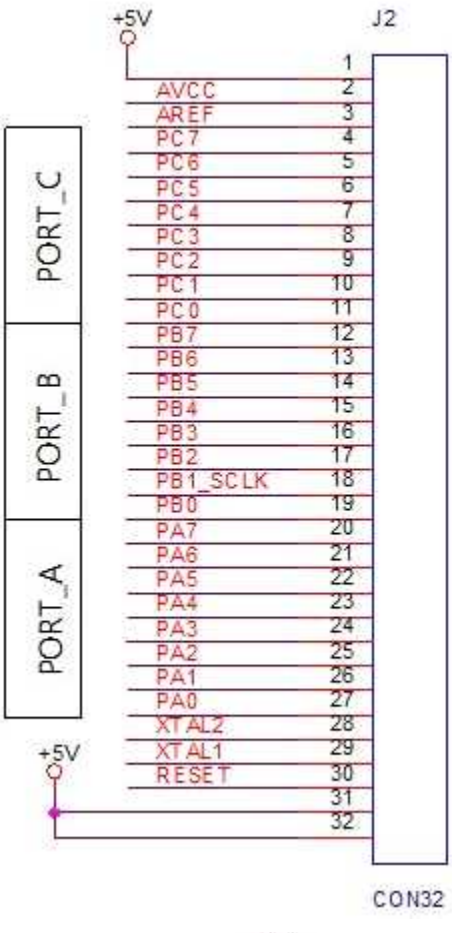
4) 핀 맵

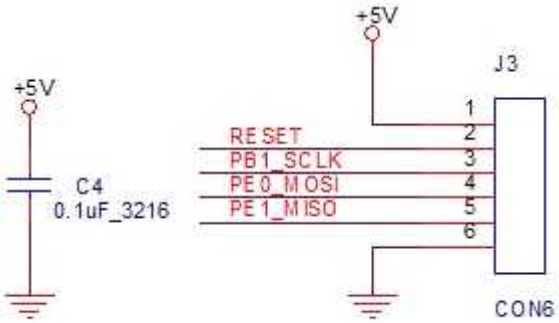
센서모듈	순번	역할	포트번호
8 BIT 선형 네오픽셀 LED	1	다양한 색을 표현하는 무드등	PB7
UV LED바	2	UV 살균을 위한 LED	PC7
LCD	3	시계구현을 위한 LCD	PA0~2, PA4~7
택트 스위치	4	각종 모듈의 제어 스위치	PD0~3

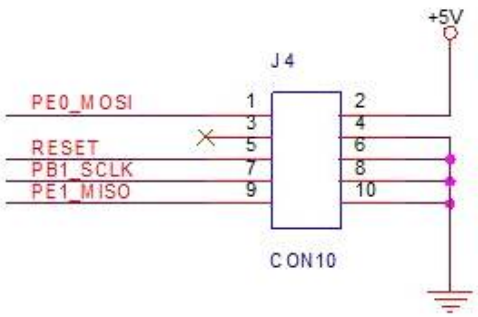
0. ATmega128 MCU 브레드보드 회로도

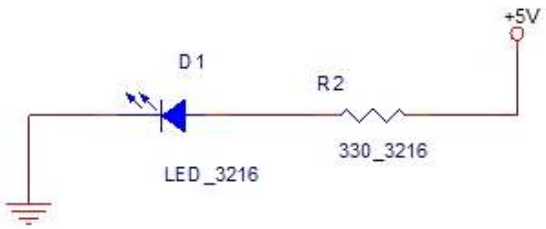



J1
1
2 PEN
3 PG4
4 PG3
5 PG2
6 PG1
7 PG0
8 PF7
9 PF6
10 PF5
11 PF4
12 PF3
13 PF2
14 PF1
15 PF0
16 PE7
17 PE6
18 PE5
19 PE4
20 PE3
21 PE2
22 PE1_MISO
23 PE0_MOSI
24 PD7
25 PD6
26 PD5
27 PD4
28 PD3
29 PD2
30 PD1
31 PD0
32
PORT_F
PORT_E
PORT_D
CON32


+5V
J2
1
AVCC 2
AREF 3
PC7 4
PC6 5
PC5 6
PC4 7
PC3 8
PC2 9
PC1 10
PC0 11
PB7 12
PB6 13
PB5 14
PB4 15
PB3 16
PB2 17
PB1_SCLK 18
PB0 19
PA7 20
PA6 21
PA5 22
PA4 23
PA3 24
PA2 25
PA1 26
PA0 27
XTAL2 28
XTAL1 29
RESET 30
31
32
+5V
PORT_C
PORT_B
PORT_A
CON32


+5V
+5V
J3
C4
0.1uF_3216
1
RESET 2
PB1_SCLK 3
PE0_MOSI 4
PE1_MISO 5
6
CON6


+5V
J4
PE0_MOSI 1
3
RESET 5
PB1_SCLK 7
PE1_MISO 9
2
4
6
8
10
CON10


+5V
D1
R2
330_3216
LED_3216

1. 8 BIT 선형 네오픽셀 LED

GND : Ground

DIN : Control data signal input

4-7VDC : Power supply LED

PIN : PB7 - DIN 연결

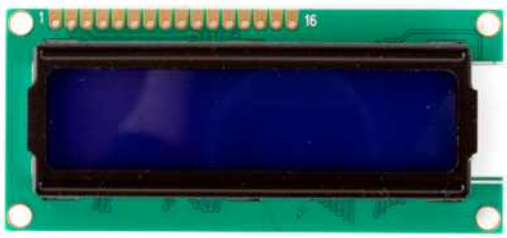
2. LED : UV LED바

GND : Ground

VCC : Power supply LED

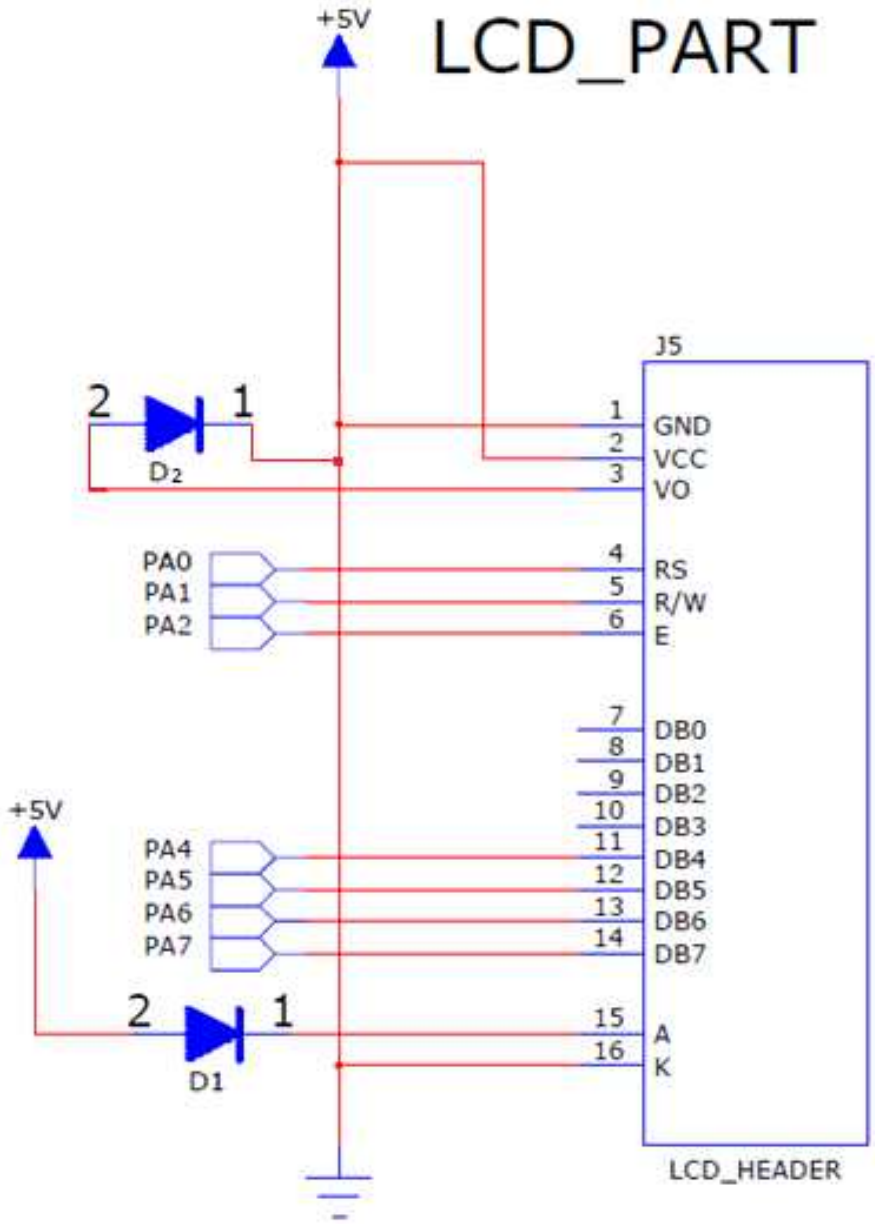
PIN : PC7 - VCC 연결

3. LCD

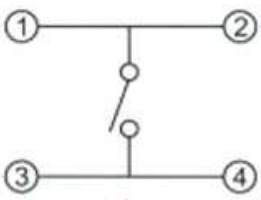


핀번호	표시	설명
1	VSS	· GND
2	VDD	· VCC · LCD 전원 · 5V
3	VO	· VEE · LCD 밝기(명암)제어 · 글자 대비 값 · 가변저항을 통해 0 ~ 5V 사이 입력

4	RS	· Register Select · 레지스터 선택 · 명령어를 처리할지 데이터를 처리할지를 선택합니다. 텍스트 LCD를 제어하기 위해서는 제어레지스터와 데이터레지스터의 두 개 레지스터가 사용되며, RS 신호는 명령을 담고 있는 레지스터(RS = LOW)와 데이터를 담고 있는 레지스터(RS = HIGH) 중 하나를 선택하기 위해 사용합니다.
5	RW	· Read/Write · 읽기/쓰기모드 선택 · 쓰기모드 시 GND로 고정시킵니다. · 읽기(R/W = HIGH) 및 쓰기(R/W = LOW)모드 선택을 위해 사용합니다. · 일반적으로 LCD는 데이터를 쓰기 위한 용도로만 사용하므로, R/W 신호는 GND에 연결하는 것이 일반적이다.
6	E	· Enable · 쓰기 모드 활성화 · 데이터 전송 시작 · 하강에지(falling edge)에서 LCD 드라이버가 데이터 처리를 시작하도록 지시하기 위한 신호로 사용됩니다.
7~14	D0~D7	· 데이터신호 · 데이터 입출력 핀 · 8비트 모드로 동작하는 경우에는 8개가 모두 사용되지만 4비트 모드로 동작하는 경우에는 4개만 사용됩니다. · 마이크로컨트롤러와 사용하는 경우에는 연결핀의 수를 줄이기 위해 4비트 모드를 사용하는 경우가 일반적입니다.
15	A	· 백라이트 LED + · 배경의 밝기 전압 입력 · 3.3V에 연결하거나 5V에 연결시 220Ω 저항을 사용합니다.
16	K	· 백라이트 LED - · 배경 밝기 · GND

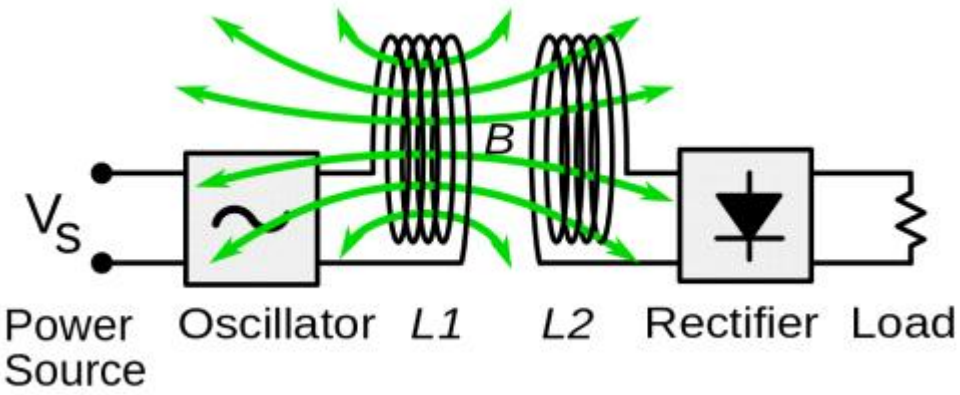

LCD_PART
+5V
J5
2
1
D2
1
GND
2
VCC
3
VO
PA0
PA1
PA2
4
RS
5
R/W
6
E
7
DB0
8
DB1
9
DB2
10
DB3
11
DB4
12
DB5
13
DB6
14
DB7
+5V
PA4
PA5
PA6
PA7
2
1
D1
15
A
16
K
LCD_HEADER

4. 택트 스위치


1
2
3
4

5. 무선 충전 모듈

충전패드와 스마트폰에는 모두 코일이 들어가 있습니다. 송신부인 충전패드에는 전자기장을 발생시키기 위한 1차 코일이, 수신부인 스마트폰은 유도전류를 수신하기 위한 2차 코일이 내장되어 있습니다. 충전패드를 전원에 연결하면 코일에 전류가 흘려 자기장을 발생합니다. 이 자기장이 스마트폰의 코일에 유도전류를 발생시킵니다. 이렇게 전자기 유도 현상에 의해 스마트폰의 코일에 발생한 전류로 스마트폰을 충전하는 것입니다.



자기유도 방식은 현재 최대 4cm로 전송 거리가 짧습니다. 그리고 송신부와 수신부의 코일을 정확히 맞추지 않으면 충전이 잘되지 않습니다. 그래서 단순히 선만 없을 뿐 충전 중에는 유선보다 더 불편합니다. 하지만 자기유도 방식은 대전력 전송에 유리하고, 충전효율이 뛰어나며, 전자파가 거의 발생하지 않아 인체에 무해합니다. 코일의 공진 주파수와 전송 주파수가 달라 소형화가 가능하다. 그래서 대부분 스마트폰이 자기유도 방식을 사용하고 있습니다.

출처
http://mvo.co.kr/%EC%8A%A4%EB%A7%88%ED%8A%B8%ED%8F%B0-%EB%AC%B4%EC%84%A0%EC%B6%A9%EC%A0%84-%EC%9B%90%EB%A6%AC%EB%8A%94-%EB%AC%B4%EC%97%87%EC%9D%BC%EA%B9%8C/

5) 시나리오 구현 방법

프로그램 시작

1. USB 포트 연결 또는 아답터를 연결하여 장치에 전원을 공급한다.
2. 전원이 공급되면 자동으로 프로그램이 시작된다.

LCD 시계

1. 프로그램이 시작되면 LCD에 표시되는 시간은 0시 0분 0초로 출력된다.
2. 시간이 정상적으로 흐른다.

시간 변경 스위치

1. 프로그램이 시작되고 시간이 정상적으로 흐른다면 시간 변경 스위치를 작동시킬 수 있다.
2. 시간 변경 스위치1을 누르면 시간이 멈추며 시간을 설정하는 화면으로 변경되고 초의 첫 번째 자리가 깜빡이기 시작한다. 그리고 시간 변경 스위치2를 누를 수 있게 된다.
3. 시간 변경 스위치2를 누르면 커서가 깜빡이는 자리의 값이 1씩 증가하게 된다.
4. 시간 변경 스위치1을 계속 누르게 되면 깜박이고 있던 커서가 왼쪽으로 한 칸씩 이동하며 초, 분, 시간 순서대로 변경할 자리를 선택하게 된다. 커서가 끝까지 가고 한 번 더 스위치를 누르면 시간 설정 화면을 빠져나와 다시 시간이 흐르게 된다.

무드등

1. 프로그램이 시작되면 무드등은 대기 상태로 있다.
2. 무드등 스위치를 누르면 기본 값 색상의 빛이 나오기 시작한다.
3. 스위치를 한 번 더 누르게 되면 무드등의 색을 변경한다.
4. 마지막에는 무드등이 대기 상태로 돌아간다.

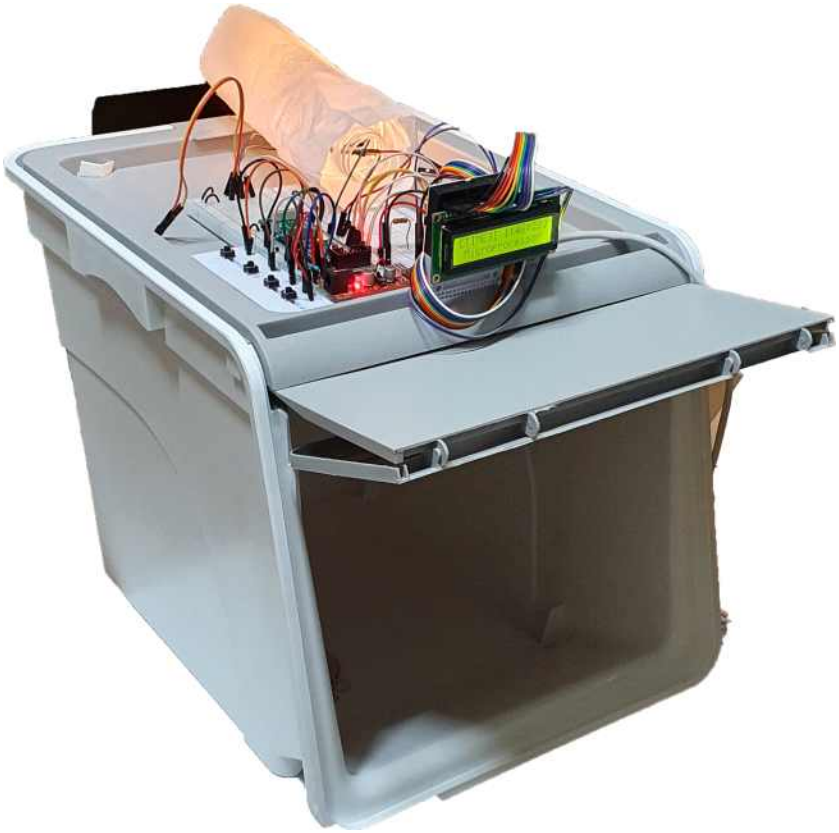
UV 살균

1. 프로그램이 시작되면 UV 살균 시스템은 대기 상태로 있다.
2. 작동 스위치를 누르면 상자 내부에 있는 UV LED가 작동되어 푸른 빛을 내며 살균을 시작한다.
3. 다시 스위치를 누르면 LED가 꺼지고 대기 상태로 들어간다.

무선 충전

1. 상자 내부에 있는 무선 충전 모듈에 스마트폰을 올려놓으면 무선 충전을 시작한다.
2. UV 살균과 무선 충전을 동시에 할 수 있다.

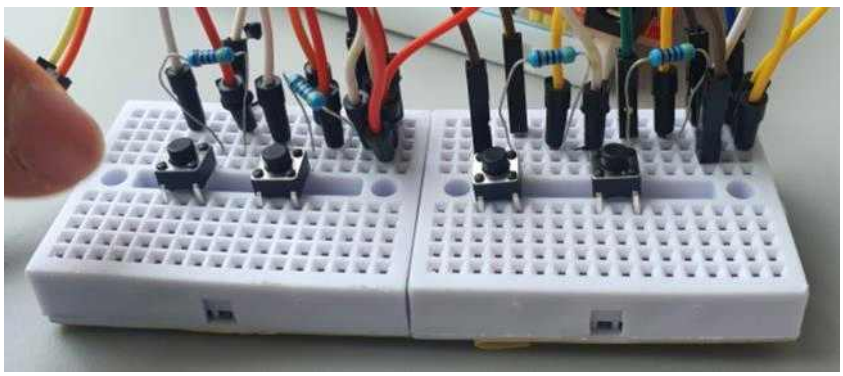
6) 실험결과



본 제품은 다양한 기능이 탑재되어 있습니다. 따라서 각 파트별로 설명하겠습니다. 제어 스위치, LCD 시계, 무드등, UV 살균, 무선 충전 순으로 설명하겠습니다. 추가로 오실로 스코프 측정 결과도 함께 설명하겠습니다.

1. 제어 스위치

먼저 각 모듈의 기능을 제어하는 택트 스위치 사진입니다.

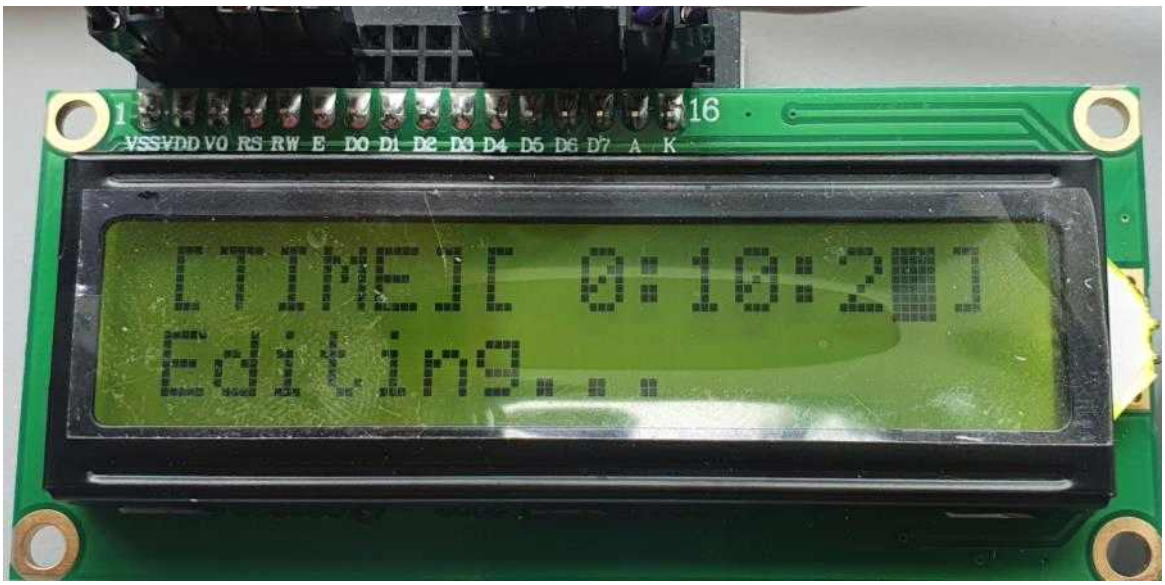


왼쪽부터 차례대로 시간 설정 스위치1, 시간 설정 스위치2, 무드등 스위치, UV LED 스위치입니다.

2. LCD 시계

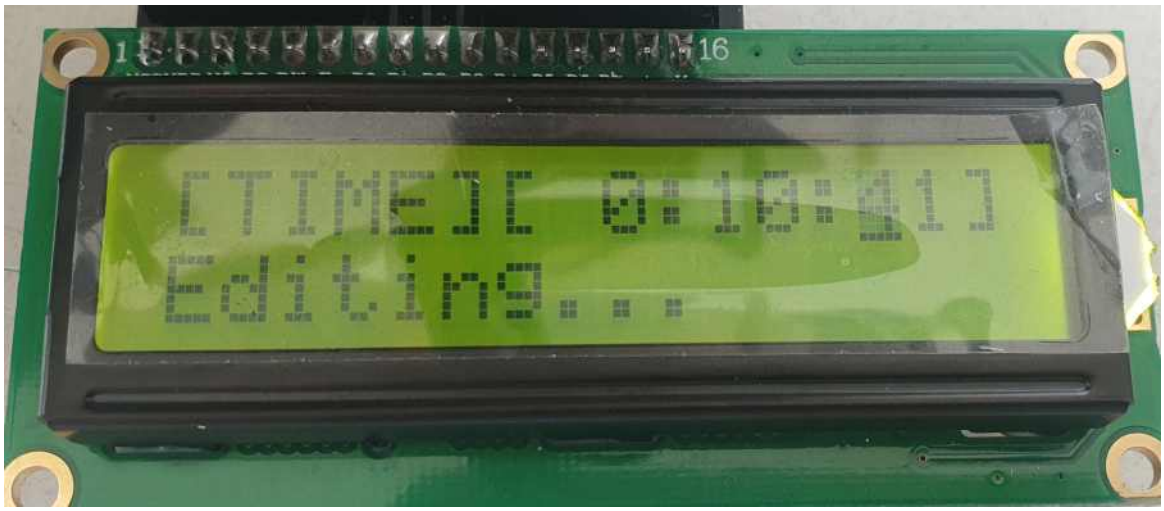
LCD 시계 기능 작동 사진

시간 설정 스위치1을 눌렀을 때 사진



시간 설정 스위치1을 한 번 더 눌렀을 때 사진

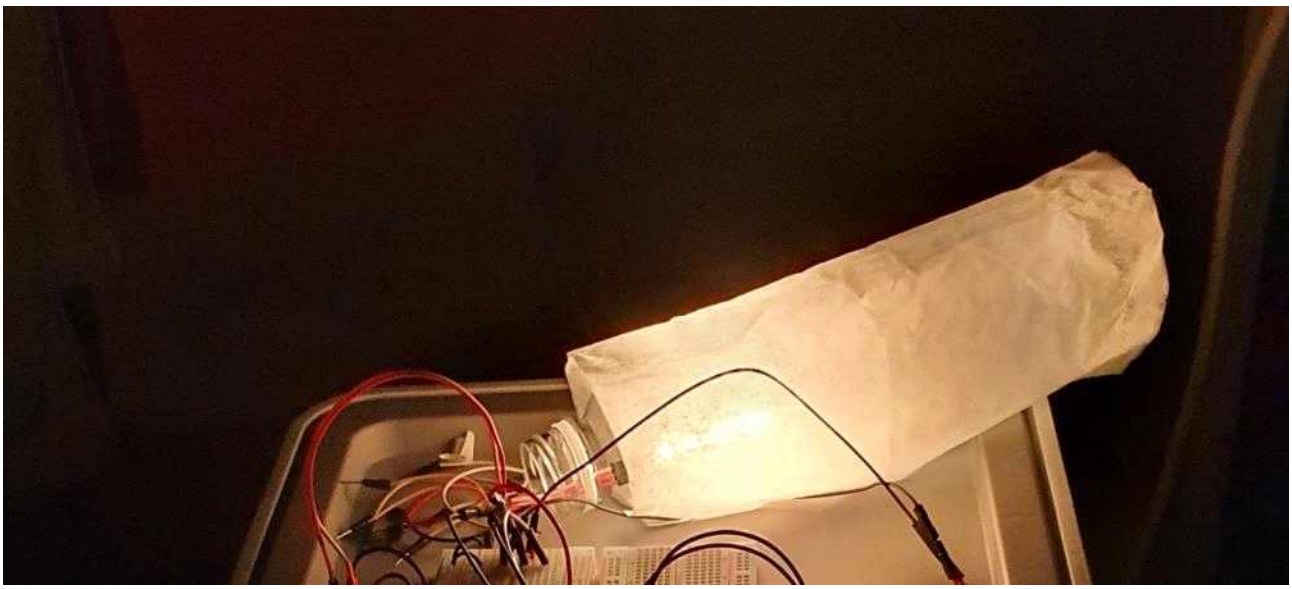
시간 설정 스위치2를 눌렀을 때 사진



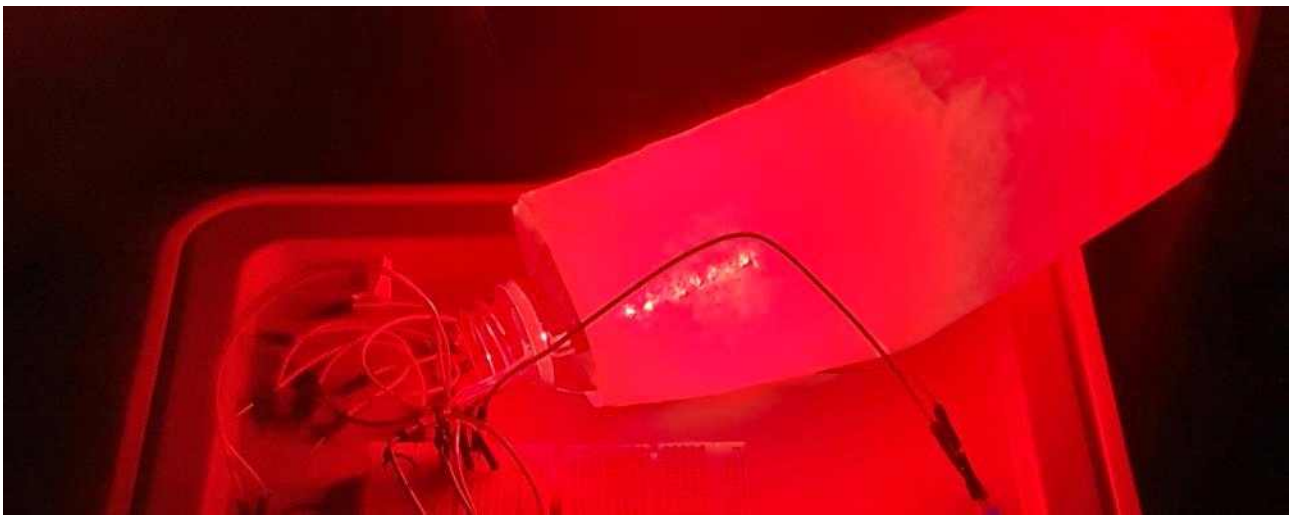
3. 무드등

무드등의 상태는 스위치를 누를 때마다 DEFAULT LIGHT - RED - GREEN - BLUE - PURPLE - PINK - OFF로 순환됩니다.

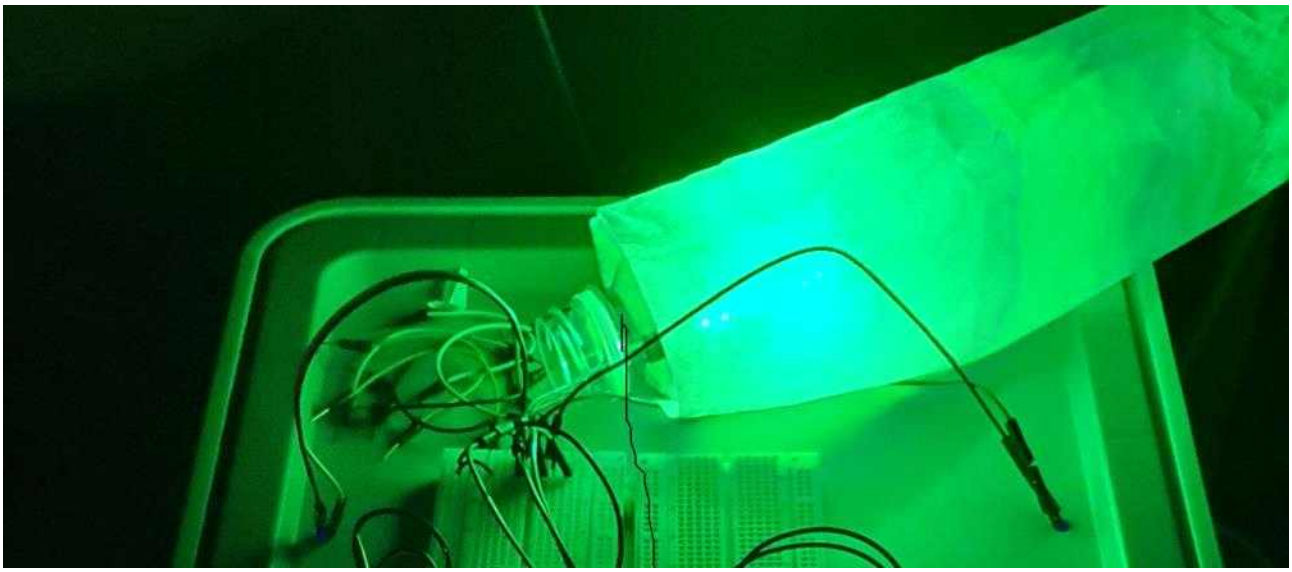
DEFAULT LIGHT 사진



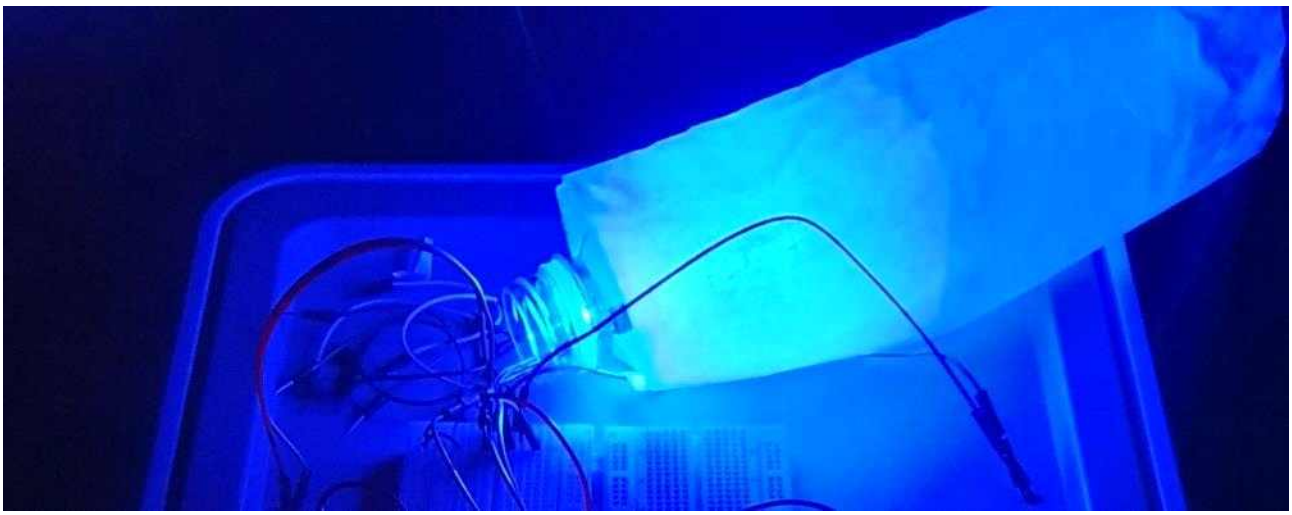
RED 사진



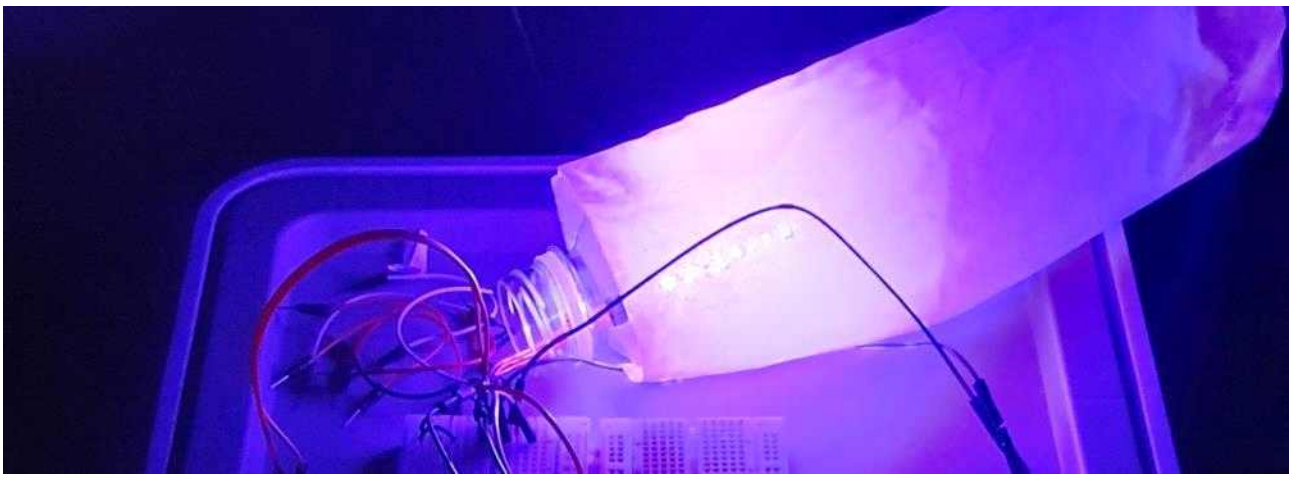
GREEN 사진



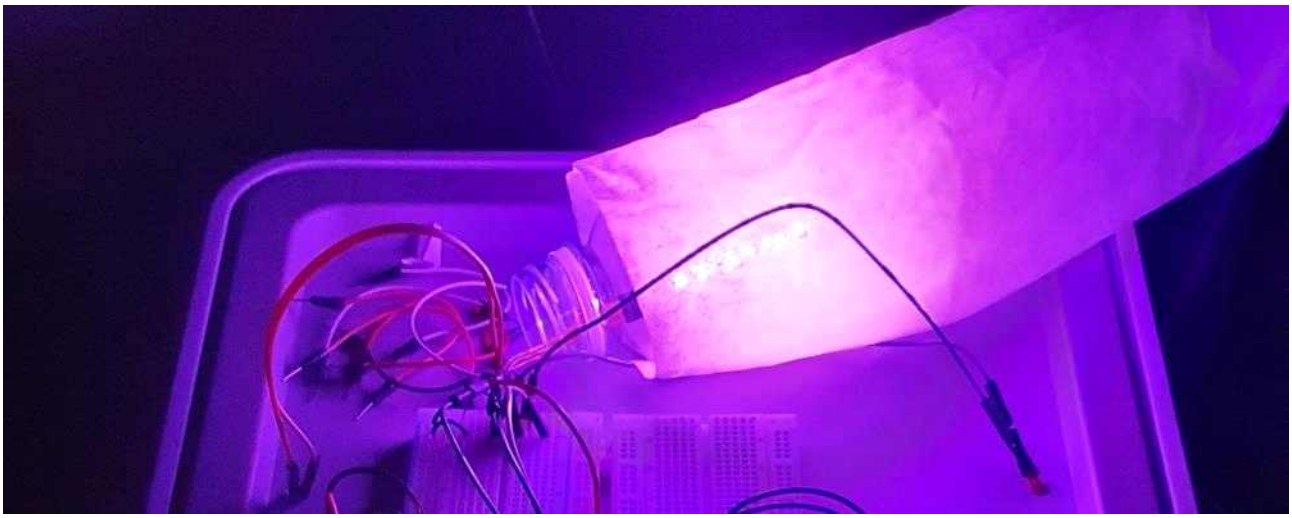
BLUE 사진



PURPLE 사진



PINK 사진



4. UV 살균

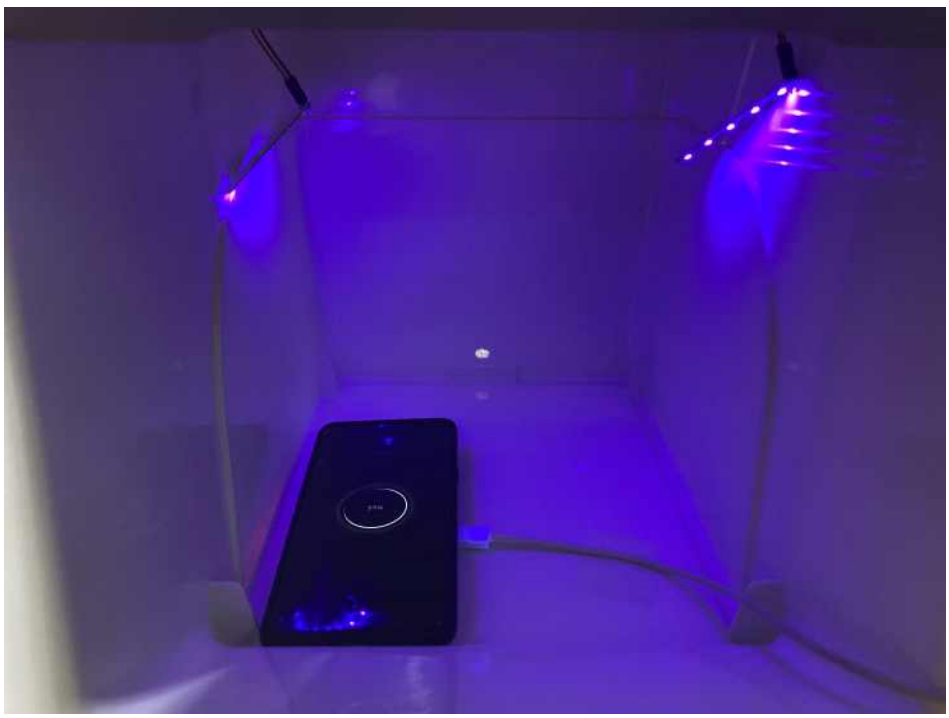
대기 상태(OFF) 사진

UV 살균 시작(ON) 사진

5. 무선 충전

대기 상태

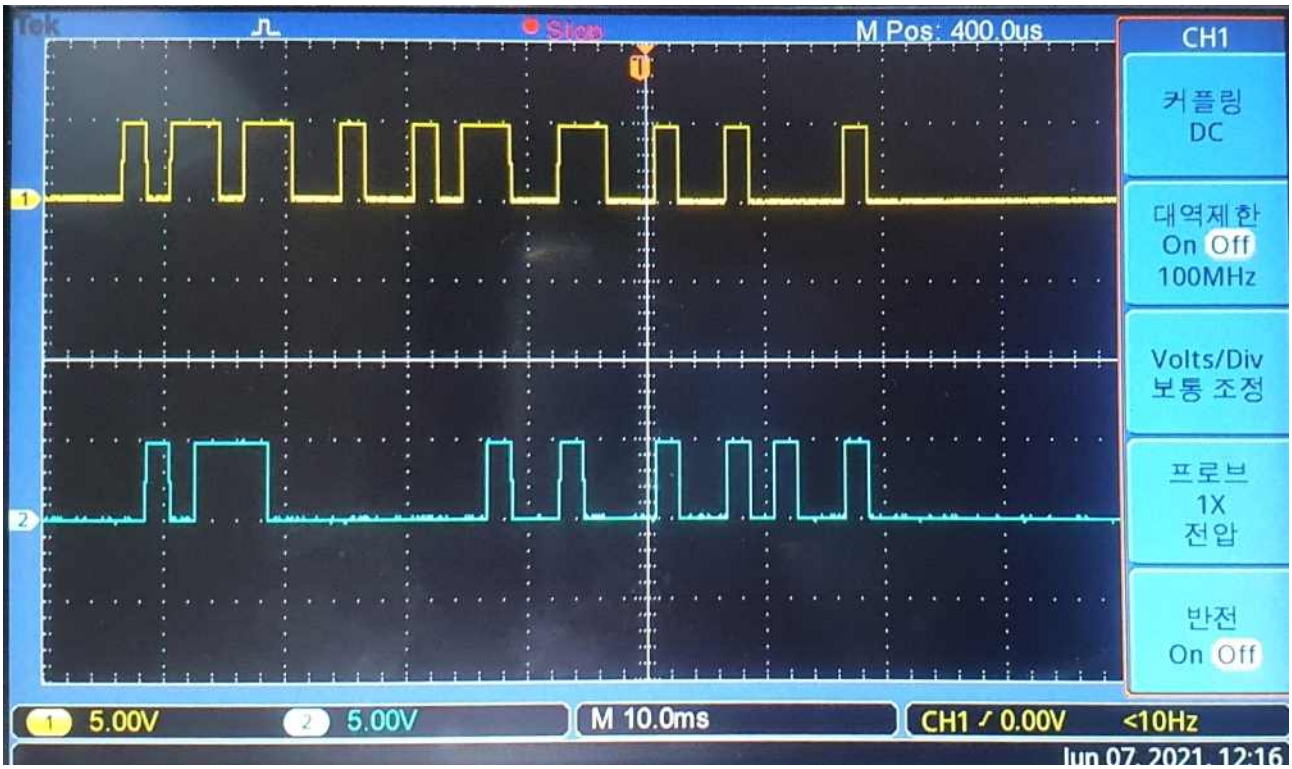
충전 시작



6. 오실로 스코프 측정 결과

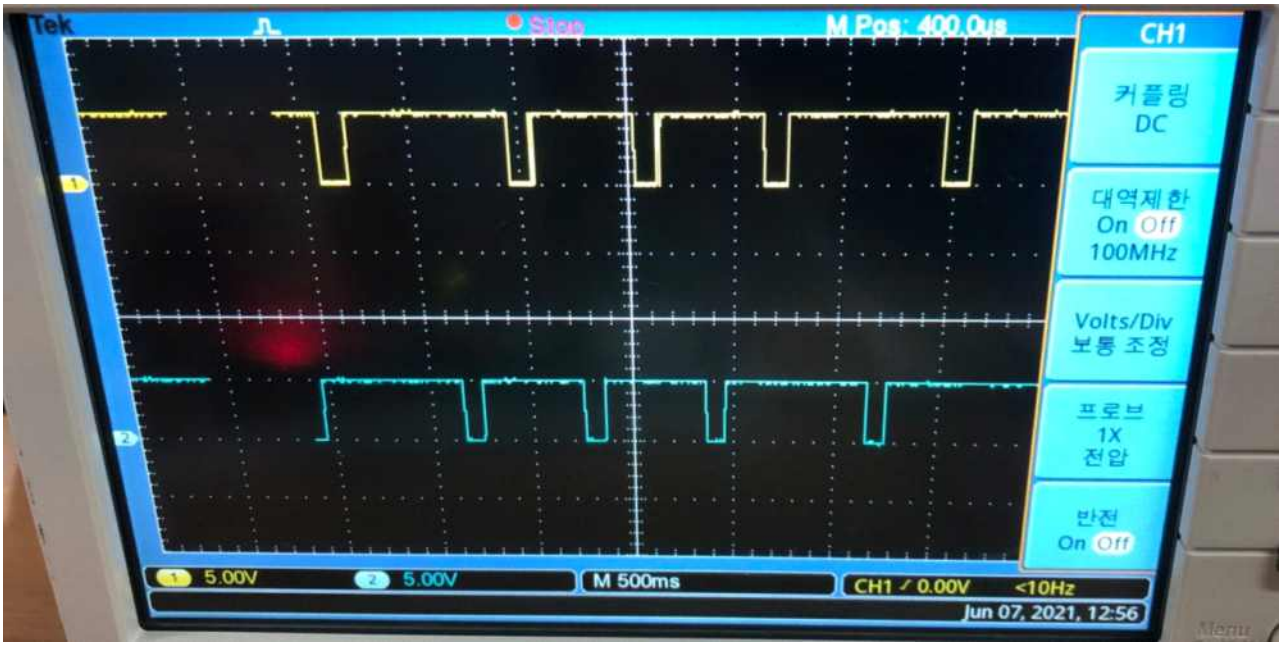
- LCD D4~D7 측정 (시계)

CH1 - D5, CH2 - D4



CH1 - D7, CH2 - D6

- 시간 설정 스위치1, 2 측정

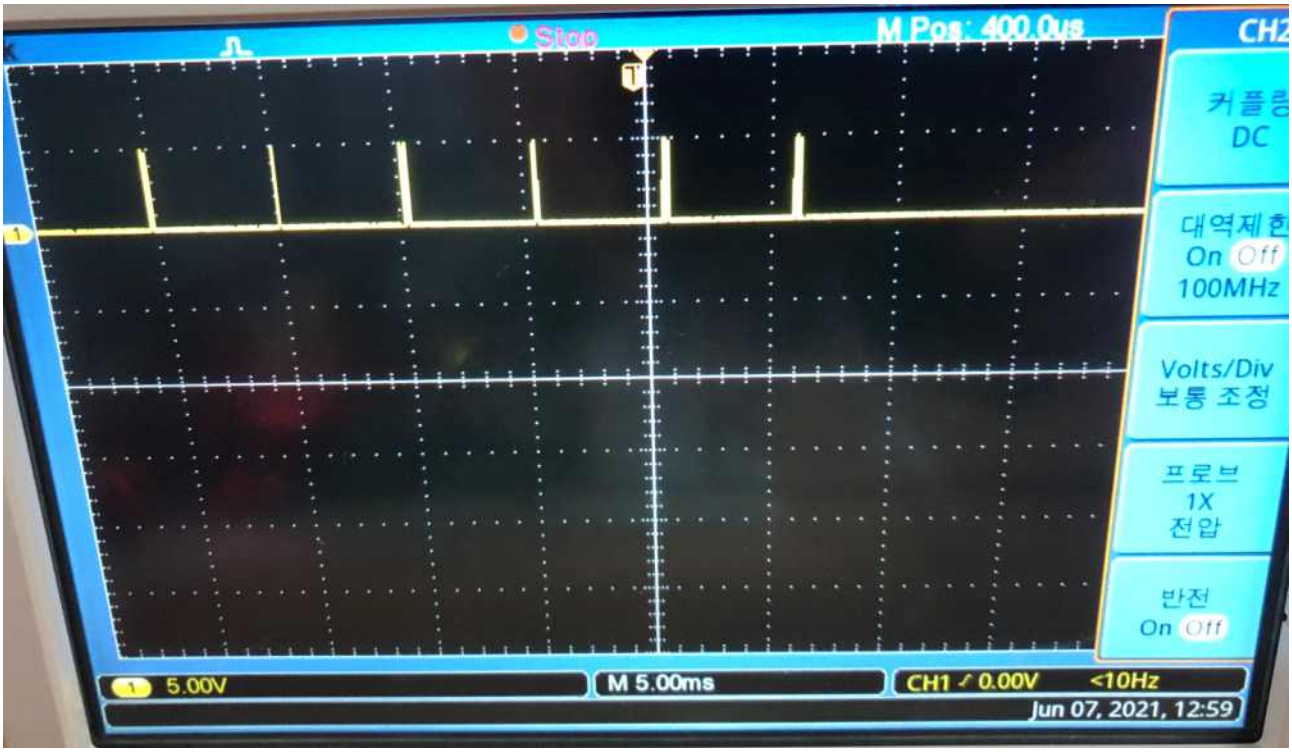


CH1 - 시간 설정 스위치1

CH2 - 시간 설정 스위치2

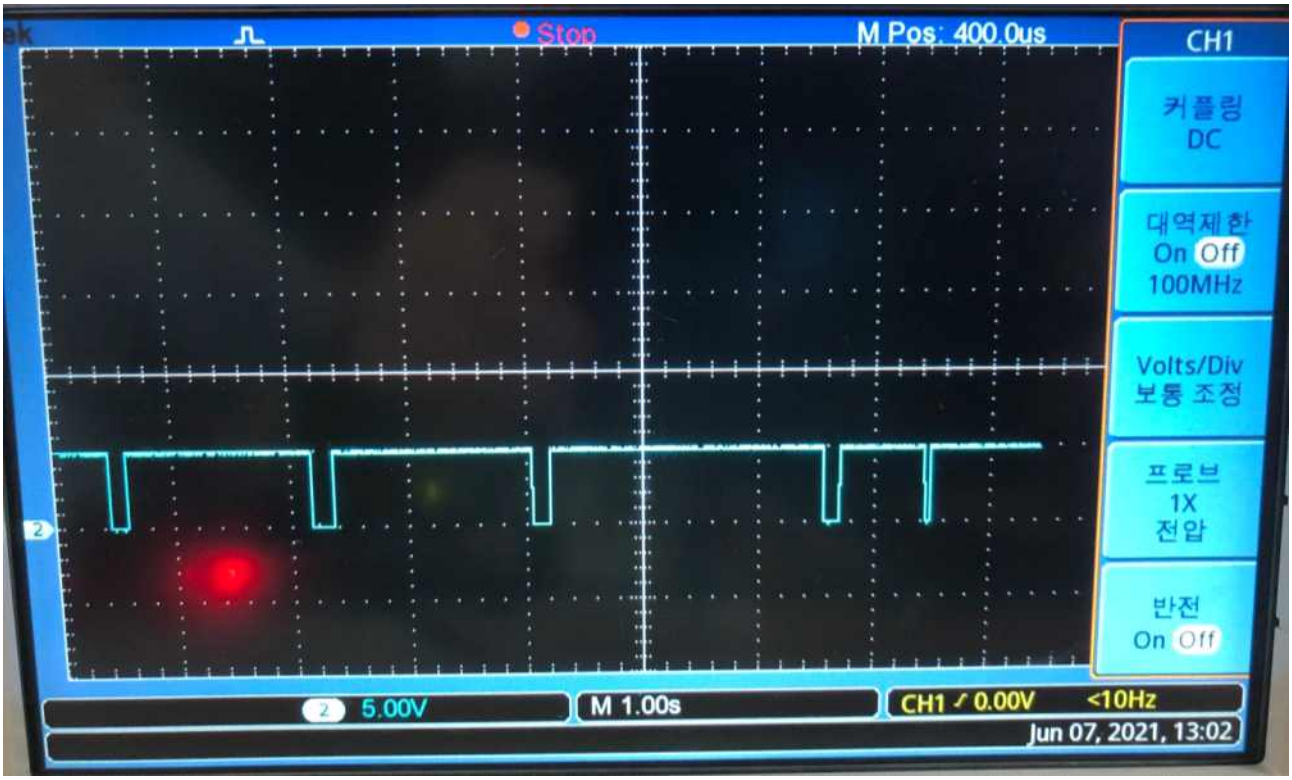
LCD 핀 특성상 스위치로 인한 데이터 변경을 측정하기 어려움. 따라서 독립적인 신호만 측정하였음.

- RGB LED 네오픽셀 (무드등)



CH1 - DIN

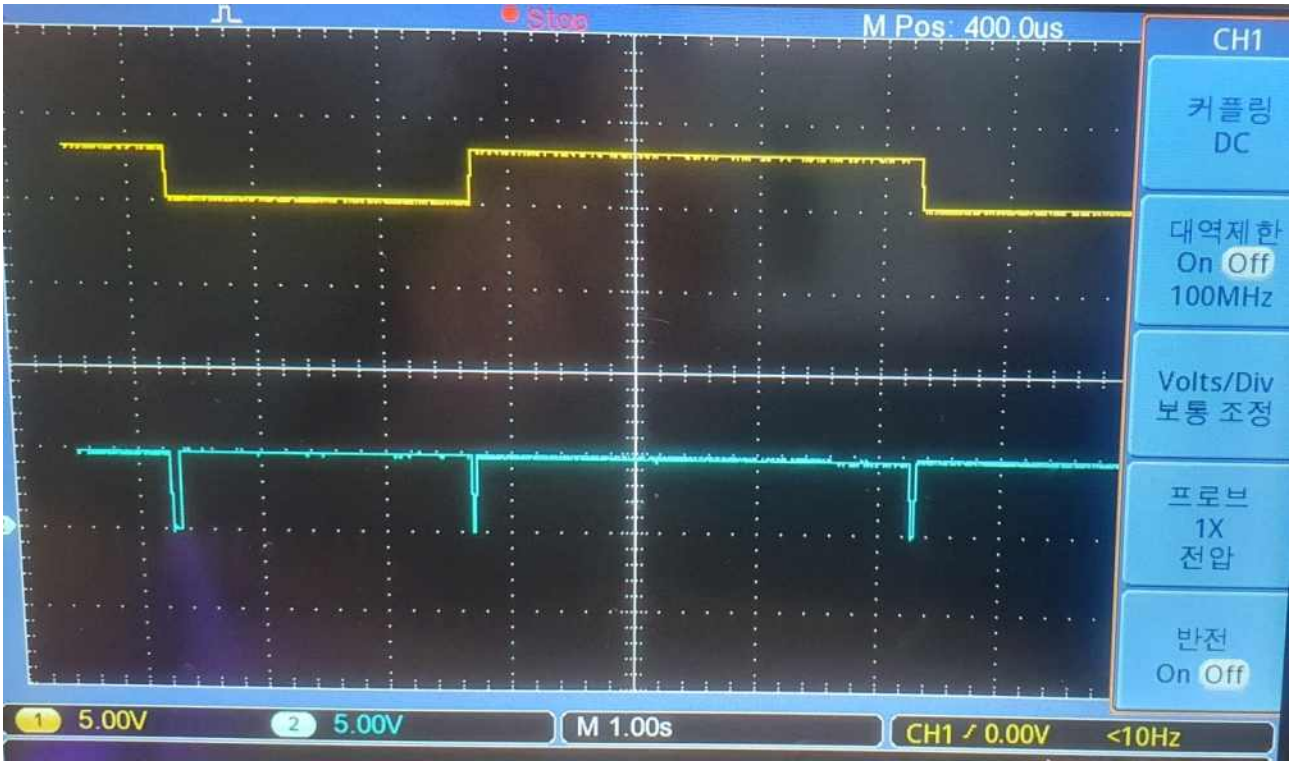
RGB LED 네오픽셀은 총 8개의 LED의 RGB 값 3개씩 총 24개의 데이터를 받게 됨. 실제로는 순서대로 4개씩 6번의 데이터를 받게 됨



CH2 - 무드등 스위치

스위치 작동으로 인한 무드등 신호는 변경사항 없음.

- UV LED와 스위치



CH1 - (UV LED) VCC

CH2 - UV 살균 스위치

스위치로 외부인터럽트 발생 시 UV LED의 전원이 ON/OFF 되는 신호를 확인할 수 있었음.

7) 소스코드

```
// 마이크로프로세서
// UV살균 무선 충전 시계 무드등
#include <mega128.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <delay.h>

#define FUNCSET 0x28 //Function Set
#define ENTMODE 0x06 // Entry Mode Set
#define ALLCLR 0x01 // All Clear
#define DISPON 0x0c // Display On
#define DISPOFF 0x08 // Display Off
#define BLINKON 0x0F // Display On & Cursor On & Cursor Blink on
#define BLINKOFF 0x0C // Display On & Cursor Off & Cursor Blink off
#define LINE2 0xC0 // 2nd Line Move 1110 0000
#define HOME 0x02 // Cursor Home
#define RSHIFT 0x1C // Display Right Shift

#define nop2 {#asm("nop"); #asm("nop");}
#define nop8 {nop2; nop2; nop2; nop2;}
#define ws2812b PORTB.7

char EDITCURSOR =0x8E; // Editing Cursor
char mood =0; // Mood Light On/Off & Color change
int duty1 =0, duty2 =0, duty3 =0; // Mood Light RGB Value
char hour=0, _min=0, sec=0, mSec=0, loc=0;
char buf[20];
void byte_out(char d);
void LCD_init(void); //LCD 초기화
void LCD_String(char *); // 문자열 출력 함수
void Busy(void);
void Command(unsigned char); //인스트럭션 쓰기 함수
void Data(unsigned char); //데이터 쓰기 함수
void time_display(void);
void edit_cursor(void);
void main(void)
{
    char i =0;

    DDRB.7 =1;
    DDRC.7 =1;

    PORTC.7 =0;

    //인터럽트
    DDRD =0x00;
    EIMSK =0b00001111;
    EICRA =0b10101010;
    TCCR1B=0x0C; OCR1A=6249; TIMSK=0x10; //16000000/256/(1+6249)=10Hz=0.1sec // A매치
    SREG =0b10000000;

    LCD_init();
```

```
    while(1)
    {
        for(i=0;i<8;i++){   //8개의 LED에 차례대로 값을 줌
            byte_out(duty1); //G
            byte_out(duty2); //R
            byte_out(duty3); //B
        }
        delay_ms(5);
    }
}
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
    mSec++;
    if(mSec>9)
    {
        mSec=0;
        if(++sec >59)
        {
            sec=0;
            if(++_min >59)
            {
                _min=0;
                if(++hour>23) hour=0;
            }
        }
    }
    time_display();
    Command(LINE2);
    LCD_String(" Microprocessor");
}
//시계 시간 조정1
interrupt [EXT_INT0] void external_int0(void)
{
    if(loc ==0)
    {
        Command(LINE2);
        LCD_String("Editing...        ");
    }
    TIMSK =0b00000000; //시간 조절중에는 시계를 멈춤
    loc++;
    edit_cursor();
    if(loc ==7)
    {
        loc=0;
        Command(ALLCLR);
        Command(BLINKOFF);
        time_display();
        TIMSK=0x10;
    }
}
//시계 시간 조정2
interrupt [EXT_INT1] void external_int1(void)
{
    if(TIMSK ==0b00000000)
    {
```

```
            if(loc ==1) sec++;
            if(loc ==2) sec +=10;
            if(loc ==3) _min++;
            if(loc ==4) _min +=10;
            if(loc ==5) hour++;
            if(loc ==6) hour +=10;
            if(sec >59) sec =0;
            if(_min >59) _min =0;
            if(hour >24) hour =0;
            time_display();
            edit_cursor();
        }
}
//RGB LED (Mood Light)
interrupt [EXT_INT2] void external_int2(void)
{
    mood++;
    if(mood ==1) { // Default Light
        duty1 =100; //G
        duty2 =255; //R
        duty3 =0; //B
    }
    else if(mood ==2) { // RED
        duty1 =0; //G
        duty2 =255; //R
        duty3 =0; //B
    }
    else if(mood ==3) { // GREEN
        duty1 =255; //G
        duty2 =0; //R
        duty3 =0; //B
    }
    else if(mood ==4) { // BLUE
        duty1 =0; //G
        duty2 =0; //R
        duty3 =255; //B
    }
    else if(mood ==5) { // PURPLE
        duty1 =0; //G
        duty2 =95; //R
        duty3 =255; //B
    }
    else if(mood ==6) { // PINK
        duty1 =0; //G
        duty2 =255; //R
        duty3 =221; //B
    }
    else if(mood ==7) { // Off
        duty1 =0; //G
        duty2 =0; //R
        duty3 =0; //B
        mood =0;
    }
}
//UV LED 전원 ON OFF
interrupt [EXT_INT3] void external_int3(void)
```

```
{
    PORTC.7 =!PORTC.7;
}
//RGB LED
void byte_out(char d)
{
    char i;
    for(i=0;i<8;i++){
        if(d&0x80){ ws2812b=1; nop8; ws2812b=0; }
        else       { ws2812b=1; nop2; ws2812b=0; }
        d<<=1;  // 비트연산자
    }
}
// 시간 출력
void time_display(void)
{
    Command(HOME);
    sprintf(buf,"[TIME][%2d:%2d:%2d]", hour, _min, sec);
    LCD_String(buf);
}
// Editting 커서 표시
void edit_cursor(void)
{
    Command(BLINKON);
    if(loc ==1) Command(EDITCURSOR);
    if(loc ==2) Command(EDITCURSOR-1);
    if(loc ==3) Command(EDITCURSOR-3);
    if(loc ==4) Command(EDITCURSOR-4);
    if(loc ==5) Command(EDITCURSOR-6);
    if(loc ==6) Command(EDITCURSOR-7);
}
// LCD 초기화
void LCD_init(void)
{
    DDRA =0xFF; // 포트 A 출력 설정
    PORTA &=0xFB; // E = 0;
    delay_ms(15);
    Command(0x20);
    delay_ms(5);
    Command(0x20);
    delay_us(100);
    Command(0x20);
    Command(FUNCSET);
    Command(DISPON);
    Command(ALLCLR);
    Command(ENTMODE);
}
// 문자열 출력 함수
void LCD_String(char *str)
{
    char *pStr=0;
    pStr = str;
    while(*pStr) Data(*pStr++);
}
// 인스트럭션 쓰기 함수
void Command(unsigned char byte)
```

```
{
    Busy();

    // 인스트럭션 상위 바이트
    PORTA = (byte &0xF0); // 데이터
    PORTA &=0xFE; //RS = 0;
    PORTA &=0xFD; //RW = 0;
    delay_us(1);
    PORTA |=0x04; //E = 1;
    delay_us(1);
    PORTA &=0xFB; //E = 0;

    // 인스트럭션 하위 바이트
    PORTA = ((byte<<4) &0xF0); // 데이터
    PORTA &=0xFE; //RS = 0;
    PORTA &=0xFD; //RW = 0;
    delay_us(1);
    PORTA |=0x04; //E = 1;
    delay_us(1);
    PORTA &=0xFB; //E = 0;
}
// 데이터 쓰기 함수
void Data(unsigned char byte)
{
    Busy();

    // 데이터 상위 바이트
    PORTA = (byte &0xF0); // 데이터
    PORTA |=0x01; //RS = 1;
    PORTA &=0xFD; //RW = 0;
    delay_us(1);
    PORTA |=0x04; //E = 1;
    delay_us(1);
    PORTA &=0xFB; //E = 0;
    // 데이터 하위 바이트
    PORTA = ((byte<<4) &0xF0); // 데이터
    PORTA |=0x01; //RS = 1;
    PORTA &=0xFD; //RW = 0;
    delay_us(1);
    PORTA |=0x04; //E = 1;
    delay_us(1);
    PORTA &=0xFB; //E = 0;
}
// Busy Flag Check -> 일반적인 BF를 체크하는 것이 아니라
// 일정한 시간 지연을 이용한다.
void Busy(void)
{
    delay_ms(2);
}
```