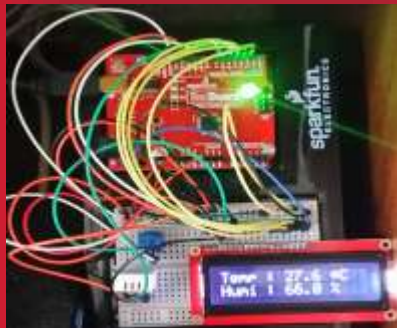




Arduino-basic

[wk05]

LED II.



Learn how to code Arduino from scratch

Comsi, INJE University

2nd semester, 2019

Email : chaos21c@gmail.com



My ID (ARnn)

AR01	염현제
AR02	강민수
AR03	구병준
AR04	김종민
AR05	박성철
AR06	이승현
AR07	이창호
AR08	변성현
AR09	손성빈
AR10	안예찬
AR11	유종인
AR12	이석민
AR13	이주원
AR14	정재영
AR15	차요신

AR16	하태성
AR17	강현이
AR18	신종원
AR19	최진솔
AR20	김경미
AR21	김경영
AR22	김규년
AR23	김민재
AR24	김영록
AR25	송다은
AR26	정지환
AR27	김종건



[Review]

◆ [wk04]

- **Arduino LED-I.**
- **Complete your project**
- **Submit folder : ARnn_Rpt03**

wk04 : Practice-03 : ARnn_Rpt03

◆ [Target of this week]

- Complete your works
- Save your outcomes
- Upload all in github.

제출폴더명 : **ARnn_Rpt03**

제출할 파일들

- ① **ARnn_2led.ino**
- ② **ARnn_4led.fzz**
- ③ **ARnn_4led.ino**
- ④ **ARnn_RGB.fzz**
- ⑤ **ARnn_RGB.ino**
- ⑥ **ARnn_RGB_Y.png**



4. LED II

FND

4 1EA



7세그먼트 1채널

공통 음극 7세그먼트
시계나 점수 등의 숫자를
표현 할 때 많이 사용됩니다.

5 1EA



74HC595N

기본 메인보드입니다.
74HC595N LED,
도트매트릭스, NFD 제어 IC 입니다.

3 1EA



7세그먼트 4채널

7세그먼트가 4개 연결된 형태의
부품입니다.
총 12개의 핀을 사용합니다.

23 1EA



8x8 도트매트릭스 모듈

LED로 다양한 연출을
할 수 있습니다.

FND (Flexible Numeric Display)

- ✓ LED의 조합으로 숫자를 표시하는 장치
- ✓ 7개의 LED를 사용하기 때문에 7-segment 라고도 함.
- ✓ 숫자뿐만 아니라 간단한 기호나 16진수 까지 표현 가능

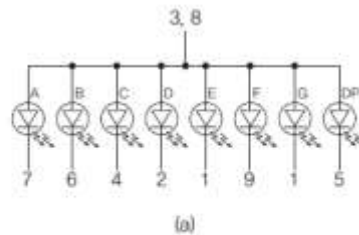
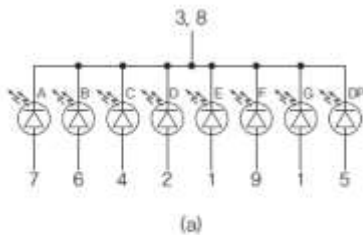
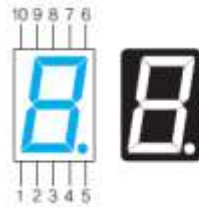
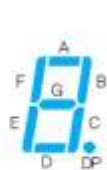
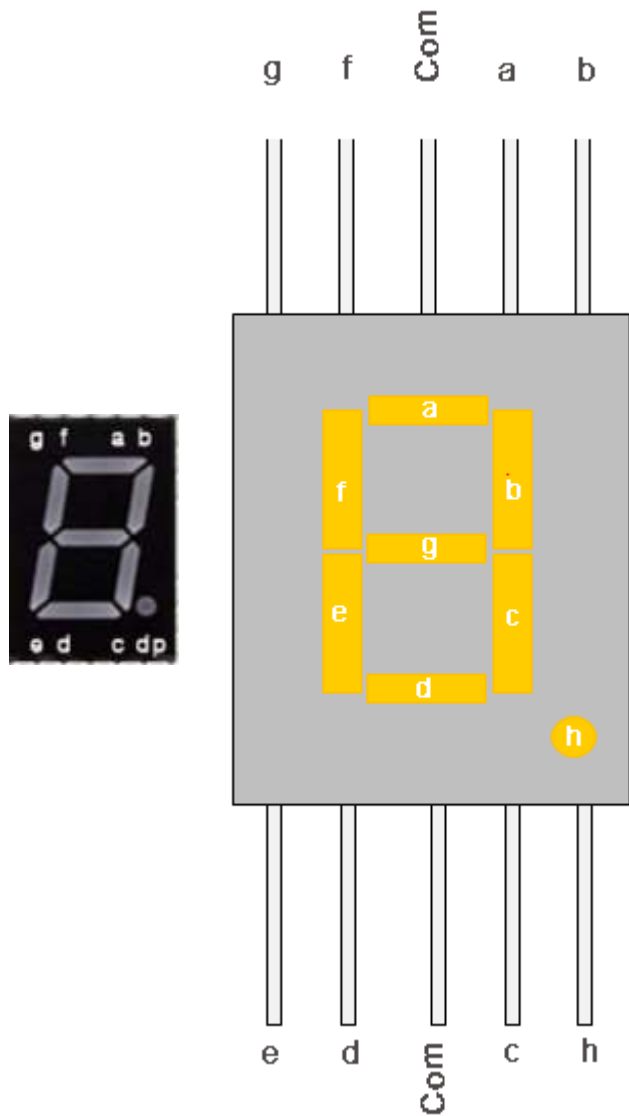


그림 4.4 Common Cathode 형(a)와 Common Anode 형(b)

표 4.1 Common Cathode FND 표시

캐소드 공통 7-세그먼트 한 자리 제어 방법										7-Seg. 출력 내용
I/O 포트 출력 내용										16진수
Q0	DP	G	F	E	D	C	B	A		
1	X	X	X	X	X	X	X	X	X	8 (소등)
0	0	0	1	1	1	1	1	1	0x3f	8 (0)
0	0	0	0	0	0	1	1	0	0x06	8 (1)
0	0	1	0	1	1	0	1	1	0x5b	8 (2)
0	0	1	0	0	1	1	1	1	0x4f	8 (3)
0	0	1	1	0	0	1	1	0	0x66	8 (4)
0	0	1	1	0	1	1	0	1	0x6d	8 (5)
0	0	1	1	1	1	1	0	1	0x7d	8 (6)
0	0	0	0	0	0	1	1	1	0x27	8 (7)
0	0	1	1	1	1	1	1	1	0x7f	8 (8)
0	0	1	1	0	1	1	1	1	0x6f	8 (9)
0	0	1	1	1	0	1	1	1	0x77	8 (A)
0	0	1	1	1	1	1	0	0	0x7c	8 (b)
0	0	0	1	1	1	0	0	1	0x39	8 (C)
0	0	1	0	1	1	1	1	0	0x5e	8 (d)
0	0	1	1	1	1	0	0	1	0x79	8 (E)
0	0	1	1	1	0	0	0	1	0x71	8 (F)
0	1	0	0	0	0	0	0	0	0x80	8 (.)

4.5 FND 제어



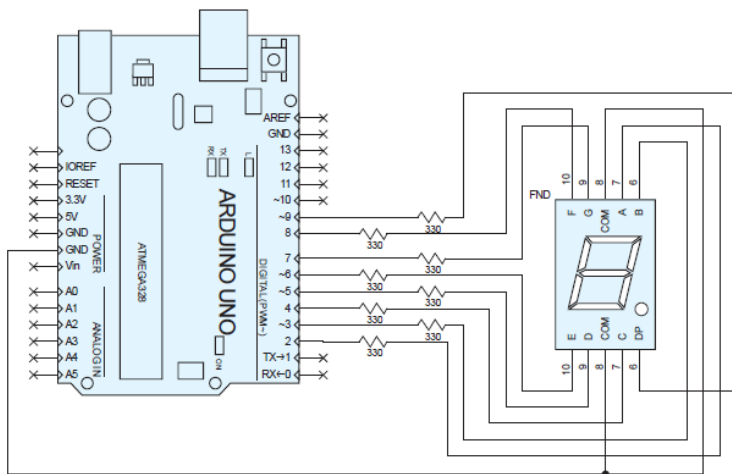
	h	g	f	e	d	c	b	a	hex value
0	0	0	1	1	1	1	1	1	3F
1	0	0	0	0	0	1	1	0	06
2	0	1	0	1	1	0	1	1	5B
3	0	1	0	0	1	1	1	1	4F
4	0	1	1	0	0	1	1	0	66
5	0	1	1	0	1	1	0	1	6D
6	0	1	1	1	1	1	0	1	7D
7	0	0	0	0	0	1	1	1	07
8	0	1	1	1	1	1	1	1	7F
9	0	1	1	0	1	1	1	1	6F
A	0	1	1	1	0	1	1	1	77
b	0	1	1	1	1	1	0	0	7C
C	0	0	1	1	1	0	0	1	39
d	0	1	0	1	1	1	1	0	5E
E	0	1	1	1	1	0	0	1	79
F	0	1	1	1	0	0	0	1	71

EX 4.4

FND 제어 (1/3)

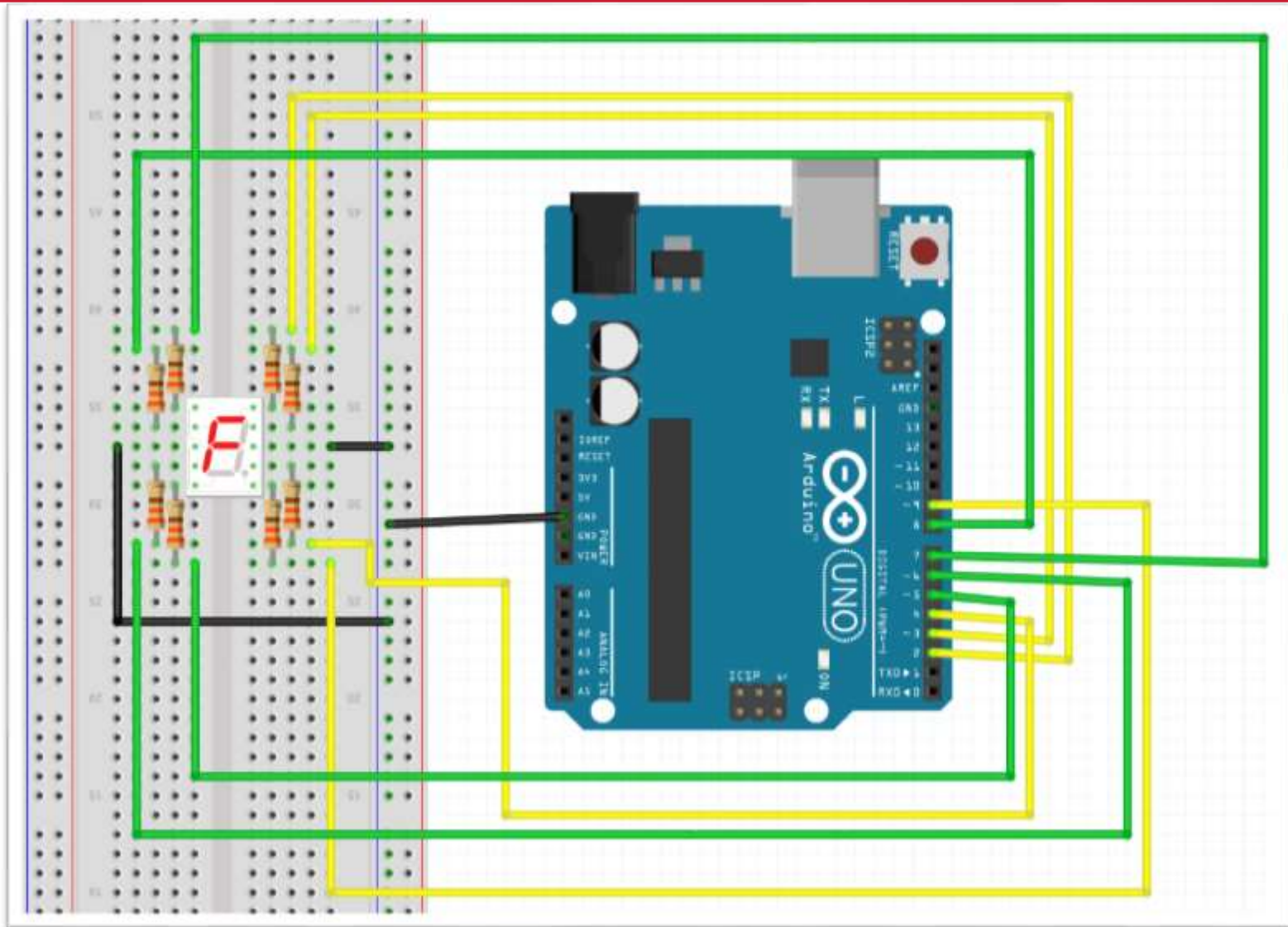
실습목표 Common Cathode FND를 이용하여 0~9의 숫자를 표시해보자.

- Hardware**
1. Common Cathode형 FND는 그림 4.2의 (a)와 같이 3번과 8번핀이 Cathode 핀으로 함께 연결되어 있다. 즉 FND의 3번과 8번핀을 GND에 연결하고 나머지 핀들에 HIGH신호를 주어 FND에 숫자를 표시한다.
 2. GND에 연결되는 3번과 8번핀을 제외한 나머지 핀들에는 FND 내의 LED의 전류를 제한하기 위해 330 Ω 저항을 연결한다.
 3. 원하는 숫자를 표시하기 위해선 2~9번핀에 표 4.1을 참고하여 신호를 출력한다.



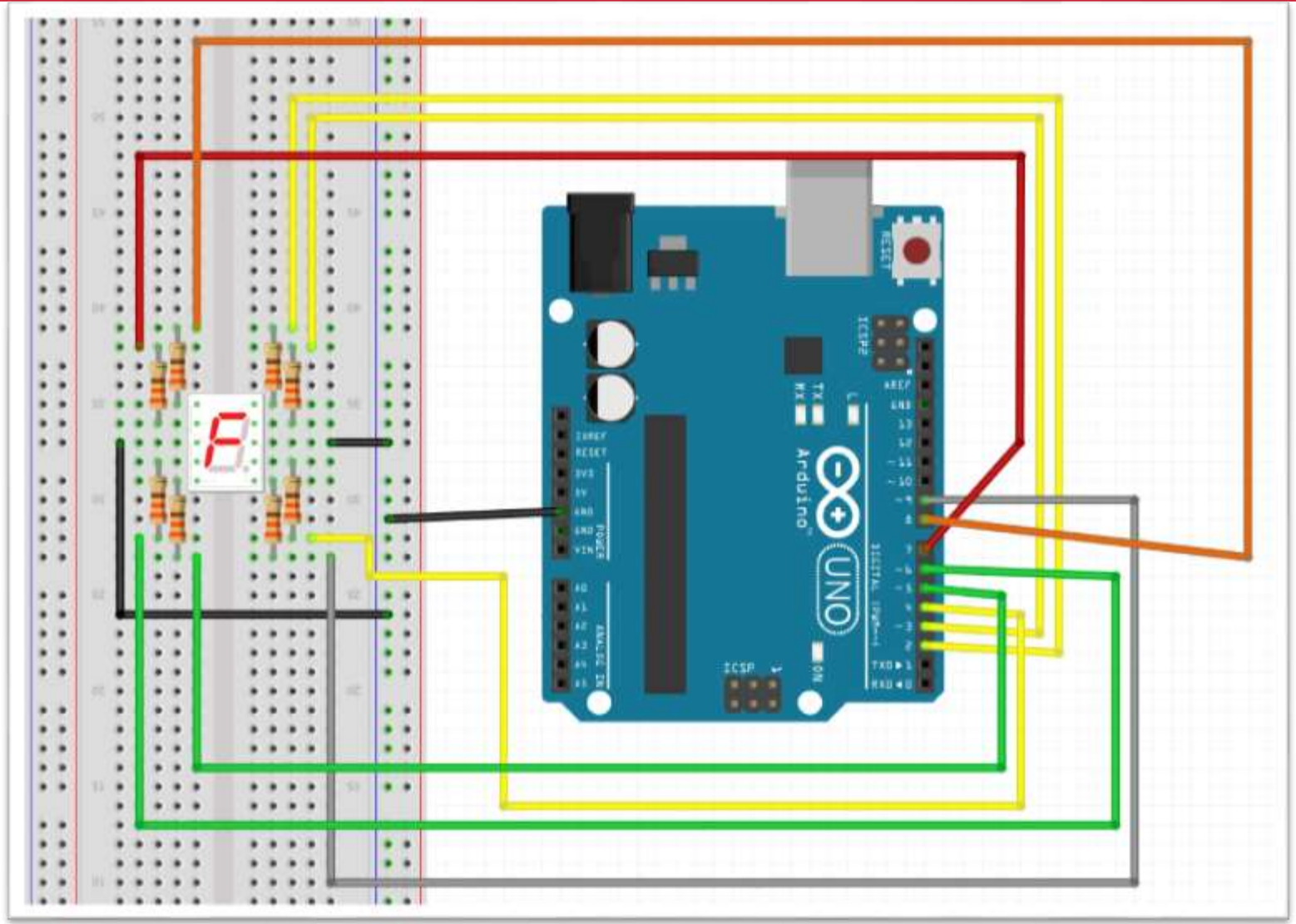


4.5.2.1 FND 제어



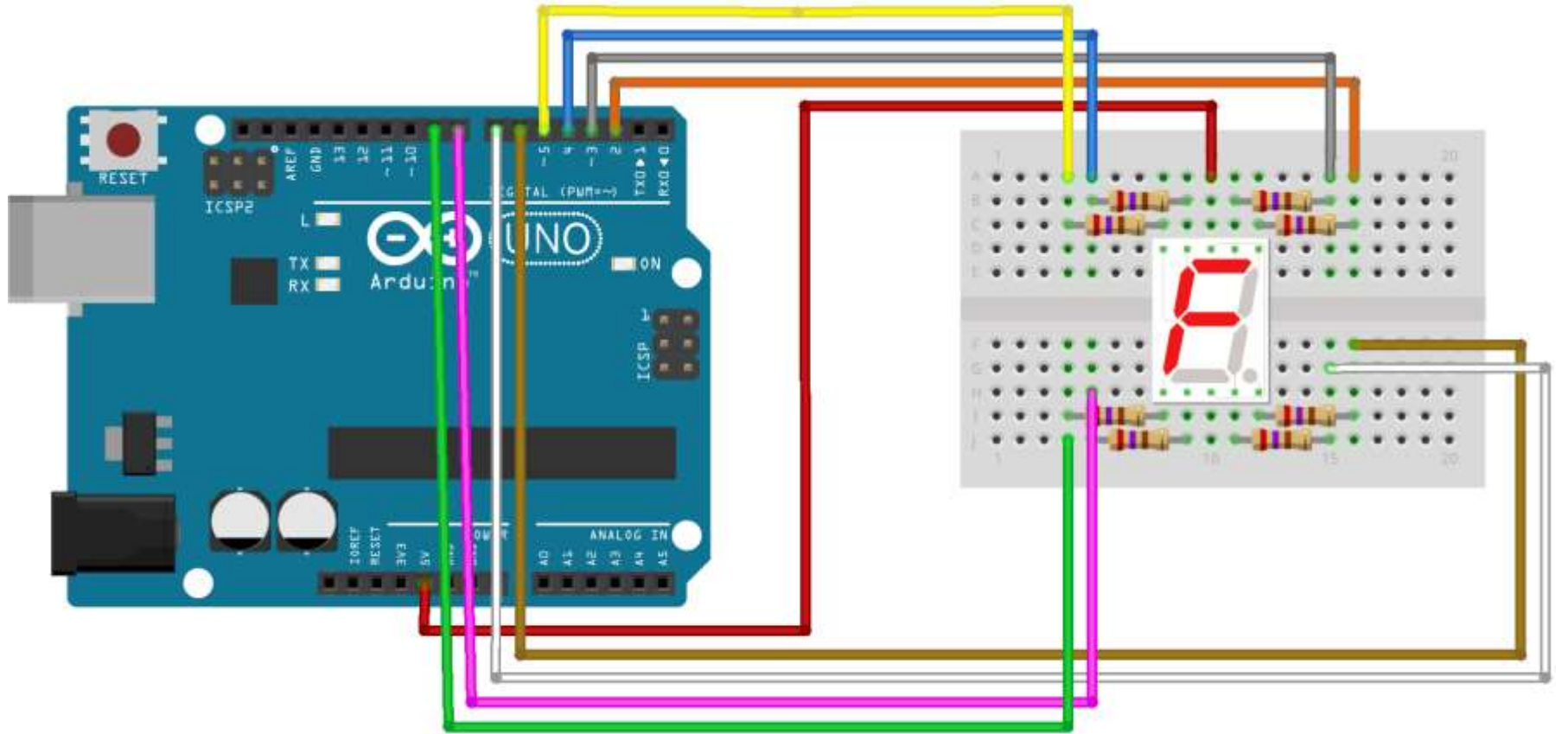
Fritzing 으로 회로를 디자인하고
[ARnn_fnd.fzz](#) 로 저장해서 제출.

4.5.2.1 FND 제어



Fritzing 으로 회로를 디자인하고
[ARnn_fnd.fzz](#) 로 저장해서 제출.

4.5.2.2 FND 제어



fritzing

EX 4.4 FND 제어 (2/3)

Commands • void 함수(변수1, 변수2, ...){
};

‘함수(변수1, 변수2)’를 이용하여 ‘{ }’ 내의 명령을 호출하여 사용한다. ‘변수1’과 ‘변수2’등을 함께 선언하면 함수 내에서 그 변수를 사용할 수 있다. 반복되는 구문을 설정해 놓고 호출하여 사용하면 편리하다.

- pinMode(핀번호, 설정)

핀의 입출력 모드를 설정한다. ‘핀번호’에는 설정하고자 하는 핀의 번호와 ‘설정’에는 입력으로 사용하기 위해선 ‘INPUT’, 출력으로 사용하기 위해선 ‘OUTPUT’, 입력이며 풀업 사용시 ‘INPUT_PULLUP’을 적는다.

- digitalWrite(핀번호, 값)

핀에 디지털 출력 (High or Low) 을 한다. ‘핀번호’에는 출력하고자 하는 핀의 번호를, ‘값’에는 ‘HIGH’ 혹은 ‘LOW’를 설정하여 High 혹은 Low 출력을 한다.

- for(변수=시작 값 ; 조건 ; 변수의 증분){ }

변수의 시작 값부터 조건이 만족하는 경우 ‘{ }’ 내의 명령을 수행한다. ‘변수의 증분’에서는 1회 명령이 수행될 때 마다 변수를 증가 혹은 감소시킨다.

4.5.4 FND 제어

EX 4.4 FND 제어 (3/3)

- Sketch 구성
1. FND에 숫자를 표시할 때 어떤 LED를 켤지에 대한 정보를 담은 상수를 설정한다.
 2. FND동작에 필요한 핀을 출력으로 설정한다.
 3. FND를 동작시키는 '`fndDisplay(int displayValue)`' 라는 함수를 만든다.
 4. 함수를 이용하여 1초 간격으로 FND에 숫자를 표시한다.

실행 결과 FND의 숫자가 0~9까지 약 1초 간격으로 변화한다.

4.5.4.1 FND 제어 - code

```
ex_4_4_1_start$
1 /*
2  예제 4.4.1
3  FND 제어 0~9까지 1초단위로 표시하기
4 */
5
6 // 0~9까지 LED 표시를 위한 상수 설정
7 const byte number[10] = {
8 //dot  gfedcba
9  B00111111,  //0
10 B00000110,  //1
11 B01011011,  //2
12 B01001111,  //3
13 B01100110,  //4
14 B01101101,  //5
15 B01111101,  //6
16 B00000111,  //7
17 B01111111,  //8
18 B01101111,  //9
19 };
20
```

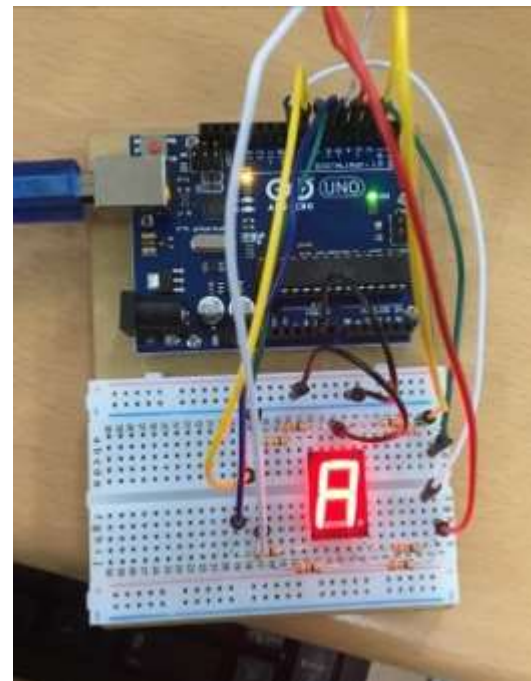
```
21 void setup()
22 {
23   // 2~9번 핀을 a b c d e f g dot 의 순서로 사용한다
24   // 2~9번핀을 출력으로 초기화 시킨다.
25   for(int i = 2; i <= 9; ++i){
26     pinMode(i, OUTPUT);
27   };
28   // 점은 표시하지 않는다
29   digitalWrite(9, LOW);
30 }
31
32 void loop()
33 {
34   // k값을 0~9로 변화시킨다.
35   for(int k = 0; k <= 9; ++k){
36     fndDisplay(k); // k값을 출력한다
37     delay(1000);
38   };
39 }
40
41 // LED 점등
42 void fndDisplay(int displayValue){
43   // bitValue 변수를 선언한다.
44   boolean bitValue;
45
46   for(int i=2; i<=9; ++i){
47     // 2~9번핀에 모두 LOW 신호를 줘서 소등시킨다
48     digitalWrite(i, LOW);
49   };
50   for(int i=0; i<=7; ++i){
51     // number 상수의 하나의 비트값을 읽는다
52     bitValue = bitRead(number[displayValue], i);
53     // 앞서 읽은 비트값을 2~9번핀에 출력시킨다
54     digitalWrite(i+2, bitValue);
55   };
56 }
```


EX 4.4 FND 제어 (3/3)

DIY 위의 예제를 0~F까지의 16진수를 표시하도록 스케치를 수정하여 보자.

(hint: LED 표시를 위한 상수에 A~F를 추가시켜서 불러와 사용하자)

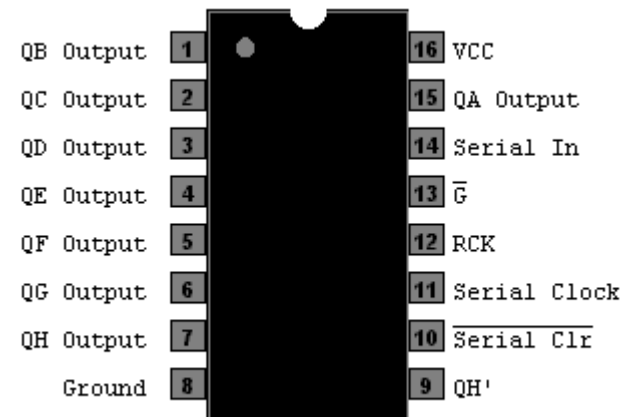
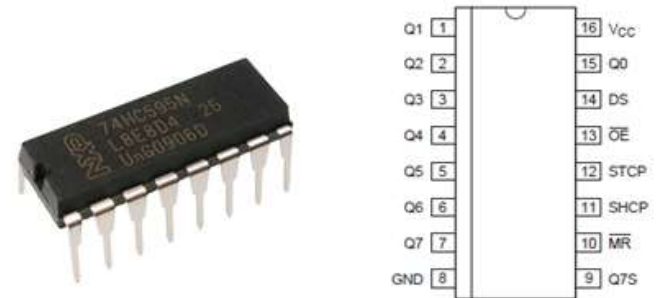
‘A’가 출력된 화면을 ARnn_A.png
로 저장해서 제출. (아두이노 회로를 포함해서 촬영)



4.6 FND 제어 : 74595 IC (74HC595N)

74595 IC

- ✓ 직렬 신호로 입력된 데이터를 병렬 신호로 변환
- ✓ FND의 8개의 LED를 켜기위한 신호를 3개의 신호선으로 입력 받아 8개의 FND 신호로 출력
- ✓ shiftout() 명령어로 구현.



SHCP : shift register clock input
STCP : storage register clock input
DS : serial data input

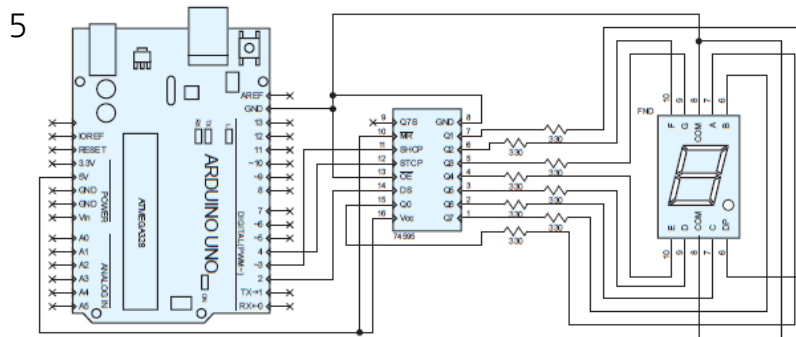
- ✓ DS, SHCP, STCP 세 핀으로 FND 제어
- ✓ 동작 순서
 1. STCP에 'LOW' 신호 입력
 2. SHCP의 클럭에 맞춰 DS로 데이터 전송
 3. 전송 후, STCP에 'HIGH' 신호를 주어 출력핀으로 신호를 출력
- ✓ Shiftout() 함수로 동작 시킴.

4.6.1 FND 제어 : 74595 IC (74HC595N)

EX 4.4.2 74595를 이용한 FND 제어 (1/3)

실습목표 Common Cathode FND를 이용하여 0~9의 숫자를 표시해보자.

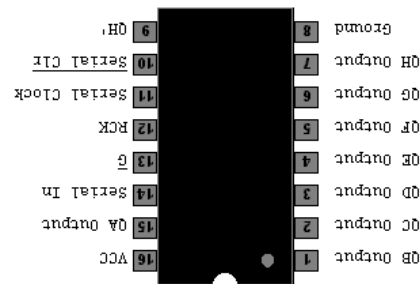
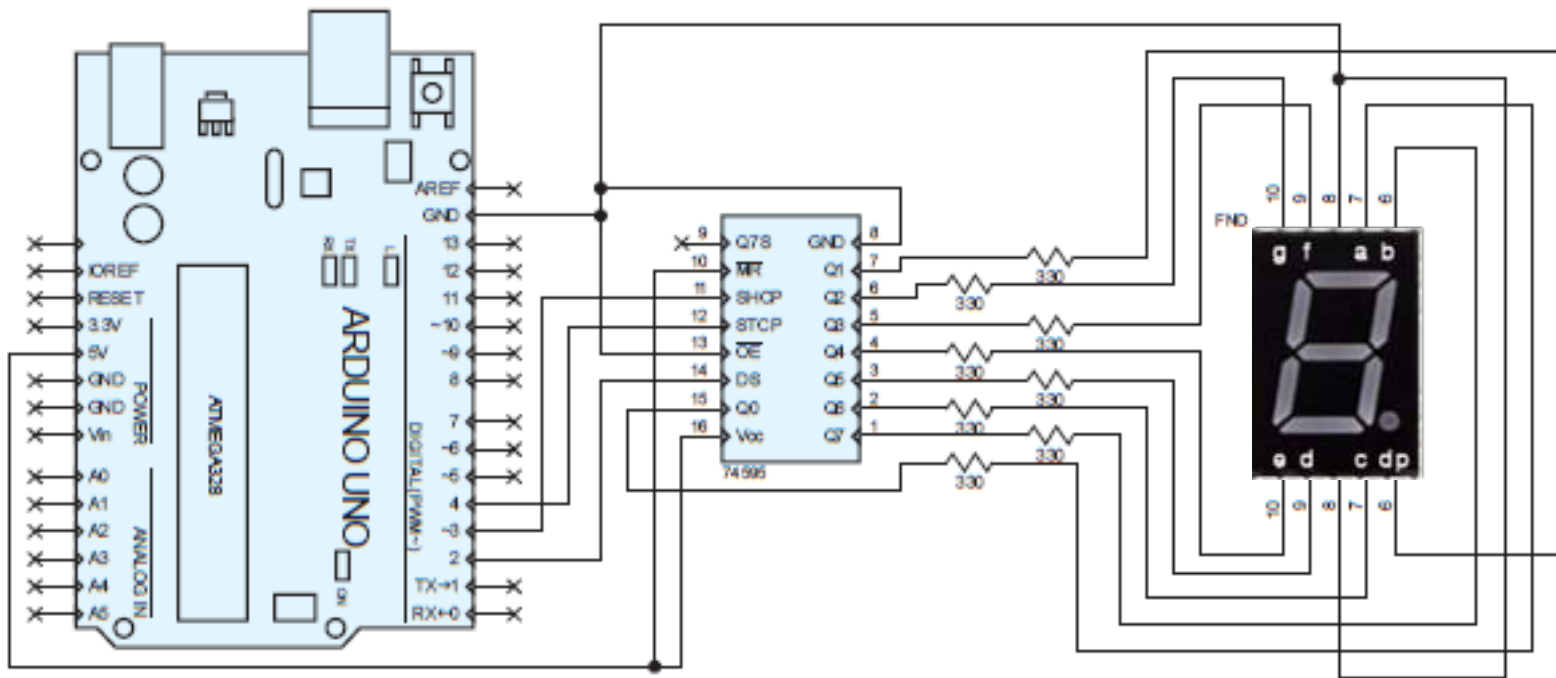
- Hardware**
1. 예제 4.4.1과 동일한 동작을 하지만 Arduino의 입출력 핀을 절약하기 위해 74595 IC를 중간에 연결한다.
 2. **Arduino에서는 2, 3, 4 세 개의 핀을 이용하여 74595 IC로 신호를 출력**한다.
각 핀을 Arduino에 연결한다.
 3. 74595 IC의 (MR) $\bar{}$ 핀과 Vcc 핀에는 5V를 연결하고 (OE) $\bar{}$ 와 GND핀은 Arduino의 GND에 연결한다.
 4. 74595 IC에서는 DS, SHCP, STCP 핀으로부터 입력된 신호를 이용하여 Q0~Q7 핀에 신호를 출력한다. Q0~Q7 핀을 FND의 Anode 핀에 연결한다.



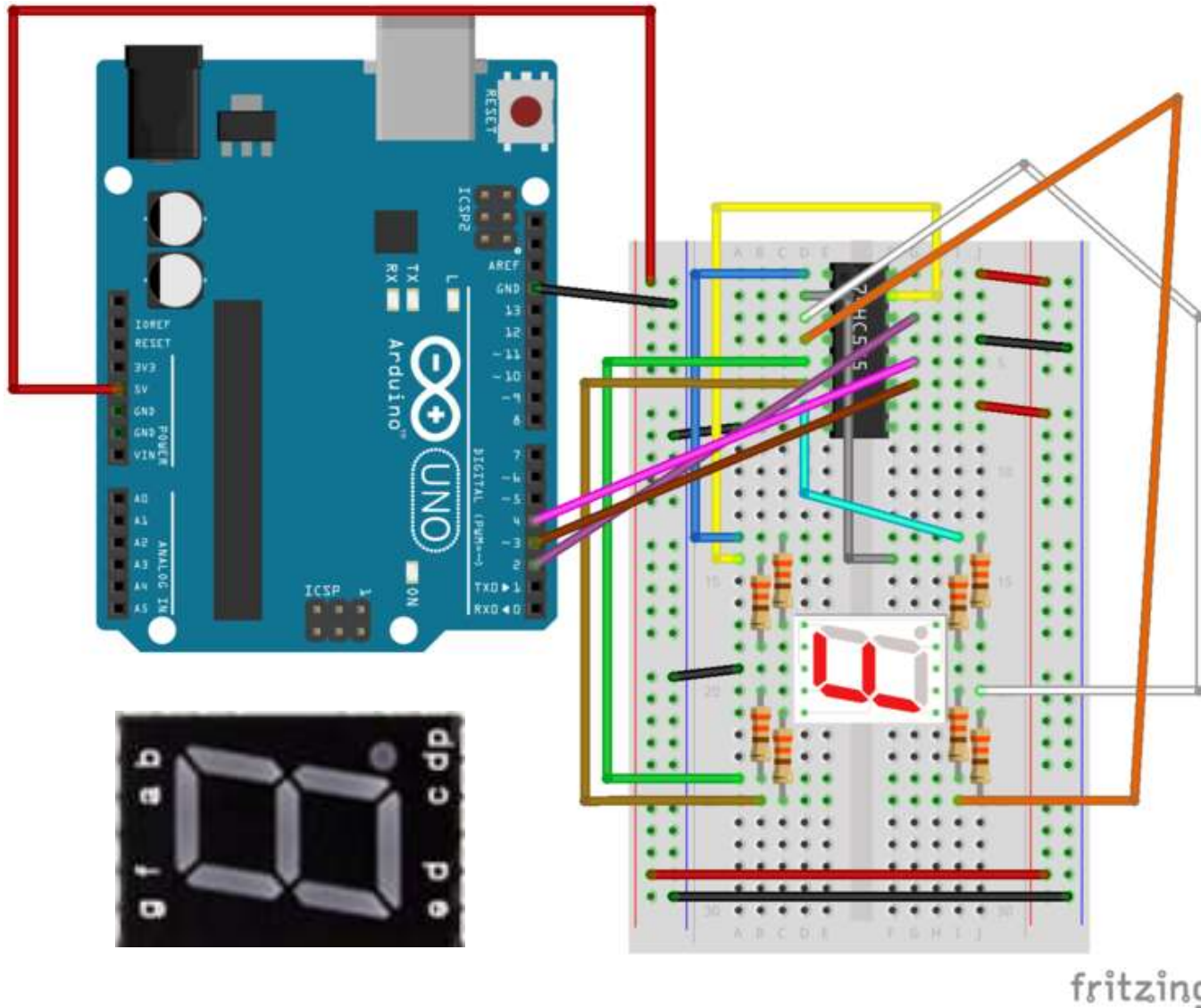
4.6.2.1 FND 제어 : 74595 IC

EX 4.4.2

74595를 이용한 FND 제어 (1/3)



4.6.2.2 FND 제어 : 74595 IC



QB Output	1	16	VCC
QC Output	2	15	QA Output
QD Output	3	14	Serial In
QE Output	4	13	\bar{G}
QF Output	5	12	RCK
QG Output	6	11	Serial Clock
QH Output	7	10	Serial Clr
Ground	8	9	QH'

11 → D3

12 → D4

14 → D2

15 → a

1 \rightarrow b

2 → c

3 → d

4 → e

5 → f

6 → g

7 → h

4.6.3 FND 제어 : 74595 IC

EX 4.4.2 74595를 이용한 FND 제어 (2/3)

Commands • void 함수(변수1, 변수2, ...){
};

‘함수(변수1, 변수2)’ 를 이용하여 { } 내의 명령을 호출하여 사용한다. ‘변수1’과 ‘변수2’등을 함께 선언하면 함수 내에서 그 변수를 사용할 수 있다. 반복되는 구문을 설정해 놓고 호출하여 사용하면 편리하다.

- pinMode(핀번호, 설정)

핀의 입출력 모드를 설정한다. ‘핀번호’ 에는 설정하고자 하는 핀의 번호와 ‘설정’에는 입력으로 사용하기 위해선 ‘INPUT’, 출력으로 사용하기 위해선 ‘OUTPUT’, 입력이며 풀업 사용시 ‘INPUT_PULLUP’을 적는다.

- digitalWrite(핀번호, 값)

핀에 디지털 출력 (High or Low) 을 한다. ‘핀번호’ 에는 출력하고자 하는 핀의 번호를, ‘값’에는 ‘HIGH’ 혹은 ‘LOW’ 를 설정하여 High 혹은 Low 출력을 한다.

- shiftOut(데이터 핀, 클럭 핀, 출력비트 순서, 출력 값)

데이터 핀으로는 비트단위로 출력될 핀 번호를 써준다. 클럭 핀에는 데이터가 출력될 때 토글되는 클럭 출력에 사용할 핀 번호를 써준다. 출력비트 순서는 비트 데이터의 맨 왼쪽부터 순차적으로 출력하고자 하면 ‘MSBFIRST’, 맨 오른쪽부터 순차적으로 출력하고자 하면 ‘LSBFIRST’를 써 분다. 출력 값에는 실제 출력할 데이터를 써 준다. 이 때 데이터는 8비트 즉 2진수 8자리의 숫자를 갖는다.

4.6.4 FND 제어 : 74595 IC

EX 4.4.2 74595를 이용한 FND 제어 (3/3)

- Sketch 구성**
1. FND에 숫자를 표시할 때 어떤 LED를 켤지에 대한 정보를 담은 상수를 설정한다.
 2. FND동작에 필요한 핀을 출력으로 설정한다.
 3. FND를 동작시키는 'fndDisplay74595(int displayValue)' 라는 함수를 만든다.
 4. 'fndDisplay74595(int displayValue)'에는 'shiftOut()' 명령어를 이용한 FND 동작 스케치를 넣는다.
 5. 함수를 이용하여 1초 간격으로 FND에 숫자를 표시한다.

실행 결과 FND의 숫자가 0~9까지 약 1초 간격으로 변화한다.

4.6.4.1 FND 제어 : 74595 IC - code

ex_4_4_2_start\$

```

1  /*
2  예제 4.4.2
3  74595를 이용하여 FND 제어 0~9까지
4  1초단위로 표시하기
5  */
6
7  // 0~9까지 LED 표시를 위한 상수 설정
8  const byte number[10] = {
9  //dot  gfedcba
10 B00111111, //0
11 B00000110, //1
12 B01011011, //2
13 B01001111, //3
14 B01100110, //4
15 B01101101, //5
16 B01111101, //6
17 B00000111, //7
18 B01111111, //8
19 B01101111, //9
20 };
21
22 int ds = 2; // 74595의 DS핀을 Arduino의 2번핀에 연결
23 int shcp = 3; // 74595의 STCP핀을 Arduino의 3번핀에 연결
24 int stcp = 4; // 74595의 SHSP핀을 Arduino의 4번핀에 연결
25
26 void setup()
27 {
28 // 2~9번핀을 출력으로 초기화 시킨다.
29 for(int i = 2; i <= 9; ++i){
30 pinMode(i,OUTPUT);
31 };
32 digitalWrite(9,LOW); // 점은 표시하지 않는다
33 }

```

```

35 void loop()
36 {
37 // k값을 0~9로 변화시킨다.
38 for(int k = 0; k <= 9; ++k){
39 fndDisplay74595(k); // k값을 출력한다
40 delay(1000); // 1초간 지연시킨다.
41 };
42 }
43
44 // LED 점등
45 void fndDisplay74595(int displayValue){
46 // STCP에 LOW 신호를 보내서 74595로 데이터전송을 시작한다.
47 digitalWrite(stcp, LOW);
48 // shiftOut 명령어로 74595에 출력을 보낸다.
49 shiftOut(ds, shcp, MSBFIRST, number[displayValue]);
50 // STCP에 HIGH 신호를 보내서 74595로 데이터전송을 종료한다.
51 digitalWrite(stcp, HIGH);
52 }

```

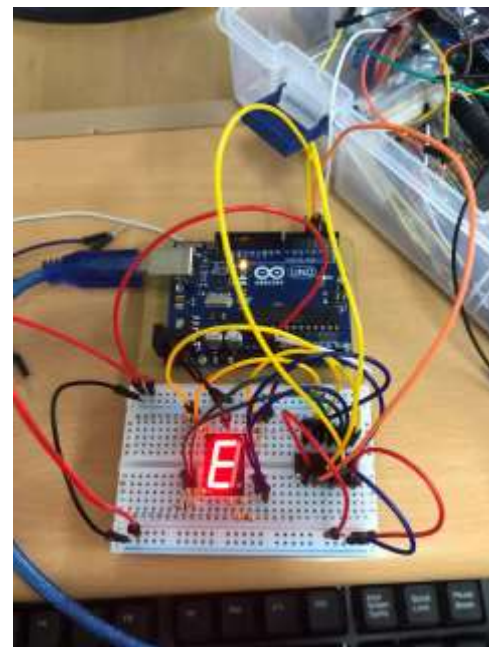
4.6.5 FND 제어 : 74595 - DIY

EX 4.4.2 74595를 이용한 FND 제어 (3/3)

DIY 위의 예제를 0~F까지의 16진수를 표시하도록 스케치를 수정하여 보자.

(hint: LED 표시를 위한 상수에 A~F를 추가시켜서 불러와 사용하자)

‘E’가 출력된 화면을 ARnn_74595_E.png
로 저장해서 제출. (아두이노 회로를 포함해서 촬영)



4.7 4-digit FND 제어

4-digit FND

- ✓ FND 네 개를 이용하여 네 자리 숫자를 표시하는 부품
- ✓ Common Cathode형과 Common Anode형
- ✓ FND와 핀 구조는 동일하지만 각 자릿수를 선택하는 핀 추가

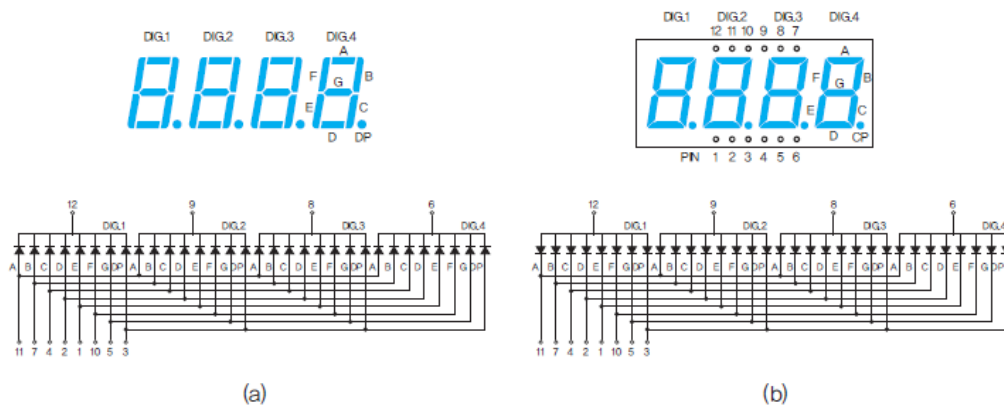


그림 4.6 4-digit FND와 내부 회로. Common Cathode (a), Common Anode (b)



그림 4.7 실험에 사용할 4-digit FND

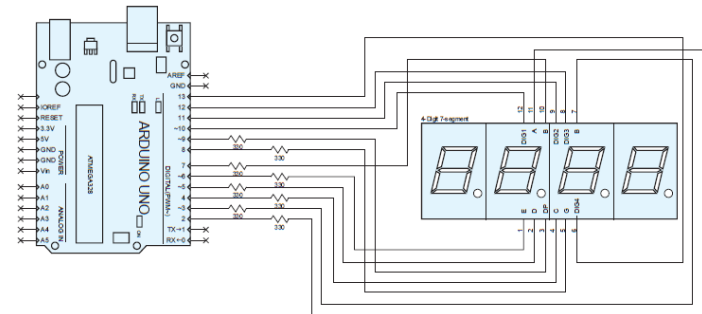


4.7.1 4-digit FND 제어

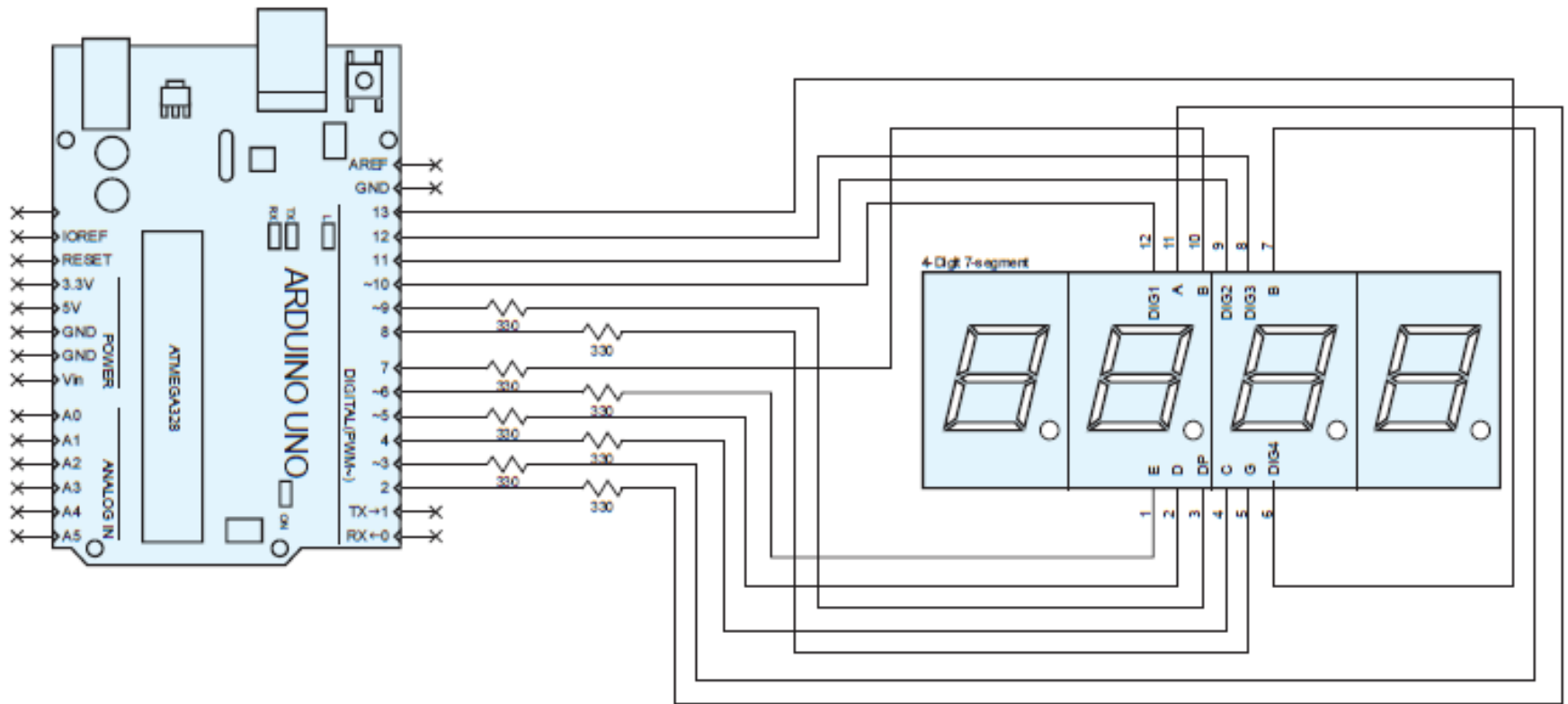
EX 4.5.1 4-digit FND로 0000~9999 숫자 표시하기 (1/3)

실습목표 Common Cathode 4-digit FND를 이용하여 0000~9999까지 1초 간격으로 증가하는 스케치를 작성해 보자.

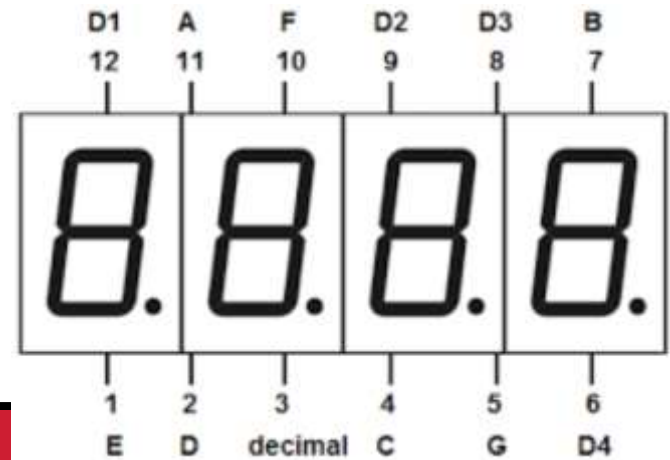
- Hardware**
1. 4-digit FND는 4개의 FND를 연결한 부품이다.
 2. 각각의 FND에는 DIG1~DIG4 네 개의 핀이 각각의 FND의 Common Cathode로 연결되어 있다.
 3. A~G, DP핀은 하나의 FND를 동작시킬 때와 같이 330Ω저항을 통하여 Arduino 2~9번핀에 연결한다.
 4. 맨 왼쪽 FND를 동작시키려면 DIG1에만 LOW신호를 준 상태에서 A~G, DP 핀에 원하는 숫자를 쓰기 위한 신호를 주어야 한다.
 5. 두번째 FND를 동작시키려면 DIG2에만 LOW신호를 준 상태에서 A~G, DP 핀에 원하는 숫자를 쓰기 위한 신호를 주어야 한다.
 6. DIG1~DIG4에 모두 LOW신호를 주면 모두 같은 숫자가 표시된다.



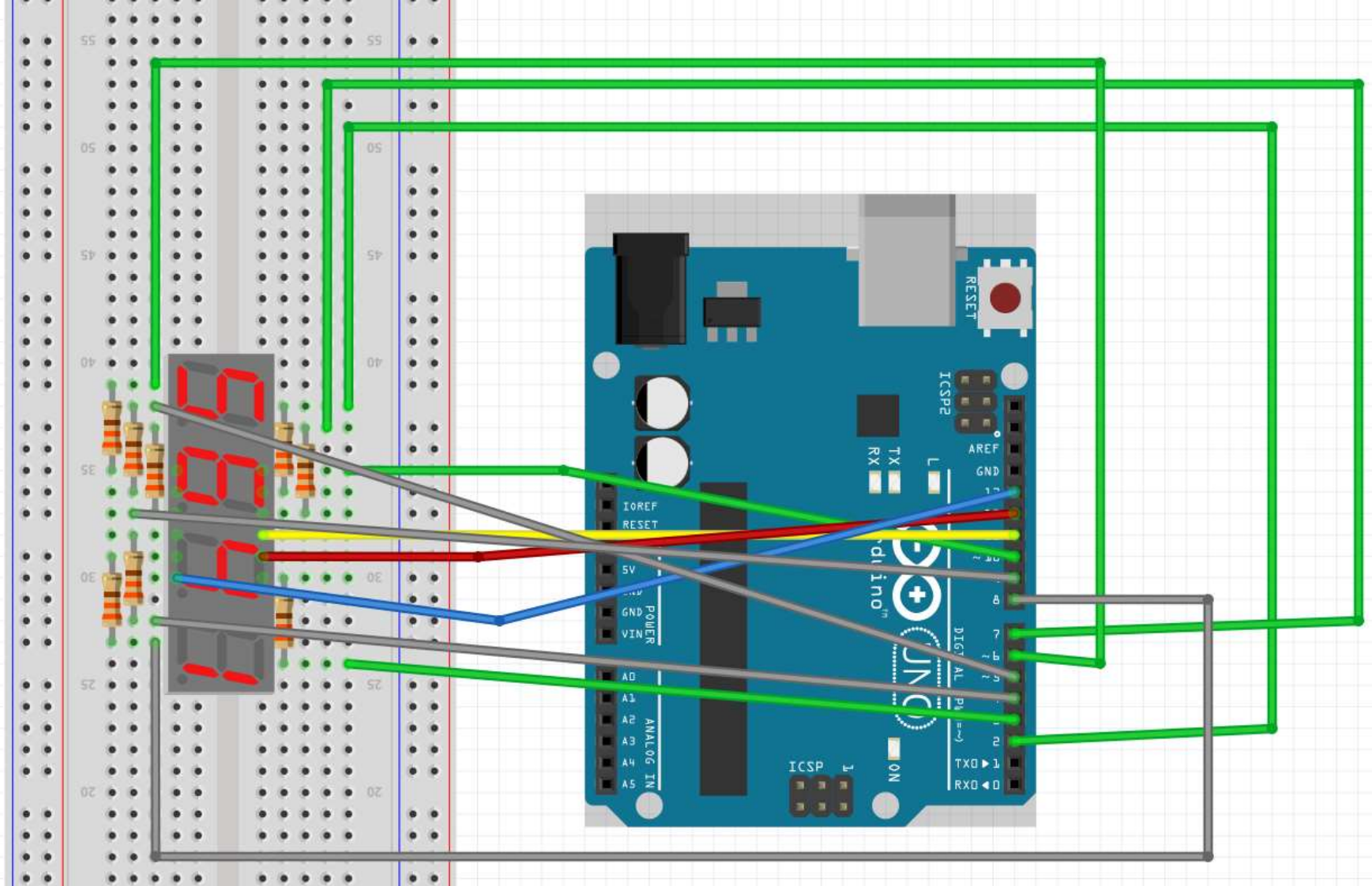
4.7.2 4-digit FND 제어



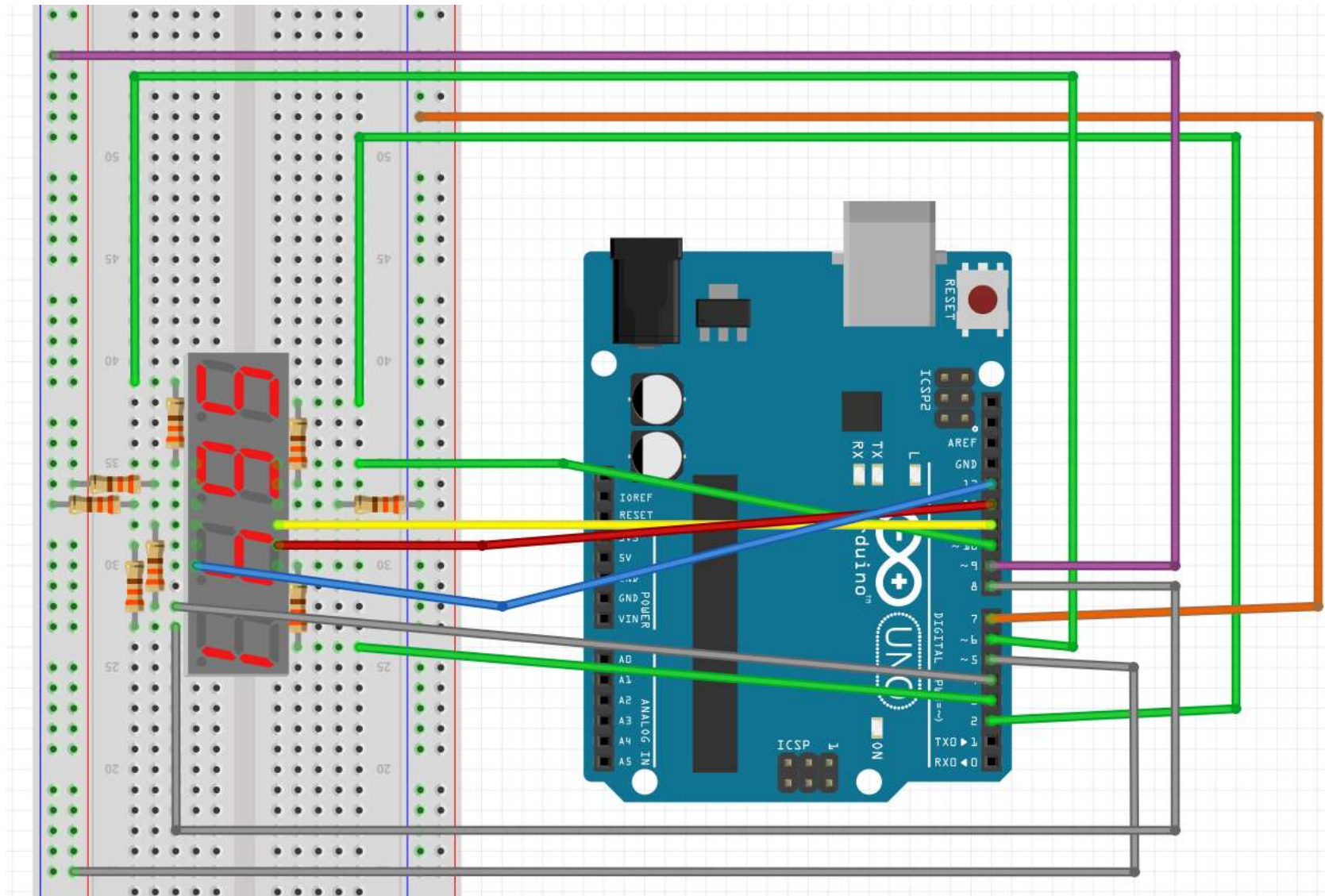
<https://m.blog.naver.com/PostView.nhn?blogId=wnddh12&logNo=220606781604&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F>



4.7.2 4-digit FND 제어 - circuit1

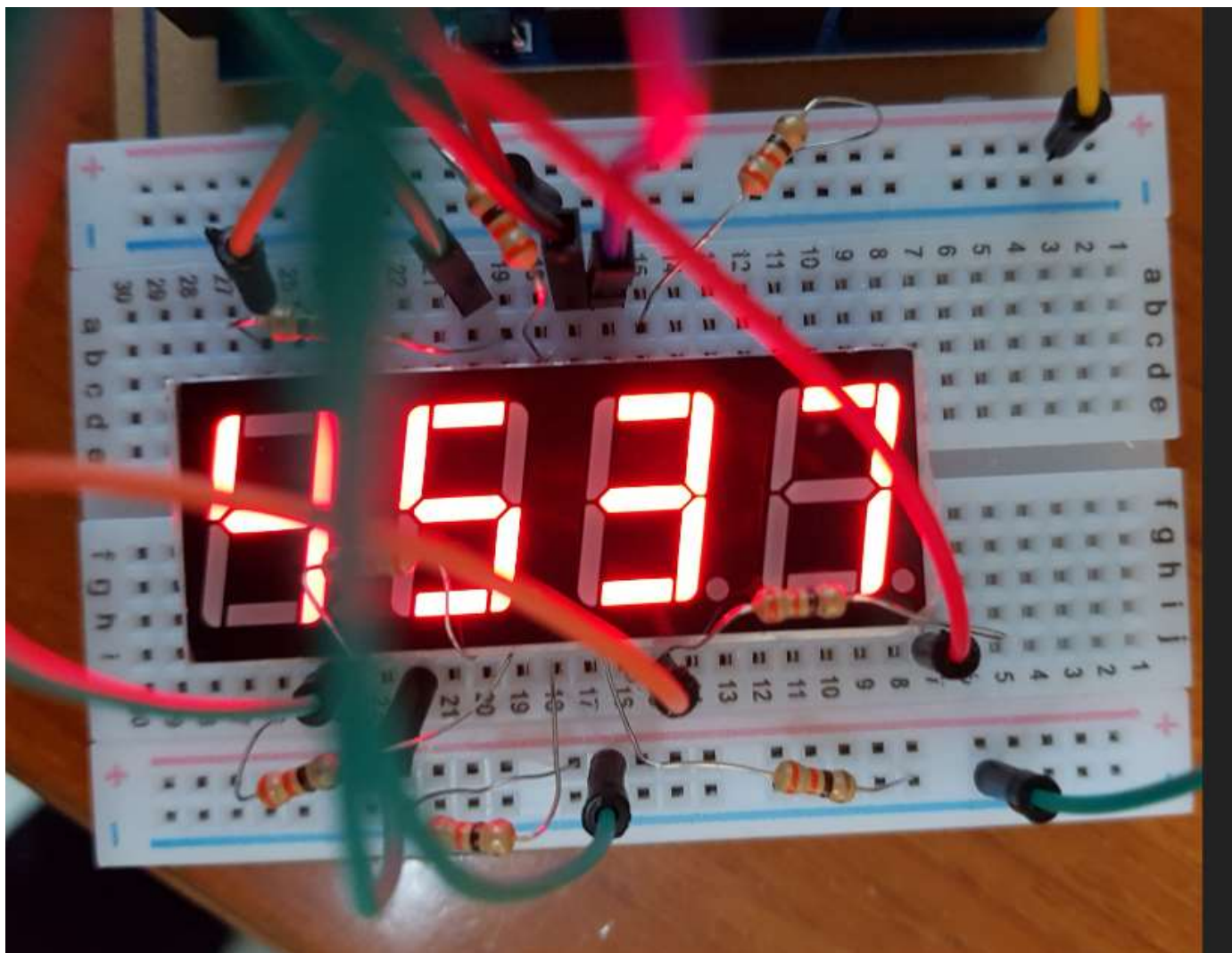


4.7.2 4-digit FND 제어 - circuit2



Fritzing 으로 회로를 디자인하고
ARnn_4digit.fzz 로 저장해서 제출.

4.7.2.1 4-digit FND 제어 (참고 회로)



4.7.3 4-digit FND 제어

EX 4.5.1 4-digit FND로 0000~9999 숫자 표시하기 (2/3)

Commands • void 함수(변수1, 변수2, ...){
};

‘함수(변수1, 변수2)’ 를 이용하여 { } 내의 명령을 호출하여 사용한다. ‘변수1’과 ‘변수2’등을 함께 선언하면 함수 내에서 그 변수를 사용할 수 있다. 반복되는 구문을 설정해 놓고 호출하여 사용하면 편리하다.

- pinMode(핀번호, 설정)

핀의 입출력 모드를 설정한다. ‘핀번호’ 에는 설정하고자 하는 핀의 번호와 ‘설정’에는 입력으로 사용하기 위해선 ‘INPUT’, 출력으로 사용하기 위해선 ‘OUTPUT’, 입력이며 풀업 사용시 ‘INPUT_PULLUP’을 적는다.

- digitalWrite(핀번호, 값)

핀에 디지털 출력 (High or Low) 을 한다. ‘핀번호’ 에는 출력하고자 하는 핀의 번호를, ‘값’에는 ‘HIGH’ 혹은 ‘LOW’ 를 설정하여 High 혹은 Low 출력을 한다.

- for(변수=시작 값 ; 조건 ; 변수의 증분){ }

변수의 시작 값부터 조건이 만족하는 경우 { } 내의 명령을 수행한다. ‘변수의 증분’에서는 1회 명령이 수행될 때 마다 변수를 증가 혹은 감소시킨다.

4.7.4 4-digit FND 제어

EX 4.5.1 4-digit FND로 0000~9999 숫자 표시하기 (3/3)

- Sketch 구성**
1. FND에 숫자를 표시할 때 어떤 LED를 점등할 지에 대한 정보를 담은 상수를 설정한다.
 2. FND동작에 필요한 핀을 출력으로 설정한다.
 3. DIG에 연결된 핀을 모두 LOW로 설정하여 모든 FND가 켜지도록 한다.
 4. 예제 4.4에서 설정한 'fndDisplay(int displayValue)' 함수를 응용하여 각 FND를 지정하여 1초 간격으로 0~9까지의 숫자를 모든 FND에 표시한다.

실행 결과 4개의 FND의 숫자가 0~9까지 1111 단위로 약 1초 간격으로 변화한다.

4.7.4.1 4-digit FND 제어 - code

```

6 // 0~9까지 LED 표시를 위한 상수
7 const byte number[10] = {
8   //dot  gfedcba
9   B00111111, //0
10  B00000110, //1
11  B01011011, //2
12  B01001111, //3
13  B01100110, //4
14  B01101101, //5
15  B01111101, //6
16  B00000111, //7
17  B01111111, //8
18  B01101111, //9
19 };
20
21 // 표시할 숫자 변수
22 int count = 0;
23
24 void setup()
25 {
26   // 2~9번 핀을 a b c d e f g dot 의 순서로 사용한다.
27   // 10~13번 핀을 Digit 1~4 의 순서로 사용한다.
28   for(int i = 2; i <= 13; ++i){
29     pinMode(i,OUTPUT); // 2~13번핀을 출력으로 설정한다.
30   };
31
32   // 4 digit와 연결된 10~13번핀에 모두 LOW 신호를 줘서
33   for(int i=10; i<=13; ++i){
34     digitalWrite(i, LOW);
35   };
36 }

```

```

35 void loop()
36 {
37   // count 변수값을 FND에 출력한다.
38   fndDisplay(count);
39
40   // count 변수값이 0~9의 범위를 갖도록한다.
41   if(count >=9) count = 0;
42   else ++count;
43
44   delay(1000);
45 }
46
47 // LED 켜는 루틴
48 void fndDisplay(int displayValue){
49   // bitValue 변수를 선언한다.
50   boolean bitValue;
51
52   // 2~9번핀에 모두 LOW 신호를 줘서 소등시킨다.
53   for(int i=2; i<=9; ++i){
54     digitalWrite(i, LOW);
55   };
56
57   for(int i=0; i<=7; ++i){
58     // number 상수의 하나의 비트값을 읽는다.
59     bitValue = bitRead(number[displayValue], i);
60     // 앞서 읽은 비트값을 2~9번핀에 출력시킨다.
61     digitalWrite(i+2, bitValue);
62   };
63 }

```

4.7.5 4-digit FND 제어 - DIY

DIY

‘XXX1’, ‘XX2X’, ‘X3XX’, ‘4XXX’의 표시가 1초 간격으로 반복하는 스케치를 작성해 보자. (X:는 꺼짐을 나타낸다)

(hint: DIG1~4에 연결된 핀을 제어해보자.)

완성된 코드를 **ARnn_4digit.ino**

로 저장해서 제출.

4.7.5 4-digit FND 제어 - DIY

DIY

‘XXX1’, ‘XX2X’, ‘X3XX’, ‘4XXX’의 표시가 1초 간격으로 반복하는

Hint

스케치를 작성해 보자. (X:는 꺼짐을 나타낸다)

완성된 코드를 **ARnn_4digit.ino**

로 저장해서 제출. (실기 시험 1.)

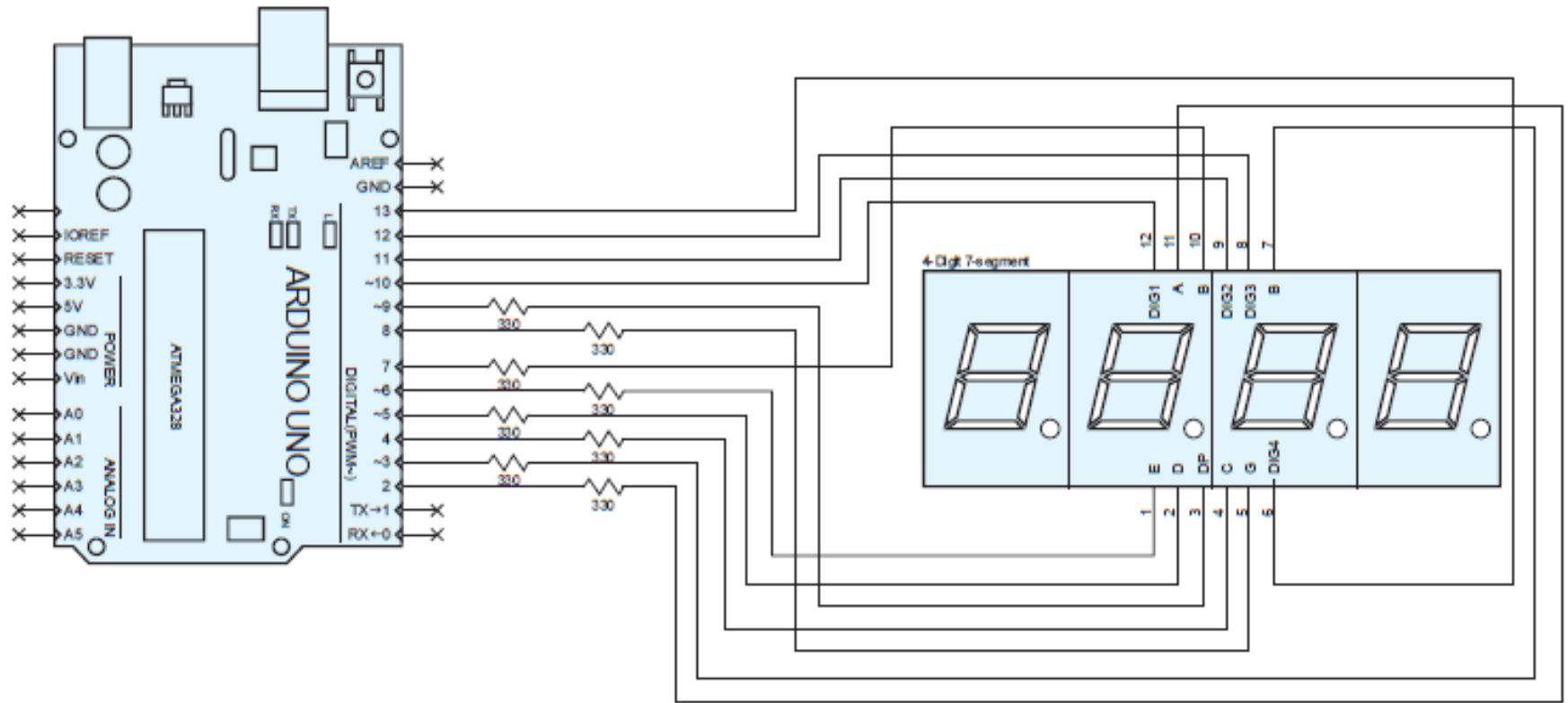
4.8.1 4-digit FND 제어 응용

EX 4.5.2 4-digit FND에 1초마다 증가하는 0~9999 숫자 표시하기 (1/3)

실습 목표 Common Cathode 4-digit FND를 이용하여 0~9999까지 1초 간격으로 증가하는 스케치를 작성해 보자.

- Hardware**
1. 4-digit FND는 4개의 FND를 연결한 부품이다.
 2. 각각의 FND에는 DIG1~DIG4 네 개의 핀이 각각의 FND의 Common Cathode로 연결되어 있다.
 3. A~G, DP핀은 하나의 FND를 동작시킬 때와 같이 330 Ω 저항을 통하여 Arduino 2~9번 핀에 연결한다.
 4. 맨 왼쪽 FND를 동작시키려면 DIG1에만 LOW신호를 준 상태에서 A~G, DP 핀에 원하는 숫자를 쓰기 위한 신호를 주어야 한다.
 5. 두번째 FND를 동작시키려면 DIG2에만 LOW신호를 준 상태에서 A~G, DP 핀에 원하는 숫자를 쓰기 위한 신호를 주어야 한다.
 6. DIG3, DIG4에 대해서도 동작을 반복한다.
 7. 각각의 FND를 선택하여 점등하는 동작을 빠른 속도로 반복하면 마치 모든 FND가 점등된 것으로 인식된다.

4.8.2 4-digit FND 제어 응용



4.8.3 4-digit FND 제어 응용

EX 4.5.2

4-digit FND에 1초마다 증가하는 0~9999 숫자 표시하기 (2/3)

Commands • void 함수(변수1, 변수2, ...){
};

‘함수(변수1, 변수2)’ 를 이용하여 ‘{ }’ 내의 명령을 호출하여 사용한다. ‘변수1’과 ‘변수2’등을 함께 선언하면 함수 내에서 그 변수를 사용할 수 있다. 반복되는 구문을 설정해 놓고 호출하여 사용하면 편리하다.

- pinMode(핀번호, 설정)

핀의 입출력 모드를 설정한다. ‘핀번호’ 에는 설정하고자 하는 핀의 번호와 ‘설정’에는 입력으로 사용하기 위해선 ‘INPUT’, 출력으로 사용하기 위해선 ‘OUTPUT’, 입력이며 풀업 사용시 ‘INPUT_PULLUP’을 적는다.

- digitalWrite(핀번호, 값)

핀에 디지털 출력 (High or Low) 을 한다. ‘핀번호’ 에는 출력하고자 하는 핀의 번호를, ‘값’에는 ‘HIGH’ 혹은 ‘LOW’ 를 설정하여 High 혹은 Low 출력을 한다.

- millis()

현재 스케치가 시작된 이후로 경과된 시간 값을 가져온다. 밀리세컨즈(1/1000초) 단위의 값을 갖는다.

4.8.4 4-digit FND 제어 응용

- Sketch 구성
1. FND에 숫자를 표시할 때 어떤 LED를 켤지에 대한 정보를 담은 상수를 설정한다.
 2. FND동작에 필요한 핀을 출력으로 설정한다.
 3. DIG1~4중 하나만 점등 한 뒤 해당 DIG 핀에 연결된 자릿수의 표시를 예제 4.4에서 설정한 'fndDisplay(int displayValue)' 함수를 응용하여 표시한다.
 5. DIG1->DIG2->DIG3->DIG4 순서로 돌아가며 점등시킨다. 해당 DIG핀에 신호를 LOW 했을 때 해당 자릿수가 점등된다.
 6. 빠른 시간으로 4개의 DIG핀을 제어하면 시각적으로 모든 FND가 점등된 것 처럼 보인다.

실습 결과 4개의 FND의 숫자가 0~9999까지 1단위로 약 1초 간격으로 변화한다.

응용 문제 숫자가 증가하는 간격을 0.5초로 변경하여라.

delay()를 사용하지 않고 **millis()**로 실제 시간을 측정하여 1초가 경과되면 숫자를 1씩 증가시키고 표시한다.

4.8.4.1 4-digit FND 제어 응용 - code1

```

6 // 0~9까지 LED 표시를 위한 상수
7 const byte number[10] = {
8   //dot  gfedcba
9   B00111111, //0
10  B00000110, //1
11  B01011011, //2
12  B01001111, //3
13  B01100110, //4
14  B01101101, //5
15  B01111101, //6
16  B00000111, //7
17  B01111111, //8
18  B01101111, //9
19 };
20
21 // 4개의 digit에 연결된 핀 설정
22 const byte digitNumber[4] = {13, 12, 11, 10};
23
24 // 표시할 숫자 변수
25 int count = 0;
26
27 // 각 자릿수를 저장하기 위한 변수
28 int value[4];
29
30 // 4개의 digit에 각각 다른 숫자를 표시하기 위해 사용하는 변수
31 int digitSelect = 1;
32
33 // 시간을 측정하는데 사용되는 변수
34 long sampleTime;
35 int count5ms;

```

```

37 void setup()
38 {
39   // 2~9번 핀을 a b c d e f g dot 의 순서로 사용한다.
40   // 10~13번 핀을 Digit 1~4 의 순서로 사용한다.
41   for(int i = 2; i <= 13; ++i){
42     pinMode(i, OUTPUT); // 2~13번핀을 출력으로 설정한다.
43   };
44
45   // 4 digit와 연결된 10~13번핀에 모두 HIGH 신호를
46   // 줘서 소등시킨다.
47   for(int i=10; i<=13; ++i){
48     digitalWrite(i, HIGH);
49   };
50 }

```


4.8.4.2 4-digit FND 제어 응용 - code2

```

51 void loop()
52 {
53     // 현재 시간을 저장한다.
54     sampleTime = millis();
55
56     // count 변수값을 FND에 출력한다.
57     fndDisplay(digitSelect, value[digitSelect-1]);
58     ++digitSelect;
59     if(digitSelect >= 5) digitSelect = 1;
60
61     // count 변수값이 0~9999의 범위를 갖도록한다.
62     if(count >= 9999) count = 0;
63     else{
64         // 앞서 저장한 시간에서 현재까지의 시간이 5ms일 경우에 다음 명령어를 실행한다.
65         while(millis()-sampleTime < 5);
66
67         ++count5ms;
68         if(count5ms > 200){ //
69             // 5ms * 200 = 1 s   때 count를 하나 올려준다
70             ++count;
71
72             // 변수를 각 자릿수로 나눈다
73             value[3] = count / 1000;
74             value[2] = (count - (value[3]*1000)) / 100;
75             value[1] = (count - (value[3]*1000) - (value[2]*100)) / 10;
76             value[0] = count - (value[3]*1000) - (value[2]*100) - (value[1]*10);
77
78             count5ms = 0;
79         }
80     }
81 }

```

```

82 // LED 켜는 루틴
83 void fndDisplay(int digit, int displayValue){
84     // bitValue 변수를 선언한다.
85     boolean bitValue;
86
87     // 4 digit와 연결된 10~13번핀에 모두 HIGH 신호를 줘서 소등시킨다.
88     for(int i=1; i<=4; ++i){
89         digitalWrite(digitNumber[i-1], HIGH);
90     };
91
92     // FND에 원하는 숫자를 표시한다.
93     for(int i=0; i<=7; ++i){
94         // number 상수의 하나의 비트값을 읽는다.
95         bitValue = bitRead(number[displayValue], i);
96         // 앞서 읽은 비트값을 2~9번핀에 출력시킨다.
97         digitalWrite(i+2, bitValue);
98     };
99
100     // 4 digit중 표시를 원하는 digit만 켜다
101     for(int i=1; i<=4; ++i){
102         // 표시하기 원하는 자릿수는 LOW신호를 주어 켜고 나머진 OFF시킨다.
103         if(digit == i) digitalWrite(digitNumber[i-1], LOW);
104         else digitalWrite(digitNumber[i-1], HIGH);
105     };
106 }

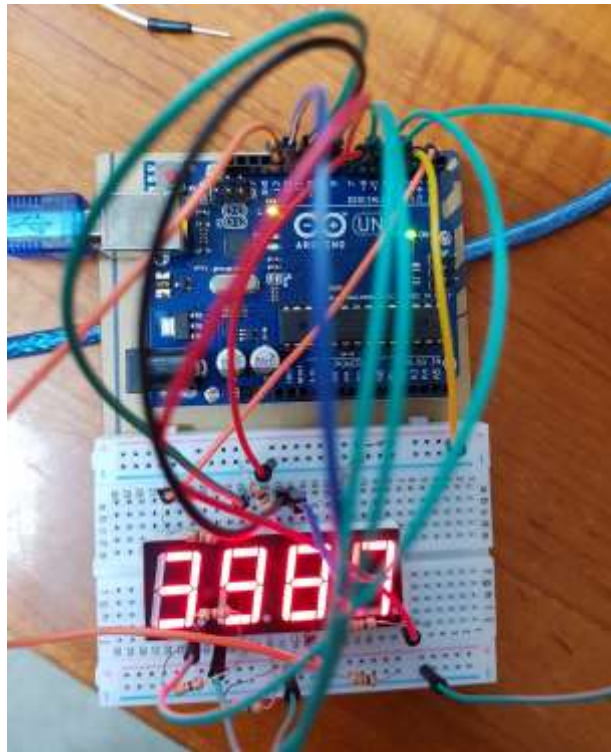
```

delay()를 사용하지 않고 **millis()**로 실제 시간을 측정하여
1초가 경과되면 숫자를 1씩 증가시키고 표시한다.

4.8.5 4-digit FND 제어 응용 - DIY

DIY 숫자가 증가하는 간격을 0.5초로 변경하여라.

4개의 숫자가 다르게 출력된 화면을 [ARnn_4digit.png](#)
로 저장해서 제출. (아두이노 회로를 포함해서 촬영)



8 X 8 Dot matrix

- ✓ 여러 개의 LED가 배열되어 문자나 기호를 표시하는 장치
- ✓ 8X8 Dot matrix는 64개의 LED를 이용
- ✓ LED를 빠르게 교차 출력하여 동시에 모든 LED가 제어되는 듯한 착시를 이용



그림 4.8 실험에 사용할 도트매트릭스

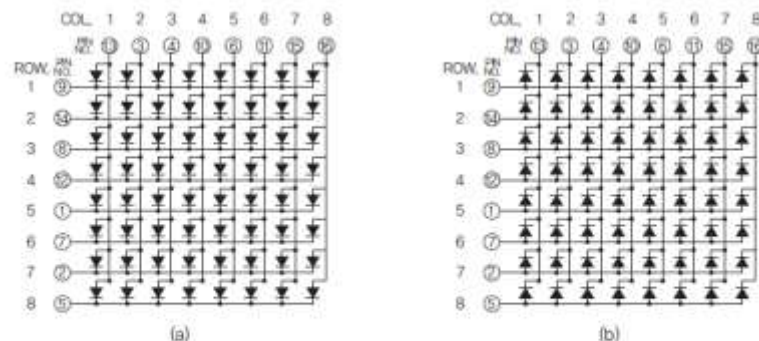


그림 4.9 행에 Anode(+연결)를 연결하고 열에 Cathode(-연결)를 연결한 형태(a)와 행에 Cathode(-연결)를 연결하고 열에 Anode(+연결)한 형태(b).

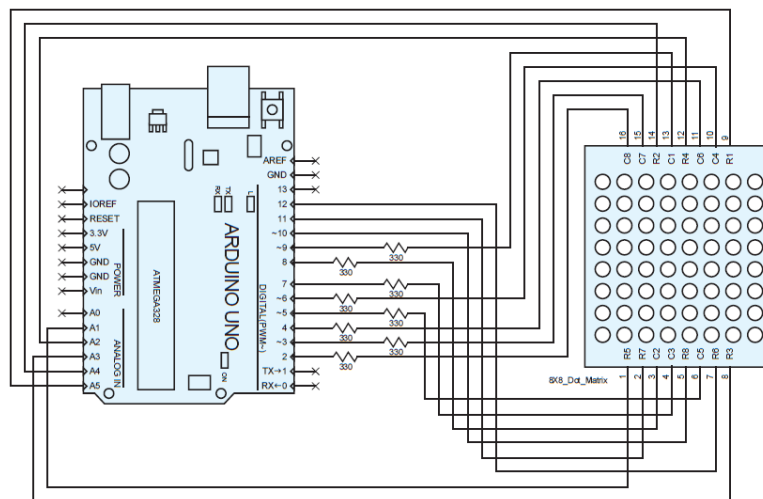
4.9.1 Dot matrix 제어

EX 4.6

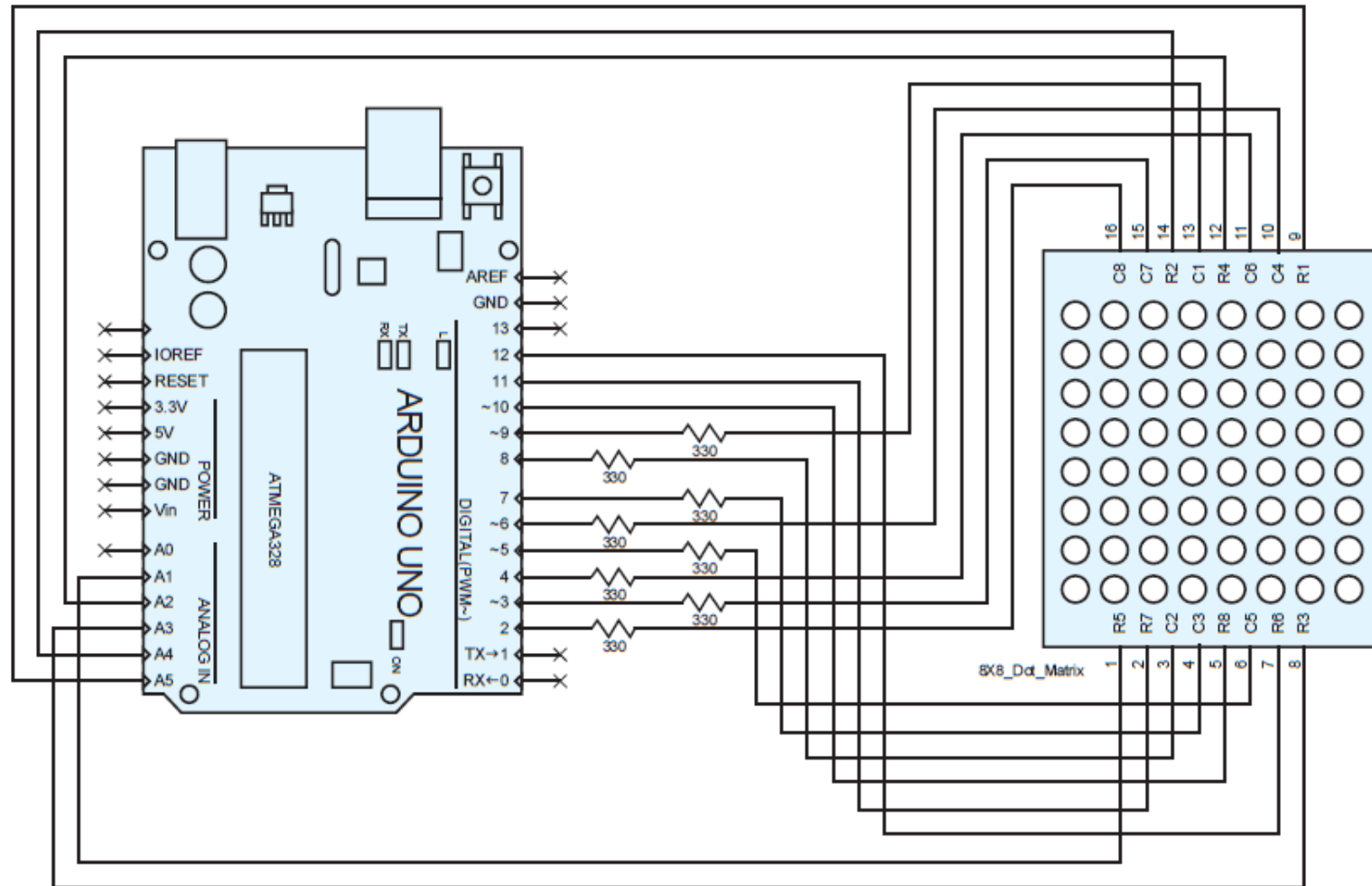
Dot matrix 제어 (1/3)

실습목표 8x8 Dot matrix로 변화하는 막대그래프를 표현해 보자.

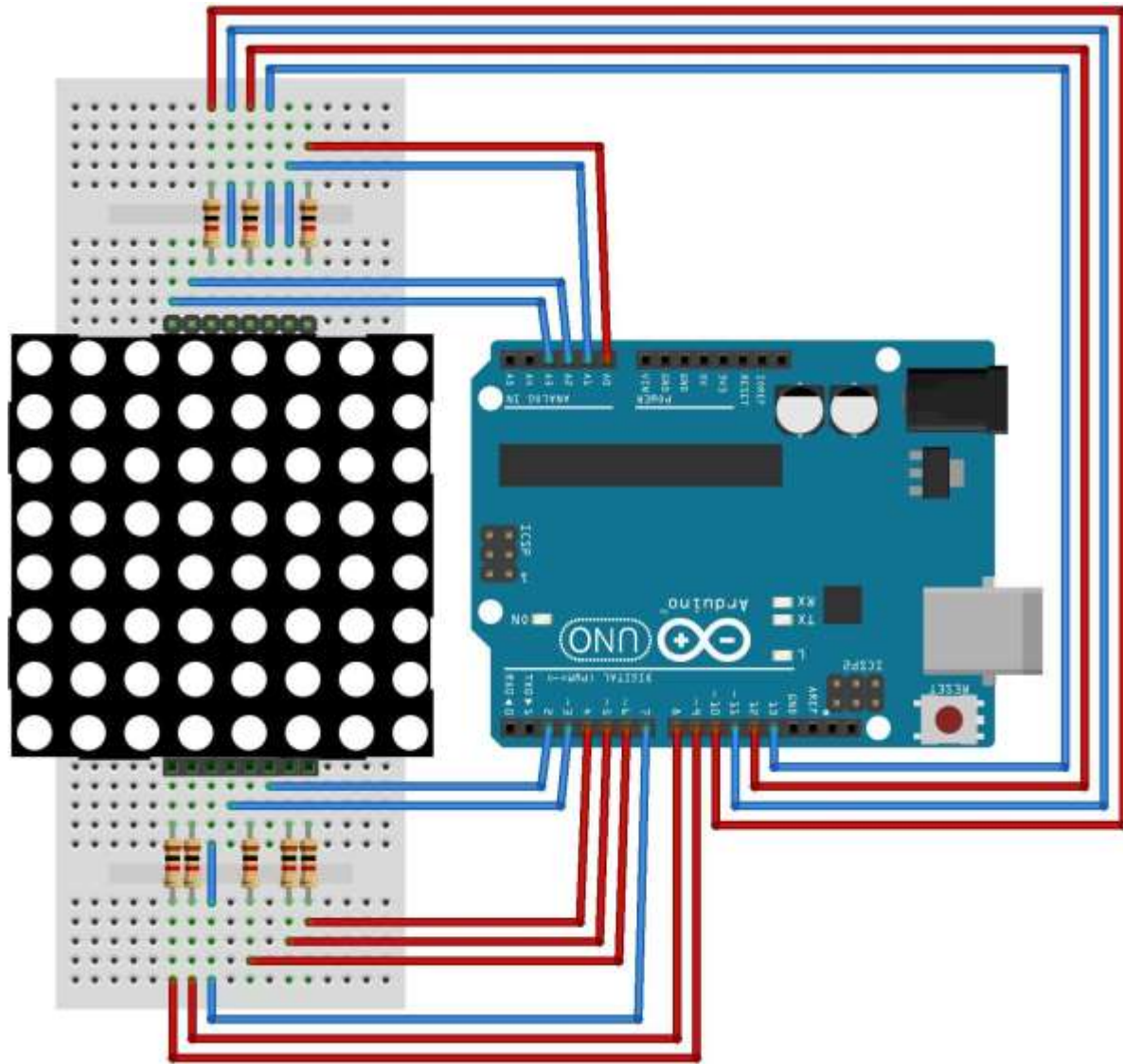
- Hardware**
1. 행은 2~9번핀에 연결하고 열은 10, 11, 12, A1~A5 번핀에 연결한다. 행을 연결할 때는 330Ω저항을 함께 연결한다.
 2. 실험에 사용할 8X8 Dot matrix는 행(column)에 Cathode, 열(row)에 Anode가 연결된 형태이다. 즉 행에 LOW신호, 열에 HIGH신호를 주어야 Dot LED가 켜진다.
 3. 특정 부분의 Dot LED를 점등하려면 그 부분의 행에 LOW신호, 열에 HIGH신호를 준다.



4.9.2 Dot matrix 제어



4.9.2 Dot matrix 제어



fritzing

4.9.3.1 Dot matrix 제어

EX 4.6 Dot matrix 제어 (2/3)

Commands • void 함수(변수1, 변수2, ...){
};

‘함수(변수1, 변수2)’ 를 이용하여 ‘{ }’ 내의 명령을 호출하여 사용한다. ‘변수1’과 ‘변수2’등을 함께 선언하면 함수 내에서 그 변수를 사용할 수 있다. 반복되는 구문을 설정해 놓고 호출하여 사용하면 편리하다.

- pinMode(핀번호, 설정)

핀의 입출력 모드를 설정한다. ‘핀번호’ 에는 설정하고자 하는 핀의 번호와 ‘설정’에는 입력으로 사용하기 위해선 ‘INPUT’, 출력으로 사용하기 위해선 ‘OUTPUT’, 입력이며 풀업 사용시 ‘INPUT_PULLUP’을 설정한다.

- digitalWrite(핀번호, 값)

핀에 디지털 출력 (High or Low) 을 한다. ‘핀번호’ 에는 출력하고자 하는 핀의 번호를, ‘값’에는 ‘HIGH’ 혹은 ‘LOW’ 를 설정하여 High 혹은 Low 출력을 한다.

- for(변수=시작 값 ; 조건 ; 변수의 증분){ }

변수의 시작 값부터 조건이 만족하는 경우 ‘{ }’ 내의 명령을 수행한다. ‘변수의 증분’에서는 1회 명령이 수행될 때 마다 변수를 증가 혹은 감소시킨다.

4.9.3.2 Dot matrix 제어

EX 4.6 Dot matrix 제어 (3/3)

- Sketch 구성**
1. 8X8 Dot matrix는 그림 4.5의 (b) 그림의 행에 Cathode, 열에 Anode가 연결된 형태를 사용할 것이다.
 2. 행과 열에 출력에 사용할 핀을 모두 출력으로 설정한다.
 3. 점등하고자 하는 행에 LOW 신호를 준 뒤 열에 HIGH 신호를 주어 LED를 점등시킨다.
 4. 행을 하나씩 증가하여 점등시킨다.
- 실험 결과** C8 부터 C1로 한 칸씩 이동하면서 쌓이는 막대그래프가 출력된다.
- 응용 문제** Dot가 한 개씩 이동하는 스케치를 만들어보자.

4.9.3.3 Dot matrix 제어 - code

ex_4_6_start

```

5
6 const int colPins[] = { 2, 3, 4, 5, 6, 7, 8, 9};
7 //          C8, C7, C6, C5, C4, C3, C2, C1
8 const int rowPins[] = { 10,11,12,15,16,17,18,19};
9 //          R8, R7, R6, R5, R4, R3, R2, R1
10
11 void setup() {
12   for (int i = 0; i < 8; i++)
13   {
14     // 행을 출력으로 설정한다
15     pinMode(colPins[i], OUTPUT);
16     // 열을 출력으로 설정한다
17     pinMode(rowPins[i], OUTPUT);
18   }
19 }
20

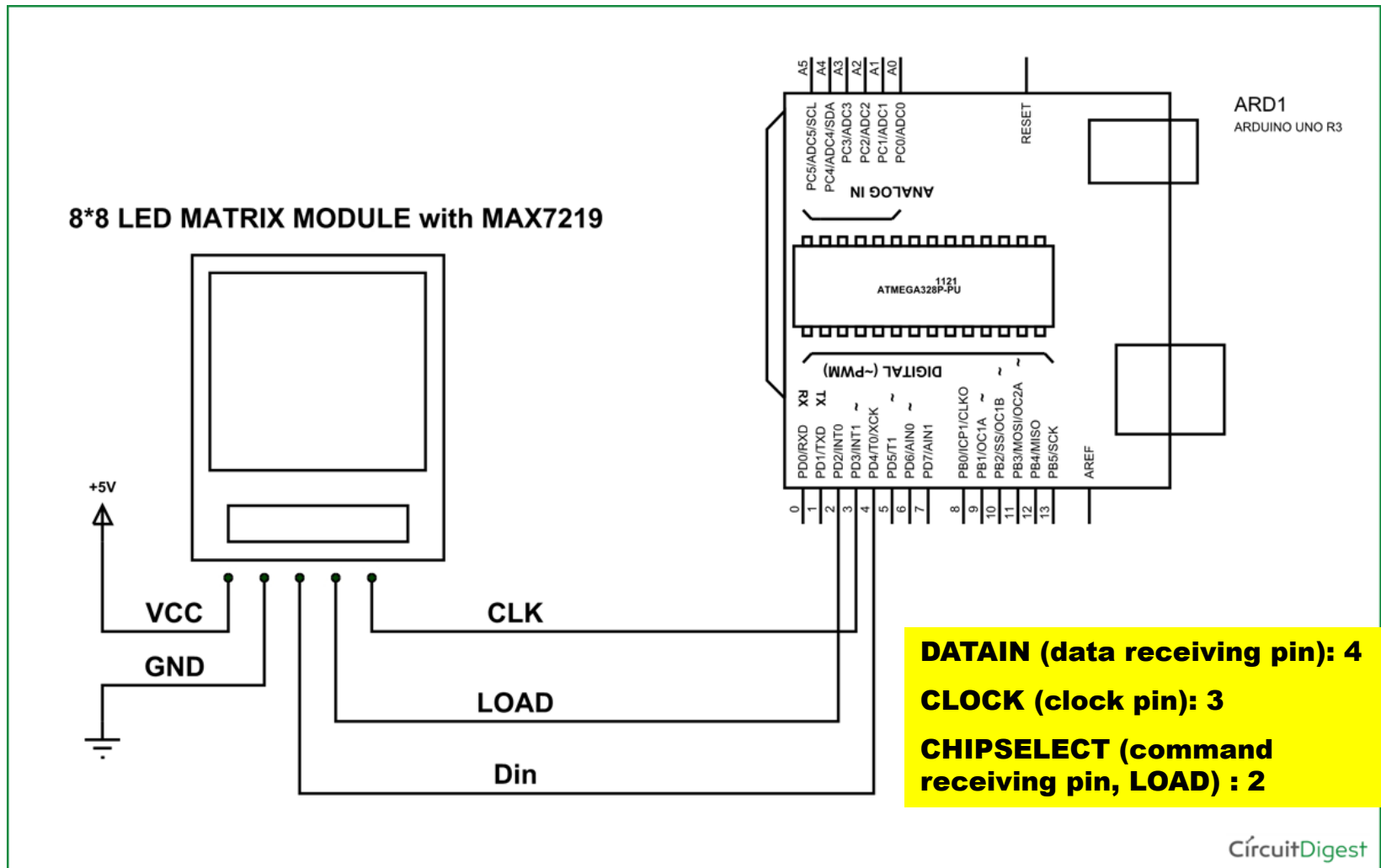
```

```

21 void loop() {
22
23   for (int column = 0; column < 8; column++)
24   {
25     // 행을 모두 초기화 한다
26     colClear();
27     // 현재의 행만 켜다
28     digitalWrite(colPins[column], LOW);
29
30     for(int row = 0; row < 8; row++)
31     {
32       // 열을 하나씩 켜다
33       digitalWrite(rowPins[row], HIGH);
34       delay(100);
35     }
36     // 열을 모두 초기화 한다
37     rowClear();
38   }
39   // 모든 행을 반복했으면 열을 모두 소등한다
40   rowClear();
41 }
42
43 // 행을 모두 초기화하는 루틴
44 void colClear(){
45   for(int i = 0; i < 8; i++){
46     digitalWrite(colPins[i], HIGH);
47   }
48 }
49
50 // 열을 모두 초기화하는 루틴
51 void rowClear(){
52   for(int i = 0; i < 8; i++){
53     digitalWrite(rowPins[i], LOW);
54   }
55 }

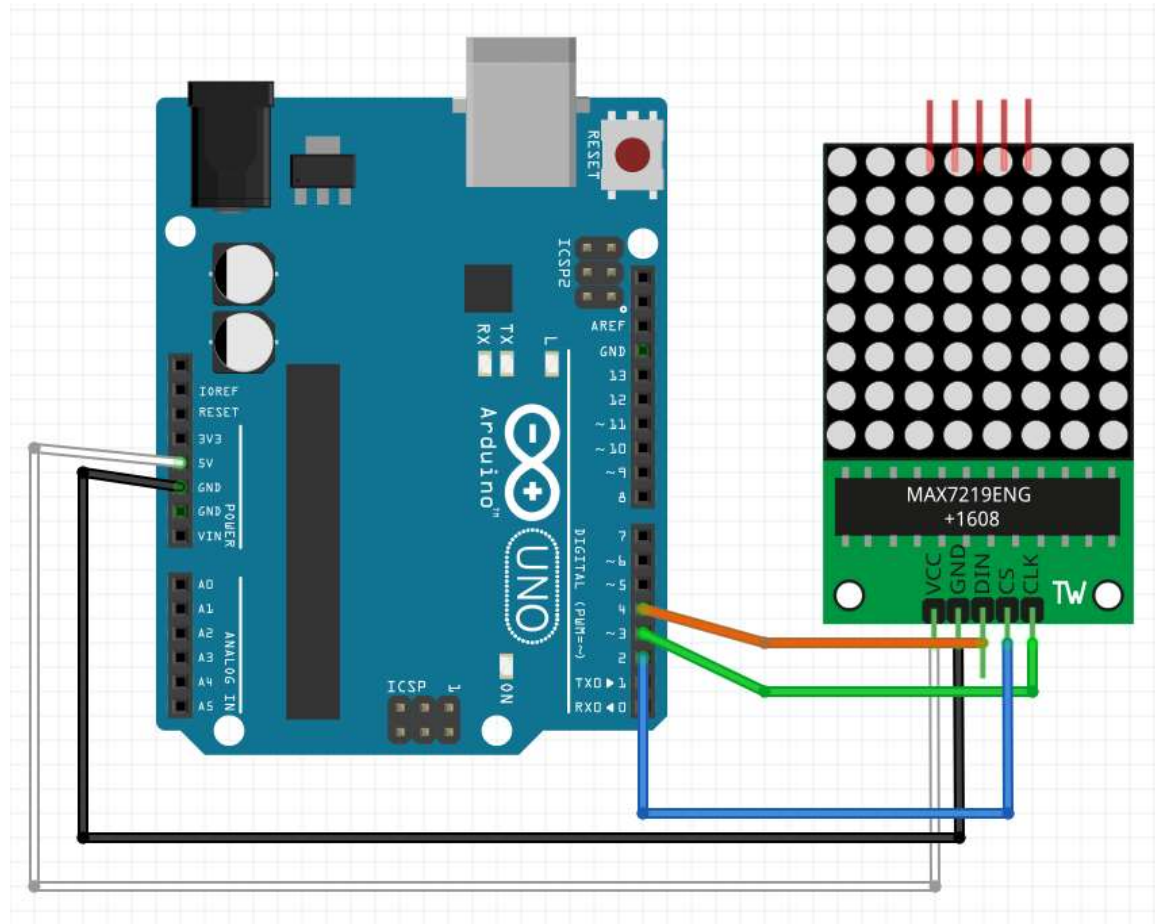
```

4.9.4 Dot matrix module 제어



<https://circuitdigest.com/microcontroller-projects/arduino-8x8-led-matrix>

4.9.4.1 Dot matrix module 제어



<http://forum.fritzing.org/t/max7219-dot-matrix-led-modul/1914>

Fritzing 으로 회로를 디자인하고

ARnn_dm_module.fzz 로 저장.

MAX7219 Dot Matrix Led Module 부품을 Fritzing에 연결

4.9.4.2 Dot matrix 제어 - code1

ex_4_7_dm_start

```
1 #include "LedControlMS.h"
2
3 // pin 4 is connected to the DataIn
4 // pin 3 is connected to the CLK
5 // pin 2 is connected to LOAD
6
7 #define NBR_MTX 1 //number of matrices attached is one
8 LedControl lc=LedControl(4,3,2, NBR_MTX);//
9
```



```
10 void setup()
11 {
12   for (int i=0; i< NBR_MTX; i++)
13   {
14     lc.shutdown(i,false);
15     /* Set the brightness to a medium values */
16     lc.setIntensity(i,8);
17     /* and clear the display */
18     lc.clearDisplay(i);
19
20     delay(100);
21   }
22 }
23
24 void loop()
25 {
26   //sending characters to display
27   lc.writeString(0,"Hello COMSI, AR00.");
28   //clearing the display
29   lc.clearAll();
30
31   delay(3000);
32 }
```

4.9.4.3 Dot matrix 제어 - code2.1

ex_dm_project

```

1 #include "LedControlMS.h"
2
3 #define NBR_MTX 1 //number of matrices attached is one
4
5 LedControl lc=LedControl(4,3,2, NBR_MTX); //
6
7 unsigned long delayTime=1200; // Delay between Frames
8
9 // Put values in arrays
10 byte invader1a[] =
11 {
12   B00011000, //
13   B00111100,
14   B01111110,
15   B11011011,
16   B11111111,
17   B00100100,
18   B01011010,
19   B10100101
20 };
21
22 byte invader1b[] =
23 {
24   B00011000, // Second frame
25   B00111100,
26   B01111110,
27   B11011011,
28   B11111111,
29   B00100100,
30   B01011010,
31   B01000010
32 };

```



```

34 byte invader2a[] =
35 {
36   B00100100, // First frame of invader #2
37   B00100100,
38   B01111110,
39   B11011011,
40   B11111111,
41   B11111111,
42   B10100101,
43   B00100100
44 };
45
46 byte invader2b[] =
47 {
48   B00100100, // Second frame of invader #2
49   B10100101,
50   B11111111,
51   B11011011,
52   B11111111,
53   B01111110,
54   B00100100,
55   B01000010
56 };

```

4.9.4.3 Dot matrix 제어 - code2.2

```

58 void setup()
59 {
60   lc.shutdown(0,false); // Wake up displays
61   // lc.shutdown(1,false);
62   lc.setIntensity(0,5); // Set intensity levels
63   // lc.setIntensity(1,5);
64   lc.clearDisplay(0); // Clear Displays
65   // lc.clearDisplay(1);
66 }
67
68
69 // Take values in Arrays and Display them
70 void sinvader1a()
71 {
72   for (int i = 0; i < 8; i++)
73   {
74     lc.setRow(0,i,invader1a[i]);
75   }
76 }
77
78 void sinvader1b()
79 {
80   for (int i = 0; i < 8; i++)
81   {
82     lc.setRow(0,i,invader1b[i]);
83   }
84 }

```

```

86 //void sinvader2a()
87 //{
88 //  for (int i = 0; i < 8; i++)
89 //  {
90 //    lc.setRow(1,i,invader2a[i]);
91 //  }
92 //}
93 //
94 //void sinvader2b()
95 //{
96 //  for (int i = 0; i < 8; i++)
97 //  {
98 //    lc.setRow(1,i,invader2b[i]);
99 //  }
100 //}

```

```

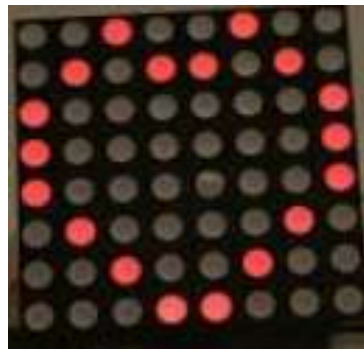
102 void loop()
103 {
104   // Put #1 frame on both Display
105   sinvader1a();
106   delay(delayTime);
107   // sinvader2a();
108   // delay(delayTime);
109
110
111   // Put #2 frame on both Display
112   sinvader1b();
113   delay(delayTime);
114   // sinvader2b();
115   // delay(delayTime);
116
117 }

```

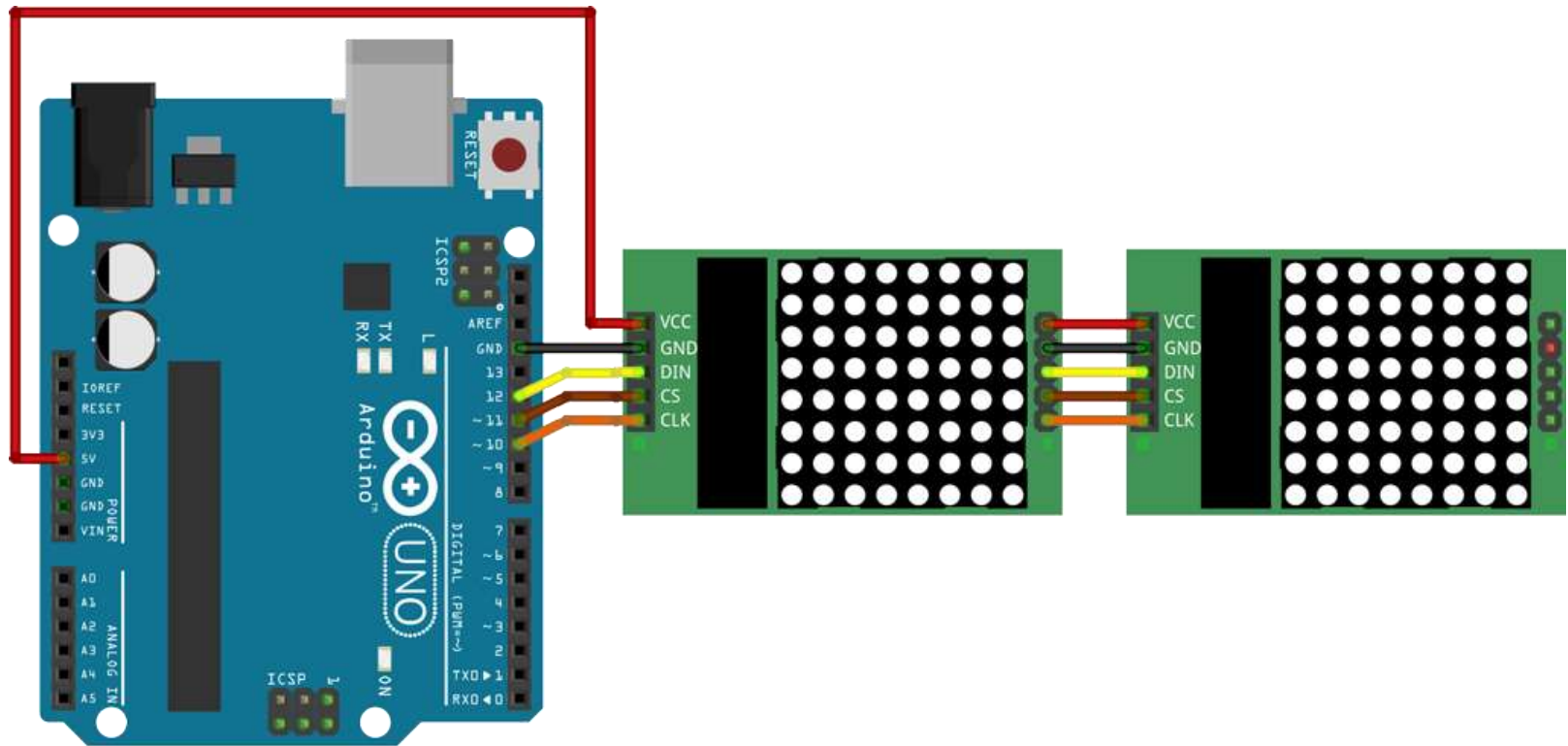
DIY dot matrix에 ♥ 를 그리는 코드와

♥ 사진을 제출하시오.

완성된 코드를 `ARnn_heart.ino` & `ARnn_heart.png`
로 저장해서 제출.



4.9.5 Dot matrix module 응용 - DM 연결



fritzing

<https://www.brainy-bits.com/how-to-control-max7219-led-matrix/>



[Practice]

◆ [wk05]

- **Arduino LED-II.**
- **Complete your project**
- **Submit folder : ARnn_Rpt04**

wk05 : Practice-04 : ARnn_Rpt04

◆ [Target of this week]

- Complete your works
- Save your outcomes
- Upload all in github.

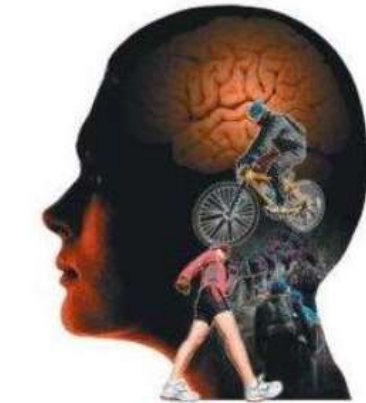
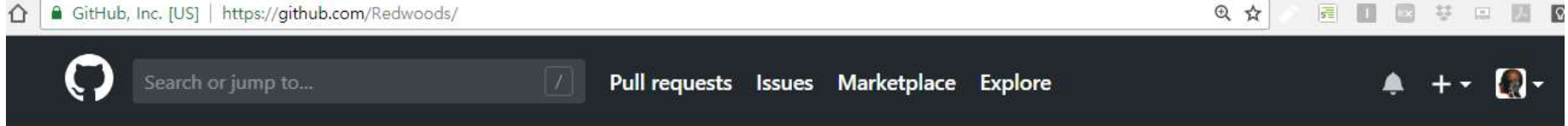
제출폴더명 : **ARnn_Rpt04**

- 제출할 파일들

- ① **ARnn_fnd.fzz**
- ② **ARnn_A.png**
- ③ **ARnn_74595_E.png**
- ④ **ARnn_4digit.fzz**
- ⑤ **ARnn_4digit.png**
- ⑥ **ARnn_4digit.ino**
- ⑦ **ARnn_heart.ino**
- ⑧ **ARnn_heart.png**
- ⑨ **All *.ino**

● References & good sites

- ✓ <http://www.arduino.cc> Arduino Homepage
- ✓ <http://www.github.com> GitHub
- ✓ <http://www.google.com> Googling
- ✓ <https://www.youtube.com> Youtube



Redwoods Yi

Redwoods

Add a bio

GimHae, Republic of Korea

chaos21c@gmail.com

Overview

Repositories 7

Stars 2

Followers 1

Following 0

Pinned repositories

Customize your pinned repositories

Py

Lectures on coding python from scratch to the advanced level.

Jupyter Notebook

Arduino

Lectures on learning Arduino from scratch to the advanced level in iot environment.

Lec

All lectures by Redwoods in Inje University

Jupyter Notebook

hw-coding

Resource for lecture of Hardware Programming (2017, Inje university)


Arduino


171 contributions in the last year

Contribution settings

Redwoods/Arduino: Lect

GitHub, Inc. [US] | https://github.com/Redwoods/Arduino

 Search or jump to... Pull requests Issues Marketplace Explore

 Search or jump to...

Unwatch 1 Star 0 Fork 0





Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings


Lectures on learning Arduino from scratch to the advanced level in iot environment. [Edit](#)

[Add topics](#)

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

 Redwoods 2018 start Latest commit 38ca9e0 28 minutes ago		
 ar-basic	2018 start	28 minutes ago
 ar-iot	2018 start	28 minutes ago
 README.md	Initial commit	43 minutes ago

 README.md

Arduino

Lectures on learning Arduino from scratch to the advanced level in iot environment.





http://arduinostory.com/goods/goods_view.php?goodsNo=1000000306

상급키트 구성품

1 1EA 아두이노 우노 R3 DIP 아두이노 우노 R3 (DIP) 호환보드 기본 메인보드입니다.	2 1EA 9V 배터리 홀더 9V 배터리를 연결하여 아두이노에 외부전원을 공급할 수 있습니다.	3 1EA 7세그먼트 4채널 7세그먼트가 4개 연결된 형태의 부품입니다. 총 12개의 핀을 사용합니다.	4 1EA 7세그먼트 1채널 공통 음극 7세그먼트 시계나 점수 등의 숫자를 표현 할 때 많이 사용됩니다.	13 1EA 수동부저 아두이노의 tone함수를 통해 소리를 내는 부저입니다.	14 6EA 택트스위치 (12x12x7) 스위치를 누르고 있을 경우만 ON됩니다.	15 각3EA 택트스위치 컵 (피랑,노랑,초록,빨강,하양) 택트스위치를 사용할 때 스위치간의 구분을 할 수 있습니다.	16 3EA 조도센서 빛을 감지하거나 빛의 밝기를 아날로그로 출력해주는 CDS 센서입니다.
5 1EA 74HC595N 기본 메인보드입니다. 74HC595N LED, 드로메트릭스, NFD 제어 IC 입니다.	6 1EA 65핀 점퍼 와이어 브레드보드에 연결할 때 사용하는 65핀 점퍼와이어 입니다.	7 1EA 무지개 점퍼선 F-M 20cm M타입과 F타입이 양쪽으로 달린 무지개 점퍼선입니다.	8 1EA 투명 부품 케이스 대,소 키트 구성품을 담을 수 있는 투명 부품 케이스입니다.	17 각5EA LED 5mm (빨강,노랑,초록,하양,파랑) 기본으로 사용되는 LED입니다. 동작전압 : 2.2~2.4V 사용전류 : 20mA 미만	18 1EA 헤더핀 1x40/2.54mm 핀 간격은 2.54mm이며 헤더핀의 길이는 약 1.15cm입니다.	19 1EA USB케이블 50cm PC와 아두이노 우노 보드를 연결하여 프로그램을 다운로드 할 때 사용합니다.	20 1EA 저항값 카드 저항값을 쉽게 확인 할 수 있는 카드입니다. 사이즈 : 60mm x 50mm
9 1EA 가변저항10K 물리변 저항값이 가능합니다. (0~10KΩ)	10 1EA 1602 I2C LCD 아두이노 16x2 I2C LCD 모듈입니다. LCD입니다.	11 각 10EA 저항 100, 220, 330, 1K, 2K, 4.7K, 10K, 47K, 100K	12 1EA 브레드 보드 830홀 브레드 보드 830홀(봉무형) 센서 테스트나, 회로 프로토타입을 작성할 때 사용됩니다.	21 1EA 능동부저 Signal 단자가 HIGH 일 때 약 2.5kHz의 음이 발생합니다.	22 1EA 5V 1채널 릴레이 모듈 아두이노의 디지털 핀과 모듈 하단의 IN 핀들을 연결해 릴레이를 제어할 수 있는 모듈입니다.	23 1EA 8x8 도트 매트릭스 모듈 LED로 다양한 연출을 할 수 있습니다.	24 1EA 4x4 16 핀패드 모듈 16개의 버튼을 사용할 수 있습니다.

아두이노 키트(Kit) : Part-2

<div>25</div> <div>1EA</div> <div></div> <div>무선 리모콘 키트</div> <div>핵파선을 사용해서 리모콘 기능을 구현할 수 있습니다.</div>	<div>26</div> <div>2EA</div> <div></div> <div>가열기 센서 스위치</div> <div>센서의 가열기에 따라 스위치 역할을 합니다.</div>	<div>27</div> <div>1EA</div> <div></div> <div>or</div> <div>사운드 센서 모듈</div> <div>아두이노와 호환되는 사운드센서 모듈입니다.</div>	<div>28</div> <div>1EA</div> <div></div> <div>불꽃 센서</div> <div>근거리 화재, 불꽃을 감지하는 센서입니다.</div>	<div>37</div> <div>1EA</div> <div></div> <div>DC 5V 스텝 모터</div> <div>28BYJ 28BYJ48 스텝 모터 중 저렴한 편에 속하는 모델입니다. 5개의 핀을 사용합니다.</div>	<div>38</div> <div>1EA</div> <div></div> <div>DS1302 RTC 모듈</div> <div>아두이노 등 마이크로컨트롤러에서 사용이 가능합니다.</div>	<div>39</div> <div>1EA</div> <div></div> <div>아두이노 우노 프로토 쉼드</div> <div>UNO 보드에서 회로를 간단히 짜기 위해 보드 위에 얹어 사용하는 쉼드입니다.</div>	<div>40</div> <div>1EA</div> <div></div> <div>3축 가속도 센서 모듈</div> <div>가속도를 측정할 수 있는 센서입니다.</div>
<div>29</div> <div>1EA</div> <div></div> <div>모터 드라이버 모듈</div> <div>ULN2003 스텝 모터 드라이버 모듈 5V ~ 12V를 사용합니다.</div>	<div>30</div> <div>1EA</div> <div></div> <div>LM35 온도 센서</div> <div>온도를 아날로그 값으로 출력합니다.</div>	<div>31</div> <div>1EA</div> <div></div> <div>수위 센서 모듈</div> <div>센서가 액체에 잠긴 정도를 아날로그 값으로 출력합니다.</div>	<div>32</div> <div>1EA</div> <div></div> <div>SG90 서보모터</div> <div>Vcc, GND, 신호선, 총 3개의 핀이 있습니다. 로봇팔이나 자동차, 비행기 조종에 사용됩니다.</div>	<div>41</div> <div>1EA</div> <div></div> <div>5V DC모터</div> <div>5V DC모터</div>	<div>42</div> <div>1EA</div> <div></div> <div>인체 감지 센서 모듈</div> <div>핵파선을 이용해 움직임 감지하는 센서입니다. 오선이 감지되면 HIGH 신호를 출력합니다.</div>	<div>43</div> <div>5EA</div> <div></div> <div>다이오드 1N4001</div> <div>다이오드 1N4001</div>	<div>44</div> <div>5EA</div> <div></div> <div>세라믹 캐패시터 (22pF)</div> <div>세라믹 캐패시터 (22pF)</div>
<div>33</div> <div>1EA</div> <div></div> <div>초음파 거리 센서 모듈</div> <div>5V를 사용하여 인식 거리는 2cm에서 500cm입니다.</div>	<div>34</div> <div>1EA</div> <div></div> <div>조이스틱 모듈</div> <div>기본적으로 조이스틱 모듈은 두개의 가변저항이 서로 수직으로 회전하는 형태로 되어있습니다.</div>	<div>35</div> <div>1EA</div> <div></div> <div>온습도 센서 모듈</div> <div>아두이노 온습도 센서중 가장 대중적으로 사용되는 DHT11 디지털 센서입니다.</div>	<div>36</div> <div>1EA</div> <div></div> <div>RGB LED 모듈</div> <div>RGB LED 모듈로 RGB LED 세개를 하나로 묶은 상품입니다.</div>	<div>45</div> <div>5EA</div> <div></div> <div>세라믹 캐패시터 (1uF)</div> <div>세라믹 캐패시터 (1uF)</div>	<div>46</div> <div>5EA</div> <div></div> <div>트랜지스터 2N2222</div> <div>트랜지스터 2N2222</div>	<div>47</div> <div>5EA</div> <div></div> <div>트랜지스터 BC547</div> <div>트랜지스터 BC547</div>	<div>48</div> <div>5EA</div> <div></div> <div>트랜지스터 BC557</div> <div>트랜지스터 BC557</div>
<div>49</div> <div>2EA</div> <div></div> <div>전해 캐패시터 (50V 10uF)</div> <div>전해 캐패시터 (50V 10uF)</div>	<div>50</div> <div>2EA</div> <div></div> <div>전해 캐패시터 (50V 100uF)</div> <div>전해 캐패시터 (50V 100uF)</div>	<div></div>					