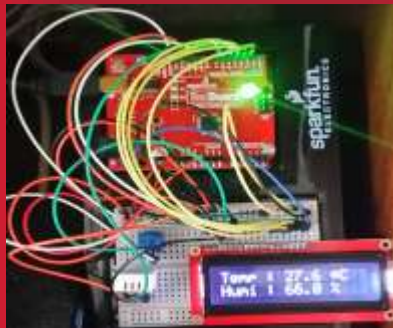
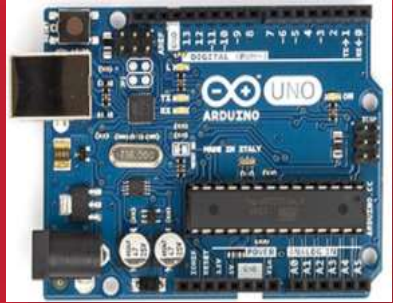


# Arduino-basic

[wk06]

## LED – III

## 4-digits FND



Learn how to code Arduino from scratch

Comsi, INJE University

1<sup>st</sup> semester, 2023

Email : [chaos21c@gmail.com](mailto:chaos21c@gmail.com)

예제제로 쉬게 배우는

# 아두이노

김진환 · 장성웅 지음

# My ID (ARnn, github repo)

AR01	강동하
AR02	정재윤
AR03	유석진
AR04	정창민
AR05	정희서
AR06	유동기
AR07	장세진
AR08	정호기

위의 **id**를 이용해서 **github**에 **repo**를 만드시오.

# wk05 : Practice-04 : ARnn\_Rpt04

## ◆ [Target of this week]

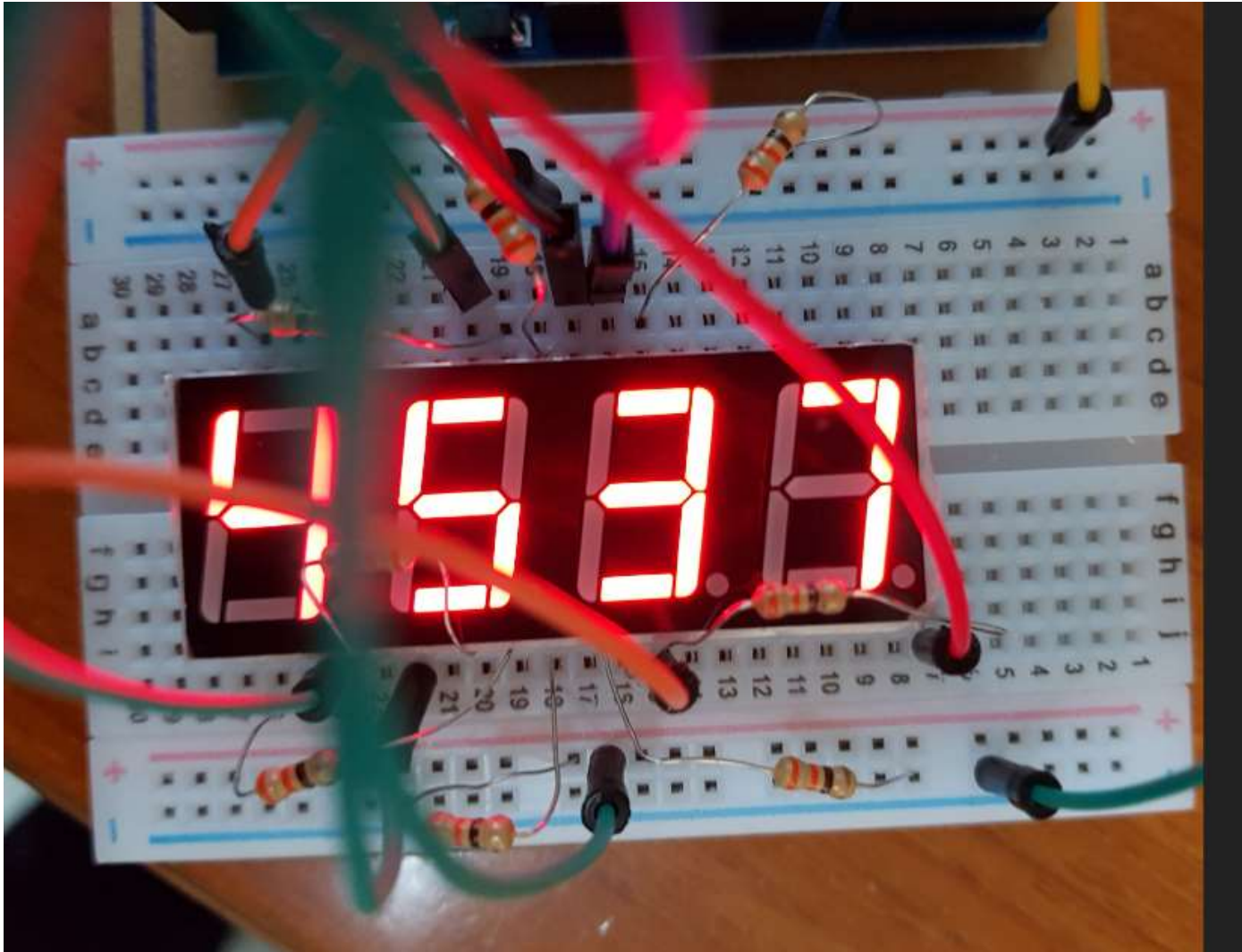
- Complete your works
- Save your outcomes
- Upload all in github.

제출폴더명 : **ARnn\_Rpt04**

### - 제출할 파일들

- ① **ARnn\_fnd.fzz**
- ② **ARnn\_A.png**
- ③ **ARnn\_74595\_E.png**
- ④ **All \*.ino**

## 4.7. 4-digit FND 제어





# 4. LED III

## 4-digits 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.

# 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

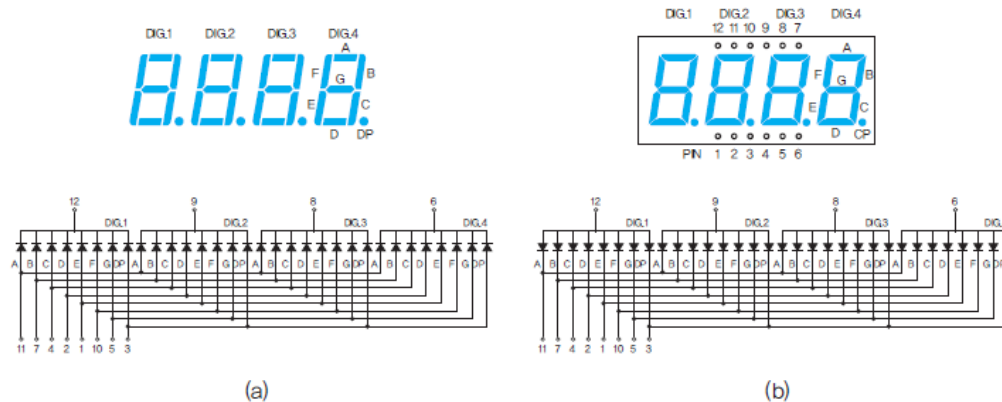


그림 4.6 4-digit FND와 내부 회로. Common Cathode (a), Common Anode (b)



그림 4.7 실험에 사용할 4-digit FND



# 4.7.1 4-digit FND 제어

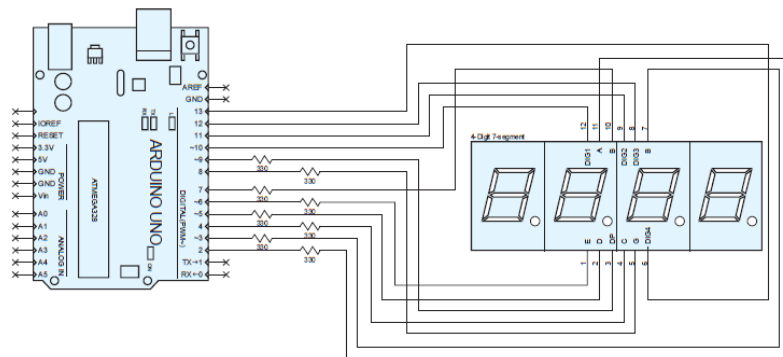
EX 4.5.1

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

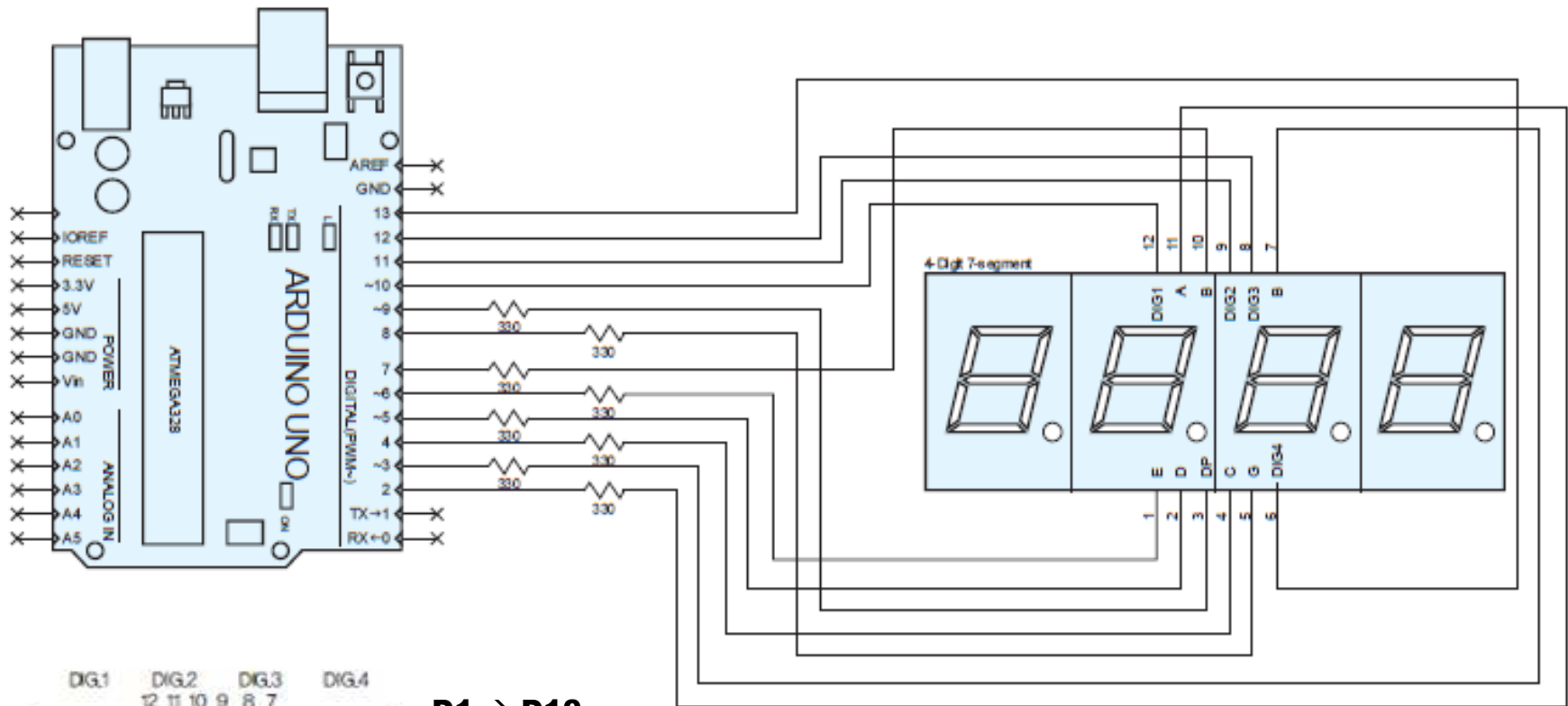
**실습목표** Common Cathode 4-digit FND를 이용하여 0000~9999까지 1초 간격으로 증가하는 스케치를 작성해 보자.

**Hardware**

1. 4-digit FND는 4개의 FND를 연결한 부품이다.
2. 각각의 FND에는 DIG1~DIG4(10~13) 네 개의 핀이 각각의 FND의 Common Cathode로 연결되어 있다.
3. A~G, DP핀은 하나의 FND를 동작시킬 때와 같이 220 $\Omega$ 저항을 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

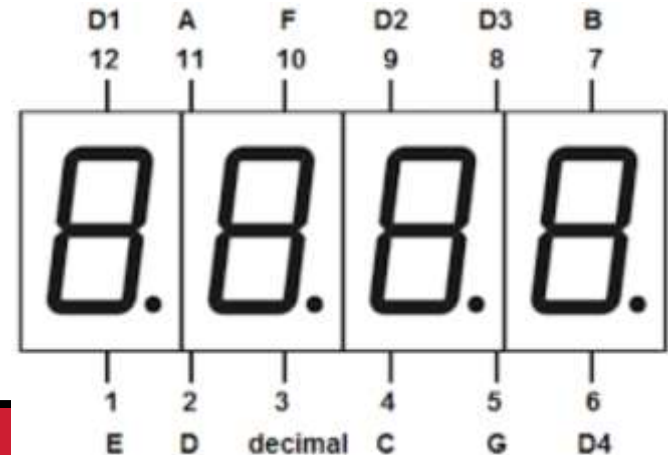


**D1 → D10**

**D2 → D11**

**D3 → D12**

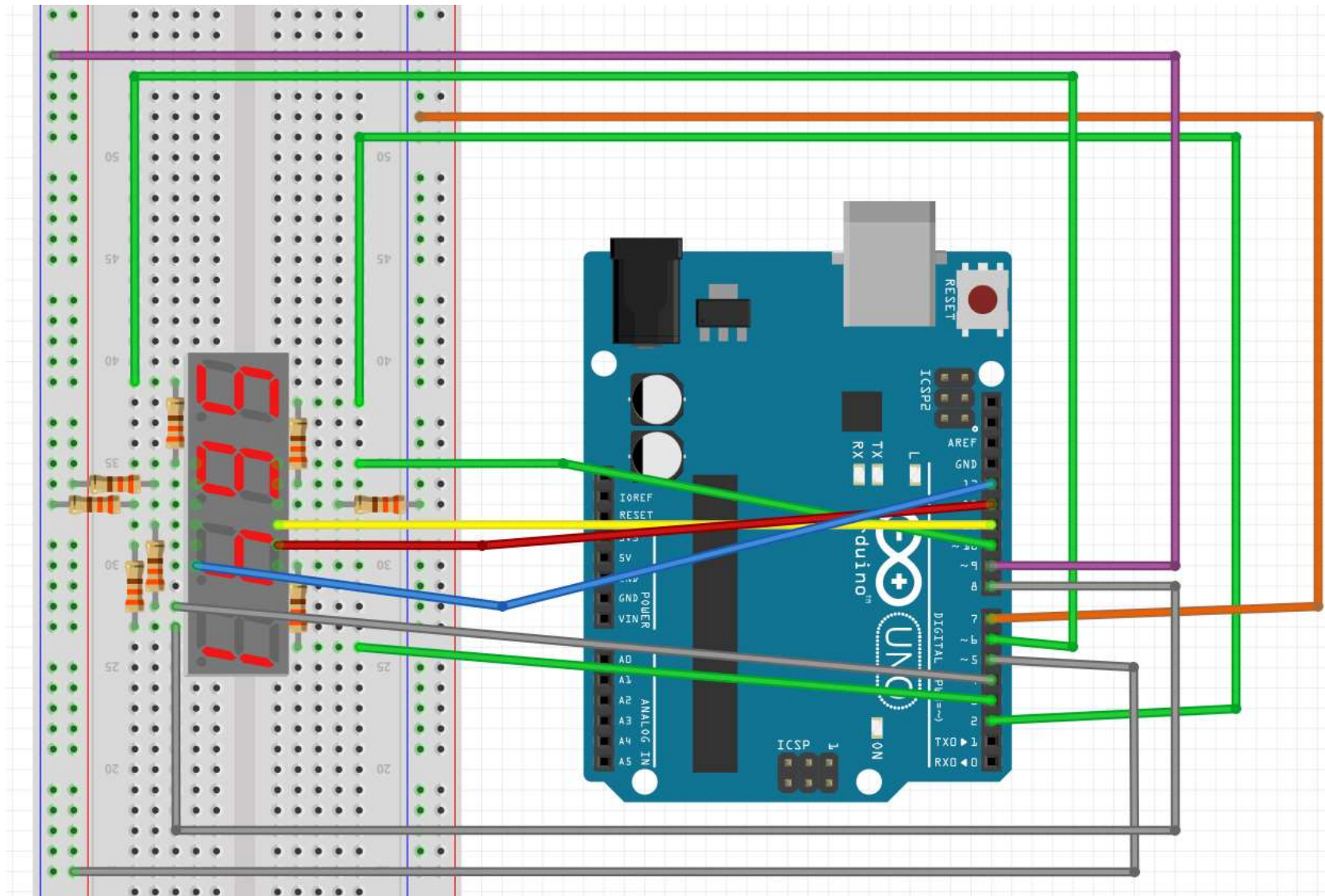
**D4 → D13**



<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 – circuit



Fritzing 으로 회로를 디자인하고  
[ARnn\\_4digit.fzz](#) 로 저장해서 제출.

## 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 같은 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

**ex\_4\_5\_1.ino**

```

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 }

```

DIY-1      '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-1

'XXX1', 'XX2X', 'X3XX', '4XXX' 의 표시가 1초 간격으로

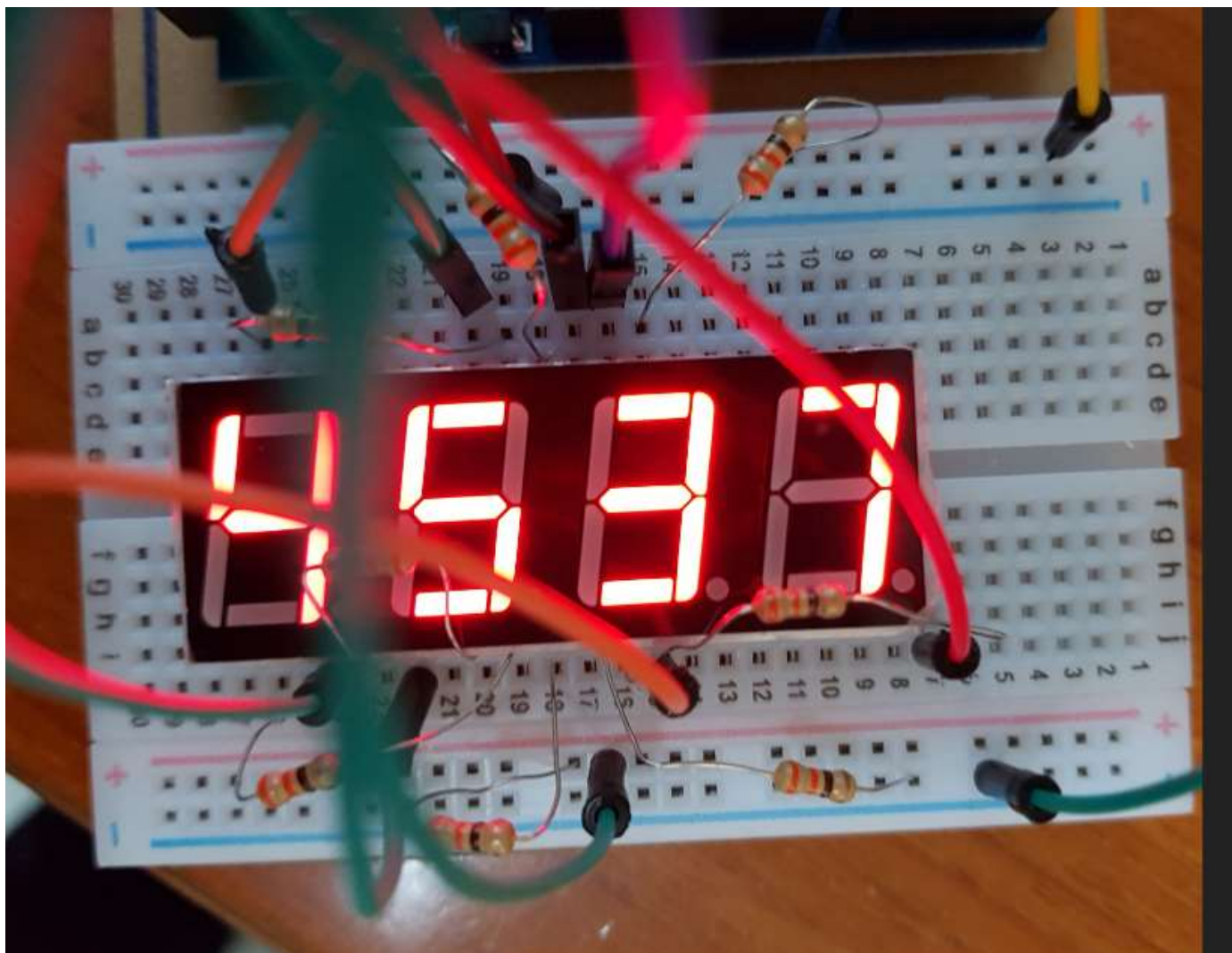
### Hint

반복하는 스케치를 작성해 보자. (X:는 꺼짐을 나타낸다)

완성된 코드를 **ARnn\_4digit.ino**

로 저장해서 제출.

## 4.8 4-digit FND 제어 응용 (참고 회로)



## 4.8.1 4-digit FND 제어 응용

EX 4.5.2

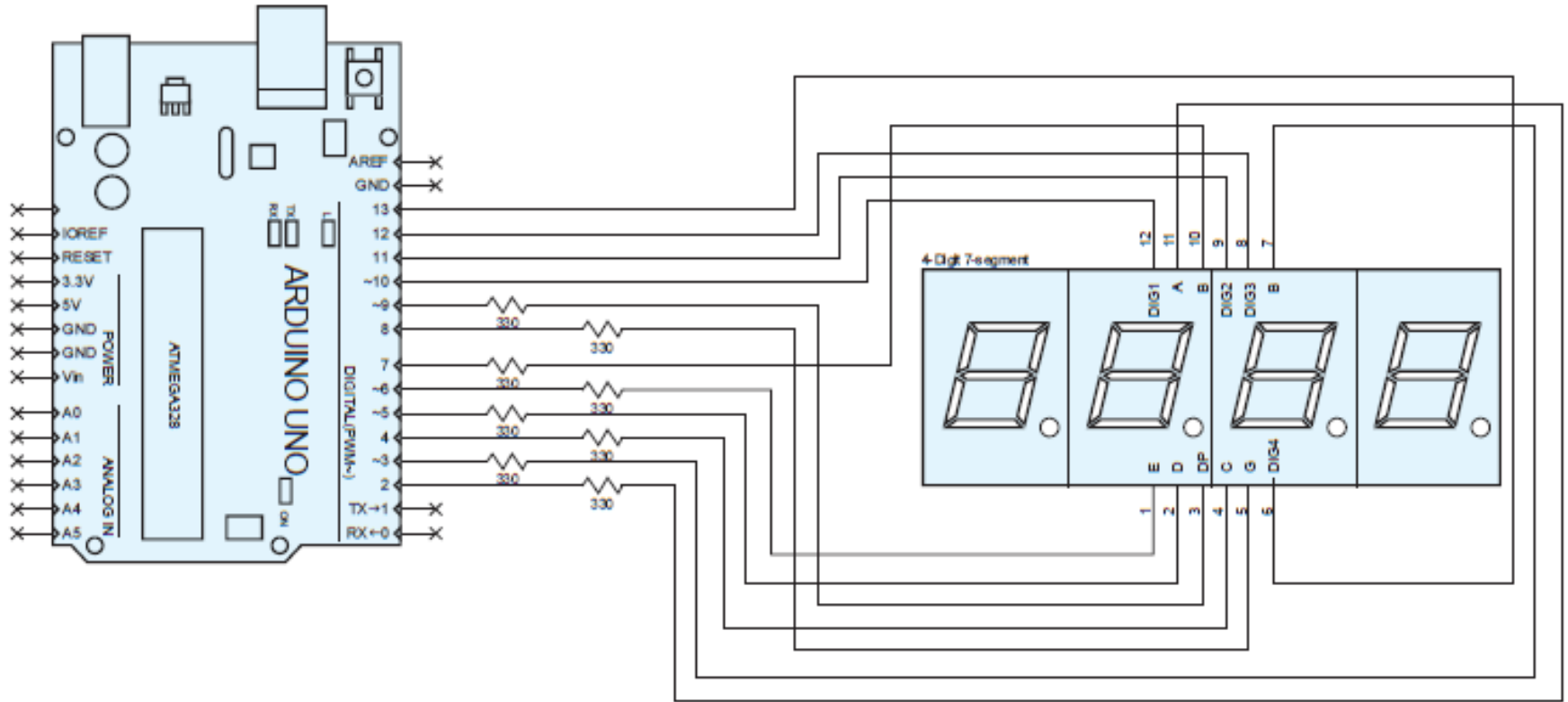
### 4-digit FND에 1초마다 증가하는 0~9999 숫자 표시하기 (1/3)

**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. (Arduino 10~13번 핀에 연결)
  3. A~G, DP핀은 하나의 FND를 동작시킬 때와 같이 220Ω저항을 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



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 켜지에 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.

```

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 }

```

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

```

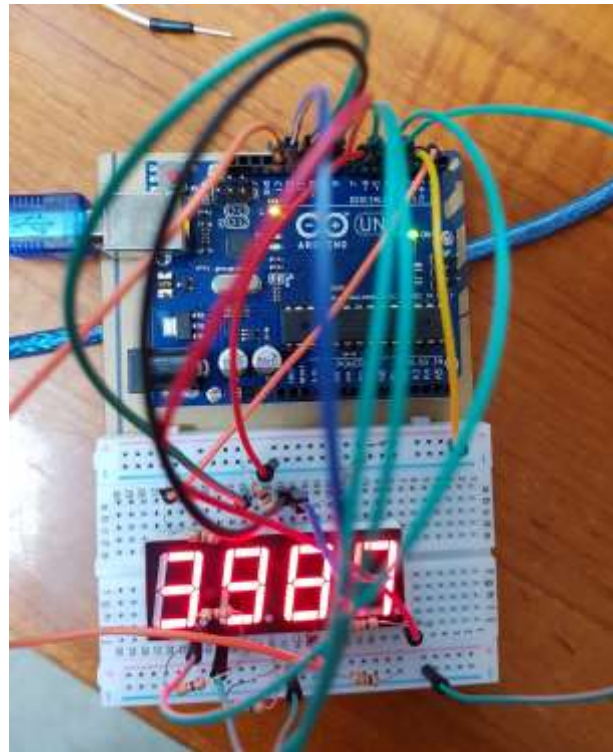
82 // LED 켜는 루틴
83 void fndDisplay(int digit, int displayValue){
84   // bitValue 변수를 선언한다.
85   boolean bitValue;
86
87   // 4 digit와 연결된 10~13번핀에 모두 HIGH 신호를 줘서 소등시킨다.
88   for(int i=1; i<=4; ++i){
89     digitalWrite(digitNumber[i-1], HIGH);
90   };
91
92   // FND에 원하는 숫자를 표시한다.
93   for(int i=0; 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 }

```

완성된 코드를 **ARnn\_4digit\_9999.ino**로 저장해서 제출.

DIY      숫자가 증가하는 간격을 0.5초로 변경하여라.

4개의 숫자가 다르게 출력된 화면을 [ARnn\\_4digit\\_9999.png](#)  
로 저장해서 제출. (아두이노 회로를 포함해서 촬영)





# [Practice]

## ◆ [wk06]

- **Arduino LED – III : 4-digits FND**
- **Complete your project**
- **Submit folder : ARnn\_Rpt05**

# wk06 : Practice-05 : ARnn\_Rpt05

## ◆ [Target of this week]

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

제출폴더명 : **ARnn\_Rpt05**

제출할 파일들

- ① **ARnn\_4digit.fzz**
- ② **ex\_4\_5\_1.ino**
- ③ **ARnn\_4digit.ino**
- ④ **ARnn\_4digit\_9999.ino**
- ⑤ **ARnn\_4digit\_9999.png**





# 4. LED IV

## DM & DM module

4 1EA



7세그먼트 1채널

---

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

5 1EA



74HC595N

---

기본 메인보드입니다.  
74HC595N LED,  
도트 매트릭스, NFD 제어 IC 입니다.

3 1EA



7세그먼트 4채널

---

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

23 1EA



8x8 도트 매트릭스 모듈

---

LED로 다양한 연출을  
할 수 있습니다.

8 X 8 Dot matrix

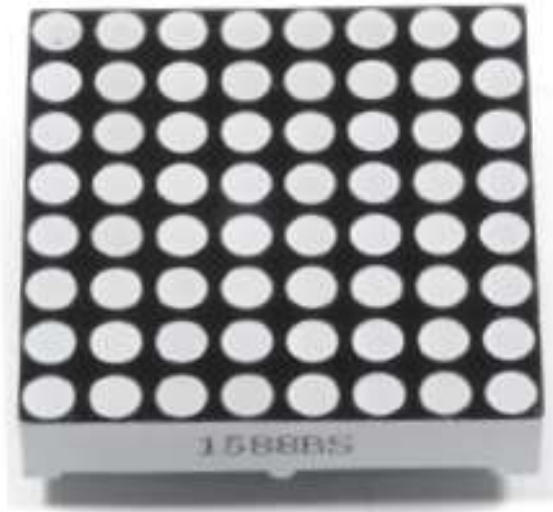


그림 4.8 실험에 사용할 도트매트릭스

## 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

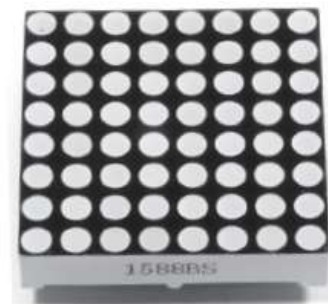


그림 4.8 실험에 사용할 도트매트릭스

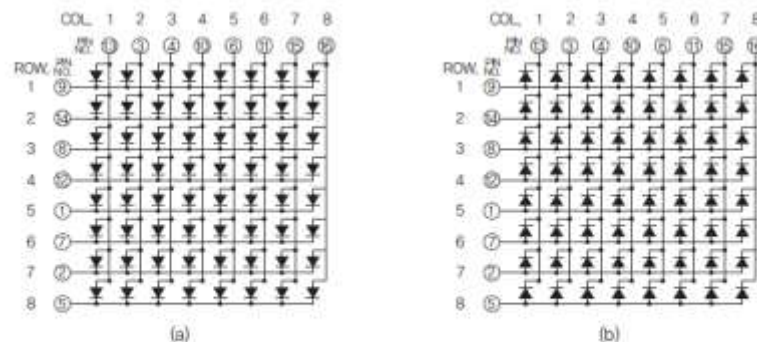
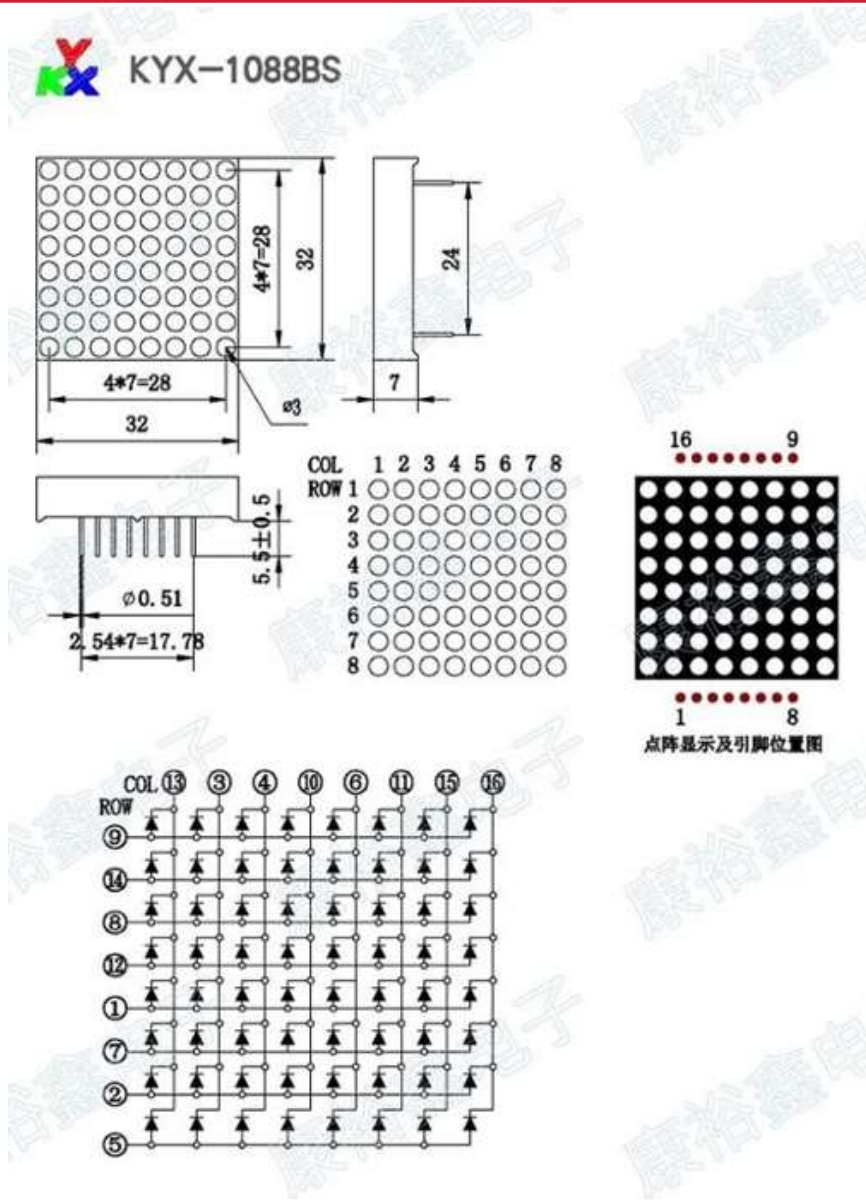


그림 4.9 행에 Anode(+연결)를 연결하고 열에 Cathode(-연결)를 연결한 형태(a)와 행에 Cathode(-연결)를 연결하고 열에 Anode(+연결)한 형태(b).

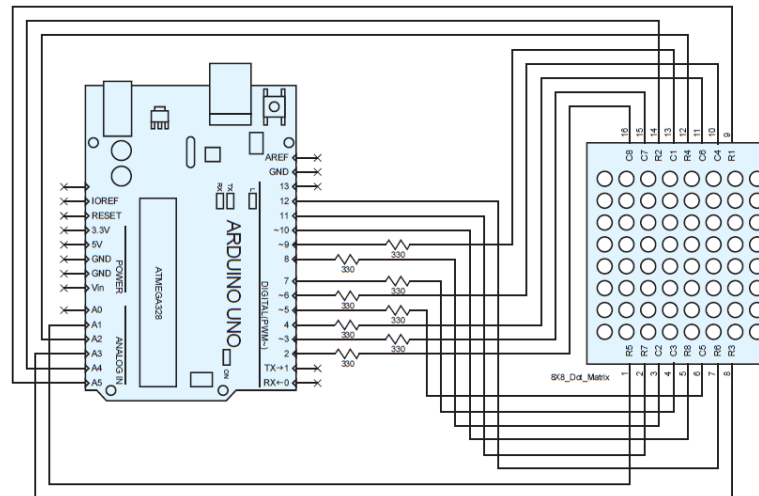


## EX 4.6

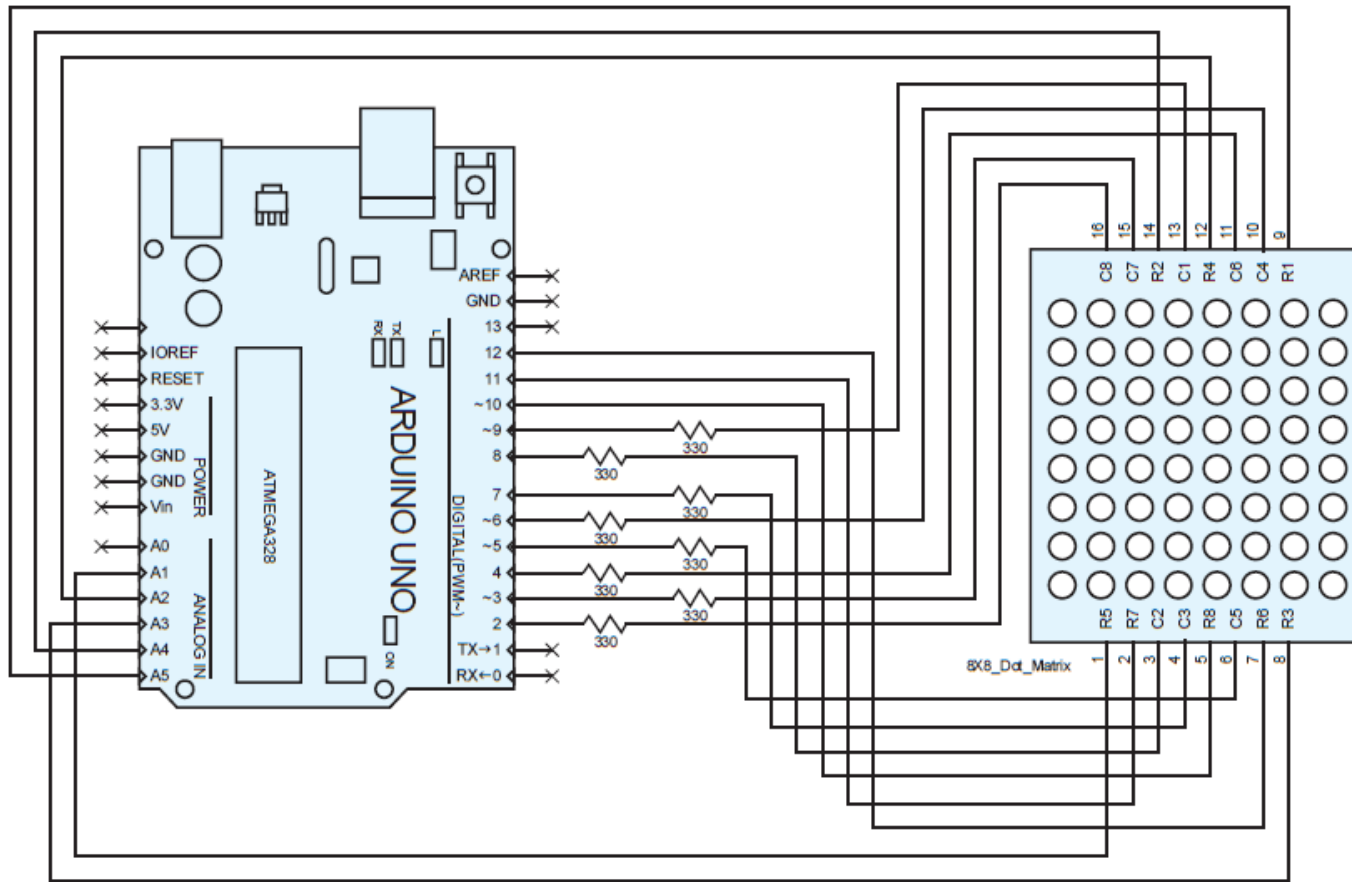
## Dot matrix 제어 (1/3)

**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 220 $\Omega$ 저항을 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



C1	C2	C3	C4	C5	C6	C7	C8
9	8	7	6	5	4	3	2
R1	R2	R3	R4	R5	R6	R7	R8
A5	A4	A3	A2	A1	12	11	10

## 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.

### 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.

실행 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

```

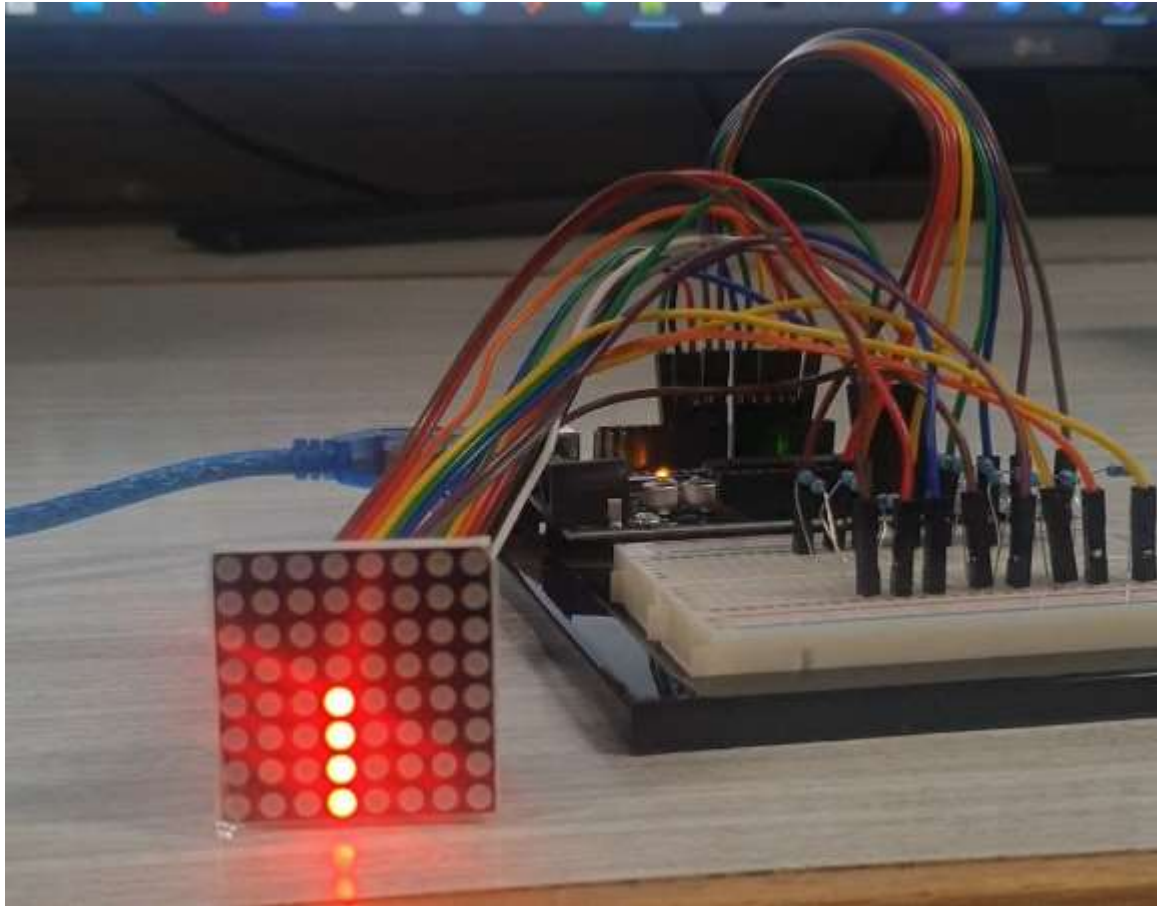
5
6 const int colPins[] = { 2, 3, 4, 5, 6, 7, 8, 9};
7 //          C8, C7, C6, C5, C4, C3, C2, C1
8 const int rowPins[] = { 10,11,12,15,16,17,18,19};
9 //          R8, R7, R6, R5, R4, R3, R2, R1
10
11 void setup() {
12   for (int i = 0; i < 8; i++)
13   {
14     // 행을 출력으로 설정한다
15     pinMode(colPins[i],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

```

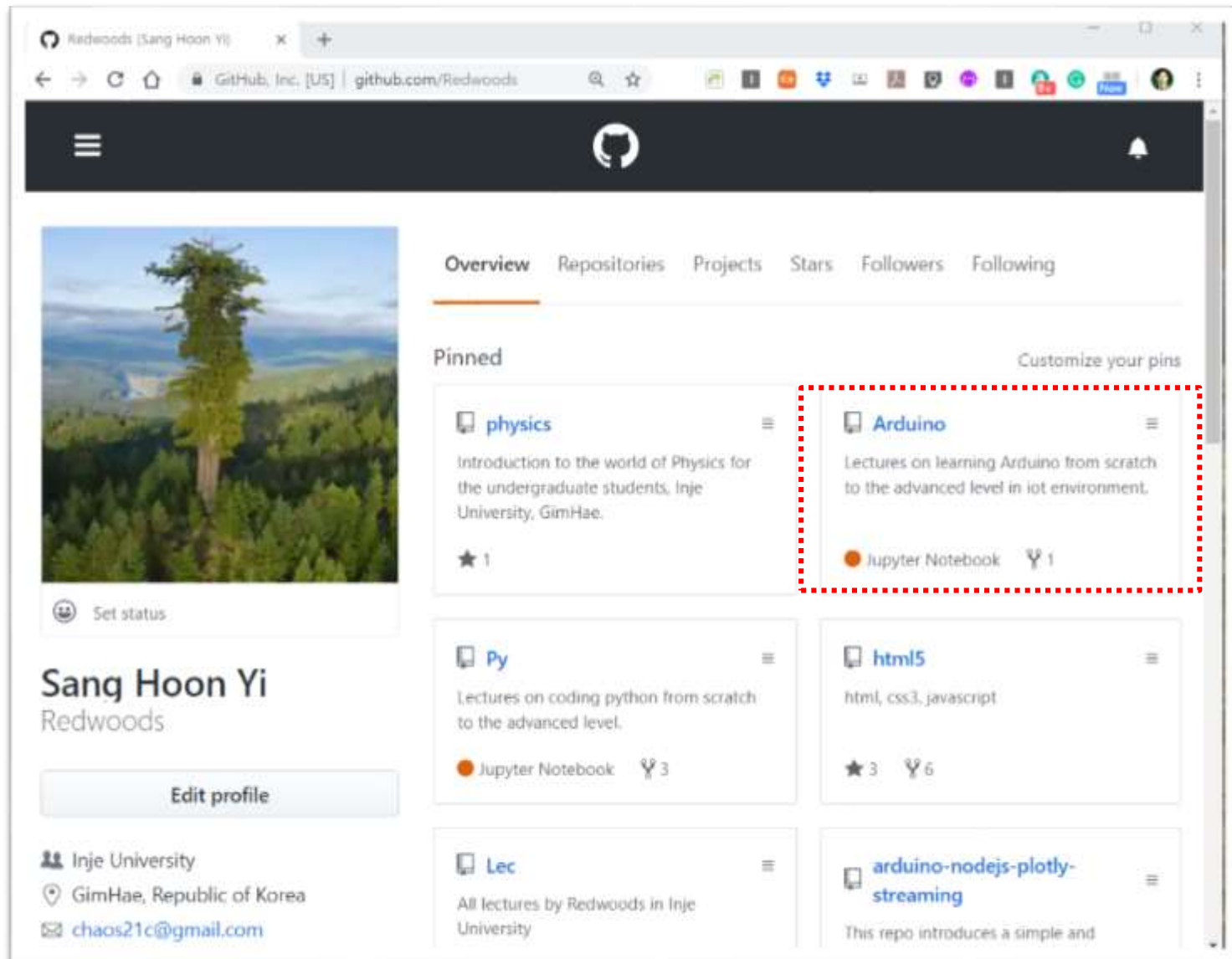


[ARnn\\_dm.png](#)

로 저장해서 제출.

## ● References & good sites

- ✓ <http://www.nodejs.org/ko> Node.js
- ✓ <http://www.arduino.cc> Arduino Homepage
- ✓ <http://www.w3schools.com> By w3schools
- ✓ <http://www.github.com> GitHub
- ✓ <http://www.google.com> Googling



Redwoods (Sang Hoon Yi)

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

Overview Repositories Projects Stars Followers Following

Pinned Customize your pins

**physics**  
Introduction to the world of Physics for the undergraduate students, Inje University, GimHae.  
★ 1

**Arduino**  
Lectures on learning Arduino from scratch to the advanced level in iot environment.  
Jupyter Notebook 🍴 1

**Py**  
Lectures on coding python from scratch to the advanced level.  
Jupyter Notebook 🍴 3

**html5**  
html, css3, javascript  
★ 3 🍴 6

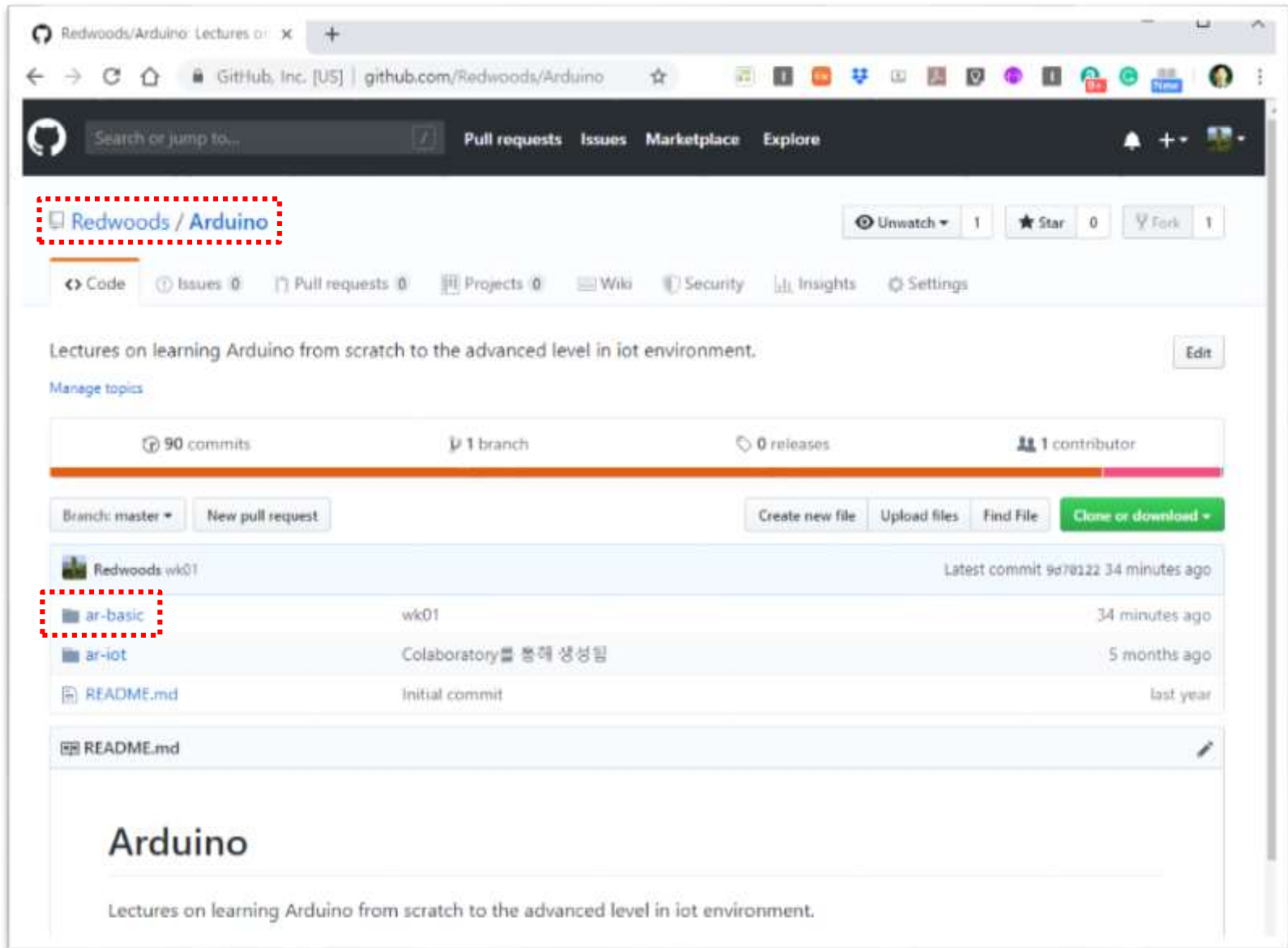
**Lec**  
All lectures by Redwoods in Inje University

**arduino-nodejs-plotly-streaming**  
This repo introduces a simple and

**Sang Hoon Yi**  
Redwoods

Edit profile

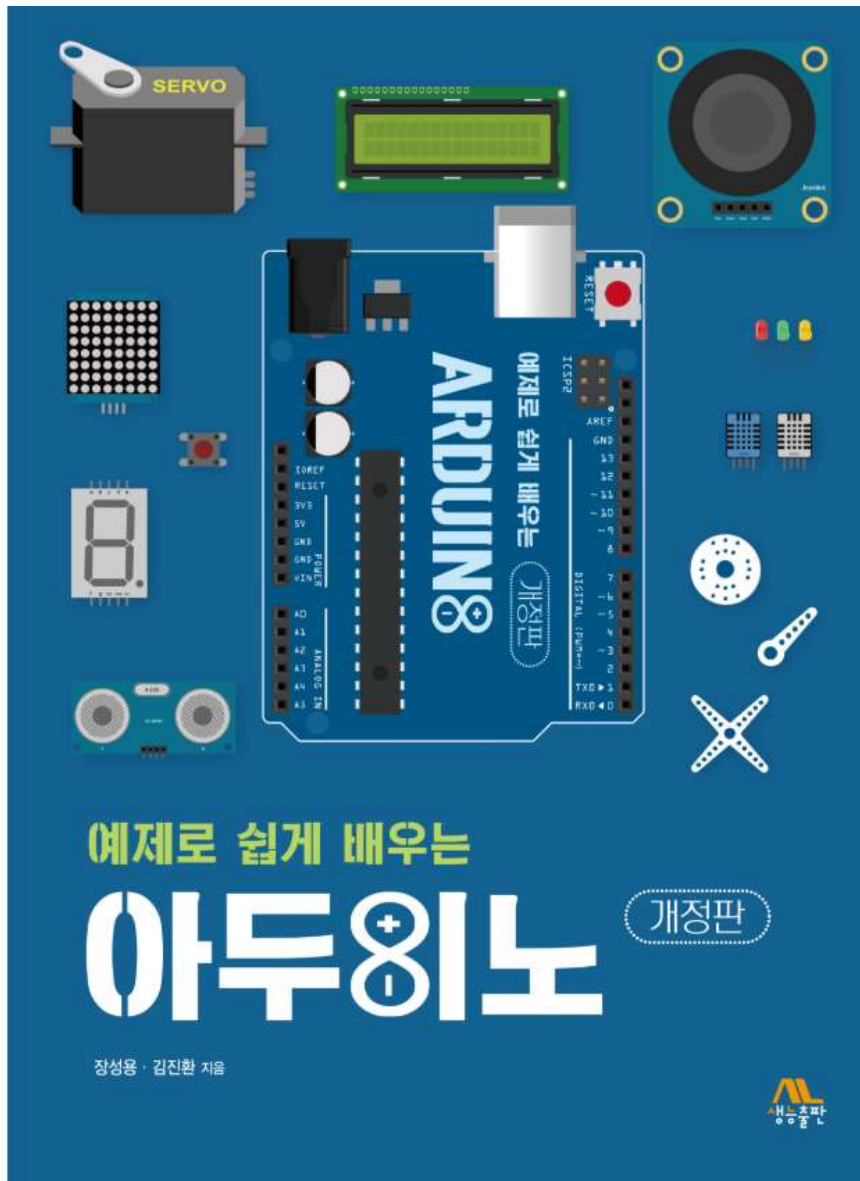
Inje University  
GimHae, Republic of Korea  
chaos21c@gmail.com



The screenshot shows the GitHub repository page for **Redwoods/Arduino**. The repository is described as "Lectures on learning Arduino from scratch to the advanced level in iot environment." It has 90 commits, 1 branch, 0 releases, and 1 contributor. The repository is currently on the **master** branch. The file list shows the following files and their commit history:

File	Commit	Time
ar-basic	wk01	34 minutes ago
ar-iot	Colaboratory를 통해 생성됨	5 months ago
README.md	Initial commit	last year

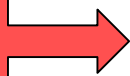
The **ar-basic** file is highlighted with a red dashed box. Below the file list, the **README.md** content is visible, starting with the title **Arduino** and the description "Lectures on learning Arduino from scratch to the advanced level in iot environment."



# 아두이노 키트(Kit)



web



<https://www.devicemart.co.kr/goods/view?no=12170416>



# 아두이노 키트(Kit) : Part-1

## 아두이노 레벨업키트(골드) 구성품

						
아두이노 UNO	USB 케이블	830핀 브레드보드	미니 브레드보드	점퍼와이어 세트	듀폰케이블 M/F	듀폰케이블 M/M
						
저항 220Ω	저항 1KΩ	저항 10KΩ	가변저항 10KΩ	빨강 LED	녹색 LED	파랑 LED
						
노랑 LED	RGB LED (CA)	RGB LED 모듈	1digit FND (CA)	4digit FND (CA)	8x8 도트 매트릭스	택트 스위치
						
택트 스위치 캡	볼 스위치	리드 스위치 센서	4x4키 매트릭스	5V 릴레이 모듈	조이스틱 모듈	수위 센서
						
온도센서 LM35	터미스터	온습도센서	CDS 조도센서	불꽃감지센서	적외선 수신기	IR 리모컨



# 아두이노 키트(Kit) : Part-2

						
TCRT5000 적외선 센서	인체감지센서 모듈	사운드센서	능동부저	수동부저	초음파센서	I2C 1602 LCD 모듈
						
서보모터	스텝모터	스텝모터드라이버	RFID 수신 모듈	RFID 카드	RFID 태그	DS1302 RTC 모듈
						
1N4001 다이오드	2N2222 트랜지스터	74HC595	1x40 핀헤더	9V 배터리 스냅	아크릴 고정판	

■ 아두이노 UNO × 1	■ USB 케이블 × 1	■ 830핀브레드보드 × 1	■ 미니 브레드보드 × 1	■ 점퍼와이어세트 × 1
■ 듀폰케이블 × 80 (M/F,M/M)	■ 저항 × 30	■ 가변저항 × 1	■ LED × 20	■ RGB LED × 1
■ RGB LED 모듈 × 1	■ 1digit FND(CA) × 1	■ 4digit FND(CA) × 1	■ 8×8도트 매트릭스 × 1	■ 탭스위치 × 5
■ 탭스위치 캡 × 5	■ 볼스위치 × 1	■ 리드 스위치 센서 × 1	■ 4×4 키 매트릭스 × 1	■ 5V 릴레이 모듈 × 1
■ 조이스틱 모듈 × 1	■ 수위 센서 × 1	■ 온도센서 LM35 × 1	■ 써미스터 × 1	■ 온습도센서 × 1
■ CdS 조도센서 × 1	■ 불꽃감지센서 × 1	■ 적외선 수신기 × 1	■ IR 리모컨 × 1	■ TCRT5000 적외선 센서 × 1
■ 인체감지센서 모듈 × 1	■ 사운드센서 × 1	■ 능동부저 × 1	■ 수동부저 × 1	■ 초음파센서 × 1
■ I2C 1602 LCD 모듈 × 1	■ 서보모터 × 1	■ 스텝모터 × 1	■ 스텝모터드라이버 × 1	■ RFID 수신 모듈 × 1
■ RFID 카드 × 1	■ RFID 태그 × 1	■ DS1302 RTC 모듈 × 1	■ 1N4001 다이오드 × 1	■ 2N2222 트랜지스터 × 1
■ 74HC595 × 1	■ 1x40 핀헤더 × 1	■ 9V 배터리 스냅 × 1	■ 아크릴 고정판 × 1	