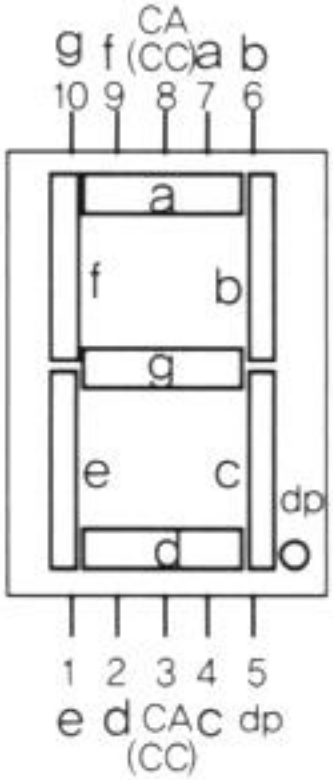
7-세그먼트

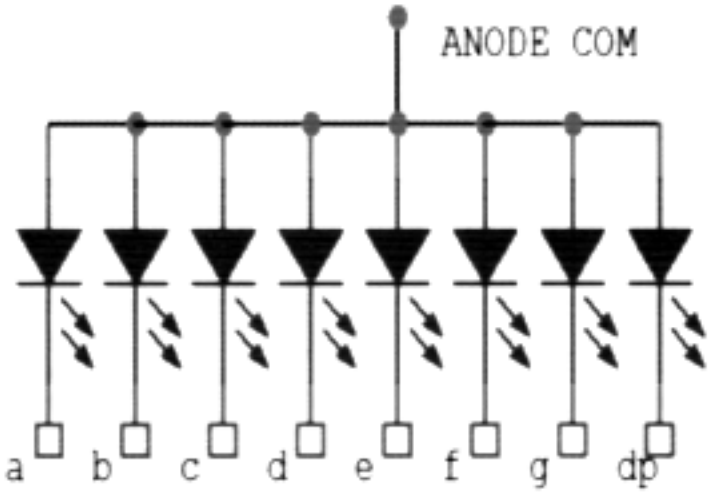
7-세그먼트

(a) 7_Segment 외형

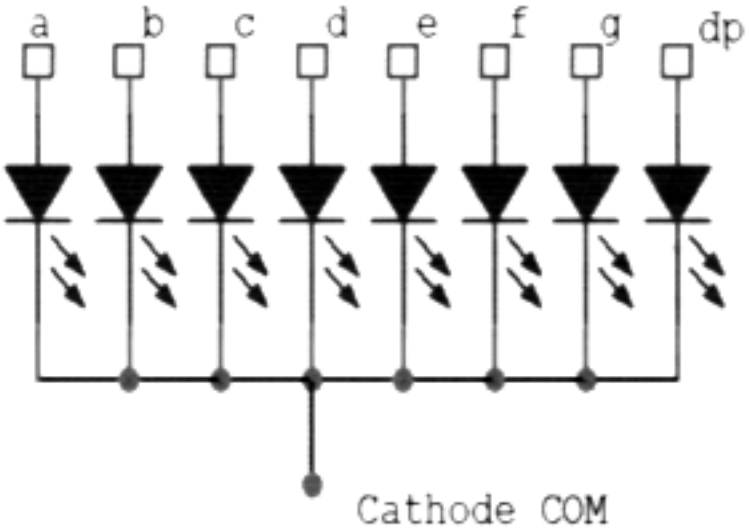


(b) LED 구성

7-세그먼트



(c) Common Anode 형



(d) Common Cathode 형

7-세그먼트

❖ 10진수(0~9)와 16진수(0~F)에 대한 패턴

표시문자	g	f	e	d	c	b	a
0	OFF	ON	ON	ON	ON	ON	ON
1	OFF	OFF	OFF	OFF	ON	ON	OFF
2	ON	OFF	ON	ON	OFF	ON	ON
3	ON	OFF	OFF	ON	ON	ON	ON
4	ON	ON	OFF	OFF	ON	ON	OFF
5	ON	ON	OFF	ON	ON	OFF	ON
6	ON	ON	ON	ON	ON	OFF	ON
7	OFF	OFF	OFF	OFF	ON	ON	ON

7-세그먼트

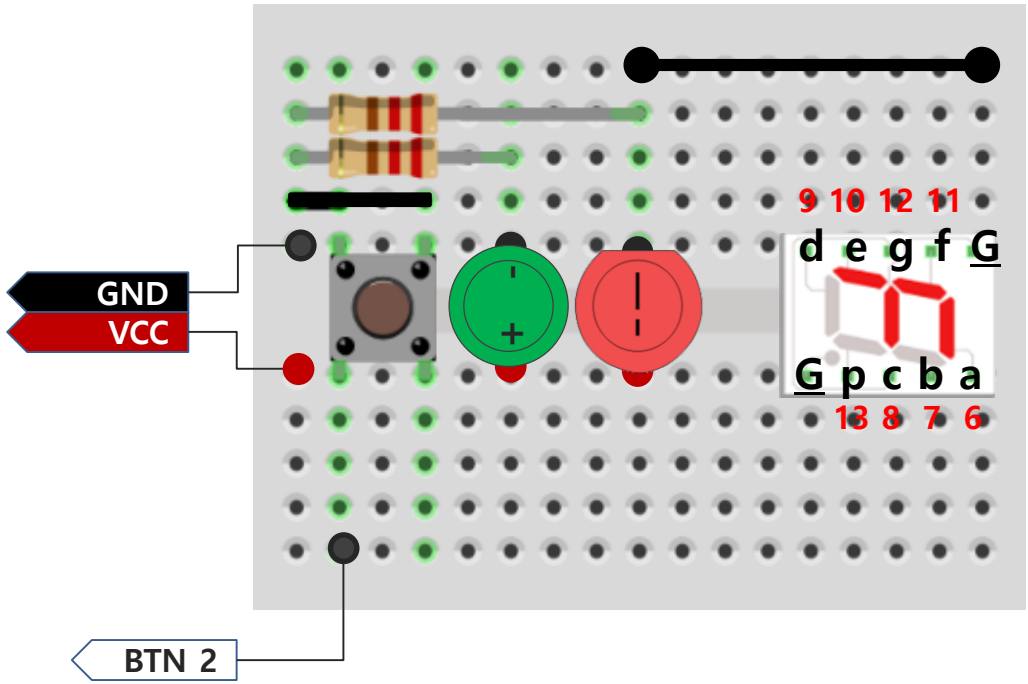
❖ 10진수(0~9)와 16진수(0~F)에 대한 패턴

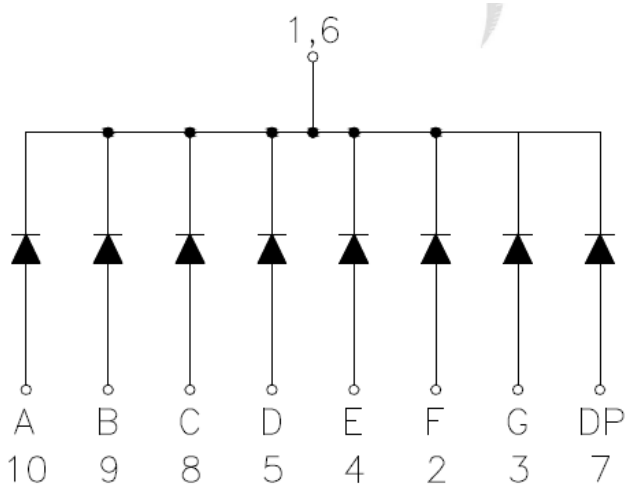
표시문자	g	f	e	d	c	b	a
8	ON	ON	ON	ON	ON	ON	ON
9	ON	ON	OFF	ON	ON	ON	ON
8	ON	ON	ON	OFF	ON	ON	ON
6	ON	ON	ON	ON	ON	OFF	OFF
C	OFF	ON	ON	ON	OFF	OFF	ON
d	ON	OFF	ON	ON	ON	ON	OFF
E	ON	ON	ON	ON	OFF	OFF	ON
F	ON	ON	ON	OFF	OFF	OFF	ON

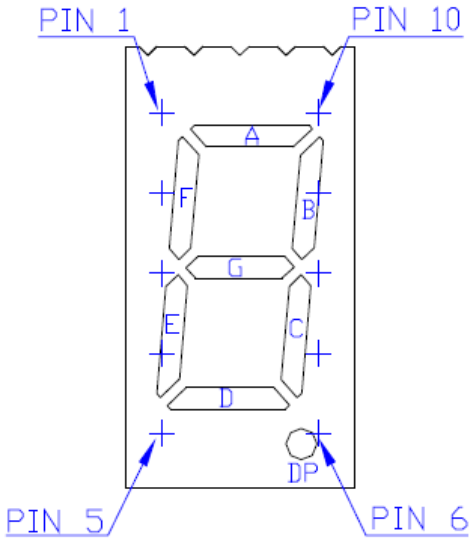
7-세그먼트

공통 캐소드(Cathode)									
숫자	a	b	c	d	e	f	g	dp	16진수
0	1	1	1	1	1	1	0	0	0xFC
1	0	1	1	0	0	0	0	0	0x60
2	1	1	0	1	1	0	1	0	0xDA
3	1	1	1	1	0	0	1	0	0xF2
4	0	1	1	0	0	1	1	0	0x66
5	1	0	1	1	0	1	1	0	0xB6
6	1	0	1	1	1	1	1	0	0xBE
7	1	1	1	0	0	1	0	0	0xE4
8	1	1	1	1	1	1	1	0	0xFE
9	1	1	1	1	0	1	1	0	0xF6

❖ 회로도


GND
VCC
BTN 2
9 10 12 11
d e g f G
G p c b a
13 8 7 6


1,6
A B C D E F G DP
10 9 8 5 4 2 3 7


PIN 1
PIN 10
A
F
B
G
E
C
D
DP
PIN 5
PIN 6

실습1: 7-세그먼트 16진수(0~F) 순차 표시하기

❖ segment1.ino

```
// A,B,C,D,E,F,G, DOT 연결 핀
const int segment_pin[8] ={6, 7, 8, 9, 10, 11, 12, 13};

// 0~F 표시 패턴 {a, b, c, d, e, f, g, dot}
const byte segment_pat[16][8] = {
  {1, 1, 1, 1, 1, 1, 0, 0},   // 0
  {0, 1, 1, 0, 0, 0, 0, 0},   // 1
  {1, 1, 0, 1, 1, 0, 1, 0},   // 2
  {1, 1, 1, 1, 0, 0, 1, 0},   // 3
  {0, 1, 1, 0, 0, 1, 1, 0},   // 4
  {1, 0, 1, 1, 0, 1, 1, 0},   // 5
  {1, 0, 1, 1, 1, 1, 1, 0},   // 6
  {1, 1, 1, 0, 0, 0, 0, 0},   // 7
  {1, 1, 1, 1, 1, 1, 1, 0},   // 8
  {1, 1, 1, 0, 0, 1, 1, 0},   // 9
  {1, 1, 1, 0, 1, 1, 1, 0},   // A
  {0, 0, 1, 1, 1, 1, 1, 0},   // b
  {1, 0, 0, 1, 1, 1, 0, 0},   // C
  {0, 1, 1, 1, 1, 0, 1, 0},   // D
  {1, 0, 0, 1, 1, 1, 1, 0},   // E
  {1, 0, 0, 0, 1, 1, 1, 0}    // F
};
```

실습1: 7-세그먼트 16진수(0~F) 순차 표시하기

❖ segment1.ino

```
int   dsp_no = 0;         // 표시 번호

void segment_dsp() {
  int n;

  for(n = 0;n < 8;n++)
    digitalWrite(segment_pin[n], segment_pat[dsp_no][n]);
}

void setup() {
  for(int n = 0;n < 8;n++) {
    pinMode(segment_pin[n], OUTPUT);
  }
}

void loop() {
  segment_dsp();
  dsp_no++;   // 표시 패턴 번호 갱신
  if(dsp_no == 16)
    dsp_no = 0;    // 16진 'F' 다음 처음부터 다시 표시
  delay(1000);
}
```

실습2: Segment7 클래스

❖ ex02/Segment7.h

```
#pragma once

#include <Arduino.h>

class Segment7 {
protected:

public:
    Segment7();
    void display(int num);

};
```

실습2: Segment7 클래스

❖ ex02/Segment7.cpp

```
#include "Segment7.h"
// 0~F 표시 패턴 {a, b, c, d, e, f, g, dot}
const byte segment_pat[16][8] = {
  {1, 1, 1, 1, 1, 1, 0, 0},   // 0
  {0, 1, 1, 0, 0, 0, 0, 0},   // 1
  {1, 1, 0, 1, 1, 0, 1, 0},   // 2
  {1, 1, 1, 1, 0, 0, 1, 0},   // 3
  {0, 1, 1, 0, 0, 1, 1, 0},   // 4
  {1, 0, 1, 1, 0, 1, 1, 0},   // 5
  {1, 0, 1, 1, 1, 1, 1, 0},   // 6
  {1, 1, 1, 0, 0, 0, 0, 0},   // 7
  {1, 1, 1, 1, 1, 1, 1, 0},   // 8
  {1, 1, 1, 0, 0, 1, 1, 0},   // 9
  {1, 1, 1, 0, 1, 1, 1, 0},   // A
  {0, 0, 1, 1, 1, 1, 1, 0},   // b
  {1, 0, 0, 1, 1, 1, 0, 0},   // C
  {0, 1, 1, 1, 1, 0, 1, 0},   // D
  {1, 0, 0, 1, 1, 1, 1, 0},   // E
  {1, 0, 0, 0, 1, 1, 1, 0}    // F
};

const int segment_pin[8] ={6, 7, 8, 9, 10, 11, 12, 13};
```

실습2: Segment7 클래스

❖ ex02/Segment7.cpp

```
Segment7::Segment7() {
    for(int n = 0;n < 8;n++) {
        pinMode(segment_pin[n], OUTPUT);
    }
}

void Segment7::display(int num) {
    for(int n = 0;n < 8;n++) {
        digitalWrite(segment_pin[n], segment_pat[num][n]);
    }
}
```

실습2: Segment7 클래스

❖ ex02/app.ino

```
#include "Segment7.h"

Segment7 fnd;

int dsp_no = 0;
void setup() {
}

void loop() {
    fnd.display(dsp_no);
    dsp_no = (++dsp_no) % 16;
    delay(1000);
}
```

실습3: 버튼으로 Segment 값 증가시키기

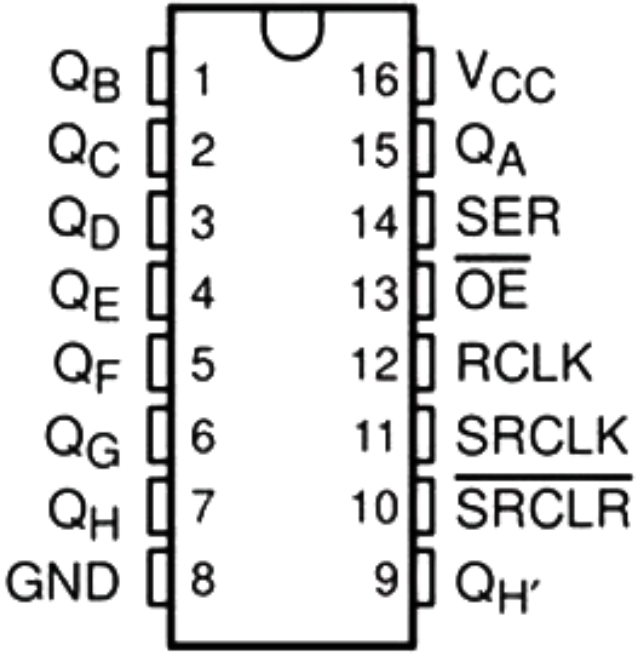
❖ ex03/app.ino

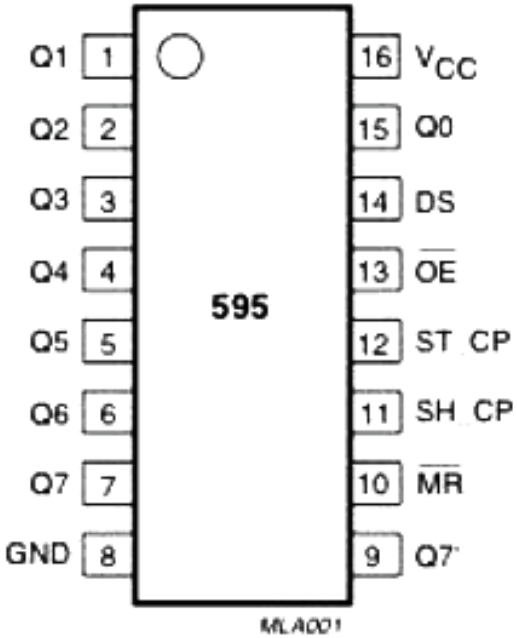
```
#include <Button.h>
#include <Segment7.h>

int   dsp_no = 0;         // 표시 번호

Button btn(2);
Segment7 fnd;

void update_segment() {
  if(!btn.debounce()) return;

  dsp_no = (dsp_no+1) % 16;
  fnd.display(dsp_no);
}

void setup() {
  btn.attachInterrupt(update_segment, FALLING);
  fnd.display(dsp_no);
}

void loop() {
}
```

74HC565 쉬프트 레지스터

❖ **74HC565**

- 쉬프트 레지스터(Shift register)
- 3개의 연결로 8개의 신호 전송





DATA SERIAL

OUTPUT ENABLE/

STORAGE(LATCH) CLOCK

SHIFT CLOCK

MASTER RESET/

74HC565 쉬프트 레지스터

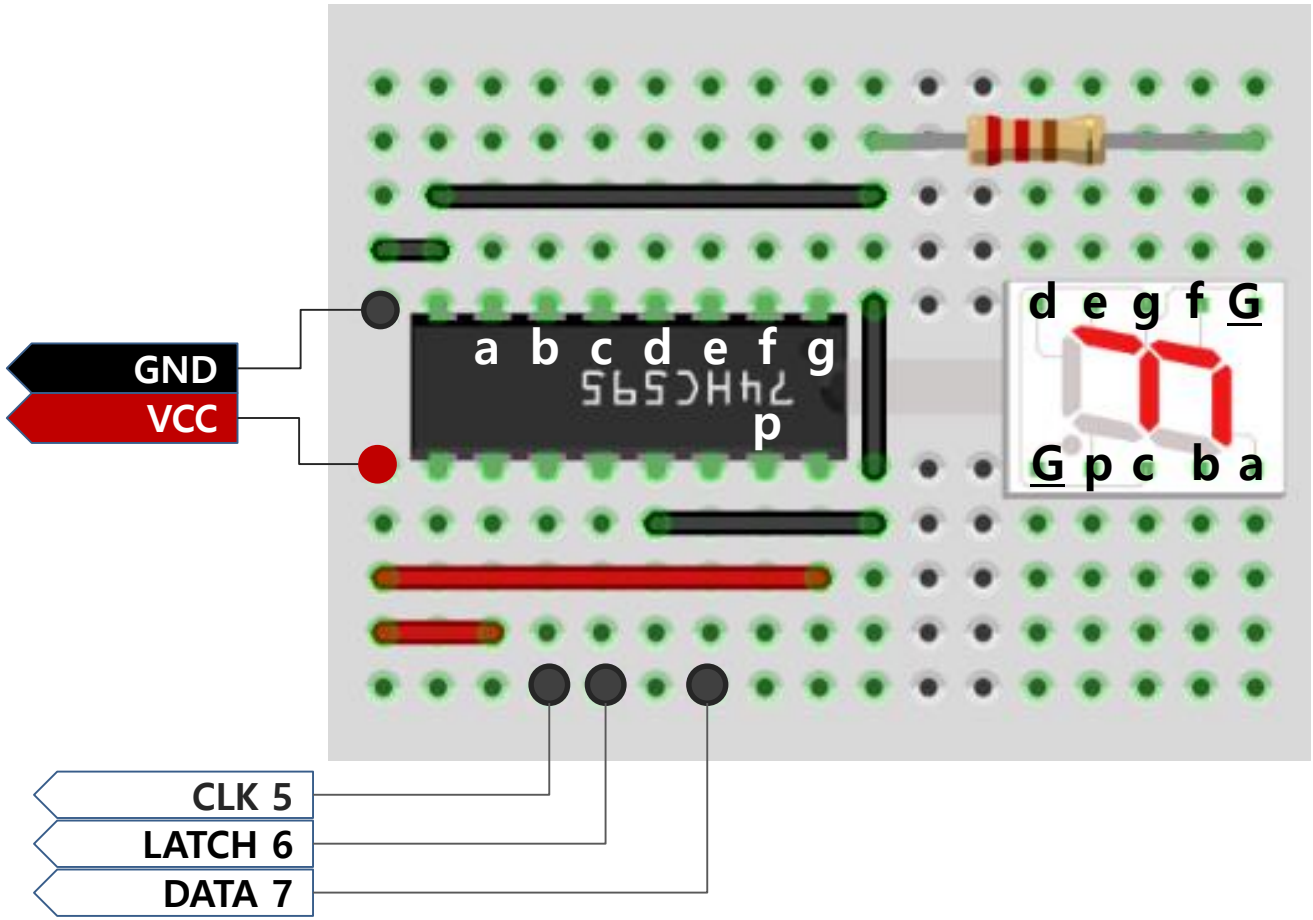
❖ 74HC565

- 작업 순서
  - STORAGE(LATCH) CLOCK에 LOW 인가
  - shiftOut()으로 데이터 인가
  - STORAGE(LATCH) CLOCK에 HIGH 인가

- shiftOut(data pin, clock pin, MSBFIRST, data)

74HC565 쉬프트 레지스터

❖ 74HC565 연결



74HC565 쉬프트 레지스터

❖ Segment7.h

```
    :

class ShiftSegment {
protected:
  int data_pin;
  int latch_pin;
  int shift_pin;

public:
  ShiftSegment(int shift_pin, int latch_pin, int data_pin);
  void display(int num);
};
```

74HC565 쉬프트 레지스터

❖ Segment7.cpp

```
    :

byte digits[] = {
  B11111100,  // 0
  B01100000,  // 1
  B11011010,  // 2
  B11110010,  // 3
  B01100110,  // 4
  B10110110,  // 5
  B10111110,  // 6
  B11100000,  // 7
  B11111110,  // 8
  B11100110,  // 9
  B11101110,  // A
  B00111110,  // b
  B10011100,  // C
  B01111010,  // D
  B10011110,  // E
  B10001110   // F
};
```

74HC565 쉬프트 레지스터

❖ Segment7.cpp

```
ShiftSegment::ShiftSegment(int shift_pin, int latch_pin, int data_pin)
  : shift_pin(shift_pin), latch_pin(latch_pin), data_pin(data_pin) {
  pinMode(shift_pin, OUTPUT);
  pinMode(latch_pin, OUTPUT);
  pinMode(data_pin, OUTPUT);
}

void ShiftSegment::display(int num) {
  digitalWrite(latch_pin, LOW);
  shiftOut(data_pin, shift_pin, MSBFIRST, digits[num]);
  digitalWrite(latch_pin, HIGH);
}
```

74HC565 쉬프트 레지스터

❖ ex04/app.ino

```
#include <Button.h>
#include "Segment.h"

int   dsp_no = 0;         // 표시 번호
Button btn(2);
ShiftSegment segment(5, 6, 7); // shift, latch, data

void update_segment() {
  if(!btn.debounce()) return;

  dsp_no = (dsp_no+1) % 16;
  segment.display(dsp_no);
}

void setup() {
  btn.attachInterrupt(update_segment);
  segment.display(dsp_no);
}

void loop() {
}
```