6-DIGIT FND를 이용한 문자 출력

- **기초 학습**
  - 연산자, 비트 함수
- **FND를 이용한 문자 출력**
  - 개 요
  - 예 제
- **응용 실습**

엣지아이랩

Basic learning

기초 학습

포인터 접근 연산자

- * (dereference operator)
  - 참조 연산자로 해당 변수의 저장된 값을 참조

- & (reference operator)
  - 주소 연산자로 해당 변수의 주소 값을 참조

```
int a, b=20, *c, *d;
c = &a;          // c에 a의 주소 값을 저장
*c = 30;         // c에 저장된 값의 주소에 30을 저장(a = 30;과 동일)
d = &b;          // d에 b의 주소 값을 저장
```

비트 연산자

- & (bitwise and)
  - 왼쪽과 오른쪽 값을 비트별 and 연산하여 반환
- | (bitwise or)
  - 왼쪽과 오른쪽 값을 비트별 or 연산하여 반환
- ^ (bitwise xor)
  - 왼쪽과 오른쪽 값을 비트별 xor 연산하여 반환
- a와 b의 and, or, xor 결과

입력 값		출력 값		
a	b	and	or	xor
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

비트 연산자

- ~ (bitwise not)
  - 비트별 값을 0이면 1로, 1이면 0으로 변환하여 반환
  - ex) a의 값이 0x5A 일 때, ~a 연산한 값은 0xA5

- << (bit shift left)
  - 각각의 비트를 좌측으로 이동
  - ex) a의 값이 0x5A 일 때, a << 1 연산한 값은 0xB4

- >> (bit shift right)
  - 각각의 비트를 우측으로 이동
  - ex) a의 값이 0x5A 일 때, a >> 2 연산한 값은 0x16

복합 연산자

- ++ (increment)
  - 변수의 값이 1 증가
- -- (decrement)
  - 변수의 값이 1 감소
- += (compound addition)
  - 좌측 변수와 오른쪽의 값을 더하여 좌측변수에 저장
- -= (compound subtraction)
  - 좌측 변수에서 오른쪽의 값을 뺀 후 좌측변수에 저장
- 예시

```
a++; -> a += 1; -> a = a + 1;    -> a 값을 1 증가하여 a에 저장
a--; -> a -= 1; -> a = a - 1;    -> a 값을 1 감소하여 a에 저장
```

복합 연산자

- *= (compound multiplication)
  - 좌측의 변수와 오른쪽의 값을 곱하여 좌측변수에 저장

- /= (compound division)
  - 좌측의 변수에서 오른쪽의 값을 나눈 후 좌측변수에 저장

- 예시

```
a *= b; -> a = a * b; -> a와 b를 곱한 값을 a에 저장
a /= b; -> a = a / b; -> a에서 b를 나눈 값을 a에 저장
```

복합 연산자

- &=(compound bitwise and)
  - 왼쪽의 변수와 오른쪽의 값을 & 연산하여 좌측변수에 저장
- |=(compound bitwise or)
  - 왼쪽의 변수와 오른쪽의 값을 | 연산하여 좌측변수에 저장
- 예시

```
a &= b; -> a = a & b; -> a와 b를 & 연산한 값을 a에 저장
a |= b; -> a = a | b; -> a와 b를 | 연산한 값을 a에 저장
```

데이터 타입

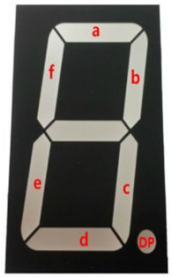
- uint8_t, uint16_t, uint32_t
  - 플랫폼마다 데이터타입의 크기가 다르기 때문에 이식성(Portability)을 지원하기 위한 데이터 타입
  - "_t"은 Primitive System Data type(초기 시스템 데이터 형식)이라고 함
  - 정수 비트 길이와 바이트가 문제가 될 때, char 또는 short, int, long 같은 데이터 형식을 사용하지 않고 고정 길이 데이터 형식을 이용
  - 32bit와 64bit의 데이터 길이의 변화에 따른 버그, 에러를 막기 위해서 정의해서 사용하는 것으로 리눅스나 임베디드 계열에서 많이 사용

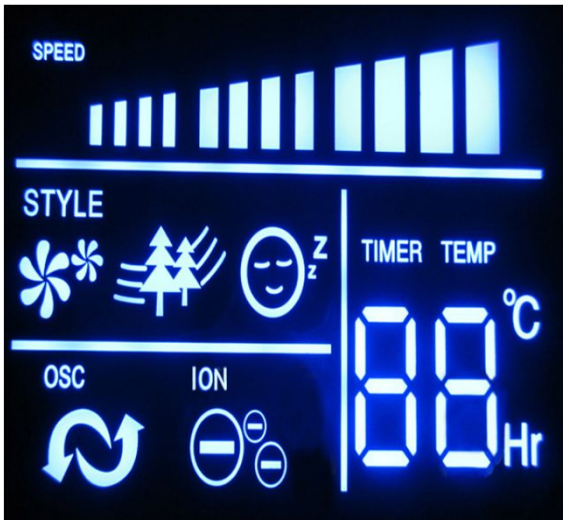
Specifier	Common Equivalent	Signing	Bits	Bytes	Minimum Value	Maximum Value
int8_t	signed char	signed	8	1	-128	127
uint8_t	unsigned char	unsigned	8	1	0	255
int16_t	short	signed	16	2	-32,768	32,767
uint16_t	unsigned short	unsigned	16	2	0	65,535
int32_t	long	signed	32	4	-2,147,483,648	2,147,483,647
uint32_t	unsigned long	unsigned	32	4	0	4,294,967,295
unt64_t	long long	signed	64	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
uint64_t	unsigned long long	unsigned	64	8	0	18,446,744,073,709,554,615

FND를 이용한 문자 출력

개요

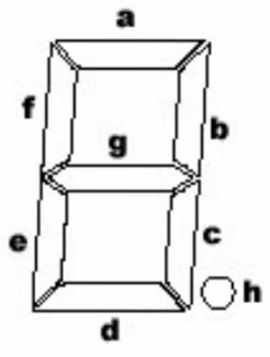
- 목적
  - 디지털 출력 이해
  - 'High', 'Low'의 출력을 통해 숫자나 문자를 표시
  - FND(Flexible Numeric Display)는 가변 숫자 표시기로 LED8개가 내장되어있는 출력장치
  - 카운터, 디지털 시계 등을 구현

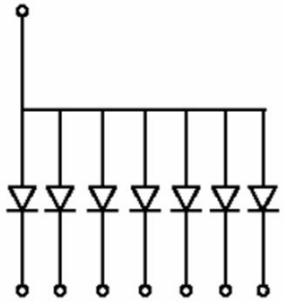




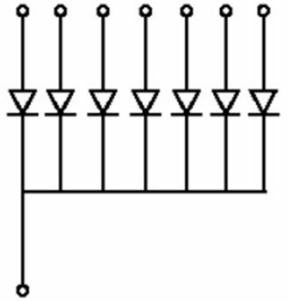
개요

- 관련이론
  - 숫자를 표시하기 위해 만들어지 LED의 조합으로, 이를 응용하여 생산 현장에서 생산실적등의 수치데이터, 현황판으로 많이 사용
  - LED의 각 핀에 해당하는 a~g의 기호가 붙어있으며, 각각을 세그먼트라고 함
  - a~g 해당하는 핀에 HIGH와 LOW를 인가하여 LED를 점등함으로써 숫자를 표현
  - 에노드가 공통인 세븐 세그먼트를 Anode Common이라고 하고, 캐소드가 공통인 세븐 세그먼트를 Cathode Common 이라고 함





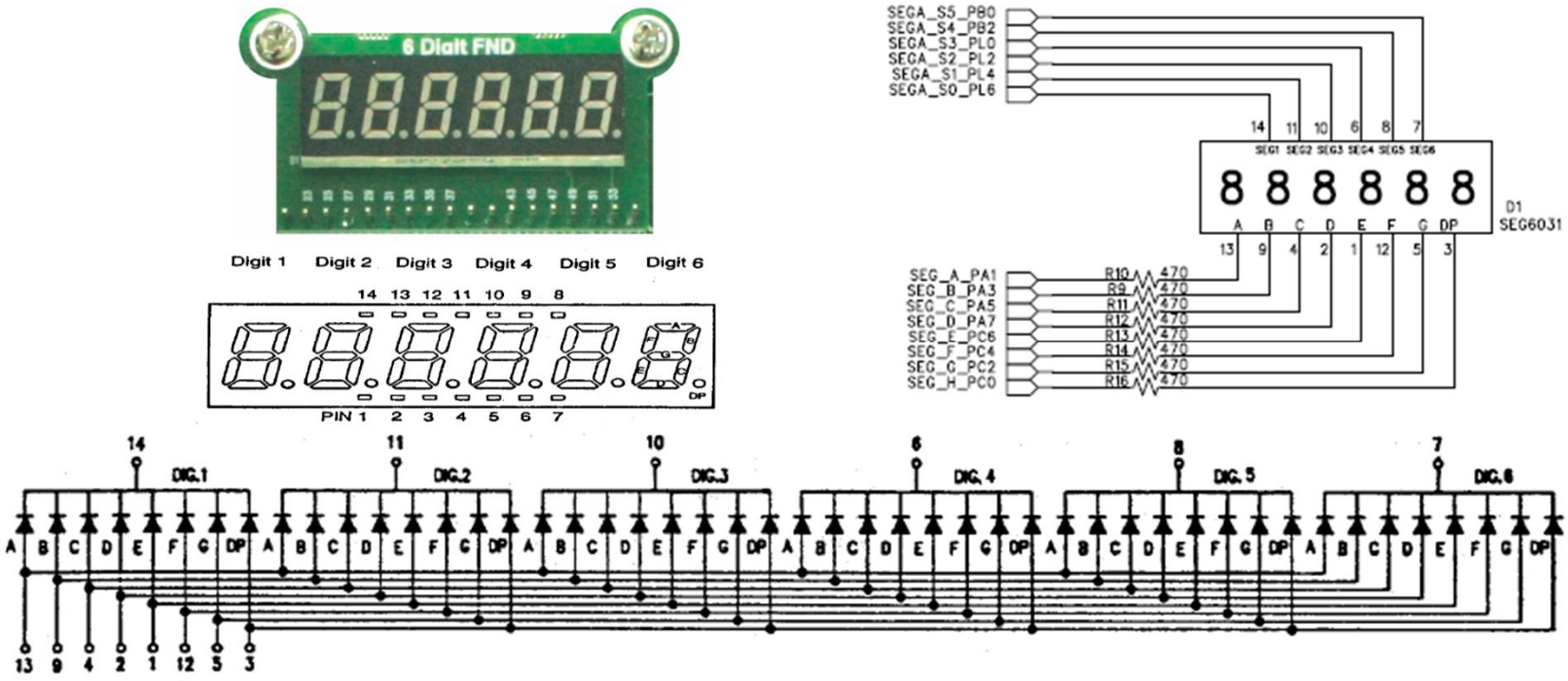
Anode Common



Cathode Common

개요

- 하드웨어 설명
  - 데이터시트 참조
  - 해당 숫자 표시기 공통 단자에 LOW 신호를 주고 표시하고자 하는 숫자에 맞게 a, b, c, d, e, f, g 에 LOW/HIGH 신호를 할당
  - 짧은 시간 간격으로 순차적으로 6개의 숫자 표시기에 원하는 숫자를 표시하면 동시에 표시되는 것처럼 보이게 됨



Cathode Common

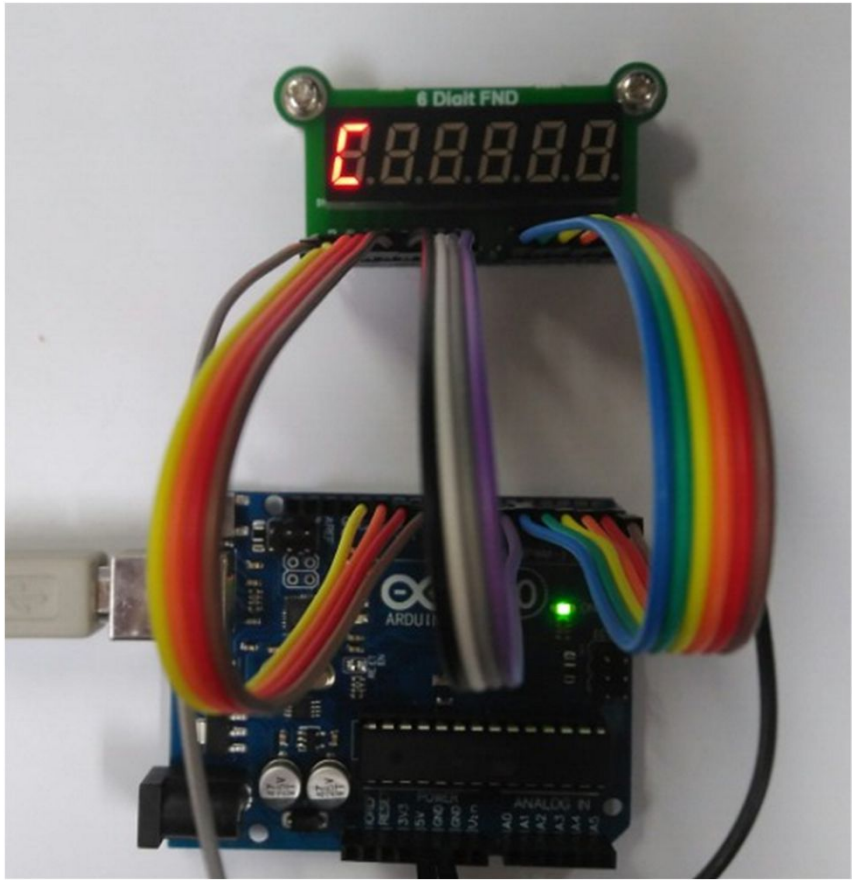
개요

- 7-Segment에 16진수 표시방법

16진수	7세그먼트의 비트 값								데이터 값		
	H	G	F	E	D	C	B	A	2진수	10진수	16진수
0	0	0	1	1	1	1	1	1	B00111111	63	0x3F
1	0	0	0	0	0	1	1	0	B00000110	6	0x06
2	0	1	0	1	1	0	1	1	B01011011	91	0x5B
3	0	1	0	0	1	1	1	1	B01001111	79	0x4F
4	0	1	1	0	0	1	1	0	B01100110	102	0x66
5	0	1	1	0	1	1	0	1	B01101101	109	0x6D
6	0	1	1	1	1	1	0	1	B01111101	125	0x7D
7	0	0	1	0	0	1	1	1	B00100111	39	0x27
8	0	1	1	1	1	1	1	1	B01111111	127	0x7F
9	0	1	1	0	1	1	1	1	B01101111	111	0x6F
A	0	1	1	1	0	1	1	1	B01110111	119	0x77
B	0	1	1	1	1	1	0	0	B01111100	124	0x7C
C	0	0	1	1	1	0	0	1	B00111001	57	0x39
D	0	1	0	1	1	1	1	1	B01011110	94	0x5E
E	0	1	1	1	1	0	0	1	B01111001	121	0x79
F	0	1	1	1	0	0	0	1	B01110001	113	0x71

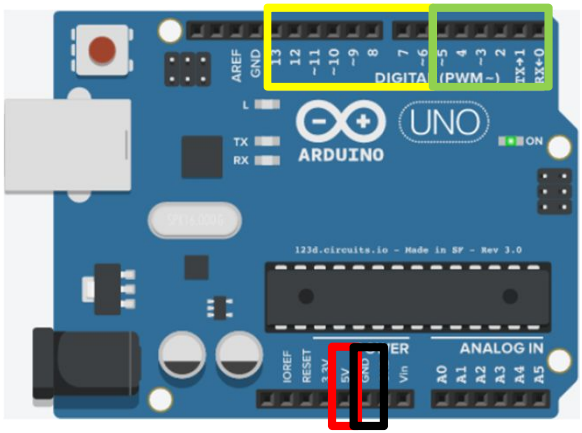
예제

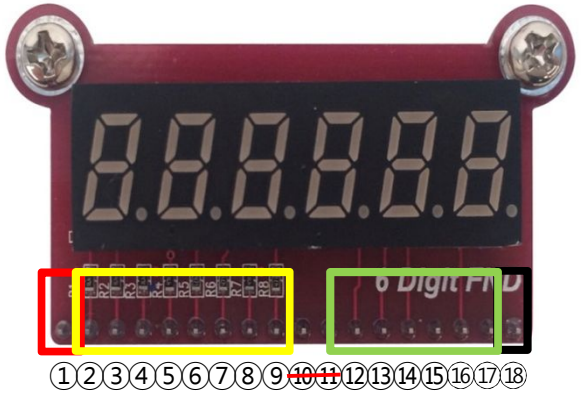
- 예제
  - 첫번째 세그먼트에서 0~F까지 1초마다 차례대로 하시오.



결선

- Uno 보드와 6-Digit FND 모듈 결선





Name	Digit FND Pin Number	Arduino Pin Number	Name	Digit FND Pin Number	Arduino Pin Number
VCC	①	5V	-	⑩	-
A	②	13	-	⑪	-
B	③	12	S0	⑫	5
C	④	11	S1	⑬	4
D	⑤	10	S2	⑭	3
E	⑥	9	S3	⑮	2
F	⑦	8	S4	⑯	1
G	⑧	7	S5	⑰	0
H	⑨	6	GND	⑱	GND

예제

- 프로그램 설명

  - 사용할 핀 정의

aFndData에 0~F까지 표시 값을 저장, aFndSelData에 S0~S5까지의 선택 값 저장

```
int aPinFndData[8] = {13, 12, 11, 10, 9, 8, 7, 6};
int aPinFndSel[6] = {5, 4, 3, 2, 1, 0};
uint8_t aFndData[16] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x27,
                        0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71};
uint8_t aFndSelData[6] = {0x3E, 0x3D, 0x3B, 0x37, 0x2F, 0x1F};
```

  - 초기화 구문

aPinFndData, aPinFndSel 핀을 출력 설정

```
for(i=0; i<8; i++) {
  pinMode(aPinFndData[i], OUTPUT);
}
for(i=0; i<6; i++) {
  pinMode(aPinFndSel[i], OUTPUT);
}
```

예제

- 프로그램 설명

  - loop 구문

선택된 0번 FND에 0~F 까지 출력

```
fndSelOut(aFndSelData[0]);              // select 0 FND
for(int i=0; i<16; i++) {
  fndDataOut(aFndData[i]);
  delay(1000);
}
```

  - FND_data_out 함수

FND_DATA 핀을 제어

```
void fndDataOut(uint8_t data)
{
  for(int i=0; i<8; i++) {
    digitalWrite(aPinFndData[i], (data>>i) & 0x1);
  }
}
```

예제

- 프로그램 설명

  - FND_sel_out 함수

FND_SEL 핀을 제어

```
void fndSelOut(uint8_t data)
{
  for(int i=0; i<6; i++)
  {
    digitalWrite(aPinFndSel[i], (data>>i) & 0x1);
  }
}
```

예제

- 전체 소스코드

```
1. int aPinFndData[8] = {13, 12, 11, 10, 9, 8, 7, 6}; // A~H pin
2. int aPinFndSel[6] = {5, 4, 3, 2, 1, 0}; // S0~S5 pin
3.
4. // 0~F data
5. uint8_t aFndData[16] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x27,
6.                         0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71};
7. // fnd select data
8. uint8_t aFndSelData[6] = {0x3E, 0x3D, 0x3B, 0x37, 0x2F, 0x1F};
9. void setup() {
10.   int i;
11.   for(i=0; i<8; i++) {
12.     // FND_DATA pin Output setup
13.     pinMode(aPinFndData[i], OUTPUT);
14.   }
15.   for(i=0; i<6; i++) {
16.     // FND_SEL pin Output setup
17.     pinMode(aPinFndSel[i], OUTPUT);
18.   }
19.}
```

예제

- 전체 소스코드

```
20.void loop() {
21.  fndSelOut(aFndSelData[0]);           // FND select out 0
22.  for(int i=0; i<16; i++) {
23.    fndDataOut(aFndData[i]);           // FND 0 ~ F output
24.    delay(1000);
25.  }
26.}
27.
28.void fndDataOut(uint8_t data) { // FND data output function
29.  for(int i=0; i<8; i++) {
30.    digitalWrite(aPinFndData[i],(data>>i) & 0x1);
31.  }
32.}
33.
34.void fndSelOut(uint8_t data) { // FND sel output function
35.  for(int i=0; i<6; i++){
36.    digitalWrite(aPinFndSel[i],(data>>i) & 0x1);
37.  }
38.}
```

Application practice

응용 실습

응용 실습

- 응용 문제
  - 첫번째 세그먼트에서 0~F까지 1초마다 출력하고 다음 세그먼트로 넘어가 이를 반복적으로 출력하시오.

- 구성
  - Arduino Uno
  - 6-Digit FND