
성 균 관 대 학 교
SIOR
로 봇 학 회

2022년 06월 19일

EMBEDDED

6 주 차

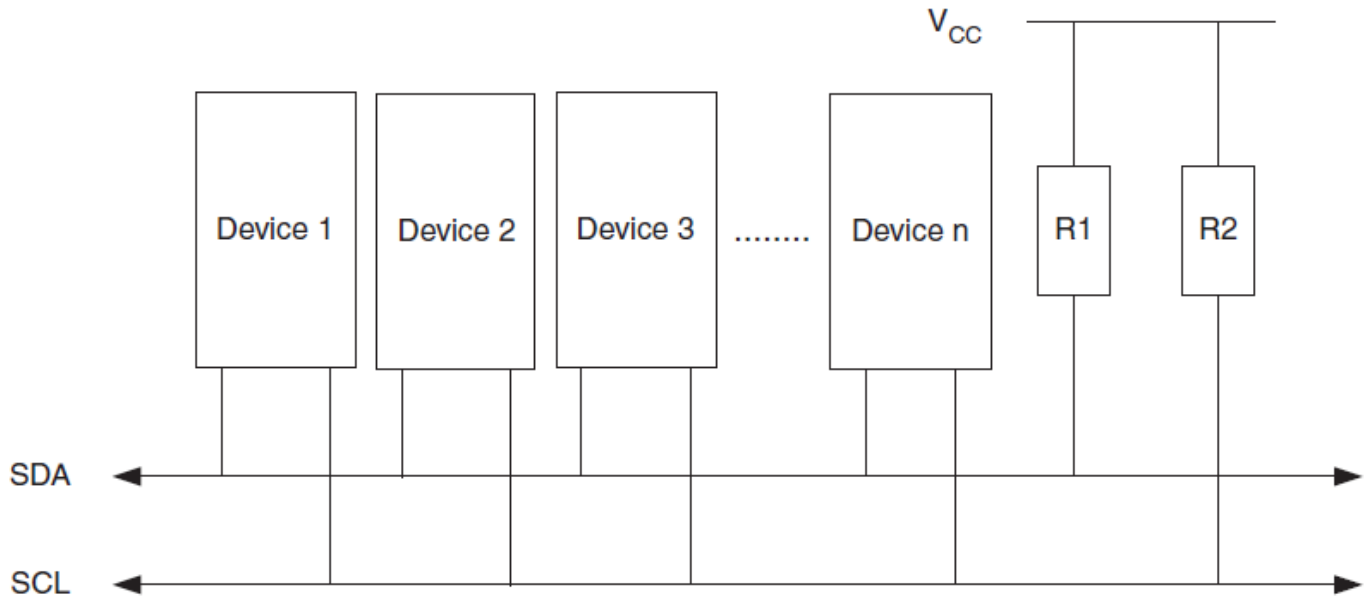
목차

EMBEDDED

I2C 소개

-시리얼 통신 방식
-여러 장치들과 최소한의 연결선만 가지고 여러 개의 장치와 연결
-저속통신방식 (<-> SPI고속통신)

Figure 22-1. TWI Bus Interconnection



EMBEDDED

I2C 소개

-시리얼 통신 방식
-여러 장치들과 최소한의 연결선만 가지고!
-저속통신방식

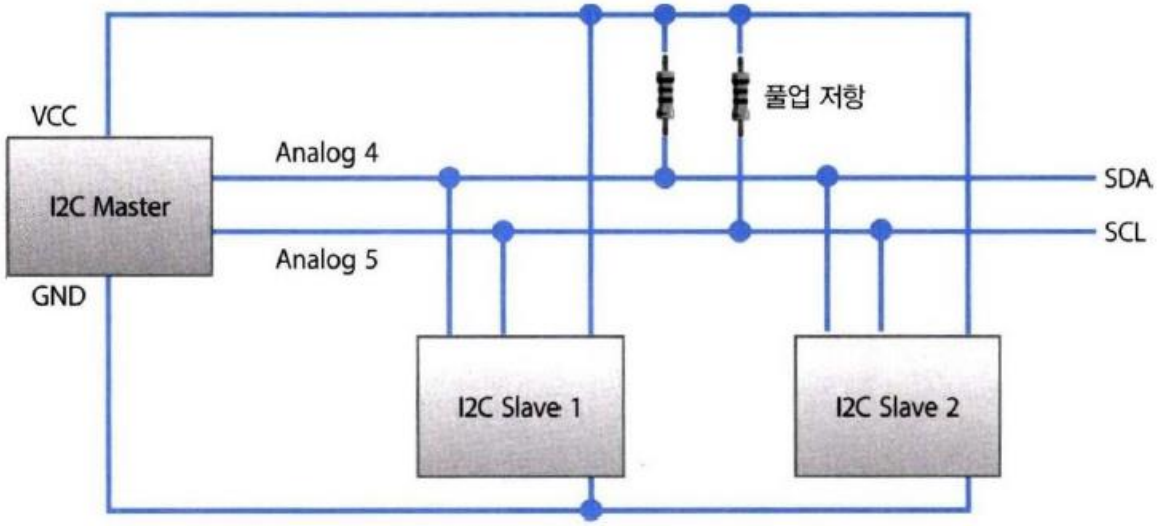


그림 17-6 I2C 연결 예

I2C 소개

...

표 17-1 시리얼 통신 방식 비교

		UART	SPI	I2C
동기/비동기		비동기	동기	동기
전이중/반이중		전이중	전이중	반이중
연결선	데이터	2	2	1(반이중)
	클록	0	1	1
	제어	0	1	0
	합계	2	4	2
	n개 슬레이브 연결	2n	3 + n	2
연결 방식		1:1	1:N(마스터-슬레이브)	1:N(마스터-슬레이브)
슬레이브 선택		-	하드웨어(SS 라인)	소프트웨어(주소 지정)

EMBEDDED

I2C 통신방법

-SDA(Serial Data): 데이터 전송선
-SCL(Serial Clock) : 데이터 통신에 필요한 클럭
-데이터는 SCL가 HIGH일 때 1바이트 단위로 샘플링
-데이터 전이는 SCL이 LOW일 때만 가능

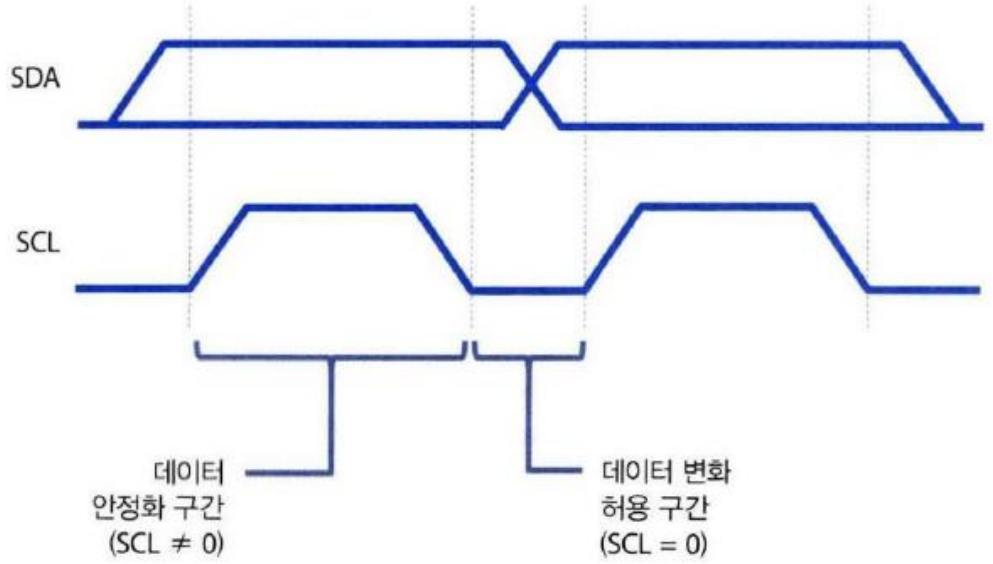


그림 17-1 데이터 샘플링 및 전이

EMBEDDED

I2C 통신방법

-시작조건: SCL = 1, SDA: H -> L
-정지조건: SCL = 1, SDA: L -> H

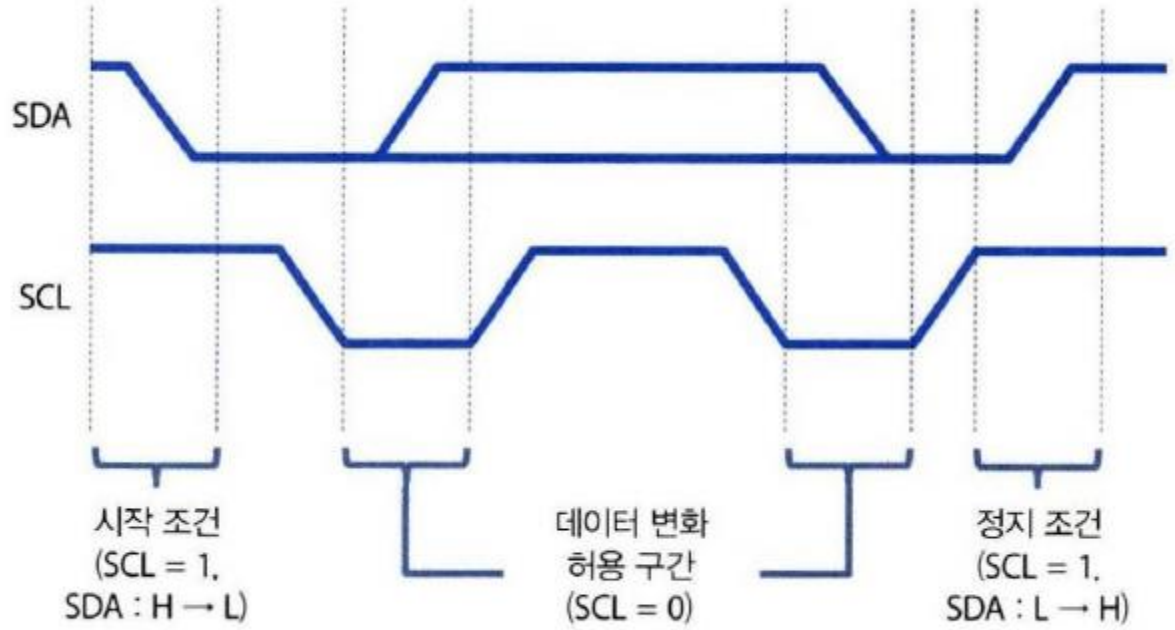


그림 17-2 데이터 전송 시작 및 종료

EMBEDDED

I2C 통신방법

-주소지정을 통해 어떤 장치와 송수신 할지 정할 수있다.
-주소는 7비트, 마지막 비트는 읽기/쓰기 모드 선택
-주소 '0000 000'은 마스터가 모든 슬레이브에 데이터를 전송할 때 사용(general call)
-LSB가 LOW: 마스터가 해당 주소로 데이터 송신 (쓰기 모드)
-LSB가 HIGH: 마스터가 해당 주소로 부터 데이터 수신 (읽기 모드)

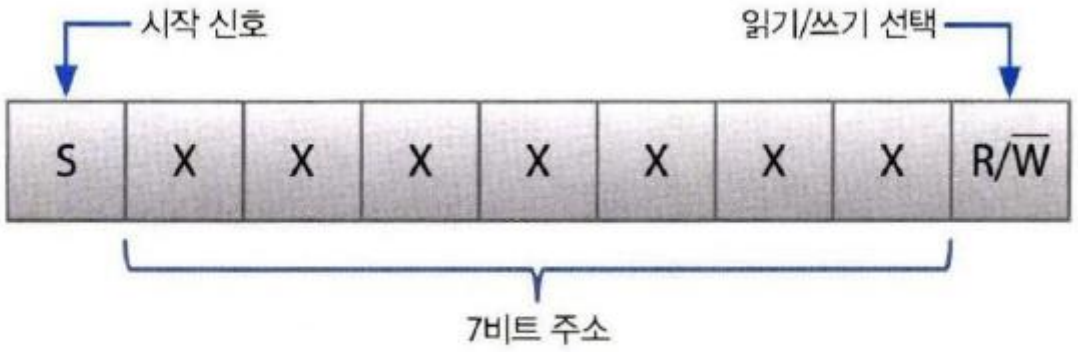


그림 17-3 어드레스 지정

I2C 통신방법

-데이터를 수신하는 장치는 데이터를 수신했음을 알려야함(ACK)
-데이터 전송이 끝난후 SDA는 HIGH
-데이터를 수신하는 장치는 LOW신호를 보내 데이터가 수신되었음을 알린다.(ACK 신호)
-만약 SDA가 HIGH라면 NACK으로 수신이 정상적으로 수신 되지 않음

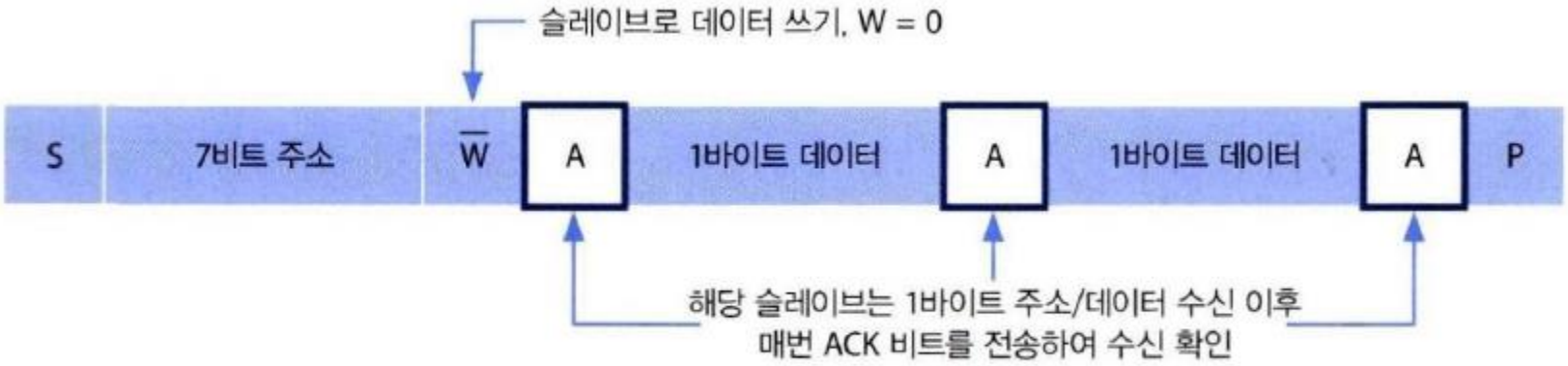


그림 17-4 마스터의 n 바이트 데이터 쓰기(■ : 마스터 전송, □ : 슬레이브 전송)

EMBEDDED

I2C 통신 방법

-데이터를 수신하는 장치는 데이터를 수신했음을 알려야함(ACK)
-데이터 전송이 끝난후 SDA는 HIGH
-데이터를 수신하는 장치는 LOW신호를 보내 데이터가 수신되었음을 알린다.(ACK 신호)
-만약 SDA가 HIGH라면 NACK으로 수신이 정상적으로 수신 되지 않음

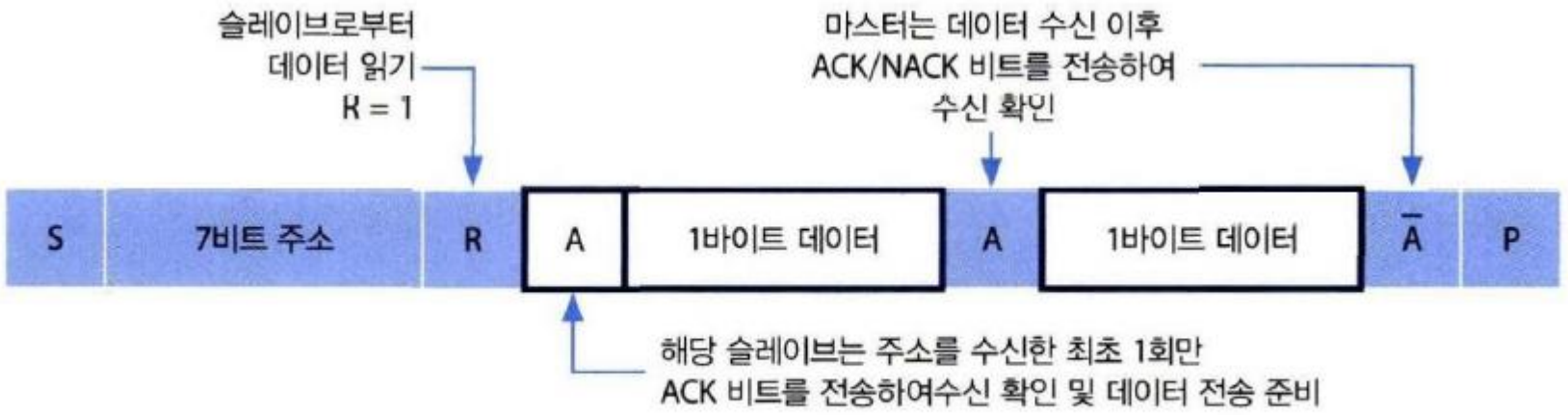


그림 17-5 마스터로의 n 바이트 데이터 읽기(■ : 마스터 전송, □ : 슬레이브 전송)

EMBEDDED

레지스터

-TWI 초기화 하기
-메인 클록은 I2C클록의 16배 이상이어야 한다.

$$\text{SCL 주파수} = \frac{\text{CPU 클록}}{16 + 2 \cdot \text{TWBR} \cdot \text{분주비}}$$

비트	7	6	5	4	3	2	1	0
	TWBR[7:0]							
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

그림 17-11 TWBR 레지스터의 구조

EMBEDDED

레지스터

-분주비는 TWSR레지스터의 TWPS1, TWPS0를 통해 설정

비트	7	6	5	4	3	2	1	0
	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0
읽기/쓰기	R	R	R	R	R	R	R/W	R/W
초깃값	1	1	1	1	1	0	0	0

그림 17-12 TWSR 레지스터의 구조

표 17-5 SCL 주파수의 분주비 설정

TWPS1	TWPS0	분주비
0	0	1
0	1	4
1	0	16
1	1	64

레지스터

비트	7	6	5	4	3	2	1	0
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
읽기/쓰기	R/W	R/W	R/W	R/W	R	R/W	R	R/W
초깃값	0	0	0	0	0	0	0	0

표 17-6 TWCR 레지스터 비트

비트	이름	설명
7	TWINT	TWI 인터럽트 플래그(TWI Interrupt Flag): TWI의 현재 작업이 완료되고 응용 소프트웨어의 응답을 기다릴 때 하드웨어에 의해 세트된다. 전역적으로 인터럽트가 SREG 레지스터에서 허용되도록 설정되고 TWCR 레지스터의 TWIE 비트가 세트되어 있을 때 TWINT 비트가 세트되면 인터럽트 서비스 루틴이 실행된다.
6	TWEA	TWI ACK 허용 비트(TWI Enable Acknowledge Bit): ACK 펄스의 생성을 제어한다. 다음과 같은 상황에서 ACK 펄스가 생성된다. 1. 슬레이브 모드에서 자신의 주소를 수신한 경우 2. 슬레이브 모드에서 TWAR 레지스터의 TWGCE 비트가 세트되고 모든 슬레이브에 전달되는 일반 호출(general call)을 수신한 경우 3. 마스터나 슬레이브 모드에서 한 바이트의 데이터를 수신한 경우
5	TWSTA	TWI 시작 조건 비트(TWI Start Condition Bit): 마스터 모드에서 전송을 시작하기 위해 TWSTA 비트를 세트한다.
4	TWSTO	TWI 정지 조건 비트(TWI Stop Condition Bit): 마스터 모드에서 전송을 끝내기 위해 TWSTO 비트를 세트한다.
3	TWWC	TWI 쓰기 충돌 플래그(TWI Write Collision Flag): TWINT 비트가 0인 경우, 즉 데이터가 전송 작업이 완료되지 않은 경우 TWI 데이터 레지스터인 TWDR 레지스터에 데이터를 기록하려고 하면 세트된다. TWINT 비트가 1인 경우 TWDR 레지스터에 데이터를 기록하면 클리어된다.
2	TWEN	TWI 허용 비트(TWI Enable Bit): TWI를 활성화시킨다.
1	-	예약됨
0	TWIE	TWI 인터럽트 허용 비트(TWI Interrupt Enable): TWI 데이터 전송이 완료된 경우 인터럽트 발생을 허용한다. SREG 레지스터의 I 비트가 세트되고, TWIE 비트가 세트된 경우 전송이 완료되면 인터럽트가 발생한다.

레지스터

...

코드 17-1 TWI 초기화 함수

```
#define I2C_SCL                 PC5
#define I2C_SDA                 PC4

void I2C_init(void) {
    DDRC |= (1 << I2C_SCL);             // SCL 핀을 출력으로 설정
    DDRC |= (1 << I2C_SDA);             // SDA 핀을 출력으로 설정

    TWBR = 32;                          // I2C 클록 주파수 설정 200KHz

    TWCR = (1 << TWEN) | (1 << TWEA);   // I2C 활성화, ACK 허용
}
```

레지스터

-TWI 시작함수
-TWRC 레지스터에 TWINT(인터럽트), TWSTA(시작조건)을 세트
-실제로 TWINT가 세트 되는 시점은 시작조건이 전송된 이후

코드 17-2 TWI 시작 함수

```
void I2C_start(void) {
    TWCR = _BV(TWINT) | _BV(TWSTA) | _BV(TWEN) | _BV(TWEA);

    while( !(TWCR & (1 << TWINT)) );              // 시작 완료 대기
}
```

레지스터

-I2C 송신함수
-TWDR레지스터에 전송할 데이터 입력
-TWCR레지스터에 TWINT, TWEN, TWEA비트를 세트
-실제로는 ACK신호가 도착해야 TWINT비트가 세트된다.

코드 17-3 TWI 바이트 송신 함수

```
void I2C_transmit(uint8_t data) {
    TWDR = data;
    TWCR = _BV(TWINT) | _BV(TWEN) | _BV(TWEA);

    while( !(TWCR & (1 << TWINT)) );              // 전송 완료 대기
}
```

비트	7	6	5	4	3	2	1	0
	TWD7..0							
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초깃값	1	1	1	1	1	1	1	1

그림 17-14 TWDR 레지스터의 구조

레지스터

-TWI 정지 함수
-TWCR레지스터에 TWSTO(정지조건)을 TWINT와 함께 세트
-TWINT가 1이 될 때까지 기다리지 않아도 된다.

코드 17-4 TWI 정지 함수

```
void I2C_stop(void) {
    TWCR = _BV(TWINT) | _BV(TWSTO) | _BV(TWEN) | _BV(TWEA);
}
```

레지스터

-TWI 수신함수
-전송이 완료되지 않았다면 ACK신호를 보내고, 마지막 데이터가 들어올 때 NACK신호를 보낸다.
-ACK신호를 보낼 때는 TWEA비트를 세트한다.
-NACK 신호를 보낼 때는 TWEA비트를 세트하지 않는다.
-수신한 데이터는 TWDR레지스터를 통해 얻는다.

코드 17-5 TWI 바이트 수신 함수

```
uint8_t I2C_receive_ACK(void) {
    TWCR = _BV(TWINT) | _BV(TWEN) | _BV(TWEA);
    while( !(TWCR & (1 << TWINT)) );              // 수신 완료 대기
    return TWDR;
}

uint8_t I2C_receive_NACK(void) {
    TWCR = _BV(TWINT) | _BV(TWEN);
    while( !(TWCR & (1 << TWINT)) );              // 수신 완료 대기
    return TWDR;
}
```

실습(DS1307, RTC모듈)

Real Time Clock : 별도의 발진기와 전원을 가지고 있어서 정확한 시간 측정이 가능하다.

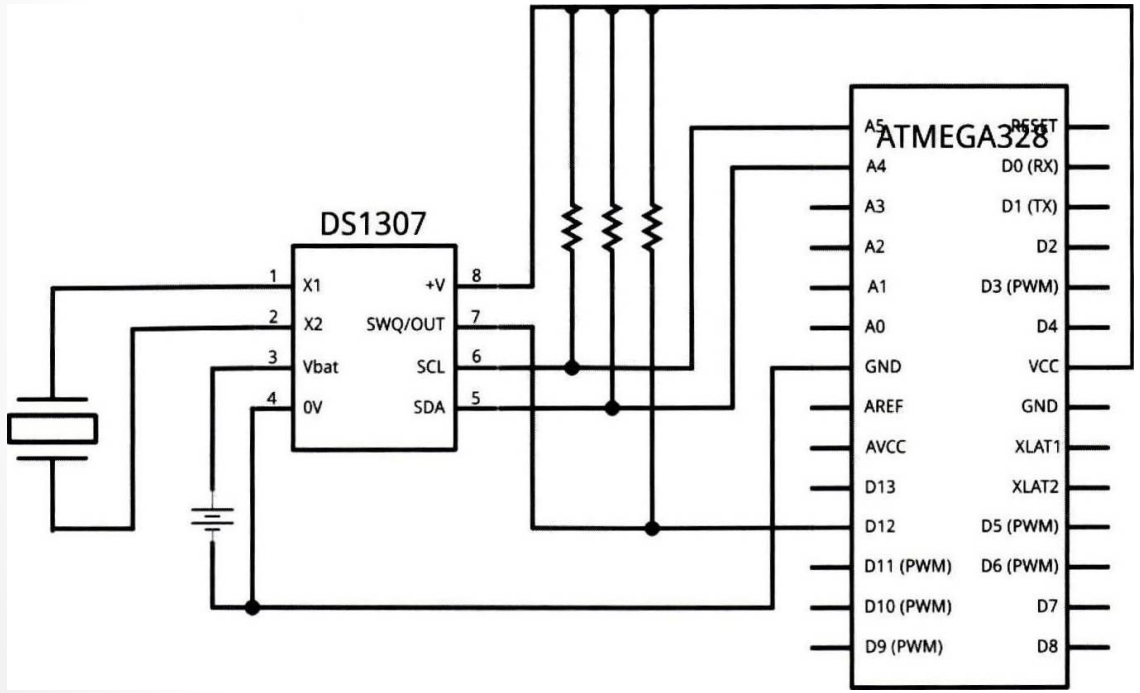


표 17-2 DS1307 핀 설명

핀 번호	이름	설명	비고
1	X1	RTC용 발진기	32.768KHz
2	X2		
3	V_{BAT}	RTC용 전원	3.0V
4	GND		
5	SDA	I2C 데이터	아날로그 4번에 연결
6	SCL	I2C 클록	아날로그 5번에 연결
7	SQW/OUT	구형파 출력	1, 4, 8, 32KHz 출력
8	VCC		4.5 ~ 5.5V

EMBEDDED

실습(DS1307, RTC모듈)

-DS1307의 메모리에 시간정보가 저장 (0x00, 0x06)
-i2c통신으로 메모리에 접근해서 시간정보를 읽어 들인다.
-시간정보는 BCD형식으로 저장되어 있기 때문에 이진수 방식으로 변환 필요

표 17-3 DS1307 메모리 구조

주소	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	내용	값 범위
0x00	CH	초(10의 자리)			초(1의 자리)				초	00~59
0x01	0	분(10의 자리)			분(1의 자리)				분	00~59
0x02	0	0 1	시(10의 자리)		시(1의 자리)				시	00~23 1~12
0x03	0	0	0	0	0	요일			요일	01~07
0x04	0	0	일(10의 자리)		일(1의 자리)				일	01~31
0x05	0	0	0	월(10의 자리)	월(1의 자리)				월	01~12
0x06	연(10의 자리)				연(1의 자리)				연	00~99
0x07	OUT	0	0	SQWE	0	0	RS1	RS0	제어	-
0x08 ~ 0x3F									RAM	00h ~ FFh

실습(DS1307, RTC모듈)

-코드 업로드 후, 아두이노를 PC에 연결해 Terminal창으로 결과 확인 가능

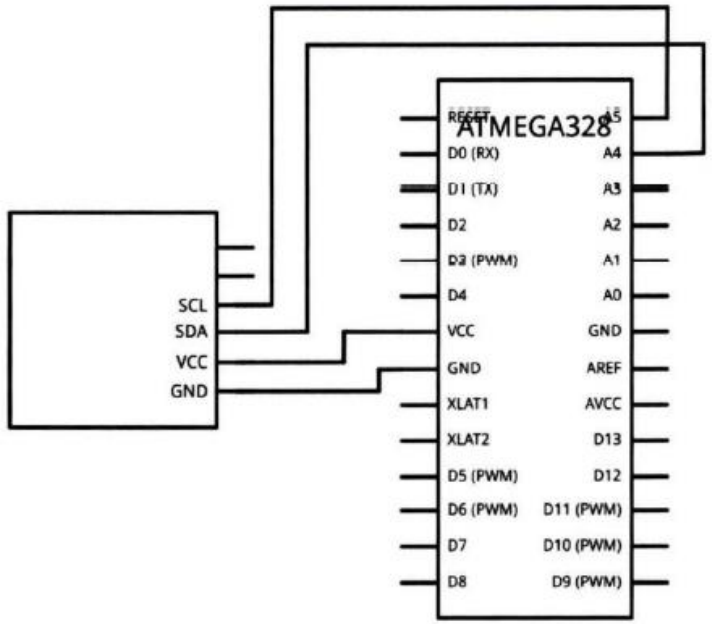
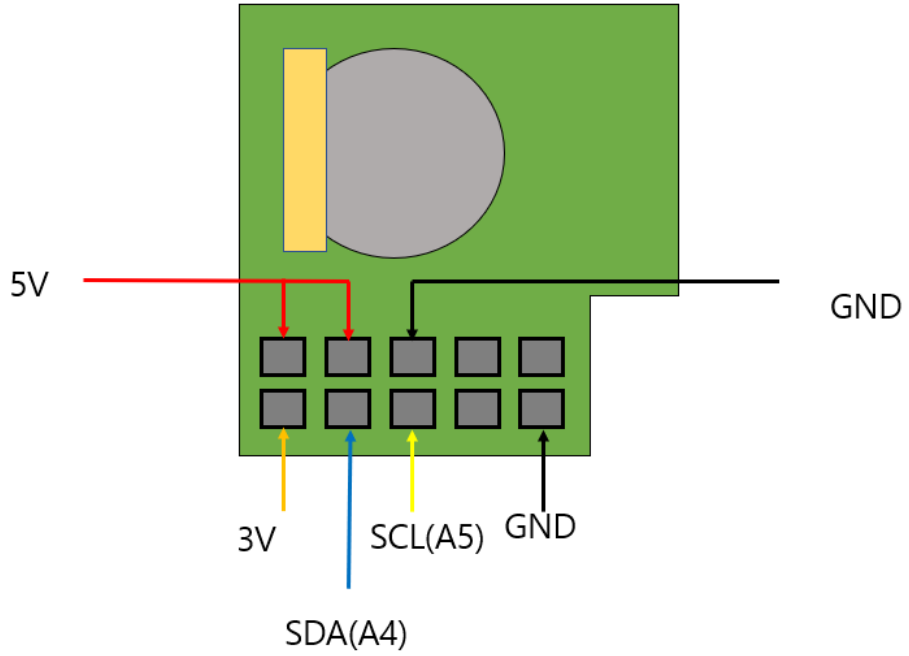


그림 17-10 Tiny RTC 연결 회로도



LCD 개념 및 핀

2줄 16 글자 씩 총 32글자를 표시 가능

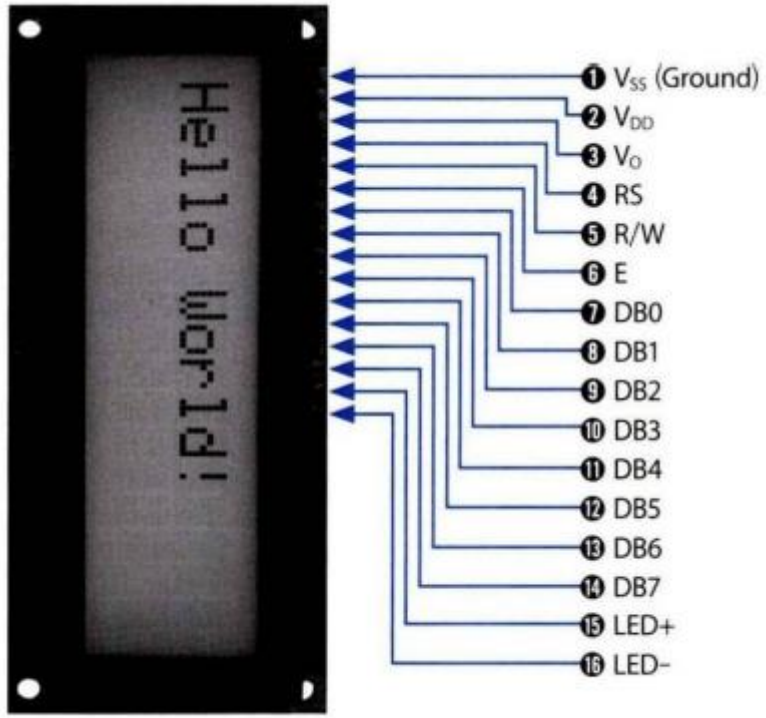


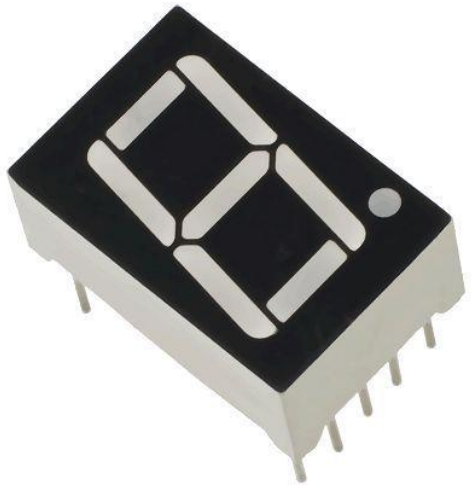
그림 21-1 텍스트 LCD 모듈

표 21-1 텍스트 LCD의 핀

핀 번호	이름	설명
1	V_{SS}	그라운드(GND)
2	V_{DD}	5V 동작 전원
3	V_O	LED 전원으로 가변저항을 통해 0~5V 사이 입력
4	RS	레지스터 선택(Register Select)
5	R/W	읽기/쓰기(Read/Write)
6	E	활성화(Enable)
7	DB0	데이터 신호 핀
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	LED+	백라이트 전원
16	LED-	

7 세그먼트 표시장치

7개의 선분으로 숫자나 글자를 표시하기 위해 LED를 사용하여 만든 출력 장치
보통 소수점 부분까지 포함하여 8개의 세그먼트로 이루어져 있기는 함.
한 자리 7세그먼트(핀 10개)와 4자리 7세그먼트(핀 12개)의 제어 방법이 다름.



한자리 7 세그먼트 표시장치 제어

공통양극방식/공통음극방식: VCC/GND
공통핀/제어핀: 항상 VCC나 GND 중 하나가 일정하게 가해짐/선택적으로 가해짐
10개 중 2개 공통핀 8개 제어핀, 핀번호는 반시계 방향으로 증가

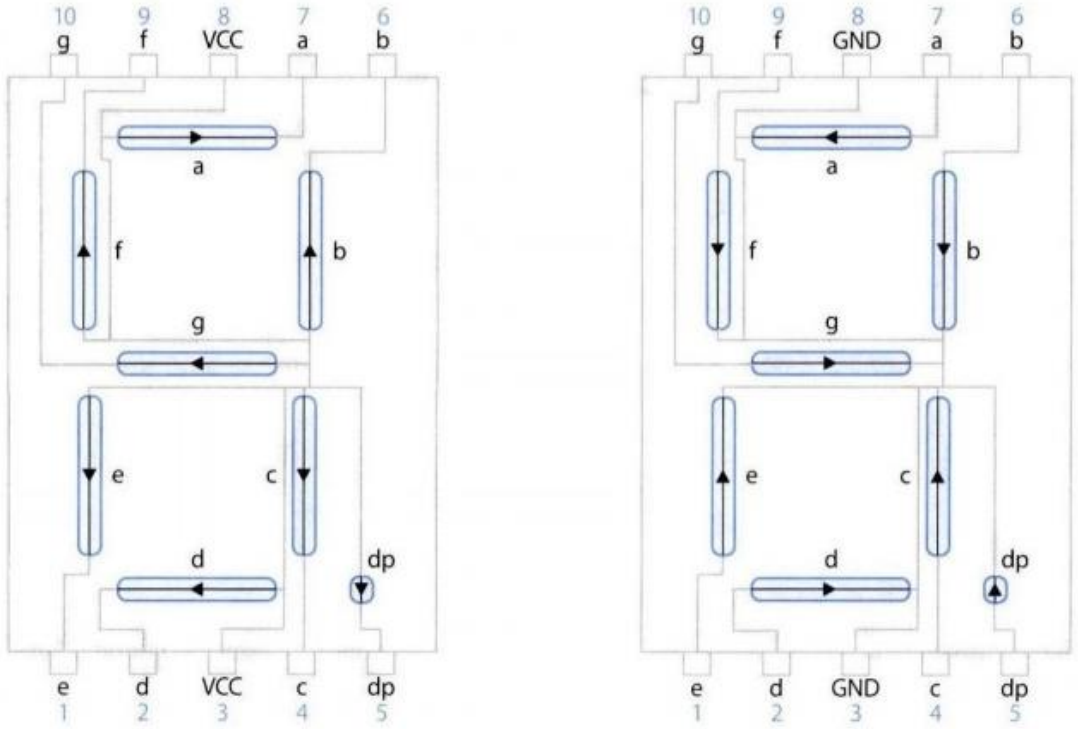


그림 19-2 7 세그먼트 표시장치 및 핀 배열

표 19-1 공통 양극 방식과 공통 음극 방식의 제어

방식		공통 양극	공통 음극
공통 핀 연결		VCC	GND
제어 핀 연결	세그먼트 ON	GND	VCC
	세그먼트 OFF	VCC	GND

한자리 7 세그먼트 표시장치 제어

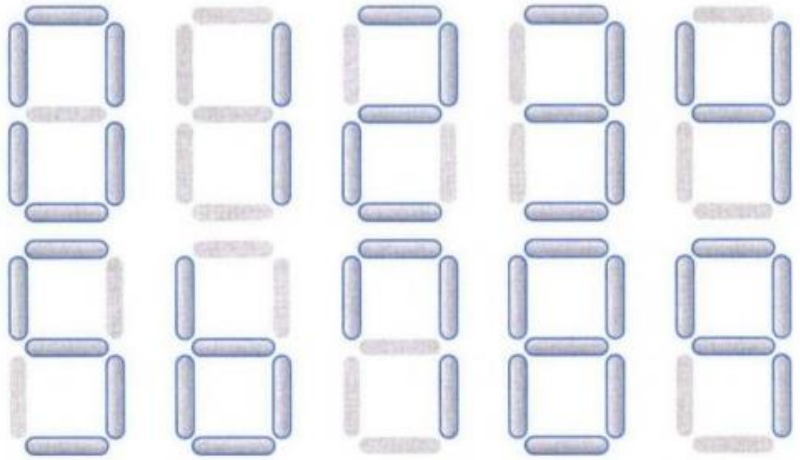
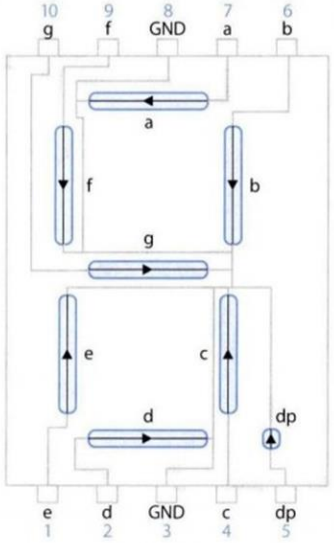


그림 19-3 7 세그먼트 표시장치에서 숫자의 표현

제어핀을 범용 입출력 핀에 연결하여 제어 가능
7 세그먼트이므로 7개의 범용 입출력 핀이 필요



(b) 공통 음극 방식

표 19-2 한 자리 7 세그먼트에 숫자를 표시하기 위한 데이터(○: on, ×: off)

숫자	세그먼트								제어값	
	a	b	c	d	e	f	g	dp	2진값	16진값
0	○	○	○	○	○	○	×	×	1111 1100	FC
1	×	○	○	×	×	×	×	×	0110 0000	60
2	○	○	×	○	○	×	○	×	1101 1010	DA
3	○	○	○	○	×	×	○	×	1111 0010	F2
4	×	○	○	×	×	○	○	×	0110 0110	66
5	○	×	○	○	×	○	○	×	1011 0110	B6
6	○	×	○	○	○	○	○	×	1011 1110	BE
7	○	○	○	×	×	○	×	×	1110 0100	E4
8	○	○	○	○	○	○	○	×	1111 1110	FE
9	○	○	○	×	×	○	○	×	1110 0110	E6

한자리 7 세그먼트 표시장치 제어

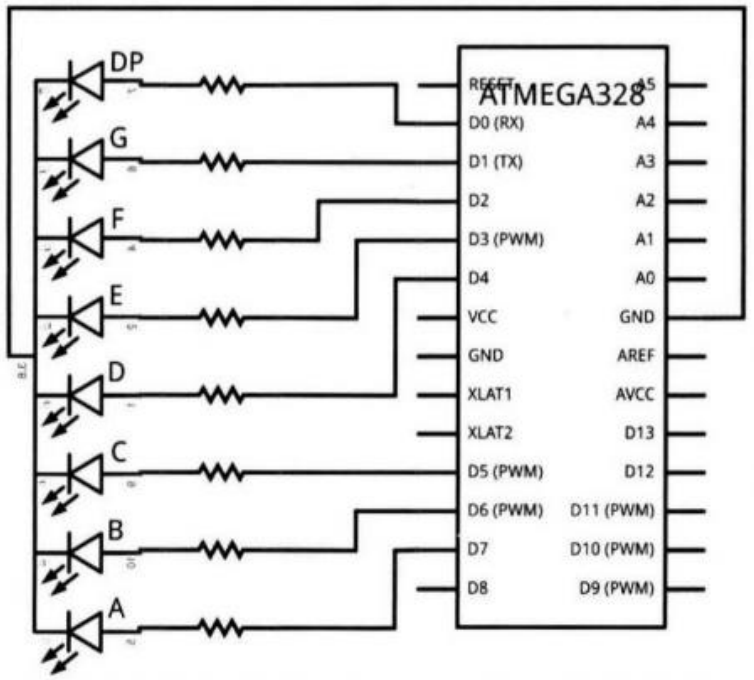


그림 19-4 한 자리 7 세그먼트 표시장치 연결 회로

제어핀을 범용 입출력 핀에 연결하여 제어 가능
7 세그먼트이므로 7개의 범용 입출력 핀이 필요

한자리 7 세그먼트 표시장치 제어

실습: 상향 카운터->1초 간격으로 0~9까지 반복적으로 표시하는 코드 (p.377~382)
제어핀은 포트D를 사용하므로 PORTD 레지스터를 통해 제어 가능

19-1: 1초 간격으로 숫자를 표시하기 위해 _delay_ms 함수 사용
문제: 이 함수가 실행 중인 동안에는 다른 작업 진행X

19-2: 타이머를 통해 인터럽트 이용->코드에서 8비트 타이머/카운터 사용
인터럽트횟수=$\frac{\text{클록 주파수/분주비}}{\text{오버플로까지 클록}} = \frac{16\cdot 10^6/2^{10}}{256}$=61회
즉, 인터럽트 서비스 루틴(ISR)에서 인터럽트 발생횟수를 카운트하고 메인 루프에서 61번의 인터럽트가 발생하였는지 여부를 검사하면 1초 경과를 확인 가능

19-3: 아두이노에서는 인터럽트를 거의 사용X
대신 폴링방식으로 프로그램 경과시간을 millis함수로 알아내고 원하는 경과시간과 비교
Mills함수는 내부적으로는 타이머/카운터 0번, 분주비64의 오버플로 인터럽트 사용
오버플로 발생하려면 64*256 clk필요, 아두이노 우노 16MHz clk사용->1마이크로초당 16개 clk발생
오버플로 발생할 때까지 시간=64*256/16=1024마이크로초
이 값을 밀리와 마이크로초로 분리하여 변수에 저장하고 오버플로 인터럽트 발생할 때마다 값 갱신

네자리 7 세그먼트 표시장치 제어

한자리 7세그먼트 표시장치를 제어하기 위해서는 (소수점 포함) 8개의 범용 입출력 핀 필요
네자리의 경우 *4한값인 32개가 필요하게 됨 ->아두이노는 최대 20개의 GPIO만 사용 가능
잔상효과(afterimage effect)이용
: 잔상효과를 이용하여 빠른 속도로 네 자리 숫자를 반복적으로 켜준다면 사람의 눈은 네 자리 숫자가 동시에 표시되는 것과 동일하게 느껴지게 된다.
12개 핀, 소수점 있는 면을 아래로 했을 때 가장 왼쪽 핀이 1번, 반시계방향 증가

그림 19-5 네 자리 7 세그먼트 표시장치

네자리 7 세그먼트 표시장치 제어

12개 핀 中 8개: 세그먼트 제어
나머지 4개: 네 자리 숫자 중 특정 순간에 어느 자리에 출력할 것인지 결정하기 위해 사용

네 자리 숫자 중 하나 선택하기 위한 핀:
공통음극방식:
켜고 싶은 자리 0
끄로 싶은 자리 1
공통 양극방식: 반대

세그먼트 제어(8개): 핀번호(세그먼트이름)
한자리와 동일하게 켜고 싶은 것에 1, 끄고 싶은 것에 0 주기->공통음극방식
공통양극반식은 반대로

그림 19-6 네 자리 7 세그먼트 표시장치 회로도

네자리 7 세그먼트 표시장치 제어

실습: 첫 번째 자리에 0부터 9까지 0.1초 간격으로 숫자를 출력하고 다음 자리로 이동하여 0부터 9까지 숫자 출력을 반복 (p.385~389)

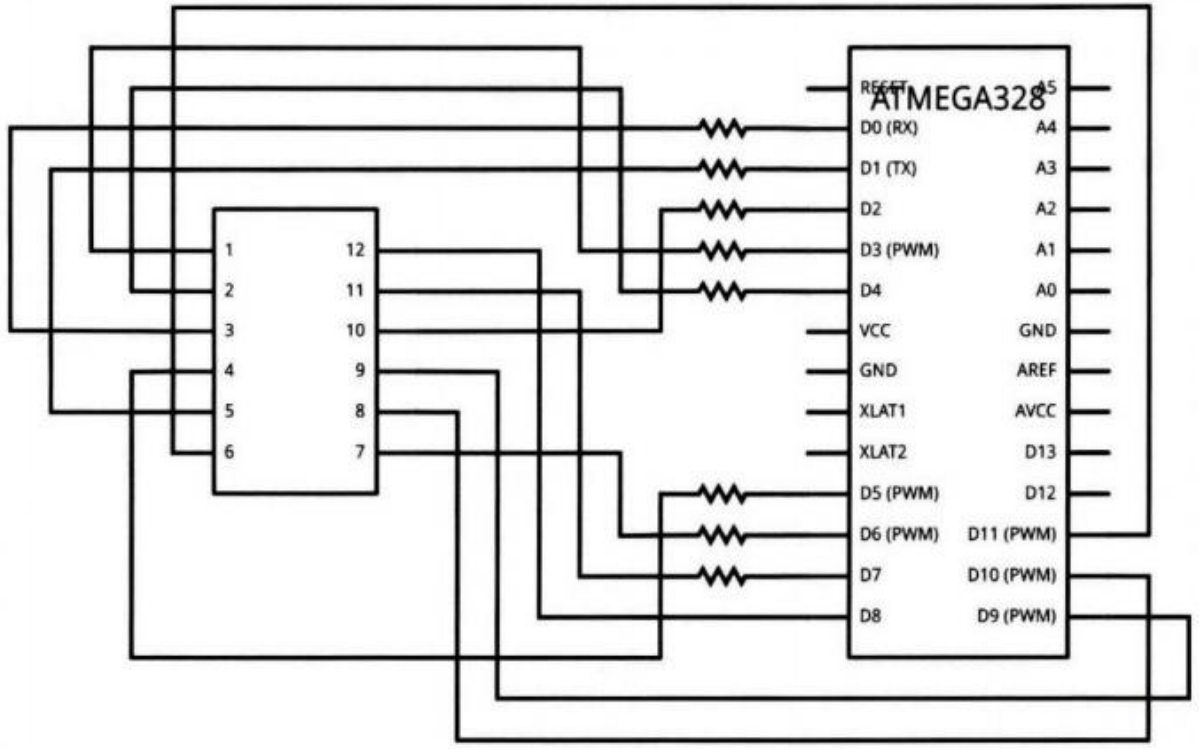


그림 19-7 네 자리 7 세그먼트 모듈 연결 회로도

네자리 7 세그먼트 표시장치 제어

실습: 첫 번째 자리에 0부터 9까지 0.1초 간격으로 숫자를 출력하고 다음 자리로 이동하여 0부터 9까지 숫자 출력을 반복 (p.385~389)
앞의 한자리 7세그먼트 표시장치 제어 실습과 거의 동일

19-6, 7: 0.1초 간격으로 숫자를 표시하기 위해 _delay_ms 함수 사용
19-6: 네 자리 7세그먼트 표시장치에 한 자리 숫자만 나타내는 코드
19-7: 네 자리 7세그먼트 표시장치에 네 자리 숫자 나타내는 코드
문제점: 앞에서 언급했듯이 _delay_ms 함수는 실행 중인 동안에는 다른 작업 진행X, 따라서 이 함수가 실행되는 동안 표시장치가 갱신되지 않아 네 자리 숫자가 정확하게 표시되지 않는다.
잔상효과가 0.1초 보다 짧은 이유도 있음.

19-8: millis 함수 사용하여 모든 자리가 동시에 켜진 것처럼 보이게 한 코드
갱신 주기를 더 짧게 하여 잔상효과가 가능하게끔 함.

LCD 제어

표 21-2 텍스트 LCD의 제어 핀

제어 핀	설명
RS (Register Select)	텍스트 LCD를 제어하기 위해서는 2개의 레지스터, 즉 제어 레지스터와 데이터 레지스터가 사용되며, RS 신호는 명령을 담고 있는 레지스터(RS = LOW)와 데이터를 담고 있는 레지스터(RS = HIGH) 중 하나를 선택하기 위해 사용된다.
R/W(Read/Write)	읽기(R/W = HIGH) 및 쓰기(R/W = LOW) 모드를 선택하기 위해 사용된다. 일반적으로 LCD는 데이터를 쓰기 위한 용도로만 사용하므로 일반적으로 R/W 신호는 GND에 연결한다.
E(Enable)	하강 에지(falling edge)에서 LCD 드라이버가 처리를 시작하도록 지시하기 위한 신호로 사용된다.

표 21-3 텍스트 LCD 제어 명령어

명령	명령 코드										설명
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Clear Display	0	0	0	0	0	0	0	0	0	1	공백문자(코드 0x20)로 화면을 지우고 커서를 홈 위치(주소 0번)로 이동시킨다.
Return Home	0	0	0	0	0	0	0	0	1	-	커서를 홈 위치로 이동시키고, 표시 영역이 이동된 경우 초기 위치로 이동시킨다. 화면에 출력된 내용(DDRAM의 값)은 변하지 않는다.
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	데이터 읽기 또는 쓰기 후 메모리의 증가(increment) 또는 감소(decrement) 방향을 지정한다. DDRAM에서 I/D = 1이면 커서를 오른쪽으로, I/D = 0이면 왼쪽으로 옮긴다. S = 1이면 I/D 값에 따라 디스플레이를 왼쪽 또는 오른쪽으로 옮기며 이 때 커서는 고정된 위치에 나타난다.
Display on/off Control	0	0	0	0	0	0	1	D	C	B	디스플레이(D), 커서(C), 커서 깜빡임(B)의 on/off를 설정한다.

표 21-3 텍스트 LCD 제어 명령어(계속)

명령	명령 코드										설명
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	-	-	화면에 출력된 내용의 변경 없이 커서와 화면을 이동시킨다(표 21-4).
Function Set	0	0	0	0	1	DL	N	F	-	-	데이터 비트 크기(DL = 1이면 8비트, DL = 0이면 4비트), 디스플레이의 행 수(N), 폰트 크기(F)를 설정한다(표 21-5).
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	주소 카운터에 CGRAM 주소를 설정한다.
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	주소 카운터에 DDRAM 주소를 설정한다.
Read Busy Flag & Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	드라이버에서 현재 명령어가 실행 중인지 여부를 나타내는 동작 중 플래그(Busy Flag) 값과 주소 카운터의 값을 읽어 온다.
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	RAM(DDRAM 또는 CGRAM)에 데이터를 기록한다.
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	RAM(DDRAM 또는 CGRAM)에서 데이터를 읽어 온다.

프로젝트

주제 선정

메이커 스타 6.30일까지 신청서 제출 -> 프로젝트 진행 -> 9.30 임베디드 SW 제출

1. 스마트 옷장: 날씨에 따라 옷 뽑아 주기
    1. 여행 활용
    2. 스마트 서랍
    3. (시각장애인을 위한)
2. 시각장애인을 위한 스마트 글래스
    1. 전방 물체 인식 -> 음성 안내
    2. 시각장애인을 위한 만능 리모컨
    3. 신호등 음성 안내 버튼 대체 -> IR 리모컨
3. 펜에 가속도 센서 -> 글씨 쓰기 (for 청각장애인)
4. 전기차 무선 충전: 터틀봇
5. 모니터 식탁
6. 유압 조절 웨이트 머신
7. 물체 크기에 따른 버튼 깊이 조절 VR 컨트롤러

성 균 관 대 학 교

Thank You

로 봇 동 아 리