

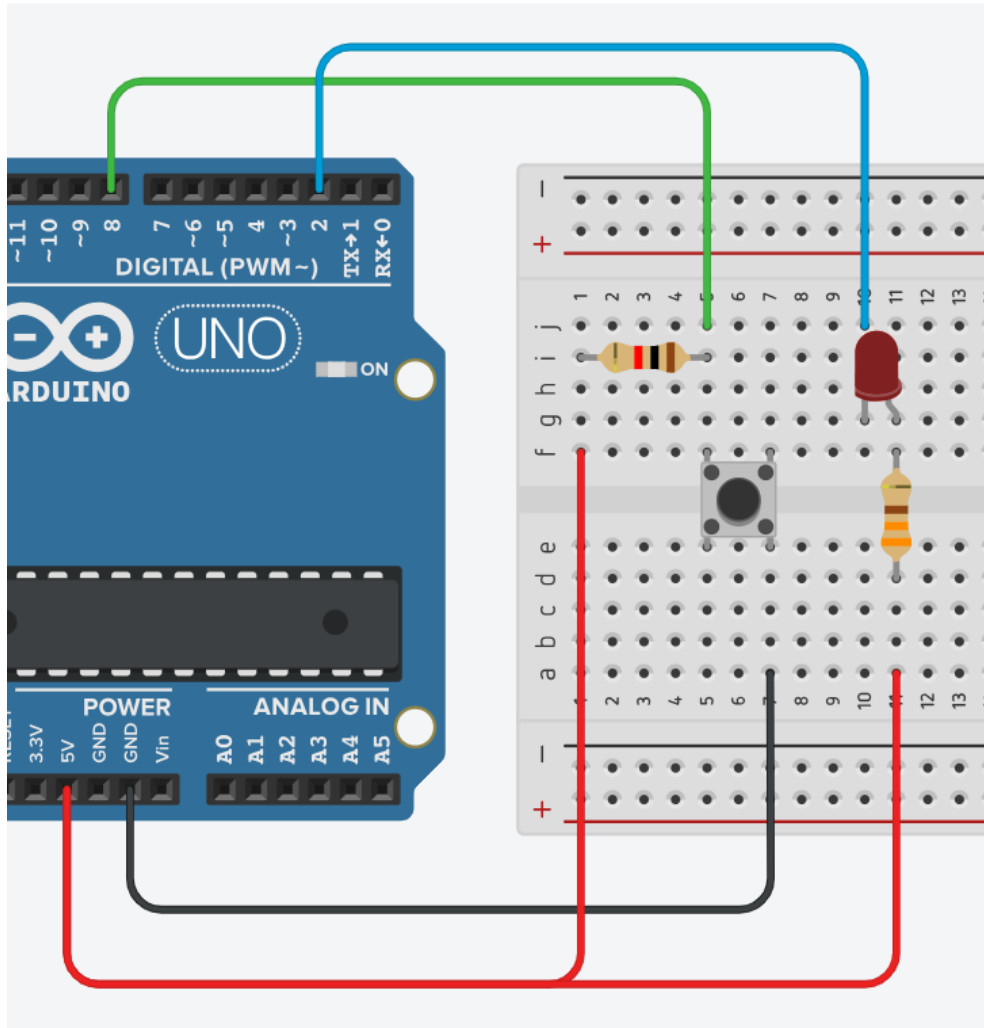
아날로그 입력, LCD 실험

마이크로프로세서 종합 설계. 8주차.



Digital GOIP 실험

digitalWrite / digitalRead 실험

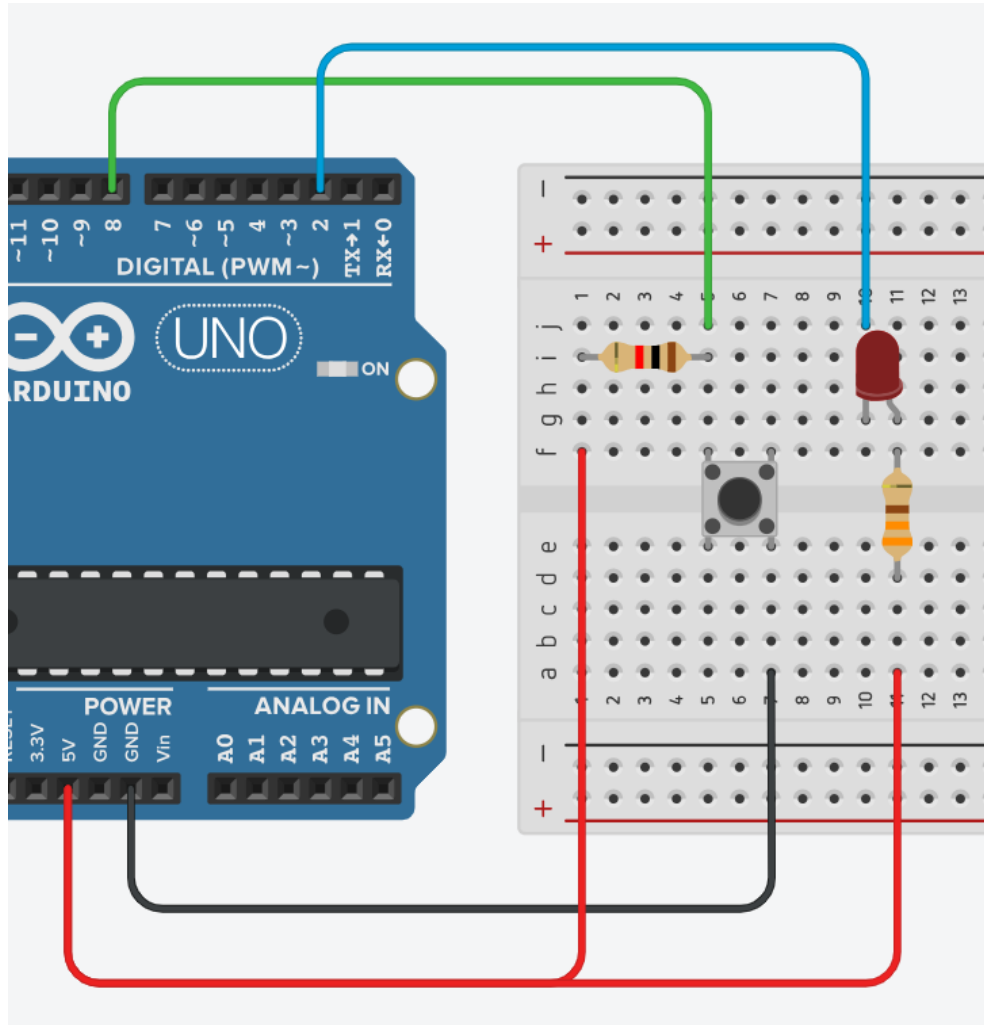


아두이노 Digital IO에 필요한 API

- pinMode(핀번호, 입출력옵션)
- digitalRead(핀번호)
- digitalWrite(핀번호, 출력값)

Digital GOIP 실험

digitalWrite / digitalRead 실험



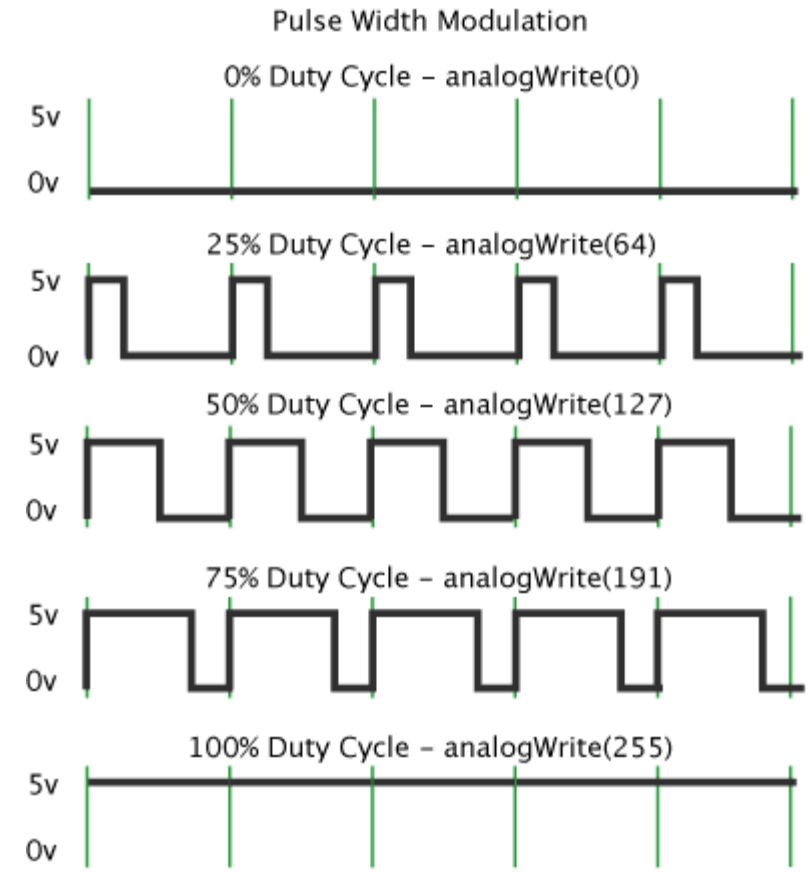
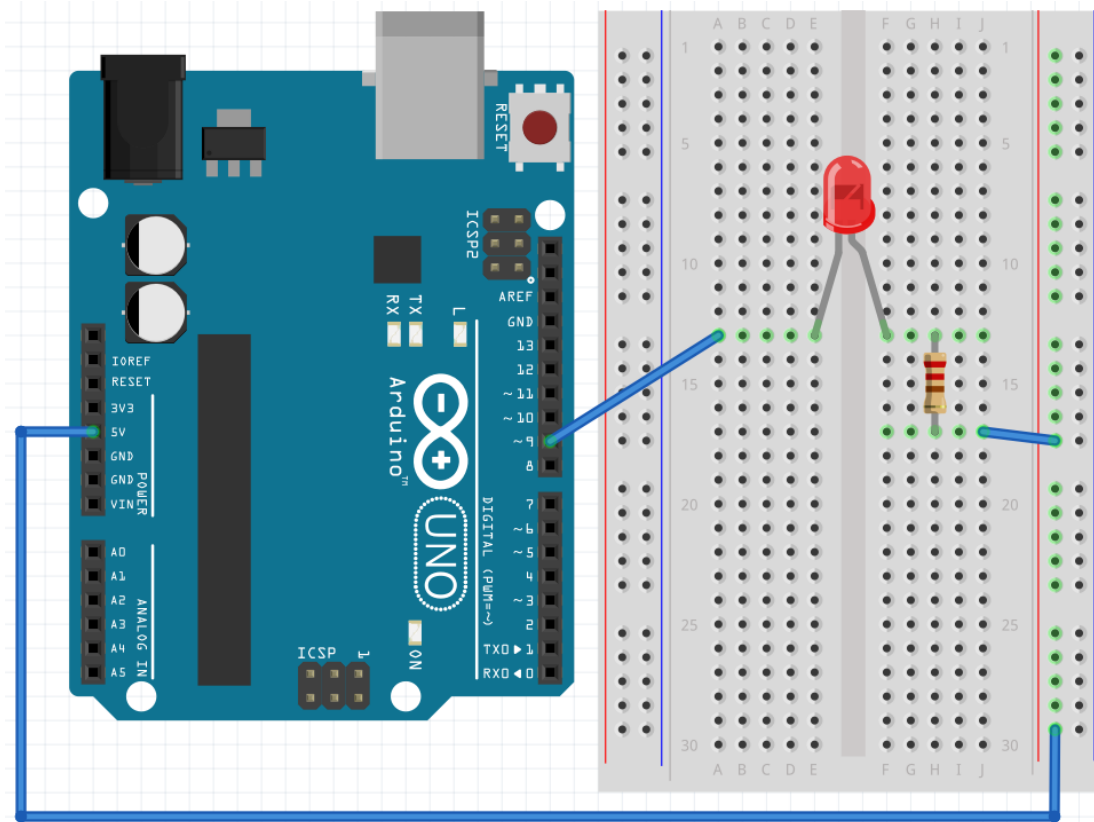
```
void setup()
{
  pinMode(8, INPUT);
  pinMode(2, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  int read = digitalRead(8);
  Serial.println(read);

  if( read == 0 )
  {
    digitalWrite(2, LOW);
  }
  else
  {
    digitalWrite(2, HIGH);
  }
}
```

analogWrite(PWM) 실험

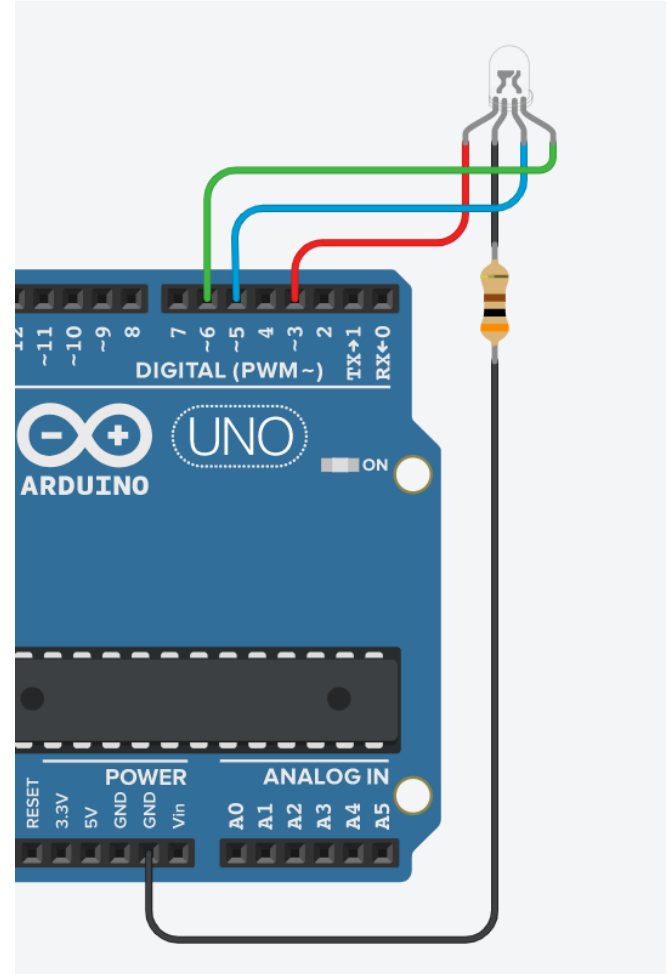
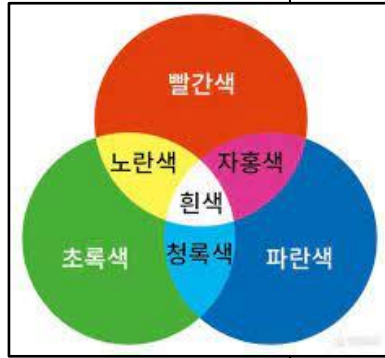
- 함수 : `analogWrite(핀번호, duty cycle)`



analogWrite(PWM)을 이용한 RGB LED제어

```
void setup()
{
  pinMode(3, OUTPUT); // 핀을 출력으로 설정
  pinMode(5, OUTPUT); // 핀을 출력으로 설정
  pinMode(6, OUTPUT); // 핀을 출력으로 설정
}
```

```
void loop()
{
  analogWrite(3, 255); //analogWrite 값은 0 부터 255까지
  analogWrite(5, 255); //analogWrite 값은 0 부터 255까지
  analogWrite(6, 0); //analogWrite 값은 0 부터 255까지
}
```



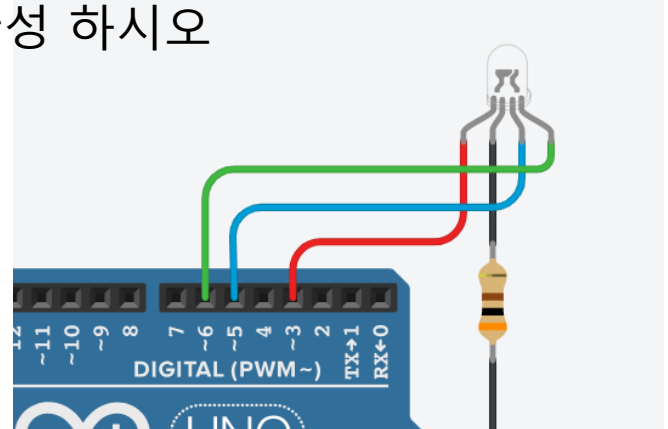
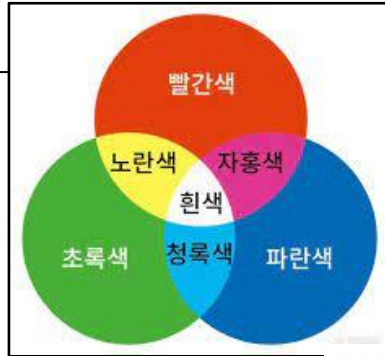
analogWrite(PWM)을 이용한 3색(full color) RGB LED제어

QUIZ : 3색 LED의 광량을 조절하여 아래의 **청록색**이 표현 되도록 코드를 완성 하시오

RGB(121, 207, 203)

```
void setup()
{
  pinMode(3, OUTPUT); // 핀을 출력으로 설정
  pinMode(5, OUTPUT); // 핀을 출력으로 설정
  pinMode(6, OUTPUT); // 핀을 출력으로 설정
}

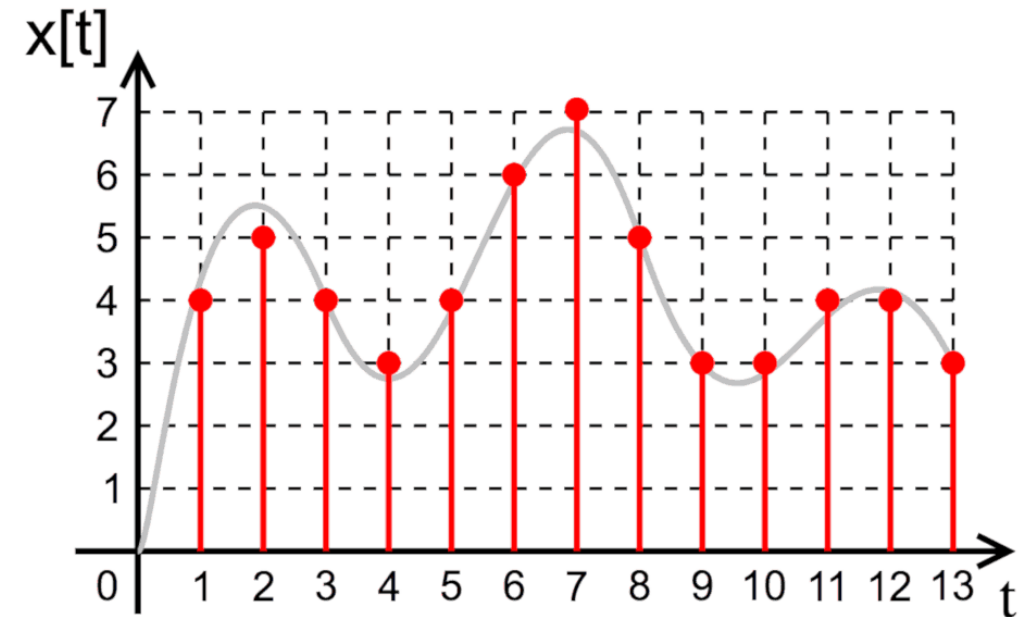
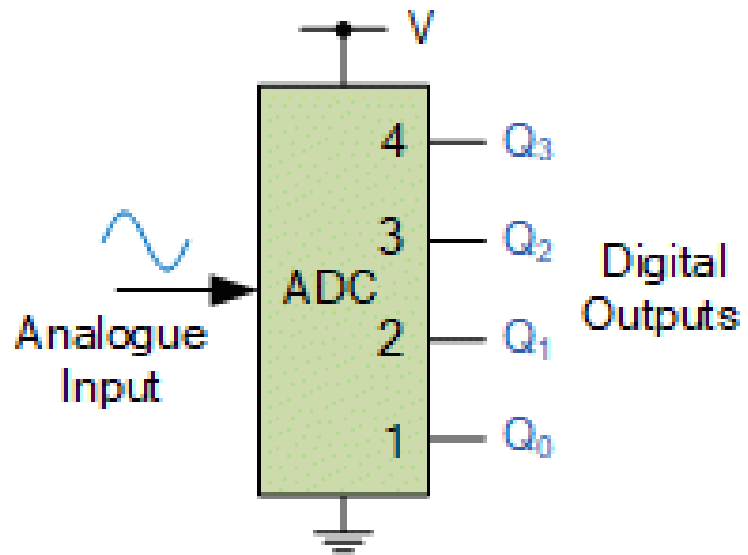
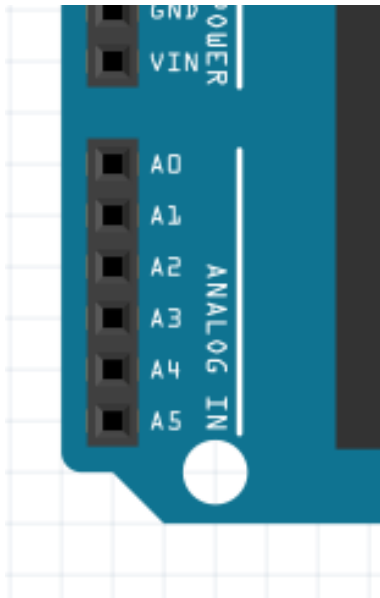
void loop()
{
  analogWrite(3, [red]); //analogWrite 값은 0 부터 255까지
  analogWrite(5, [blue]); //analogWrite 값은 0 부터 255까지
  analogWrite(6, [green]); //analogWrite 값은 0 부터 255까지
}
```



색 편집

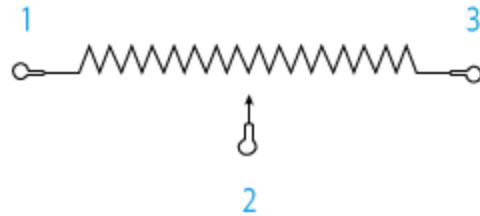
Color picker interface showing a rainbow gradient and a selected color circle. The selected color is #79CFCB. The interface includes a vertical color bar, a brightness slider, and input fields for RGB values: 121 (Red), 207 (Green), and 203 (Blue). The labels '빨강', '녹색', and '파랑' are next to the respective input fields.

아날로그 입력(ADC)



가변저항(Potentiometer, 볼륨)

- 저항값을 변경

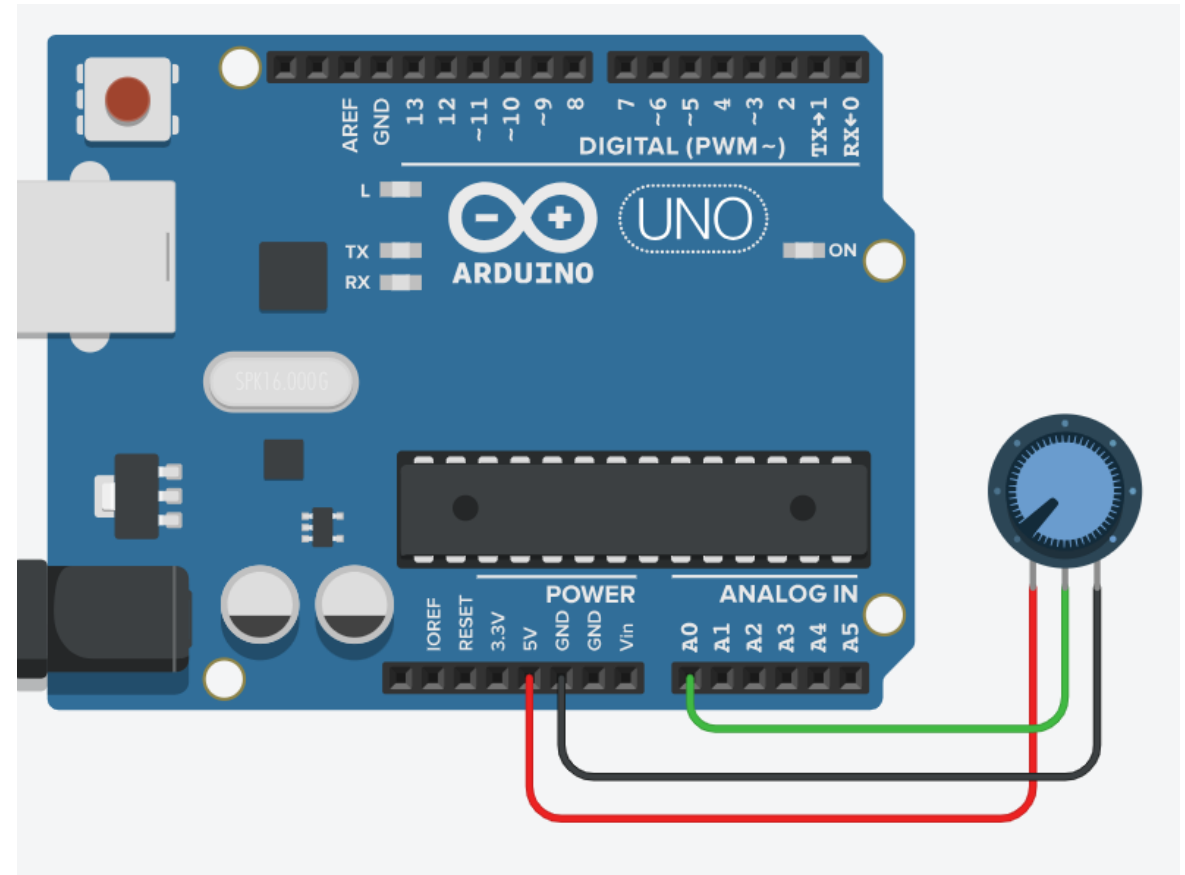


가변저항의 동작방식

가변저항(Potentiometer, 볼륨)

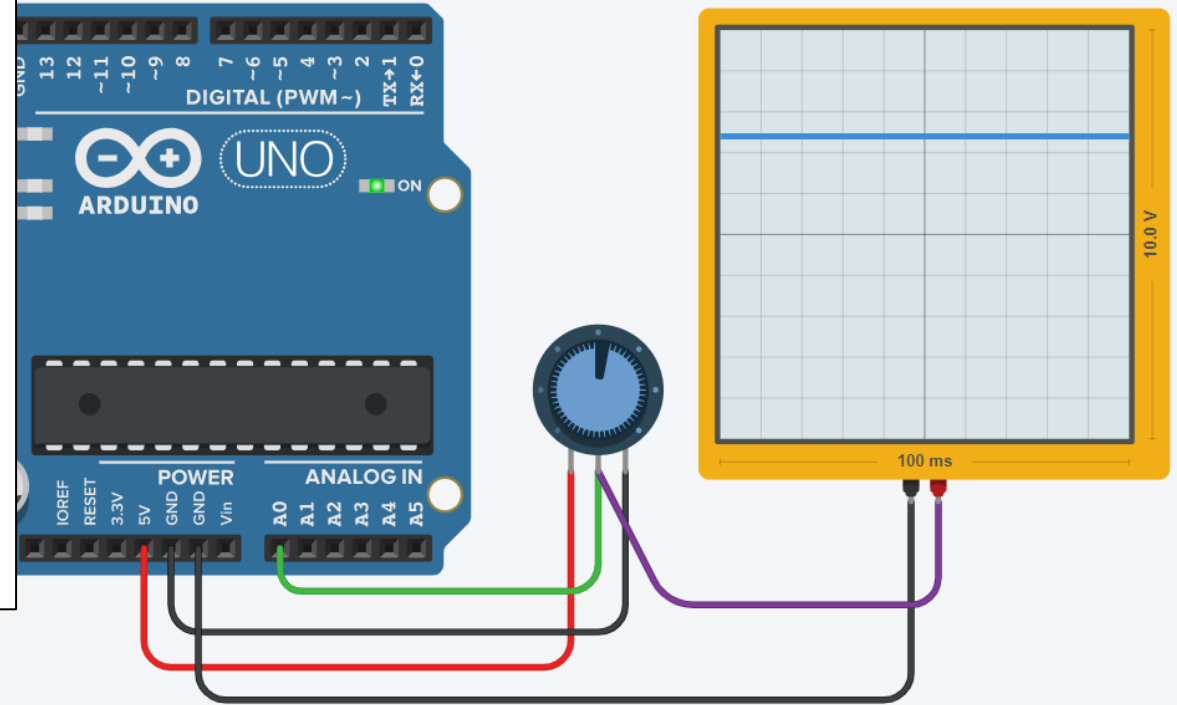
```
void setup ()
{
  Serial.begin(9600);
}

void loop()
{
  int val = analogRead(A0);
  Serial.print("Analog : ");
  Serial.println(val);
}
```



가변저항(Potentiometer, 볼륨)

```
void setup ()  
{  
  Serial.begin(9600);  
}  
  
void loop()  
{  
  int val = analogRead(A0);  
  Serial.print("Analog : ");  
  Serial.println(val);  
}
```



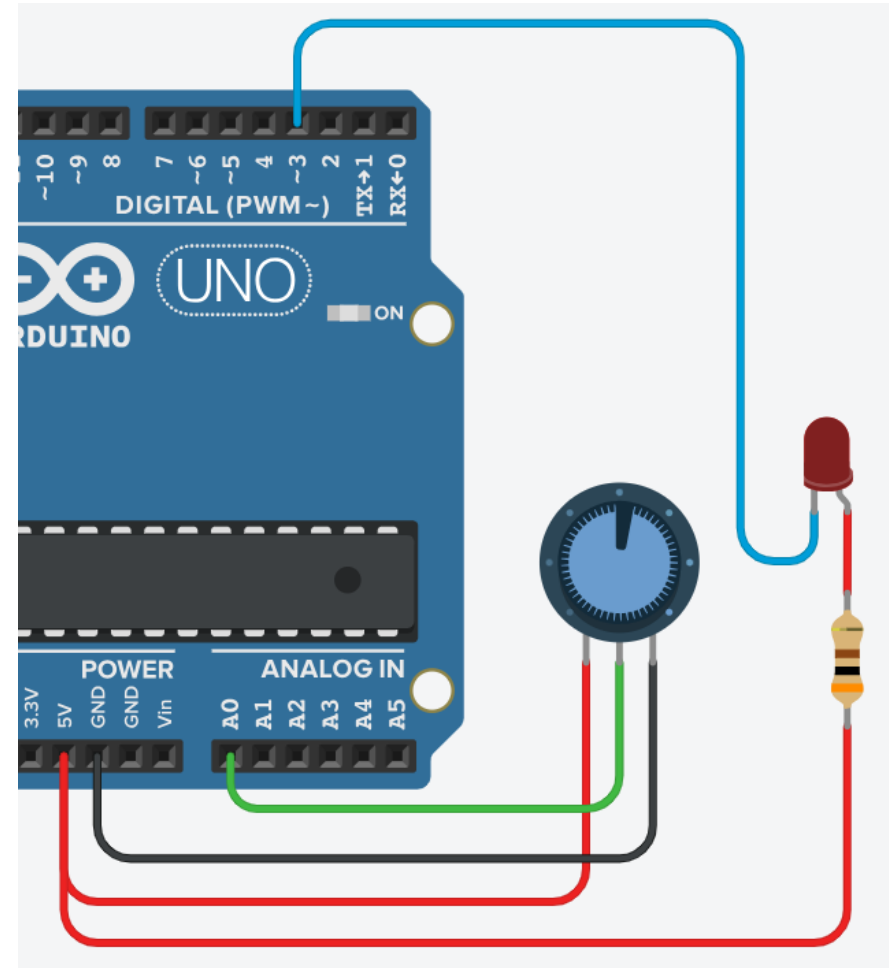
analogWrite / analogRead 실험

```
void setup ()
{
  Serial.begin(9600);
  pinMode(3, OUTPUT);
}

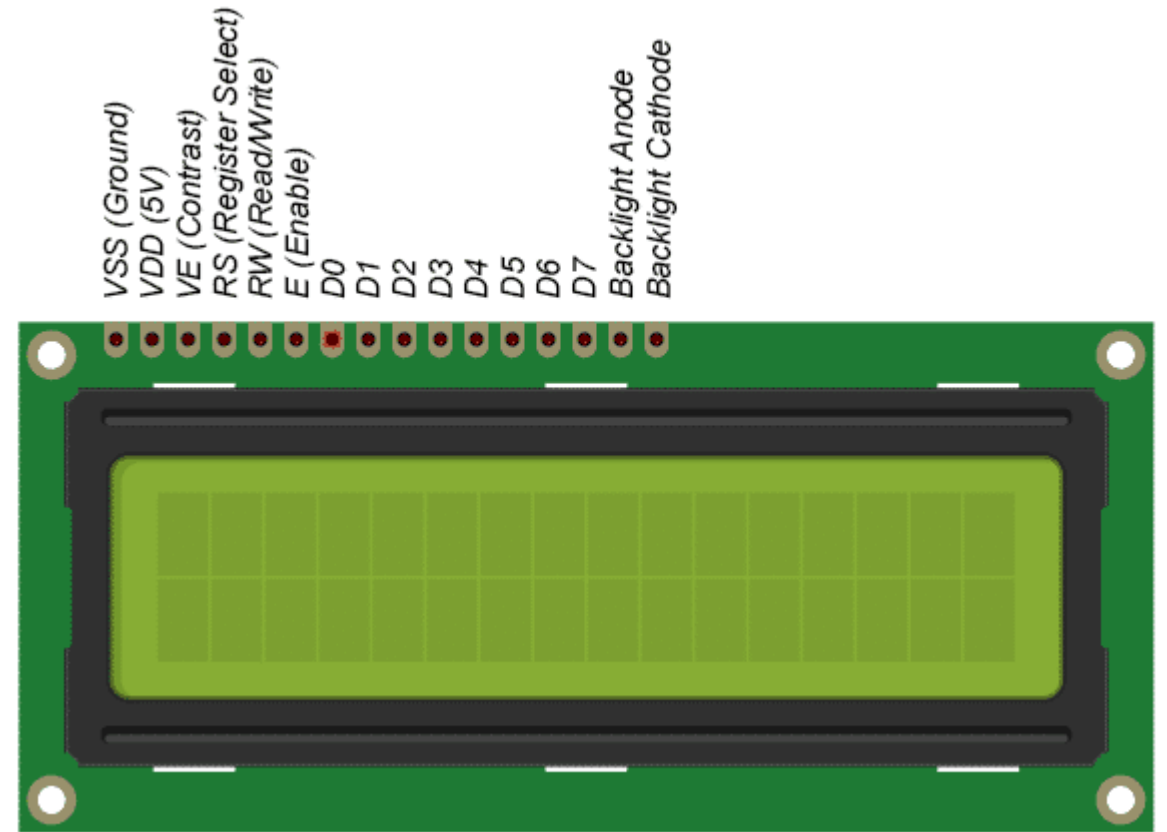
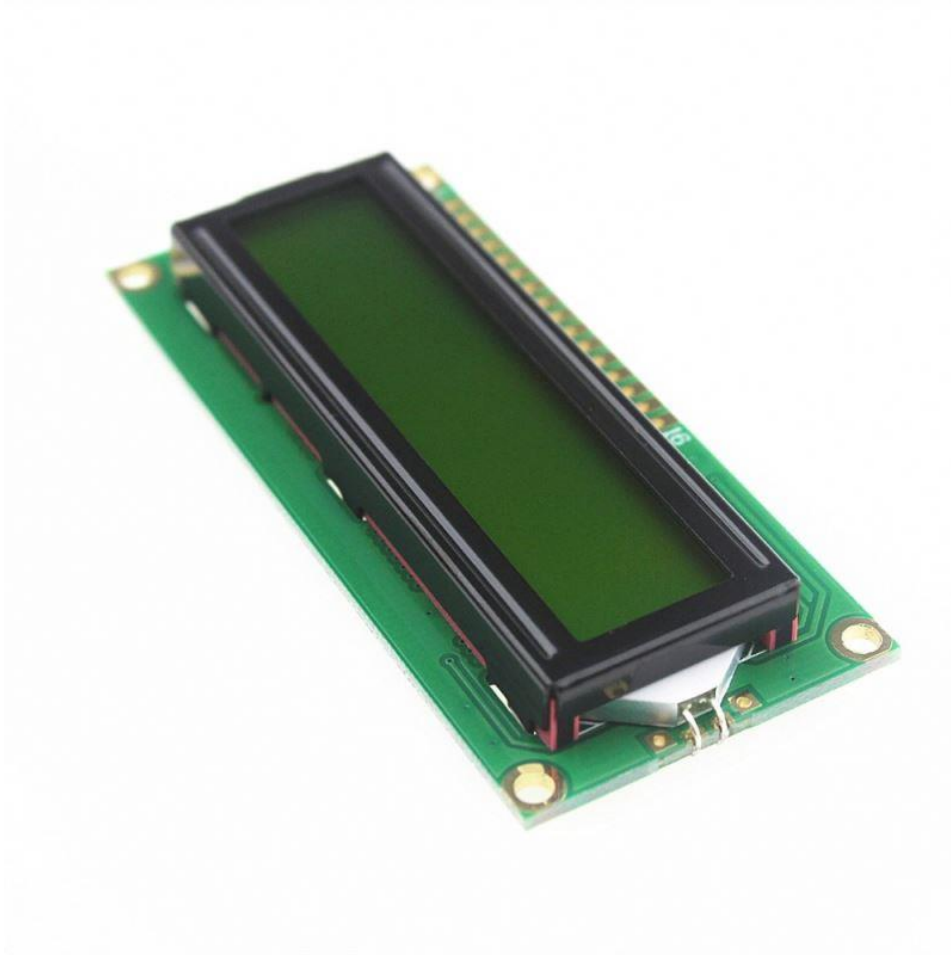
void loop()
{
  int val = analogRead(A0);
  Serial.print("Analog : ");
  Serial.println(val);

  val = val / 4 ;

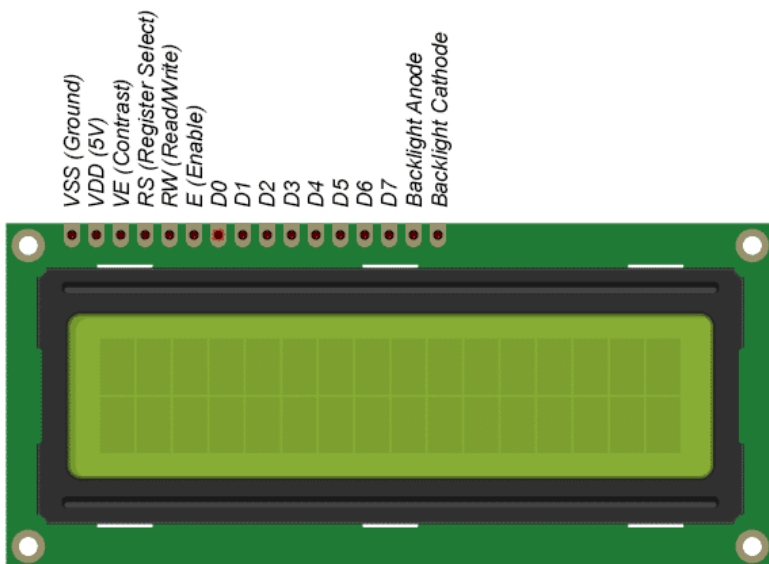
  analogWrite(3, val);
}
```



16x2 Character LCD

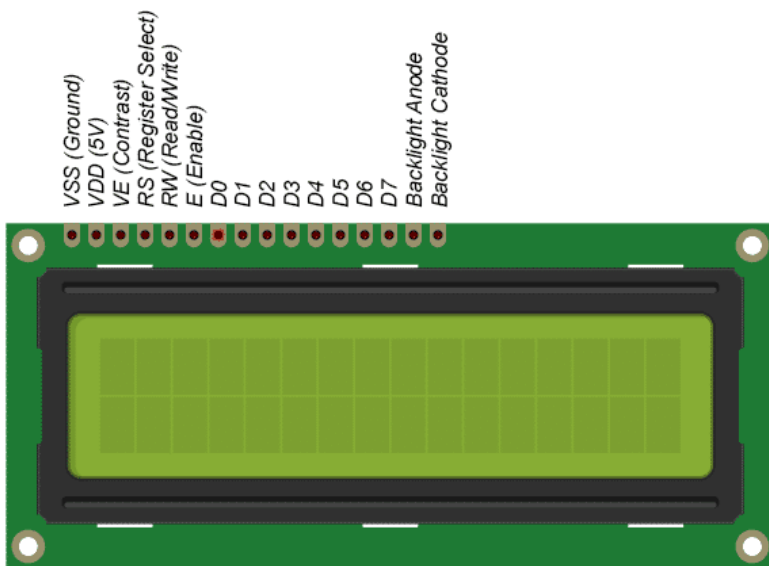


16x2 Character LCD



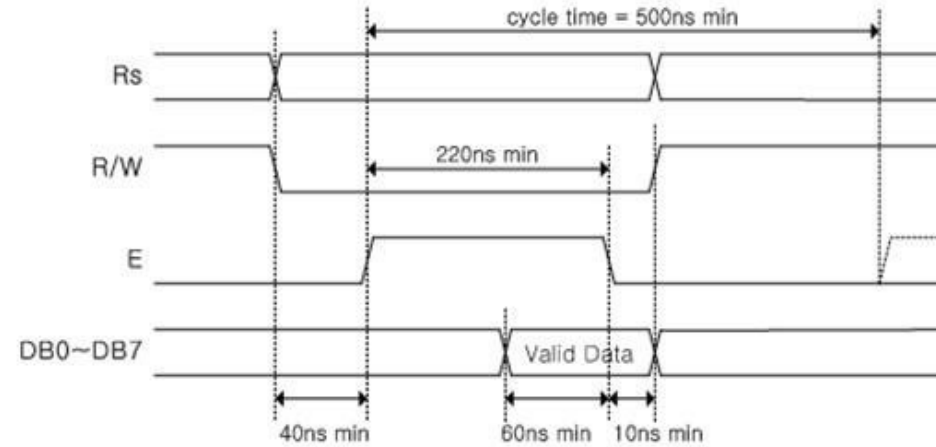
- **VSS(Ground)** : LCD에 전원을 인가하는 단자로, 0V(GND)에 연결.
- **VDD(5V)** : LCD에 전원을 인가하는 단자로, +5V에 연결.
- **VE(Contrast)** : LCD의 밝기를 조절하는 단자로서, 10kΩ의 가변 저항을 연결하여 밝기를 조정. 밝기 조절을 하지 않으려면 GND에 연결.
- **RS(Register Select)** : 레지스터 종류를 선택
 - 0 : 명령 레지스터
 - 1 : 데이터 레지스터
- **RW(Read/Write)** : 데이터 혹은 명령을 읽는지 쓰는지 설정
 - 0 : 레지스터의 데이터를 씀. (Write : 아두이노 → LCD)
 - 1 : 레지스터에 데이터를 읽음. (Read : 아두이노 ← LCD)
- **E(Enable)** : Enable 신호(LCD동작 허가)
 - 0 : LCD 동작 X
 - 1 : LCD 동작 O
- **D0~D7(Data Bus)** : 아두이노와 LCD 사이에 데이터를 주고받기 위한 데이터 핀. 만약 4비트를 사용할 경우에는 D4~D7만 사용.
- **Backlight Anode** : 백 라이트의 전원 단자로서 보통 저항과 IN4001(다이오드)을 연결
- **Backlight Cathode** : 백 라이트의 전원 단자로서 GND에 연결.

16x2 Character LCD

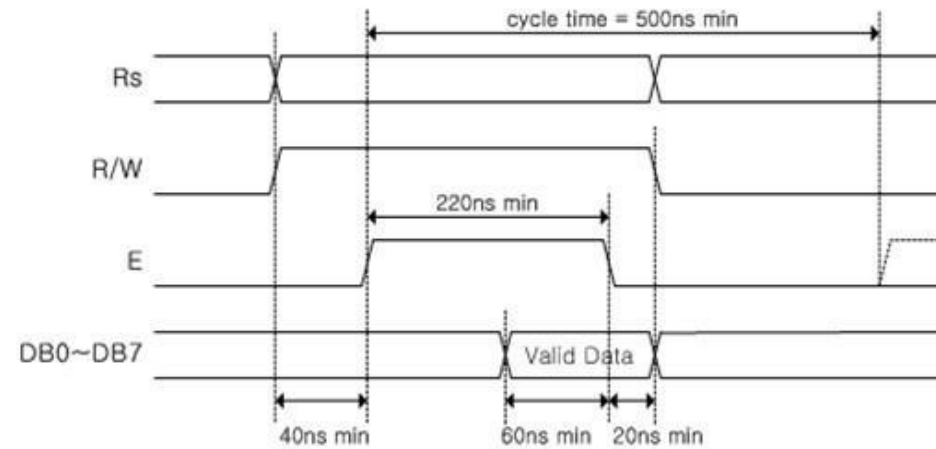


- **2개의 8-bit 레지스터**
 - RS(Register Select) 신호로 어떤 레지스터를 사용할 것인지 선택
 - **명령 레지스터** (Instruction Register, IR)
 - D.D.RAM, C.G.RAM에 대한 주소정보, 클리어, 커서 이동등 LCD 제어 명령
 - **데이터 레지스터** (Data Register, DR)
 - D.D.RAM, C.G.RAM에 써넣은 데이터나 읽어낸 데이터를 일시적으로 저장
- **D.D.RAM(Display Data RAM)**
 - 80x8비트 용량으로 80개의 8비트 아스키(ASCII)코드를 저장할 수 있다.
 - **0x00~0F** 주소가 LCD의 **1행의 1~16째**.
 - **0x40~4f** 주소가 LCD의 **2행의 1~16번째 문자로 표시 된다**.
 - 빈 주소에는 자유롭게 RAM 데이터 메모리로 사용이 가능하다.
- **C.G.RAM(Charactor Generator RAM)**
 - 사용자가 문자를 만들 때 사용하는 메모리로 5x7은 8개, 5x10은 4개를 만들어서 저장이 가능.
- **C.G.ROM(Charactor Generator ROM)** : 5x7, 5x10의 도트문자를 내장하고 있음. 아스키코드와 일치
- **Address Counter (AC)**
 - 주소를 저장하는 레지스터로 D.D.RAM과 C.G.RAM의 주소를 지정할 때 사용, 명령레지스터에 주소정보를 지정하면 AC로 주소정보가 전달 되고, D.D.RAM에 데이터를 쓰면 AC는 자동으로 (설정 값에 따라) 1증가하거나 1감소한다.
- **Busy Flag (BF)**
 - Busy flag가 '1' → 내부에서 LCD가 명령을 처리 중으로 명령을 받을 수 없음.
 - Busy flag가 '0' → 명령 가능.
 - Busy flag를 확인 하려면? RS=0, R/W=1일 때, D7핀으로 출력

16x2 Character LCD



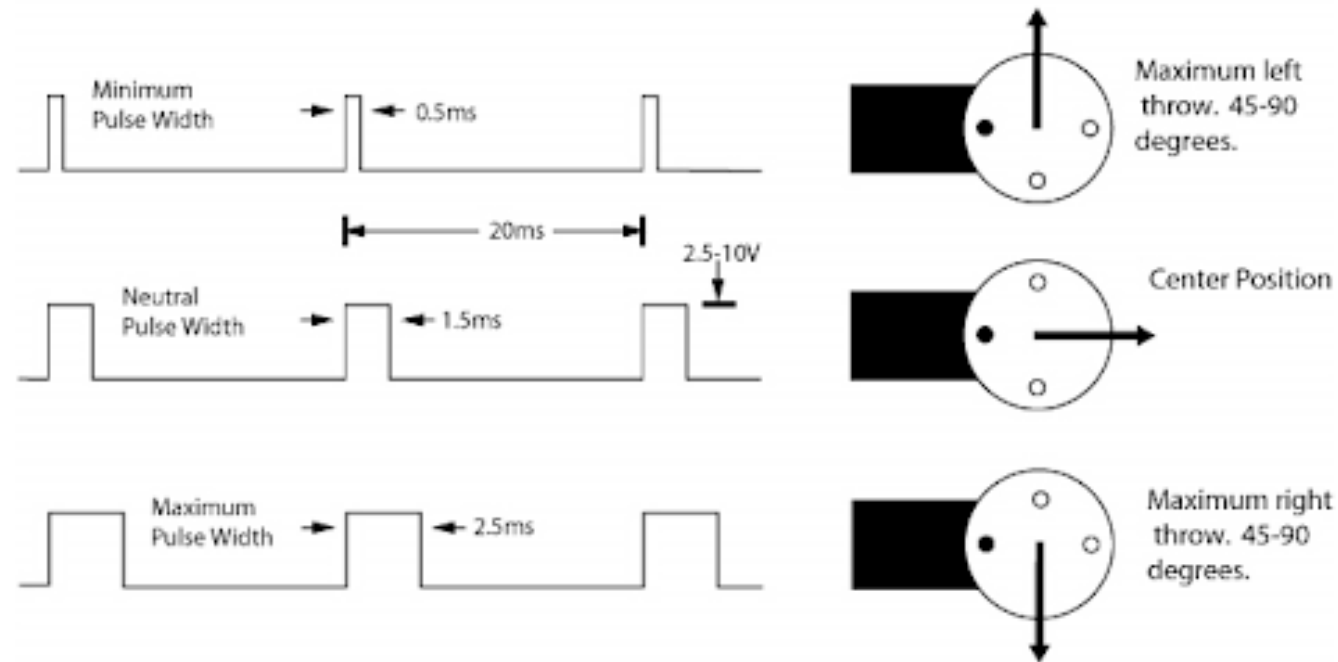
(a) Write from to LCD



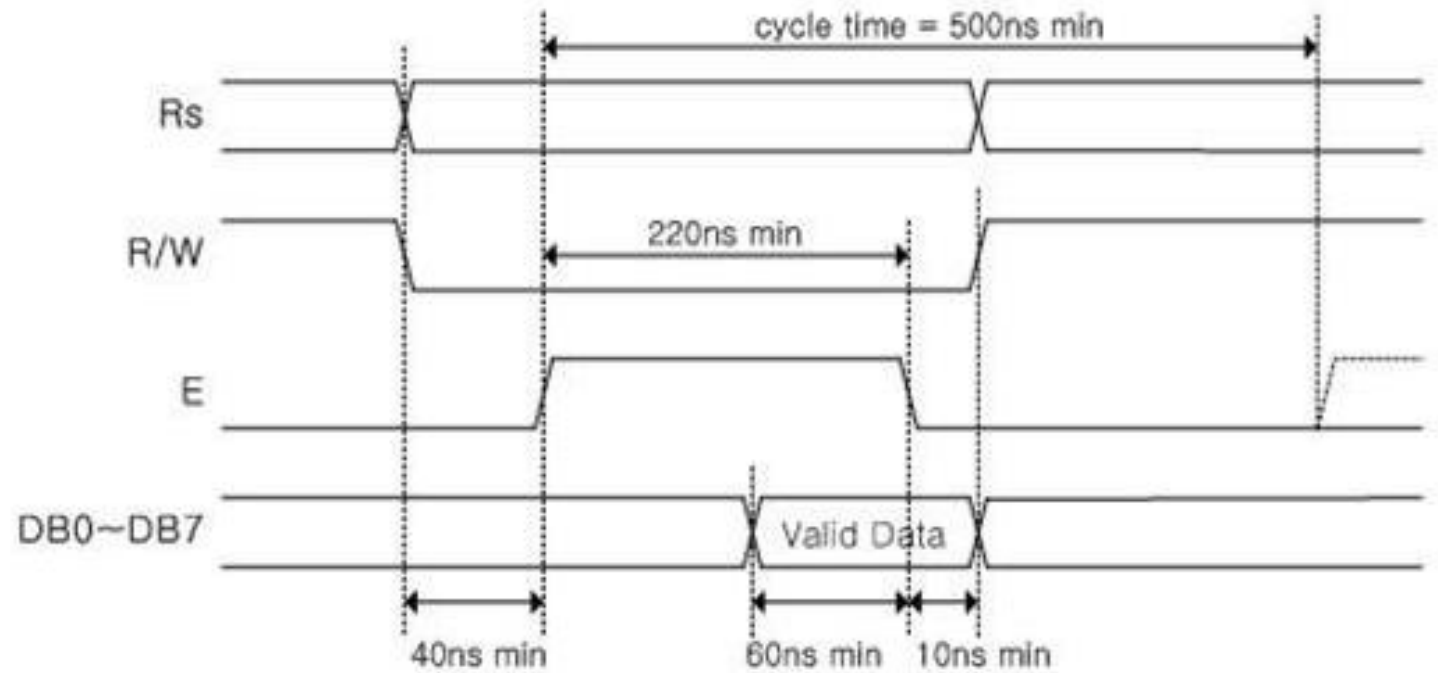
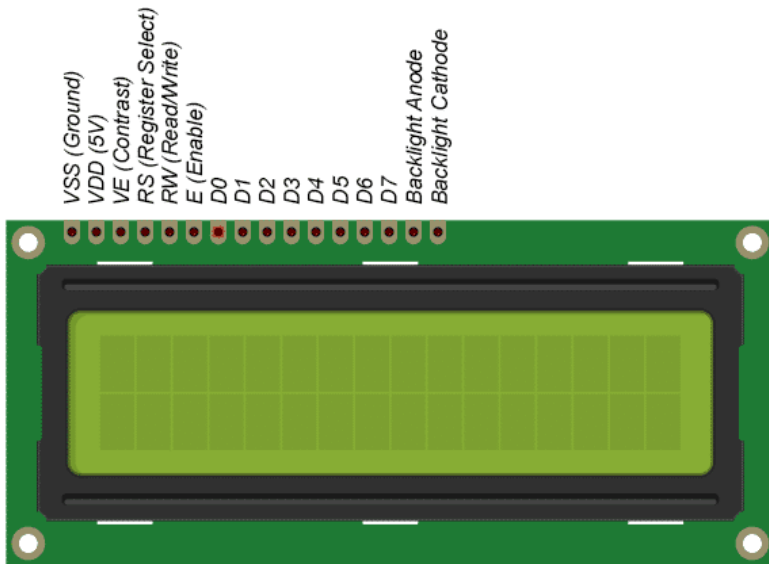
(b) Read from LCD

16x2 Character LCD

R/C Control Signal Theory

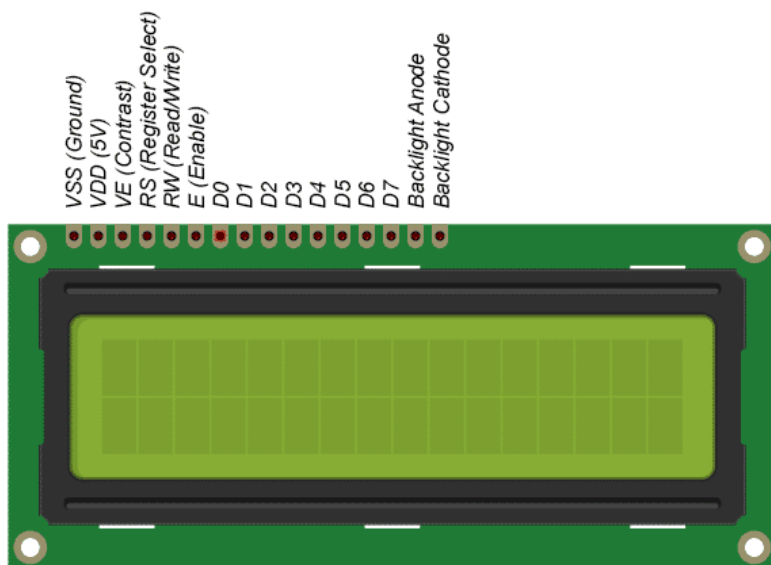


16x2 Character LCD



(a) Write from to LCD

16x2 Character LCD



0x38 = **001**(기능셋)1 1000

명령	명령	데이터										설명	실행 시간
		RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
명령 쓰기	화면지우기	0	0	0	0	0	0	0	0	1		화면지우기, 커서홈	1,52ms
	커서홈	0	0	0	0	0	0	0	0	1	-	커서 처음 위치로 이동	1,52ms
	엔트리 모드 셋	0	0	0	0	0	0	0	1	I/D	S	어드레스자동증가/감소(I/D) 표시 쉬프트(S)	37us
	표시 On/Off 제어	0	0	0	0	0	0	1	D	C	B	디스플레이(D), 커서(C), 깜박임(B) On/Off	37us
	표시, 커서 쉬프트	0	0	0	0	0	1	S/C	R/L	-	-	표시, 커서 이동	37us
	기능 셋	0	0	0	0	1	DL	N	F	-	-	인터페이스라인(DL), 라인수(N), 문자폰트(F)	37us
	CGRAM 어드레스	0	0	0	1	CGRAM 어드레스(ACG)						CGRAM 어드레스 설정	37us
	DDRAM 어드레스	0	0	1	DDRAM 어드레스(ADD)							DDRAM 어드레스 설정	37us
명령 읽기	비지체크, 어드레스	0	1	BF	어드레스 카운터(AC)							비지플래그 읽기 어드레스 카운터 읽기	0us
데이터 쓰기	데이터 쓰기	1	0	write data								CGRAM 또는 DDRAM에 데이터 쓰기	37us
데이터 읽기	데이터 읽기	1	1	read data								CGRAM 또는 DDRAM에서 데이터 읽기	37us

I/D=1 : 어드레스 자동증가

I/D=0 : 어드레스 자동감소

DDRAM : 표시 데이터 RAM

S=1 : 전체 쉬프트

S=0 : 쉬프트 하지 않음

CGRAM : 폰트 제작 RAM

S/C=1 : 표시 쉬프트

S/C=0 : 커서 이동

ACG : CGRAM 어드레스

R/L=1 : 오른쪽으로 쉬프트

R/L=0 : 왼쪽으로 쉬프트

ADD : DDRAM 어드레스

DL=1 : 8비트

DL=0 : 4비트

AC : 어드레스 카운터

N=1 : 2라인

N=0 : 1라인

(DDRAM, CGRAM 어드레스)

F=1 : 5x10 dots

F=0 : 5x8 dot

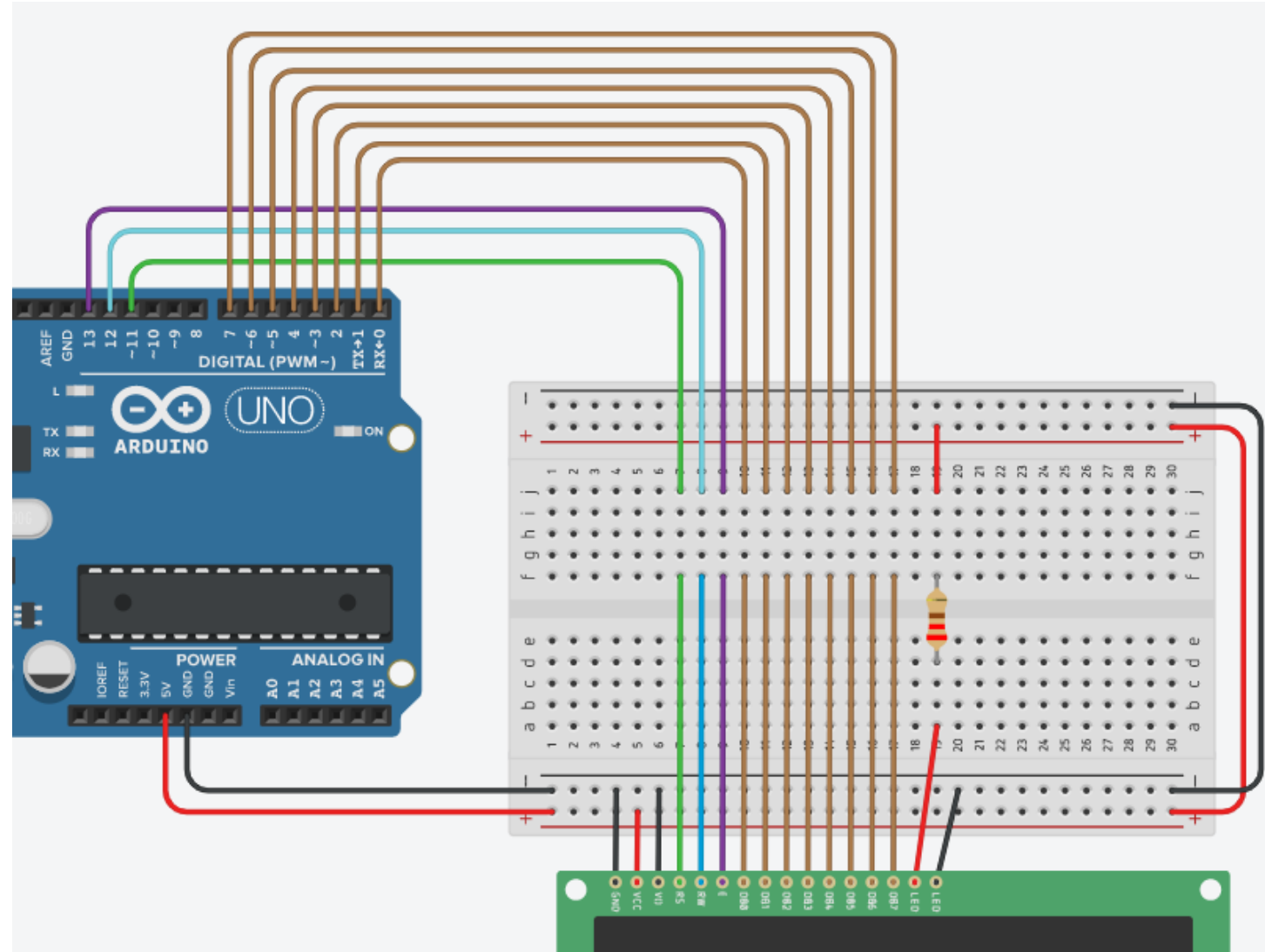
BF=1 : 내부 동작중

BF=0 : 명령/데이터 받기 가능

16x2 Character LCD 실험

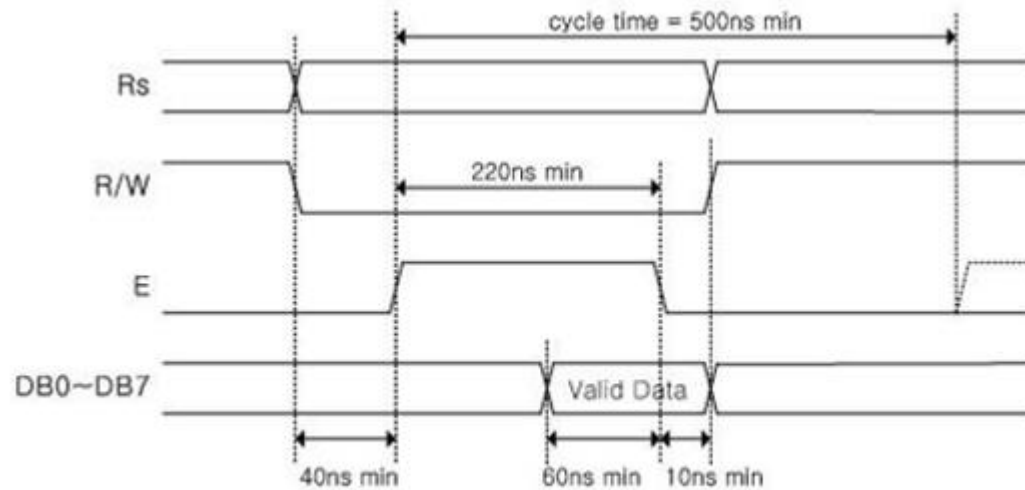
- 회로 구성

- RS : 11번핀(PORTB3)
- RW : 12번핀(PORTB4)
- E : 13번핀(PORTB5)
- D0~D7 : 0번핀~7번핀(PORTD)



16x2 Character LCD 실험

- LCD 초기화(명령 Write to LCD)



(a) Write from to LCD

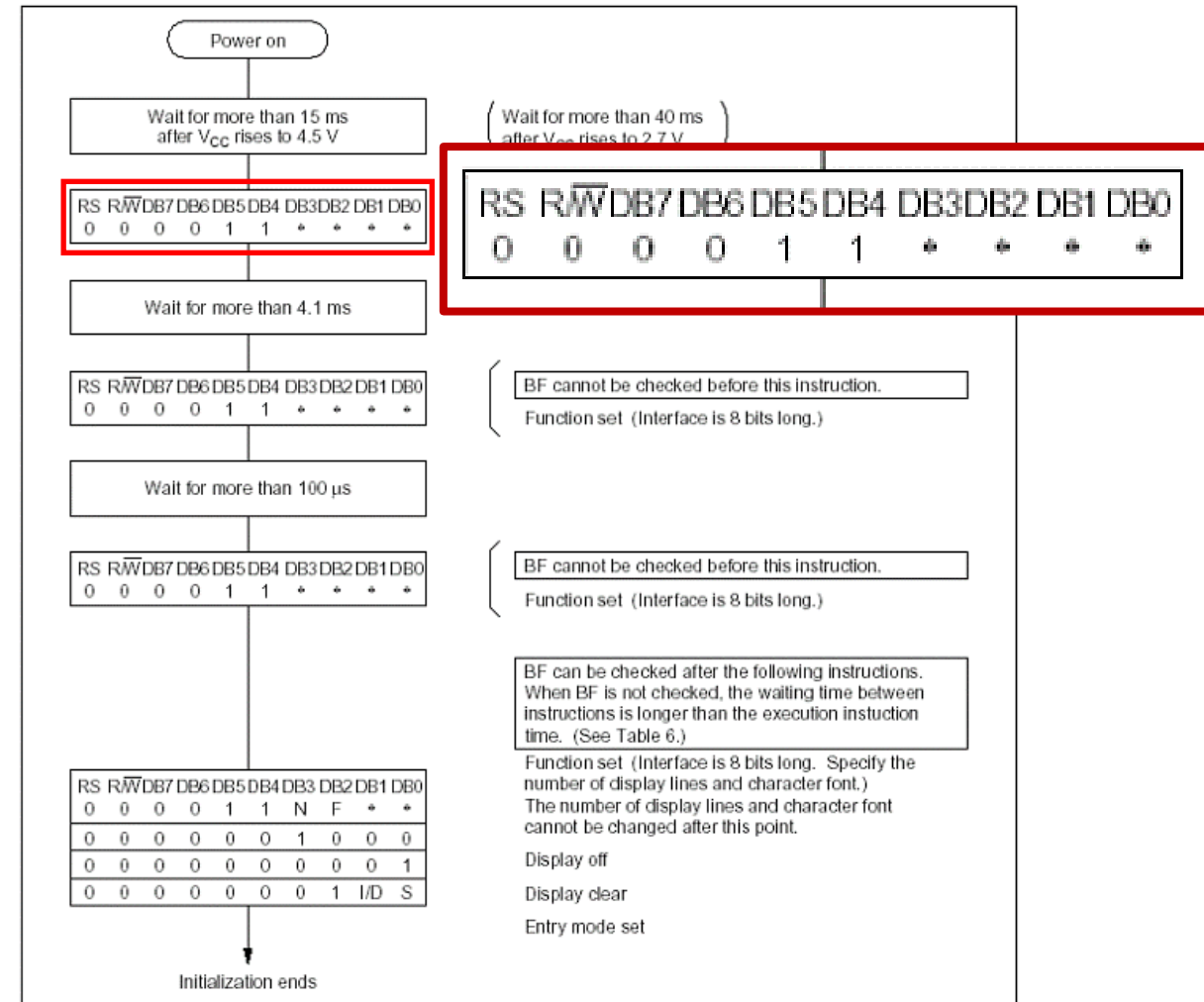
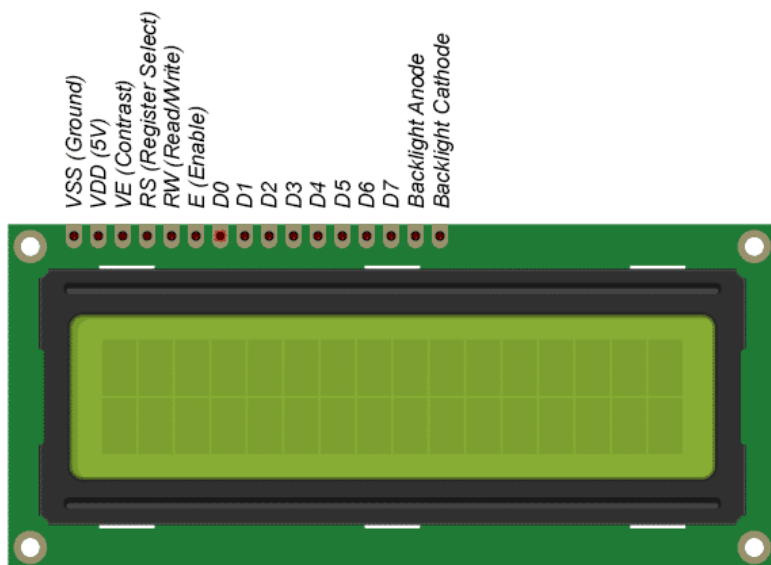


Figure 23 8-Bit Interface

16x2 Character LCD



0x38 = **001**(기능셋)1 1000

명령	명령	데이터										설명	실행 시간
		RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
명령 쓰기	화면지우기	0	0	0	0	0	0	0	0	1		화면지우기, 커서홈	1,52ms
	커서홈	0	0	0	0	0	0	0	0	1	-	커서 처음 위치로 이동	1,52ms
	엔트리 모드 셋	0	0	0	0	0	0	0	1	I/D	S	어드레스자동증가/감소(I/D) 표시 쉬프트(S)	37us
	표시 On/Off 제어	0	0	0	0	0	0	1	D	C	B	디스플레이(D), 커서(C), 깜박임(B) On/Off	37us
	표시, 커서 쉬프트	0	0	0	0	0	1	S/C	R/L	-	-	표시, 커서 이동	37us
	기능 셋	0	0	0	0	1	DL	N	F	-	-	인터페이스라인(DL), 라인수(N), 문자폰트(F)	37us
	CGRAM 어드레스	0	0	0	1	CGRAM 어드레스(ACG)						CGRAM 어드레스 설정	37us
	DDRAM 어드레스	0	0	1	DDRAM 어드레스(ADD)							DDRAM 어드레스 설정	37us
명령 읽기	비지 체크, 어드레스	0	1	BF	어드레스 카운터(AC)							비지플래그 읽기 어드레스 카운터 읽기	0us
데이터 쓰기	데이터 쓰기	1	0	write data								CGRAM 또는 DDRAM에 데이터 쓰기	37us
데이터 읽기	데이터 읽기	1	1	read data								CGRAM 또는 DDRAM에서 데이터 읽기	37us

I/D=1 : 어드레스 자동증가

I/D=0 : 어드레스 자동감소

DDRAM : 표시 데이터 RAM

S=1 : 전체 쉬프트

S=0 : 쉬프트 하지 않음

CGRAM : 폰트 제작 RAM

S/C=1 : 표시 쉬프트

S/C=0 : 커서 이동

ACG : CGRAM 어드레스

R/L=1 : 오른쪽으로 쉬프트

R/L=0 : 왼쪽으로 쉬프트

ADD : DDRAM 어드레스

DL=1 : 8비트

DL=0 : 4비트

AC : 어드레스 카운터

N=1 : 2라인

N=0 : 1라인

(DDRAM, CGRAM 어드레스)

F=1 : 5x10 dots

F=0 : 5x8 dot

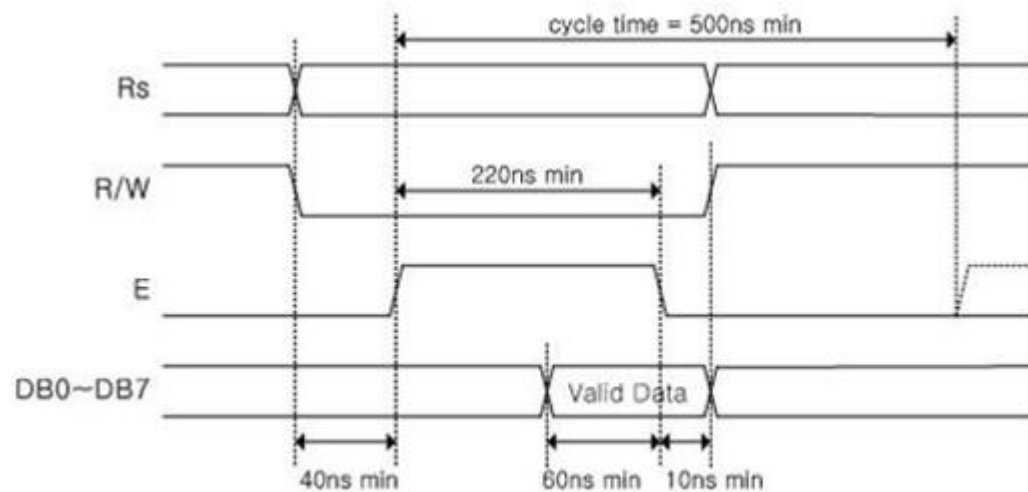
BF=1 : 내부 동작중

BF=0 : 명령/데이터 받기 가능

16x2 Character LCD 실험

• LCD 초기화

RS(Register Select) : 0 → 명령 레지스터



(a) Write from to LCD

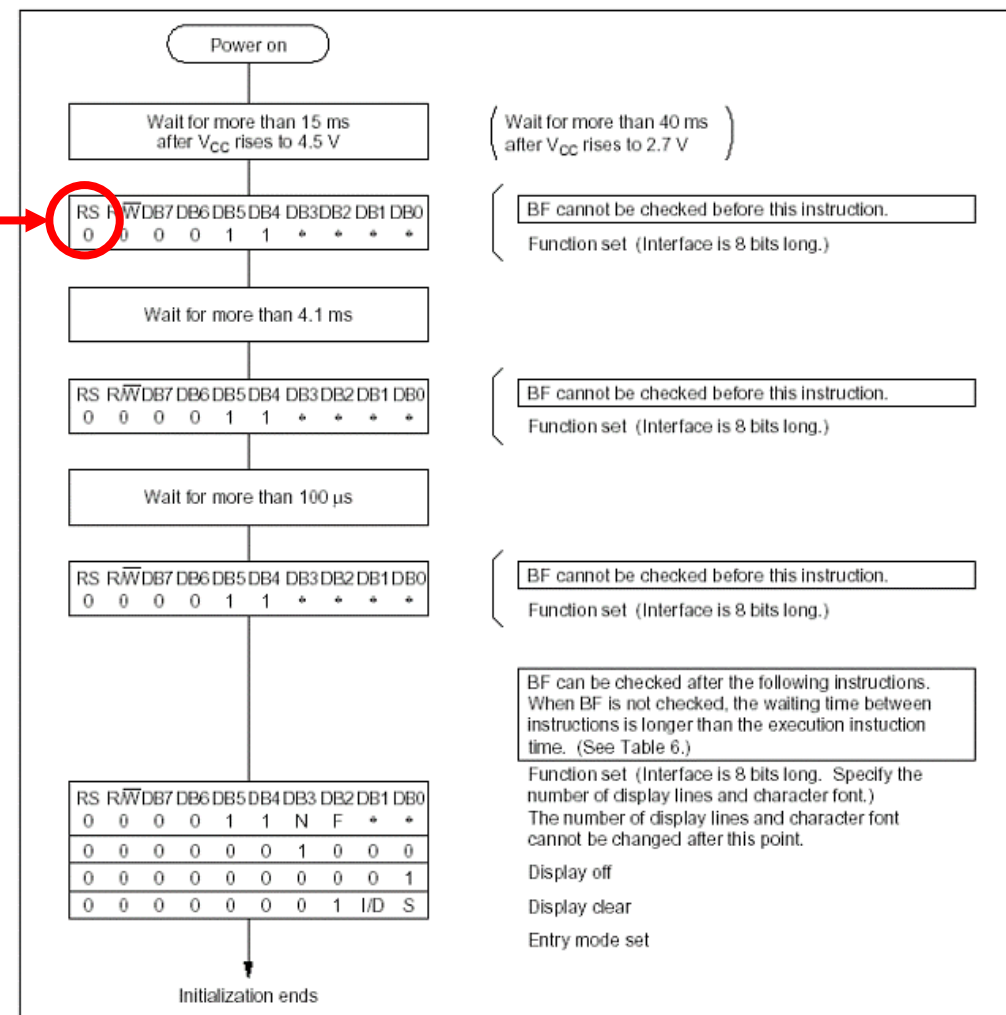
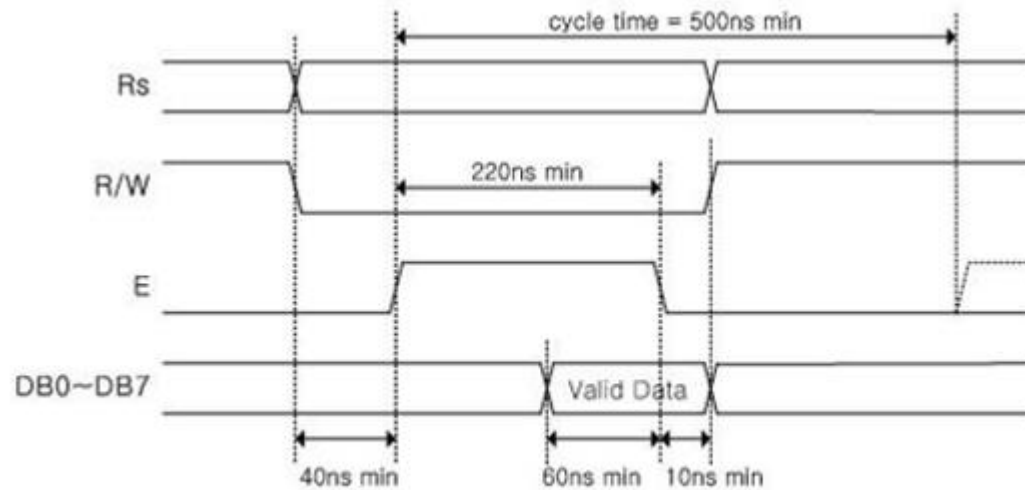


Figure 23 8-Bit Interface

16x2 Character LCD 실험

• LCD 초기화

RW(Read/Write) : 0 → Write



(a) Write from to LCD

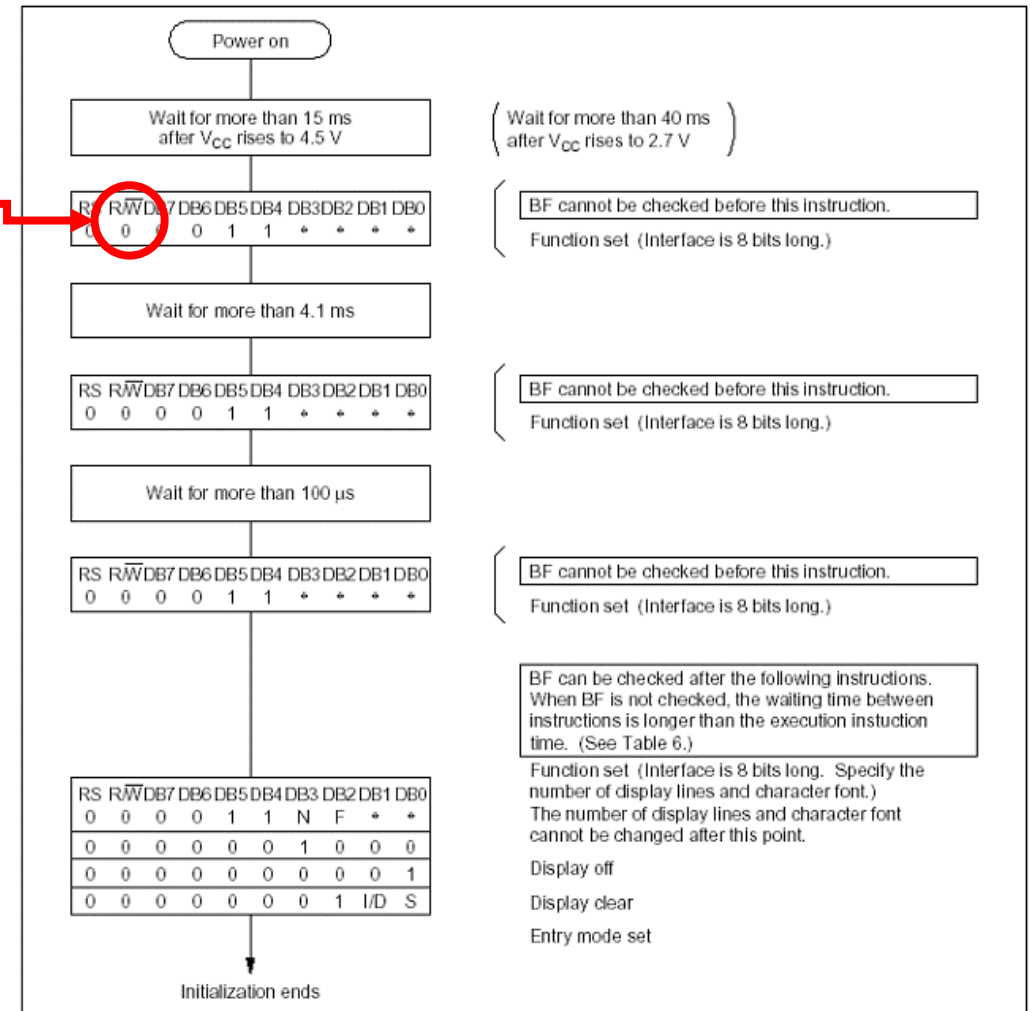
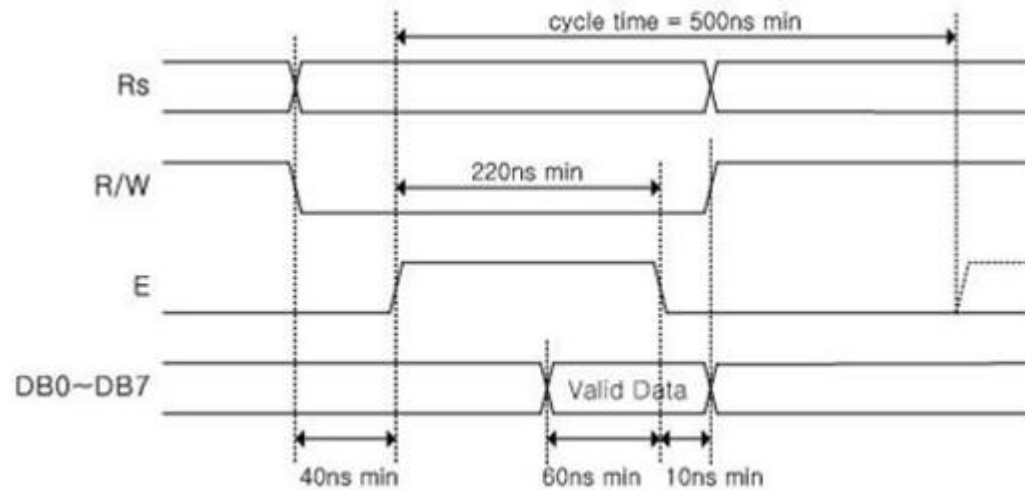


Figure 23 8-Bit Interface

16x2 Character LCD 실험

• LCD 초기화

DB5 bit가 1이면 "기능셋" 설정



(a) Write from to LCD

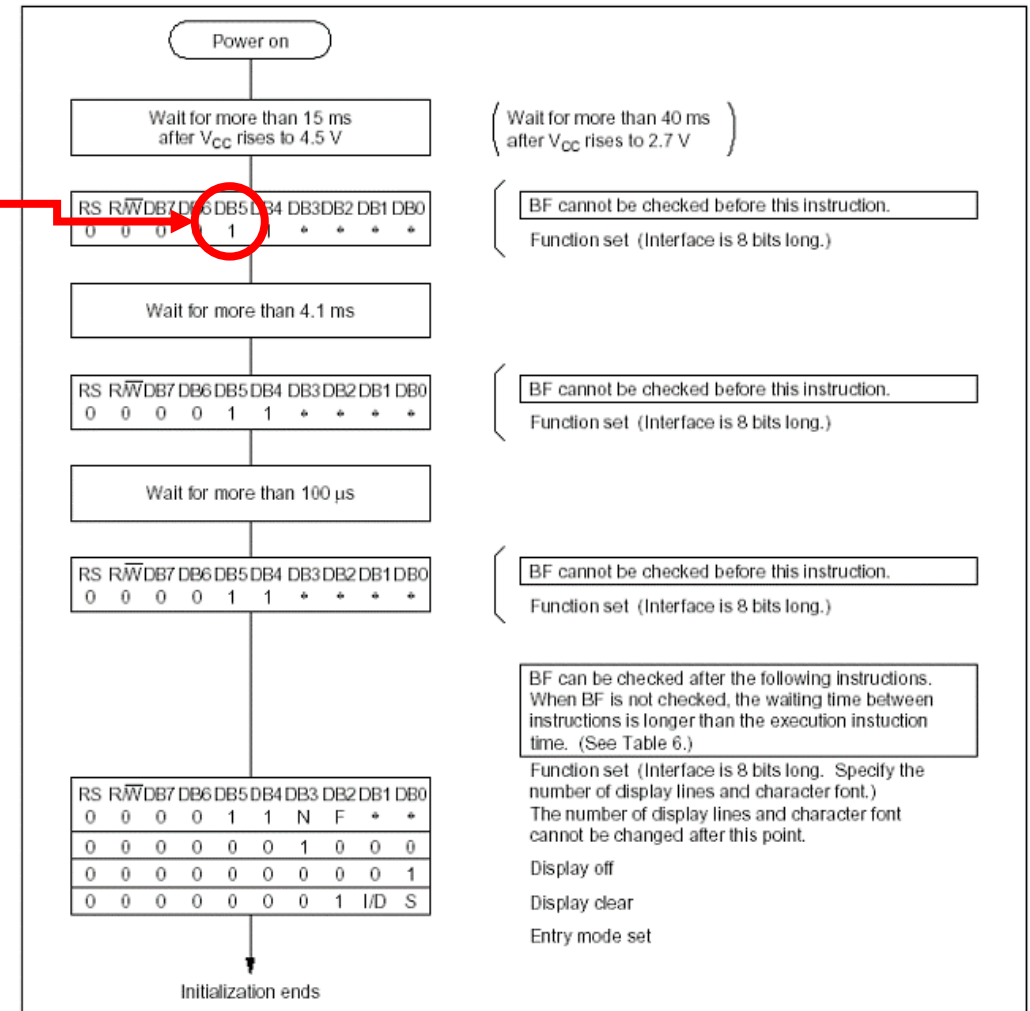
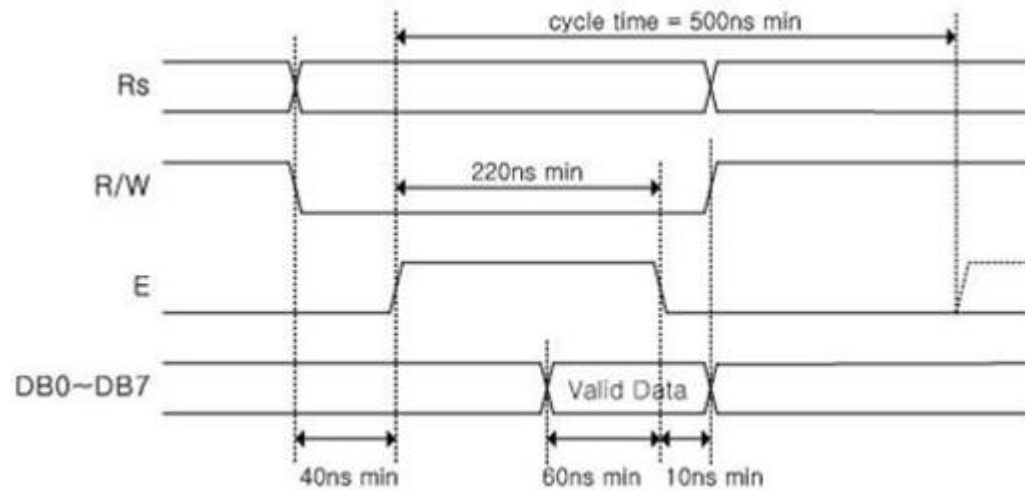


Figure 23 8-Bit Interface

16x2 Character LCD 실험

• LCD 초기화

DL(DB4 bit) : Data Line : 1 → 8Bit



(a) Write from to LCD

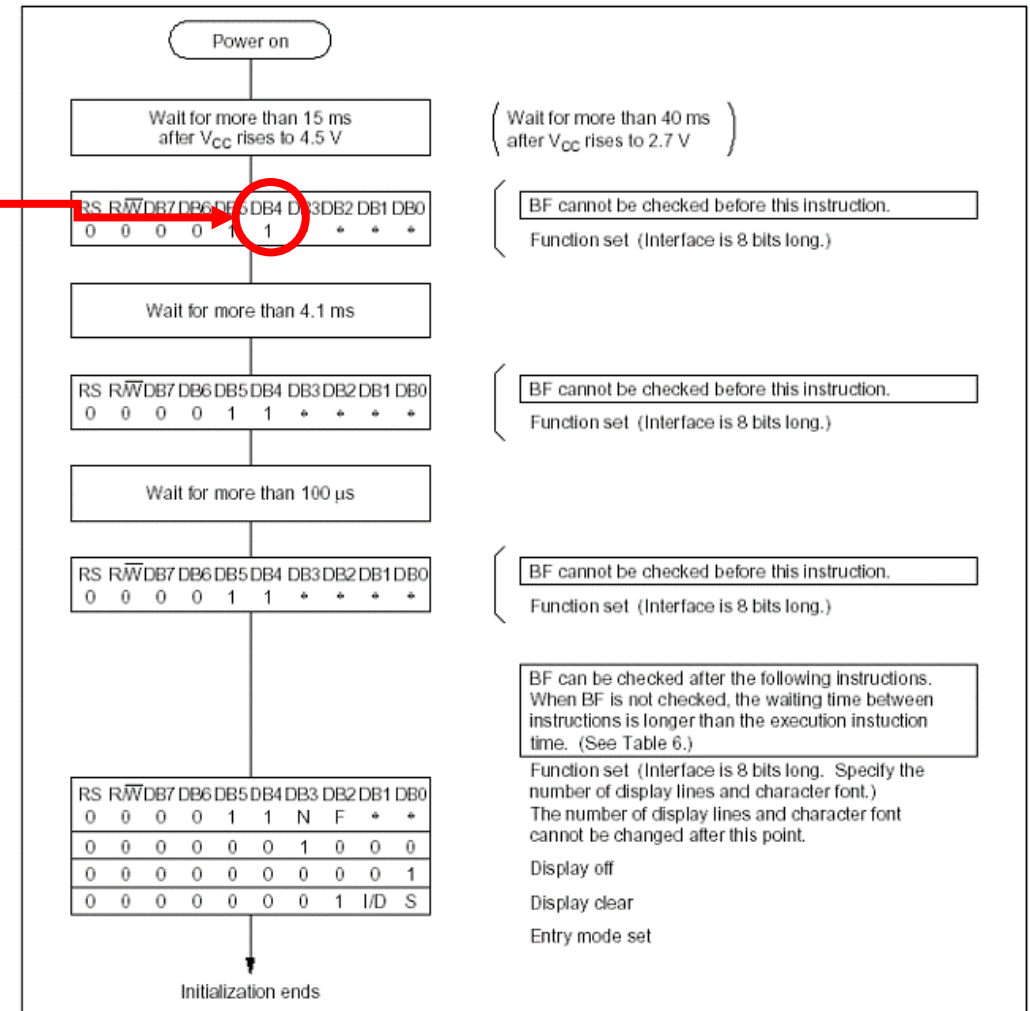


Figure 23 8-Bit Interface

16x2 Character LCD 실험

• LCD 초기화

명령	명령	데이터										설명	실행 시간
		RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
명령 쓰기	화면지우기	0	0	0	0	0	0	0	0	0	1	화면지우기, 커서줄	1.52ms
	커서줄	0	0	0	0	0	0	0	0	1	-	커서 처음 위치로 이동	1.52ms
	엔트리 모드 셋	0	0	0	0	0	0	0	1	I/D	S	어드레스 자동증가/감소(I/D) 표시 쉬프트(S)	37us
	표시 On/Off 제어	0	0	0	0	0	0	1	D	C	B	디스플레이(D), 커서(C), 깜박임(B) On/Off	37us
	표시, 커서 쉬프트	0	0	0	0	0	1	S/C	R/L	-	-	표시, 커서 이동	37us
	기능 셋	0	0	0	0	1	DL	N	F	-	-	인터페이스라인(DL), 라인수(N), 문자폰트(F)	37us
	CGRAM 어드레스	0	0	0	1	CGRAM 어드레스(ACG)						CGRAM 어드레스 설정	37us
	DDRAM 어드레스	0	0	1		DDRAM 어드레스(ADD)						DDRAM 어드레스 설정	37us
명령 읽기	비지 체크, 어드레스	0	1	BF	어드레스 카운터(AC)							비지 플래그 읽기 어드레스 카운터 읽기	0us
데이터 쓰기	데이터 쓰기	1	0	write data								CGRAM 또는 DDRAM에 데이터 쓰기	37us
데이터 읽기	데이터 읽기	1	1	read data								CGRAM 또는 DDRAM에서 데이터 읽기	37us

I/D=1 : 어드레스 자동증가
S=1 : 전체 쉬프트
S/C=1 : 표시 쉬프트
R/L=1 : 오른쪽으로 쉬프트
DL=1 : 8비트
N=1 : 2라인
F=1 : 5x10 dots
BF=1 : 내부 동작중

I/D=0 : 어드레스 자동감소
S=0 : 쉬프트 하지 않음
S/C=0 : 커서 이동
R/L=0 : 왼쪽으로 쉬프트
DL=0 : 4비트
N=0 : 1라인
F=0 : 5x8 dot
BF=0 : 명령/데이터 받기 가능

DDRAM : 표시 데이터 RAM
CGRAM : 폰트 제작 RAM
ACG : CGRAM 어드레스
ADD : DDRAM 어드레스
AC : 어드레스 카운터
(DDRAM, CGRAM 어드레스)

0x30 (0011 xxxx) : 초기화

0x38 (0011 1000) : 기능셋

**0x08 (0000 1000)
표시On/Off 제어**

**0x01 (0000 0001)
화면지우기**

**0x04 (0000 0100)
엔트리모드셋**

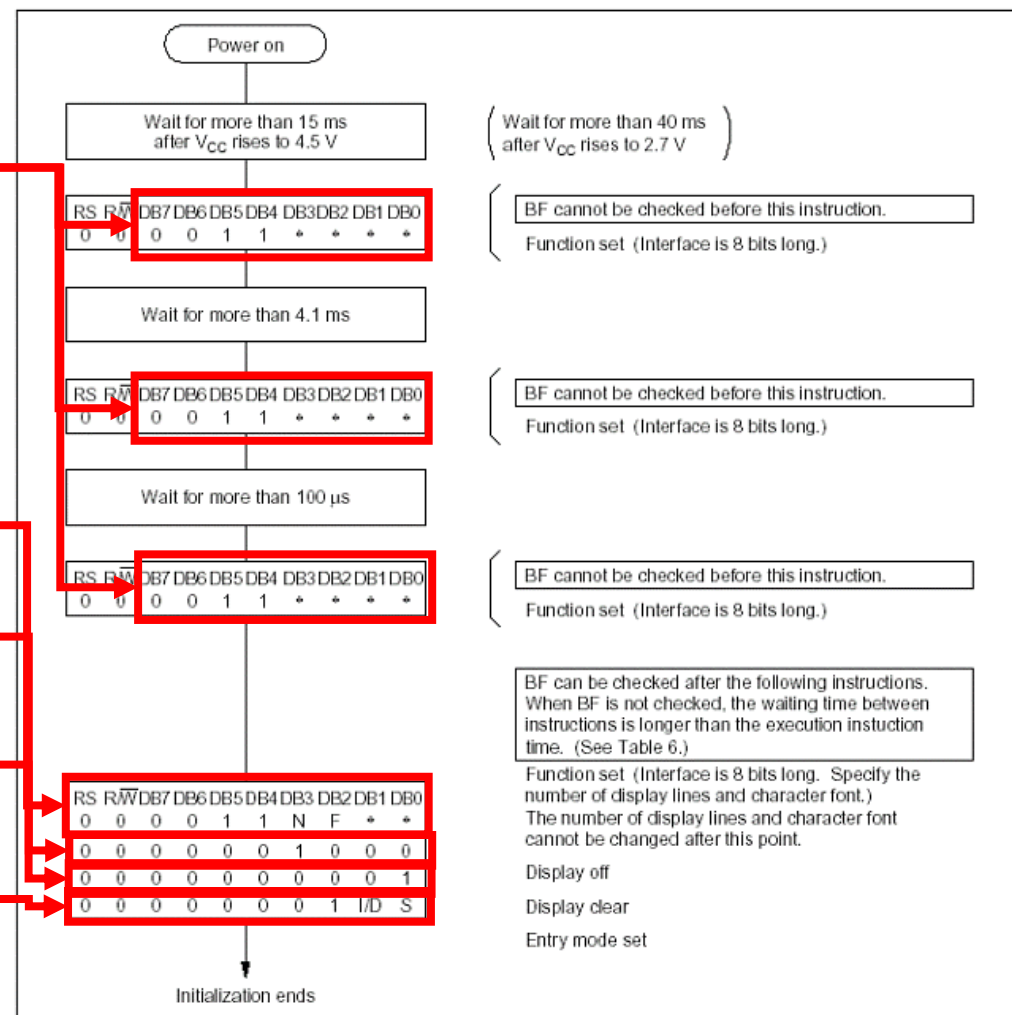
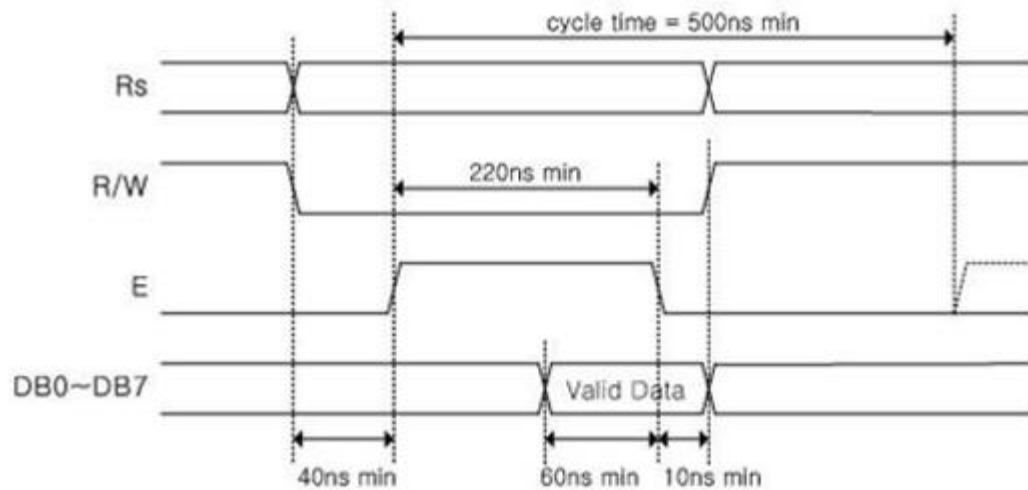


Figure 23 8-Bit Interface

16x2 Character LCD 실험 – example_25

- LCD 초기화 코드 작성



(a) Write from to LCD



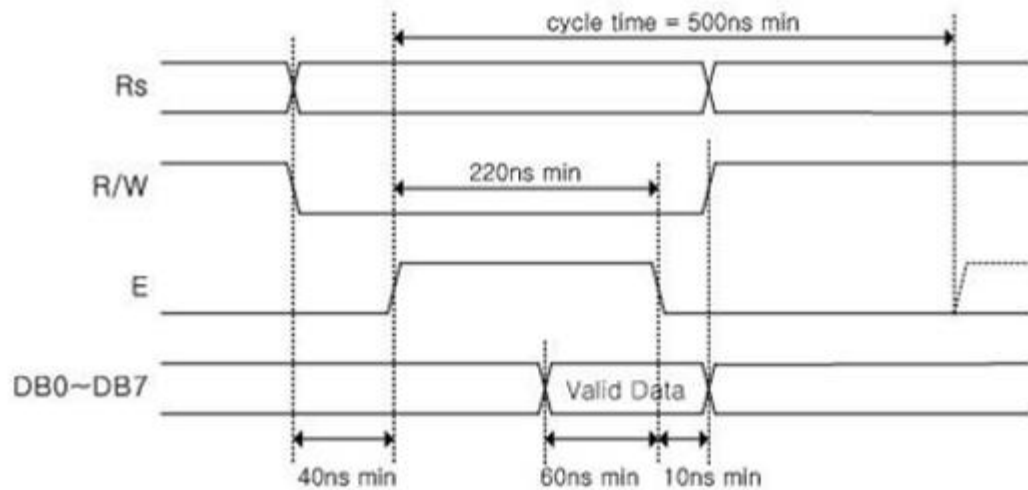
```
void setup()
{
    //포트 방향 설정
    pinMode(11, OUTPUT);
    pinMode(12, OUTPUT);
    pinMode(13, OUTPUT);
    DDRD = B11111111;

    //명령 1개 전달
    digitalWrite(11, LOW); // RS = 0, 명령
    digitalWrite(12, LOW); // RW = 0, 쓰기
    digitalWrite(13, HIGH); // E = 1
    PORTD = 0x38; // 데이터 출력
    delayMicroseconds(1);
    digitalWrite(13, LOW); // 데이터 쓰기 동작 끝
    delayMicroseconds(1);
}

void loop()
{
}
```

16x2 Character LCD 실험 – example_26

- LCD 초기화 코드 작성



(a) Write from to LCD

0x38 = 0011 1000

```
#define RS      11
#define RW      12
#define EN      13
```

```
void setup()
```

```
{
```

```
  //포트 방향 설정
```

```
  pinMode(RS, OUTPUT) ;
```

```
  pinMode(RW, OUTPUT) ;
```

```
  pinMode(EN, OUTPUT) ;
```

```
  DDRD = B11111111 ;
```

```
  //명령 1개 전달
```

```
  digitalWrite(RS, LOW); // RS = 0, 명령
```

```
  digitalWrite(RW, LOW) ; // RW = 0, 쓰기
```

```
  digitalWrite(EN, HIGH) ; // E = 1
```

```
  PORTD = 0x38;      // 데이터 출력
```

```
  delayMicroseconds(1);
```

```
  digitalWrite(EN, LOW); // 데이터 쓰기 동작 끝
```

```
  delayMicroseconds(1);
```

```
}
```

```
void loop()
```

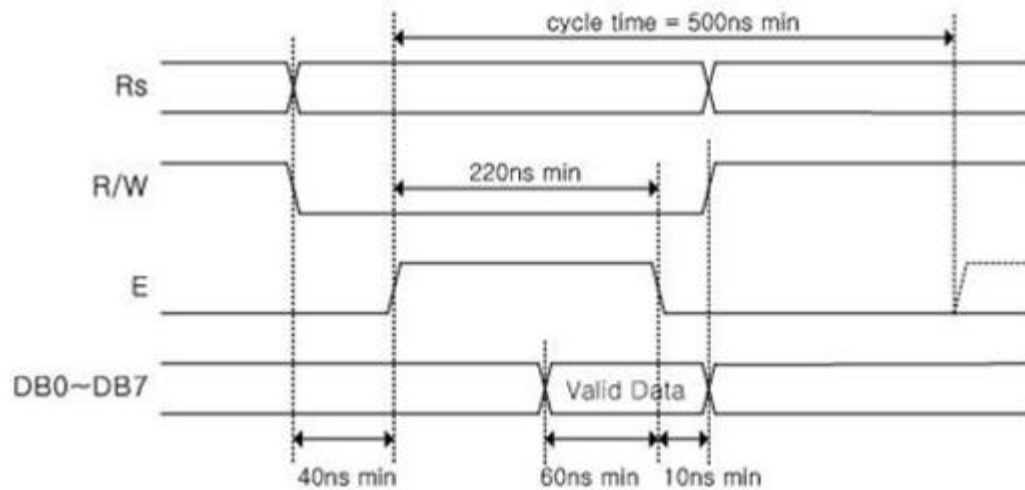
```
{
```

```
}
```



16x2 Character LCD 실험 – example_27

- LCD 초기화 코드 작성



(a) Write from to LCD

```
#define RS    11
#define RW    12
#define EN    13
```

```
void LCD_Command_Write(char cmd)
```

```
{
    delayMicroseconds(100000);           //100msec
    digitalWrite(RS, LOW); //8 - RS
    digitalWrite(RW, LOW); //9 - RW
    digitalWrite(EN, HIGH); //10 - Enable

    PORTD = cmd;
    delayMicroseconds(1);
    digitalWrite(EN, LOW); //10 - Enable
    delayMicroseconds(1);
}
```

```
void setup()
```

```
{
    //포트 방향 설정
    pinMode(RS, OUTPUT);
    pinMode(RW, OUTPUT);
    pinMode(EN, OUTPUT);
    DDRD = B11111111;

    //명령 1개 전달
    LCD_Command_Write(0x38);
}
```

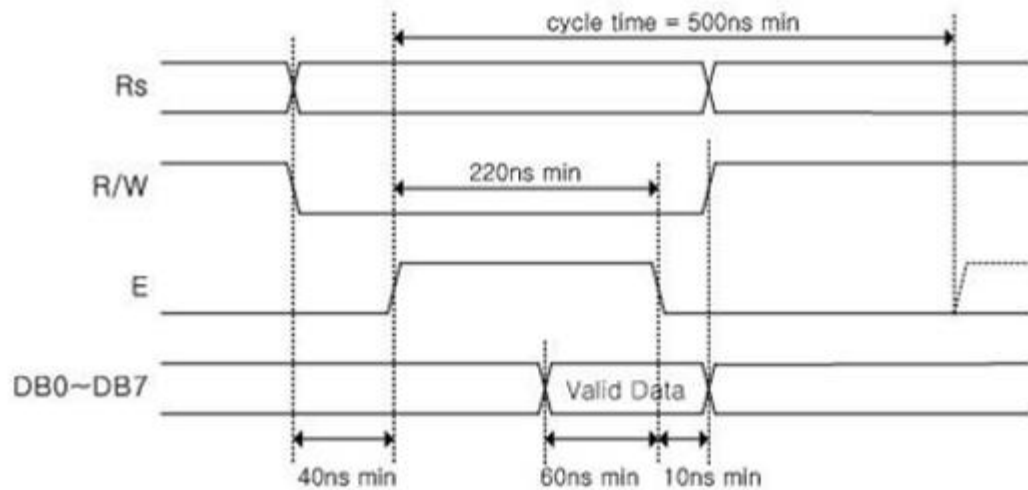
```
void loop()
```

```
{
}
```



16x2 Character LCD 실험 – example_28

- LCD 초기화 코드 작성



(a) Write from to LCD

```
void LCD_Command_Write(char cmd)
{
  ....
}

void setup()
{
  ....

  //LCD초기화
  delay(150);
  LCD_Command_Write(0x38); //0x38 = 0011 1000
  delayMicroseconds(4100) ;
  LCD_Command_Write(0x38);
  delayMicroseconds(100);
  LCD_Command_Write(0x38);
  delayMicroseconds(100) ;

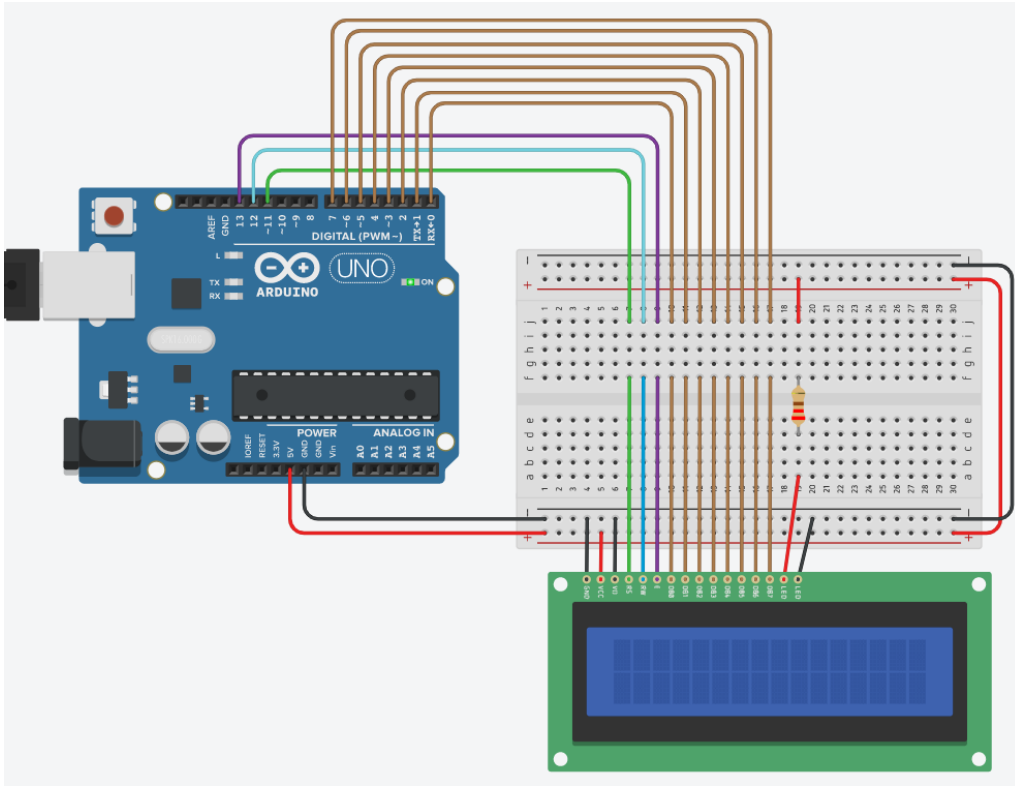
  LCD_Command_Write(0x38);
  LCD_Command_Write(0x0E);
  LCD_Command_Write(0x01);
  LCD_Command_Write(0x04);
}

void loop()
{
  ...
}
```



16x2 Character LCD 실험 – example_28

- LCD 초기화 코드 작성



```
#define RS    11
#define RW    12
#define EN    13

void LCD_Command_Write(char cmd)
{
    ....
}

void setup()
{
    //포트 방향 설정
    pinMode(RS, OUTPUT);
    pinMode(RW, OUTPUT);
    pinMode(EN, OUTPUT);
    DDRD = B11111111;

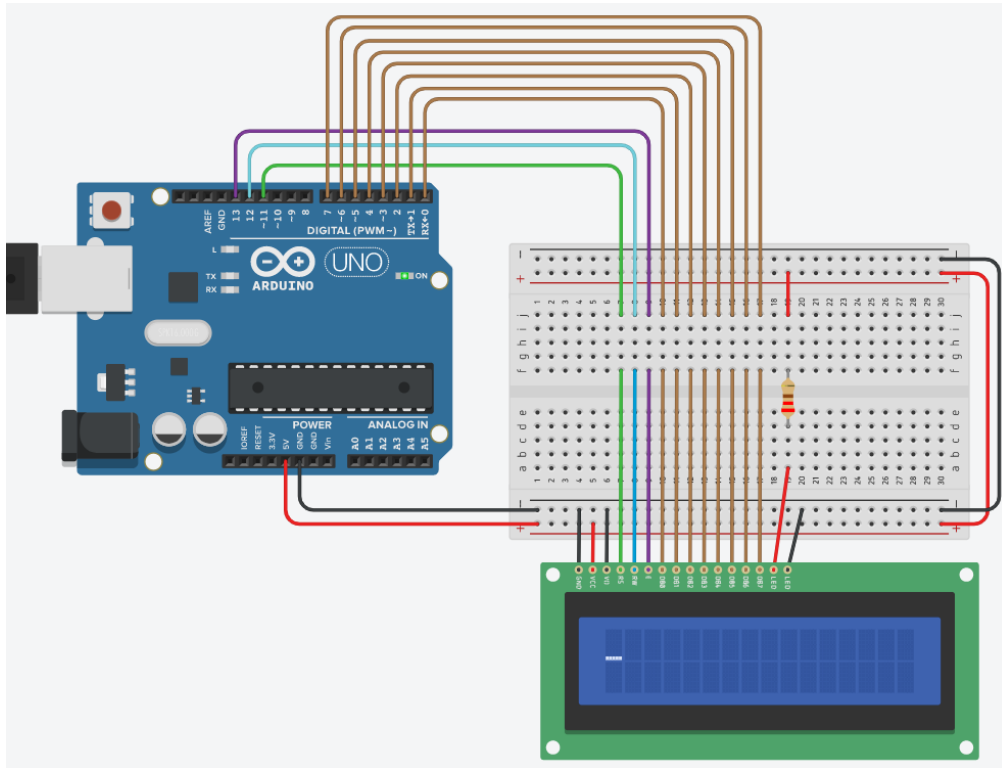
    //LCD초기화
    delayMicroseconds(150000);
    LCD_Command_Write(0x38); //0x38 = 0011 1000
    delayMicroseconds(4100);
    LCD_Command_Write(0x38);
    delayMicroseconds(100);
    LCD_Command_Write(0x38);
    delayMicroseconds(100);

    LCD_Command_Write(0x38);
    LCD_Command_Write(0x08);
    LCD_Command_Write(0x01);
    LCD_Command_Write(0x04);
}
```



16x2 Character LCD 실험 – example_29

- LCD 초기화 코드 작성



```
#define RS      11
#define RW      12
#define EN      13
```

```
void LCD_Command_Write(char cmd)
{
    ....
}
```

```
void setup()
{
    //포트 방향 설정
    pinMode(RS, OUTPUT) ;
    pinMode(RW, OUTPUT) ;
    pinMode(EN, OUTPUT) ;
    DDRD = B11111111 ;
}
```

```
//LCD초기화
delayMicroseconds(150000);
LCD_Command_Write(0x38); //0x38 = 0011 1000
delayMicroseconds(4100) ;
LCD_Command_Write(0x38);
delayMicroseconds(100);
LCD_Command_Write(0x38);
delayMicroseconds(100) ;
```

```
LCD_Command_Write(0x38);
LCD_Command_Write(0x0E);           //Display On, 커서표시
LCD_Command_Write(0x01);
LCD_Command_Write(0x04);
}
```



16x2 Character LCD 실험 – example_30

• LCD 문자표시 코드 작성

명령	명령	데이터										설명	실행 시간
		RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
명령 쓰기	화면지우기	0	0	0	0	0	0	0	0	0	1	화면지우기, 커서홈	1,52ms
	커서홈	0	0	0	0	0	0	0	0	1	-	커서 처음 위치로 이동	1,52ms
	엔트리 모드 셋	0	0	0	0	0	0	0	1	I/D	S	어드레스자동증가/감소(I/D) 표시 쉬프트(S)	37us
	표시 On/Off 제어	0	0	0	0	0	0	1	D	C	B	디스플레이(D), 커서(C), 깜박임(B) On/Off	37us
	표시, 커서 쉬프트	0	0	0	0	0	1	S/C	R/L	-	-	표시, 커서 이동	37us
	기능 셋	0	0	0	0	1	DL	N	F	-	-	인터페이스라인(DL), 라인수(N), 문자폰트(F)	37us
	CGRAM 어드레스	0	0	0	1	CGRAM 어드레스(ACG)					CGRAM 어드레스 설정		37us
명령 읽기	DDRAM 어드레스	0	0	1	DDRAM 어드레스(ADD)					DDRAM 어드레스 설정		37us	
	비지블레고 어드레스	0	1	BF	어드레스 카운터(AC)					비지블레고 읽기 어드레스 카운터 읽기		0us	

D.D.RAM(Display Data RAM)

- 80x8비트 용량으로 80개의 8비트 아스키(ASCII)코드를 저장할 수 있다.
- 0x00~0F 주소가 LCD의 1행의 1~16째.
- 0x40~4f 주소가 LCD의 2행의 1~16번째 문자로 표시 된다.
- 빈 주소에는 자유롭게 RAM 데이터 메모리로 사용이 가능하다.

DL=1 : 8비트	DL=0 : 4비트	AC : 어드레스 카운터
N=1 : 2라인	N=0 : 1라인	(DDRAM, CGRAM 어드레스)
F=1 : 5x10 dots	F=0 : 5x8 dot	
BF=1 : 내부 동작중	BF=0 : 명령/데이터 받기 가능	

```
void LCD_Command_Write(char cmd)
```

```
{
...
}
```

```
void setup()
```

```
{
//포트 방향 설정
....
```

```
//LCD초기화
....
```

```
}
```

```
void loop()
```

```
{
```

```
LCD_Command_Write(0x80 | 0x00); // DDRAM Address = 0 설정
```

```
digitalWrite(RS, HIGH); // 0번 비트 설정, RS = 1, 데이터
digitalWrite(RW, LOW); // 1번 비트 클리어, RW = 0, 쓰기
digitalWrite(EN, HIGH); // 2번 비트 설정, E = 1
PORTD = 'a'; // 데이터 출력
delayMicroseconds(1);
digitalWrite(EN, LOW); // 데이터 쓰기 동작 끝
```

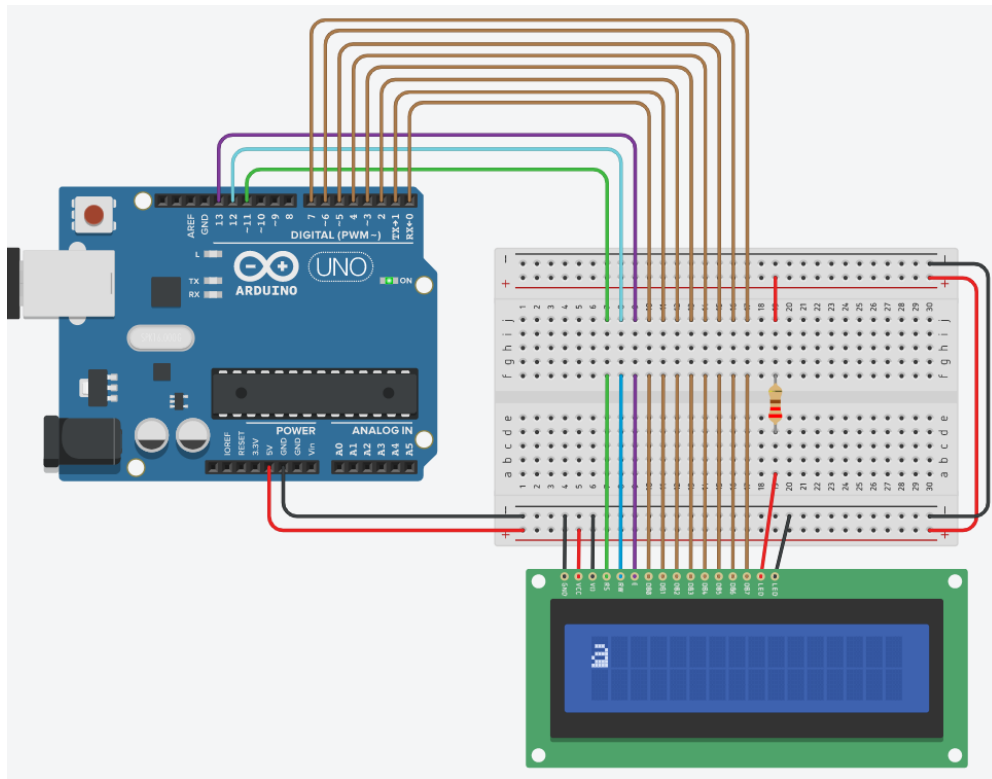
```
delayMicroseconds(1);
```

```
}
```



16x2 Character LCD 실험 – example_31

- LCD 문자표시 코드 작성



```
void LCD_Command_Write(char cmd) {  
....  
}
```

```
void LCD_Data_Write(char data)
```

```
{  
    digitalWrite(RS, HIGH); // 0번 비트 설정, RS = 1, 데이터  
    digitalWrite(RW, LOW) ; // 1번 비트 클리어, RW = 0, 쓰기  
    digitalWrite(EN, HIGH) ; // 2번 비트 설정, E = 1
```

```
    PORTD = data;      // 데이터 출력  
    delayMicroseconds(1);  
    digitalWrite(EN, LOW); // 데이터 쓰기 동작 끝
```

```
    delayMicroseconds(1);  
}
```

```
void setup() {  
    //포트 방향 설정
```

```
....  
    //LCD초기화  
....  
}
```

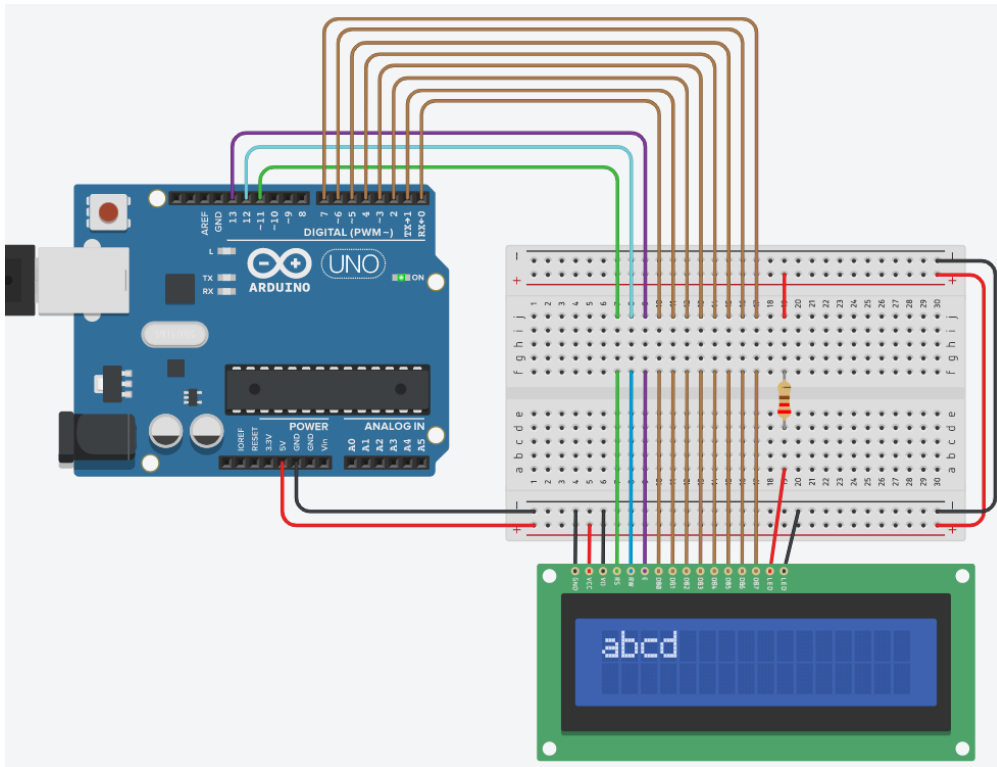
```
void loop()
```

```
{  
    LCD_Command_Write(0x80 | 0x00); // DDRAM Address = 0 설정  
    LCD_Data_Write('a') ;  
}
```



16x2 Character LCD 실험 – example_32

- LCD 문자표시 코드 작성



```
void LCD_Command_Write(char cmd) {  
....  
}
```

```
void LCD_Data_Write(char data) {  
....  
}
```

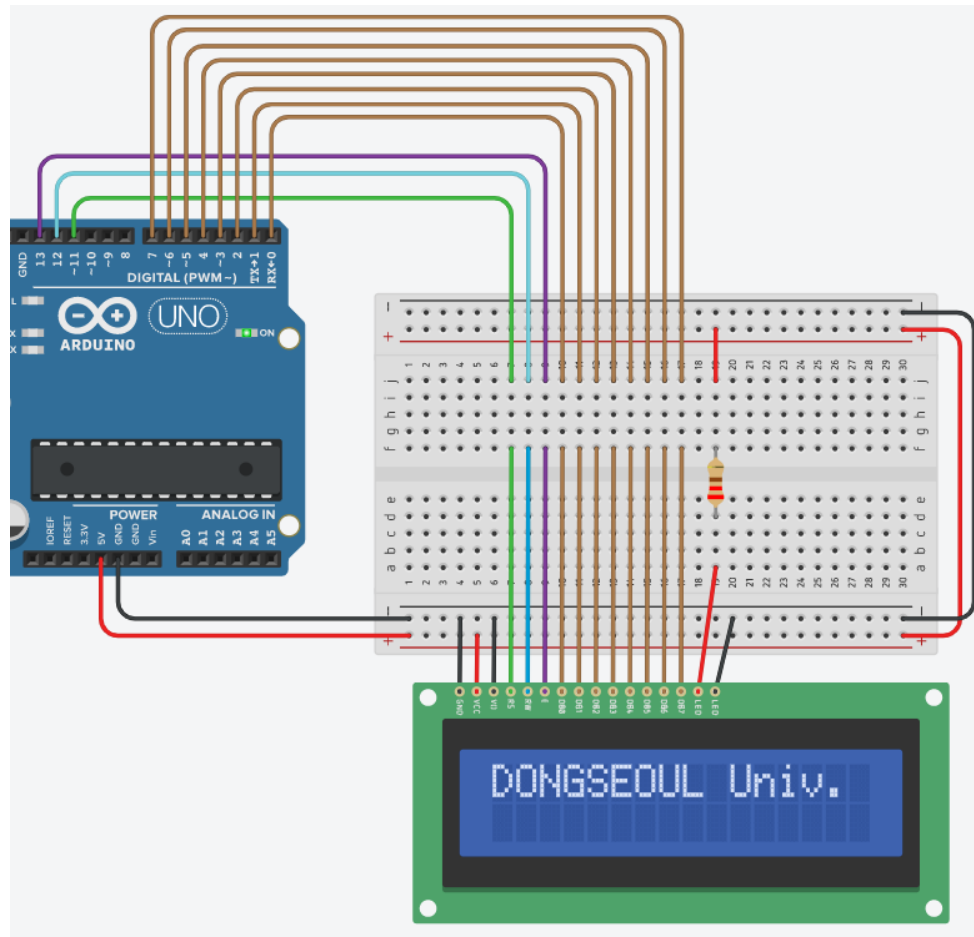
```
void setup() {  
  //포트 방향 설정  
  ....  
  //LCD초기화  
  ....  
}
```

```
void loop()  
{  
  LCD_Command_Write(0x80 | 0x00); // DDRAM Address = 0 설정  
  LCD_Data_Write('a') ;  
  
  LCD_Command_Write(0x80 | 0x01); // DDRAM Address = 1 설정  
  LCD_Data_Write('b') ;  
  
  LCD_Command_Write(0x80 | 0x02); // DDRAM Address = 2 설정  
  LCD_Data_Write('c') ;  
  
  LCD_Command_Write(0x80 | 0x03); // DDRAM Address = 3 설정  
  LCD_Data_Write('d') ;  
}
```



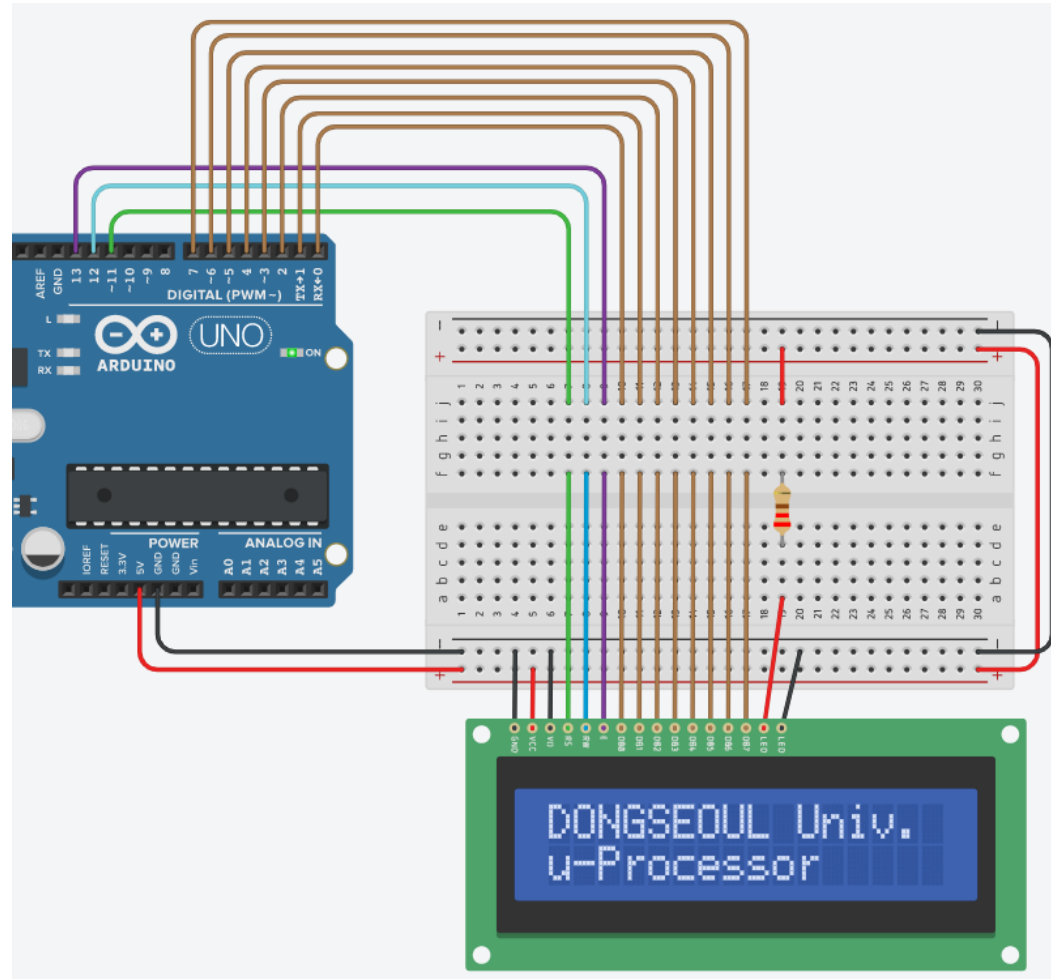
16x2 Character LCD 실험 – quiz1

- Example_32를 참고하여 아래의 그림과 같이 LCD에 문자를 출력 하시오



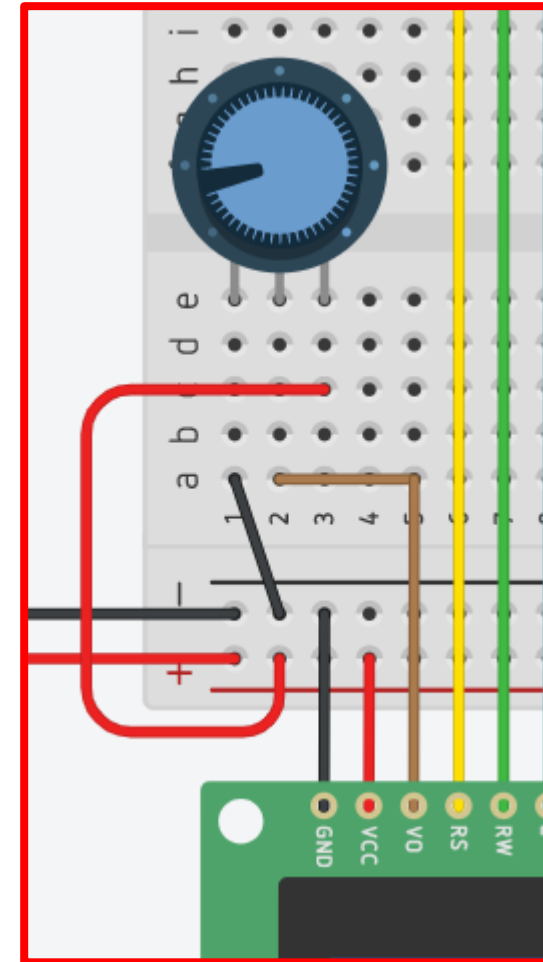
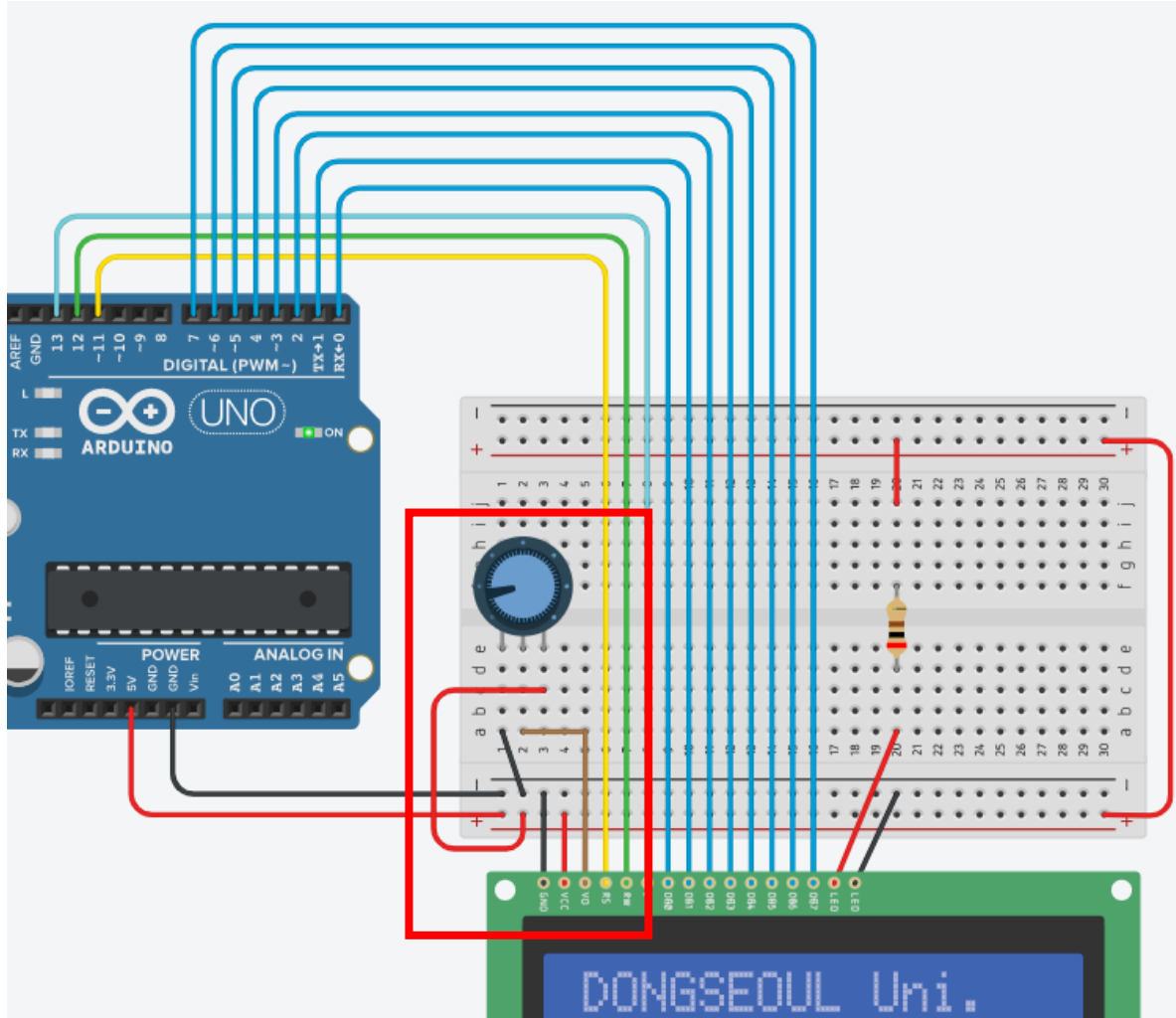
16x2 Character LCD 실험 – quiz2

- Example_32를 참고하여 아래의 그림과 같이 LCD에 문자를 출력 하시오



16x2 Character LCD 실험

- Contrast 조절

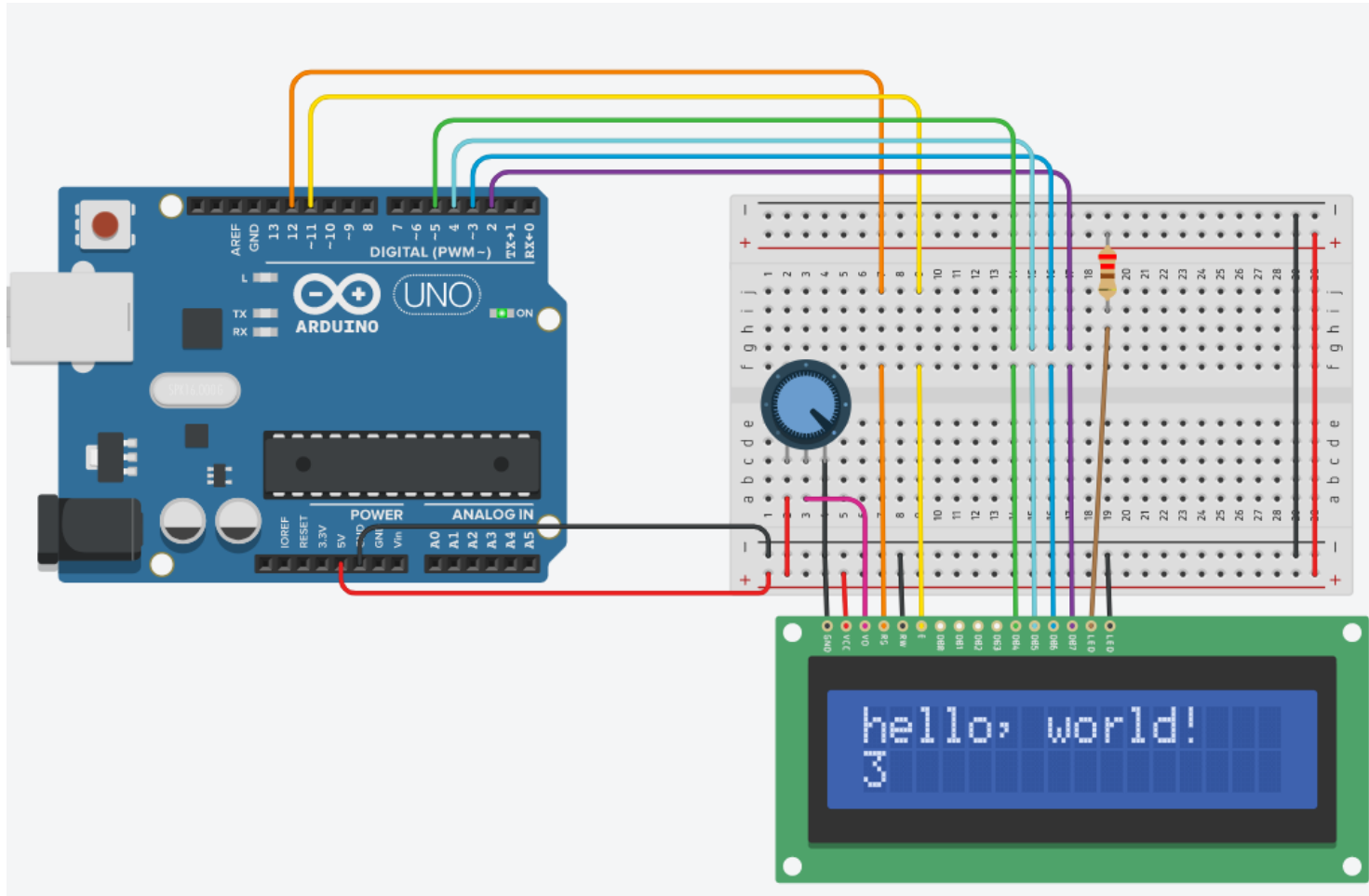


16x2 Character LCD 실험



16x2 Character LCD 실험 – example_35

- 아두이노 라이브러리 사용한 LCD 실험



```
#include <LiquidCrystal.h>
```

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

```
void setup() {  
  lcd.begin(16, 2);  
  lcd.print("hello, world!");  
}
```

```
void loop() {  
  lcd.setCursor(0, 1);  
  lcd.print(millis() / 1000);  
}
```


16x2 Character LCD 실험 – example_38

- 아두이노 라이브러리 사용한 LCD실험, Backlight 포트 제어

