CNRLAB

CIM&ROBOTICS LABORATORY

로봇공학입문설계

4주차 모바일 로봇(2)

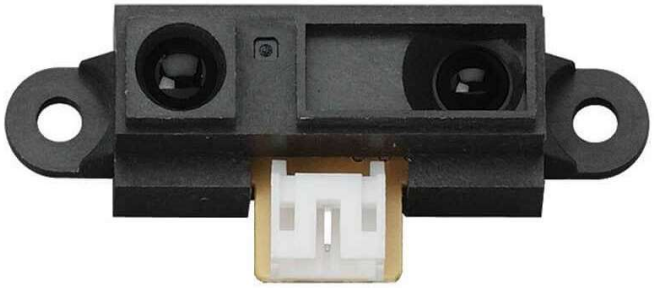
로봇공학과

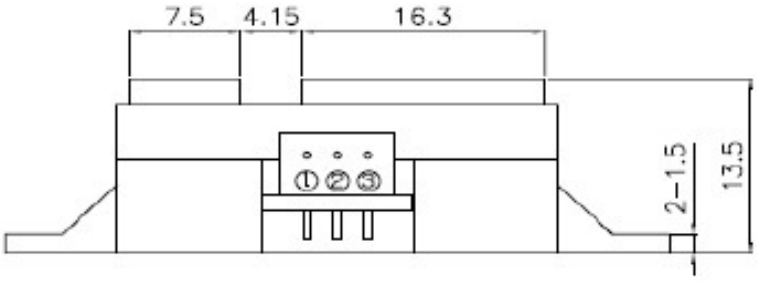
Contents

- PSD 센서
- LCD
- Encoder
- Interrupt

PSD 센서

❑ Specification : SHARP GP2Y0A41SK0F





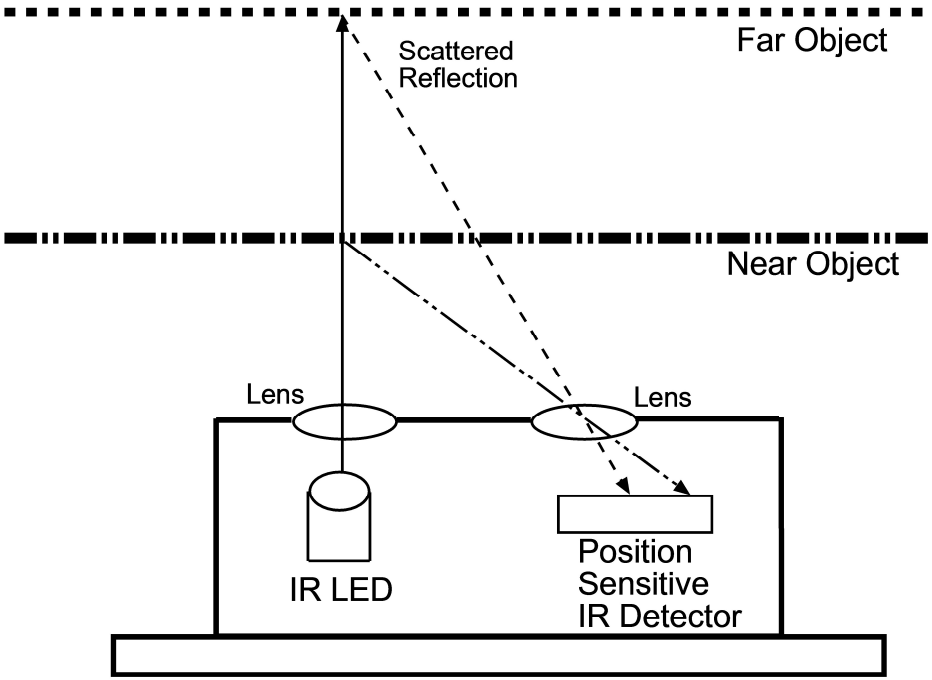
Connector signal

	Signal name
①	Vo
②	GND
③	Vcc

- PSD(Position Sensitive Distance), Infrared LED, Signal Processing Circuit 으로 구성된 거리 측정 센서 모듈
- 측정 범위는 4cm~30cm
- Analog output 타입

PSD 센서

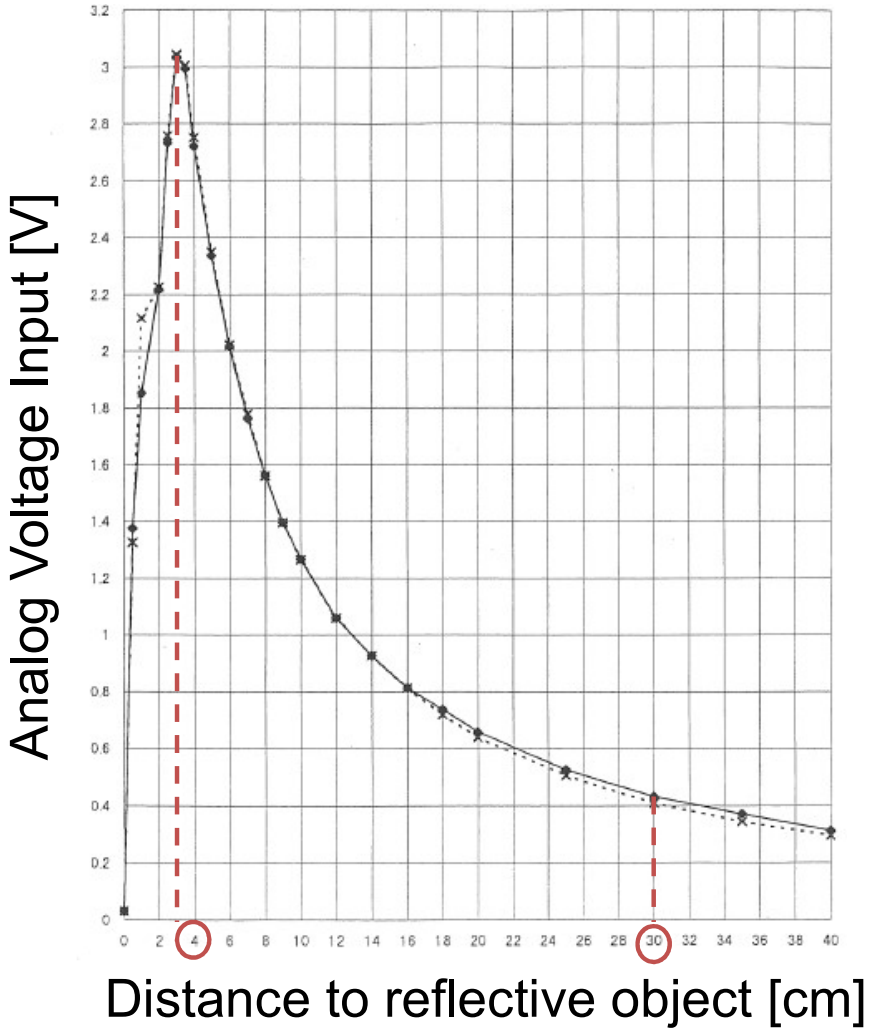
❑ 작동 원리



- ❍ 발광부에서 발산한 적외선이 물체에 반사되어 수광부에 도달
- ❍ 수광부의 PSD가 반사된 빛이 도달한 지점을 측정
- ❍ 물체와의 거리에 따라 PSD가 측정하는 빛의 위치가 달라짐을 이용

PSD 센서

❑ 거리 측정



$$distance = 12.67 \times volts^{-1.069}$$

1) analogRead(A0) → volts :

volts = analogRead(A0)x(5.0/1023.0)

2) volts → distance :

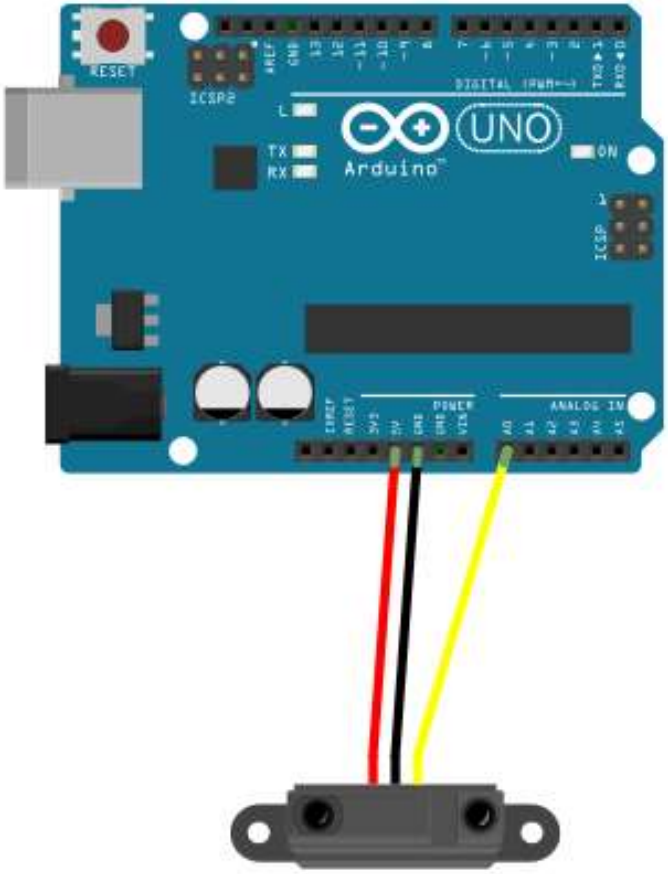
distance = 12.67xpow(volts, -1.069)

pow(*base, exponent*)

$base^{exponent}$의 값을 반환한다.

PSD 센서

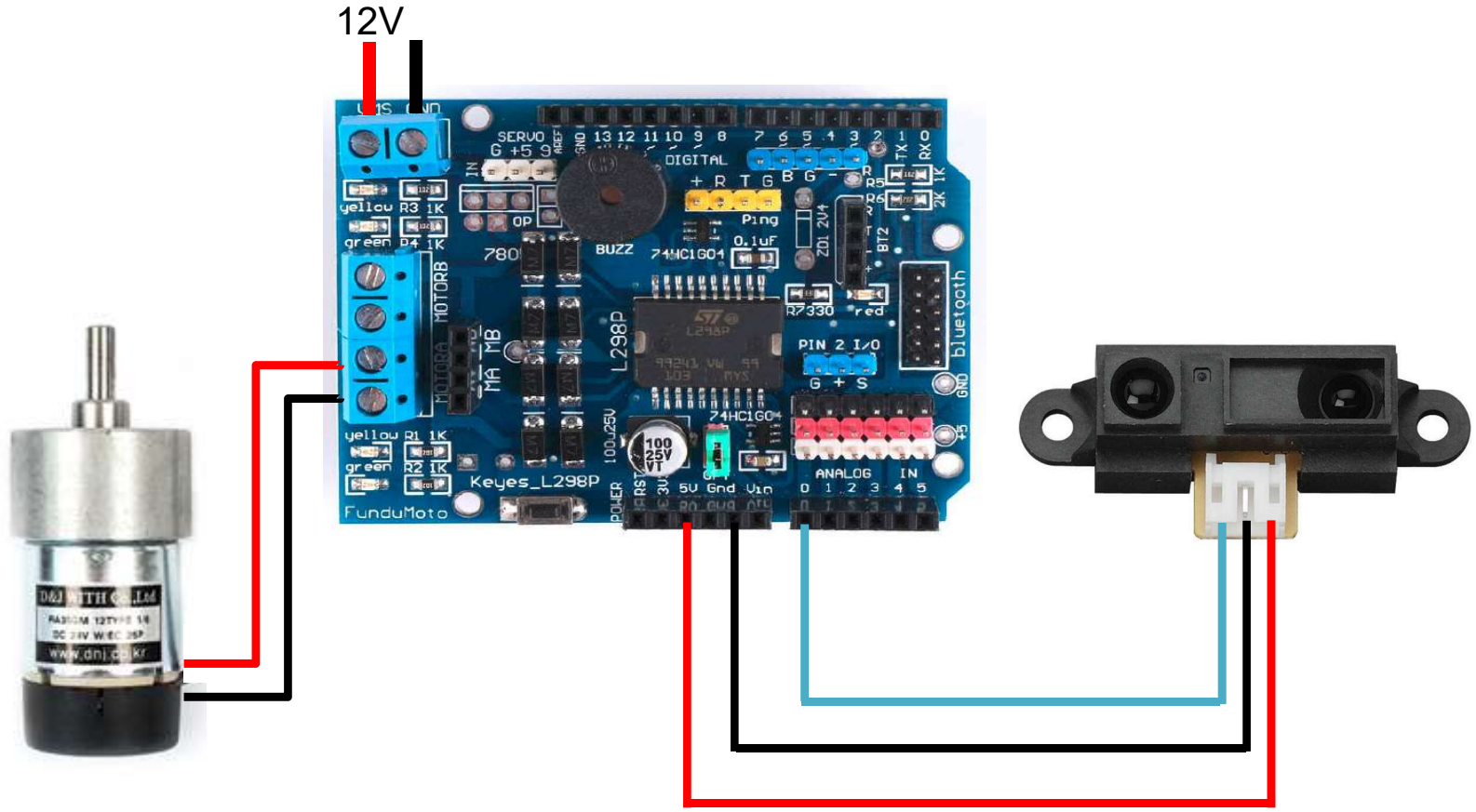
[예제1] PSD 센서로 거리 측정




```
#define PSD_PIN A0
float volts;
float distance;

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

void loop() {
  volts = analogRead(PSD_PIN)*(5.0/1023.0);
  distance = 12.67*pow(volts, -1.069);
  Serial.print(volts);
  Serial.print(", ");
  Serial.println(distance);
  delay(100);
}
```

PSD 센서

[예제2] PSD 센서로 DC Motor 회전방향 및 속도 조절



PSD 센서

[예제2] PSD 센서로 DC Motor 회전방향 및 속도 조절

```
#define MA_DIR 12
#define MA_PWM 10
#define PSD_PIN A0
#define THRESH 15
#define K_P 30
// PSD
float volts;
float distance;
// Motor
float err_dist;
int velocity;

void setup() {
  pinMode(MA_DIR, OUTPUT);
  pinMode(MA_PWM, OUTPUT);
  pinMode(PSD_PIN, INPUT);
  Serial.begin(9600);
}
```

```
void loop(){
  volts = analogRead(PSD_PIN)*5.0/1023.0;
  distance = 12.67*pow(volts, -1.069);
  err_dist = distance - THRESH;
  velocity = err_dist*K_P;

  if(err_dist >= 0){
    digitalWrite(MA_DIR, HIGH);
    analogWrite(MA_PWM, velocity);
  }
  else{
    digitalWrite(MA_DIR, LOW);
    analogWrite(MA_PWM, -velocity);
  }

  Serial.print("ERR: ");
  Serial.print(err_dist);
  Serial.print(", VEL: ");
  Serial.println(velocity);
}
```

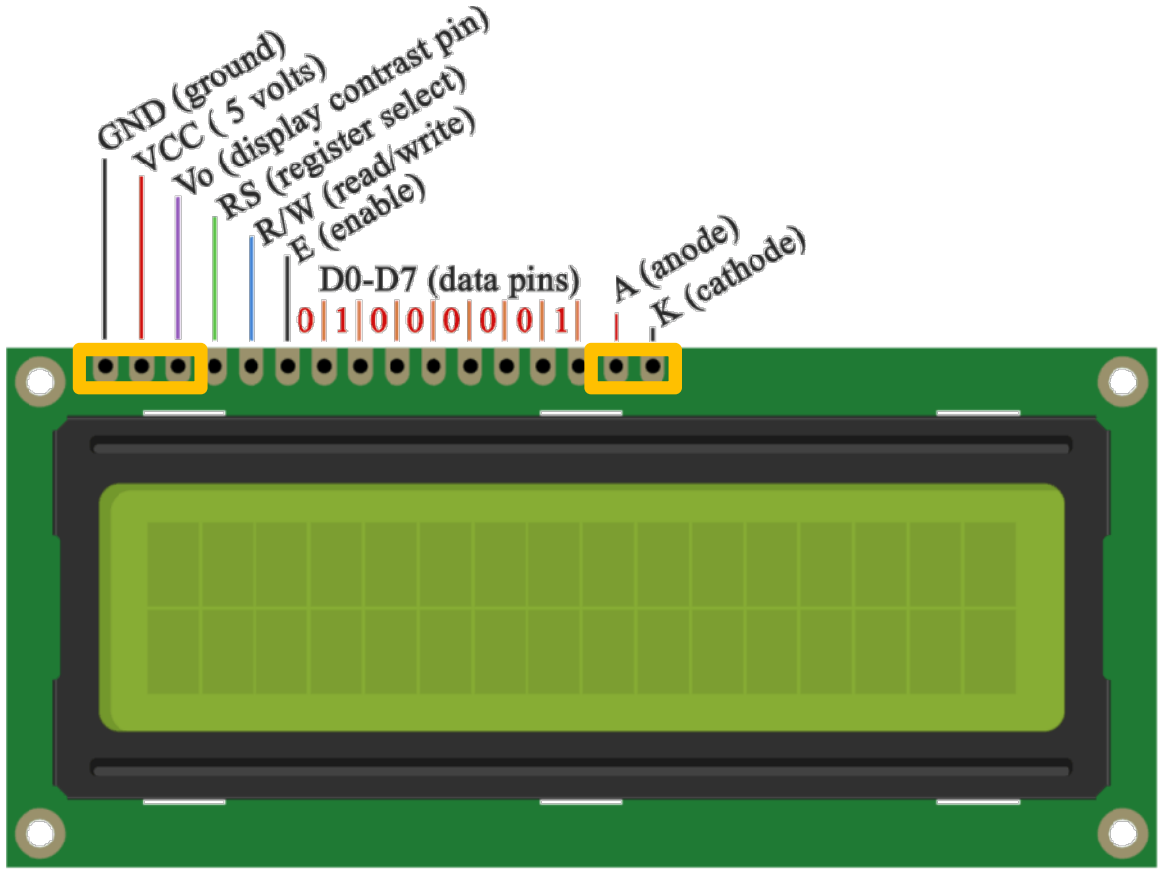
LCD

❑ Pin Assignment

No.	Symbol	Level	Function	
1	Vss	--	0V	Power Supply
2	Vdd	--	+5V	
3	V0	--	for LCD	
4	RS	H/L	Register Select: H:Data Input L:Instruction Input	
5	R/W	H/L	H--Read L--Write	
6	E	H,H-L	Enable Signal	
7	DB0	H/L	Data bus used in 8 bit transfer	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L	Data bus for both 4 and 8 bit transfer	
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		
15	BLA	--	BLACKLIGHT +5V	
16	BLK	--	BLACKLIGHT 0V-	

LCD

❑ Pin Assignment



Display

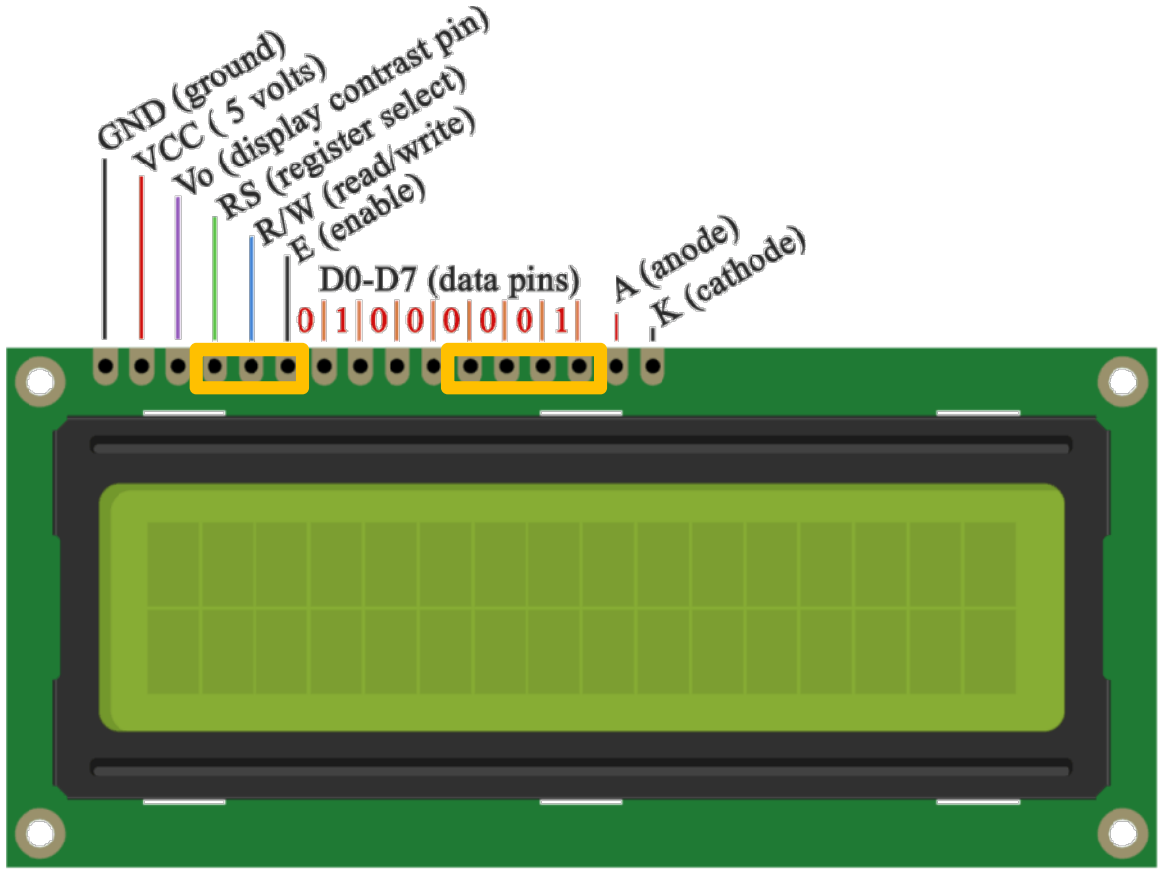
1. Backlight ON

pin	volts
A	5V
K	GND

2. Display 밝기 조절 → 가변저항

pin	volts
VSS	GND
VDD	5V
V0	0~5V

❑ Pin Assignment



Data

3. 7pins → Using Arduino

pin	volts
RS	H: Data Input
	L: Instruction input
R/W	H: Reading mode
	L: Writing mode
E	Enables writing to the registers
D4~7	Data

LCD

❑ LCD 관련 함수

LiquidCrystal lcd(*RS, E, D4, D5, D6, D7*)

LiquidCrystal 클래스 생성자, 위의 순서로 연결된 6개의 디지털 핀 번호를 입력

lcd.begin(*width, height*)

넓이가 *width*, 높이가 *height* 의 크기를 가지는 LCD의 크기 설정

lcd.print()

()안의 값을 LCD에 출력

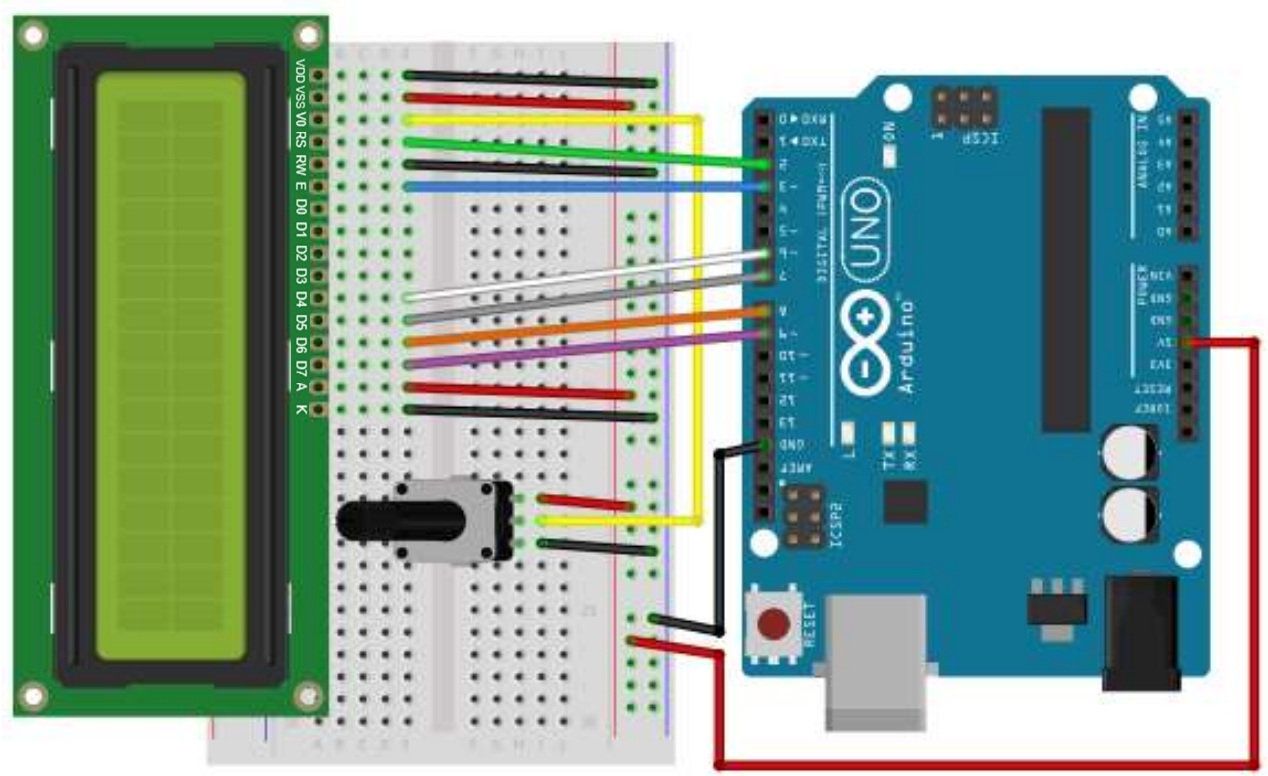
lcd.setCursor(*row, column*);

원하는 행과 열에 커서의 위치를 지정(0행, 0열부터 시작)
지정한 위치에서부터 디스플레이 시작

lcd.clear()

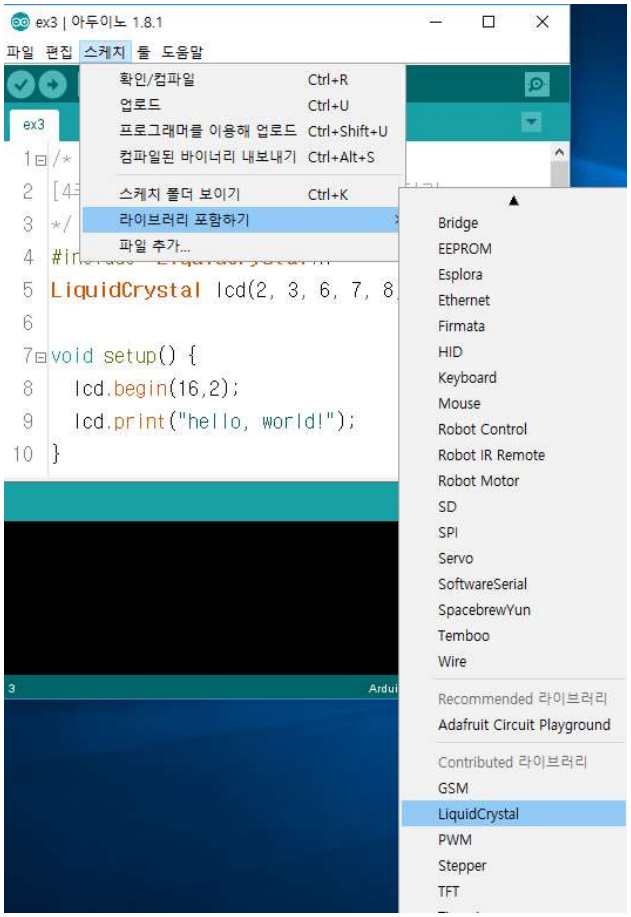
lcd에 출력된 이전의 내용들을 모두 clear

[예제3] LCD에 hello world 출력하기



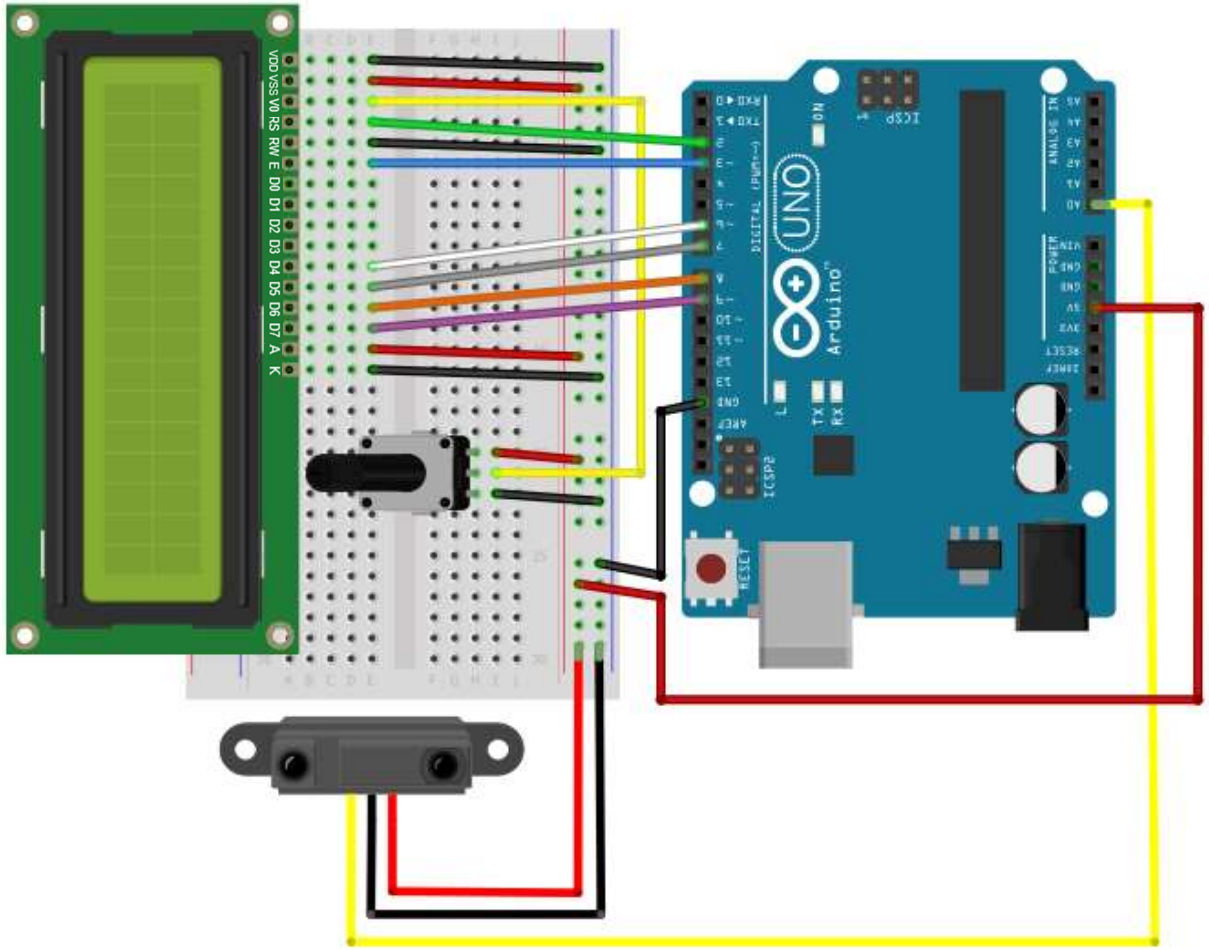
[예제3] LCD에 hello world 출력하기

LiquidCrystal 라이브러리 추가




```
#include <LiquidCrystal.h>
LiquidCrystal lcd(2, 3, 6, 7, 8, 9);

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

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

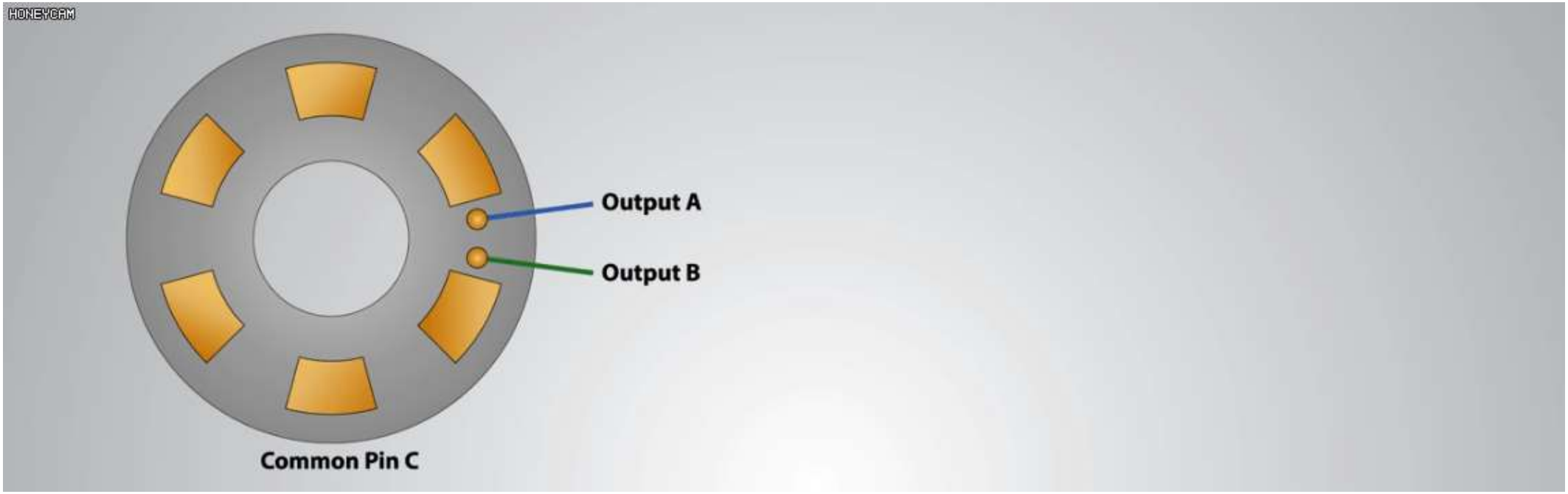
[예제4] LCD에 PSD로 측정한 거리 출력하기




```
#include <LiquidCrystal.h>
#define PSD_PIN A0
LiquidCrystal lcd(2, 3, 6, 7, 8, 9);
// PSD
float volts;
float distance;

void setup() {
  lcd.begin(16,2);
}

void loop() {
  volts = analogRead(PSD_PIN)*(5.0/1023.0);
  distance = 12.67*pow(volts, -1.069);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Distance: ");
  lcd.setCursor(10, 0);
  lcd.print(distance);
  delay(500);
}
```

Encoder

❑ 구동 원리

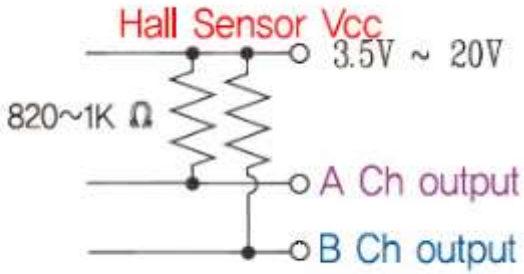


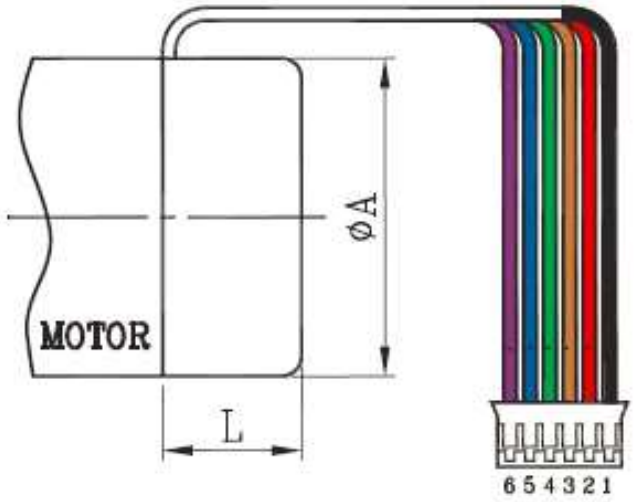
Encoder

❑ Encoder Specification (RA-35GM 11TYPE (12V) with 2channel Encoder)

- Reduction ratio : 1/30
- Encoder : 26P/R

출력회로 구성
Output Circuit :





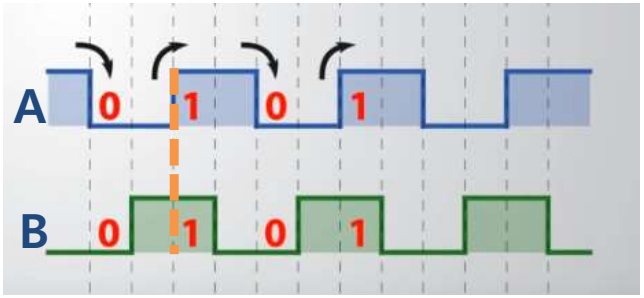
엔코더 컨넥터 핀별 내용 :
Two Channel Encoder Connections :

1.	Black	: -MOTOR
2.	Red	: +MOTOR
3.	Brown	: HALL SENSOR Vcc
4.	Green	: HALL SENSOR GND
5.	Blue	: HALL SENSOR B Vout
6.	Purple	: HALL SENSOR A Vout

Encoder

❑ 구동 원리(1) 회전방향

- ❍ ChA가 LOW→HIGH 일 때
  - ▹ ChB가 HIGH이면 정방향
  - ▹ ChB가 LOW이면 역방향

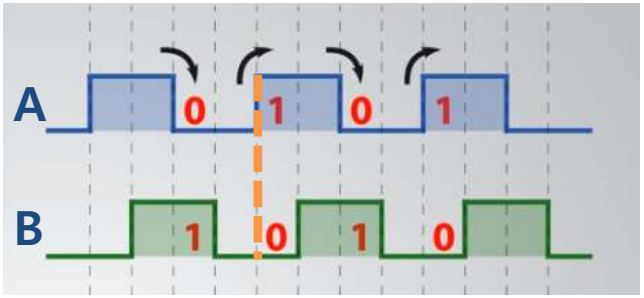


정방향

- ❍ 4체배 적용 : 한 pulse당 방향 4번 체크

Ch A \ Ch B	LOW	HIGH
L→H	역방향	정방향
H→L	정방향	역방향

Ch B \ Ch A	LOW	HIGH
L→H	정방향	역방향
H→L	역방향	정방향



역방향

Encoder

❑ 구동 원리(2) 회전속도

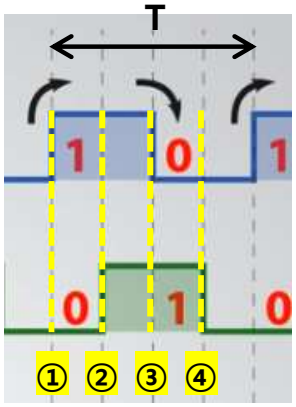
❍ 26P/R, 2channel Encoder

▹ 1회전시, ChA, ChB에서 각각 13pulse씩 출력
(=한 주기(T)가 13번 반복된다.)

❍ 4체배 적용

▹ 한 주기당, 4번 pulse 체크

▹ 1회전시, 한 채널에서 13pulse 출력하므로
총 13[pulse] x 4 = 52번 pulse 체크

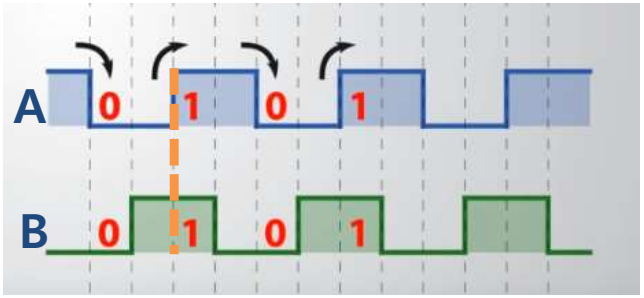


① ChA : L→H, Rising Edge

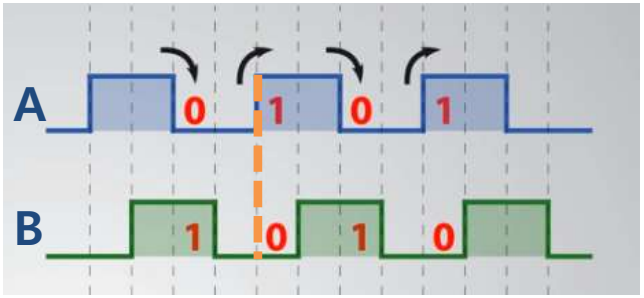
② ChB : L→H, Rising Edge

③ ChA : H→L, Falling Edge

④ ChB : H→L, Falling Edge



정방향



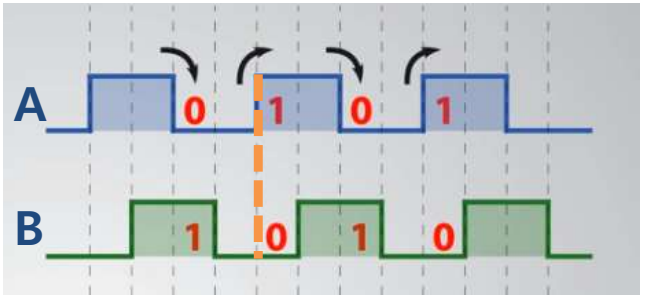
역방향

Encoder

❑ 구동 원리

```
// a low-to-high edge on channel A
if(digitalRead(EA_CHA) == HIGH) {
  if (digitalRead(EA_CHB) == LOW)        enAPos = enAPos - 1;
  else                                   enAPos = enAPos + 1;
}

// must be a high-to-low edge on channel A
else {
  if (digitalRead(EA_CHB) == HIGH)       enAPos = enAPos - 1;
  else                                   enAPos = enAPos +1;
}
```

```
// a low-to-high edge on channel B
if(digitalRead(EA_CHB) == HIGH) {
  if (digitalRead(EA_CHA) == HIGH)       enAPos = enAPos - 1;
  else                                    enAPos = enAPos + 1;
}

// must be a high-to-low edge on channel B
else {
  if (digitalRead(EA_CHA) == LOW)        enAPos = enAPos - 1;
  else                                    enAPos = enAPos + 1;
}
```



역방향

Ch A \ Ch B	LOW	HIGH
L→H	역방향	정방향
H→L	정방향	역방향

Ch B \ Ch A	LOW	HIGH
L→H	정방향	역방향
H→L	역방향	정방향

Encoder

❑ 구동 원리

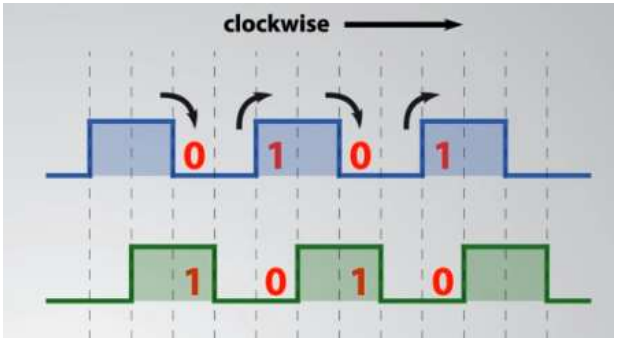
❍ RPM계산

- 52번 pulse 체크(=1회전)하는 동안 걸리는 시간을 측정한다.
- 기어비 1/30이므로, 모터 축이 회전하는 속도는 1/30배를 해주어야 한다.

$$RPM = \frac{1\text{회전}}{\text{걸린시간}[min]} \times \text{기어비} = \frac{1{,}000{,}000}{dt[\mu s]} \times \frac{1}{30} \times 60 = \frac{2{,}000{,}000}{dt}$$

Encoder

- **Edge를 detect하기 위해서는...?**



Interrupt 이용!

Interrupt는 edge detecting을 할 수 있을 뿐만 아니라 특정 순간(rising edge, falling edge 등)에 원하는 일을 수행할 수 있도록 한다.

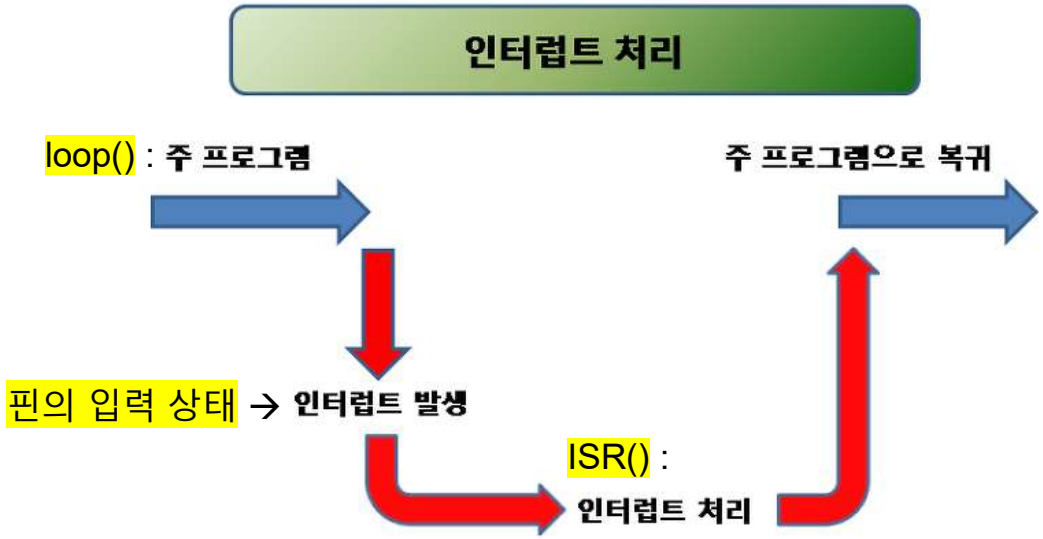
Encoder

❑ Interrupt 란?

Encoder

❑ Interrupt 란?

❍ 인터럽트는 특정 핀의 입력 상태가 바뀔 때, 아두이노가 이를 자동으로 감지하여 모든 동작을 잠시 멈춘 다음, ISR이라 부르는 함수를 실행하고 다시 원래 작업으로 복귀 하는 기능을 뜻한다.



Encoder

❑ Interrupt 관련 함수

attatchInterrupt(*digitalPinToInterrupt(interruptPin), ISR, mode*)

digitalPinToInterrupt(interruptPin) : interrupt 번호

Board	int.0	int.1	int.2	int.3	int.4	int.5
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
32u4 based (e.g Leonardo, Micro)	3	2	0	1	7	
Due, Zero, MKR1000, 101	interrupt number = pin number					

ISR : interrupt가 발생할 때 부르는 함수 명(interrupt service routine)

mode : interrupt가 trigger되는 시점을 정의

- CHANGE : LOW→HIGH, HIGH→LOW
- RISING : LOW→HIGH
- FALLING : HIGH→LOW

Encoder

❑ Interrupt 관련 함수

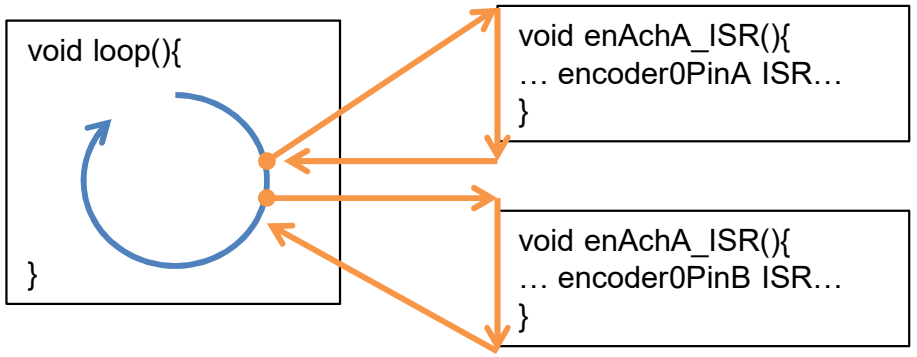
attatchInterrupt(*digitalPinToInterrupt(interruptPin), ISR, mode*)

Board	int.0	int.1	int.2	int.3	int.4	int.5
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
32u4 based (e.g Leonardo, Micro)	3	2	0	1	7	
Due, Zero, MKR1000, 101	interrupt number = pin number					

적용예시

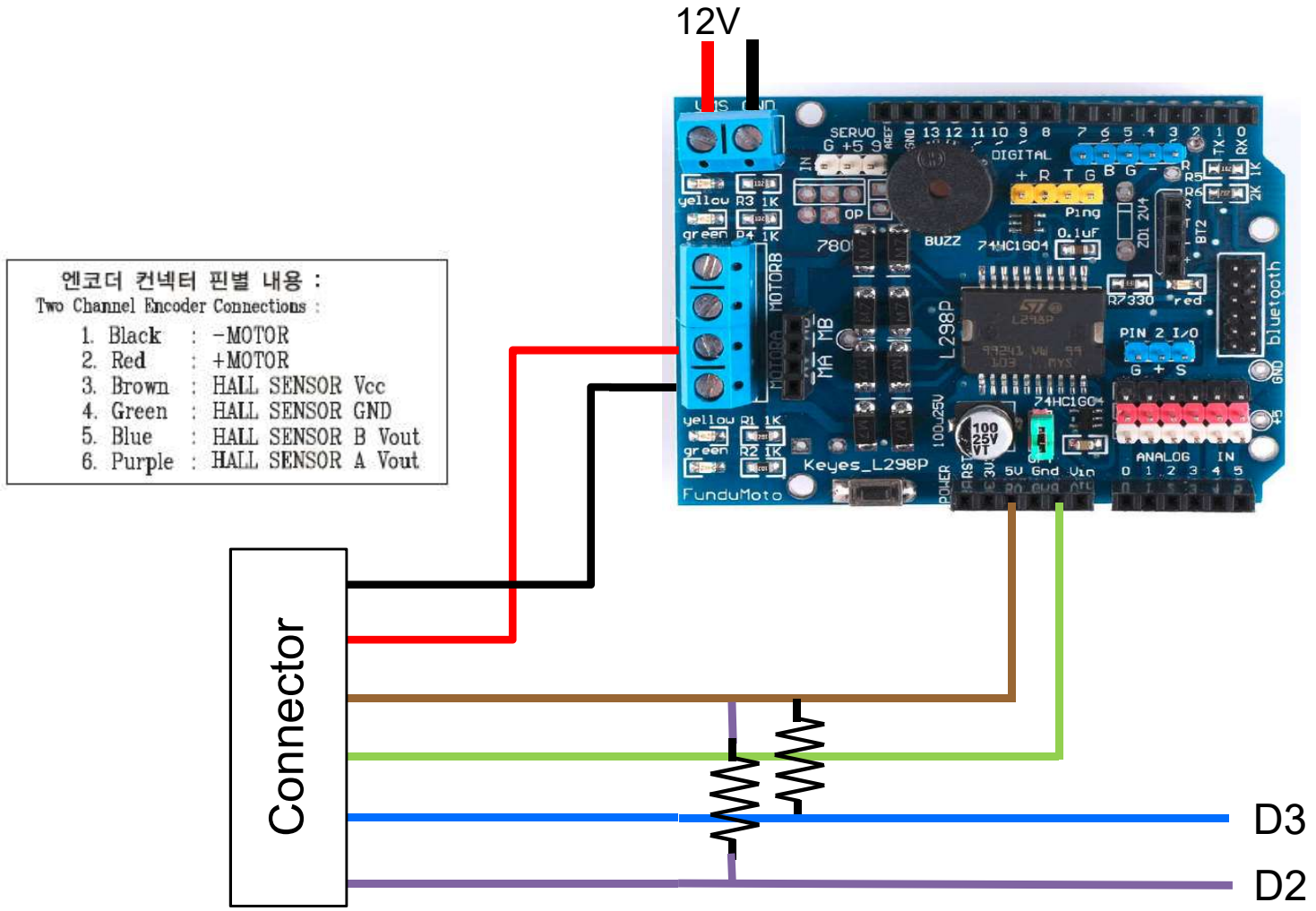
```
#define EA_CHA 2
#define EA_CHB 3

void setup() {
  pinMode(EA_CHA, INPUT);
  pinMode(EA_CHB, INPUT);
  attachInterrupt(digitalPinToInterrupt(EA_CHA), enAchA_ISR, CHANGE);
  attachInterrupt(digitalPinToInterrupt(EA_CHB), enAchB_ISR, CHANGE);
}
```

```
attachInterrupt(0, enAchA_ISR, CHANGE);
attachInterrupt(1, enAchB_ISR, CHANGE);
```



Encoder

[예제5] 엔코더로 모터의 회전속도 읽기



Encoder

[예제5] 엔코더로 모터의 회전속도 읽기

```
#define EA_CHA 2
#define EA_CHB 3
#define MA_DIR 12
#define MA_PWM 10
// Encoder
int enAPos = 0;
unsigned long current=0;
unsigned long previous=0;
long dt;
int rpm;

void setup() {
  pinMode(EA_CHA, INPUT);
  pinMode(EA_CHB, INPUT);
  pinMode(MA_DIR, OUTPUT);
  pinMode(MA_PWM, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(EA_CHA), enAchA_ISR,
CHANGE);
  attachInterrupt(digitalPinToInterrupt(EA_CHB), enAchB_ISR,
CHANGE);
  Serial.begin (115200);
}
```

```
void loop(){
  if(enAPos==(13*4)){       //13pulse*4=52pulse
    current=micros();
    dt=current-previous;    //us
    rpm = 2000000/dt;       //1000000*(1/30)*(60)=2000000
    Serial.print("rpm:");
    Serial.println(rpm);
    enAPos=0;
    previous=current;
  }
  digitalWrite(MA_DIR, HIGH);
  analogWrite(MA_PWM, 100);
}
```

Encoder

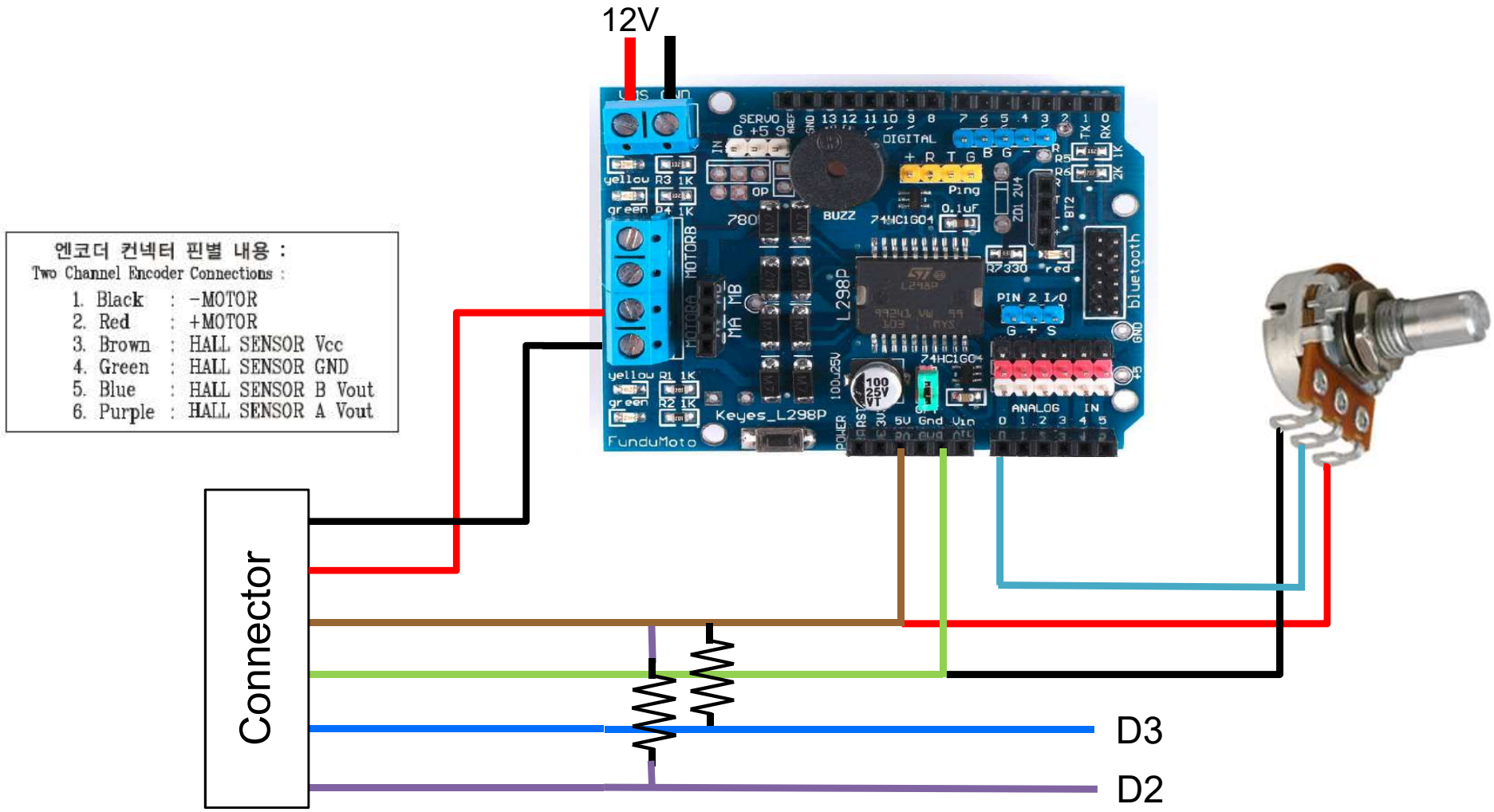
[예제5] 엔코더로 모터의 회전속도 읽기

```
void enAchA_ISR(){
  // a low-to-high edge on channel A
  if(digitalRead(EA_CHA) == HIGH) {
    if(digitalRead(EA_CHB) == LOW) enAPos = enAPos - 1;
    else                           enAPos = enAPos + 1;
  }

  // must be a high-to-low edge on channel A
  else {
    if(digitalRead(EA_CHB) == HIGH) enAPos = enAPos - 1;
    else                            enAPos = enAPos + 1;
  }
}
```

```
void enAchB_ISR(){
  // a low-to-high edge on channel B
  if(digitalRead(EA_CHB) == HIGH) {
    if(digitalRead(EA_CHA) == HIGH) enAPos = enAPos - 1;
    else                            enAPos = enAPos +1;
  }

  // must be a high-to-low edge on channel B
  else {
    if(digitalRead(EA_CHA) == LOW) enAPos = enAPos - 1;
    else                           enAPos = enAPos + 1;
  }
}
```

Encoder

[예제6] 가변저항을 이용한 DC Motor 속도조절 값 출력



Encoder

[예제6] 가변저항을 이용한 DC Motor 속도조절 값 출력

```
#define EA_CHA 2
#define EA_CHB 3
#define MA_DIR 12
#define MA_PWM 10
#define POT_PIN A0
// Encoder
int enAPos = 0;
unsigned long current=0;
unsigned long previous=0;
long dt;
int rpm;
int potValue;
int velocity;

void setup() {
  pinMode(EA_CHA, INPUT);
  pinMode(EA_CHB, INPUT);
  pinMode(POT_PIN, INPUT);
  pinMode(MA_DIR, OUTPUT);
  pinMode(MA_PWM, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(EA_CHA), enAchA_ISR, CHANGE);
  attachInterrupt(digitalPinToInterrupt(EA_CHB), enAchB_ISR, CHANGE);
  Serial.begin(115200);
}
```

```
void loop(){
 potValue = analogRead(POT_PIN);
 velocity = map(potValue, 0, 1023, 0, 255);

  if(enAPos==(13*4)){       //13pulse*4=52pulse
    current=micros();
    dt=current-previous;    //us
    rpm = 2000000/dt;       //1000000*(1/30)*(60)=2000000
    Serial.print("VEL:");
    Serial.print(velocity);
    Serial.print(",               RPM:");
    Serial.println(rpm);
    enAPos=0;
    previous=current;
  }

digitalWrite(MA_DIR, HIGH);
analogWrite(MA_PWM, velocity);

}

void enAchA_ISR(){…}
void enAchB_ISR(){…}
```