GPIO를 활용한 예제 및 실습

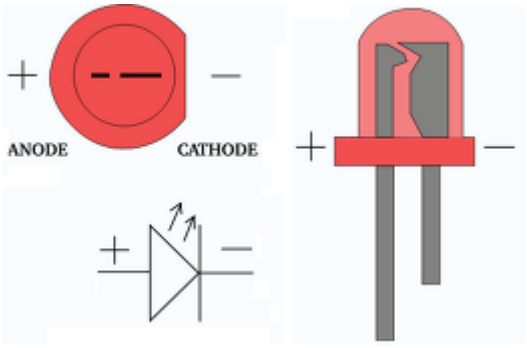
- GPIO를 활용한 제어(LED, SWITCH 등)

엣지아이랩

GPIO를 활용한 제어

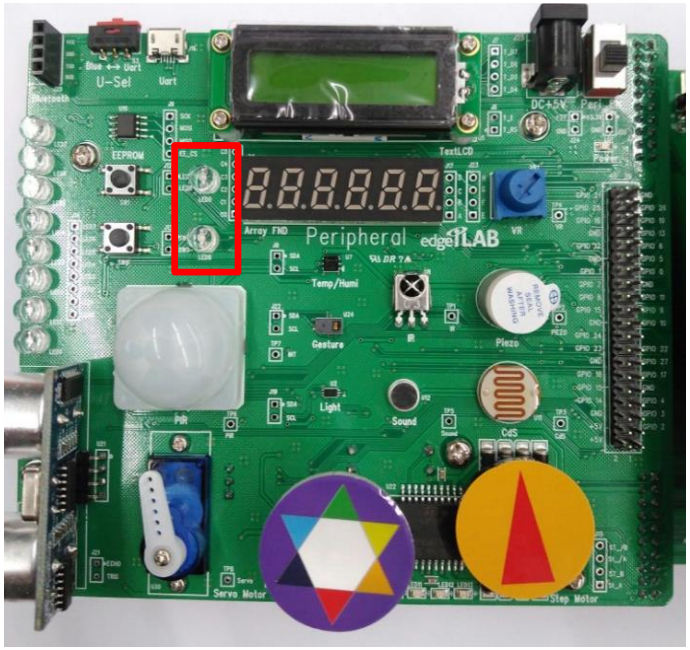
GPIO를 활용한 제어

- LED 제어
  - LED(Light Emitting Diode)란
    - 화합물 반도체의 특성을 이용해 전기에너지를 광 에너지로 변환시켜주는 반도체 소자
    - 전류방향이 일정 전극 방향과 일치할 경우 발광
    - anode(+) 와 cathode(-) 핀 구성



GPIO를 활용한 제어

- LED 동작 제어
  - LED는 핀 21, 22 번(BCM Mode)와 연결
  - 출력 값에 따라 On/Off 제어
    - HIGH(3.3V) 혹은 LOW(0V)



Connect	BCM	Name	Physical		Name	BCM	Connect
		+3.3V	1	2	+5V		
I2C	GPIO 02	SDA1	3	4	+5V		
I2C	GPIO 03	SCL1	5	6	GND		
DC_M	GPIO 04	GPIO	7	8	TxD	GPIO 14	UART
		GND	9	10	RxD	GPIO 15	UART
SERVO	GPIO 17	GPIO	11	12	GPIO	GPIO 18	TLCD
TLCD	GPIO 27	GPIO	13	14	GND		
TLCD	GPIO 22	GPIO	15	16	GPIO	GPIO 23	TLCD
		+3.3V	17	18	GPIO	GPIO 24	PIR
SPI	GPIO 10	MOSI	19	20	GND		
SPI	GPIO 09	MISO	21	22	GPIO	GPIO 25	DC_M
SPI	GPIO 11	SCLK	23	24	CE0	GPIO 08	
		GND	25	26	CE1	GPIO 07	SPI
ULTRA	GPIO 00	SDA0	27	28	SCL0	GPIO 01	ULTRA
SWITCH	GPIO 05	GPIO	29	30	GND		
SWITCH	GPIO 06	GPIO	31	32	GPIO	GPIO 12	DC_M
PIEZO	GPIO 13	GPIO	33	34	GND		
IR	GPIO 19	GPIO	35	36	GPIO	GPIO 16	TLCD
TLCD	GPIO 26	GPIO	37	38	GPIO	GPIO 20	LED
		GND	39	40	GPIO	GPIO 21	LED

GPIO를 활용한 제어

- LED 동작 제어 예제(1)
  - 터미널 창에 "mkdir Example" 입력
    - Example 디렉터리 생성

```
pi@raspberryPi:~ $ mkdir Example
```

  - 터미널 창에 "cd Example" 입력
    - 디렉터리 이동

```
pi@raspberryPi:~ $ cd Example
```

```
pi@raspberryPi:~/Example $
```

  - 터미널 창에 "nano 01_LED_01.c" 입력
    - 나노 편집기를 이용하여 01_LED_01.c 파일 생성

```
pi@raspberryPi:~/Example $ nano 01_LED_01.c
```

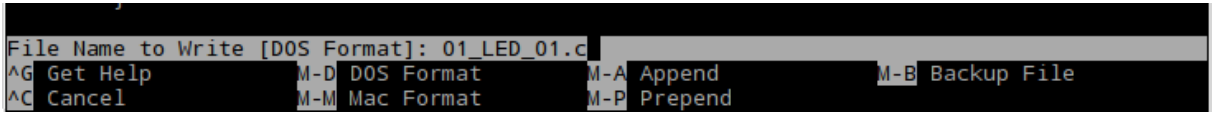
GPIO를 활용한 제어

- LED 동작 제어 예제(1)
  - 21번 핀과 연결된 LED를 1초간 On/Off 를 반복하는 예제

```
1.  // File : 01_LED_01.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>                    // wiringPi 헤더파일 참조

4.  const int pinLed = 21;                   // LED가 연결된 21번 핀(BCM Mode)

5.  int main(void)
6.  {
7.      wiringPiSetupGpio();                // 핀 번호를 BCM Mode로 설정

8.      pinMode(pinLed, OUTPUT);            // LED 핀을 출력모드로 설정

9.      while(1)
10.     {
11.         digitalWrite(pinLed, HIGH);     // LED 핀의 상태를 HIGH로 변경
12.         delay(1000);                    // 1000msec 지연
13.         digitalWrite(pinLed, LOW);      // LED 핀의 상태를 LOW로 변경
14.         delay(1000);
15.     }
16.     return 0;
17. }
```

GPIO를 활용한 제어

- LED 동작 제어 예제(1)
  - 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료



  - GCC 컴파일러를 사용하여 빌드

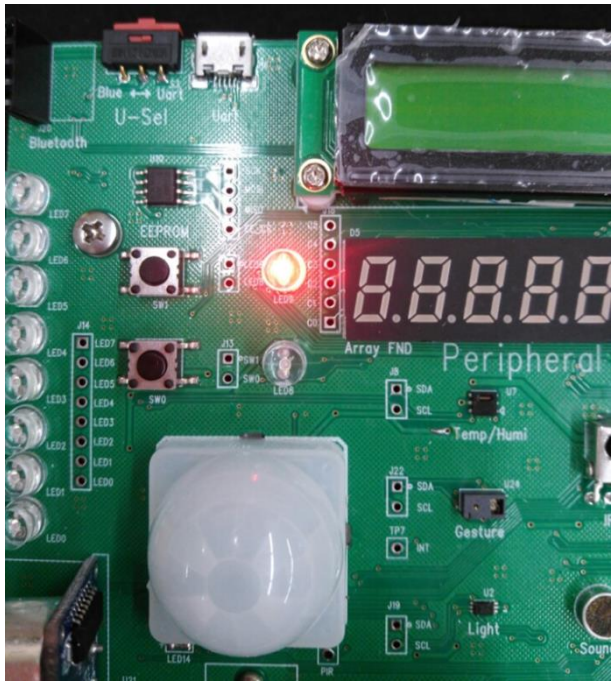
```
pi@raspberryPi:~/Example $ gcc -o 01_LED_01 01_LED_01.c -lwiringPi
```

  - 생성된 "01_LED_01" 파일 실행

```
pi@raspberryPi:~/Example $ ./01_LED_01
```

GPIO를 활용한 제어

- LED 동작 제어 예제(1)
  - 결과
    - 해당 LED 가 1초마다 On/Off 를 반복
    - "ctrl + C"를 이용하여 종료



GPIO를 활용한 제어

- LED 동작 제어 예제(2)
  - 터미널 창에 "nano 01_LED_02.c" 입력
    - 나노 편집기를 이용하여 01_LED_02.c 파일 생성

```
pi@raspberryPi:~/Example $ nano 01_LED_02.c
```

  - LED 2개를 이용해서 1초 간격으로 반전 On/Off 하는 예제

```
1.  // File : 01_LED_02.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>

4.  const int pinLed1 = 21;                 // LED가 연결된 21번, 20번 핀(BCM Mode)
5.  const int pinLed2 = 20;

6.  int main(void)
7.  {
8.      wiringPiSetupGpio();                // 핀 번호를 BCM Mode로 설정

9.      pinMode(pinLed1, OUTPUT);           // LED1~2 핀을 출력모드로 설정
10.     pinMode(pinLed2, OUTPUT);
```

GPIO를 활용한 제어

- LED 동작 제어 예제(2)

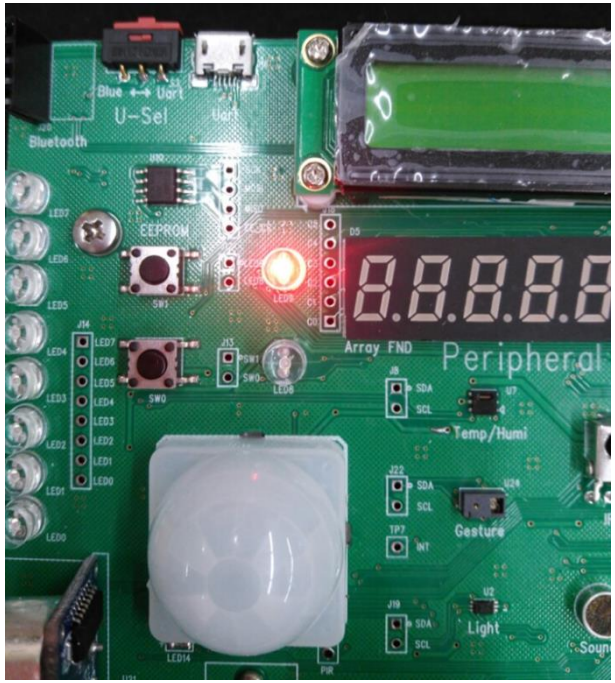
```
11.     while(1)
12.     {
13.         digitalWrite(pinLed1, HIGH);      // LED1 핀의 상태를 HIGH로 변경
14.         digitalWrite(pinLed2, LOW);       // LED2 핀의 상태를 LOW로 변경
15.         delay(1000);                      // 1000msec 지연
16.         digitalWrite(pinLed1, LOW);       // LED1 핀의 상태를 LOW로 변경
17.         digitalWrite(pinLed2, HIGH);      // LED2 핀의 상태를 HIGH로 변경
18.         delay(1000);
19.      }
20.      return 0;
21. }
```

- 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
- GCC 컴파일러를 사용하여 빌드 및 생성된 "01_LED_02" 파일 실행

```
pi@raspberryPi:~/Example $ gcc -o 01_LED_02 01_LED_02.c -lwiringPi
pi@raspberryPi:~/Example $ ./01_LED_02
```

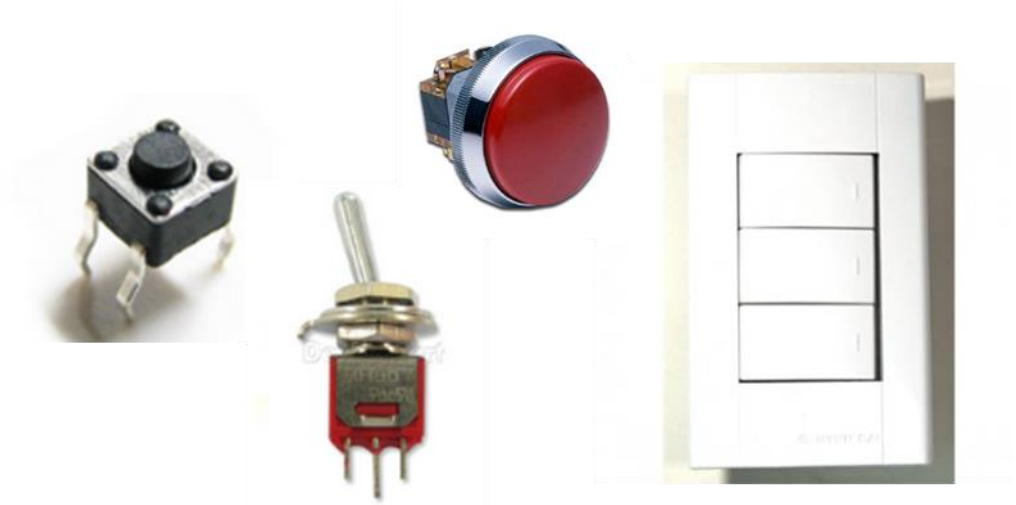
GPIO를 활용한 제어

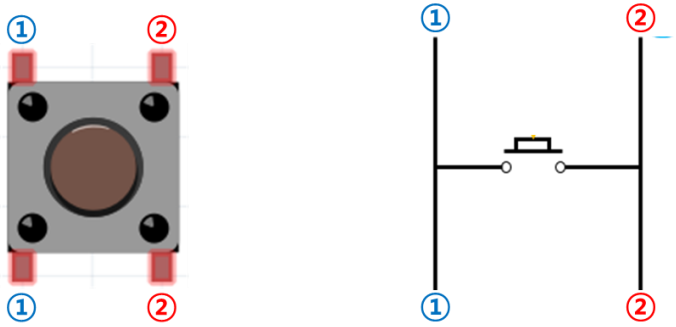
- LED 동작 제어 예제(2)
  - 결과
    - LED 2개가 1초마다 1개씩 반전되어 On/Off 를 반복
    - "ctrl + C"를 이용하여 종료



GPIO를 활용한 제어

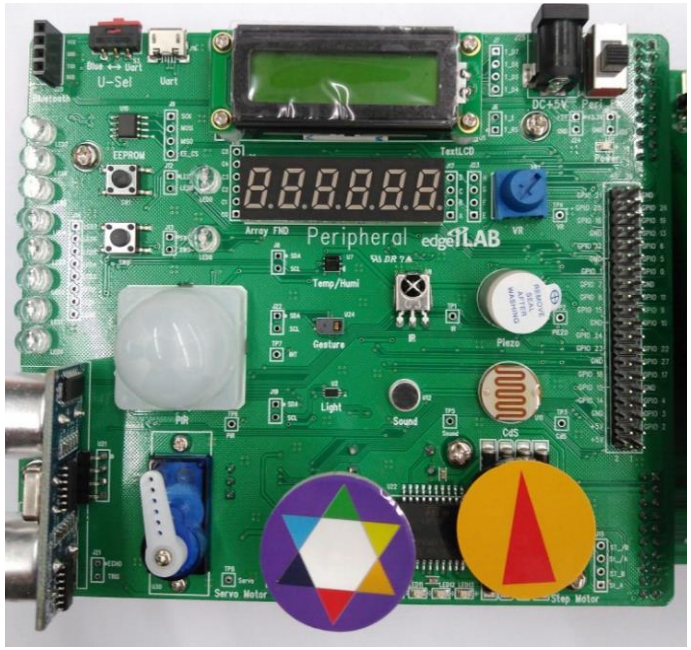
- SWITCH 동작 제어
  - 스위치(Switch)는 전원을 On/Off 하거나 사용하고자 하는 기능의 On/Off에 주로 사용
  - 스위치 중, Tact 스위치는 누르면 On 때면 Off 되는 스위치를 말함





GPIO를 활용한 제어

- SWITCH 동작 제어
  - SWITCH는 핀 29, 30 번(BCM Mode)와 연결
  - 입력 값에 따라 On/Off 판단
    - 누르면 HIGH(3.3V) 혹은 떼면 LOW(0V)



Connect	BCM	Name	Physical		Name	BCM	Connect
		+3.3V	1	2	+5V		
I2C	GPIO 02	SDA1	3	4	+5V		
I2C	GPIO 03	SCL1	5	6	GND		
DC_M	GPIO 04	GPIO	7	8	TxD	GPIO 14	UART
		GND	9	10	RxD	GPIO 15	UART
SERVO	GPIO 17	GPIO	11	12	GPIO	GPIO 18	TLCD
TLCD	GPIO 27	GPIO	13	14	GND		
TLCD	GPIO 22	GPIO	15	16	GPIO	GPIO 23	TLCD
		+3.3V	17	18	GPIO	GPIO 24	PIR
SPI	GPIO 10	MOSI	19	20	GND		
SPI	GPIO 09	MISO	21	22	GPIO	GPIO 25	DC_M
SPI	GPIO 11	SCLK	23	24	CE0	GPIO 08	
		GND	25	26	CE1	GPIO 07	SPI
ULTRA	GPIO 00	SDA0	27	28	SCL0	GPIO 01	ULTRA
SWITCH	GPIO 05	GPIO	29	30	GND		
SWITCH	GPIO 06	GPIO	31	32	GPIO	GPIO 12	DC_M
PIEZO	GPIO 13	GPIO	33	34	GND		
IR	GPIO 19	GPIO	35	36	GPIO	GPIO 16	TLCD
TLCD	GPIO 26	GPIO	37	38	GPIO	GPIO 20	LED
		GND	39	40	GPIO	GPIO 21	LED

GPIO를 활용한 제어

- SWITCH 동작 제어 예제(1)
  - 터미널 창에 "nano 02_SWITCH_01.c" 입력
    - 나노 편집기를 이용하여 02_SWITCH_01.c 파일 생성

```
pi@raspberryPi:~/Example $ nano 02_SWITCH_01.c
```

  - digitalRead()함수를 이용해 스위치에 연결된 GPIO의 상태를 반복적으로 확인하고, 상태가 "HIGH"로 변하면 터미널에 "Pushed"를 출력하는 예제

```
1.  // File : 02_SWITCH_01.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>

4.  const int pinSwitch = 6;                  // SWITCH가 연결된 6번 핀(BCM Mode)

5.  int main(void)
6.  {
7.      wiringPiSetupGpio();                  // 핀 번호를 BCM Mode로 설정

8.      pinMode(pinSwitch, INPUT);            // SWITCH 핀을 입력모드로 설정
```

GPIO를 활용한 제어

- SWITCH 동작 제어 예제(1)

```
9.      while(1)
10.     {
11.         if(digitalRead(pinSwitch))          // SWITCH 핀의 상태를 확인
12.         {
13.             printf("Pushed₩n");             // HIGH일 경우, "Pushed" 출력
14.             delay(500);
15.         }
16.     }
17.     return 0;
18. }
```

- 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
- GCC 컴파일러를 사용하여 빌드 및 생성된 "02_SWITCH_01" 파일 실행

```
pi@raspberryPi:~/Example $ gcc -o 02_SWITCH_01 02_SWITCH_01.c -lwiringPi
pi@raspberryPi:~/Example $ ./02_SWITCH_01
```

GPIO를 활용한 제어

- SWITCH 동작 제어 예제(1)
  - 결과
    - 스위치를 누르면 터미널 창에 "Pushed" 출력
    - "ctrl + C"를 이용하여 종료

```
pi@raspberrypi:~/Example $ ./02_SWITCH_01
Pushed
Pushed
Pushed
```

GPIO를 활용한 제어

- SWITCH 동작 제어 예제(2)
  - 터미널 창에 "nano 02_SWITCH_02.c" 입력
    - 나노 편집기를 이용하여 02_SWITCH_02.c 파일 생성

```
pi@raspberryPi:~/Example $ nano 02_SWITCH_02.c
```

  - SWITCH 2개 중 임의의 스위치를 눌러 해당 스위치 번호를 출력하는 예제

```
1.  // File : 02_SWITCH_02.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>

4.  const int aPinSwitch[2] = {6, 5};          // SWITCH가 연결된 6,5 번 핀을 배열로 정의

5.  int main(void)
6.  {
7.      wiringPiSetupGpio();                   // 핀 번호를 BCM Mode로 설정

8.      int i;
9.      for(i=0; i<2; i++)
10.     {
11.         pinMode(aPinSwitch[i], INPUT);     // SWITCH1~2핀을 입력모드로 설정
12.     }
```

GPIO를 활용한 제어

- SWITCH 동작 제어 예제(2)

```
13.     while(1)
14.     {
15.         for(i=0; i<2; i++)
16.         {
17.             if(digitalRead(aPinSwitch[i]))              // 임의의 SWITCH를 누르면
18.             {
19.                 printf("SWITCH[%d] Pushed₩n", i);    // 해당하는 번호와 함께 출력
20.                 delay(500);
21.             }
22.         }
23.     }
24.     return 0;
25. }
```

- 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
- GCC 컴파일러를 사용하여 빌드 및 생성된 "02_SWITCH_02" 파일 실행

```
pi@raspberryPi:~/Example $ gcc -o 02_SWITCH_02 02_SWITCH_02.c -lwiringPi
pi@raspberryPi:~/Example $ ./02_SWITCH_02
```

GPIO를 활용한 제어

- SWITCH 동작 제어 예제(2)
  - 결과
    - 해당 SWITCH를 누르면 터미널 화면에 해당 스위치 번호와 함께 “Pushed” 출력

```
SWITCH[1] Pushed
SWITCH[0] Pushed
SWITCH[1] Pushed
SWITCH[0] Pushed
SWITCH[1] Pushed
```

GPIO를 활용한 제어

- SWITCH 동작 제어 예제(3)
  - 터미널 창에 "nano 02_SWITCH_03.c" 입력
    - 나노 편집기를 이용하여 02_SWITCH_02.c 파일 생성

```
pi@raspberryPi:~/Example $ nano 02_SWITCH_03.c
```

  - SWITCH와 LED를 이용해서 SWITCH를 한번 누르면 해당 번호의 LED가 켜지고, 손을 떼면 LED가 꺼지는 예제

```
1.  // File : 02_SWITCH_03.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>

4.  const int aPinSwitch[2] = {6, 5};        // SWITCH가 연결된 6, 5 번 핀을 배열로 정의
5.  const int aPinLed[2] = {21, 20};         // LED가 연결된 21, 20번 핀을 배열로 정의

6.  int main(void)
7.  {
8.      wiringPiSetupGpio();                 // 핀 번호를 BCM Mode로 설정

9.      int i;
```

GPIO를 활용한 제어

- SWITCH 동작 제어 예제(3)

```
10.     for(i=0; i<2; i++)
11.     {
12.         pinMode(aPinSwitch[i], INPUT);          // SWITCH1~2핀을 입력모드로 설정
13.         pinMode(aPinLed[i], OUTPUT);            // LED1~2핀을 출력모드로 설정
14.     }
15.     while(1)
16.     {
17.         for(i=0; i<2; i++)
18.         {
19.             if(digitalRead(aPinSwitch[i]))      // 임의의 스위치를 누를 경우
20.             {
21.                 digitalWrtie(aPinLed[i], HIGH); // 해당 번호 LED의 상태를 HIGH 로 변경
22.              }
23.         else                                    // 누르지 않을 경우
24.             {
25.                 digitalWrite(aPinLed[i], LOW);  // 해당 번호 LED의 상태를 LOW로 변경
26.             }
27.         }
28.     }
29. return 0;
30. }
```

GPIO를 활용한 제어

- SWITCH 동작 제어 예제(3)
  - 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
  - GCC 컴파일러를 사용하여 빌드 및 생성된 "02_SWITCH_03" 파일 실행

```
pi@raspberryPi:~/Example $ gcc -o 02_SWITCH_03 02_SWITCH_03.c -lwiringPi
pi@raspberryPi:~/Example $ ./02_SWITCH_03
```

  - 결과
    - SWITCH에 따라 LED가 On/Off 되는 것을 확인

GPIO를 활용한 제어

- SWITCH 동작 제어 예제(4)
  - 터미널 창에 "nano 02_SWITCH_04.c" 입력
    - 나노 편집기를 이용하여 02_SWITCH_04.c 파일 생성

```
pi@raspberryPi:~/Example $ nano 02_SWITCH_04.c
```

  - Interrup를 이용해 SWITCH를 누르면 해당 번호의 LED가 켜지고 다시 누르면 꺼지는 예제

```
1.  // File : 02_SWITCH_04.c
2.
3.  #include <stdio.h>
4.  #include <wiringPi.h>
5.
6.  const int aPinSwitch[2] = {6, 5};          // SWITCH가 연결된 6, 5 번 핀을 배열로 정의
7.  const int aPinLed[2] = {21, 20};           // LED가 연결된 21, 20번 핀을 배열로 정의
8.
9.  int ledFlag1 = 0;                         // 인터럽트 함수에서 사용될 변수 정의
10. int ledFlag2 = 0;
11. // 인터럽트 서비스 루틴 정의
12. void ledBlink1(void)                      // SWITCH1에 사용될 인터럽트 함수 정의
13. {
14.     if(ledFlag1 == 0)                     // Flag가 '0'일 경우
15.     {
```

GPIO를 활용한 제어

- SWITCH 동작 제어 예제(4)

```
16.          digitalWrite(aPinLed[0], HIGH);       // LED핀을 HIGH로 설정
17.          ledFlag1 = 1;                          // Flag를 '1'로 변경
18.      }

19.      else                                       // 아닐 경우
20.      {
21.          digitalWrite(aPinLed[0], LOW);        // LED 핀을 LOW로 설정
22.          ledFlag1 = 0;                          // Flag를 '0'으로 변경
23.      }
24.  }

25.  void ledBlink2(void)                           // SWITCH1과 동일
26.  {
27.      if(ledFlag2 == 0)
28.      {
29.          digitalWrite(aPinLed[1], HIGH);
30.          ledFlag2 = 1;
31.      }
32.      else
33.      {
34.          digitalWrite(aPinLed[1], LOW);
35.          ledFlag2 = 0;
36.      }
37.  }
```

GPIO를 활용한 제어

- SWITCH 동작 제어 예제(4)

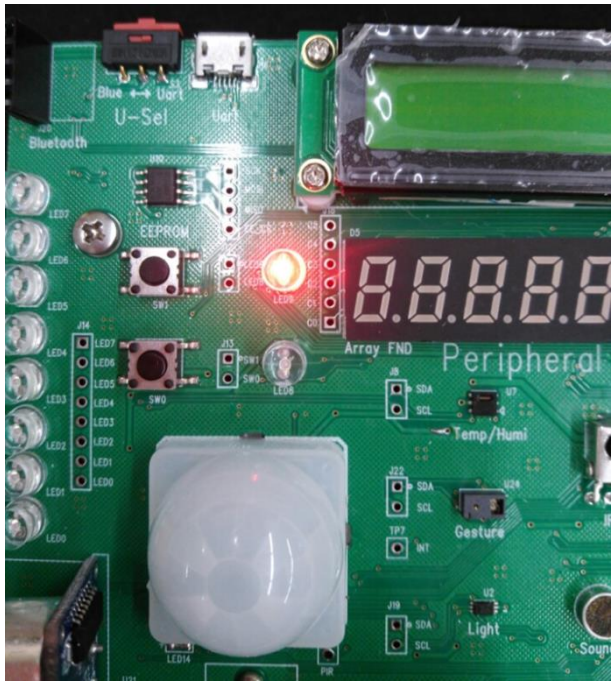
```
38. int main(void)
39. {
40.     wiringPiSetupGpio();                          // 핀 번호를 BCM Mode로 설정

41.     int i;
42.     for(i=0; i<2; i++)
43.     {                                             // for문을 이용해서
44.         pinMode(aPinSwitch[i], INPUT);            // SWITCH 1~2핀을 입력모드로 설정
45.         pinMode(aPinLed[i], OUTPUT);              // LED 1~2핀을 출력모드로 설정
46.     }

47.     // SWITCH1의 인터럽트를 설정, RISING Mode, ledBlink1 함수 호출
48.     wiringPiISR(aPinSwitch[0], INT_EDGE_RISING, ledBlink1);

49.     // SWITCH2의 인터럽트를 설정, FALLING Mode, ledBlink2 함수 호출
50.     wiringPiISR(aPinSwitch[1], INT_EDGE_FALLING, ledBlink2);

51.     while(1)
52.     {
53.     }
54.     return 0;
55. }
```

GPIO를 활용한 제어

- SWITCH 동작 제어 예제(4)
  - 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
  - GCC 컴파일러를 사용하여 빌드 및 생성된 "02_SWITCH_04" 파일 실행

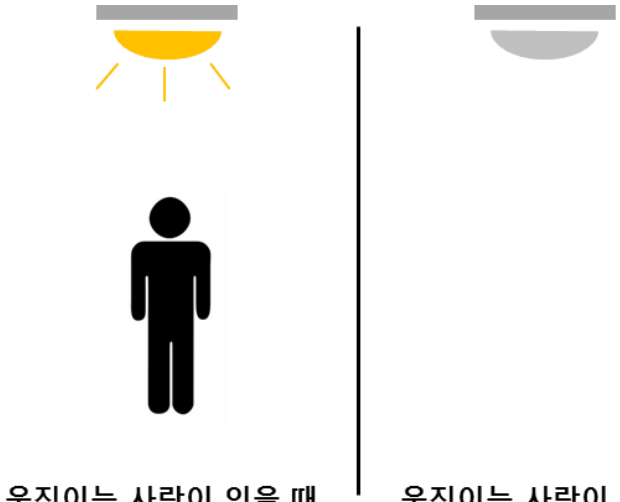
```
pi@raspberryPi:~/Example $ gcc -o 02_SWITCH_04 02_SWITCH_04.c -lwiringPi
pi@raspberryPi:~/Example $ ./02_SWITCH_04
```

  - 결과
    - SWITCH를 누르면 같은 번호의 LED가 켜지고 다시 누르면 꺼짐



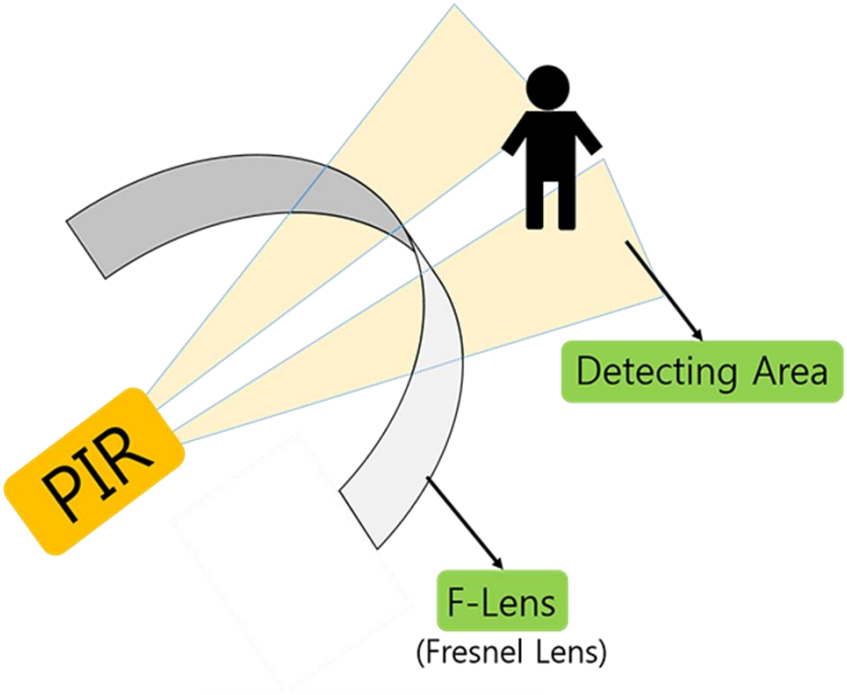
GPIO를 활용한 제어

- PIR 제어
  - 적외선을 이용하여 인체의 움직임을 감지하는 센서
  - 모션이나 동작 인식센서로 사용
  - 소형, 저가격, 저전력에 사용법 간단
  - 주로 화장실의 LED 자동 On/Off,
  - CCTV와 결합하여 움직임이 감지 됐을 때, CCTV 촬영 등에 사용



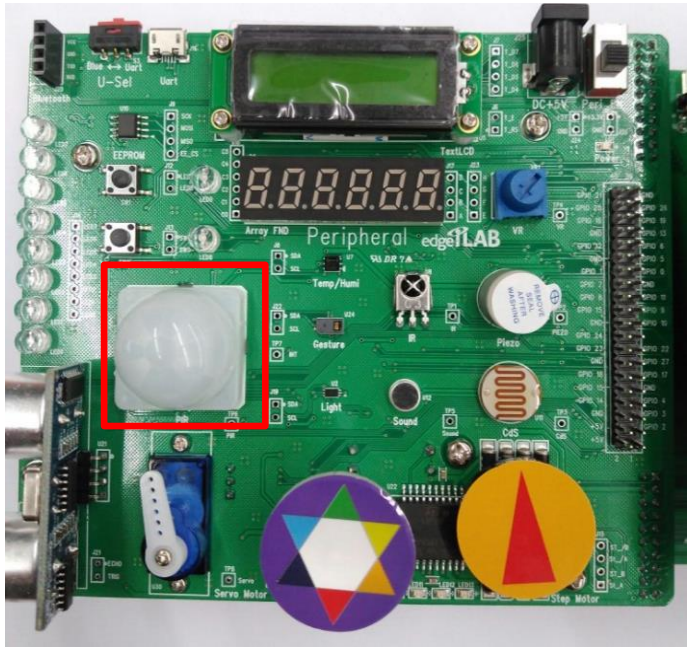
GPIO를 활용한 제어

- PIR 제어
  - 적외선에 민감한 특수물질로 만들어진 두 개의 슬롯
  - 일정 범위 내에 체온을 가진 사물이 지나가면 한 슬롯이 변화를 감지
  - 범위를 벗어나면 반대 현상이 일어나 모션 측정이 일어나는 원리



GPIO를 활용한 제어

- PIR 제어
  - PIR은 핀 24번(BCM Mode)과 연결
  - 감지 되지 않은 상태 : HIGH
  - 감지 된 상태 : LOW



Connect	BCM	Name	Physical		Name	BCM	Connect
		+3.3V	1	2	+5V		
I2C	GPIO 02	SDA1	3	4	+5V		
I2C	GPIO 03	SCL1	5	6	GND		
DC_M	GPIO 04	GPIO	7	8	TxD	GPIO 14	UART
		GND	9	10	RxD	GPIO 15	UART
SERVO	GPIO 17	GPIO	11	12	GPIO	GPIO 18	TLCD
TLCD	GPIO 27	GPIO	13	14	GND		
TLCD	GPIO 22	GPIO	15	16	GPIO	GPIO 23	TLCD
		+3.3V	17	18	GPIO	GPIO 24	PIR
SPI	GPIO 10	MOSI	19	20	GND		
SPI	GPIO 09	MISO	21	22	GPIO	GPIO 25	DC_M
SPI	GPIO 11	SCLK	23	24	CE0	GPIO 08	
		GND	25	26	CE1	GPIO 07	SPI
ULTRA	GPIO 00	SDA0	27	28	SCL0	GPIO 01	ULTRA
SWITCH	GPIO 05	GPIO	29	30	GND		
SWITCH	GPIO 06	GPIO	31	32	GPIO	GPIO 12	DC_M
PIEZO	GPIO 13	GPIO	33	34	GND		
IR	GPIO 19	GPIO	35	36	GPIO	GPIO 16	TLCD
TLCD	GPIO 26	GPIO	37	38	GPIO	GPIO 20	LED
		GND	39	40	GPIO	GPIO 21	LED

GPIO를 활용한 제어

- PIR 제어 예제(1)
  - 터미널 창에 "nano 03_PIR_01.c" 입력

```
pi@raspberryPi:~/Example $ nano 03_PIR_01.c
```

  - digitalRead()함수를 이용해 물체가 감지 됐을 때 터미널 창에 "Detected" 문구를 출력하는 예제

```
1.  // File : 03_PIR_01.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>

4.  const int pinPir = 24;                    // PIR이 연결된 24 번 핀

5.  int main(void)
6.  {
7.      wiringPiSetupGpio();                  // 핀 번호를 BCM Mode로 설정

8.      pinMode(pinPir, INPUT);
```

GPIO를 활용한 제어

- PIR 제어 예제(1)

```
9.      while(1)
10.     {
11.         if(!digitalRead(pinPir))                    // 동작 인식이 될 경우, LOW 출력
12.         {
13.             printf("Detected₩n");
14.             delay(500);
15.         }
16.     }
17.     return 0;
18. }
```

- 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
- GCC 컴파일러를 사용하여 빌드 및 생성된 "03_PIR_01" 파일 실행

```
pi@raspberryPi:~/Example $ gcc -o 03_PIR_01 03_PIR_01.c -lwiringPi
pi@raspberryPi:~/Example $ ./03_PIR_01
```

GPIO를 활용한 제어

- PIR 제어 예제(1)
  - 결과
    - PIR 근처에서 사람의 움직임을 감지하면 "Detected" 문구 출력

```
pi@raspberrypi:~/Example $ ./pir
Detected
Detected
Detected
Detected
Detected
Detected
```

GPIO를 활용한 제어

- PIR 제어 예제(2)
  - 터미널 창에 "nano 03_PIR_02.c" 입력

```
pi@raspberryPi:~/Example $ nano 03_PIR_02.c
```

  - LED 2개를 이용하여 평상시에는 LED 1이 켜져 있다가, 동작 감지가 일어나면 LED1이 꺼지고 LED2가 켜지는 예제

```
1.  // File : 03_PIR_02.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>

4.  const int pinPir = 24;
5.  const int aPinLed[2] = {21, 20};

6.  int main(void)
7.  {
8.      wiringPiSetupGpio();

9.      pinMode(pinPir, INPUT);

10.     int i;
```

GPIO를 활용한 제어

- PIR 제어 예제(2)

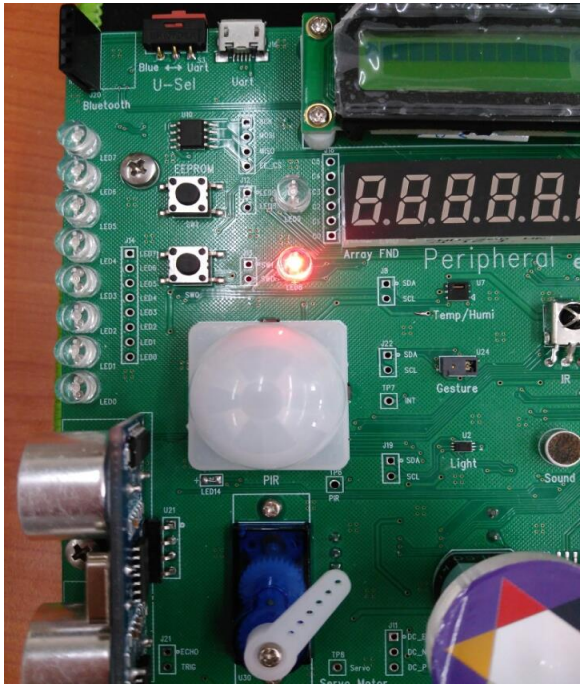
```
11.     for(i=0; i<2; i++)
12.     {
13.         pinMode(aPinLed[i], OUTPUT);
14.     }

15.     while(1)
16.     {
17.         if(!digitalRead(pinPir))                    // 동작 인식할 경우
18.         {
19.             digitalWrite(aPinLed[0], LOW);
20.             digitalWrite(aPinLed[1], HIGH);         // LED2핀의 상태를 HIGH로 변경
21.         }
22.         else                                        // 동작 인식 하지 않을 경우
23.         {
24.             digitalWrite(aPinLed[0], HIGH);         // LED1 핀의 상태를 HIGH로 변경
25.             digitalWrite(aPinLed[1], LOW);
26.         }
27.         delay(500);
28.     }
29.     return 0;
30. }
```

GPIO를 활용한 제어

- PIR 제어 예제(2)
  - 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
  - GCC 컴파일러를 사용하여 빌드 및 생성된 "03_PIR_02" 파일 실행

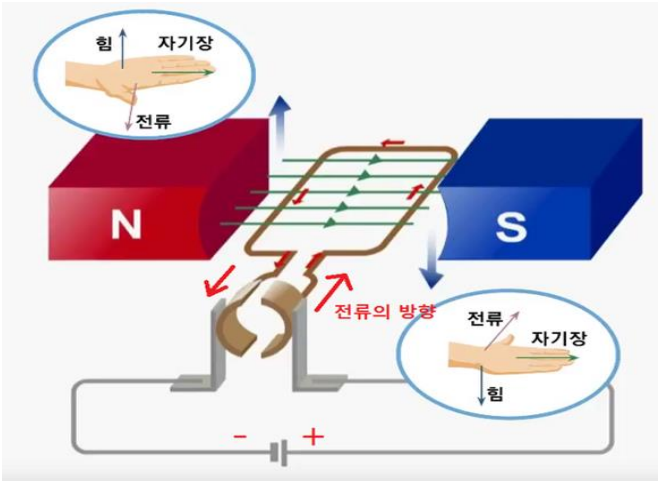
```
pi@raspberryPi:~/Example $ gcc -o 03_PIR_02 03_PIR_02.c -lwiringPi
pi@raspberryPi:~/Example $ ./03_PIR_02
```

  - 결과
    - PIR 근처에서 사람의 움직임을 감지하면 LED 2개가 반전



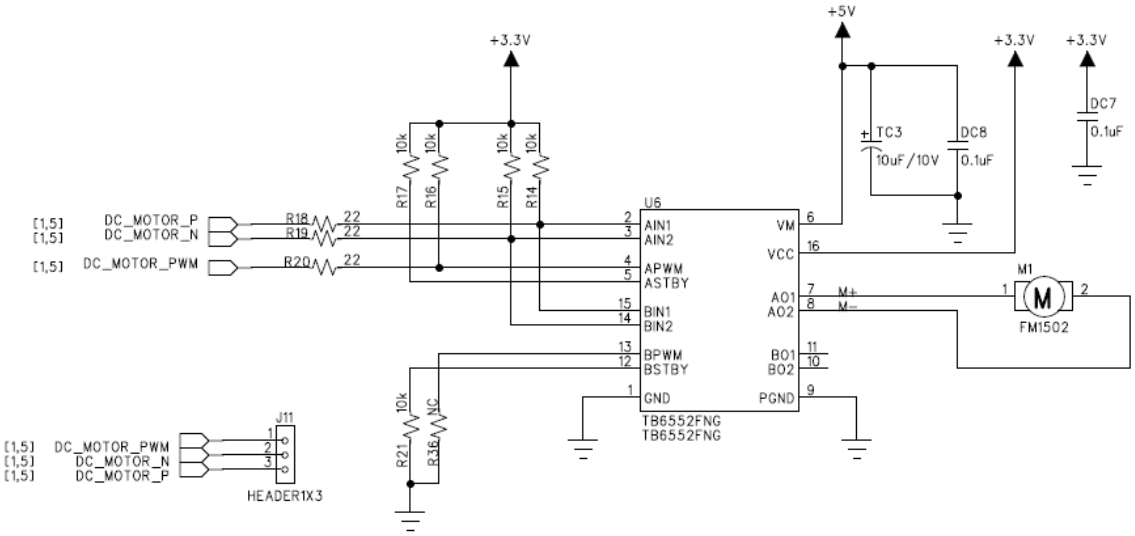
GPIO를 활용한 제어

- DC MOTOR 제어
  - 모터란 전기에너지를 기계에너지로 바꾸는 동력 기계장치를 말함
  - 전력을 받아 회전하고, 축에 회전력을 발생
  - DC모터는 직류전원으로 동작
  - 외부의 고정된 부분에 고정자를 영구자석으로 사용
  - 회전자(전기자)로 코일을 사용
  - 회전체에 흐르는 전류의 방향을 전환하여 발생하는 자기장과 자력의 반발력을 이용하여 회전력을 얻음
  - 다른 구동장치에 비해 가볍고 구조가 간단
  - DC모터는 양방향으로 동작(전류의 방향에 따라 회전방향 변경)
  - 회전속도는 코일에 흐르는 전류와 정비례



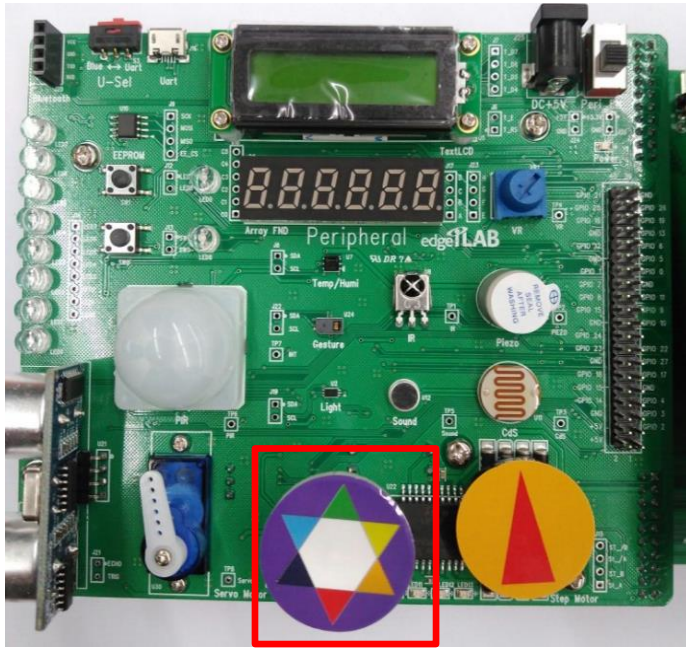
GPIO를 활용한 제어

- DC MOTOR 제어
  - TB6552FNG 칩에 DC모터의 Positive, Negative 연결
  - DC 모터용 드라이버 IC, PWM 구동 방식 지원
  - PWM 핀을 이용해 DC 모터의 회전 속도 조절 및 동작 여부 설정



GPIO를 활용한 제어

- DC MOTOR 제어
  - DC 모터는 핀 4, 12, 25 번(BCM Mode)과 연결
    - GPIO 04 : Positive 핀
    - GPIO 25 : Negative 핀
    - GPIO 12 : Enable 핀



Connect	BCM	Name	Physical		Name	BCM	Connect	
		+3.3V	1	2	+5V			
I2C	GPIO 02	SDA1	3	4	+5V			
I2C	GPIO 03	SCL1	5	6	GND			
DC_M	GPIO 04	GPIO	7	8	TxD	GPIO 14	UART	
		GND	9	10	RxD	GPIO 15	UART	
SERVO	GPIO 17	GPIO	11	12	GPIO	GPIO 18	TLCD	
TLCD	GPIO 27	GPIO	13	14	GND			
TLCD	GPIO 22	GPIO	15	16	GPIO	GPIO 23	TLCD	
		+3.3V	17	18	GPIO	GPIO 24	PIR	
SPI	GPIO 10	MOSI	19	20	GND			
SPI	GPIO 09	MISO	21	22	GPIO	GPIO 25	DC_M	
SPI	GPIO 11	SCLK	23	24	CE0	GPIO 08		
		GND	25	26	CE1	GPIO 07	SPI	
ULTRA	GPIO 00	SDA0	27	28	SCL0	GPIO 01	ULTRA	
SWITCH	GPIO 05	GPIO	29	30	GND			
SWITCH	GPIO 06	GPIO	31	32	GPIO	GPIO 12	DC_M	
PIEZO	GPIO 13	GPIO	33	34	GND			
IR	GPIO 19	GPIO	35	36	GPIO	GPIO 16	TLCD	
TLCD	GPIO 26	GPIO	37	38	GPIO	GPIO 20	LED	
		GND	39	40	GPIO	GPIO 21	LED	

Positive 핀

Negative 핀

Enable 핀

GPIO를 활용한 제어

- DC MOTOR 제어 예제(1)
  - 터미널 창에 "nano 04_DCMOTOR_01.c" 입력

```
pi@raspberryPi:~/Example $ nano 04_DCMOTOR_01.c
```

  - DC모터를 정방향, 정지, 역방향, 정지 순으로 2초마다 작동하는 예제

```
1.  // File : 04_DCMOTOR_01.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>

4.  const int pinEnable = 12;          // DC Motor Enable 핀 설정
5.  const int pinPositive = 4;         // DC Motor Positive 핀 설정
6.  const int pinNegative = 25;        // DC Motor Negative 핀 설정

7.  int main(void)
8.  {
9.      wiringPiSetupGpio();

10.     pinMode(pinEnable, OUTPUT);
11.     pinMode(pinPositive, OUTPUT);
12.     pinMode(pinNegative, OUTPUT);
```

GPIO를 활용한 제어

- DC MOTOR 제어 예제(1)

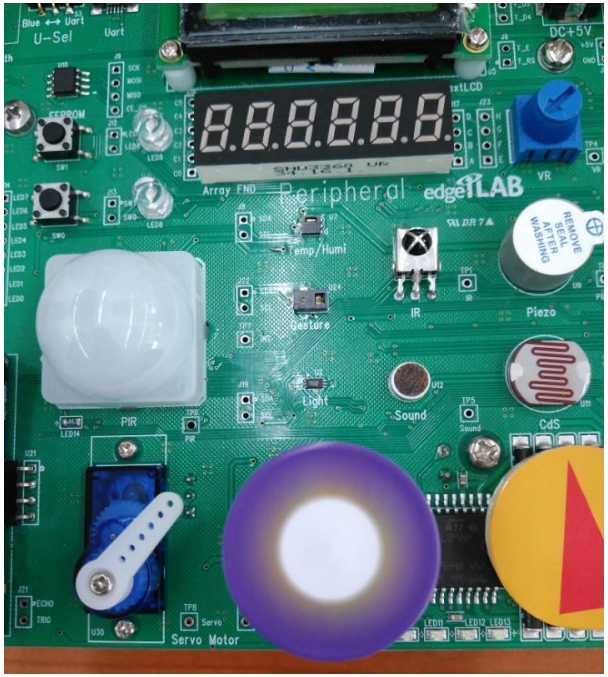
```
13.     while(1)
14.     {
15.         // 정방향 회전
16.         digitalWrite(pinPositive, HIGH);        // Positive 핀을 HIGH 상태로 변경
17.         digitalWrite(pinNegative, LOW);         //  Negative 핀을 LOW 상태로 변경
18.         digitalWrite(pinEnable, HIGH);
19.         delay(2000);

20.         // 정지
21.         digitalWrite(pinEnable, LOW);
22.         delay(2000);

23.         // 역방향 회전
24.         digitalWrite(pinPositive, LOW);         // Positive 핀을 LOW 상태로 변경
25.         digitalWrite(pinNegative, HIGH);        //  Negative 핀을 HIGH상태로 변경
26.         digitalWrite(pinEnable, HIGH);
27.         delay(2000);

28.         // 정지
29.         digitalWrite(pinEnable, LOW);
30.         delay(2000);
31.     }
32.     return 0;
33. }
```

GPIO를 활용한 제어

- DC MOTOR 제어 예제(1)
  - 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
  - GCC 컴파일러를 사용하여 빌드 및 생성된 "04_DCMOTOR_01" 파일 실행

```
  pi@raspberryPi:~/Example $ gcc -o 04_DCMOTOR_01 04_DCMOTOR_01.c
-lwiringPi
  pi@raspberryPi:~/Example $ ./04_DCMOTOR_01
```

  - 결과
    - DC 모터가 정방향 회전, 정지, 역방향 회전, 정지 순으로 동작 반복



GPIO를 활용한 제어

- DC MOTOR 제어 예제(2)
  - 터미널 창에 "nano 04_DCMOTOR_02.c" 입력

```
pi@raspberryPi:~/Example $ nano 04_DCMOTOR_02.c
```

  - 두개의 SWITCH 버튼을 이용하여 SWITCH 1을 누르면 회전, SWITCH 2를 누르면 정지 하는 예제

```
1.  // File : 04_DCMOTOR_02.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>

4.  const int pinEnable = 12;                // DC Motor Enable 핀 설정
5.  const int pinPositive = 4;               // DC Motor Positive 핀 설정
6.  const int pinNegative = 25;              // DC Motor Negative 핀 설정
7.  const int aPinSwitch[2] = {6, 5};        // SWITCH 핀 설정

8.  int main(void)
9.  {
10.     wiringPiSetupGpio();

11.     pinMode(pinEnable, OUTPUT);
12.     pinMode(pinPositive, OUTPUT);
13.     pinMode(pinNegative, OUTPUT);
```

GPIO를 활용한 제어

- DC MOTOR 제어 예제(2)

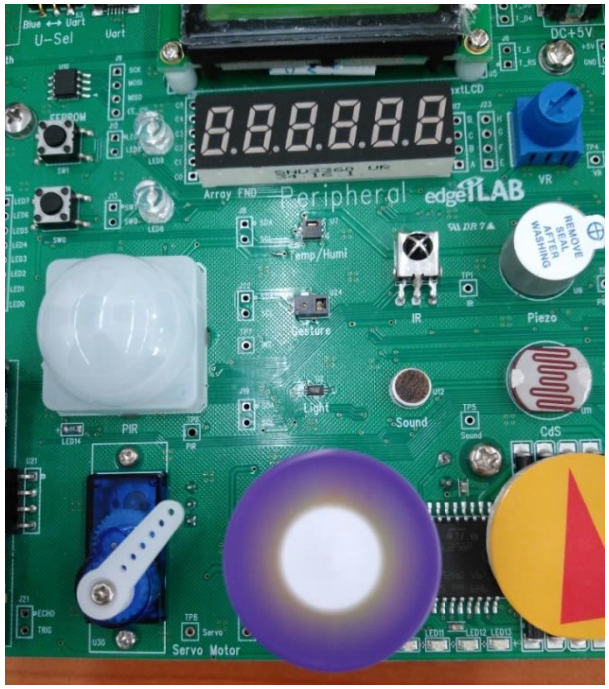
```
14.     int i;

15.     for(i=0; i<2; i++)
16.     {
17.         pinMode(aPinSwitch[i], INPUT);        // SWITCH 입력모드로 설정
18.     }

19.     while(1)
20.     {
21.         if(digitalRead(aPinSwitch[0]))          // SWITCH1번 누르면 회전
22.         {
23.             digitalWrite(pinPositive, HIGH);
24.             digitalWrite(pinNegative, LOW);
25.             digitalWrite(pinEnable, HIGH);
26.         }
27.         else if(digitalRead(aPinSwitch[1])     // SWITCH2번 누르면 정지
28.         {
29.             digitalWrite(pinEnable, LOW);
30.         }
31.     }
32.     return 0;
33. }
```

GPIO를 활용한 제어

- DC MOTOR 제어 예제(2)
  - 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
  - GCC 컴파일러를 사용하여 빌드 및 생성된 "04_DCMOTOR_02" 파일 실행

```
pi@raspberryPi:~/Example $ gcc -o 04_DCMOTOR_02 04_DCMOTOR_02.c
-lwiringPi
pi@raspberryPi:~/Example $ ./04_DCMOTOR_02
```

  - 결과
    - SWITCH 1을 누르면 회전, SWITCH 2를 누르면 정지



GPIO를 활용한 제어

- DC MOTOR 제어 예제(3)
  - 터미널 창에 "nano 04_DCMOTOR_03.c" 입력

```
pi@raspberryPi:~/Example $ nano 04_DCMOTOR_03.c
```

  - 인터럽트를 이용하여 SWITCH 1을 눌렀을 때(Rising) 회전, SWITCH 2를 눌렀다 떼면(Falling) 정지 하는 예제

```
1.  // File : 04_DCMOTOR_03.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>

4.  const int pinEnable = 12;              // DC Motor Enable 핀 설정
5.  const int pinPositive = 4;             // DC Motor Positive 핀 설정
6.  const int pinNegative = 25;            // DC Motor Negative 핀 설정
7.  const int aPinSwitch[2] = {6, 5};      // SWITCH 핀 설정

8.  // 인터럽트 서비스 루틴 정의
9.  void motorSpin(void)                   // DC Motor를 오른쪽으로 회전시키는 함수 정의
10. {
11.     digitalWrite(pinEnable, HIGH);
12.     digitalWrite(pinPositive, HIGH);
13.     digitalWrite(pinNegative, LOW);
14. }
```

GPIO를 활용한 제어

- DC MOTOR 제어 예제(3)

```
1.  void motorStop(void)                    // DC Motor를 정지시키는 함수 정의
2.  {
3.      digitalWrite(pinEnable, LOW);
4.  }

5.  int main(void)
6.  {
7.      wiringPiSetupGpio();

8.      pinMode(pinEnable, OUTPUT);
9.      pinMode(pinPositive, OUTPUT);
10.     pinMode(pinNegative, OUTPUT);

11.     int i;

12.     for(i=0; i<2; i++)
13.     {
14.         pinMode(aPinSwitch[i], INPUT);
15.     }

16.     // SWITCH1을 누를 경우, RISING Mode, motorSpin 함수 호출
17.     wiringPiISR(aPinSwitch[0], INT_EDGE_RISING, motorSpin);
18.     // SWITCH2을 누를 경우, FALLING Mode, motorStop함수 호출
19.     wiringPiISR(aPinSwitch[1], INT_EDGE_FALLING, motorStop);
```

GPIO를 활용한 제어

- DC MOTOR 제어 예제(3)

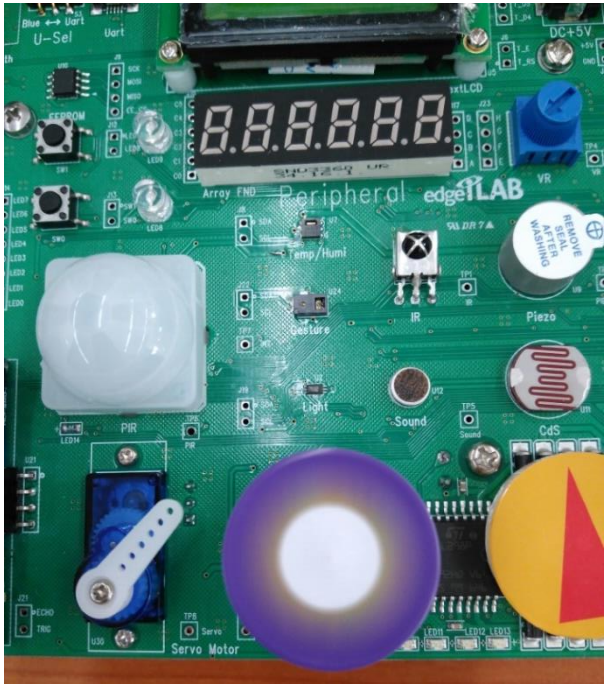
```
1.      while(1)
2.      {
3.      }
4.      return 0;
5.  }
```

- 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
- GCC 컴파일러를 사용하여 빌드 및 생성된 "04_DCMOTOR_03" 파일 실행

```
pi@raspberryPi:~/Example $ gcc -o 04_DCMOTOR_03 04_DCMOTOR_03.c
-lwiringPi
pi@raspberryPi:~/Example $ ./04_DCMOTOR_03
```

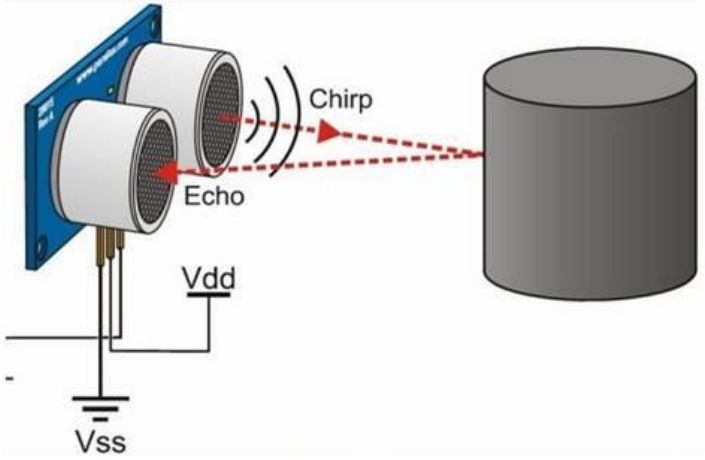
GPIO를 활용한 제어

- DC MOTOR 제어 예제(3)
  - 결과
    - SWITCH 1을 눌렀을 때, 인터럽트가 발생하여 DC 모터 회전
    - SWITCH 2를 눌렀다 뗐을 때, DC 모터 정지



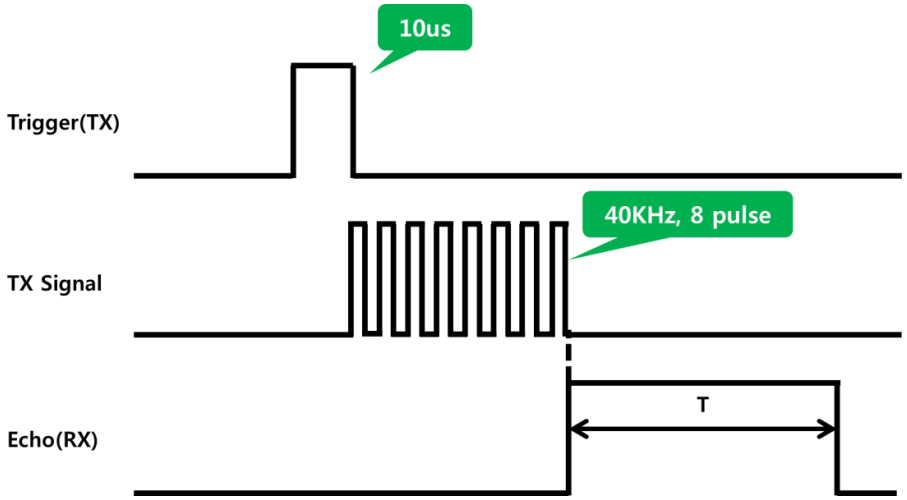
GPIO를 활용한 제어

- ULTRASONIC(초음파) 제어
  - 초음파란 인간이 들을 수 있는 영역(가청 영역, 20~20kHz) 보다 높은 음파를 말함
  - 주파수가 높고 파장이 짧아 강한 진동이 발생
  - 비파괴검사 기기 및 세척기 등 많은 분야에 활용
  - 초음파 센서는 Trigger와 Echo로 구성
  - Trigger는 초음파를 발생시켜 송신
  - Echo는 Trigger에서 발생한 초음파 수신
  - 송/수신의 시간차이를 이용하여 물체와의 거리를 계산하는 방식
  - 주파수가 높은 만큼 파장이 짧아 거리 방향의 분해력이 우수하고 정밀도가 높음



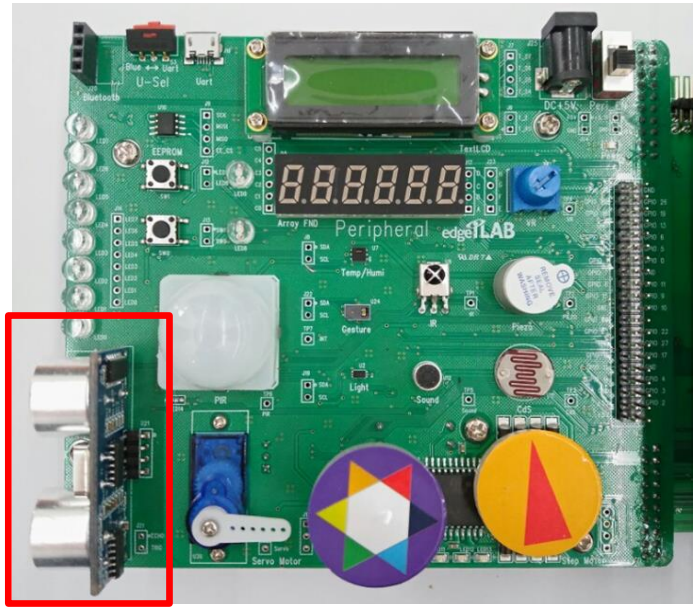
GPIO를 활용한 제어

- ULTRASONIC(초음파) 제어
  - Trigger에 5V 전압을 인가하면 초음파 모듈에서 8개의 40kHz 음파 발생
  - 이때, Echo는 HIGH상태가 되고, 음파가 물체에 맞아 수신되면 LOW상태로 바뀜
  - Echo가 HIGH에서 LOW로 상태가 변한 시간을 측정하여 거리를 구하는 원리
  - $t = \frac{2* \ Width(\text{물체와의 거리 } m)}{340 \ m/s(\text{초음파 속도})}$
  - t : 신호기 되돌아올 때 까지 걸리는 시간(s)
  - 거리가 1cm 인 경우 t = 2 * 0.01(m) / 340 (m/s) = 58.8us 가 됨
  - 즉, **거리(cm) = 왕복시간 / 58** 이라는 식이 도출



GPIO를 활용한 제어

- ULTRASONIC(초음파) 제어
  - ULTRASONIC은 핀 00, 01 번(BCM Mode)과 연결
    - GPIO 00 : Trigger 핀
    - GPIO 01 : Echo 핀



Connect	BCM	Name	Physical		Name	BCM	Connect
		+3.3V	1	2	+5V		
I2C	GPIO 02	SDA1	3	4	+5V		
I2C	GPIO 03	SCL1	5	6	GND		
DC_M	GPIO 04	GPIO	7	8	TxD	GPIO 14	UART
		GND	9	10	RxD	GPIO 15	UART
SERVO	GPIO 17	GPIO	11	12	GPIO	GPIO 18	TLCD
TLCD	GPIO 27	GPIO	13	14	GND		
TLCD	GPIO 22	GPIO	15	16	GPIO	GPIO 23	TLCD
		+3.3V	17	18	GPIO	GPIO 24	PIR
SPI	GPIO 10	MOSI	19	20	GND		
SPI	GPIO 09	MISO	21	22	GPIO	GPIO 25	DC_M
SPI	GPIO 11	SCLK	23	24	CE0	GPIO 08	
		GND	25	26	CE1	GPIO 07	SPI
ULTRA	GPIO 00	SDA0	27	28	SCL0	GPIO 01	ULTRA
SWITCH	GPIO 05	GPIO	29	30	GND		
SWITCH	GPIO 06	GPIO	31	32	GPIO	GPIO 12	DC_M
PIEZO	GPIO 13	GPIO	33	34	GND		
IR	GPIO 19	GPIO	35	36	GPIO	GPIO 16	TLCD
TLCD	GPIO 26	GPIO	37	38	GPIO	GPIO 20	LED
		GND	39	40	GPIO	GPIO 21	LED

Trigger 핀

Echo 핀

GPIO를 활용한 제어

- ULTRASONIC(초음파) 제어 예제(1)
  - 터미널 창에 "nano 05_ULTRASONIC_01.c" 입력

```
pi@raspberryPi:~/Example $ nano 05_ULTRASONIC_01.c
```

  - Trigger를 10us 이상동안 HIGH로 둬 초음파를 발생시키고, Echo로 수신되기까지의 시간을 micro()함수를 이용하여 구하는 예제

```
1.  // File : 05_ULTRASONIC_01.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>

4.  const int pinEcho = 1;                  // 초음파 센서의 수신부 핀 설정
5.  const int pinTrigger = 0;               // 초음파 센서의 송신부 핀 설정

6.  int main(void)
7.  {
8.      wiringPiSetupGpio();               // 핀 번호를 BCM Mode로 설정

9.      pinMode(pinEcho, INPUT);           // Echo핀을 입력 모드로 설정
10.     pinMode(pinTrigger, OUTPUT);       // Trigger핀을 출력 모드로 설정

11.     digitalWrite(pinTrigger, LOW);
12.     delay(30);
```

GPIO를 활용한 제어

- ULTRASONIC(초음파) 제어 예제(1)

```
13.     while(1)
14.     {
15.         digitalWrite(pinTrigger, HIGH);          // Trigger 핀을 20us 동안 HIGH로 출력
16.         delayMircoseconds(20);
17.         digitalWrite(pinTrigger, LOW);

18.         while(digitalRead(pinEcho) == LOW);      // 반사된 신호가 들어올 때 까지 대기
19.         long startTime = micros();               // 반사된 신호가 수신될 때 시간 초기화

20.         while(digitalRead(pinEcho) == HIGH);     // 반사된 신호가 없을 때 까지 대기
21.         long endTime = micros() – startTime;     // 반사된 신호가 없어진 직후 시간차 계산

22.         int distance = endTime / 58;             // 시간을 이용해 거리 계산

23.         printf("Distance : %d₩n", distance);
24.         delay(1000);
25.     }
26.     return 0;
27. }
```

GPIO를 활용한 제어

- ULTRASONIC(초음파) 제어 예제(1)
  - 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
  - GCC 컴파일러를 사용하여 빌드 및 생성된 "05_ULTRASONIC_01" 파일 실행

```
pi@raspberryPi:~/Example $ gcc -o 05_ULTRASONIC_01 05_ULTRASONIC_01.c
-lwiringPi
pi@raspberryPi:~/Example $ ./05_ULTRASONIC_01
```

  - 결과
    - 초음파 센서로부터 거리를 측정하여 화면에 출력

```
Distance : 22
Distance : 17
Distance : 23
Distance : 24
Distance : 27
```

GPIO를 활용한 제어

- ULTRASONIC(초음파) 제어 예제(2)
  - 터미널 창에 "nano 05_ULTRASONIC_02.c" 입력

```
pi@raspberryPi:~/Example $ nano 05_ULTRASONIC_02.c
```

  - Ultrasonic을 이용해 거리가 10cm 이내일 경우, LED가 켜지는 예제

```
1.  // File : 05_ULTRASONIC_02.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>

4.  const int pinEcho = 1;                  // 초음파 센서의 수신부 핀 설정
5.  const int pinTrigger = 0;               // 초음파 센서의 송신부 핀 설정

6.  const int pinLed = 21;

7.  int main(void)
8.  {
9.      wiringPiSetupGpio();                // 핀 번호를 BCM Mode로 설정

10.     pinMode(pinEcho, INPUT);            // Echo핀을 입력으로 설정
11.     pinMode(pinTrigger, OUTPUT);        // Trigger 핀을 출력으로 설정
12.     pinMode(pinLed, OUTPUT);            // LED 핀을 출력으로 설정
```

GPIO를 활용한 제어

- ULTRASONIC(초음파) 제어 예제(2)

```
1.      digitalWrite(pinTrigger, LOW);
2.      delay(30);
3.      while(1)
4.      {
5.          digitalWrite(pinTrigger, HIGH);          // Trigger 핀을 20us 동안 HIGH로 출력
6.          delayMircoseconds(20);
7.          digitalWrite(pinTrigger, LOW);

8.          while(digitalRead(pinEcho) == LOW);      // 반사된 신호가 들어올 때 까지 대기
9.          long startTime = micros();               // 반사된 신호가 수신될 때 시간 초기화

10.         while(digitalRead(pinEcho) == HIGH);     // 반사된 신호가 없을 때 까지 대기
11.         long endTime = micros() – startTime();   // 반사된 신호가 없어진 직후 시간차 계산

12.         int distance = endTime / 58;             // 시간을 이용해 거리 계산
13.         if(distance < 10)                        // 거리가 10cm 이내인 경우
14.         {
15.             digitalWrite(pinLed, HIGH);
16.         }
17.         else                                     // 그렇지 않을 경우
18.         {
19.             digitalWrite(pinLed, LOW);
20.         }
```

GPIO를 활용한 제어

- ULTRASONIC(초음파) 제어 예제(2)

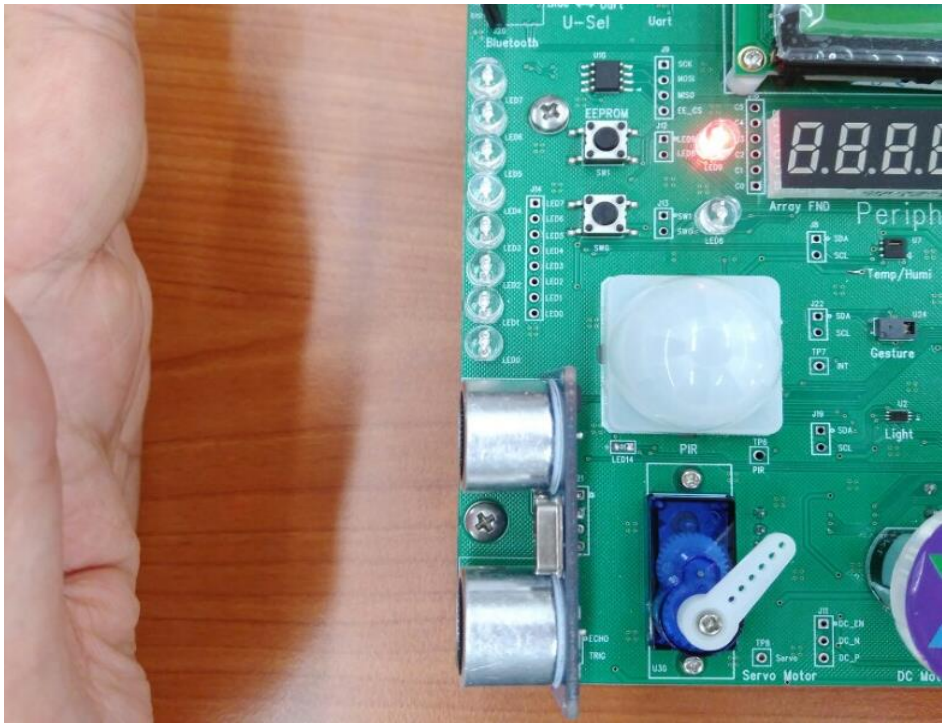
```
1.              delay(1000);
2.          }
3.          return 0;
4.      }
```

  - 작성 후 “crtl + o” 를 눌러 저장 및 “ctrl + X”를 눌러 종료
  - GCC 컴파일러를 사용하여 빌드 및 생성된 “05_ULTRASONIC_02” 파일 실행

```
pi@raspberryPi:~/Example $ gcc -o 05_ULTRASONIC_02 05_ULTRASONIC_02.c
-lwiringPi
pi@raspberryPi:~/Example $ ./05_ULTRASONIC_02
```

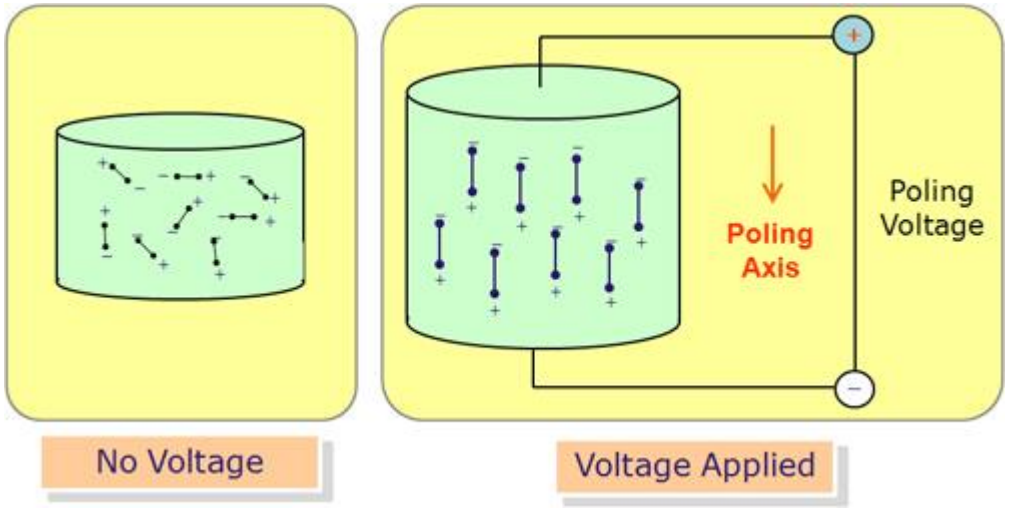
GPIO를 활용한 제어

- ULTRASONIC(초음파) 제어 예제(2)
  - 결과
    - 초음파 센서로부터 거리를 측정하여 10cm 이내인 경우, LED가 켜짐



GPIO를 활용한 제어

- PIEZO 제어
  - 피에조(Piezo)란 수정이나 세라믹 같은 결정체의 성질(압전 물질)을 이용한 소자
  - 압전 물질에 얇은 판을 닿게 하여 압전 효과에 의해 소리가 발생하는 원리
    - 압전 물질 : 어떤 결정에 힘을 가했을 때 전압이 발생
      이들의 물질에 반대로 전압이 인가하면 물리적 변위가 생기는 원리
    - 압전 효과 : 압력을 가했을 때 전압이 발생되고, 전압이 가해져 압력이
      생기는 효과
  - 저렴하고 단순하며 2만Hz 이상의 초음파 발생 가능
  - 큰 소리를 내지 못함



GPIO를 활용한 제어

- PIEZO 제어
  - 주파수에 따른 음계

(단위 : Hz)

옥타브 / 음계	1	2	3	4	5	6	7	8
C(도)	32.70	65.40	130.81	261.62	523.25	1046.50	2093.00	4186.00
C#	34.64	69.29	138.59	277.18	554.36	1108.73	2217.46	4435.92
D(레)	36.70	73.41	146.83	293.66	587.32	1174.65	2349.31	4698.63
D#	38.89	77.78	155.56	311.12	622.24	1244.50	2489.01	4978.03
E(미)	41.20	82.40	164.81	329.62	659.25	1318.51	2637.02	5274.04
F(파)	43.65	87.30	174.61	349.22	698.45	1396.91	2793.82	5587.65
F#	46.24	92.49	184.99	369.99	739.98	1479.97	2959.95	5919.91
G(솔)	48.99	97.99	195.99	391.99	783.99	1567.98	3135.96	6271.92
G#	51.91	103.82	207.65	415.30	830.60	1661.21	3322.43	6644.87
A(라)	55.00	110.00	220.00	440.00	880.00	1760.00	3520.00	7040.00
A#	58.27	116.54	233.08	466.16	932.32	1864.65	3729.31	7458.62
B(시)	61.73	123.47	246.94	493.88	987.76	1975.53	3951.06	7902.13

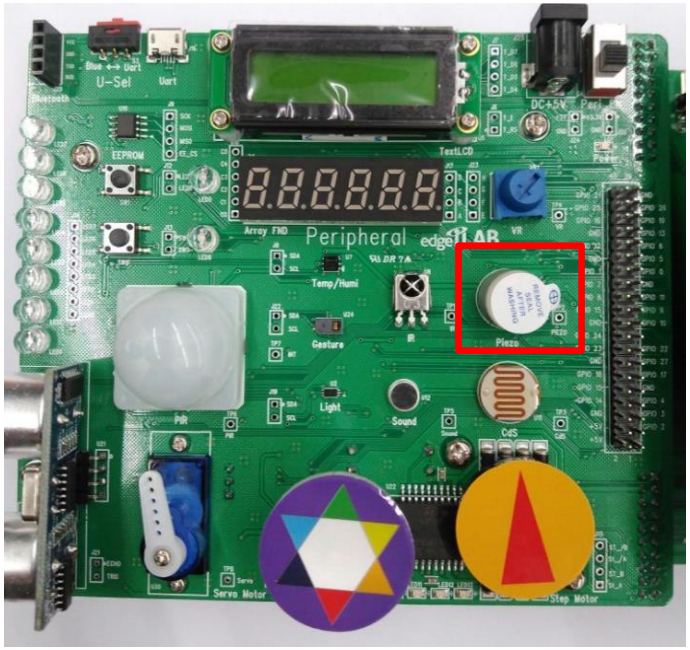
GPIO를 활용한 제어

- PIEZO 제어
  - Software Tone 라이브러리
    - 주파수를 GPIO에 간단한 Tone이나 구형파 펄스를 주기 위한 wiringPi지원 라이브러리

GPIO 설정 함수	
int softToneCreate(int pin); - 반환 값 : 성공시 0	해당 핀을 소프트웨어로 제어할 수 있는 Tone 핀으로 설정
GPIO 출력 함수	
void softToneWrite(int pin, int freq);	해당 핀에 주파수 값을 입력, 주파수가 0이 입력될 때까지 소리를 출력

GPIO를 활용한 제어

- PIEZO 제어
  - PIEZO는 핀 13번(BCM Mode)과 연결



Connect	BCM	Name	Physical		Name	BCM	Connect
		+3.3V	1	2	+5V		
I2C	GPIO 02	SDA1	3	4	+5V		
I2C	GPIO 03	SCL1	5	6	GND		
DC_M	GPIO 04	GPIO	7	8	TxD	GPIO 14	UART
		GND	9	10	RxD	GPIO 15	UART
SERVO	GPIO 17	GPIO	11	12	GPIO	GPIO 18	TLCD
TLCD	GPIO 27	GPIO	13	14	GND		
TLCD	GPIO 22	GPIO	15	16	GPIO	GPIO 23	TLCD
		+3.3V	17	18	GPIO	GPIO 24	PIR
SPI	GPIO 10	MOSI	19	20	GND		
SPI	GPIO 09	MISO	21	22	GPIO	GPIO 25	DC_M
SPI	GPIO 11	SCLK	23	24	CE0	GPIO 08	
		GND	25	26	CE1	GPIO 07	SPI
ULTRA	GPIO 00	SDA0	27	28	SCL0	GPIO 01	ULTRA
SWITCH	GPIO 05	GPIO	29	30	GND		
SWITCH	GPIO 06	GPIO	31	32	GPIO	GPIO 12	DC_M
PIEZO	GPIO 13	GPIO	33	34	GND		
IR	GPIO 19	GPIO	35	36	GPIO	GPIO 16	TLCD
TLCD	GPIO 26	GPIO	37	38	GPIO	GPIO 20	LED
		GND	39	40	GPIO	GPIO 21	LED

GPIO를 활용한 제어

- PIEZO 제어 예제(1)
  - 터미널 창에 "nano 06_PIEZO_01.c" 입력

```
pi@raspberryPi:~/Example $ nano 06_PIEZO_01.c
```

  - 3옥타브 '도'에서 4옥타브 '도'까지 softTone라이브러리를 이용해 순차적으로 소리를 내고 이를 반복하는 예제

```
1.  // File : 06_PIEZO_01.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>
4.  #include <softTone.h>                    // Software Tone library 참조

5.  const int pinPiezo = 13;                 // Piezo 핀 설정

6.  // 음계 초기화
7.  const int aMelody[8] = {131, 147, 165, 175, 196, 220, 247, 262};

8.  int main(void)
9.  {
10.     wiringPiSetupGpio();                 // 핀 번호를 BCM Mode로 설정

11.     softToneCreate(pinPiezo);            // 해당 핀을 Tone 핀으로 설정
```

GPIO를 활용한 제어

- PIEZO 제어 예제(1)

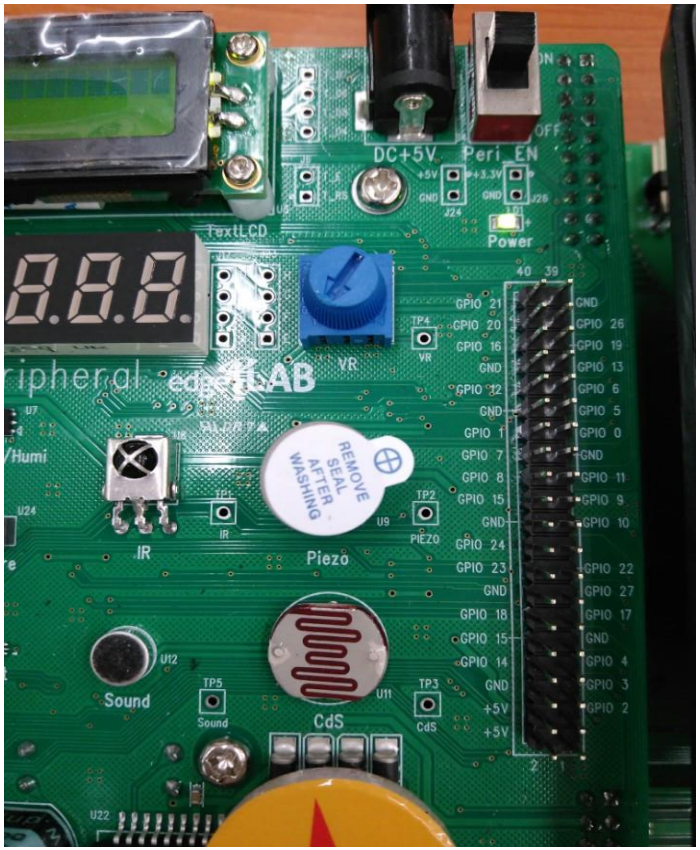
```
1.      while(1)
2.      {
3.          int i;
4.          for(i=0; i<8; i++)
5.          {
6.              softToneWrite(pinPiezo, aMelody[i]);        // 소리 출력
7.              delay(100);
8.          }
9.              softToneWrite(pinPiezo, 0);                 // 소리 출력 정지
10.             delay(1000);
11.     }
12.     return 0;
13. }
```

- 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
- GCC 컴파일러를 사용하여 빌드 및 생성된 "06_PIEZO_01" 파일 실행

```
pi@raspberryPi:~/Example $ gcc -o 06_PIEZO_01 06_PIEZO_01.c -lwiringPi
pi@raspberryPi:~/Example $ ./06_PIEZO_01
```

GPIO를 활용한 제어

- PIEZO 제어 예제(1)
  - 결과
    - PIEZO에서 0.1초마다 "도레미파솔라시도" 소리가 나고, 이를 1초마다 반복



GPIO를 활용한 제어

- PIEZO 제어 예제(2)
  - 터미널 창에 "nano 06_PIEZO_02.c" 입력

```
pi@raspberryPi:~/Example $ nano 06_PIEZO_02.c
```

  - PIR을 통해 동작 감지가 되면 PIEZO로 경고음을 내는 예제

```
1.  // File : 06_PIEZO_02.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>
4.  #include <softTone.h>                    // Software Tone library 참조

5.  const int pinPiezo = 13;                 // Piezo 핀 설정
6.  const int pinPir = 24;                   // Pir 핀 설정

7.  // 음계 초기화
8.  const int aMelody[8] = {131, 147, 165, 175, 196, 208, 220, 247, 262};

9.  int main(void)
10. {
11.     wiringPiSetupGpio();                // 핀 번호를 BCM Mode로 설정
12.     pinMode(pinPir, INPUT);             // Pir핀 입력모드로 설정
13.     softToneCreate(pinPiezo);           // 해당 핀을 Tone 핀으로 설정
```

GPIO를 활용한 제어

- PIEZO 제어 예제(2)

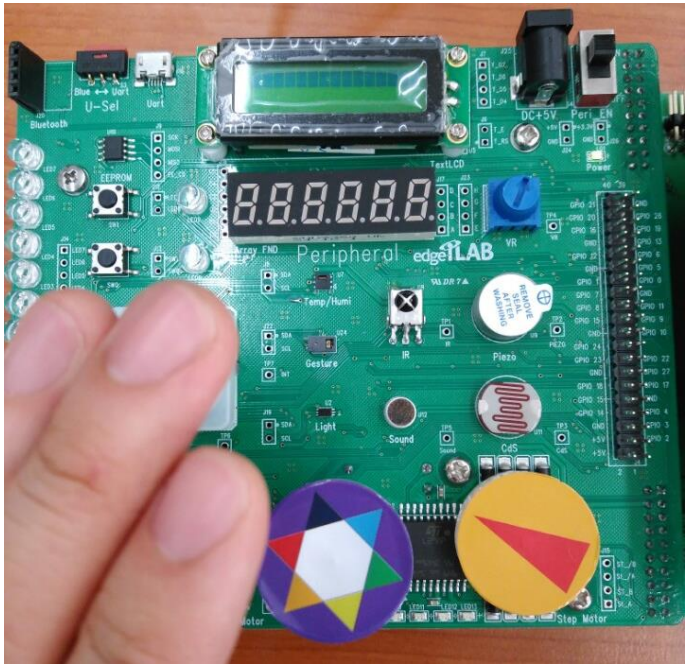
```
14.     while(1)
15.     {
16.         if(!digitalRead(pinPir))
17.         {
18.             softToneWrite(pinPiezo, aMelody[2]);
19.             delay(100);
20.             softToneWrite(pinPiezo, aMelody[0]);
21.             delay(100);

22.         }
23.         else
24.         {
25.             softToneWrite(pinPiezo, 0);
26.             delay(100);
27.         }
28.     }
29.     return 0;
30. }
```

GPIO를 활용한 제어

- PIEZO 제어 예제(2)
  - 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
  - GCC 컴파일러를 사용하여 빌드 및 생성된 "06_PIEZO_02" 파일 실행

```
pi@raspberryPi:~/Example $ gcc -o 06_PIEZO_02 06_PIEZO_02.c -lwiringPi
pi@raspberryPi:~/Example $ ./06_PIEZO_02
```

  - 결과
    - 동작 감지 시, 경고음 출력

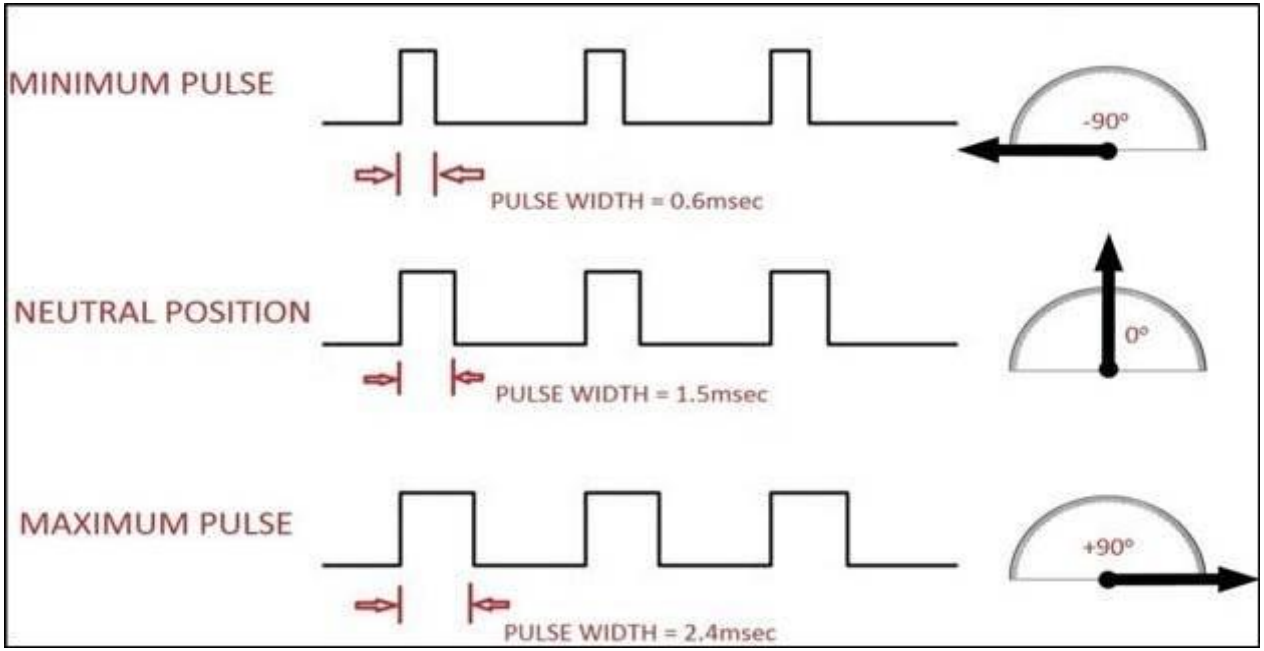


GPIO를 활용한 제어

- SERVO MOTOR 제어
  - 서보모터란 목표치에 대한 위치, 방위, 자세 등의 제어가 자동화 되어 있는 장치
  - 주로 모터와 기어박스, 제어회로로 구성
  - 특정 위치로 이동하거나, 특정 수치만큼 가동시킬 때, 모터로부터 피드백을 통해 정확하게 제어할 수 있는 구조
  - 자동화 생산 시스템 및 가전제품 등 광범위하게 사용

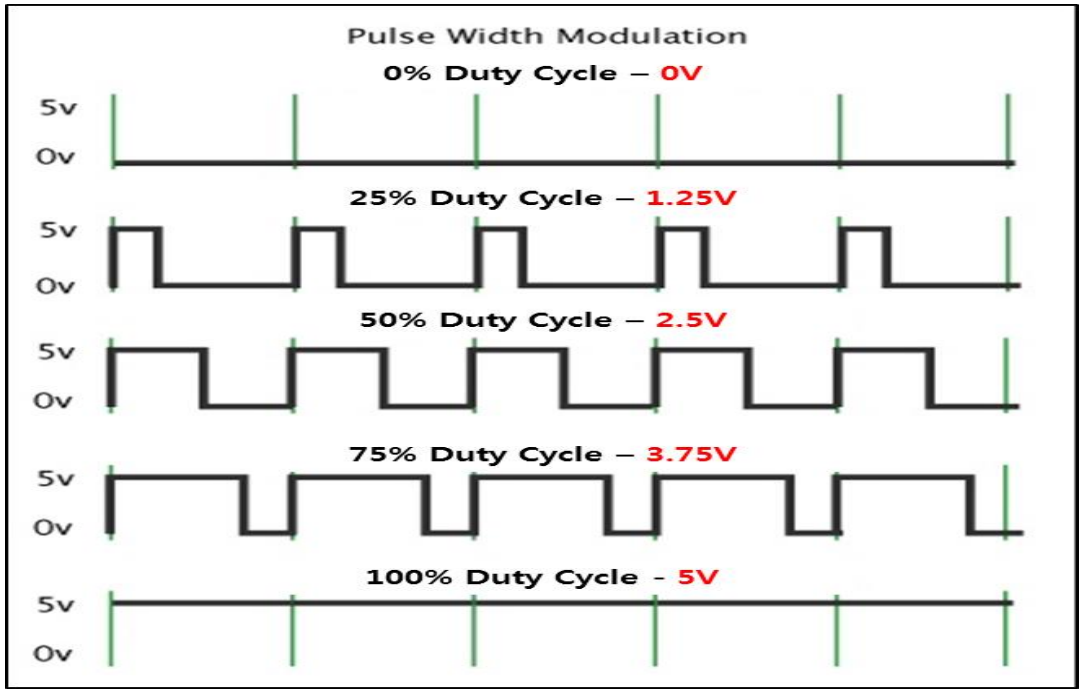
GPIO를 활용한 제어

- SERVO MOTOR 제어
  - 서보모터는 회전 반경이 정해져 있음(약 0 ~ 170 도)
  - 회전 각도는 펄스폭 변조(PWM, Pulse Width Modulation) 방식에 의해 제어



GPIO를 활용한 제어

- SERVO MOTOR 제어
  - PWM(Pulse Width Modulation)이란?
    - 펄스의 폭을 컨트롤하는 주기 제어 방법
    - 펄스의 HIGH구간과 LOW구간의 지연시간에 대한 변화는 주는 방식
    - HIGH와 LOW 구간의 상대적인 시간 비율을 달리하여 사각파 출력
    - LOW구간에 대한 HIGH 구간의 시간비율을 듀티(Duty)비율이라고 함
    - PWM을 사용하면 디지털 핀을 0~5V 사이의 아날로그처럼 출력 가능



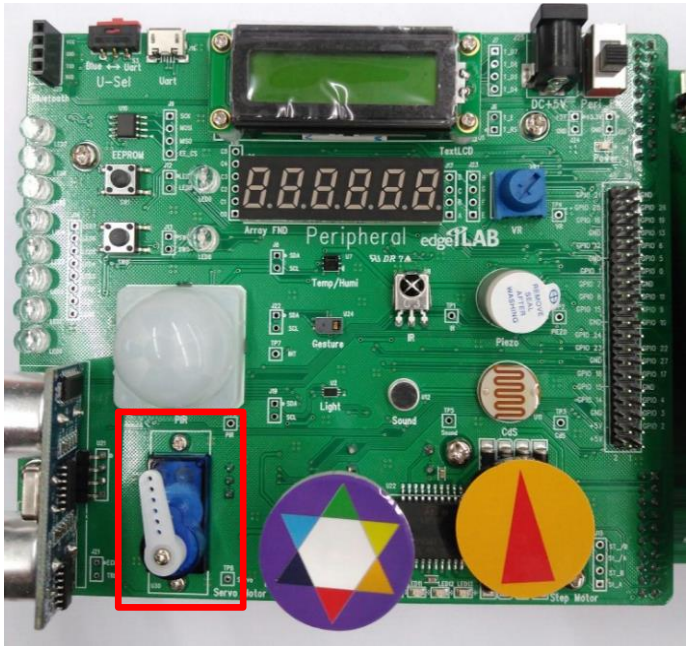
GPIO를 활용한 제어

- SERVO MOTOR 제어
  - Software PWM 라이브러리
    - 라즈베리파이의 GPIO핀에 PWM펄스를 출력할 수 있는 소프트웨어 기반의 구동 PWM 핸들러

GPIO 설정 함수	
int softPwmCreate(int pin, int init, int pwmRange); - 반환 값 : 성공시 0	소프트웨어 제어 PWM 핀으로 설정, pwmRange는 PWM의 범위를 설정
GPIO 출력 함수	
void softPwmWrite(int pin, int val);	해당 핀에 PWM(val) 값을 입력

GPIO를 활용한 제어

- SERVO MOTOR 제어
  - SERVO MOTOR는 핀 17번(BCM Mode)과 연결



Connect	BCM	Name	Physical		Name	BCM	Connect
		+3.3V	1	2	+5V		
I2C	GPIO 02	SDA1	3	4	+5V		
I2C	GPIO 03	SCL1	5	6	GND		
DC_M	GPIO 04	GPIO	7	8	TxD	GPIO 14	UART
		GND	9	10	RxD	GPIO 15	UART
SERVO	GPIO 17	GPIO	11	12	GPIO	GPIO 18	TLCD
TLCD	GPIO 27	GPIO	13	14	GND		
TLCD	GPIO 22	GPIO	15	16	GPIO	GPIO 23	TLCD
		+3.3V	17	18	GPIO	GPIO 24	PIR
SPI	GPIO 10	MOSI	19	20	GND		
SPI	GPIO 09	MISO	21	22	GPIO	GPIO 25	DC_M
SPI	GPIO 11	SCLK	23	24	CE0	GPIO 08	
		GND	25	26	CE1	GPIO 07	SPI
ULTRA	GPIO 00	SDA0	27	28	SCL0	GPIO 01	ULTRA
SWITCH	GPIO 05	GPIO	29	30	GND		
SWITCH	GPIO 06	GPIO	31	32	GPIO	GPIO 12	DC_M
PIEZO	GPIO 13	GPIO	33	34	GND		
IR	GPIO 19	GPIO	35	36	GPIO	GPIO 16	TLCD
TLCD	GPIO 26	GPIO	37	38	GPIO	GPIO 20	LED
		GND	39	40	GPIO	GPIO 21	LED

GPIO를 활용한 제어

- Software PWM 라이브러리 예제(1)
  - 터미널 창에 "nano 07_SERVO_01.c" 입력

```
pi@raspberryPi:~/Example $ nano 07_SERVO_01.c
```

  - SoftPwm 라이브러리를 이용하여 '0도' -> '+90도' -> '0도' -> '-90도' 로 1초마다 회전하는 예제

```
1.  // File : 07_SERVO_01.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>
4.  #include <softPwm.h>                    // Software PWM 라이브러리 참조

5.  const int pinServo = 17;                // Servo Motor 핀 설정

6.  int main(void)
7.  {
8.      wiringPiSetupGpio();                // 핀 번호를 BCM Mode로 설정

9.      softPwmCreate(pinServo, 0, 200);    // 해당 핀을 PWM 핀, 0~200까지 범위 설정
```

GPIO를 활용한 제어

- Software PWM 라이브러리 예제(1)

```
10.     while(1)
11.     {
12.         softPwmWrite(pinServo, 5);          // - 90도
13.         delay(1000);
14.         softPwmWrite(pinServo, 15);         // 0도
15.         delay(1000);
16.         softPwmWrite(pinServo, 25);        // +90도
17.         delay(1000);
18.         softPwmWrite(pinServo, 15);        // 0도
19.         delay(1000);
20.     }
21.     return 0;
22. }
```

- 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
- GCC 컴파일러를 사용하여 빌드 및 생성된 "07_SERVO_01" 파일 실행

```
pi@raspberryPi:~/Example $ gcc -o 07_SERVO_01 07_SERVO_01.c -
lwiringPi
pi@raspberryPi:~/Example $ ./07_SERVO_01
```

GPIO를 활용한 제어

- Software PWM 라이브러리 예제(1)
  - 결과
    - SERVO가 1초마다 정해진 각도로 회전

GPIO를 활용한 제어

- Software PWM 라이브러리 예제(2)
  - 터미널 창에 "nano 07_DCMOTOR_01.c" 입력

```
pi@raspberryPi:~/Example $ nano 07_DCMOTOR_01.c
```

  - SoftPwm 라이브러리를 이용하여 DC 모터의 회전속도를 조절하는 예제

```
1.  // File : 07_DCMOTOR_01.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>
4.  #include <softPwm.h>                  // Software PWM 라이브러리 참조

5.  const int pinEnable = 12;            // DC Motor Enable 핀 설정
6.  const int pinPositive = 4;           // DC Motor Positive 핀 설정
7.  const int pinNegative = 25;          // DC Motor Negative 핀 설정

8.  int main(void)
9.  {
10.     wiringPiSetupGpio();             // 핀 번호를 BCM Mode로 설정

11.     pinMode(pinPositive, OUTPUT);    // DC Motor의 Positive핀을 출력모드로 설정
12.     pinMode(pinNegative, OUTPUT);    // DC Motor의 Negative 핀을 출력모드로 설정
13.     softPwmCreate(pinEnable, 0, 200); // DC Motor의 Enable 핀을 PWM핀으로 설정
```

GPIO를 활용한 제어

- Software PWM 라이브러리 예제(2)

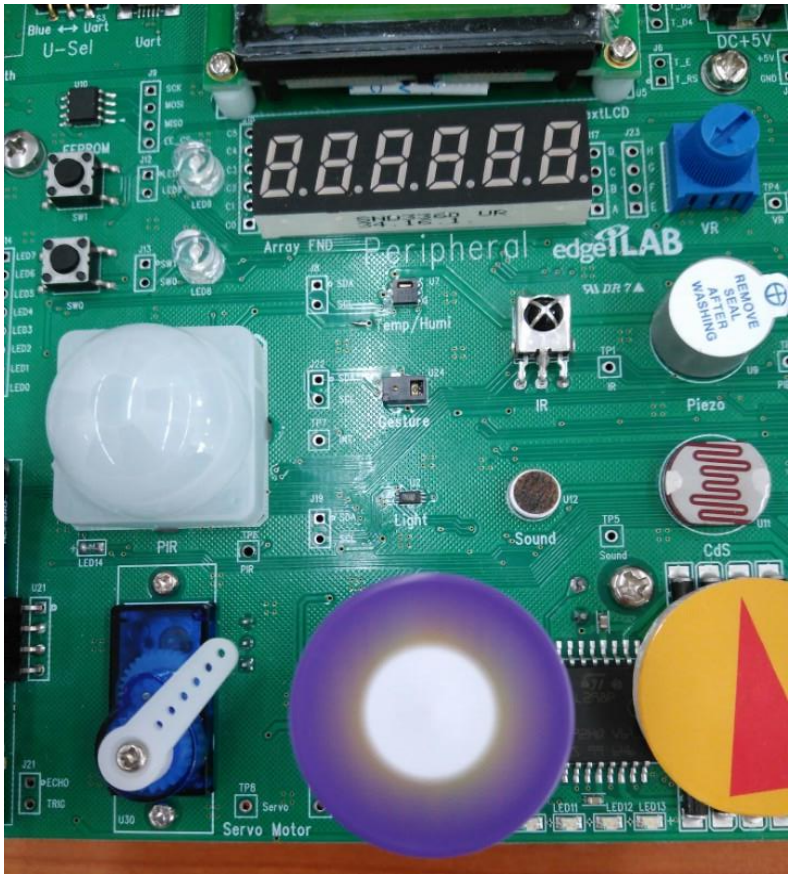
```
14.     digitalWrite(pinPositive, HIGH);     // 정방향 회전을 위해 Positive 핀 HIGH로 설정
15.     digitalWrite(pinNegative, LOW);

16.     while(1)
17.     {
18.         softPwmWrite(pinEnable, 30);
19.         delay(2000);
20.         softPwmWrite(pinEnable, 0);
21.         delay(1000);
22.         softPwmWrite(pinEnable, 80);
23.         delay(2000);
24.         softPwmWrite(pinEnable, 0);
25.         delay(1000);
26.     }
27.     return 0;
28. }
```

  - 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
  - GCC 컴파일러를 사용하여 빌드 및 생성된 "07_SERVO_01" 파일 실행

```
pi@raspberryPi:~/Example $ gcc -o 07_DCMOTOR_01 07_DCMOTOR_01.c
 -lwiringPi
pi@raspberryPi:~/Example $ ./07_DCMOTOR_01
```

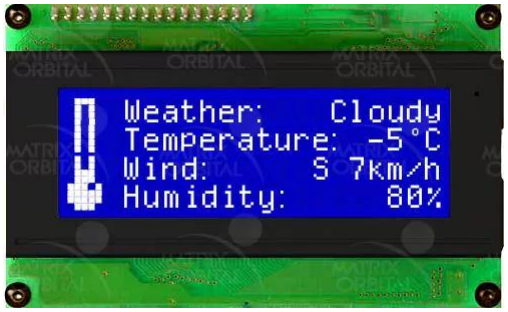
GPIO를 활용한 제어

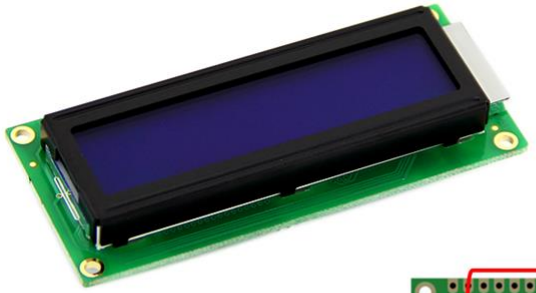
- Software PWM 라이브러리 예제(2)
  - 결과
    - DC Motor가 속도를 다르게 하여 2초마다 회전 후, 정지

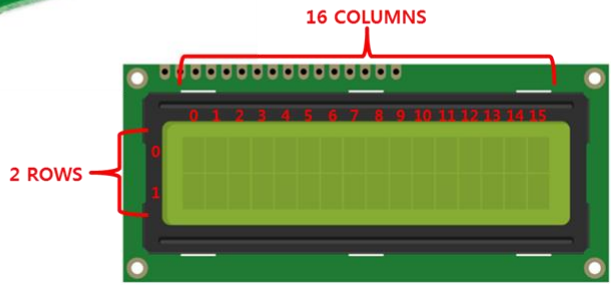


GPIO를 활용한 제어

- Text LCD 제어
  - LCD(Liquid Crystal Display, 액정 디스플레이)는 액체와 고체의 중간 상태인 액정의 전기 성질을 응용하여 시각적인 효과를 주는 제품
  - 저전압, 저전력으로 작동
  - ASCII코드를 입력받아 영문자를 출력
    - 임베디드 시스템에 주로 사용
  - Text LCD는 칸과 줄 수에 따라 종류가 다양

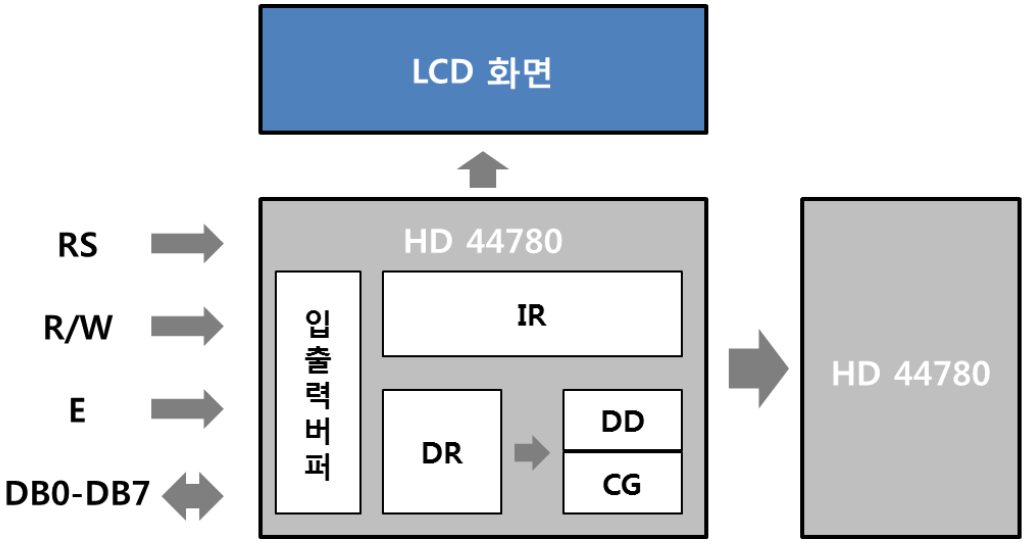






GPIO를 활용한 제어

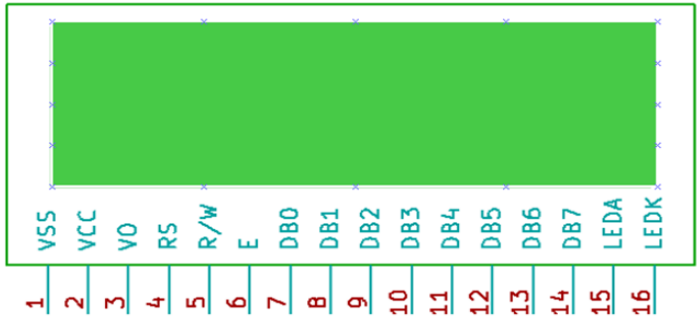
- Text LCD 제어



- LCD는 Panel과 제어기(Controller)가 하나의 모듈로 구성
- 데이터 버스를 통해 제어기에 명령과 데이터를 전송
- 제어기는 명령에 따라 문자 코드에 해당하는 글씨를 LCD Panel에 출력
- LCD Panel이 출력상태를 유지하도록 제어

GPIO를 활용한 제어

- Text LCD 제어
  - 외부 인터페이스는 총 16개로 구성
    - 이중 8개의 데이터라인과 3개의 컨트롤 라인을 주로 사용



Pin Number	Symbol	Pin Function
1	VSS	Ground
2	VCC	+5V
3	VEE	Contrast adjustment(VO)
4	RS	Resister Select
5	R/W	Read/Write
6	En	Enable
7	D0	Data Bit 0
8	D1	Data Bit 1
9	D2	Data Bit 2
10	D3	Data Bit 3
11	D4	Data Bit 4
12	D5	Data Bit 5
13	D6	Data Bit 6
14	D7	Data Bit 7
15	A/LED+	Back-Light Anode(+)
16	K/LED-	Back-Light Cathode(-)

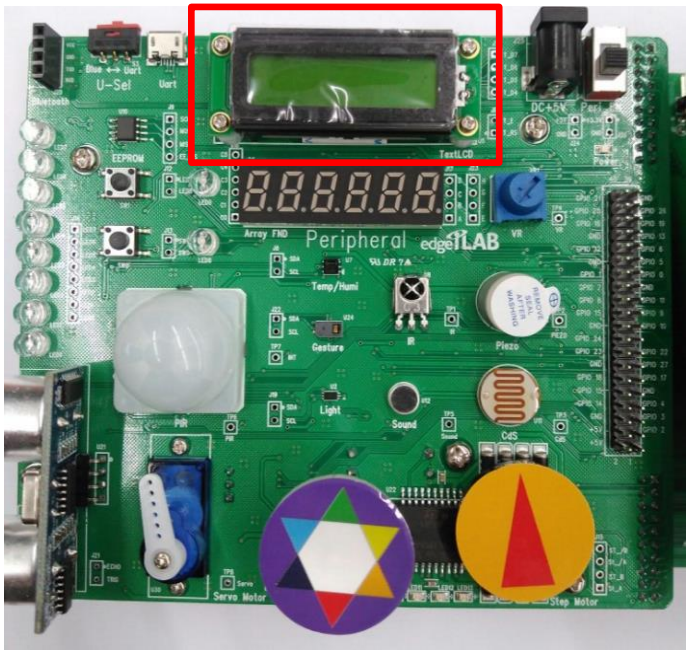
GPIO를 활용한 제어

- Text LCD 제어
  - LCD 라이브러리

초기화 함수	
int lcdInit(int rows, int cols, int bit, int rs, int strb, int d0, int d1, int d2, int d3, int d4, int d5, int d6, int d7); - 반환 값 : 성공시 'handle'을 리턴, 이를 통해 제어 함수에 활용, 실패시 -1	Lcd 초기화 함수, rows과 cols은 디스플레이의 행과 열을 의미, bits는 비트 폭, RS 및 STRB는 디스플레이의 RS핀과 Enable 핀 번호, d0~d7은 디스플레이로 연결된 8개의 데이터 핀 번호 4비트 모드일 경우 d0-d3만 사용
여러 기능의 제어 함수	
void lcdPostion(int handle, int x, int y);	커서의 위치를 설정 x는 col이고, 0은 가장 왼쪽 자리, y는 row이고, 0은 가장 위의 라인을 뜻함
void lcdPuts(int handle, char *string);	LCD에 문자(열)을 출력
void lcdPutchar(int handle, uint_t data);	
void lcdPrintf(int handle, char *string);	
void lcdClear(int handle);	LCD 전체 화면을 클리어

GPIO를 활용한 제어

- Text LCD 제어
  - Text LCD는 핀 12, 16, 22, 26, 27, 36번(BCM Mode)과 연결
  - GPIO 12, 27, 22, 23 : D0~D3
  - GPIO 26 : STRB
  - GPIO 36 : RS



Connect	BCM	Name	Physical		Name	BCM	Connect	
		+3.3V	1	2	+5V			
I2C	GPIO 02	SDA1	3	4	+5V			
I2C	GPIO 03	SCL1	5	6	GND			
DC_M	GPIO 04	GPIO	7	8	TxD	GPIO 14	UART	
		GND	9	10	RxD	GPIO 15	UART	
SERVO	GPIO 17	GPIO	11	12	GPIO	GPIO 18	TLCD	D0 핀
TLCD	GPIO 27	GPIO	13	14	GND			
TLCD	GPIO 22	GPIO	15	16	GPIO	GPIO 23	TLCD	D3 핀
		+3.3V	17	18	GPIO	GPIO 24	PIR	
SPI	GPIO 10	MOSI	19	20	GND			
SPI	GPIO 09	MISO	21	22	GPIO	GPIO 25	DC_M	
SPI	GPIO 11	SCLK	23	24	CE0	GPIO 08		
		GND	25	26	CE1	GPIO 07	SPI	
ULTRA	GPIO 00	SDA0	27	28	SCL0	GPIO 01	ULTRA	
SWITCH	GPIO 05	GPIO	29	30	GND			
SWITCH	GPIO 06	GPIO	31	32	GPIO	GPIO 12	DC_M	
PIEZO	GPIO 13	GPIO	33	34	GND			
IR	GPIO 19	GPIO	35	36	GPIO	GPIO 16	TLCD	RS 핀
TLCD	GPIO 26	GPIO	37	38	GPIO	GPIO 20	LED	
		GND	39	40	GPIO	GPIO 21	LED	

D1 핀
D2 핀
STRB 핀

GPIO를 활용한 제어

- Text LCD 제어 예제(1)
  - 터미널 창에 "nano 08_TEXTLCD_01.c" 입력

```
pi@raspberryPi:~/Example $ nano 08_TEXTLCD_01.c
```

  - LCD 라이브러리를 이용하여 LCD에 해당 문자열을 출력하는 예제

```
1.  // File : 08_TEXTLCD_01.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>
4.  #include <lcd.h>                          // lcd 라이브러리 참조

5.  int main(void)
6.  {
7.      wiringPiSetupGpio();                 // 핀 번호를 BCM Mode로 설정

8.      // 2*16 Text LCD, 4bit 폭, rs, strb, d0, d1, d2, d3 핀 설정
9.      int lcd = lcdInit(2, 16, 4, 16, 26, 18, 27, 22, 23, 0, 0, 0, 0);

10.     lcdClear(lcd);                       // lcd 화면 클리어

11.     lcdPosition(lcd, 0, 0);              // Cursor를 0, 0으로 이동
12.     lcdPuts(lcd, "edgeiLAB");            // lcd화면에 "edgeiLAB" 출력
```

GPIO를 활용한 제어

- Text LCD 제어 예제(1)

```
13.     lcdPosition(lcd, 0, 1);                 // Cursor를 0, 1으로 이동
14.     lcdPuts(lcd, "Hello World");            // lcd 화면에 "Hello World" 출력

15.     delay(1000);

16.     while(1)
17.     {
18.     }
19.     return 0;
20. }
```

  - 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
  - GCC 컴파일러를 사용하여 빌드 및 생성된 "08_TEXTLCD_01" 파일 실행
    - 컴파일 시, wiringPiDev 라이브러리를 추가하여 컴파일

```
pi@raspberryPi:~/Example $ gcc -o 07_TEXTLCD_01 07_TEXTLCD_01.c
-lwiringPi -lwiringPiDev
pi@raspberryPi:~/Example $ ./07_TEXTLCD_01
```

GPIO를 활용한 제어

- Text LCD 제어 예제(1)
  - 결과
    - Text LCD의 첫째 줄에는 "EdgeiLAB"
    - 둘째 줄에는 "Hello World"가 출력

GPIO를 활용한 제어

- Text LCD 제어 예제(2)
  - 터미널 창에 "nano 08_TEXTLCD_02.c" 입력

```
pi@raspberryPi:~/Example $ nano 08_TEXTLCD_02.c
```

  - Ultrasonic으로 측정한 거리 값을 Text LCD 에 출력하는 예제

```
1.  // File : 08_TEXTLCD_02.c

2.  #include <stdio.h>
3.  #include <wiringPi.h>
4.  #include <lcd.h>                          // lcd 라이브러리 참조

5.  const int pinEcho = 1;
6.  const int pinTrigger = 0;

7.  int main(void)
8.  {
9.      wiringPiSetupGpio();                 // 핀 번호를 BCM Mode로 설정

10.     pinMode(pinEcho, INPUT);             // Echo핀을 입력으로 설정
11.     pinMode(pinTrigger, OUTPUT);         // Trigger 핀을 출력으로 설정
```

GPIO를 활용한 제어

- Text LCD 제어 예제(2)

```
1.      // 2*16 Text LCD, 4bit 폭, rs, strb, d0, d1, d2, d3 핀 설정
2.      int lcd = lcdInit(2, 16, 4, 16, 26, 18, 27, 22, 23, 0, 0, 0, 0);

3.      digitalWrite(pinTrigger, LOW);
4.      delay(30);

5.      while(1)
6.      {
7.          digitalWrite(pinTrigger, HIGH);          // Trigger 핀을 20us 동안 HIGH로 출력
8.          delayMircoseconds(20);
9.          digitalWrite(pinTrigger, LOW);

10.         while(digitalRead(pinEcho) == LOW);     // 반사된 신호가 들어올 때까지 대기
11.         long startTime = micros();              // 반사된 신호가 수신될 때 시간 초기화

12.         while(digitalRead(pinEcho) == HIGH);    // 반사된 신호가 없을 때까지 대기
13.         long endTime = micros() – startTime;    // 반사된 신호가 없어진 직후 시간차 계산

14.         int distance =  endTime / 58;           // 시간을 이용해 거리 계산

15.         lcdClear(lcd);                          // lcd 화면 클리어
```

GPIO를 활용한 제어

- Text LCD 제어 예제(2)

```
1.          lcdPosition(lcd, 0, 0);                    // Cursor를 0, 0으로 이동
2.          lcdPuts(lcd, "Distance");                  // lcd화면에 "Distance" 출력

3.          lcdPosition(lcd, 0, 1);
4.          lcdPrintf(lcd, "%d cm ", distance);        // lcd 화면에 거리 출력

5.          delay(500);
6.      }
7.      return 0;
8.  }
```

  - 작성 후 "crtl + o" 를 눌러 저장 및 "ctrl + X"를 눌러 종료
  - GCC 컴파일러를 사용하여 빌드 및 생성된 "08_TEXTLCD_02" 파일 실행
    - 컴파일 시, wiringPiDev 라이브러리를 추가하여 컴파일

```
pi@raspberryPi:~/Example $ gcc -o 08_TEXTLCD_02 08_TEXTLCD_02.c
-lwiringPi -lwiringPiDev

pi@raspberryPi:~/Example $ ./08_TEXTLCD_02
```

GPIO를 활용한 제어

- Text LCD 제어 예제(2)
  - 결과
    - Text LCD에 Ultrasonic으로부터 측정한 거리가 출력