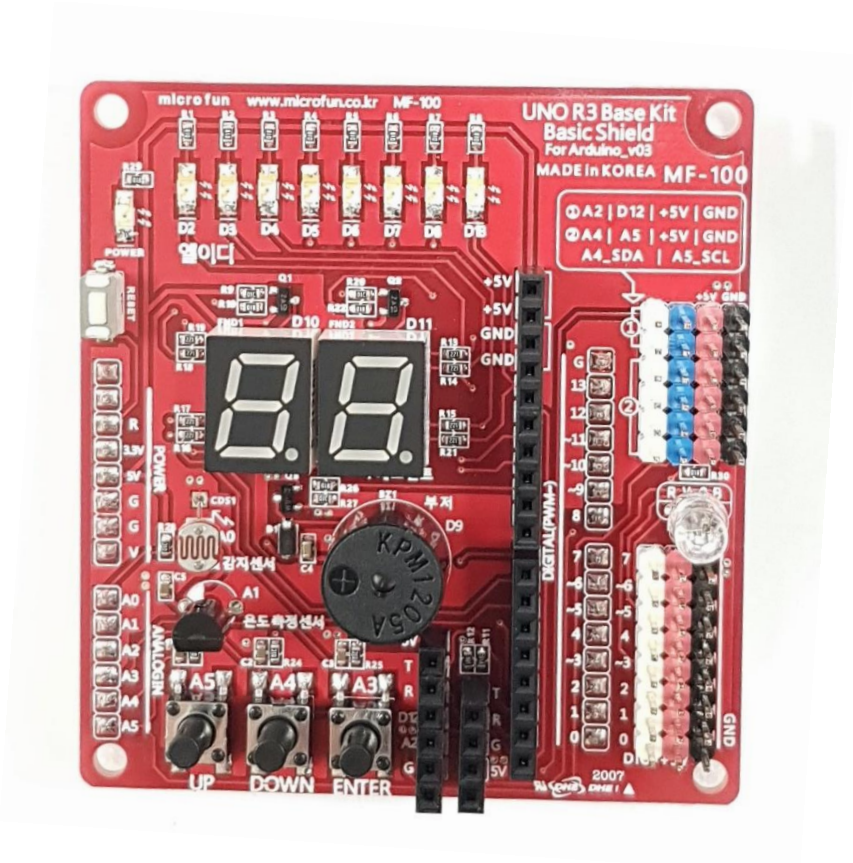
MF_100_UNO_BASE_KIT

제품설명서

Manual Ver 1.0



www.bitbus.co.kr

㈜ 비트버스

bitbus

1. 제품 모델명

MF_100_UNO_BASE_KIT (아두이노 우노 R3 쉴드)

2. 제품 설명

이 제품은 여러가지의 입출력 부품들을 모아 하나의 아두이노 쉴드로 구성한 제품입니다.

초보자도 쉽게 사용할 수 있도록 구성된 교육용 제품입니다.

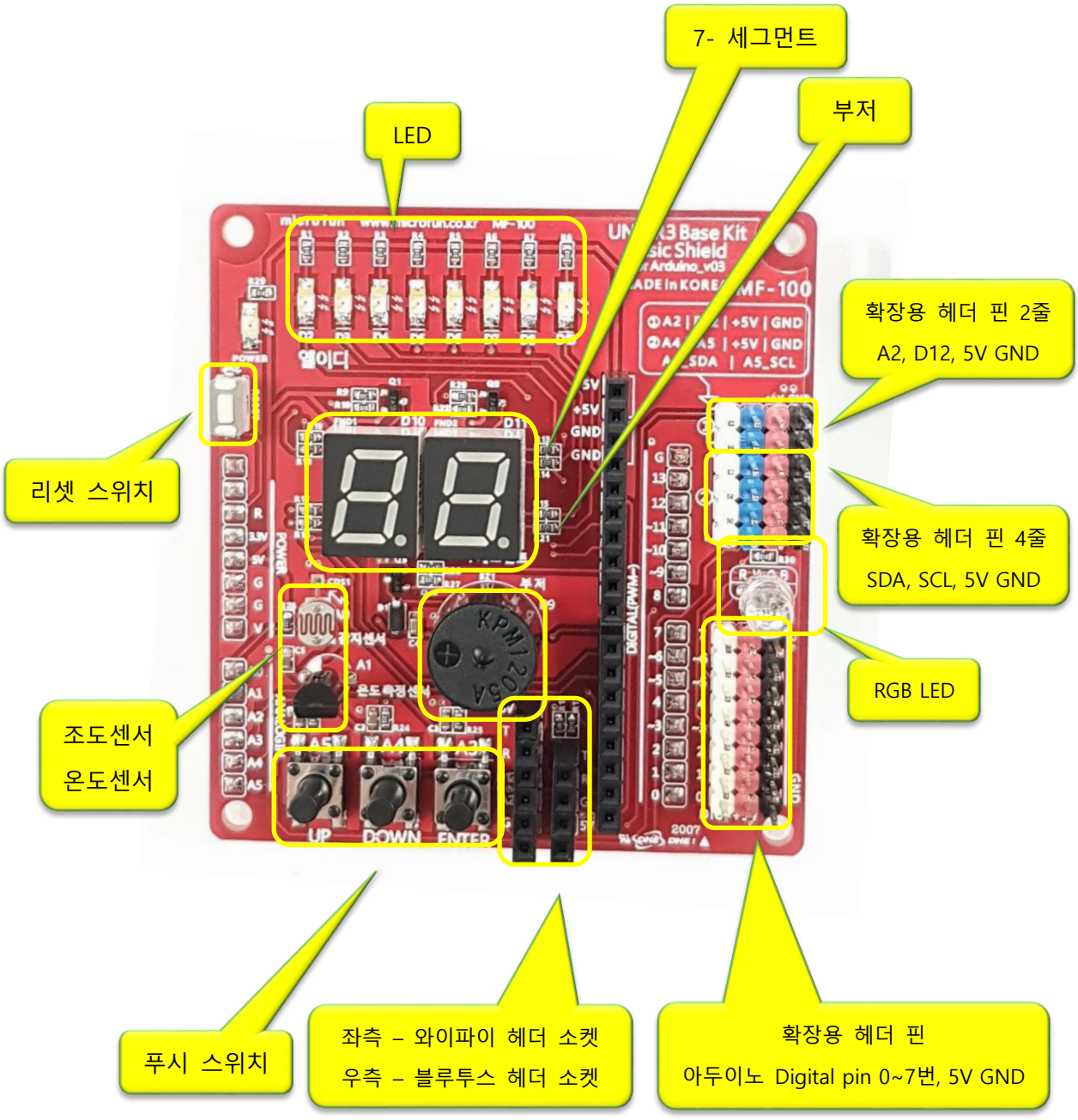
제품에는 총 6개의 (LED, 7-세그먼트, 푸시 스위치, 온도센서, 조도센서, 부저) 센서 및 입출력 부품들이 들어가 있으며 여분의 헤더 핀을 이용해서 추가로 센서 등의 여러가지 기기들을 제어하는 것이 가능합니다.

확장성을 위해 여분의 핀을 사용할 수 있도록 구성했으며 사용이 편리하도록 전원도 같이 구성 되어있습니다.

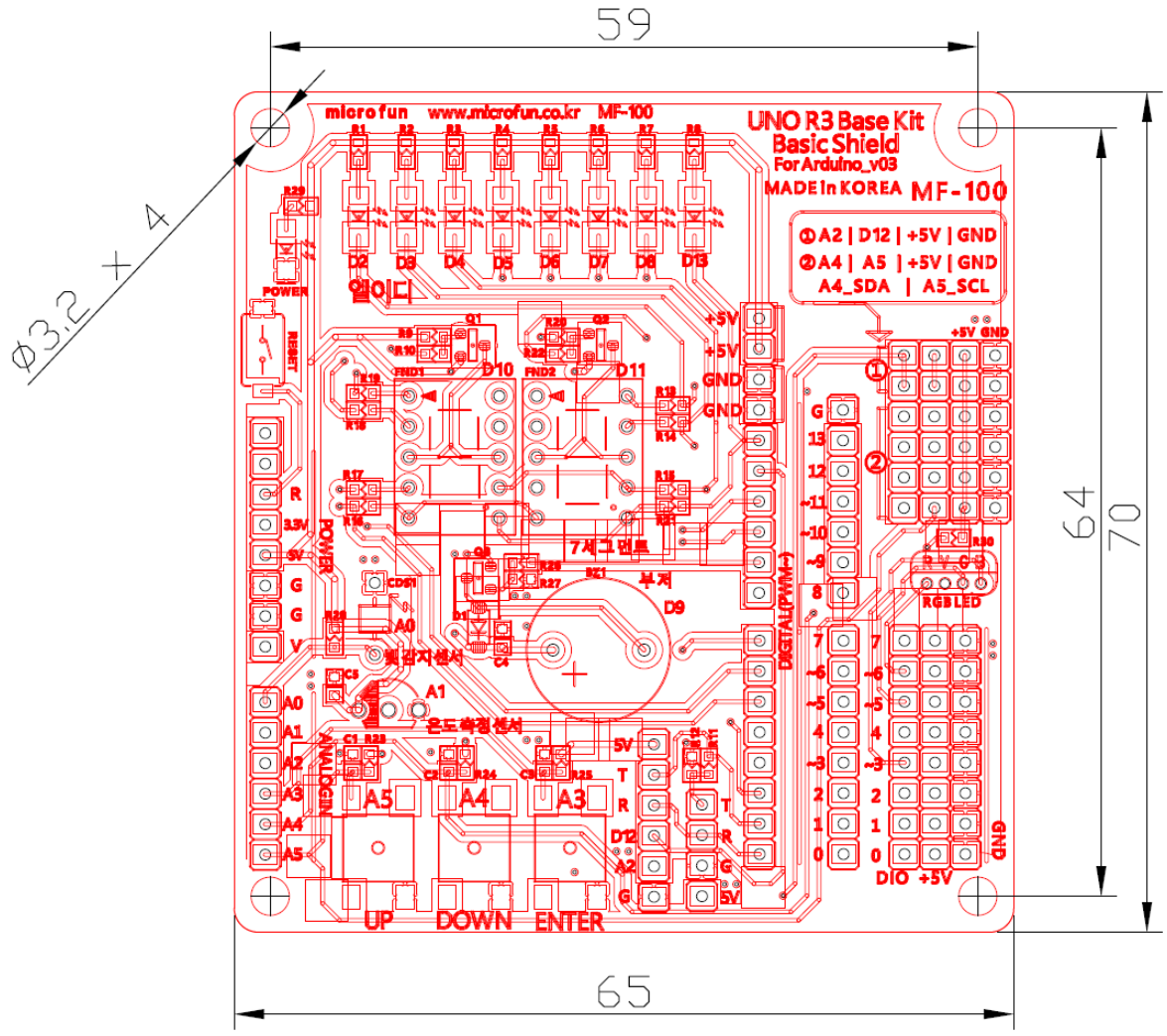
전원 헤더 핀을 구분이 용이 하도록 색상을 구분해서 제작되었습니다.
(5V – 빨강, GND - 검정)

- 아두이노 우노 R3의 쉴드 타입
- 여러 센서 및 입출력 장치 구성
- 여분의 헤더 핀으로 다른 센서 및 입출력 장치 제어 가능
- 아두이노 스케치 프로그램으로 프로그래밍 가능
- 블루투스용 헤더 소켓 구성 (Serial핀 사용 D0, D1)
- 애노드 타입 RGB LED 구성
- 확장용 헤더 핀 구성 (2개)
- I2C용 헤더 핀 구성 (4개)

아래 사진은 제품의 각 부품 별 안내 사항입니다.

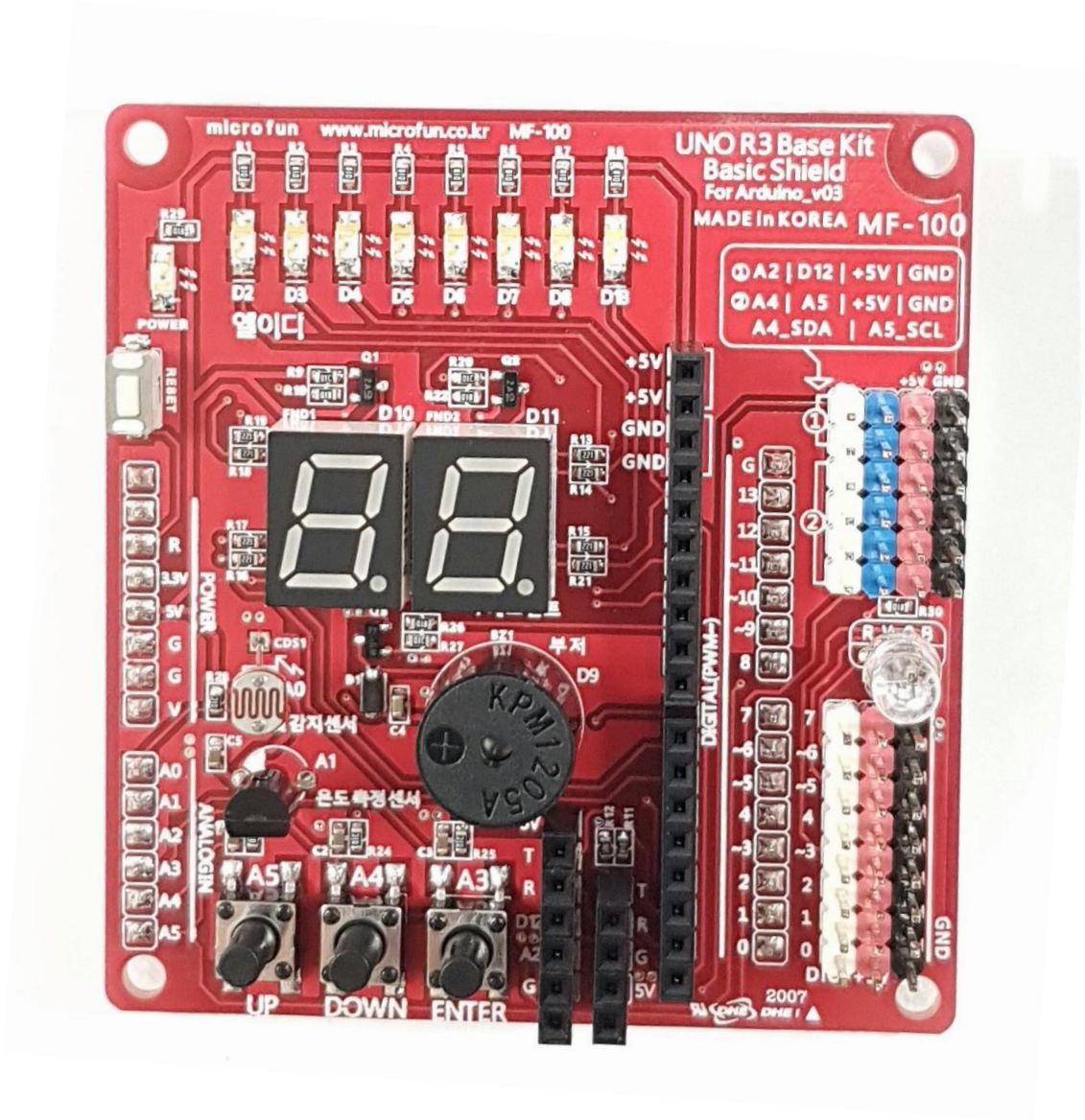


아래 사진은 제품의 도면 및 치수에 대한 안내 사항입니다.



3. 제품 구성

- MF_100_UNO_BASE_KIT (아두이노 우노 R3 쉴드) 1EA



+

4. 예제 따라하기

- **사용 예시 1**

<LED>

LED 는 각각 핀들이 한쪽에는 5V 가 연결되어 있기 때문에 제어는 LOW 로 프로그래밍 해야 ON / OFF 제어가 됩니다.

LED 제어를 시작 하겠습니다.

<LED 전체 ON/OFF 제어 프로그램>

```
void setup() {
  // put your setup code here, to run once:

  // LED 및 FND 사용위한 출력 핀 설정
  pinMode(2, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(3, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(4, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(5, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(6, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(7, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(8, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(13, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)

  //LED OFF
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(13, HIGH);
}

void loop() {
```

```
  // 전체 LED OFF
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(13, HIGH);
  delay(1000); // 1 초 지연
  // 전체 LED ON
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
  digitalWrite(13, LOW);
  delay(1000); // 1 초 지연
}
```

pinMode(핀 번호, 값);

핀 번호는 현재 LED 의 -극과 연결이 되어있는 아두이노의 핀 번호를 넣어주면 됩니다.

값은 해당 핀번호가 어떤 역할을 할지 정해주는 것입니다.

(INPUT – 입력 OUTPUT – 출력)

digitalWrite(핀 번호, 값);

핀번호에 HIGH 또는 LOW 값을 출력합니다.

위 프로그램을 업로드 하면 전체 LED 가 1 초동안 ON, 다시 1 초동안 OFF 되는 모습을 보실 수 있습니다.

<LED 전체 ON/OFF 제어 프로그램>

```
void setup() {
  // put your setup code here, to run once:

  // LED 및 FND 사용위한 출력 핀 설정
  pinMode(2, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(3, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(4, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(5, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(6, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(7, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(8, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(13, OUTPUT);   // LED OUTPUT(LED ==> HIGH : OFF)

  //LED OFF
```

```
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(13, HIGH);
}

void loop() {

  for (int a = 2; a < 10; a++)
  {
    if (a >= 9)
    {
      digitalWrite(13, LOW);
    }
    else
    {
      digitalWrite(a, LOW);
    }
  }
  delay(1000); // 1 초 지연
  for (int a = 2; a < 10; a++)
  {
    if (a >= 9)
    {
      digitalWrite(13, HIGH);
    }
    else
    {
      digitalWrite(a, HIGH);
    }
  }
  delay(1000); // 1 초 지연
}
```

이처럼 for 문을 이용하여 프로그램의 길이를 줄이는 것도 가능합니다.

이는 좌측 LED 부터 순서대로 켜지며 전체 다 켜지면 이어서 순서대로 꺼지는 동작을 합니다.

사용 예시 2

다음은 부저를 이용하는 방법입니다.
부저는 9 번 핀에 연결되어 있으며 이 부저는 다음과 같이 사용합니다.

```
void setup() {
  // put your setup code here, to run once:

  pinMode(9, OUTPUT);   // Buzzer OUTPUT

  //수동부저 사용
  tone(9, 523);  // 도
  delay(500);
  tone(9, 587);  // 레
  delay(500);
  tone(9, 659);  // 미
  delay(500);
  noTone(9);
}

void loop() {

}
```

먼저 pinMode를 이용하여 9번 핀을 출력으로 설정합니다.
이후 부저를 사용하기 위해 tone 이라는 함수와 noTone 이라는 함수를 이용하고 있습니다.

Tone (핀번호, 주파수값);
해당하는 핀호가 pwm 출력이 가능한 핀에서만 사용이 가능합니다.
해당 핀번호에 주파수 값을 출력합니다.

noTone(핀번호); 해당핀 번호의 주파수 출력을 멈춥니다.

사용 예시 3

다음은 세그먼트를 사용하는 예제입니다.
세그먼트의 핀의 일부가 LED 와 같이 연결되어 있기 때문에 LED 가 같이 켜집니다.
고장이 아닙니다. 다음 예제를 봐주시기 바랍니다.

```
int segment_pin[] = {2, 3, 4, 5, 6, 7, 8, 13};  //A~G, Dot

//0~9 까지의 숫자 표시를 위한 세그먼트 a~g 의 점멸 패턴
byte digits_data[10] = {0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE4, 0xFE,
0xE6};
int count1 = 9;
int count10 = 9;
int delay_cnt = 0;
void setup() {
  Serial.begin(9600);
  // put your setup code here, to run once:
  pinMode(2, OUTPUT);   // FND_A or LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(3, OUTPUT);   // FND_B or LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(4, OUTPUT);   // FND_C or LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(5, OUTPUT);   // FND_D or LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(6, OUTPUT);   // FND_E or LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(7, OUTPUT);   // FND_F or LED OUTPUT(LED ==> HIGH : OFF)
  pinMode(8, OUTPUT);   // FND_G or LED OUTPUT(LED ==> HIGH : OFF)

  pinMode(10, OUTPUT);   // D10_FND_COM1
  pinMode(11, OUTPUT);   // D11_FND_COM2
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  COUNT1();
  delay(10);
  digitalWrite(11, HIGH);
  digitalWrite(10, HIGH);
  COUNT10();
  delay(10);
```

```
  delay_cnt++;
  if (delay_cnt >= 50)
  {
    Serial.print("COUNT10 || COUNT1: ");
    Serial.print(count10);
    Serial.print(" || ");
    Serial.println(count1);

    delay_cnt = 0;
    count1--;
    if (count1 < 0)
    {
      count1 = 9;
      count10--;
      if (count10 < 0)
      {
        count10 = 9;
      }
    }
  }
}
void COUNT1()
{
  digitalWrite(11, LOW);
  for (int i = 0; i < 8; i++) //a~g 까지의 세그먼트 핀을 선택하여 사용하기위한
for 문
  {
    byte segment_data = (digits_data[count1] & (0x01 << i)) >> i;
    if (segment_data == 1)
      digitalWrite(segment_pin[7 - i], LOW);
    else
      digitalWrite(segment_pin[7 - i], HIGH);
  }
}
void COUNT10()
{
  digitalWrite(10, LOW);
  for (int i = 0; i < 8; i++) //a~g 까지의 세그먼트 핀을 선택하여 사용하기위한
for 문
  {
    byte segment_data = (digits_data[count10] & (0x01 << i)) >> i;
    if (segment_data == 1)
      digitalWrite(segment_pin[7 - i], LOW);
    else
      digitalWrite(segment_pin[7 - i], HIGH);
  }
}
```

위 예제는 세그먼트의 A~G 까지에 연결된 2 번부터 8 번까지의 핀을 모두 출력으로 사용하고 두개의 세그먼트를 선택할 10 번과 11 번 핀을 각각 출력으로 설정합니다.

이후 byte digits_data[10] = {0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE4, 0xFE, 0xE6}; 를 통해 0~9 까지의 숫자를 표현하고

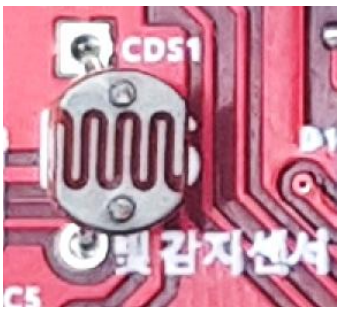
```
for (int i = 0; i < 8; i++) //a~g 까지의 세그먼트 핀을 선택하여 사용하기위한 for 문
  {
    byte segment_data = (digits_data[count] & (0x01 << i)) >> i;
    if (segment_data == 1)
      digitalWrite(segment_pin[7 - i], LOW);
    else
      digitalWrite(segment_pin[7 - i], HIGH);
  }
```

위 for 문을 통하여 FND 의 a~g 까지의 핀을 빠르게 선택하여 출력합니다.

동작은 99~00 까지의 값으로 점차 줄이도록 구현되어 있습니다.

10 번 핀과 11 은 통해 왼쪽 FND 와 오른쪽 FND 를 선택하여 제어가 가능합니다.

사용 예시 4



다음은 CDS 센서입니다.
CDS 센서는 조도 센서로 빛을 감지하는 센서입니다.
아래 예제를 통해 사용해봅니다.

```
int sensorValue_cds = 0;  // variable to store the value coming from the sensor
void setup() {
  // put your setup code here, to run once:
  // 시리얼 설정
  Serial.begin(9600);
  // 핀 설정
  pinMode(A0, INPUT);   // CDS 빛 감지 센서
}

void loop() {
```

```
  // put your main code here, to run repeatedly:
  sensorValue_cds = analogRead(A0);
  Serial.print("ADC0(SDC) : " );
  Serial.println(sensorValue_cds);
  delay(1000); // 1 초 지연

}
```

해당 예제는 아날로그 핀을 이용합니다.
A0 핀을 사용하며 pinMode(A0, INPUT); 을 통해 해당 핀을 입력으로 사용합니다.
analogRead(핀 번호); 핀번호에 입력되는 신호 값을 0~1023 의 ADC 값으로 출력합니다.
위 함수를 통해 CDS 에 연결된 A0 핀에 들어오는 신호 값을 읽어서 변수에 저장하고 해당 변수를 시리얼 모니터 창에 출력하는 예제입니다.

시리얼 모니터의 설정으로는
Serial.begin(통신속도); 해당 통신속도로 시리얼 통신을 시작합니다.

Serial.print(내용); 내용에는 원하는 문자나 변수를 넣게 되면 시리얼 모니터 창으로 출력되며 줄바꿈이 적용되지 않습니다.

Serial.print(내용); 내용에는 원하는 문자나 변수를 넣게 되면 시리얼 모니터 창으로 출력되며 줄바꿈이 적용됩니다.

사용 예시 5

해당 센서는 LM35DZ 온도측정 센서입니다.
센서의 사용법에 대해서는 아래의 예제를 통해 확인하겠습니다.

```ino
int sensorValue_lm35 = 0;  // variable to store the value coming from the
sensor
void setup() {
  // put your setup code here, to run once:
  // 시리얼 설정
  Serial.begin(9600);
  // 핀 설정
  pinMode(A1, INPUT);   // LM35DZ 온도센서
}

void loop() {
  // put your main code here, to run repeatedly:
  sensorValue_lm35 = analogRead(A1);
  Serial.print("ADC1(lm35) : " );
  Serial.print((sensorValue_lm35 * 5.0 * 100.0) / 1024.0);
  Serial.println("도");
  delay(1000); // 1 초 지연
}
```

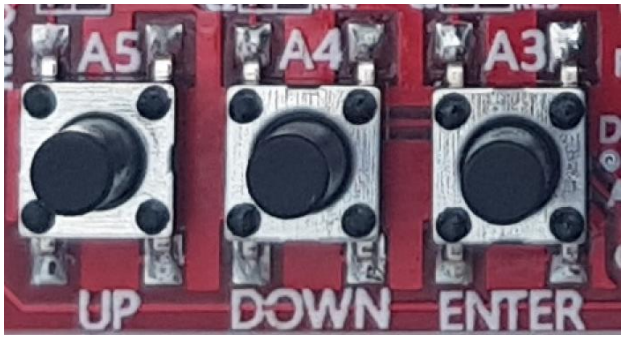
해당 예제는 아날로그 핀을 이용합니다.
A1 핀을 사용하며 pinMode(A1, INPUT); 을 통해 해당 핀을 입력으로 사용합니다.
analogRead(핀 번호); 핀 번호에 입력되는 신호 값을 0~1023 의 ADC 값으로 출력합니다.
위 함수를 통해 LM35Z 에 연결된 A1 핀에 들어오는 신호 값을 읽고 저장된 변수를 이용해서 수식을 통해 온도 값을 계산해서 시리얼 모니터에 출력하는 예제입니다.

시리얼 모니터의 설정으로는
Serial.begin(통신속도); 해당 통신속도로 시리얼 통신을 시작합니다.

Serial.print(내용); 내용에는 원하는 문자나 변수를 넣게 되면 시리얼 모니터 창으로 출력되며 줄 바꿈이 적용되지 않습니다.

Serial.print(내용); 내용에는 원하는 문자나 변수를 넣게 되면 시리얼 모니터 창으로 출력되며 줄 바꿈이 적용됩니다.

사용 예시 6

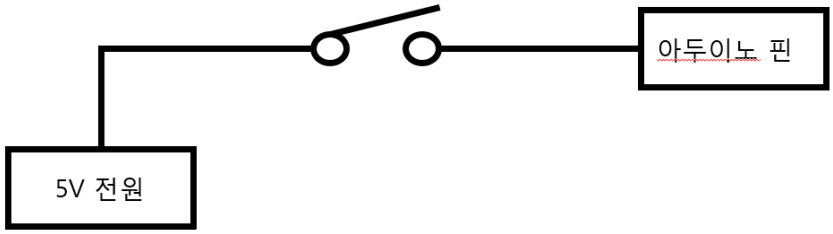


다음은 버튼의 사용에 대해서 알아보겠습니다.
아래의 프로그램을 참고하여 주세요

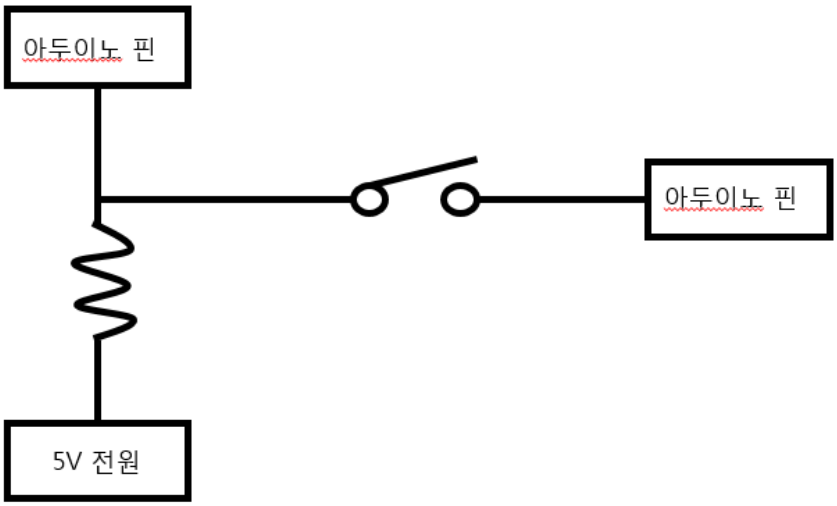
```
void setup() {
  // put your setup code here, to run once:
  // 시리얼 설정
  Serial.begin(9600);
  // 스위치 입력 설정
  pinMode(A3, INPUT);   // SWITCH_ENTER
  pinMode(A4, INPUT);   // SWITCH_DOWN
  pinMode(A5, INPUT);   // SWITCH_UP
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.print("Switch(ENTER) : " );
  Serial.println(digitalRead(A3));
  Serial.print("Switch(DOWN) : " );
  Serial.println(digitalRead(A4));
  Serial.print("Switch(UP) : " );
  Serial.println(digitalRead(A5));
  delay(1000); // 1초 지연
}
```

버튼은 현재 풀 업의 상태로 있습니다.
풀 업이란. 스위치가 아두이노 핀에 연결될 때 저항이 없이 바로 연결이 되면 플로팅이라는 붕 떠있는 상태가 됩니다. 이는 HIGH 도 LOW 도 아닌 애매모호한 상태로 있는 것입니다. 이러한 상태는 오작동을 유발하기 때문에 이러한 현상을 없애기 위해 풀 업이나 풀 다운으로 연결을 해 주어야 합니다.
연결은 다음과 같습니다.



이는 아두이노에 스위치를 바로 연결한 모습입니다.
스위치가 닫히면 5V 전원이 들어가 HIGH 상태를 읽을 수 있으나 열려 있는 경우에는 정확한 상태를 알 수 없는 플로팅 상태가 되어있습니다.



이렇게 연결해주면 스위치가 닫히면 LOW 상태가 되고 열려 있는 상태에는 HIGH 상태를 유지하는 풀 업회로가 구성되는 것입니다.

이미 제품에는 이 회로가 구성 되어있습니다.
따로 구성할 필요가 없습니다.

digitalRead(핀 번호); 핀 번호로 입력되는 신호 값을 디지털 형태로 출력합니다.

예제에서는
A3 번 핀부터 A5 번 핀까지 모두 입력으로 설정이 되어있습니다.
loop 문에서 이 핀들의 상태를 읽어 각각의 핀의 상태를 시리얼 모니터에 출력하는 예제입니다.

이렇게 풀 업 회로로 만들어진 스위치의 상태 값을 읽었습니다.

사용 예시 7

다음은 RGB LED 를 제어하는 것입니다.
애노드 타입의 RGB LED 이기 때문에 핀은 LOW 에 가까워질수록 밝아집니다.

다음 예제를 보고 따라해 봅니다.

```
void setup() {
  // put your setup code here, to run once:
  pinMode(3, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  Serial.begin(9600);

  analogWrite(3, 255); // Blue
  analogWrite(5, 255); // Green
  analogWrite(6, 255); // Red
}

void loop() {
  // put your main code here, to run repeatedly:

  for (int a = 255; a >= 0; a--) // Blue
  {
    analogWrite(3, a);
    Serial.println(a);
    if (a <= 0)analogWrite(3, 255);
    delay(10);
  }
  delay(1000);
  for (int b = 255; b >= 0; b--) // Green
  {
    analogWrite(5, b);
    Serial.println(b);
    delay(10);
    if (b <= 0)analogWrite(5, 255);
  }
  delay(1000);
  for (int c = 255; c >= 0; c--) // Red
```

```
  {
    analogWrite(6, c);
    Serial.println(c);
    delay(10);
    if (c <= 0)analogWrite(6, 255);
  }
  delay(1000);

}
```

해당 예제는 아날로그 출력을 이용한 예제입니다.
analogWrite 를 이용해서 0~255 까지의 값을 출력하는데 for 문을 이용해서 255 에서 0 으로 값을 줄여주며 밝기를 변화시켜줍니다.
RGB 값을 각각 제어하면서 원하는 색상을 표현하는 것이 가능합니다.

* 기술 문의

e-mail help@bitbus.co.kr
홈페이지 www.bitbus.co.kr

* 제품 설명서 업데이트 이력 *

- V1.0 신규 작성.