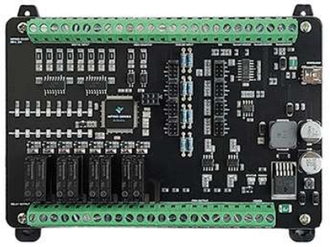
MPINO SERIES MPINO-8A8R

사 용 설 명 서

저희 ㈜아이로직스 제품을 구입해 주셔서 감사합니다.



사용 전에 안전을 위한 주의사항을 반드시 읽고 사용하십시오.

❒ 안전을 위한 주의사항

※ '안전을 위한 주의사항'은 제품을 안전하고 올바르게 사용하여 사고나 위험을 미리 막기 위한 것이므로 반드시 지켜야 합니다.

※ 주의사항은 '경고'와 '주의' 두가지로 구분되어 있으며,
'경고'와 '주의'의 의미는 다음과 같습니다.
지시사항을 위반하였을 때.

⚠**경고** 심각한 상해나 사망이 발생할 가능성이 있는 경우

⚠**주의** 경미한 상해나 제품 손상이 발생할 가능성이 있는 경우

※ 제품과 취급설명서에 표시된 그림기호의 의미는 다음과 같습니다.

⚠는 특정조건 하에서 위험이 발생할 우려가 있으므로 주의하라는 기호입니다.

⚠ 경고

1. 인명이나 재산상에 영향이 큰 기기(예: 원자력 제어장치, 의료기기, 선박, 차량, 철도, 항공기, 연소장치, 안전장치, 방법/방재장치 등)에 사용할 경우에는 반드시 2중으로 안전장치를 부착한 후 사용해야 합니다.
화재, 인사사고, 재산상의 막대한 손실이 발생할 수 있습니다.

2. 자사 수리 기술자 이외에는 제품을 개조하지 마십시오.
감전이나 화재의 우려가 있습니다.

⚠ 주의

1. 실외에서 사용하지 마십시오.
제품의 수명이 짧아지는 원인이 되며 감전의 우려가 있습니다.
본 제품은 실내 환경에 적합하도록 제작되었습니다. 실내가 아닌 외부환경으로부터 영향을 받을 수 있는 장소에서 사용할 수 없습니다.
(예 : 비, 황사, 먼지, 서리, 햇빛, 결로 등)

2. 인화성, 폭발성 가스 환경에서 사용하지 마십시오.
화재 및 폭발의 우려가 있습니다.

3. 사용 전압 범위를 초과하여 사용하지 마십시오.
제품이 파손될 수 있습니다.

4. 전원의 극성 등 오배선을 하지 마십시오.
제품이 파손될 수 있습니다.

5. 진동이나 충격이 많은 곳에서 사용하지 마십시오.
제품이 파손될 수 있습니다.

6. 청소 시 물, 유기 용제를 사용하지 마십시오.
감전 및 화재의 우려가 있습니다.

❒ 손해배상책임

㈜아이로직스는 제품을 사용하다 발생하는 인적, 물적자원에 대해 책임을 지지 않습니다. 충분한 테스트와 안전장치를 사용하여 주시기 바랍니다.

❒ 사양서

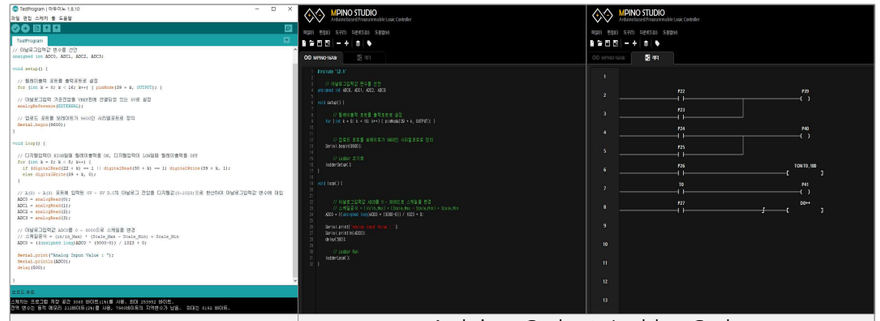
구 분	개 수	접점명	설 명
전 원	-	전원전압	• DC 12 ~ 24V
디지털 입력	8 포인트 (절연) (양방향)	I(22) ~ I(29)	• 오퍼레이팅 입력 전압 : DC 0 ~ 80V • HIGH 인식 전압 :DC 5V 이상 • 입력저항 : 2.2kΩ
릴레이 출력	8 포인트 (절연)	R(62) ~ R(69)	• 오퍼레이팅 연결 전압 - 0 ~ 30V D.C , 0 ~ 250V A.C • 최대 출력 허용전류 : 5A / 포인트
아날로그 입력	4 포인트 (비절연)	A(0) ~ A(3)	• 오퍼레이팅 입력 전압 : DC 0 ~ 5V • 오퍼레이팅 입력 전압 : DC 0 ~ 10V • 오퍼레이팅 입력 전류: DC 0 ~ 20mA • 온도센서 : NTC 10kΩ(25℃) β-3950 • 분해능 : 10Bit (0~1023) • 입력저항 : 2kΩ (0~5V 전압입력) • 입력저항 : 4kΩ (0~10V 전압입력) • 입력저항 : 250Ω (전류입력) • 입력저항 : 10kΩ Pull-Up (온도센서)
고속펄스 입력	2 포인트 (절연 / 2채널)	TCNT1, TCNT5	• 오퍼레이팅 입력 전압 : DC 0 ~ 80V • HIGH 인식 전압 : DC 5V 이상 • LOW 인식 전압 : DC 2V 이하 • 최대입력 주파수 : 5Khz • TCNT1 : TIMER1 자원사용 • TCNT5 : TIMER5 자원사용
펄스 출력	6 포인트 (비절연 / 2채널)	PWM 5,2,3 PWM 6,7,8	• 오퍼레이팅 출력 전압 - LOW (0V D.C), HIGH (5V D.C) • 오퍼레이팅 최대 출력 전류 : 30mA • 출력 저항: 150Ω (쇼트 보호저항) • Max 16Bit • PWM5,2,3 : TIMER3 자원사용 • PWM6,7,8 : TIMER4 자원사용
통신 채널	4 채널 (비절연)	I2C, RS232, RS485, UART	• I2C 1채널 지원 • RS232 1채널 지원 (Serial1) • RS485 1채널 지원 (Serial2) • UART 1채널 지원 (Serial3)

❒ 메모리 사양서

✎ 256Kbyte Flash Memory (8K byte BootLoader Memory)
✎ 8Kbyte Data Memory
✎ 4kByte EEPROM Memory

❒ 프로그램 코딩 및 다운로드 소프트웨어

✎ MPINO 제품은 Arduino IDE와 MPINO STUDIO를 이용하여 프로그램할 수 있습니다.
✎ Arduino IDE는(https://www.arduino.cc/en/Main/Software)에서 다운로드 받을 수 있습니다.
✎ MPINO STUDIO는 아이로직스 쇼핑몰 자료실에서 다운로드 받을 수 있습니다.
(http://www.ilogics.co.kr/page/07_view.php?idx=365&startPage=)



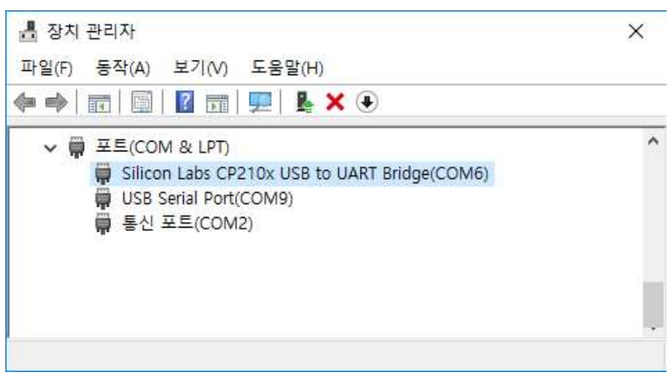
Arduino IDE | Arduino Code + Ladder Code MPINO STUDIO

❒ 사용방법 [요약]

✎ 컴퓨터의 USB포트와 제품(MPINO-8A8R)에 "MP 다운로드 케이블"을 연결합니다.

✎ 아이로직스 자료실에서 Silicon Labs CP210x Usb Driver를 다운로드를 받아 설치합니다. (http://ilogics.synology.me/Drivers/CP210x_Windows_Drivers.zip)

✎ 윈도우의 장치관리자에서 COM포트를 확인합니다.
(PC와 MPINO-8A8R 제품이 USB로 연결되어 있어야 합니다)



✎ Arduino IDE에서 위에서 확인한 COM포트를 선택합니다. (툴 -> 포트)

✎ Arduino IDE에서 Arduino Mega or Mega 2560을 선택합니다. (툴 -> 보드)

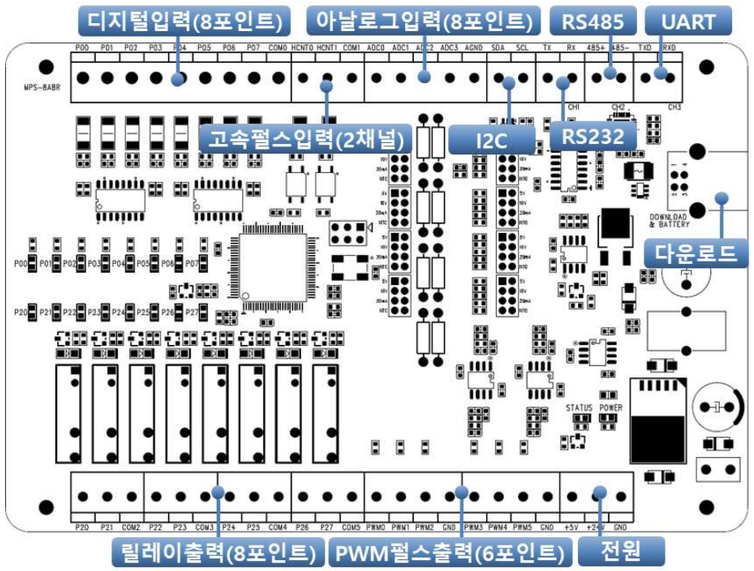
✎ 프로그래밍을 하고, 업로드를 합니다.

❒ 명령어 설명서

✎ Arduino IDE에서 도움말 -> 참조를 실행하거나 다음 링크에서 확인할 수 있습니다.
(https://www.arduino.cc/reference/en/)

✎ EEPROM과 I2C(Wire) 등을 보다 쉽게 사용할 수 있는 라이브러리는 다음 링크에서 확인할 수 있습니다.
(https://www.arduino.cc/en/Reference/Libraries)

❒ 기능별 위치



❒ 전원

✎ 전원입력은 12V ~ 24V D.C를 사용할 수 있습니다. 12V ~ 24V D.C는 LM2576 DC-DC Regulator를 통하여 5V D.C로 전환되어 내부회로에 전원을 공급합니다.

✎ 다운로드 포트에 USB 연결선으로 컴퓨터와 연결하면, 컴퓨터의 5V 전원을 사용하여 제품이 동작됩니다. 아날로그 기준전압은 REF3025 레퍼런스 레귤레이터를 사용하여 2.5V가 사용됩니다. 때문에, USB 전원으로도 모든 기능을 제한없이 사용하실 수 있습니다.

✎ 전원입력포트의 +5V 단자는 전원입력 또는 전원출력 공통으로 사용할 수 있습니다. 단, 전원출력 사용할 경우, 최대 1A 이하로 사용해 주시기 바랍니다.

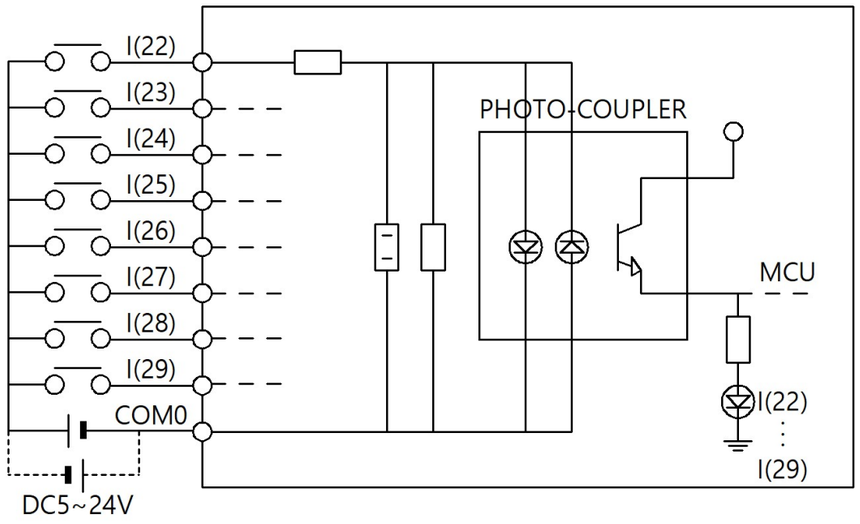
❒ 정전유지

✎ MPINO-8A8R 제품은 5V전원으로 모든 동작이 가능하도록 설계되어 있습니다. 따라서, DOWNLOAD (USB-B TYPE) 컨넥터에 배터리 등을 연결하여, 정전시에도 제품 동작 및 메모리를 유지하게 할 수 있습니다.

✎ 배터리 연결이 불가할 경우와 장기간 정전시에도 데이터의 보존을 원할 경우에는 MCU에 내장된 EEPROM을 사용해주시기 바랍니다. 비활성 메모리인 EEPROM을 이용하여 메모리를 보존할 수 있습니다. 단, EEPROM은 100,000번 이상 기록(Write)를 할 경우, 해당 섹션의 불량이 발생할 수 있으므로 수시로 변경되는 데이터를 기록하는 것은 올바르지 않습니다.

❒ 디지털 입력

✎ I(22) ~ I(29)에 5V ~ 24V D.C가 스위치, 센서 등에 의해 입력된다면, COM0는 GND를 연결해야 합니다. 반대로, I(22) ~ I(29)에 GND가 스위치, 센서 등에 의해 입력된다면, COM0는 5V ~ 24V D.C를 연결해야 합니다.



✎ 관련 명령어
pinMode(Pin, INPUT/OUTPUT) Pin포트를 INPUT 또는 OUTPUT으로 설 정.
digitalRead(Pin) Pin포트의 입력상태를 “0” 또는 “1”로 반환.
digitalWrite(pin, 0/1) pin포트의 출력상태를 LOW 또는 HIGH로 변환.

❒ 디지털 입/출력 예제

```
void setup() {
  pinMode(62, OUTPUT); // R(62)를 출력모드로 설정합니다.
}
void loop() {
   // I(22)가 HIGH 이면, R(62)를 ON 시킵니다.
  if (digitalRead(22) == 1) { digitalWrite(62, HIGH); }
  // I(22)가 HIGH가 아니면, 즉 LOW 이면, R(62)를 OFF 시킵니다.
  else { digitalWrite(62, LOW); }
}
```

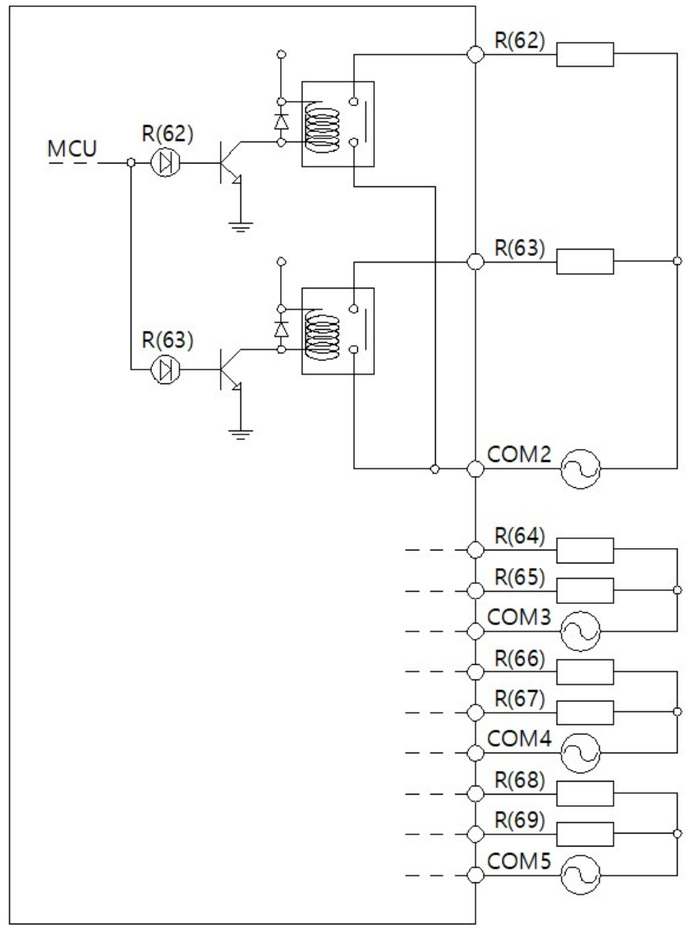
❒ 상태 LED

✎ LED_BUILTIN 변수명 또는 D13핀으로 제품에 삽입되어 있는 STATUS LED를 ON/OFF 시킬 수 있습니다.

```
void setup() {
  pinMode(LED_BUILTIN, OUaTPUT); //LED_BUILTIN을 출력모드로 설정
}
void loop() {
   // I(22)가 HIGH 이면, LED_BUILTIN를 ON 시킵니다.
  if (digitalRead(22) == 1) { digitalWrite(LED_BUILTIN, HIGH); }
  // I(22)가 HIGH가 아니면, 즉 LOW 이면, LED_BUILTIN를 OFF 시킵니다.
  else { digitalWrite(LED_BUILTIN, LOW); }
}
```

❒ 릴레이 출력

✎ COM2에 연결한 전원이 R(62),R(63)으로 출력됩니다. 릴레이는 스위치를 누르는 것과 같이 COM2와 R(62),R(63)을 물리적으로 연결하기 때문에 D.C 와 A.C를 모두 ON/OFF 시킬 수 있습니다.



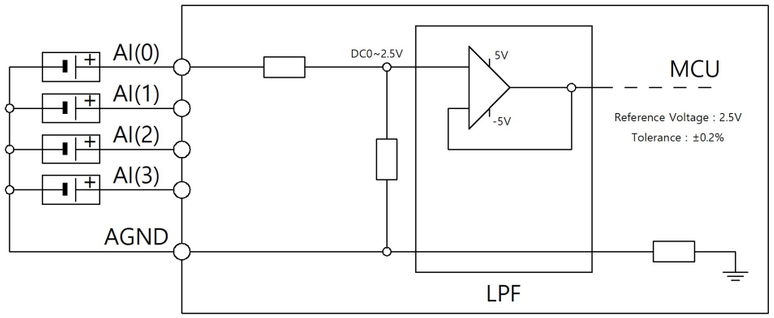
❒ 1초마다 출력을 ON/OFF 시키는 예제

✎ delay(ms) 명령어를 사용하여 시간지연을 사용할 수 있습니다.

```
void setup() {
  pinMode(62, OUTPUT); // R(62)를 출력모드로 설정합니다.
}
void loop() {
  digitalWrite(62, HIGH); // R(62)를 ON 시킵니다.
  delay(1000);              // 1000ms 동안 기다립니다.
  digitalWrite(62, LOW);   // R(62)를 OFF 시킵니다.
  delay(1000);              // 1000ms 동안 기다립니다.
}
```

❒ 아날로그 입력

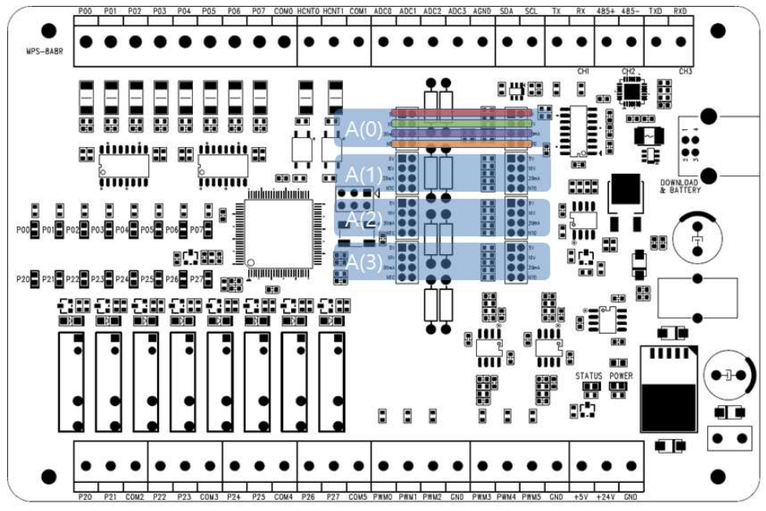
✎ 아날로그 입력포트 A(0) ~ A(3)에 입력된 아날로그 신호를 analogRead(pin) 명령어를 사용하여 디지털 값으로 변환하여 사용합니다.



✎ 아날로그 입력범위 변경

아날로그 입력포트는 아래의 4가지중 하나를 선택하여 사용할 수 있습니다.

① 0V ~ 5V D.C (적색부분의 헤더핀 두 부분을에 점퍼핀 연결) (Default)
② 0V ~ 10V D.C (초록색부분의 헤더핀 두 부분에 점퍼핀 연결)
③ 0mA ~ 20mA D.C (보라색부분의 헤더핀 두 부분에 점퍼핀 연결)
④ NTC 10KΩ 온도센서(주황색부분의 헤더핀 두 부분에 점퍼핀 연결)



❒ 아날로그 입력 프로그램 예

✎ A(0) 포트에 입력된 0V~5V의 전기신호를 0~1023의 디지털값으로 변환하여 ADC0 변수에 저장하는 프로그램입니다. analogReference(EXTERNAL)로 기준전압 값을 설정하고, analogRead(pin)으로 아날로그값을 읽어 옵니다.

```
unsigned int ADC0; // ADC0 변수 생성

void setup() {
  // 아날로그 최대값 기준을 VREF핀에 입력된 전압으로 설정
  analogReference(EXTERNAL);
}
void loop() {
// A(0)에 입력된 0~5V를 0~1023의 디지털값으로 변환하여 ADC0에 저장
  ADC0 = analogRead(0);
}
```

✎ 관련 명령어

- **analogReference(EXTERNAL)** 아날로그 입력의 기준전압을 MCU의 Vref핀에 연결된 전압으로 설정합니다.
- **analogRead(Pin)** : Pin 포트의 아날로그 신호를 디지털 수치로 변환하여 반환합니다.

❒ 아날로그 입력값 스케일 계산

✎ analogRead(pin) 명령어를 이용하여 읽어온 0 ~ 1023 디지털값을 프로그램에서 로직으로서 사용하기 위해 실제 센서의 Range로 디지털값을 변환하기 위해 스케일 계산법으로 디지털 범위를 재설정하는 방법입니다.

✎ Scale = (In / 1023) * (Scale_Max - Scale_Min) - Scale_Min

```
unsigned int ADC0; // ADC0 변수 생성

void setup() {
  // 아날로그 최대값 기준을 VREF핀에 입력된 전압으로 설정
  analogReference(EXTERNAL);
}

void loop() {
   //A(0)에 입력된 0~5V를 0~1023의 디지털값으로 변환하여 ADC0에 저장
  ADC0 = analogRead(0);
  // 스케일공식 = (in/in_Max) * (Scale_Max - Scale_Min) + Scale_Min
  // 0V일때는 0, 5V일때는 3000으로 디지털값 범위를 재설정.
  ADC0 = ((unsigned long)ADC0 * (3000-0)) / 1023 + 0;
}
```

❒ NTC 써미스터를 이용한 온도값 읽어오는 예제

✎ 아날로그 포트를 NTC 모드로 점퍼핀으로 설정해야 합니다. NTC 모드는 10kΩ (±1%) Pull-Up 저항이 2.5V(±0.2%) 전압으로 걸려 있습니다.

✎ NTC는 25℃일 때, 10kΩ 저항이 되는 제품을 사용해야 합니다.

✎ 예제는 β값이 3950인 NTC입니다. β값이 다르면 3950.0F를 변경해야 합니다.

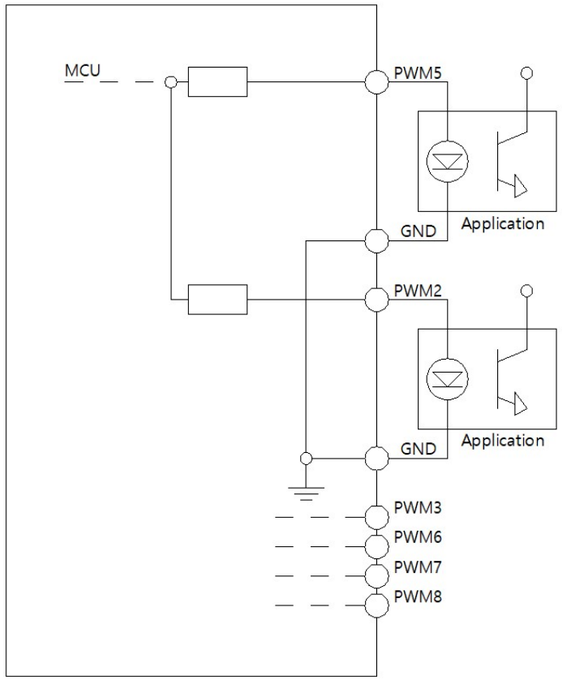
```
unsigned int Temp // Temp 변수 생성

void setup() {
    // 아날로그 최대값 기준을 VREF핀에 입력된 전압으로 설정
  analogReference(EXTERNAL);
}

void loop() {
    // 아날로그입력 0번 포트의 온도값을 읽어와서 Temp변수에 저장.
  Temp = ntcRead(analogRead(0)); // Temp가 251 이면, 25.1도입니다.
 }
int ntcRead(unsigned int RawADC)
{
        float v;
        v = (1023.0F / (float)RawADC) - 1.0F;
        v = 10000.0F / v;
        float steinhart;
        steinhart = v / 10000.0F;
        steinhart = log(steinhart);
        steinhart /= 3950.0F;
        steinhart += 1.0F / (25.0F + 273.15F);
        steinhart = 1.0F / steinhart;
        steinhart -= 273.15F;
        return (unsigned int)(steinhart * 10);
}
```

❒ PWM 고속펄스 출력

✎ PWM은 Pulse Width Modulation 의 약자로서 Width를 조절할 수 있는 펄스라는 의미입니다. PWM은 다른 장비와의 인터페이스로 많이 사용되어 집니다. 많이 사용하는 곳은 모터 드라이버의 속도 및 회전각을 조절하기 위해 사용됩니다.



✎ 관련 명령어

analogWrite(Pin, Duty) : PWM(Pin)포트에 Duty길이의 펄스를 출력합니다.

Pin : PWM5는 5, PWM3은 3을 사용합니다.

Duty : 0 ~ 255(Defaul)를 사용합니다.

✎ PWM은 MCU의 타이머 자원을 사용합니다.

PWM5,2,3은 타이머3을 PWM6,7,8은 타이머4 자원을 사용합니다.

❒ PWM 펄스출력 예제

✎ 디지털입력 I(22)가 ON되면, 고속펄스출력 PWM5포트에 Duty비가 50%인 연속적인 펄스를 출력하고 I(23)가 ON되면, 펄스출력을 정지시킵니다.

```
void setup(void) {
}

void loop(void) {
  if (digitalRead(22)==1) { analogWrite(5, 127); }
  else { analogWrite(5, 0); }
}
```

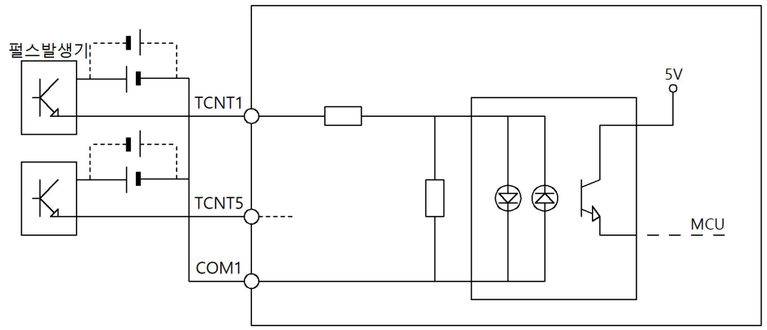
❒ PWM의 Duty를 16비트로 변경하는 방법

✎ MCU에 내장되어 있는 타이머 자원의 레지스터리를 수정하여 Duty의 최대값을 16비트인 65535로 변경할 수 있습니다.

```
void setup(void) {
    // PWM5,2,3의 Timer3의 카운트 최대값을 65535로 변경
  TCCR3A=0xAA; TCCR3B=0x1A; ICR3 = 65535;
    // PWM6,7,8의 Timer4의 카운트 최대값을 65535로 변경
  TCCR4A=0xAA; TCCR4B=0x1A; ICR4 = 65535;
}
void loop(void) {
  if (digitalRead(22)==1) { analogWrite(5, 32767); }
  else { analogWrite(5, 0); }
}
```

❒ 고속카운터 입력

✎ 고속으로 들어오는 펄스의 수를 카운트하기 위해 사용됩니다. loop(void) 함수문에서 디지털입력이 ON될 때 카운트를 하는 프로그램을 하였을 경우, loop(void)가 실행되는 시간보다 빠른 펄스를 감지를 못하기 때문에 사용됩니다. 많이 사용되는 곳은 모터의 엔코더의 펄스를 카운트하여 속도 또는 위치변화를 감지하기 위해 사용됩니다.



✎ 고속카운터는 MCU의 타이머 자원을 사용합니다.

✎ TCNT1은 타이머1을 TCNT5는 타이머5 자원을 사용합니다.

❒ 고속카운터 입력 사용방법 (16비트)

```
unsigned int HCNT1, HCNT5;

void setup(void) {
    // 타이머1 자원을 고속카운터 모드로 설정
  TIMSK1 = 0x00; TCCR1A = 0x00; TCCR1B = 0x1F; TCNT1 = 0x00;
    // 타이머5 자원을 고속카운터 모드로 설정
  TIMSK5 = 0x00; TCCR5A = 0x00; TCCR5B = 0x1F; TCNT5 = 0x00;
}

void loop(void) {
  HCNT1 = TCNT1; // 타이머1 카운트값을 HCNT1 변수에 저장
  HCNT5 = TCNT5; // 타이머5 카운트값을 HCNT5 변수에 저장
}

void hcntReset() {
    // 타이머1과 타이머5 카운트값을 0으로 리셋
  TCNT1 = 0; TCNT5 = 0;
}
```

❒ 고속카운터 입력 사용방법 (32비트)

✎ 고속카운터 입력이 16비트를 초과해서 사용해야 할 경우, MCU에 내장된 타이머 레지스터가 16비트이므로 오버플로우 인터럽트를 이용하여 오버플로우 된 값을 변수로 카운트해야 합니다.

```
unsigned long HCNT1, HCNT5;
unsigned int _ofcH1, _ofcH5;

void setup(void) {
    // 타이머1 자원을 고속카운터 모드로 설정
  TIMSK1 = 0x01; TCCR1A = 0x00; TCCR1B = 0x1F; TCNT1 = 0x00;
    // 타이머5 자원을 고속카운터 모드로 설정
  TIMSK5 = 0x01; TCCR5A = 0x00; TCCR5B = 0x1F; TCNT5 = 0x00;
}

void loop(void) {
    // 타이머1 카운트값과 오버플로우 카운트값을 HCNT1 변수에 저장
  HCNT1 = (unsigned long)_ofcH1 << 16 | TCNT1;
    // 타이머5 카운트값과 오버플로우 카운트값을 HCNT5 변수에 저장
  HCNT5 = l(unsigned long)_ofcH5 << 16 | TCNT5;
}
  // 타이머1 카운트 오버플로우 인터럽트
ISR ( TIMER1_OVF_vect )          { TCNT1=0;  _ofcH1++; }
  // 타이머5 카운트 오버플로우 인터럽트
ISR ( TIMER5_OVF_vect )          { TCNT5=0;  _ofcH5++; }
```

❒ 인터럽트 (attachInterrupt)

✎ 디지털신호의 입력을 받아 빠르게 처리해야 하는 사항이 있을 때 사용합니다.

✎ MPINO-8A8R에서는 PWM2와 PWM3 포트를 통하여 두 개의 인터럽트 핀을 사용할 수 있습니다.

✎ 입력전압은 DC 3V ~ 5V입니다. 과전압 인가시 MCU가 소손될 수 있습니다.

✎ 관련 명령어

attachInterrupt(digitalPinToInterrupt(pin)), ISR, mode);

- pin : 2 또는 3 (PWM2 또는 PWM3포트)
- ISR : 호출되는 함수명
- mode : LOW, CHANGE, RISING, FALLING
  - LOW : 하강검출 (입력상태가 ON에서 OFF로 될 때)
  - CHANGE : 변경검출 (입력상태가 변경될 때)
  - HIGH : 상승검출 (입력상태가OFF에서 ON으로 될 때)

✎ 인터럽트는 PWM2와 PWM3 포트를 통하여 사용이 가능합니다.

✎ PWM2 입력이 OFF였다가 ON될 때, _INT2() 함수를 호출하는 예제입니다.

```
void setup() {
  Serial.begin(9600); // 다운로드포트를 9600보레이트로 오픈
    // PWM2포트에 상승엣지 입력이 검출되면, _INT2 함수 호출실행
  attachInterrupt(digitalPinToInterrupt(2),_INT2,   RISING);
}
void loop() {
}

    // PWM2포트에 상승엣지 입력이 검출되면, 실행되는 함수.
void _INT2(){
    // 다운로드포트로 "ok'를 송신
  Serial.println("ok");
}
```

❒ 디버깅

✎ Debug는 Serial 함수를 이용해 주세요.

```
unsigned int ADC0; // 아날로그입력값 변수를 선언
void setup() {
  // 아날로그입력 기준전압을 VREF핀에 연결되어 있는 5V로 설정
  analogReference(EXTERNAL);
  Serial.begin(9600); // 업로드 포트를 보레이트가 9600인 시리얼포트로 정의
}
void loop() {
  // 디지털입력이 ON 되면 릴레이출력을 ON, OFF일때 릴레이출력을 OFF
  for (int k = 0; k < 8; k++) {
    if (digitalRead(22+k)==1) digitalWrite(62+k, 1);
    else digitalWrite(39 + k, 0);
  }
  // A(0)에 입력된 아날로그 신호를 0~1023으로 변환하여 ADC0 변수에 저장
  ADC0 = analogRead(0);
  // 아날로그입력값 ADC0을 스케일 연산하여 0 ~ 3000으로 범위 변경
  // 스케일공식 = (in/in_Max) * (Scale_Max - Scale_Min) + Scale_Min
  ADC0 = ((unsigned long)ADC0 * (3000-0)) / 1023 + 0;
  Serial.print("Analog Input Value : ");
  Serial.println(ADC0);
  delay(500);
}
```

❒ I2C 통신포트

✎ 1개의 I2C 통신포트를 지원합니다.

✎ 명령어는 링크를 참조해 주시기 바랍니다. https://www.arduino.cc/en/Reference/Wire

❒ 캐릭터 LCD 연결

✎ I2C 통신포트에 캐릭터 LCD를 연결하여 디스플레이를 구현할 수 있습니다.

✎ 아두이노 IDE에서 라이브러리 관리에서 “TM1637“로 검색하여 다양한 라이브러리를 사용하실 수 있습니다.

✎ 저희가 제공하는 통합라이브러리인 “ILIB”를 사용하실 수 있습니다.

✎ ILIB로 캐릭터 LCD를 제어하는 예제를 참고해 주시기 바랍니다.
(https://blog.naver.com/ilogics/222451135999)

❒ 시리얼 통신포트

통신	RS-232	RS-485	UART
키워드	Serial1	Serial2	Serial3

✎ 시리얼 통신방법은 아래의 아두이노에서 제공하는 설명을 참고해 주시기 바랍니다.
(아두이노 IDE의 Serial 함수 사용설명서)

✎ 산업현장에서는 산업 범용 프로토콜인 모드버스 통신 프로토콜을 주로 사용합니다.

✎ ILIB로 Modbus RTU Master를 사용하는 예제를 참고해 주시기 바랍니다.
(https://blog.naver.com/ilogics/222453991523)

✎ ILIB로 Modbus RTU Slave를 사용하는 예제를 참고해 주시기 바랍니다.
(https://blog.naver.com/ilogics/222453993604)

❒ MPINO STUDIO

✎ 저희 ㈜아이로직스에서는 산업에서 사용하기 쉽도록 Arduino 와 Ladder Logic을 모두 사용하여 프로그램 할 수 있는 MPINO STUDIO를 무료로 제공하고 있습니다. (단, MPINO-8A8R, MPINO-16A8R, MPINO-16A8R8T, MPAINO Series 에서만 사용이 가능합니다)

❒ MP STUDIO

✎ 저희 ㈜아이로직스에서는 Ladder Logic만을 사용하여 프로그램 할 수 있는 MP STUDIO를 무료로 제공하고 있습니다. MP STUDIO는 MPS 및 MPA 시리즈 제품군에 사용할 수 있습니다.

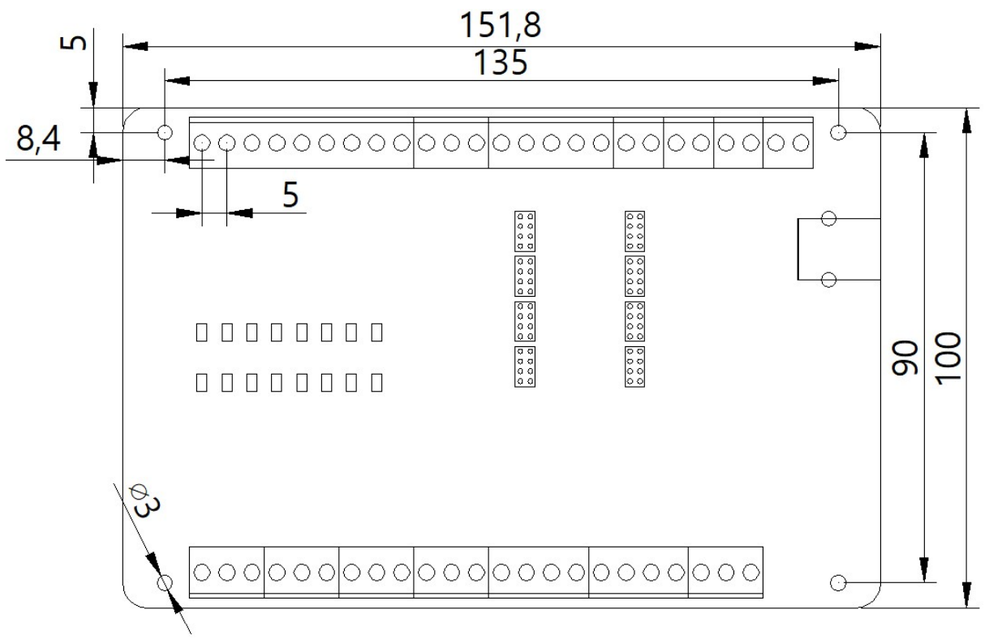
❒ 감사드립니다.

✎ 저희 (주)아이로직스의 제품을 구매해주셔서 감사드립니다.

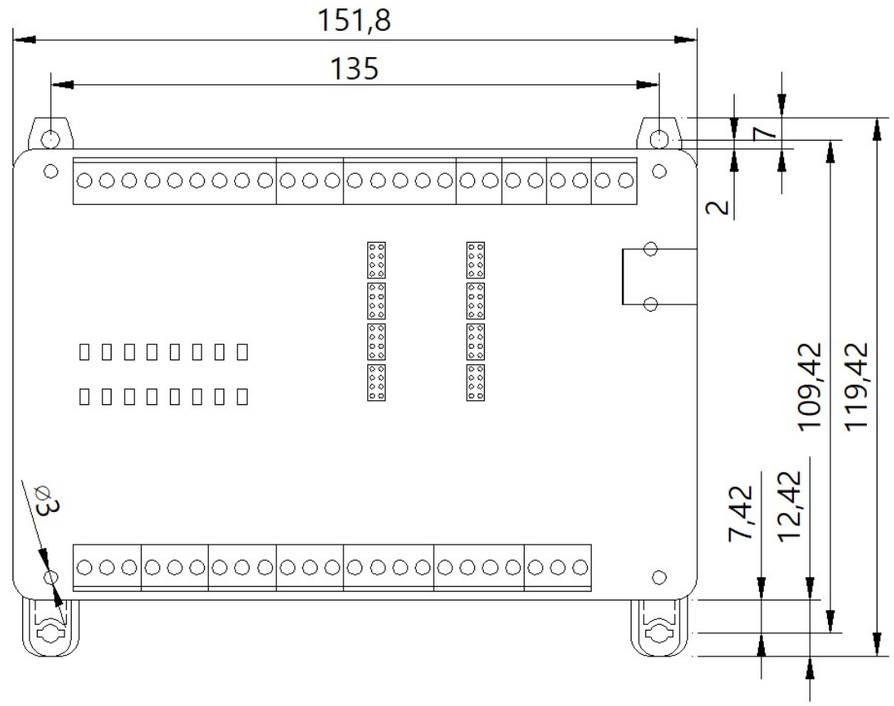
✎ 구매는 https://www.ilogics.co.kr 쇼핑몰에서 하실 수 있습니다.

✎ 구매/기술 상담은 0507-1362-5020으로 전화 주시기 바랍니다.
(상담시간은 오전10시~오후5시 입니다)

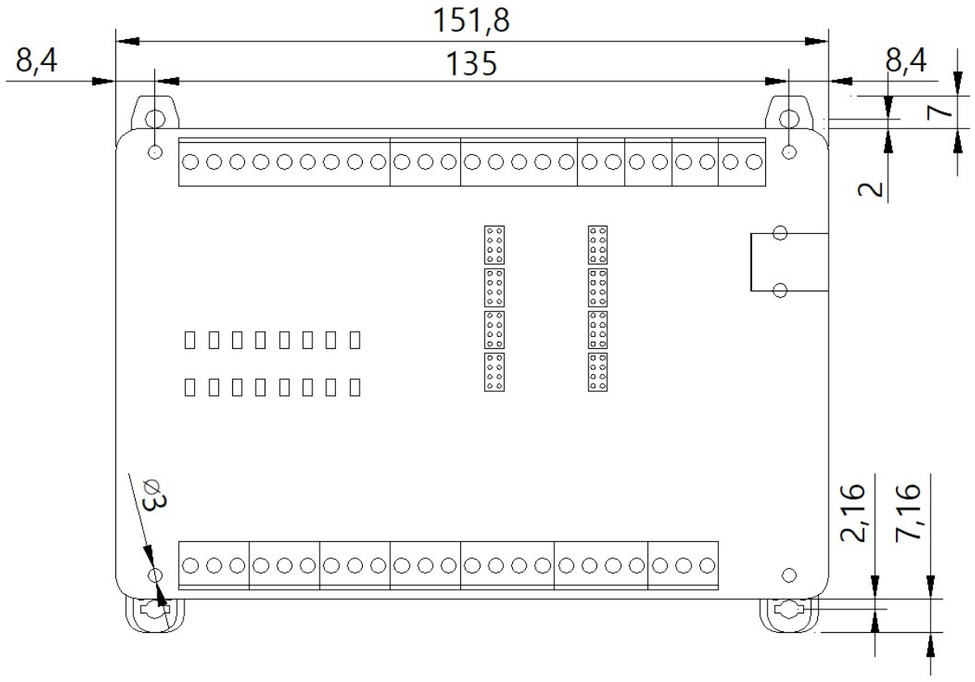
❒ DIMENSION (PCB / 딘레일 사용하지 않을 경우)



❒ DIMENSION (클립 열었을 때 / 딘레일 체결 전)



❒ DIMENSION (클립 닫았을 때 / 딘 레일 체결 후)



❒ DIMENSION (딘 레일 : 35mm)