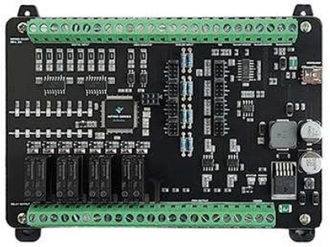
MPINO SERIES MPINO-8A8R

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 24V • DC 5V
디지털 입력	8 포인트 (절연) (양방향)	I(22) ~ I(29)	• 오퍼레이팅 입력 전압 : DC 0 ~ 80V
			• HIGH 인식 전압 :DC 5V 이상
			• LOW 인식 전압 : DC 2V 이하
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 • NTC 10kΩ(25℃) 온도센서
			• 분해능 : 10Bit (0~1023)
			• 입력저항 : 2kΩ (0~5V 전압입력) • 입력저항 : 4kΩ (0~10V 전압입력) • 입력저항 : 250Ω (전류입력) • 입력저항 : 10kΩ Pull-Up (온도센서)
펄스 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
펄스 출력	6 포인트 (비절연 / 2채널)	PWM 5,2,3 PWM 6,7,8	• 오퍼레이팅 출력 전압 - LOW (0 VDC), HIGH (5 VDC)
			• 오퍼레이팅 최대 출력 전류 : - 30mA
			• 출력 저항 - 150Ω (쇼트 보호저항)
			• INIT() 함수실행 후, 16bit로 사용
통신 채널	4 채널 (비절연)	I²C, RS232 RS485, UART	• 선택적 1채널 지원 - Modbus RTU Slave 지원 (I²C 제외)

❒ MPINO

• 아두이노의 C코드와 PLC의 LADDER LOGIC을 모두 사용하여 프로그램할 수 있도록 소프트웨어와 산업용 하드웨어를 제공하는 제품입니다.

• 아두이노 C코드와 동일한 명령어를 사용할 수 있습니다.

❒ 메모리 사양서

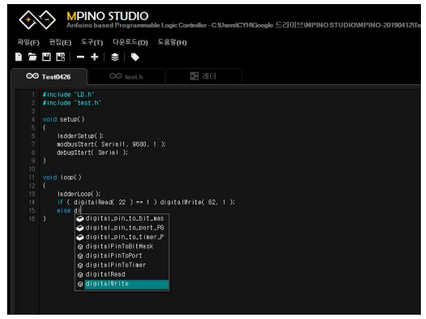
• 200Kbyte Flash Memory
• 7Kbyte Data Memory
• 4Kbyte EEPROM Memory (비휘발성 메모리)

❒ MPINO STUDIO 사용방법 [요약]

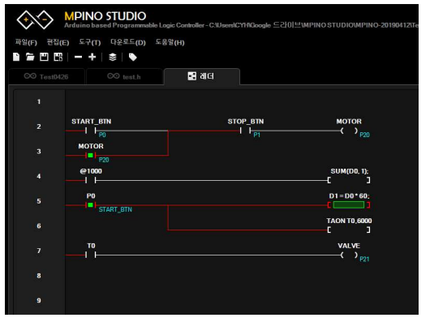
① MPINO STUDIO 설치

• 아이로직스 쇼핑몰 자료실 에서 "MPINO STUDIO"를 다운로드 받고, 설치합니다.
(http://www.ilogics.co.kr/page/07_view.php?idx=365&startPage=)

• 아래는 "MPINO STUDIO"의 화면입니다. 자세한 사용법은 MPINO STUDIO 사용설명서를 참조 바랍니다.



[아두이노 C언어]



[LADDER LOGIC]

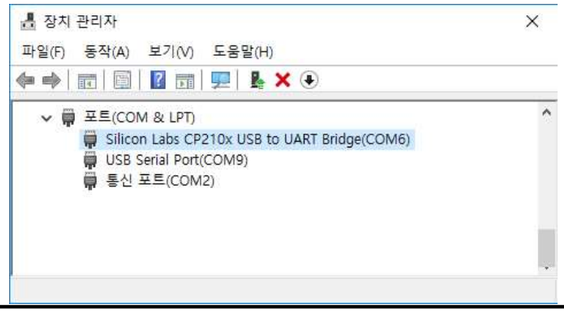
② 컴퓨터의 USB포트와 제품(MPINO-8A8R)에 "MP 다운로드 케이블"을 연결합니다.

③ 아이로직스 자료실에서 Silicon Labs CP210x Usb Driver를 다운로드를 받아 설치합니다. (http://ilogics.synology.me/Drivers/CP210x_Windows_Drivers.zip)

④ "MPINO STUDIO"를 실행하고 도구창에서 "도구 -> 디바이스 -> 아이로직스-> MPINO-8A8R" 을 선택합니다.

⑤ 다운로드 포트를 확인합니다 설정합니다.

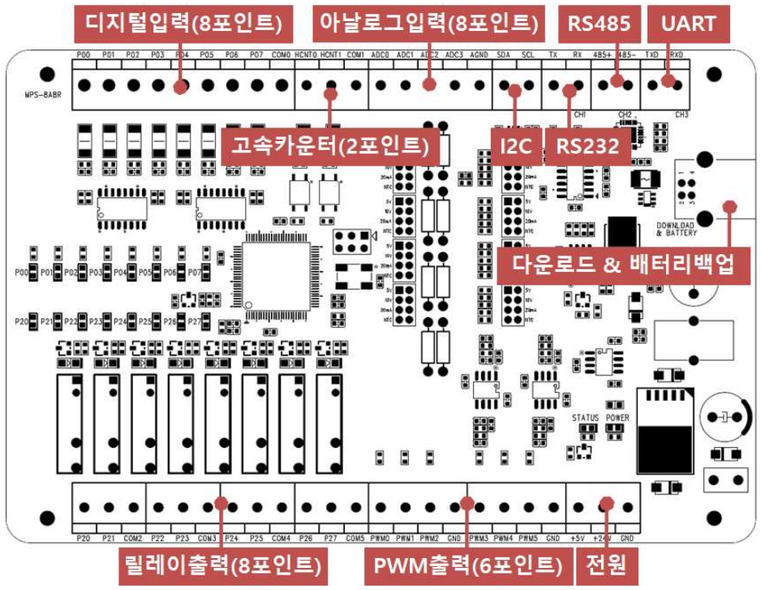
• 윈도우의 장치관리자에서 아래 그림처럼 선택된 COM포트를 확인합니다.



⑥ MPINO STUDIO의 도구창에서 "도구 -> 포트"의 COM포트 목록중에 위에서 확인한 다운로드 COM포트를 선택합니다.

⑦ 프로그램을 작성하고 다운로드 합니다.

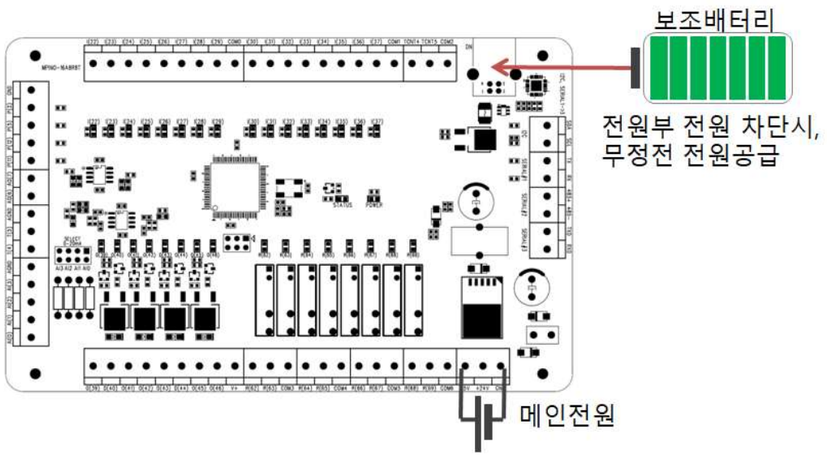
❒ 기능별 위치



❒ 전 원

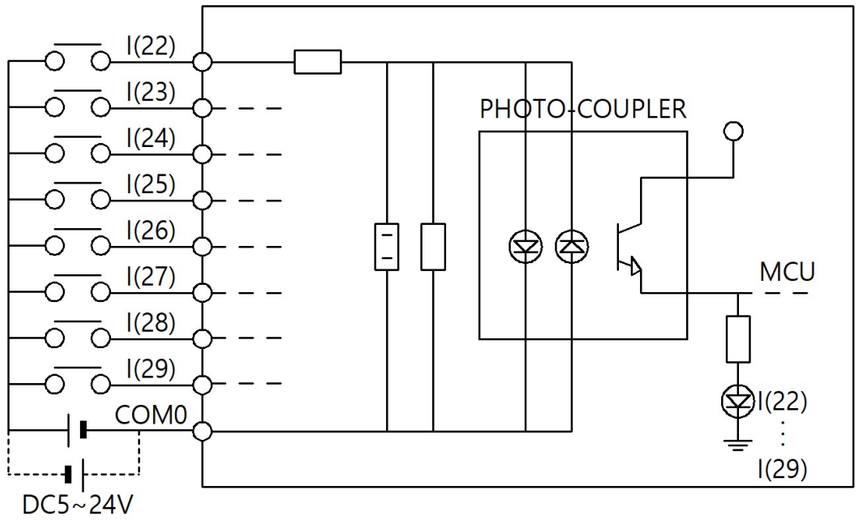
- +24V D.C 또는 +5V D.C를 사용할 수 있습니다.
- +24V 단자와 GND 단자에 +24V D.C를 연결하여 사용하실 수 있습니다.
  이 경우, +5V 단자를 통하여 +5V D.C를 최대 1A까지 사용할 수 있습니다.
- +5V 단자와 GND단자에 +5V D.C를 연결하여 사용하실 수 있습니다.
- 다운로드 포트에 보조배터리 등을 연결하여, 전원공급을 할 수 있습니다.

- 정전유지를 위하여 전원부에 전원을 공급하고 보조배터리를 연결하여 사용하실 수 있습니다. 전원부의 전원이 차단되어도, 배터리백업의 전원으로 무정전 전원공급이 됩니다. (스마트폰 보조배터리중 일부는 충전중이 아닐 때, 자동으로 전원이 꺼지는 제품들이 있습니다. 그러한 보조배터리는 사용하실 수 없습니다.)



❒ 디지털 입력

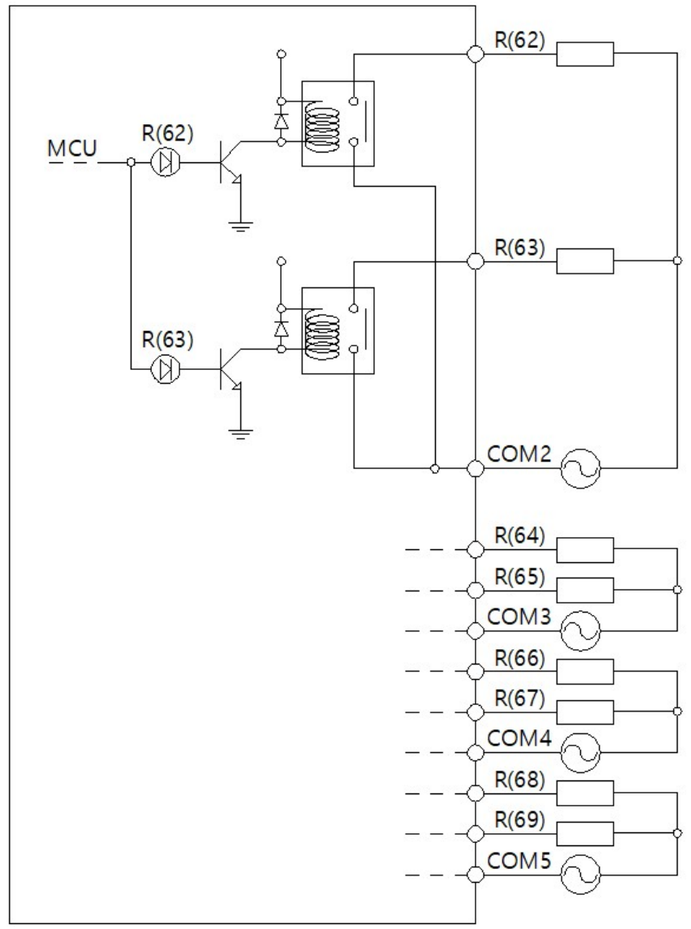
- 입력포트 I(22) ~ I(29)은 +5V ~ +24V D.C의 전압이 각각의 터미널블럭에 인가되었을 때, 프로그램에서 각 포트의 상태가 High의 상태가 됩니다.



- 관련 명령어 (디지털 입력/출력)
  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.

❒ 릴레이 출력

- 프로그램에서 출력접점 R(62) ~ R(69)의 상태가 On될 때, 각각의 릴레이 접점이 COM과 물리적으로 연결되는 상태가 됩니다.



❒ 디지털 입력 및 릴레이 출력 프로그램 간단 예제

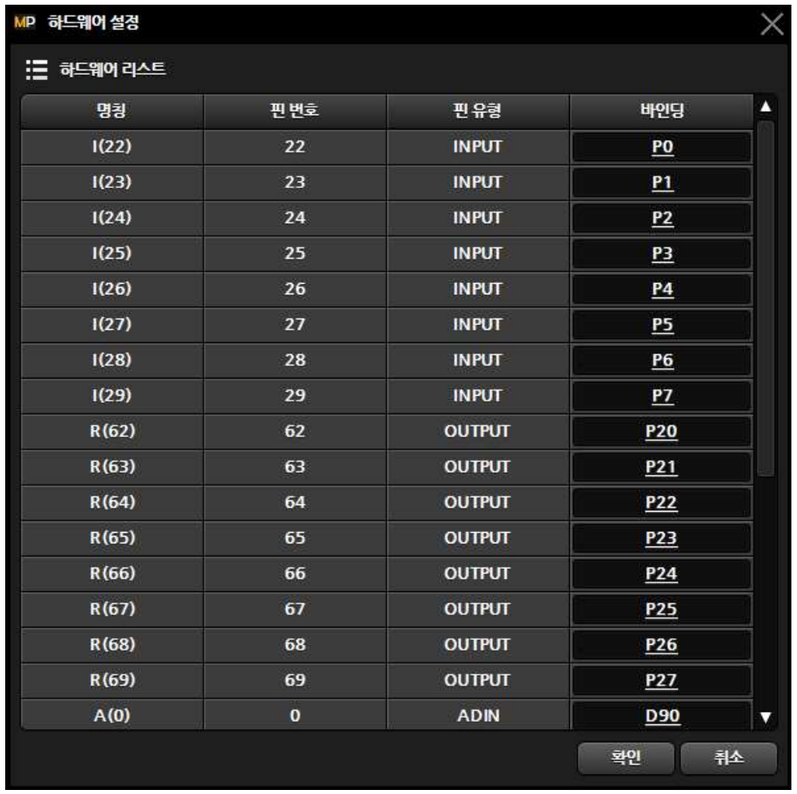
- MPINO STUDIO : C코드를 사용할 경우

pinMode() 함수로 입/출력 설정은 MPINO STUDIO에서 디바이스를 선택함으로서 자동으로 실행되므로 코딩하실 필요가 없습니다

```
#include "LD.h" // Ladder 명령어를 사용하기 위한 함수참조

void setup() {
    ladderSetup(); // Ladder 초기화
}
void loop() {
    // 22번 디지털 입력이 ON되면, 릴레이 출력 62번을 ON
    if (digitalRead(22) == 1) digitalWrite(62, 1);
    // 그렇지 않다면, 릴레이 출력 62번을 OFF
    else digitalWrite(62, 0);
    ladderLoop(); // Ladder LOGIC 실행
}
```

• MPINO STUDIO : LADDER LOGIC을 사용할 경우

하드웨어 설정에서 I(22)를 P0로 바인딩하고, R(62)를 P20으로 바인딩합니다. P0... P20.. 등의 바인딩은 P로 시작해야 하며 숫자는 사용자가 결정합니다.

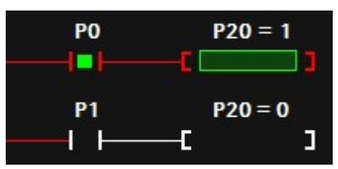


디지털 입력 I(22) 포트에 전기신호가 입력되면 I(22):P0 접점이 On되고, R(62):P20 출력접점이 On되어 R(62) 포트와 COM2 포트가 연결됩니다.
반대로 I(22):P0 접점이 OFF되면, R(62):P20 점점도 OFF됩니다.

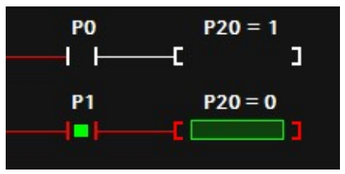
[P0 Off -> P20 Off]

[P0 On -> P20 On]

I(22):P0 점점이 ON되면, R(62):P20 출력접점이 ON되고,
I(22):P1 접점이 ON되면, R(62):P20 출력점점이 OFF됩니다.



[P0 ON -> P20 ON]



[P1 On -> P20 OFF]

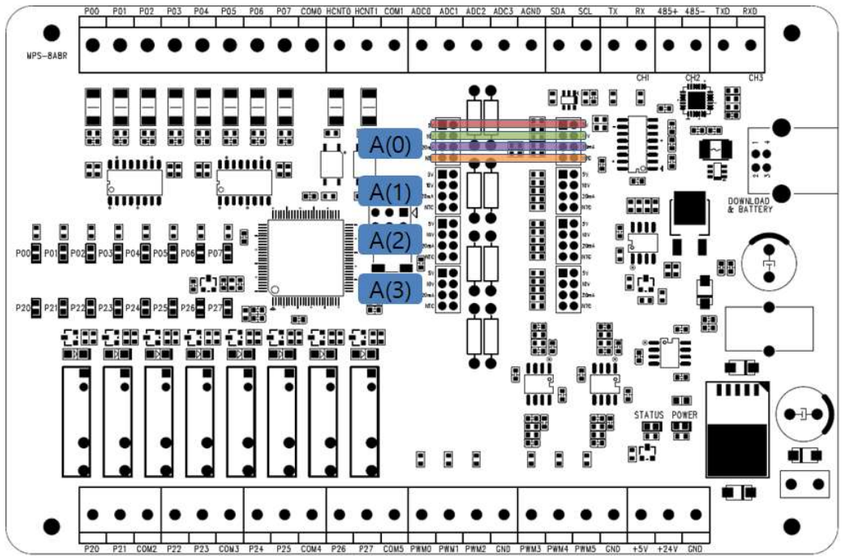
❒ 아날로그 입력

• 아날로그 입력포트 A(0) ~ A(3)에 입력된 아날로그 신호를 디지털 값(0~1023, 10 BIT)으로 변환하여 사용합니다.

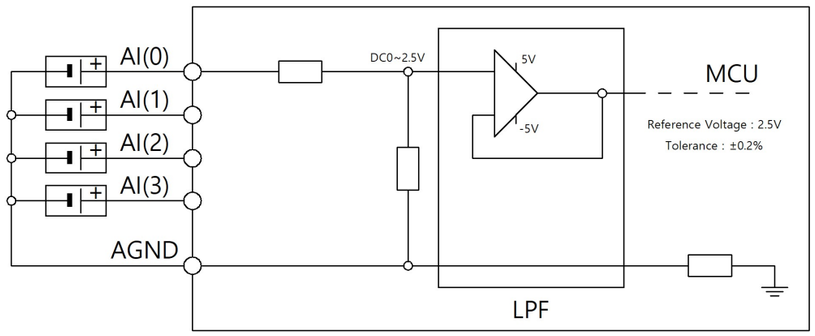
• 아날로그 입력 선택
아날로그 입력포트는 아래의 4가지중 하나를 선택하여 사용할 수 있습니다

.

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)



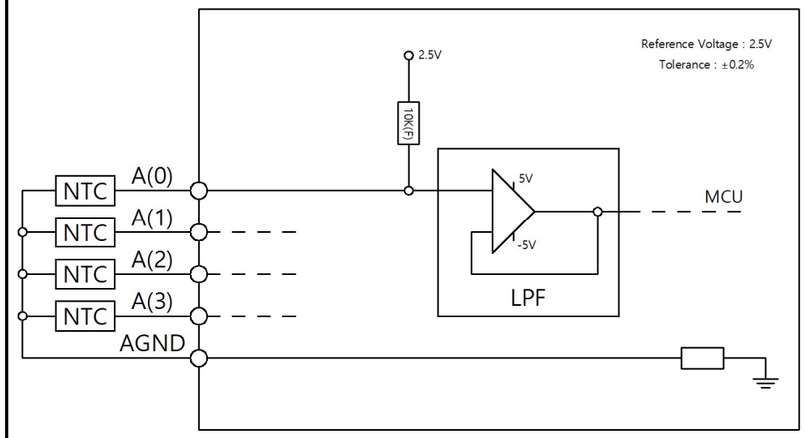
• 전압 또는 전류입력 선택시



• 관련 명령어
analogRead(Pin) Pin 포트의 아날로그 값을 디지털 수치로 변경합니다.
Return : 0 또는 1

• 온도센서 입력시

온도센서 입력 A(0) ~ A(3)에 연결된 써미스터(NTC 10KΩ(25℃) 온도센서의 저항 값을 프로그램에서 디지털 값(온도℃)으로 변환하여 사용합니다.



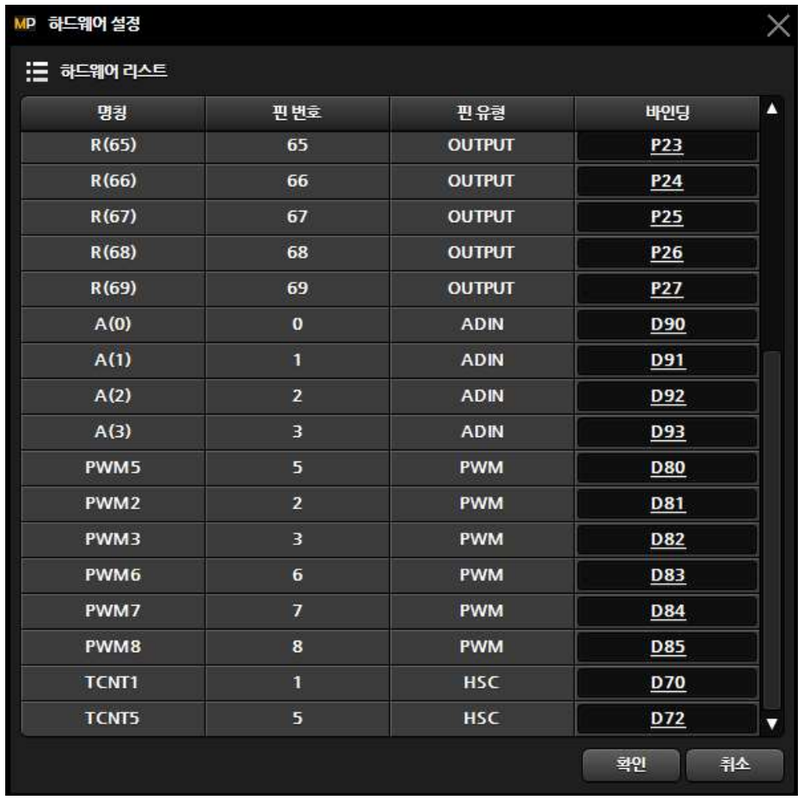
• 관련 명령어
NTEMP(ANALOG VALUE) 채널에 연결된 NTC 온도센서값을 측정합니다.
Return : unsigned int
ANALOG VALUE는 아두이노 C코드에서는 analogRead()를 사용하고, LADDER LOGIC에서는 하드웨어 설정에서 바인딩한 D메모리를 사용합니다.

❒ 아날로그 입력, 온도센서 입력 프로그램 간단 예제

• MPINO STUDIO : C코드를 사용할 경우

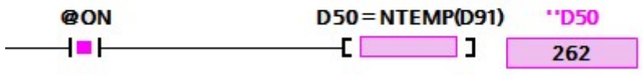
```
#include "LD.h" // Ladder 명령어를 사용하기 위한 함수참조

unsigned int ADC0 = 0; // ADC0 메모리를 정의
int TEMP = 0; // TEMP 메모리를 정의

void setup() {
  ladderSetup(); // Ladder를 실행하기 위한 설정함수
}

void loop() {
  ADC0 = analogRead(0); //A(0)포트의 전압을 디지털값으로 변환하여 AD0에 저장

  Temp = NTEMP(analogRead(1)); A(1)에 연결된 NTC-3950 온도센서의 값을 Temp 변수에 저장. (243 = 24.3℃를 의미합니다)
}
```

• MPINO STUDIO : LADDER LOGIC을 사용할 경우
아래와 같이 하드웨어설정에서 바인딩을 해주어야 합니다.



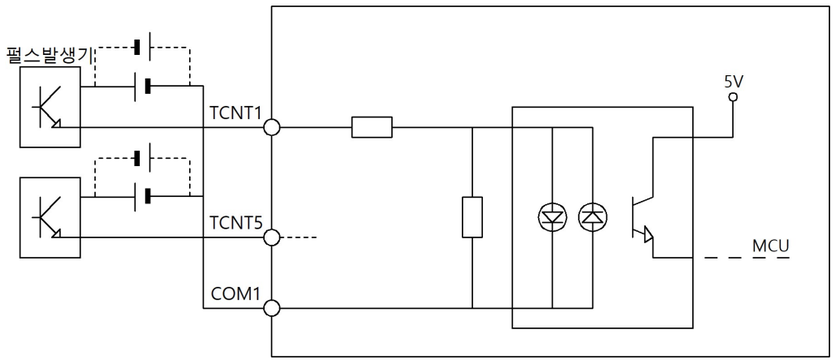
하드웨어 설정만으로 D 메모리에 아날로그 값이 자동으로 기록되고, D 메모리를 통해 아날로그 출력을 사용할 수 있습니다.

아날로그 입력 AI(0)에 입력된 아날로그 값이 512가 초과되면 트렌지스터 출력 P20을 On시키고, D80 메모리에 25000을 기록시킵니다.



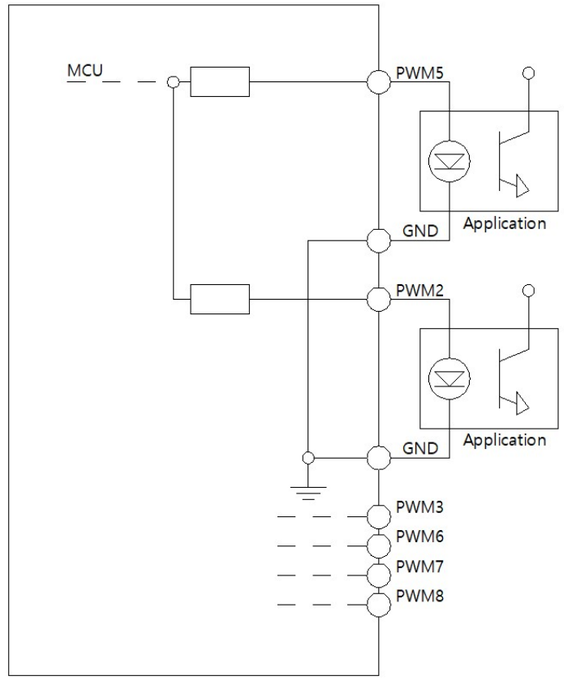
NTEMP 함수를 통해서 온도센서 입력포트 A(0)에 연결한 NTC-3950센서의 온도값을 D50에 저장합니다. 262은 26.2℃를 의미합니다.

❒ 고속펄스카운터
• 고속으로 들어오는 펄스의 개수를 하드웨어적으로 카운트합니다.



• 관련 명령어
hcntRead(Channel) : 고속펄스 카운터값을 읽어옵니다.
Return : unsigned long
Channel : TCNT1은 1, TCNT5는 5를 사용합니다.
hcntReset(Channel) : 고속펄스 카운터값을 0으로 초기화합니다.

❒ PWM 펄스출력



• 관련 명령어
analogWrite(Pin, Duty) : PWM(Pin)포트에 Duty길이의 펄스를 출력합니다.
Pin : PWM5는 5, PWM3은 3을 사용합니다.
Duty : 0 ~ 65535를 사용합니다.

❒ 고속펄스카운터 및 PWM 펄스출력 간단 예제

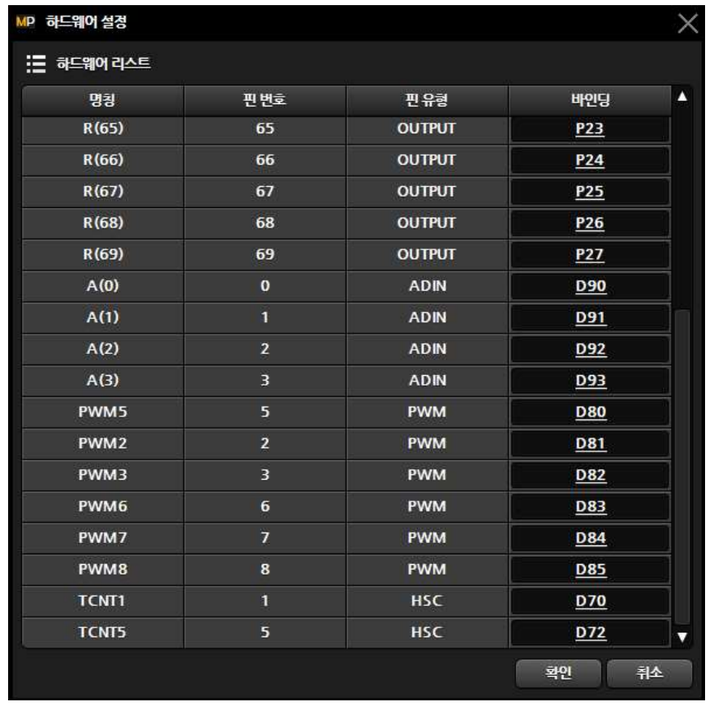
• MPINO STUDIO : C코드를 사용할 경우

```
#include "LD.h"

unsigned long HCNT0, HCNT1;

void setup(void) {
        ladderSetup();
}

void loop(void) {
        HCNT0 = hcntRead(1);
        HCNT1 = hcntRead(5);
        if (HCNT0 > 5000)
        {
            hcntReset(1);
            analogWrite(5, 32767);
        }
}
```

TCNT1과 TCNT5포트의 고속펄스카운터값을 각각 HCNT0과 HCNT1 변수에 저장합니다. 이후, HCNT0 변수값이 5000을 초과하면 TCNT1 고속펄스카운터값을 0으로 초기화 하고 PWM5포트에 Duty비가 50%인 PWM펄스를 발생합니다.

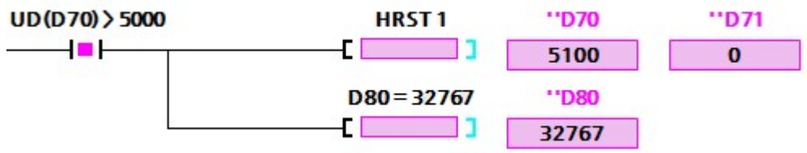
• MPINO STUDIO : LADDER LOGIC을 사용할 경우

아래와 같이 하드웨어설정에서 바인딩을 해주어야 합니다.



D70,D71에 TCNT1 고속카운터 값이 저장되고, D72,D73에 TCNT5 고속카운터 값이 저장됩니다.

하드웨어 설정에서 TCNTx는 Double Word영역으로 바인딩됩니다. 때문에, D영역이 겹치는지 주의하셔야 하며, LADDER LOGIC에서 UD()를 사용하여 아래처럼 사용해야 합니다.

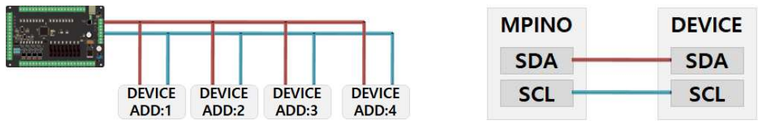


위 로직은 TCNT1:UD(70)가 5000 보다 크면 TCNT1의 카운터값을 0으로 초기화 하고 PWM5:D80에 32767를 저장하여 PWM5포트에 Duty비가 50%인 PWM 펄스를 발생하는 로직입니다.

⚠ PWM 펄스출력은 하드웨어 설정에서 PWM포트에 바인딩한 D영역을 사용하여 LADDER LOGIC에서 자동으로 analogWrite()함수를 사용하므로, 하드웨어 설정에서 PWM포트에 바인딩을 하였을 경우, 아두이노 C코드에서 사용할 수 없습니다.

❒ I2C 통신

I2C 통신은 2개의 선으로 구성되며, 아래와 같이 여러 장비와의 통신이 가능합니다.



전송 속도 : 100k bps(Default), 400k bps
전송 길이 : 최대 10m
결선 방법 : 마스터와 모든 슬레이브 디바이스를 다이렉트로 연결합니다.

• 참조

#include "Wire.h" 를 C코드 창 상단에 기입하여, Wire 함수들을 사용할 수 있습니다.

• LADDER LOGIC에서의 사용

LADDER LOGIC에서 I2C 명령을 지원하지 않습니다. LADDER LOGIC에서 C코드 함수를 호출하거나 C코드 명령을 사용해야 합니다.

• 관련 명령어

- 마스터 모드 (송신)

Wire.begin() : I2C를 마스터 모드로 시작합니다.
Wire.beginTransmission(Address) : 슬레이브 Address를 저장합니다.
Wire.write(Data) : Data를 저장합니다.
Wire.endTransmission(Stop) : 저장한 Address와 Data를 전송합니다. Stop이 True이면, 전송이 끝나고 I2C라인을 정지시키고 false이면 정지시키지 않습니다. 정지되지 않는다는 것은 ReStart 된다는 것입니다.
Wire.endTransmission() : 위 함수에서 Stop이 True인것과 동일한 함수입니다.

- 마스터 모드 (수신)

Wire.requestFrom(Address, Quantity, Stop) : Address 주소의 디바이스에서 Quantity 개수의 Byte를 요청합니다. STOP이 True이면, 마지막에 I2C라인을 정지시키고 False이면 정지시키지 않습니다.
Wire.requestFrom(Address, Quantity) : 위 함수에서 Stop이 True인 것과 동일한 함수입니다.
Wire.available() : 수신된 데이터의 바이트 수를 리턴합니다.
Wire.read() : 수신된 데이터의 1개의 바이트일 읽어 옵니다.

그 외의 함수들은 Arduino Wire Reference에서 확인할 수 있습니다.

• MPINO STUDIO : C코드에서만 지원합니다.

```
#include "LD.h"
#include "Wire.h"

void setup(void) {
	ladderSetup();
	Wire.begin();  // I2C를 마스터 모드로 시작합니다.
	Serial.begin( 9600 ); // 0번채널(다운로드포트)를 Open 합니다.
}

void loop(void) {
	ladderLoop();
	Wire.beginTransmission( 1 ); // 슬레이브 Address를 1로 저장합니다.
	Wire.write( 0x30 ); // 전송할 Data를 0x30으로 저장합니다.
	Wire.endTransmission( false ); // I2C시작, Address 전송, 0x30과 0x31전송,
                                       I2C재시작을 순서대로 실행합니다.

	Wire.requestFrom( 1, 10, true ); // 슬레이브 Address가 1인 디바이스에서
	   10 바이트를 읽어오고 I2C를 정지시킵니다.
	while ( Wire.available() ) // 읽은 바이트가 0이 될 때까지 루프를 형성
	{
		byte iRxData = Wire.read(); // 한 바이트를 읽어서iRxData에 저장
		Serial.println( iRxData ); // iRxData를 시리얼 모니터로 전송
	}
}
```

• LADDER LOGIC에서의 사용

LADDER LOGIC에서 Serial 통신명령을 지원하지 않습니다. LADDER LOGIC에서 C코드 함수를 호출하거나 C코드 명령을 사용해야 합니다.

❒ 시리얼 통신

시리얼 통신은 RS232, RS485, UART를 사용할 수 있습니다.

• RS232 : Serial1
1 : 1 통신이며, 최대 10m 거리의 통신을 할 수 있습니다.

• RS485 : Serial2
1 : N 통신이며, 최대 1km 거리의 통신을 할 수 있습니다.

• UART : Serial3
1 : 1 통신이며, 최대 1m 이하의 통신을 할 수 있습니다.

• 관련 명령어

✎ 함수의 Serial에는 각 채널명 Serial1,Serial2...로 치환하여 사용
- **Serial.begin()** : 시리얼 포트를 Open합니다.
- **Serial.write(byte)** : 1개의 Byte를 전송합니다.
- **Serial.write(array, length)** : Array에서 Length만큼 전송합니다
- **Serial.available()** : 수신된 Data(Byte)의 개수를 리턴합니다.
- **Serial.Read()** : 수신된 1개의 Byte를 읽어 옵니다.

✎ 더 많은 함수들을 Arduino Serial Reference에서 확인할 수 있습니다.

❒ Modbus RTU Slave

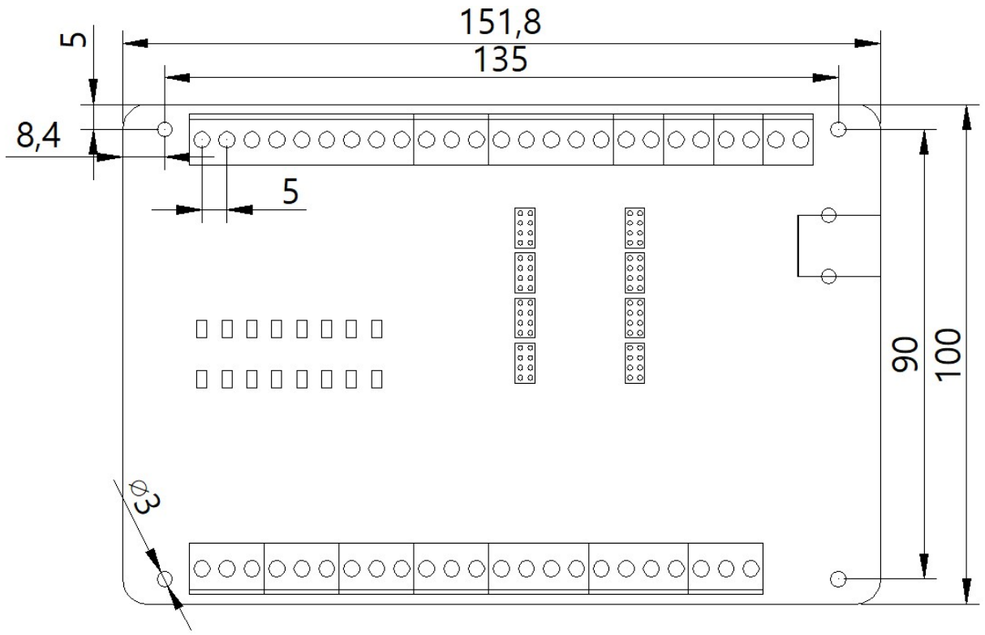
✎ 산업용에서 범용적으로 쓰이는 프로토콜입니다.
✎ 통신 영역은 LADDER LOGIC의 메모리를 사용합니다.
✎ MPINO STUDIO 사용설명서에서 자세한 사용방법을 확인하실 수 있습니다.

• 관련 명령어

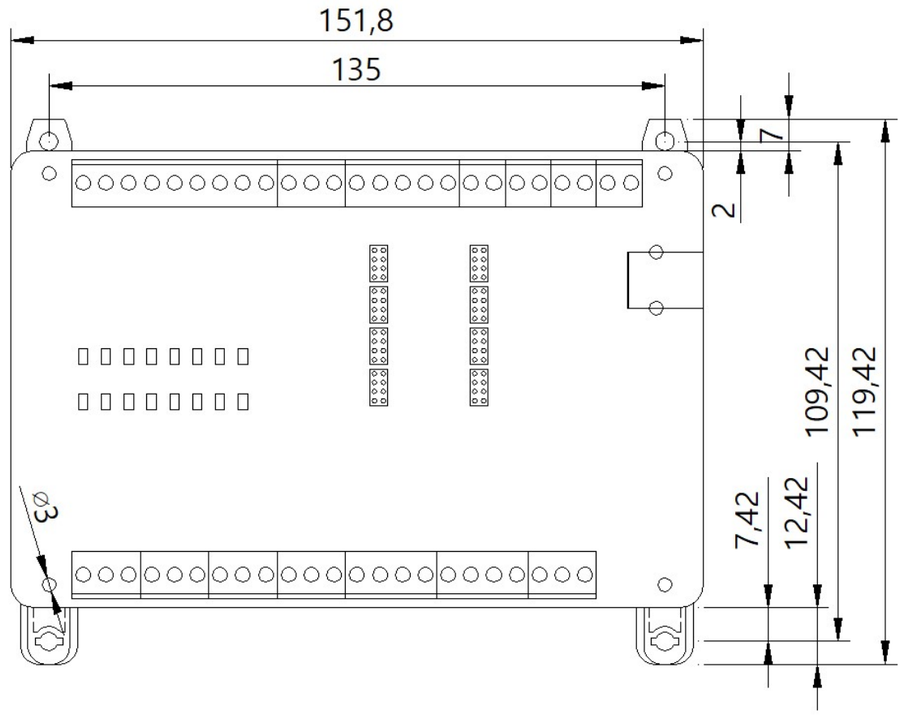
- **modbusStart(Serial, BaudRate, SlaveAddress)** : Serial 포트를 BaudRate와 SlaveAddress로 modbus RTU slave로 지정.
- **modbusStop()** : Modus RTU Slave를 모두 해제 합니다.

```
#include "LD.h"

void setup(void) {
	ladderSetup();
	modbusStart( Serial1, 9600, 1 ); // Serial1 채널을 9600 보레이트와 1 슬레이브 어드레스로 modbus RTU slave를 시작합니다.
}

void loop(void) {
	D0 = 1234; //D0레지스터리에 1234값을 저장
	// D0는 0x0000 시작어드레스
}
```

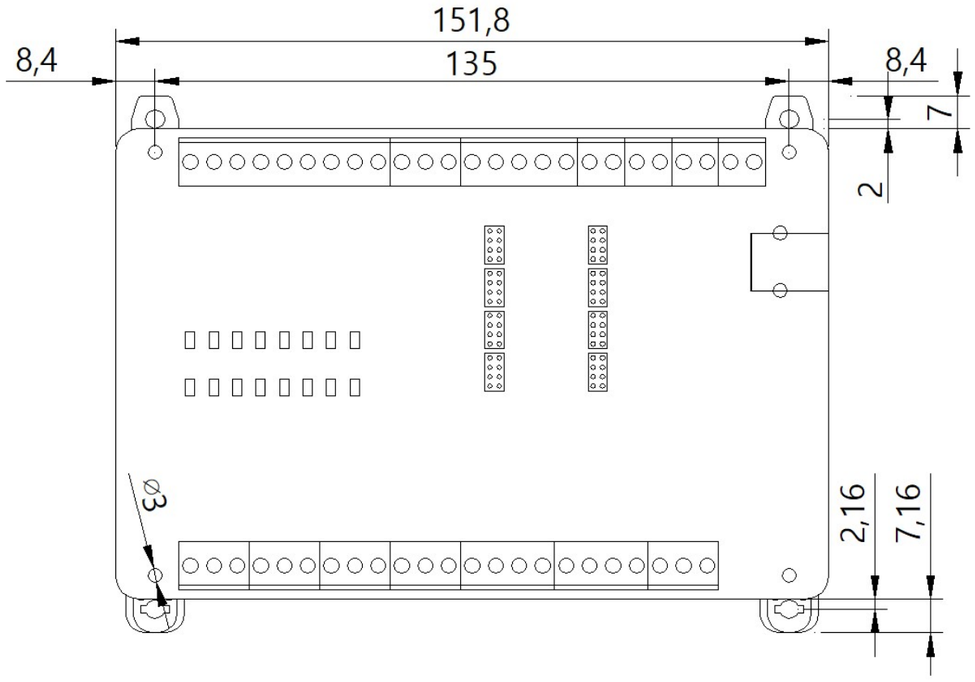
❒ DIMENSION (PCB / 딘레일 사용하지 않을 경우)



❒ DIMENSION (클립 열었을 때 / 딘레일 체결 전)



❒ DIMENSION (클립 닫았을 때 / 딘 레일 체결 후)



❒ DIMENSION (딘 레일 : 35mm)