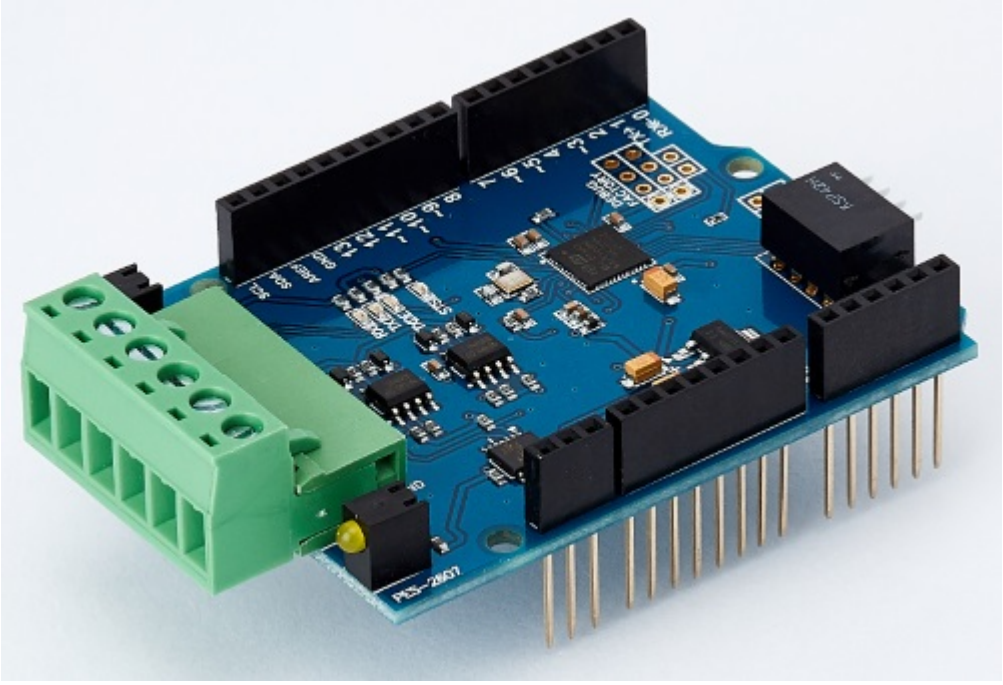
제품 소개



PES-2607

스마트 RS-422/485 보드 PES-2607은 아두이노용 PHPoC 쉴드 제품 전용 스마트 확장보드입니다. 이 보드를 이용하면 아두이노에 RS-422 또는 RS-485 통신기능을 쉽게 구현할 수 있습니다.

PES-2607의 주요 특징

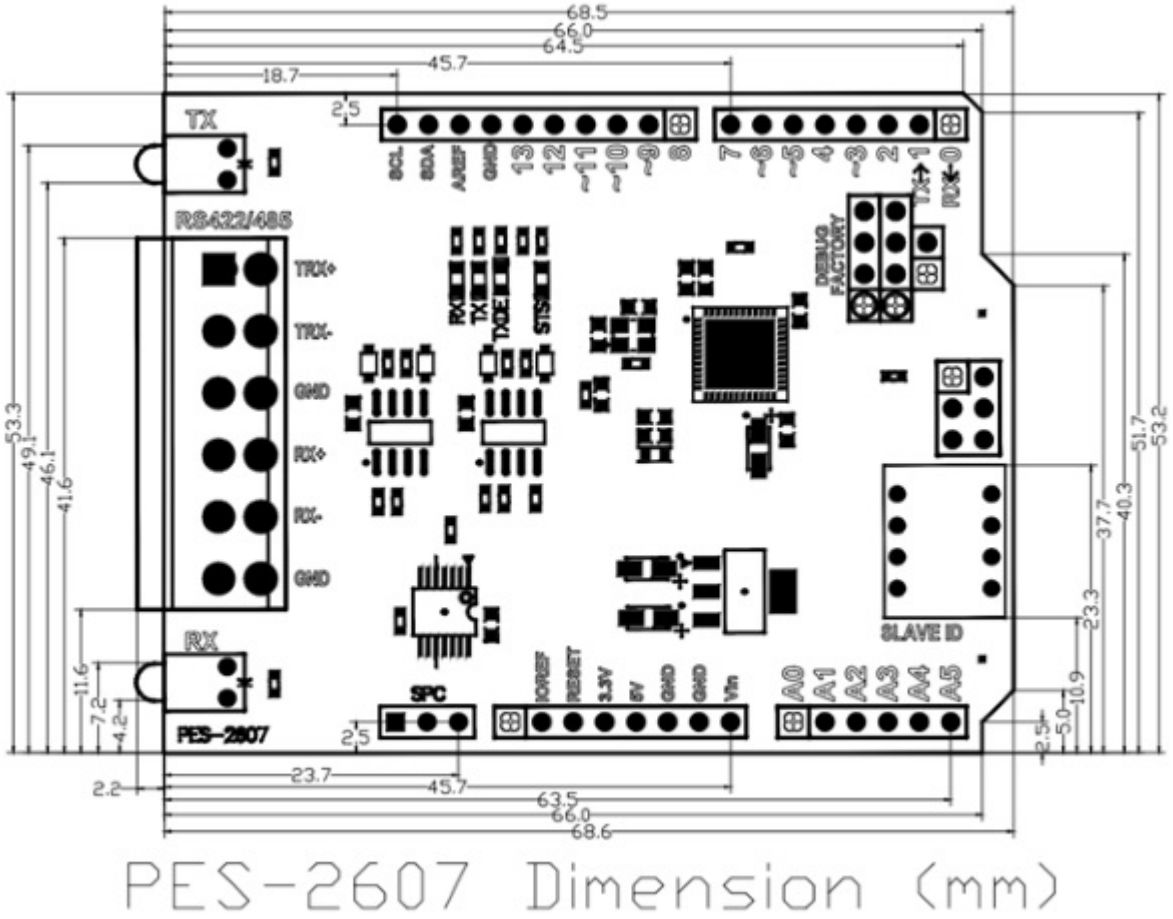
- 1 X RS-422(또는RS-485) 포트: 1200bps ~ 115200bps

※ 주의 : 이 확장보드를 사용하기 위해서는 반드시 아두이노와 R2 이상 버전의 PHPoC 쉴드가 필요합니다!

> PHPoC 쉴드용 스마트 확장보드란?
>
> PHPoC 쉴드용 스마트 확장보드는 자체 디바이스와 전용 펌웨어를 내장하고 있습니다. 이 보드는 PHPoC 쉴드와 전용 통신 포트를 이용해 마스터-슬레이브 방식으로 통신합니다. 하나의 PHPoC 쉴드에 여러 개의 스마트 확장보드를 연결할 수 있으며 각각의 스마트 확장보드에는 반드시 슬레이브 아이디를 설정해야 합니다.

치수

제품 본체



PES-2607 Dimension (mm)

※ 치수(단위 : mm)는 제품 상태 및 재는 각도 등에 따라 약간의 오차가 있을 수 있습니다.

터미널블록

이 보드는 6폴 터미널블록을 사용합니다. 치수는 각 터미널블록의 데이터시트를 참조하시기 바랍니다.

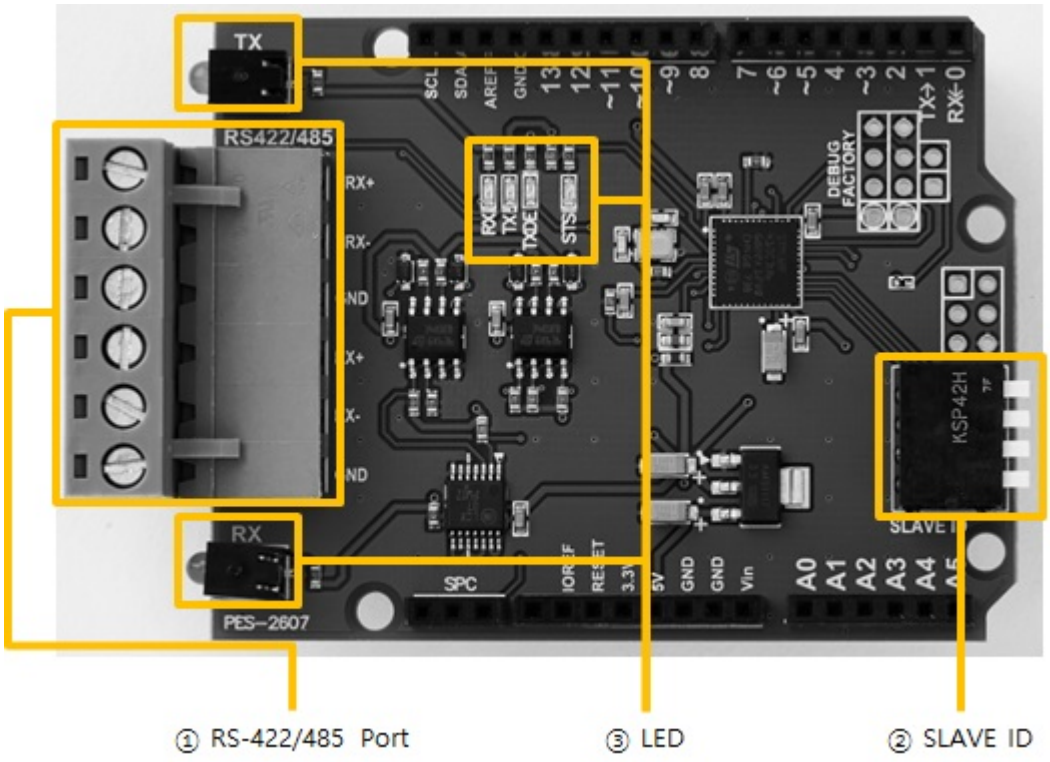
- T형 터미널블록 데이터시트
- S형 터미널블록 데이터시트

회로도

PES-2607의 회로도 입니다.

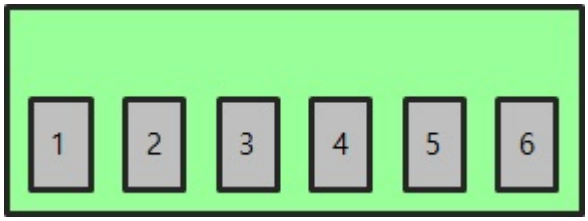
- PES-2607-V10-PO.pdf

레이아웃



1. RS-422/485 포트

이 보드의 RS-422/485 포트는 5mm간격의 1 by 6 터미널 블록으로 되어 있습니다.



RS-422

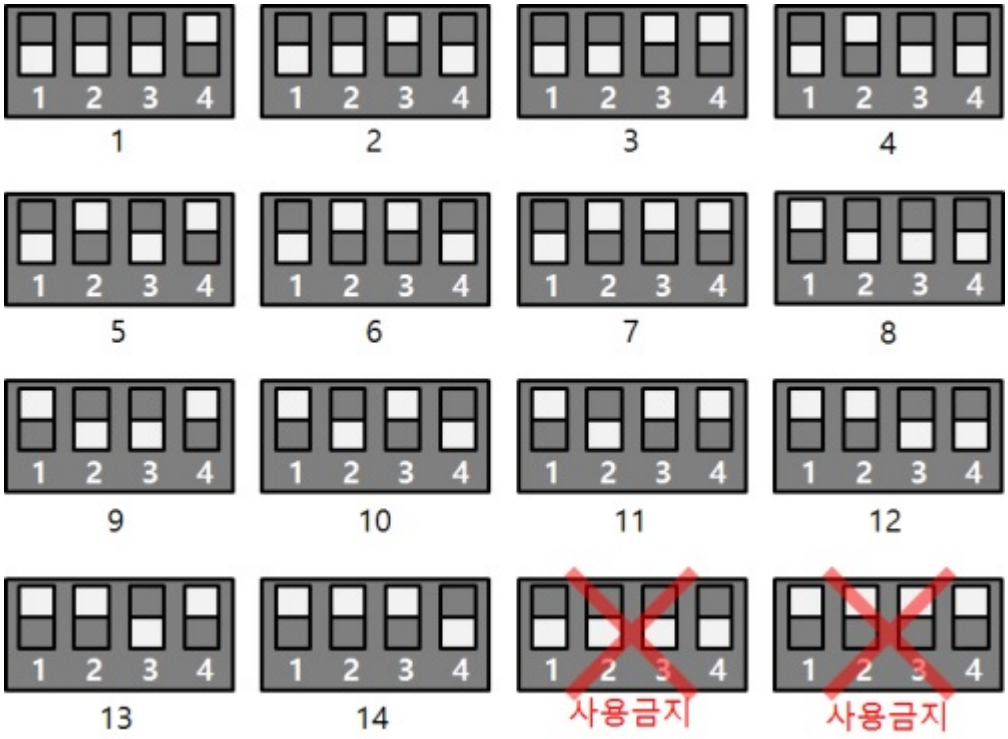
번호	이름	설명	신호레벨	방향	외부결선
1	TX+	Transmit Data+	RS-422	출력	필수
2	TX-	Transmit Data-	RS-422	출력	필수
3	GND	Ground	Ground	-	필수
4	RX+	Receive Data+	RS-422	입력	필수
5	RX-	Receive Data-	RS-422	입력	필수
6	GND	Ground	Ground	-	필수

RS-485

번호	이름	설명	신호레벨	방향	외부결선
1	TRX+	Transmit / Receive Data+	RS-485	입/출력	필수
2	TRX-	Transmit / Receive Data-	RS-485	입/출력	필수
3	GND	Ground	Ground	-	필수
6	GND	Ground	Ground	-	필수

2. 슬레이브 아이디 스위치

슬레이브 아이디는 PHPoC 쉴드가 스마트 확장보드 각각을 구분하는데 사용됩니다. 따라서 PHPoC 쉴드에 연결되는 각 스마트 확장보드는 고유한 슬레이브 아이디를 사용해야 합니다. 슬레이브 아이디는 1부터 14까지 14개 중 하나로 설정할 수 있으며 다음과 같이 4개의 딥스위치를 조정하여 설정합니다.



3. LED

이 보드에는 총 6개의 LED가 있으며 동작은 다음과 같습니다.

구분	개수	종류	색	동작 설명
STS	1	SMD	빨간색	아이디 설정이 정상일 때 > 1초마다 켜짐/꺼짐 반복 아이디 설정이 올바르지 않을 때 > 빠르게 깜박임
TX	2	DIP, SMD	초록색	시리얼포트로 데이터 송신시 깜박임
RX	2	DIP, SMD	노란색	시리얼포트로 데이터 수신시 깜박임
TXDE	1	SMD	초록색	TxDE제어를 사용할 때 > 데이터 송신시 깜박임 TxDE제어를 사용하지 않을 때 > 항상 켜짐

사용하기

이 보드를 사용하는 방법은 다음과 같습니다.

1. PHPoC 쉴드와 아두이노에 연결

이 보드는 단독으로 사용할 수 없습니다. 반드시 아두이노와 아두이노용 PHPoC 쉴드에 연결하여 사용하시기 바랍니다.

2. 아두이노용 라이브러리 설치

아두이노 IDE의 라이브러리 매니저를 통해 Phpoc 라이브러리와 PhpocExpansion 라이브러리를 설치합니다. 아두이노용 PHPoC 쉴드와 스마트 확장보드를 사용하려면 반드시 두 라이브러리를 모두 설치해야 합니다. 라이브러리에 대한 자세한 내용은 다음 문서를 참조하시기 바랍니다.

- PHPoC 쉴드 라이브러리 레퍼런스

3. 예제코드 활용

본 매뉴얼과 라이브러리에 포함된 예제코드를 활용하여 프로그래밍 하시기 바랍니다.

클래스 및 함수

클래스

이 확장보드를 사용하기 위해서는 아두이노 PHPoC 라이브러리의 ExpansionSerial 클래스를 사용합니다.

멤버 함수

ExpansionSerial 클래스의 사용 가능한 멤버함수는 다음과 같습니다.

멤버 함수	설명
int getPID(void)	제품 아이디 읽기
char *getName(void)	제품명 읽기
ExpansionSerial(int sid)	시리얼 포트의 인스턴스 생성
void begin(void)	시리얼 통신 파라미터 설정
int available(void)	수신 데이터 크기 확인
int peek(void)	수신 데이터 1바이트 확인
int read(void)	수신 데이터 1바이트 읽기
int availableForWrite(void)	수신버퍼 여유공간 확인
void flush(void)	데이터 송신 완료 대기
int write(int wbuf, int wlen)	데이터 송신

설정하기

통신 파라미터 설정하기

begin() 함수로 시리얼 통신 파라미터를 설정하십시오.

```
port.begin(baud)
port.begin(sets)
```

- baud - 통신 속도를 나타내는 bps단위의 정수(1200 ~ 115200)
- sets - 통신 속도, 패리티, 데이터 비트, 정지 비트 및 흐름제어를 나타내는 문자열

```
"(baudrate)[parity[data bit[stop bit[flow control]]]]"
```

※ (): 사용 필수, []: 생략 가능

파라미터	설정 범위	설명	기본 값
baudrate	1200 ~ 115200	통신 속도(bps)	115200
parity	N, E, O, M 또는 S	패리티 비트 (N: 없음, E: 짝수, O: 홀수, M: Mark, S: Space)	N
data bit	8 또는 7	데이터 비트	8
stop bit	1 또는 2	정지 비트	1
flow control	T 또는 N	TxDE제어 사용(T), TxDE제어 사용 안 함(N)	T

※ 주의 : flow control 설정은 항상 기본 값인 T로 사용하는 것을 권장합니다.

예제

- 아두이노 소스코드

```
#include <PhpocExpansion.h>
#include <Phpoc.h>
#define BUFFER_SIZE 100  // read and write buffer size, reduce it if memory of Arduino is not
enough

byte spcId = 1;

ExpansionSerial port(spcId);

byte rwbuf[BUFFER_SIZE];  // read and write buffer

void setup() {
    Serial.begin(9600);
    while(!Serial)
        ;
```

```
    Phpoc.begin(PF_LOG_SPI | PF_LOG_NET);
    Expansion.begin();

    // sets the parameters for serial data communication
    port.begin("115200N81T");
}

void loop() {
    int txfree = port.availableForWrite();
    int rxlen = port.available();

    if(rxlen > 0) {
        if(rxlen <= txfree) {
            int rwlen; // read and write length

            if(rxlen <= BUFFER_SIZE)
                rwlen = rxlen;
            else
                rwlen = BUFFER_SIZE;

            // receive data
            rwlen = port.readBytes(rwbuf, rwlen);

            // send data
            port.write(rwbuf, rwlen);

            // print data to serial monitor of Arduino IDE
            Serial.write(rwbuf, rwlen);
        }
    }

    delay(1);
}
```

데이터 수신하기

수신 데이터 크기 읽기

available() 함수를 이용하여 시리얼 포트로 수신된 데이터 크기를 읽을 수 있습니다.

```
port.available()
```

이 함수는 호출 시점에 시리얼포트에서 읽을 수 있는 데이터의 크기(바이트 수)를 정수형태로 반환합니다.

수신 데이터 1바이트 확인

peek() 함수를 이용하여 시리얼 포트로 수신된 데이터 중 첫 번째 바이트를 확인할 수 있습니다.

```
port.peek()
```

이 함수는 버퍼의 데이터를 읽지는 않고 확인만 합니다. 따라서 이 함수를 호출해도 해당 데이터는 버퍼에 그대로 남아있습니다.

수신 데이터 1바이트 읽기

read() 함수를 이용하여 시리얼 포트로 수신된 데이터 중 첫 번째 바이트를 읽을 수 있습니다.

```
port.read()
```

예제

- 아두이노 소스코드

```
#include <PhpocExpansion.h>
#include <Phpoc.h>
#define BUFFER_SIZE 100  // read and write buffer size, reduce it if memory of Arduino is not enough

byte spcId = 1;

ExpansionSerial port(spcId);

byte rwbuf[BUFFER_SIZE];  // read and write buffer

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

```
    while(!Serial)
        ;

    Phpoc.begin(PF_LOG_SPI | PF_LOG_NET);
    Expansion.begin();
    port.begin("115200N81T");
}

void loop() {
    int txfree = port.availableForWrite();

    // gets the size of received data
    int rxlen = port.available();

    if(rxlen > 0) {

        // reads the next byte of incoming serial data
        int value = port.read();
        Serial.print("read : ");
        Serial.println(value);

    }
    delay(1);
}
```

데이터 송신하기

송신버퍼 여유공간 확인

availableForWrite() 함수를 이용하여 송신버퍼의 여유공간을 확인할 수 있습니다.

```
port.availableForWrite()
```

이 함수는 송신버퍼의 여유공간(바이트)을 정수형태로 반환합니다.

데이터 송신 완료 대기

flush() 함수를 이용하여 송신버퍼의 데이터가 모두 전송될 때까지(송신 버퍼가 비워질 때까지) 대기할 수 있습니다.

```
port.flush()
```

데이터 송신하기

write() 함수를 이용하여 데이터를 송신할 수 있습니다.

```
port.write(byte)
port.write(wbuf, wlen)
```

- byte - 정수형 1바이트 데이터
- wbuf - 연속된 바이트
- wlen - 보낼 데이터 크기(바이트)

예제

- 아두이노 소스코드

```
#include <PhpocExpansion.h>
#include <Phpoc.h>
#define BUFFER_SIZE 100  // read and write buffer size, reduce it if memory of Arduino is not
enough

byte spcId = 1;

ExpansionSerial port(spcId);

byte rwbuf[BUFFER_SIZE];  // read and write buffer
```

```
void setup() {
    Serial.begin(9600);
    while(!Serial)
        ;

    Phpoc.begin(PF_LOG_SPI | PF_LOG_NET);
    Expansion.begin();
    port.begin("115200N81T");
}

void loop() {

    int txfree = port.availableForWrite();
    int rxlen = port.available();

    if(rxlen > 0) {
        if(rxlen <= txfree) {
            int rwlen; // read and write length

            if(rxlen <= BUFFER_SIZE)
                rwlen = rxlen;
            else
                rwlen = BUFFER_SIZE;

            // receive data
            rwlen = port.readBytes(rwbuf, rwlen);

            // send data
            port.write(rwbuf, rwlen);

            // print data to serial monitor of Arduino IDE
            Serial.write(rwbuf, rwlen);
        }
    }

    delay(1);
}
```