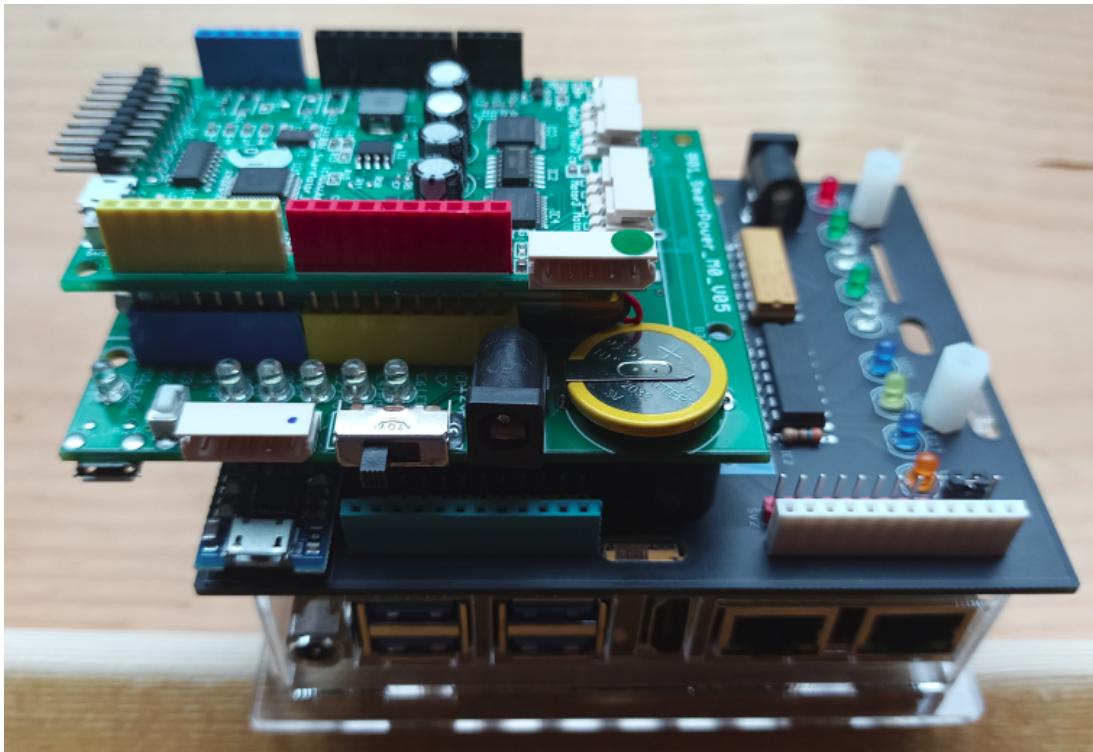


RISC-V 64 비트 StarFive JH7110 40 핀맵 호환

# RISC-V 입출력 확장 보드 매뉴얼

RISC-V  
입출력 확장



커널연구회 ([www.kernel.bz](http://www.kernel.bz))  
정재준 ([rabi3307@nate.com](mailto:rabi3307@nate.com))

# 목차

## Table of Contents

● RISC-V 입출력 확장 보드 매뉴얼.....	1
● 목차.....	2
● RISC-V 실습용 입출력 보드.....	4
기능 요약.....	4
입출력 연결 구성도.....	5
40 핀맵 구성도.....	6
실습용 소스 제공(GitHub).....	7
● 입출력 확장 기능.....	9
입출력 적층 연결핀 구성도.....	9
배터리 Power 보드 연결 구성도.....	10
Motor 제어 보드 연결 구성도.....	11
리눅스 커널에서 모터 드라이버 구동 방법.....	13
DC 모터 제어 명령.....	15
서보 모터 제어 명령.....	16
스테핑 모터 제어 명령.....	17
● 확장 보드 조합 구성.....	19
모터 제어 보드.....	19

목차

---

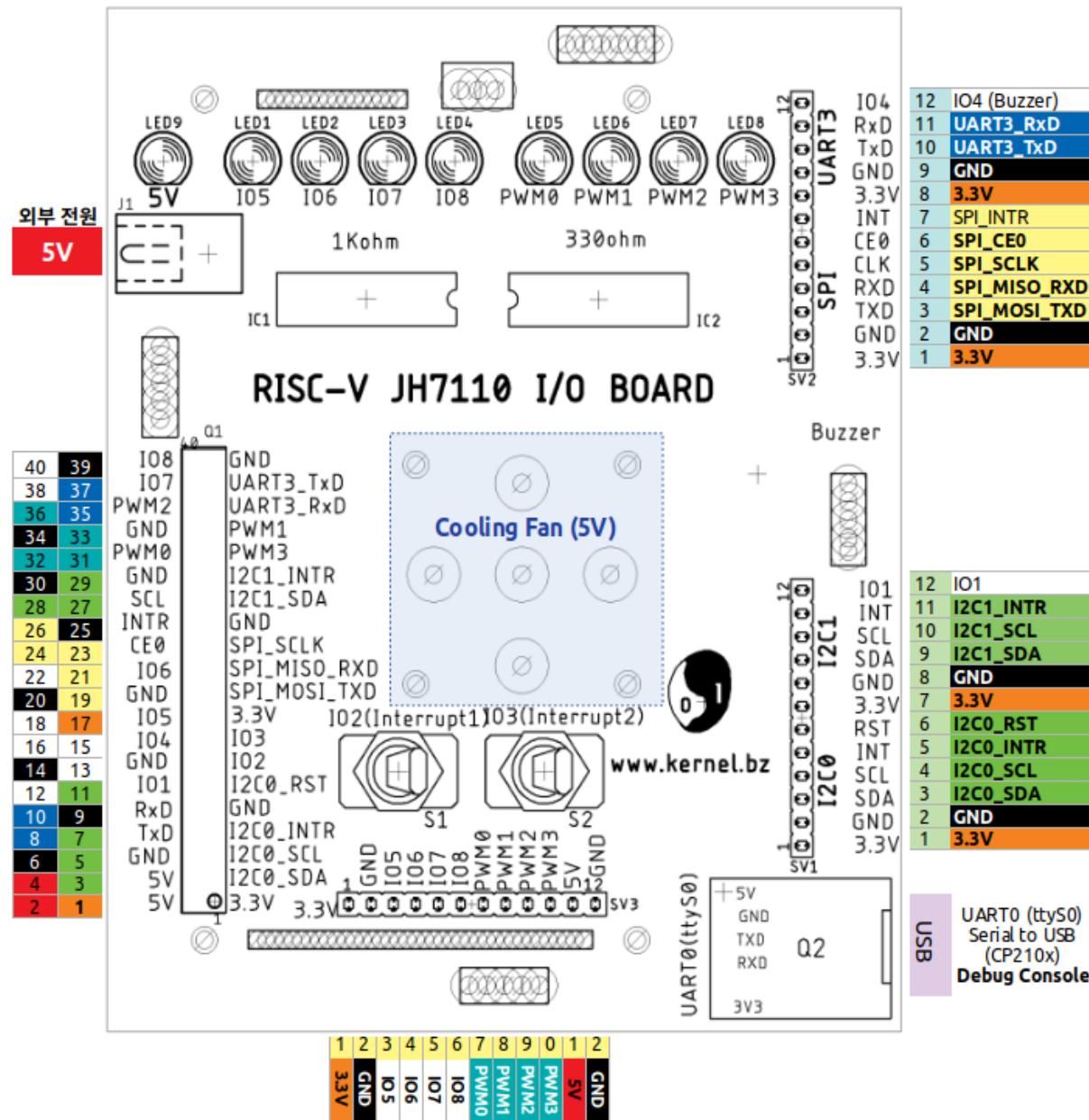
모터 제어 + RISCV 입출력 보드.....	19
배터리 파워 보드.....	20
배터리 파워 + RISCV 입출력 보드.....	20
배터리 파워 + 모터 제어 + RISCV 입출력 보드.....	21

## RISC-V 실습용 입출력 보드

### 기능 요약

- RISC-V 64 비트 StarFive JH7110 40 핀맵 호환
- Serial(UART0, ttyS0) to USB 포트 탑재 (디버그 콘솔)
- Serial(UART3, ttyS3) 포트 지원
- I2C0, I2C1 포트 지원 (센서 연결)
- SPI 포트 지원 (센서 연결)
- PWM0, PWM1, PWM2, PWM3 포트 출력 (모터 연결)
- GPIO 포트 및 LED 출력 지원
- Switch1, Switch2 외부 인터럽트 처리 지원
- 5V 외부 전원 입력 RCA 잭
- 쿨링팬 연결 지원
- 40 핀 입출력핀 확장 (적층) 지원 (기능 확장)
- 실습 소스 제공(<https://github.com/kernel-bz/riscv-jh7110-io.git>)

## 입출력 연결 구성도



## 40 핀맵 구성도

40 핀맵은 RISC-V 64 비트 StarFive JH7110 보드와 호환되고 세로 높이 방향으로 적층하여 확장해 갈 수 있습니다. (라즈베리파이 40 핀맵과도 호환됨)

device	function	GPIO	pin number	GPIO	function	device
LED4	IO8	44	40	39	GND	
LED3	IO7	61	38	37	60	UART3 (TxD)
LED8	PWM2	36	36	35	63	UART3 (RxD)
	GND		34	33	59	PWM1
LED5	PWM0	46	32	31	39	PWM3
	GND		30	29	37	I2C1 (INTR)
I2C1	I2C1 (SCL)	40	28	27	45	I2C1 (SDA)
SPI	SPI (INTR)	56	26	25	GND	
	SPI (CE0)	49	24	23	48	SPI (SCLK)
LED2	IO6	50	22	21	53	SPI (MISO) RxD
	GND		20	19	52	SPI (MOSI) TxD
LED1	IO5	51	18	17	3.3V	
Buzzer	IO4	54	16	15	47	IO3
	GND		14	13	43	IO2
	IO1	38	12	11	42	I2C0 (REST)
ttyS0	UART0 (RxD)	6	10	9	GND	
	UART0 (TxD)	5	8	7	55	I2C0 (INTR)
	GND		6	5	57	I2C0 (SCL)
	5V		4	3	58	I2C0 (SDA)
	5V		2	1	3.3V	

## 실습용 소스 제공(GitHub)

커널연구회에서 제공하는 RISC-V 입출력 보드 실습용 소스는 아래에서 다운로드 할 수 있습니다.

<https://github.com/kernel-bz/riscv-jh7110-io>

### 소스 경로 구조:

```
$ git clone https://github.com/kernel-bz/riscv-jh7110-io.git
$ cd riscv-jh7110-io
$ tree

|-- README.md
|-- app
|   |-- gpio (GPIO 실습 소스)
|   |   |-- gpio-test.c
|   |   `-- gpio.c
|   |-- hello
|   |   |-- hello
|   |   `-- hello.c
|   |-- i2c (I2C 실습 소스)
|   |   |-- i2c
|   |   `-- i2c.c
|   |-- pwm (PWM 실습 소스)
|   |   |-- pwm
|   |   |-- pwm-test
|   |   `-- pwm-test.c
|   `-- uart (UART 실습 소스)
|       |-- build.sh
|       |-- recv
|       |-- recv.c
|       |-- send
|       |-- send.c
|       |-- serial.c
|       `-- serial.h
|           |-- gpio
```

## RISC-V 실습용 입출력 보드

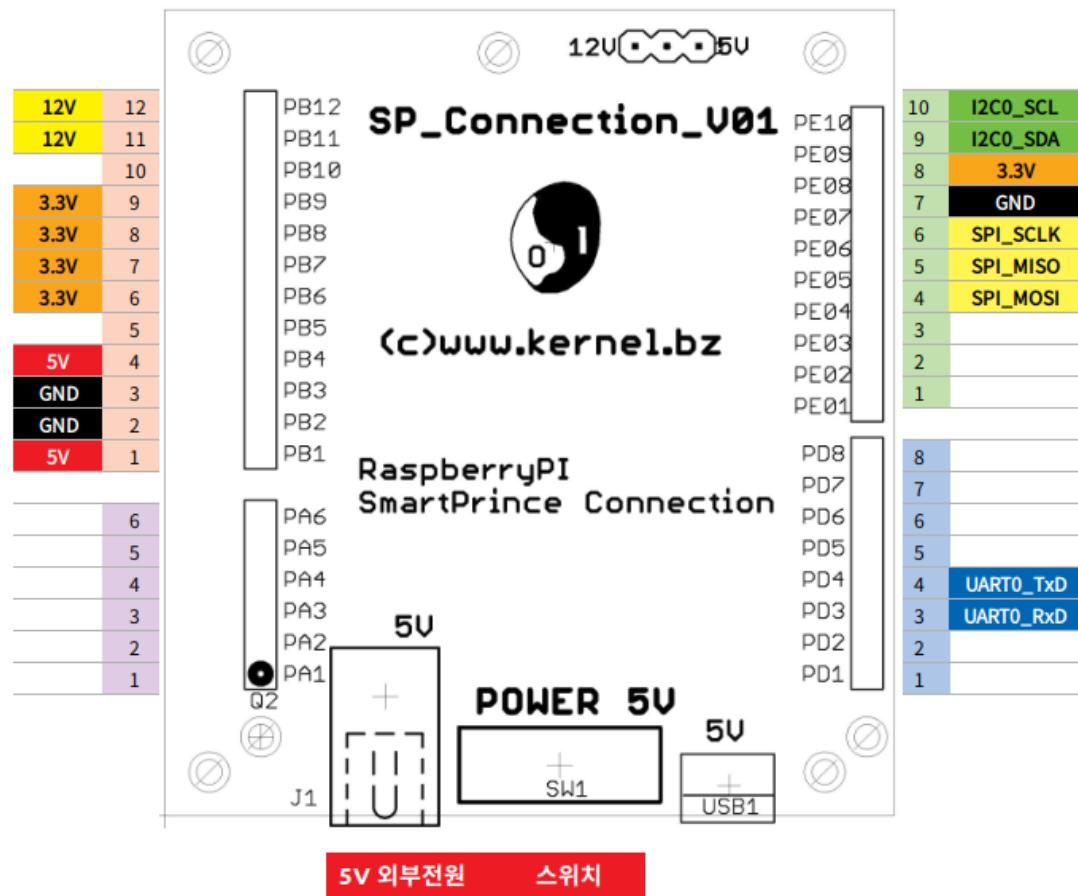
```
|   |   |-- gpio-test  
|-- dtb  
|   `-- overlay (DTB 추가 소스)  
|       |-- run.sh  
|       |-- vf2-overlay-i2c1.dtbo  
|       `-- vf2-overlay-uart3.dtbo  
`-- modules (드라이버 모듈)  
    |-- user-gpio-driver.ko  
    |-- user-i2c-smartmotor.ko  
    `-- user-i2c-smartpower.ko
```

소스 사용 방법들은 아래부터 설명 됩니다.

## 입출력 확장 기능

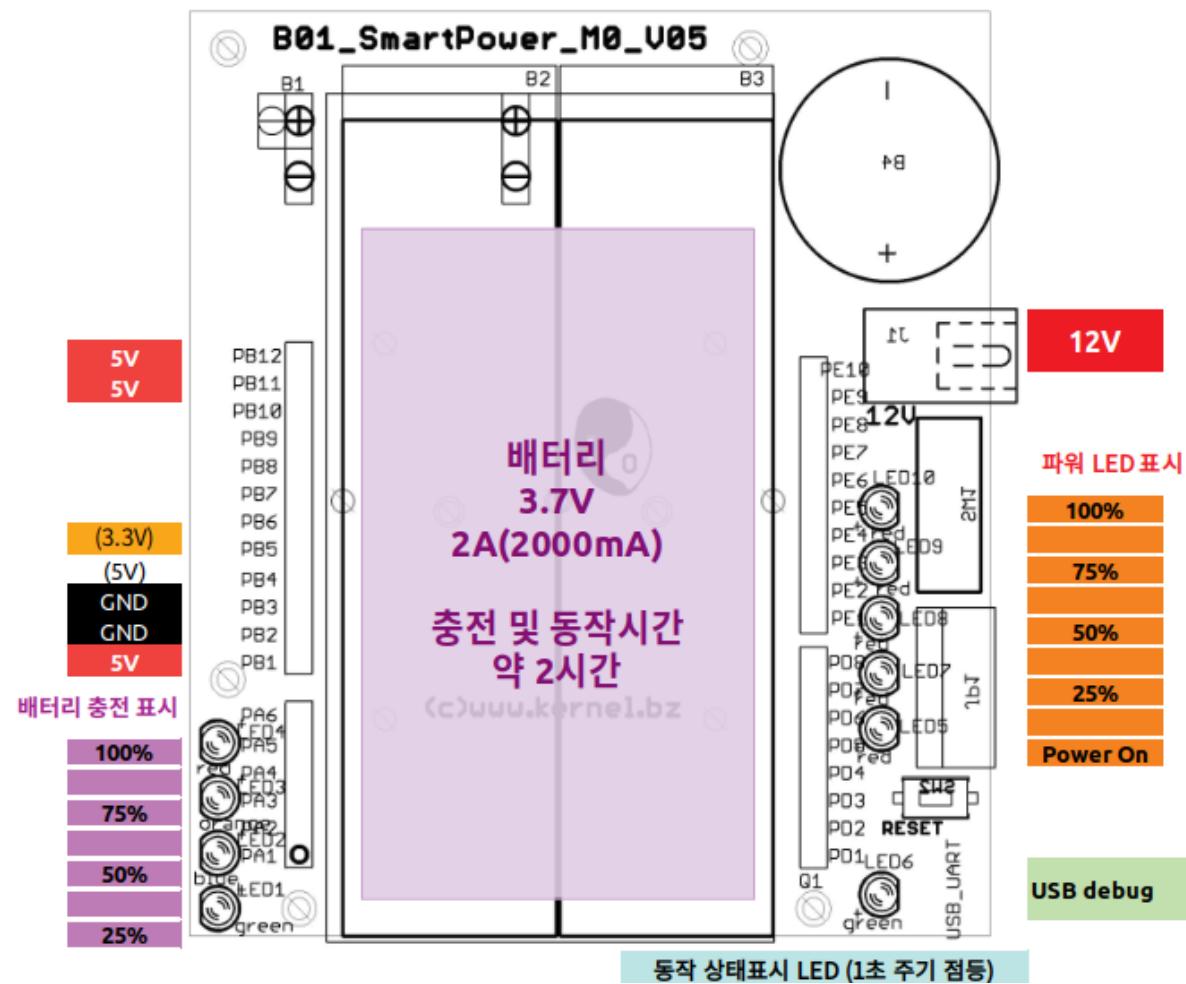
입출력 확장 기능은 RISC-V 입출력 보드의 40 핀맵에 세로 방향으로 적층하여 확장할 수 있도록 구성했습니다. 아래 내용들은 추가적인 보드 구매가 필요하며 커널연구회에서 진행하는 교육을 통하여 관련 소스와 함께 기능들을 학습할 수 있습니다.

## 입출력 적층 연결핀 구성도



## 배터리 Power 보드 연결 구성도

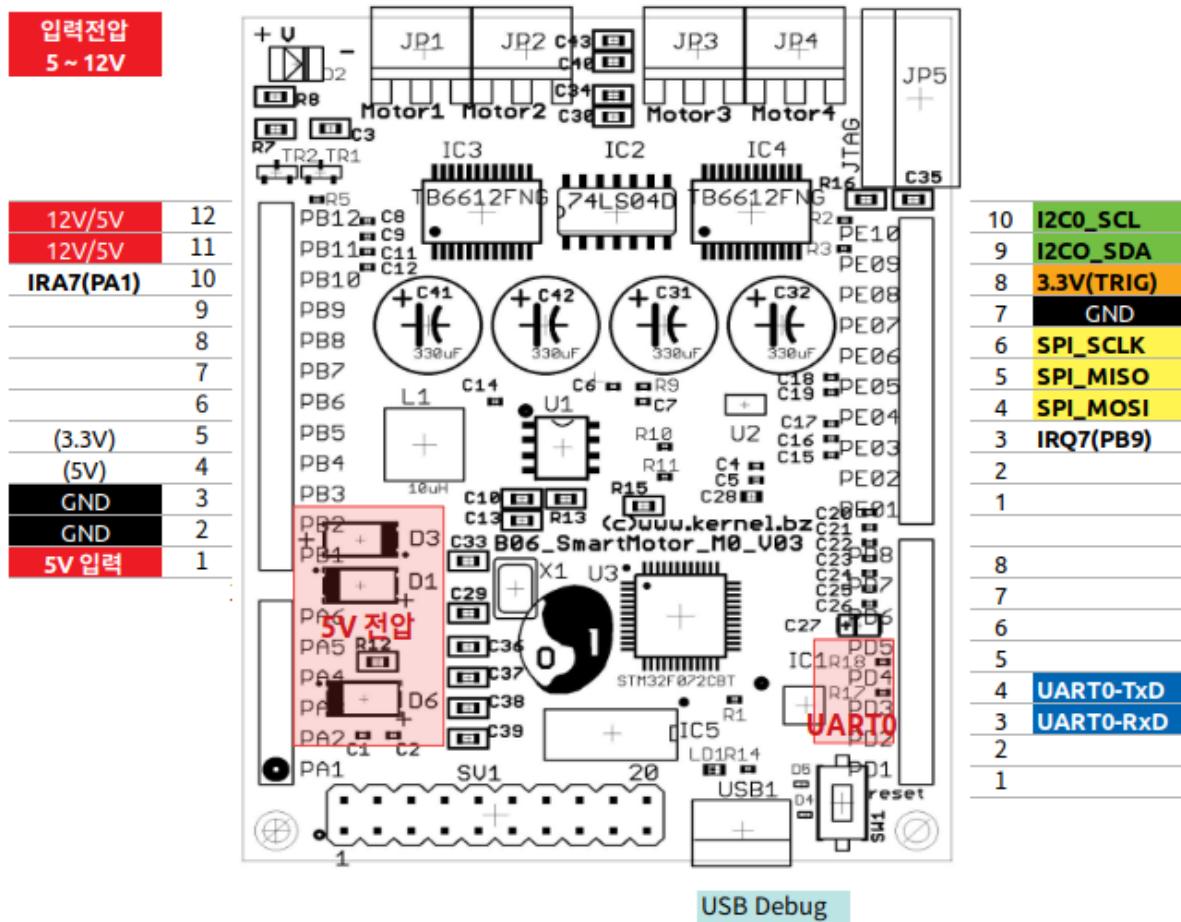
- Cortex-M0(STM32F0) MCU 내장
- 12V 외부 어댑터 전원공급(전류는 2A 이상)
- 배터리(3.7V 리튬폴리머) 충전기능
- 배터리 용량 2000mA
- 배터리 충전 상태 및 전류사용량 모니터링
- 보드 표면 온도 모니터링
- 배터리 충전 및 동작시간: 약 2 ~ 3 시간
- 리눅스 커널 모터제어 드라이버 모듈 제공



## Motor 제어 보드 연결 구성도

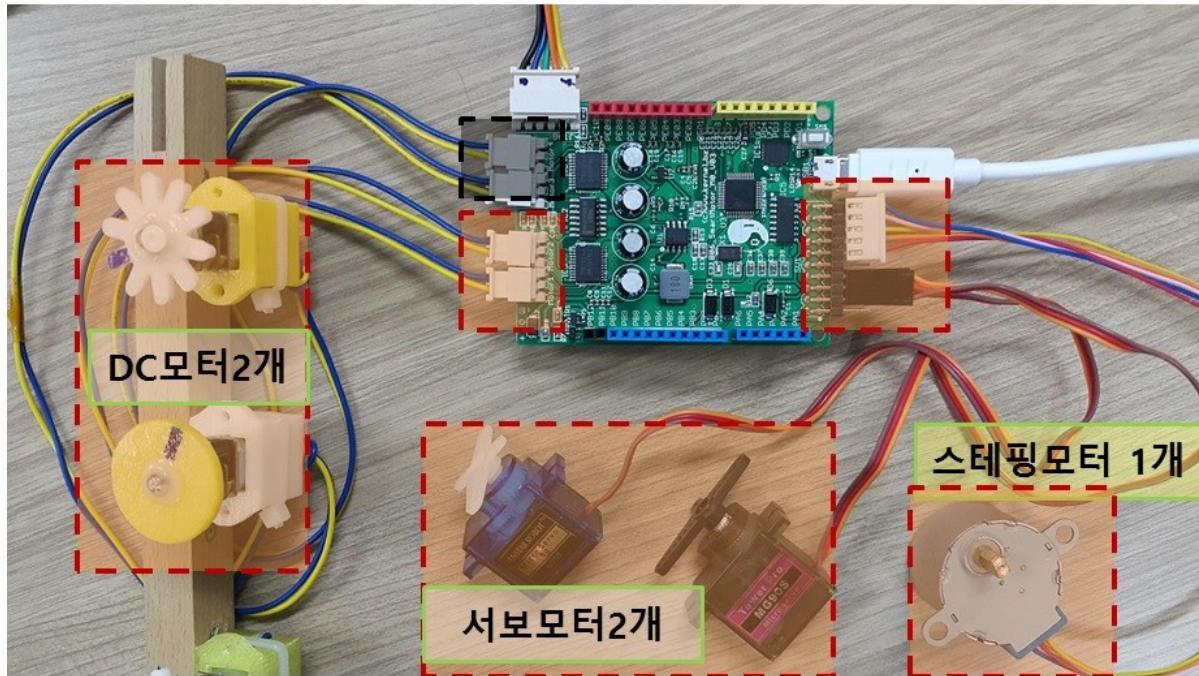
- Cortex-M0(STM32F0) MCU 내장
- 5V DC 모터 4 개 제어(4륜구동)
- PWM 신호로 모터 속도 및 방향 제어
- 5V Servo 모터 2 개 제어
- 5V Stepping 모터 1 개 제어
- 외부 확장 헤더핀(20 핀) 제공하여 ADC 센서 4 개, 외부 GPIO 6 개 연결가능

### Motor 보드 핀맵

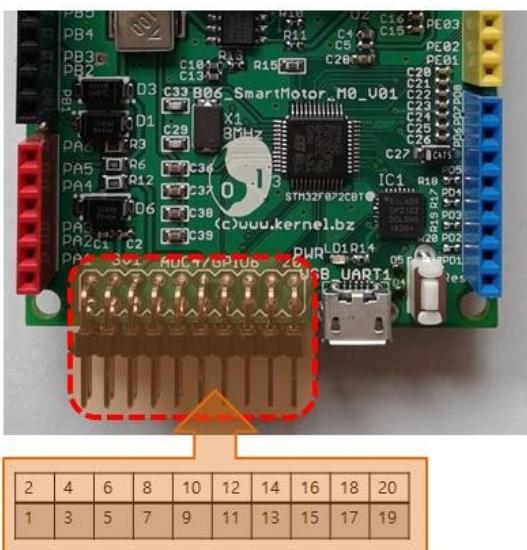


## 입출력 확장 기능

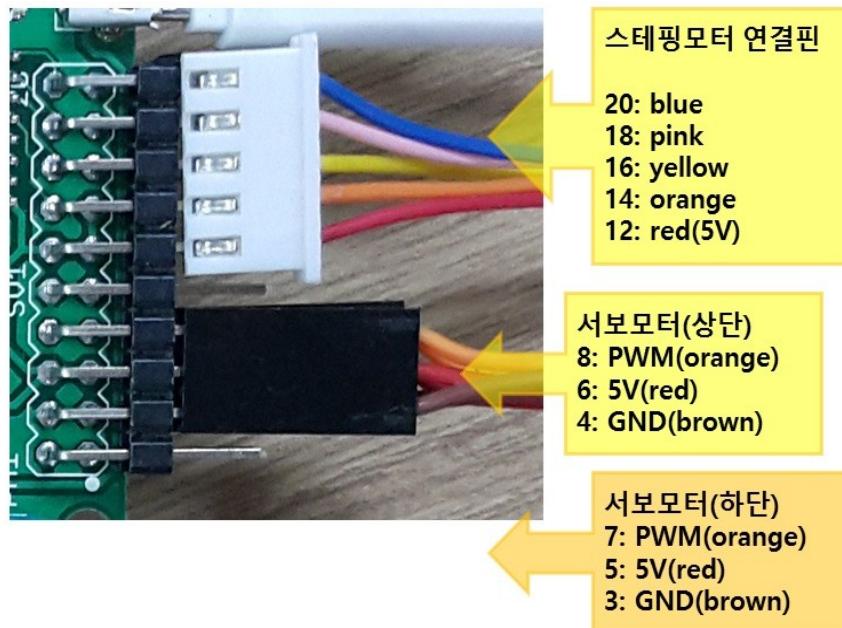
## 모터 연결 구성도



## 확장 핀맵 번호



## 확장 핀에 모터 연결



## 리눅스 커널에서 모터 드라이버 구동 방법

위의 모터 보드에 연결한 DC 모터 4 개와 서보 모터 2 개, 스텝 모터 1 개를 리눅스 커널 디바이스 드라이버 모듈에서 제어할 수 있습니다. 먼저 다음과 같이 커널연구회에서 제공한 디바이스 드라이버 모듈을 다음과 같이 `insmod` 명령어로 커널에 삽입 합니다.

### 모터 드라이버 모듈 삽입(`root` 권한에서 실행)

```
//커널연구회에서 제공한 실습 경로로 이동
# cd riscv-jh7110-io/modules/
```

```
//user-i2c-smartmotor.ko을 커널에 삽입
# insmod user-i2c-smartmotor.ko

[ 416.951882] user_i2c_probe:: STARTING...
[ 416.955906] user_i2c_probe:: user_i2c_kobj_init(): ret:<0>
[ 416.961524] user_i2c_probe:: user_i2c_gpio_init(): ret:<0>
```

user-i2c-smartmotor.ko 커널 드라이버 모듈을 삽입하면 다음과 같이 /sys/kernel/ 에 user-motor-works 경로가 생성되어 DC 모터와 서보, 스텝 모터를 제어할 수 있는 파일들이 생성 됩니다.

### /sys/kernel/user-motor-works 경로 확인

```
# ll /sys/kernel/user-motor-works/
drwxr-xr-x 2 root root 0 Oct 13 22:38 .
drwxr-xr-x 12 root root 0 Jan 1 2001 ..
-rw-rw-r-- 1 root root 4096 Oct 13 22:38 dc_motor
-rw-rw-r-- 1 root root 4096 Oct 13 22:38 servo_motor
-rw-rw-r-- 1 root root 4096 Oct 13 22:38 step_motor
```

## DC 모터 제어 명령

/sys/kernel/user-motor-works 경로에 있는 dc\_motor 파일에 다음과 같은 파라미터 값을 쓰넣어 DC 모터를 제어할 수 있습니다.

- index: DC 모터 조합 번호
- dir: DC 모터 회전 방향 [0:시계반대방향, 1:시계방향]
- speed: DC 모터 회전 속도 [0 부터 100]
- running: DC 모터 동작 시간 [미리초 단위로 입력]

index 번호는 4 가지 DC 모터를 다음과 같이 조합시켜 동작 시킵니다.

index	DC 모터 번호				동작모터조합
	1	2	3	4	
1	O				1
2		O			2
3	O	O			1, 2
4			O		3
5	O		O		1, 3
6		O	O		2, 3
7	O	O	O		1, 2, 3
8				O	4
9	O			O	1, 4
10		O		O	2, 4
11	O	O		O	1, 2, 4
12			O	O	3, 4
13	O		O	O	1, 3, 4
14		O	O	O	2, 3, 4
15	O	O	O	O	1, 2, 3, 4

running 시간은 최대 650000ms(650 초)까지 입력 가능하며 0을 입력하면 무한 동작 합니다.

## DC 모터 제어 명령(root 권한에서 실행)

```
//sys/kernel/user-motor-works 경로로 이동
# cd /sys/kernel/user-motor-works/

//DC 모터 1번을 시계 방향으로 80% 속도로 3초간 회전
# echo "index=1 dir=0 speed=80 running=3000" > dc_motor

//DC 모터 2번을 시계 반대방향으로 90% 속도로 1초간 회전
# echo "index=2 dir=1 speed=90 running=1000" > dc_motor

//DC 모터 1번과 2번을 동시에 시계 방향으로 77% 속도로 5초간 회전
# echo "index=3 dir=0 speed=77 running=5000" > dc_motor

//DC 모터 1번과 2번, 3번을 동시에 시계 반대방향으로 55% 속도로 8초간 회전
# echo "index=7 dir=1 speed=55 running=8000" > dc_motor

//DC 모터 1번, 2번, 3번, 4번 모두를 시계 방향으로 32% 속도로 6초간 회전
# echo "index=15 dir=0 speed=32 running=6000" > dc_motor
```

## 서보 모터 제어 명령

/sys/kernel/user-motor-works 경로에 있는 servo\_motor 파일에 다음과 같은 파라미터 값을 쓰넣어 서보 모터를 제어할 수 있습니다.

- index: 서보 모터 조합 번호
- dir: 서보 모터 회전 방향 [0:중앙, 1:왼쪽, 2:오른쪽]

index 번호는 2 가지 서보 모터를 다음과 같이 조합시켜 동작 시킵니다.

index	서보 모터 번호				동작모터조합
	1	2	3	4	
1	O				1
2		O			2
3	O	O			1, 2

## 서보 모터 제어 명령(root 권한에서 실행)

```
//sys/kernel/user-motor-works 경로로 이동
# cd /sys/kernel/user-motor-works/

//서보 모터 1번을 중앙 위치로 회전
# echo "index=1 dir=0" > servo_motor

//서보 모터 2번을 왼쪽으로 회전
# echo "index=2 dir=1" > servo_motor

//서보 모터 1번과 2번을 동시에 오른쪽으로 회전
# echo "index=3 dir=2" > servo_motor
```

## 스테핑 모터 제어 명령

/sys/kernel/user-motor-works 경로에 있는 step\_motor 파일에 다음과 같은 파라미터 값을 쓰넣어 스테핑 모터를 제어할 수 있습니다.

- index: 스테핑 모터 번호
- dir: 스테핑 모터 회전 방향 [0:시계반대방향, 1:시계방향]
- speed: 스테핑 모터 회전 속도 [0 부터 100]

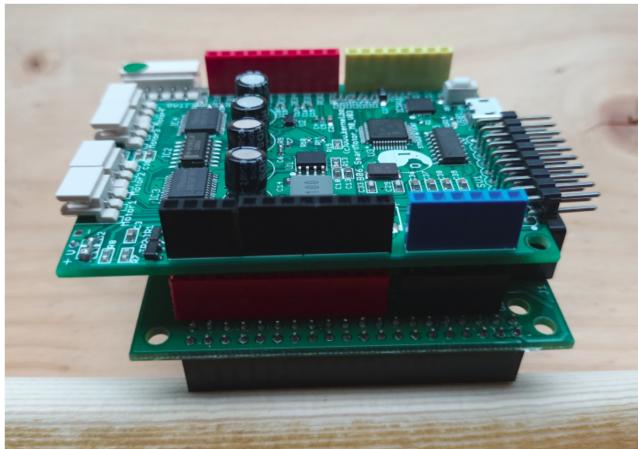
### 스테핑 모터 제어 명령(root 권한에서 실행)

```
//sys/kernel/user-motor-works 경로로 이동  
# cd /sys/kernel/user-motor-works/  
  
//스테핑 모터 1번을 시계 방향으로 50% 속도로 회전  
# echo "index=1 dir=0 speed=50" > step_motor  
  
//스테핑 모터 1번을 시계 반대방향으로 90% 속도로 회전  
# echo "index=1 dir=1 speed=90" > step_motor
```

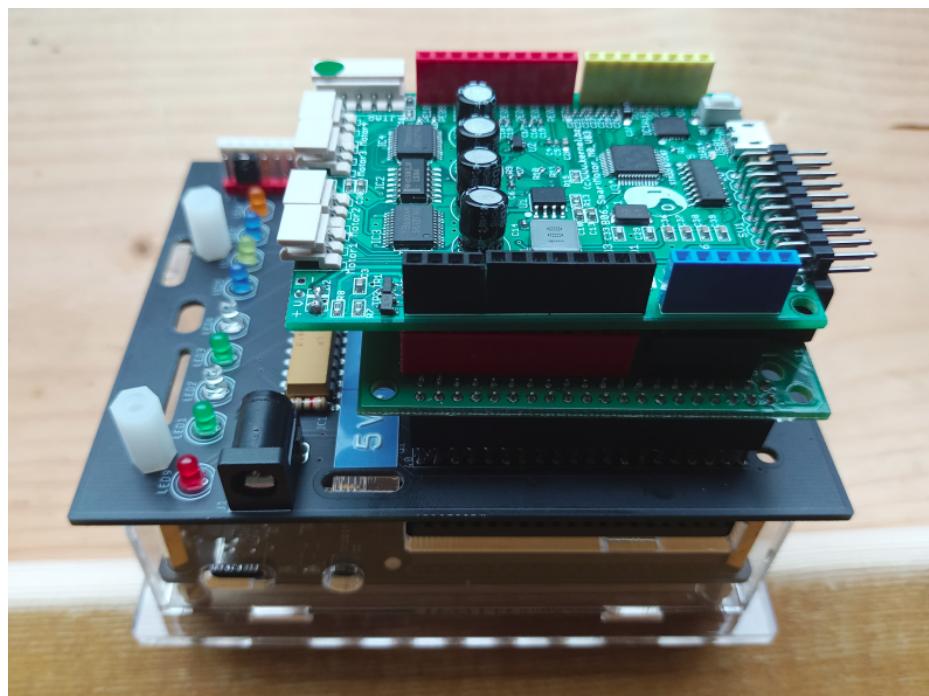
좀더 자세한 내용들은 커널연구회 홈페이지를 통하여 확인해 주시기 바랍니다.

## 확장 보드 조합 구성

### 모터 제어 보드



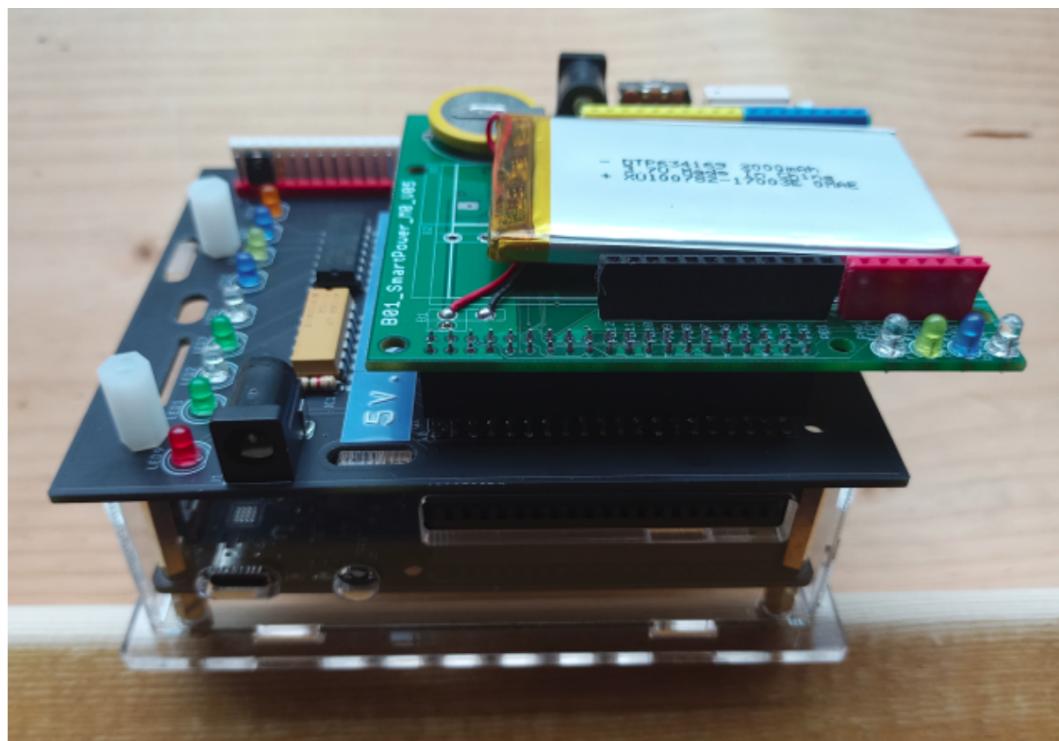
### 모터 제어 + RISCV 입출력 보드



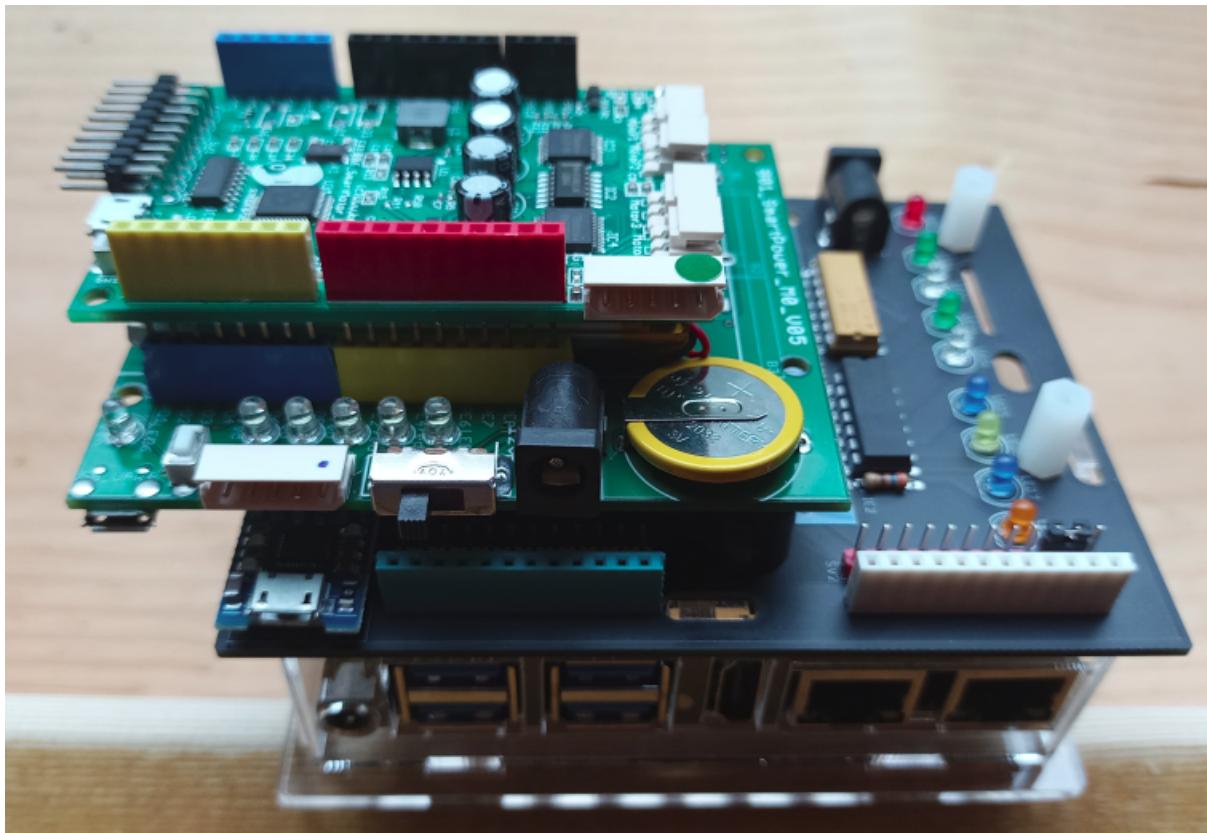
## 배터리 파워 보드



## 배터리 파워 + RISCV 입출력 보드



## 배터리 파워 + 모터 제어 + RISCV 입출력 보드



좀더 자세한 내용들은 커널연구회 홈페이지를 통하여 확인해 주시기 바랍니다.

감사합니다.