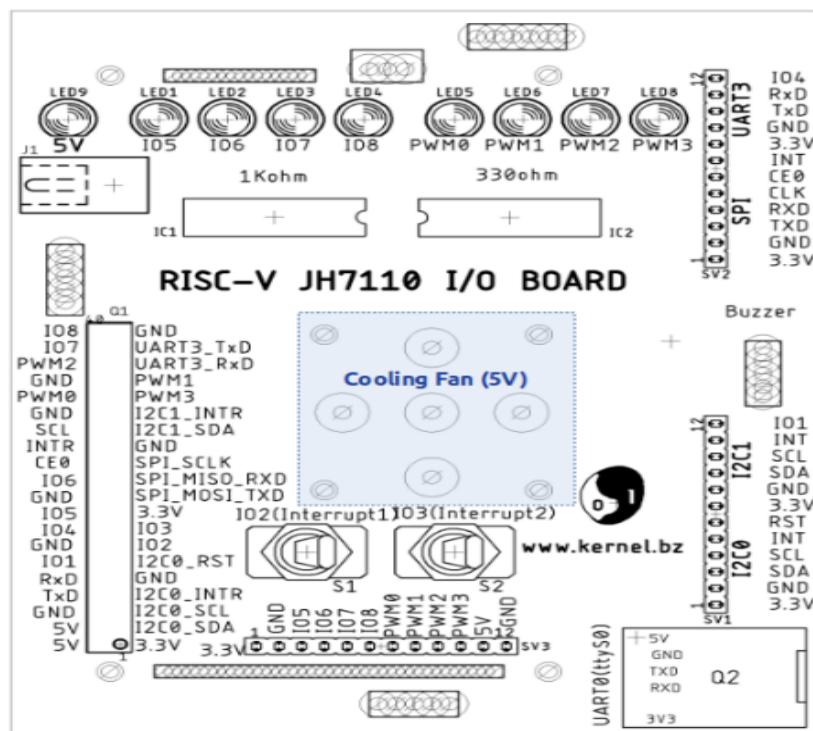


RISC-V 64 비트 StarFive JH7110 40 핀맵 호환

RISC-V 실습용 입출력 보드 매뉴얼

RISC-V 입출력 보드



커널연구회 (www.kernel.bz)
정재준 (rghi3307@nate.com)

목차

Table of Contents

● RISC-V 실습용 입출력 보드 매뉴얼.....	1
● 목차.....	2
● RISC-V 실습용 입출력 보드.....	5
기능 요약.....	5
외형 사진.....	6
입출력 연결 구성도.....	10
40 핀맵 구성도.....	11
실습용 소스 제공(GitHub).....	12
● 입출력 장치 활용 방법.....	14
Serial(UART) 사용 방법.....	14
실습용 보드 핀맵 위치.....	16
UART3 포트 추가.....	18
UART3 시리얼 테스트.....	19
GPIO 기능 사용 방법.....	21
실습용 보드 핀맵 위치.....	21
GPIO LED 출력 테스트.....	22
GPIO 제어용 소스 활용.....	23

인터럽트 스위치 사용 방법.....	24
실습용 보드 핀맵 위치.....	24
인터럽트 디바이스 드라이버 모듈.....	26
I2C 사용 방법.....	27
실습용 보드 핀맵 위치.....	27
i2c 테스트 도구.....	28
i2c0 포트 확인.....	30
i2c1 포트 추가.....	30
I2C 온도 습도 센서(HT30) 사용 방법.....	32
온도 습도 센서(HT30) 정보.....	32
온도 습도 센서(HT30) 데이터 읽기.....	34
I2C 조도센서(BH1750) 사용 방법.....	36
조도센서(BH1750) 정보.....	36
조도센서(BH1750) 데이터 읽기.....	37
PWM 활용 방법.....	39
실습용 보드 핀맵 위치.....	39
PWM 출력 테스트.....	40
PWM 제어용 소스 활용.....	41
모터 제어 PWM.....	42
SPI 활용 방법.....	44
실습용 보드 핀맵 위치.....	44
SPI 제어용 소스 활용.....	45
● 입출력 확장 기능.....	48

목차

입출력 적층 연결핀 구성도.....	48
배터리 Power 보드 연결 구성도.....	49
Motor 제어 보드 연결 구성도.....	50
리눅스 커널에서 모터 드라이버 구동 방법.....	52
DC 모터 제어 명령.....	54
서보 모터 제어 명령.....	56
스테핑 모터 제어 명령.....	57
● 확장 보드 조합 구성.....	58
모터 제어 보드.....	58
모터 제어 + RISCV 입출력 보드.....	58
배터리 파워 보드.....	59
배터리 파워 + RISCV 입출력 보드.....	59
배터리 파워 + 모터 제어 + RISCV 입출력 보드.....	60

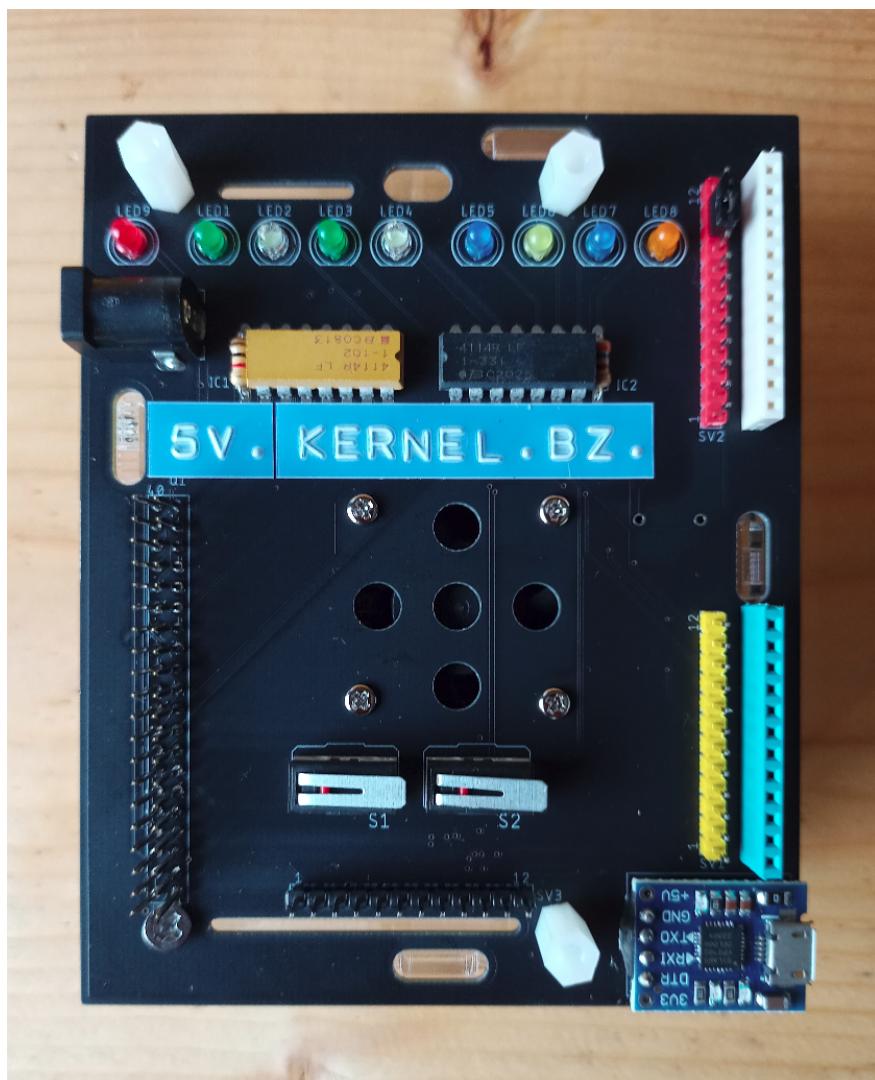
RISC-V 실습용 입출력 보드

기능 요약

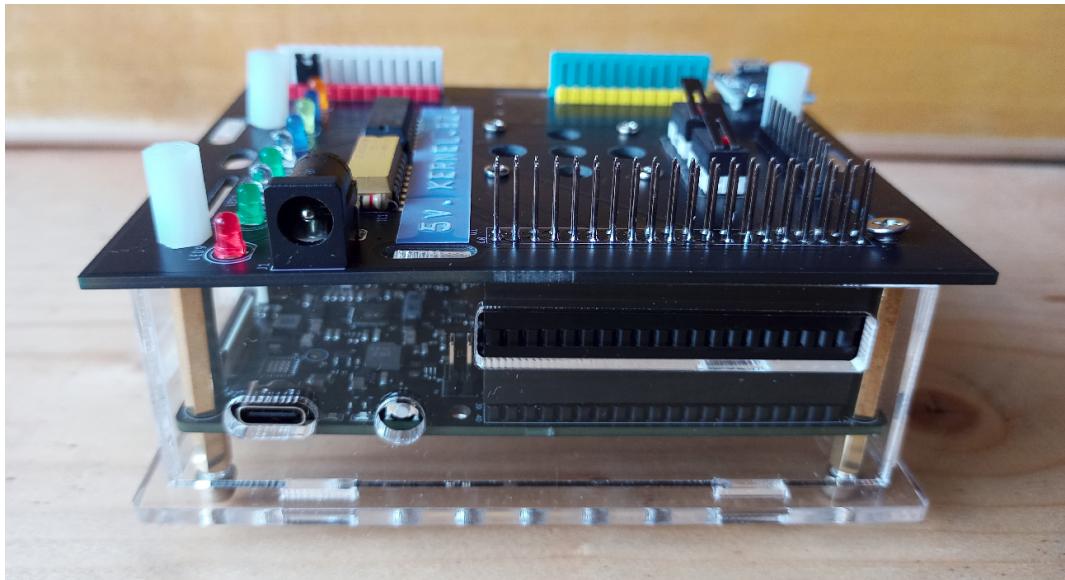
- RISC-V 64 비트 StarFive JH7110 40 핀맵 호환
- Serial(UART0, ttyS0) to USB 포트 탑재 (디버그 콘솔)
- Serial(UART3, ttyS3) 포트 지원
- I2C0, I2C1 포트 지원 (센서 연결)
- SPI 포트 지원 (센서 연결)
- PWM0, PWM1, PWM2, PWM3 포트 출력 (모터 연결)
- GPIO 포트 및 LED 출력 지원
- Switch1, Switch2 외부 인터럽트 처리 지원
- 5V 외부 전원 입력 RCA 잭
- 쿨링팬 연결 지원
- 40 핀 입출력핀 확장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

외형 사진

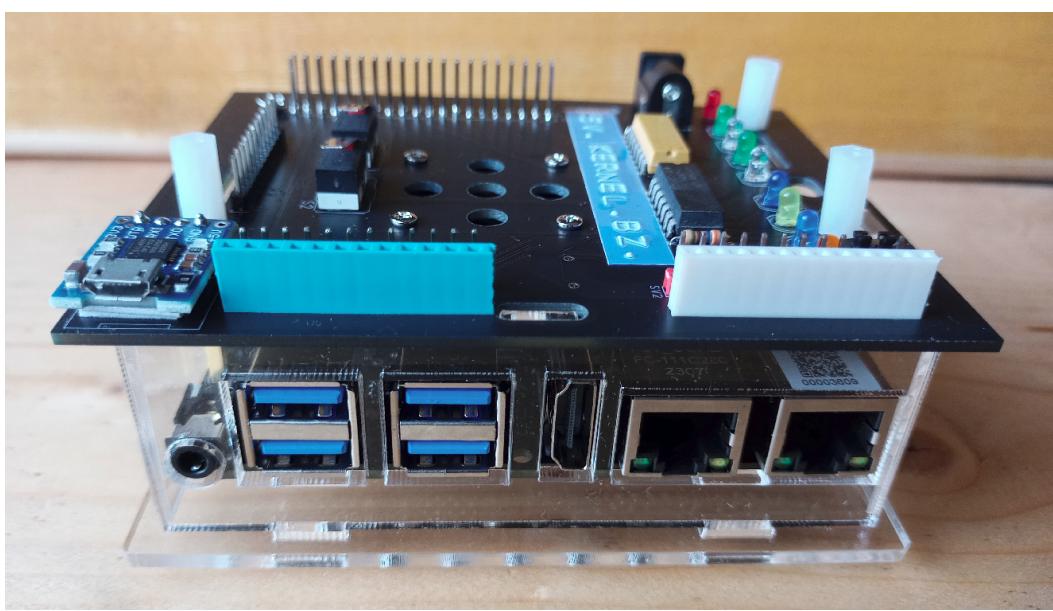
상단부



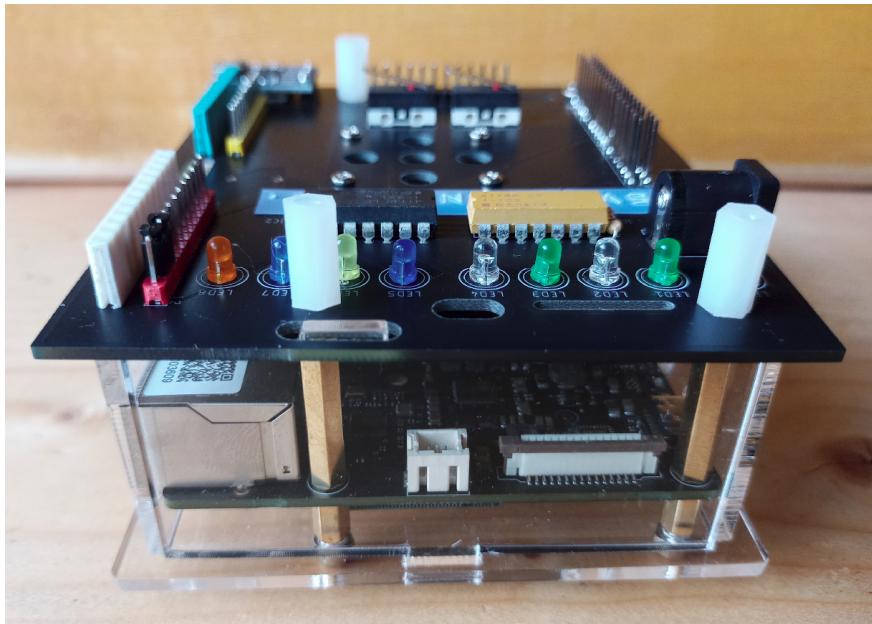
측면부 1 (정면)



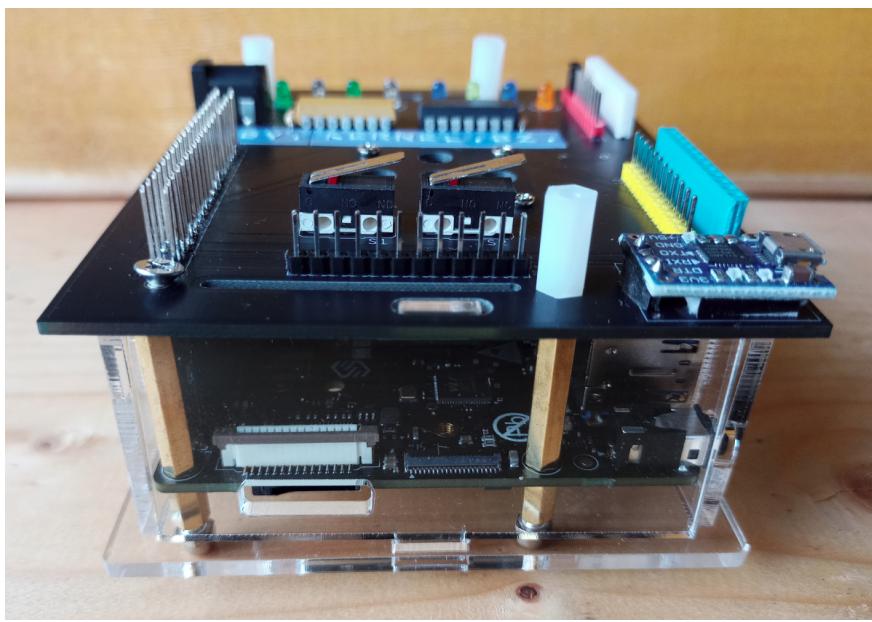
측면부 2 (후면)



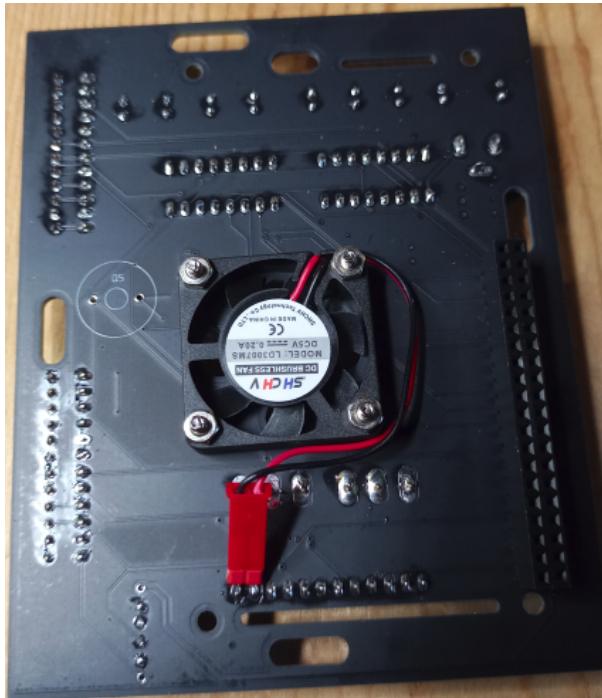
측면부 3 (좌측면)



측면부 4 (우측면)

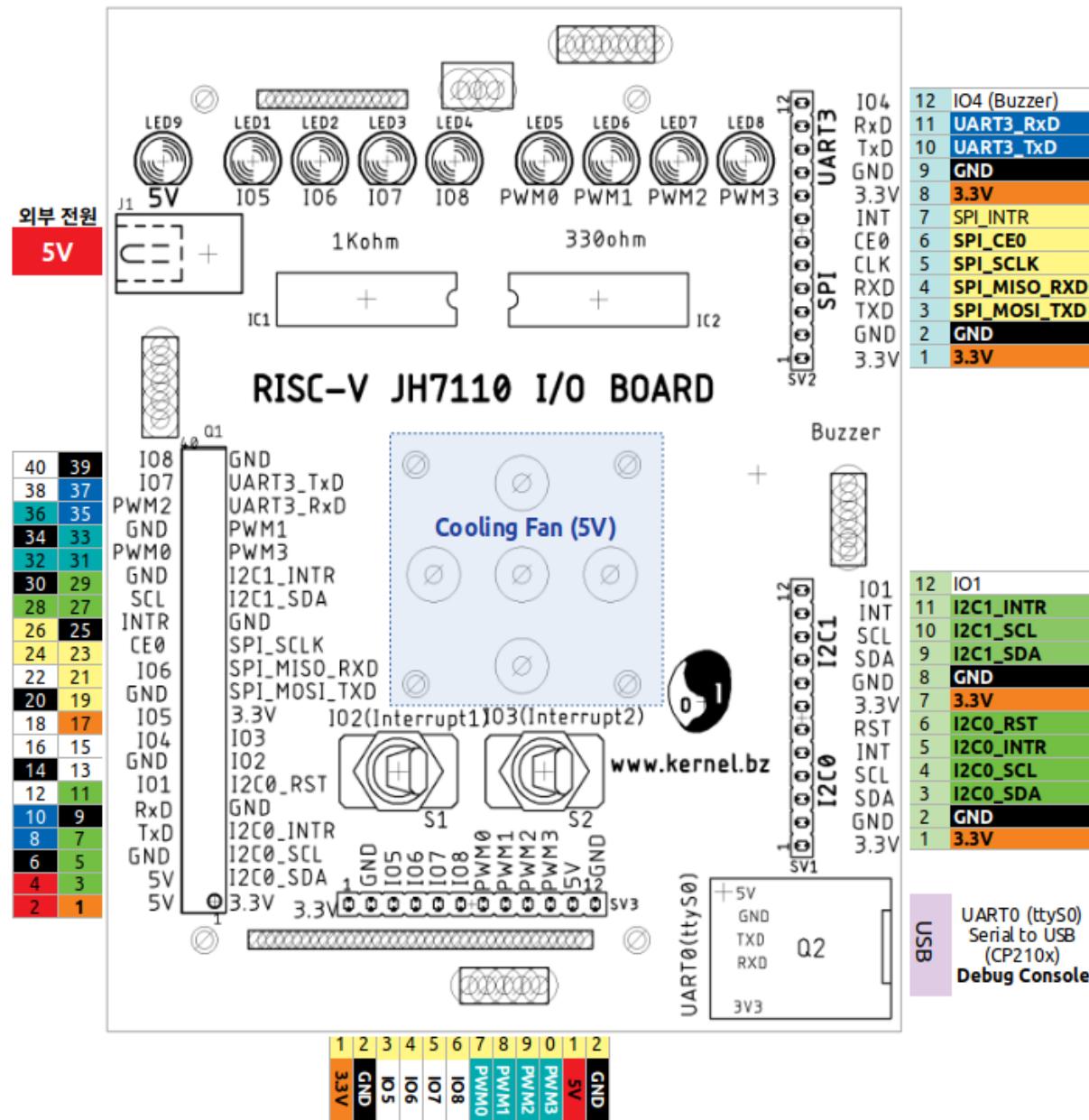


보드 하단부(쿨링팬 옵션)



쿨링팬에 보드 하단부에 장착할 수 있습니다. 전원은 왼쪽 1 번핀이 GND(흑색), 2 번핀이 +5V(적색)입니다. GND(흑색)과 +5V(적색)이 바뀌지 않도록 주의해 주세요. 쿨링팬은 추가 선택사항(옵션)입니다.

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	UART3 (TxD)	
LED8	PWM2	36	36	35	UART3 (RxT)	ttyS3
	GND		34	33	PWM1	LED6
LED5	PWM0	46	32	31	PWM3	LED7
	GND		30	29	I2C1 (INTR)	
I2C1	I2C1 (SCL)	40	28	27	I2C1 (SDA)	I2C1
SPI	SPI (INTR)	56	26	25	GND	
	SPI (CE0)	49	24	23	SPI (SCLK)	
LED2	IO6	50	22	21	SPI (MISO) RxD	SPI
	GND		20	19	SPI (MOSI) TxD	
LED1	IO5	51	18	17	3.3V	
Buzzer	IO4	54	16	15	IO3	Switch2
	GND		14	13	IO2	Switch1
ttyS0	IO1	38	12	11	I2C0 (REST)	
	UART0 (RxT)	6	10	9	GND	
	UART0 (TxT)	5	8	7	I2C0 (INTR)	I2C0
	GND		6	5	I2C0 (SCL)	
	5V		4	3	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
|   |   |   |-- gpio-test
|   |   |   |-- 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
|   |   |   |   `-- pwm.c
|   |   |-- spi (SPI 실습 소스)
|   |   |   |-- build.sh
|   |   |   |-- spi
|   |   |   `-- spi.c
|   `-- uart (UART 실습 소스)
|       |-- build.sh
|       |-- recv
```

RISC-V 실습용 입출력 보드

```
|      |-- recv.c
|      |-- send
|      |-- send.c
|      |-- serial.c
|      `-- serial.h
|-- docs (매뉴얼 문서)
|  |-- riscv-jh7110-io-man-adv.pdf
|  `-- riscv-jh7110-io-manual.pdf
|-- 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 장치 활용 방법

RISC-V 64 비트 StarFive JH7110 보드에서 제공하는 40 핀맵의 입출력 장치들을 쉽게 연결하여 활용할 수 있도록 구성하였습니다. I2C1 포트와 UART3, PWM3, PWM4는 커널연구회에서 디바이스 트리를 추가 하였고, 인트럽트를 실습하기 위한 Switch1, Switch2는 커널연구회에서 디바이스 드라이버 모듈로 기능을 추가 하였습니다.

Serial(UART) 사용 방법

Serial UART0(/dev/ttys0) 포트는 StarFive JH7110 커널 디바이스 드라이버에서 기본적으로 제공하는 기능입니다. 커널연구회에서 Serial to USB(CP210x)을 추가로 장착하여 USB 케이블만 간편하게 PC에 연결하여 디버그 콘솔을 사용할 수 있도록 했습니다. Serial to USB(CP210x) 장치는 리눅스 터미널 명령어 lsusb로 다음과 같이 확인 가능 합니다. (자동으로 장치 설정됨)

Serial to USB(CP210x) 장치 정보 확인

```
$ lsusb
Bus 001 Device 012: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x
UART Bridge / myAVR mySmartUSB light
$ ll /dev/ttys*
crw-rw---- 1 root dialout 188, 0 10월  9 16:18 /dev/ttys0
```

シリアル 통신 파라미터 설정

シリアル 통신 속도:	115200bps
데이터 비트:	8 비트
Stop 비트:	1 비트
Parity 비트:	1 비트
하드웨어 흐름 제어:	None
소프트웨어 흐름 제어:	None

Linux minicom 시리얼 포트 설정 정보

```
+-----+
| A - Serial Device      : /dev/ttyUSB0
| B - Lockfile Location  : /var/lock
| C - Callin Program     :
| D - Callout Program    :
| E - Bps/Par/Bits       : 115200 8N1
| F - Hardware Flow Control : No
| G - Software Flow Control : No
|
| Change which setting? [ ]
+-----+
```

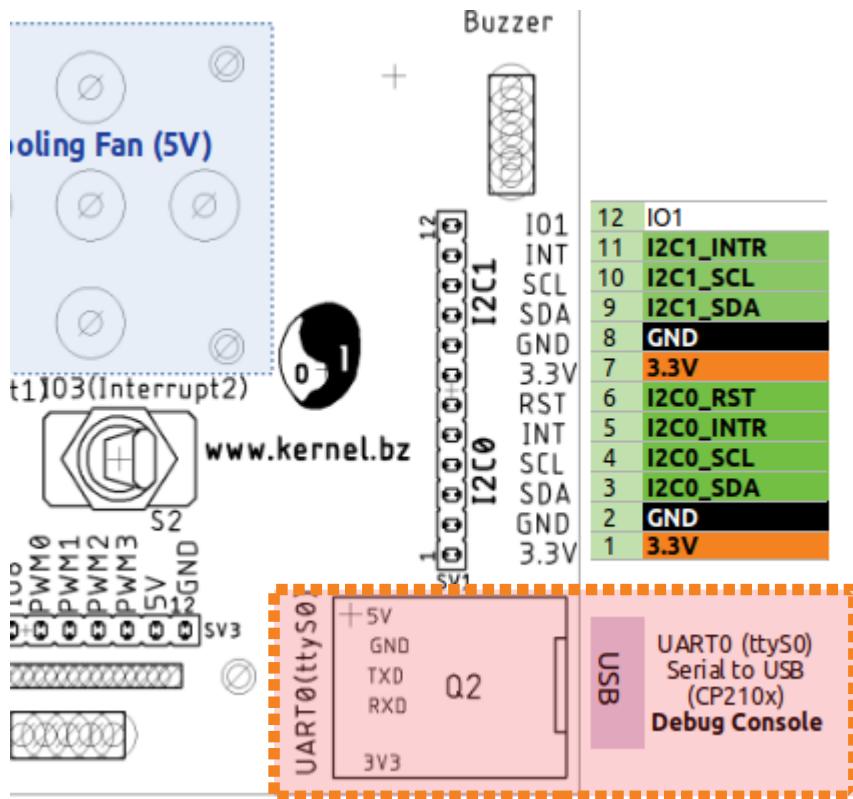
실습용 보드 핀맵 위치

UART0 핀맵 위치

device	function	GPIO	pin number	GPIO	function	device
LED4	IO8	44	40	39	GND	
LED3	IO7	61	38	37	UART3 (TxD)	ttyS3
LED8	PWM2	36	36	35	UART3 (RxD)	
	GND		34	33	PWM1	LED6
LED5	PWM0	46	32	31	PWM3	LED7
	GND		30	29	I2C1 (INTR)	
I2C1	I2C1 (SCL)	40	28	27	I2C1 (SDA)	I2C1
SPI	SPI (INTR)	56	26	25	GND	
	SPI (CEO)	49	24	23	SPI (SCLK)	
LED2	IO6	50	22	21	SPI (MISO) RxD	SPI
	GND		20	19	SPI (MOSI) TxD	
LED1	IO5	51	18	17	3.3V	
Buzzer	IO4	54	16	15	IO3	Switch2
	GND		14	13	IO2	Switch1
	IO1	38	12	11	I2C0 (REST)	
ttyS0	UART0 (RxD)	6	10	9	GND	
	UART0 (TxD)	5	8	7	I2C0 (INTR)	I2C0
	GND		6	5	I2C0 (SCL)	
	5V		4	3	I2C0 (SDA)	
	5V		2	1	3.3V	

입출력 장치 활용 방법

실습용 보드 위치(UART0)

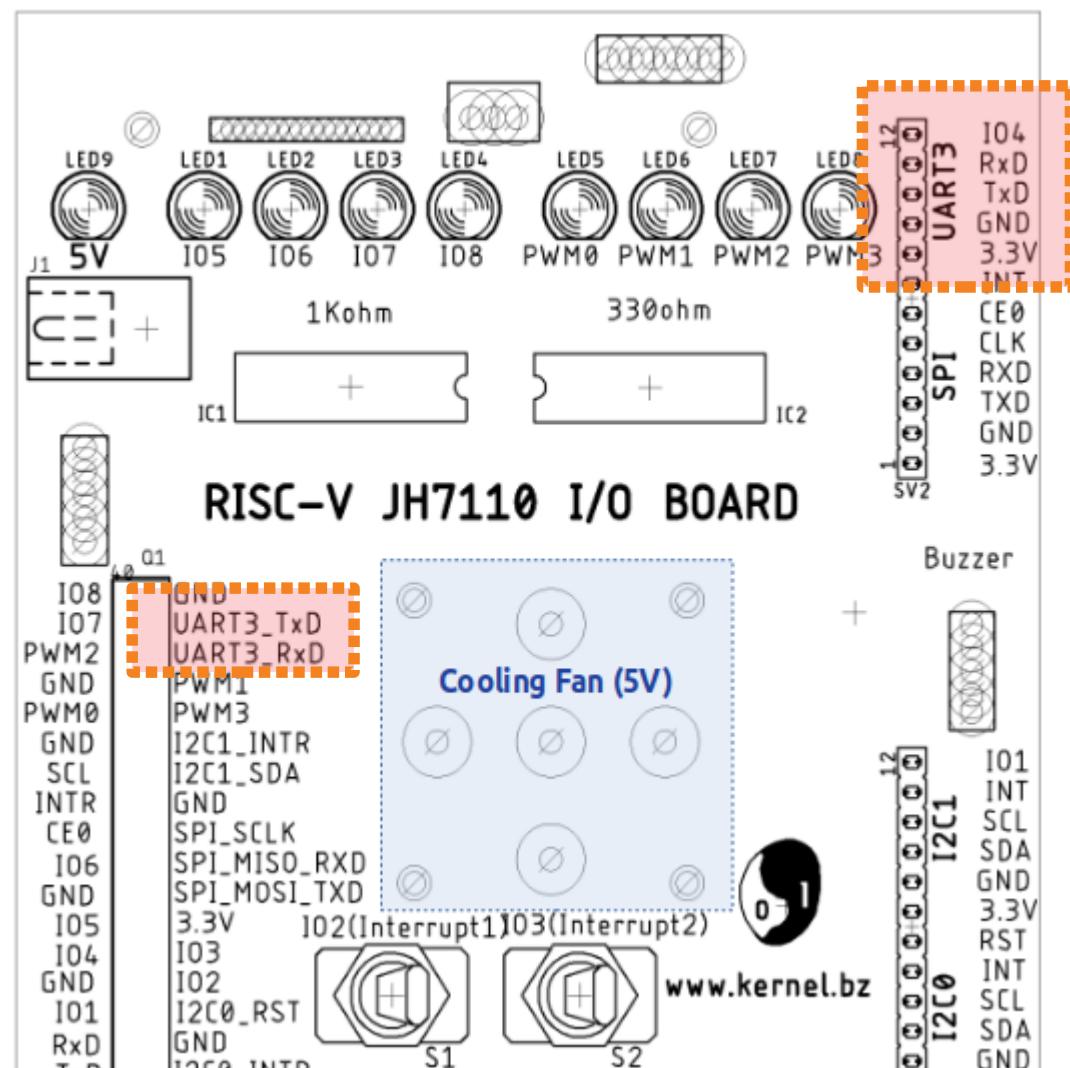


위의 Serial UART0(/dev/ttyS0) 포트에 USB 케이블을 연결하고 Serial 터미널 프로그램에서 통신 속도를 115200 bps로 설정하면 Debug Console 기능을 사용할 수 있습니다.

UART3 포트 추가

uart3 포트는 StarFive JH7110 커널에서 기본적으로 제공하지 않으므로, 다음과 같이 커널연구회에서 제공하는 디바이스 트리 overlay를 적용하여 사용 가능 합니다.

실습보드 위치



overlay 디바이스트리 적용 (root 권한에서 실행)

```
//커널연구회에서 제공한 실습 소스 경로로 이동
# cd riscv-jh7110-io/dtb/overlay/

# ll

-rwxr-xr-x 1 user user 275 Oct 10 19:51 run.sh
-rwxr-xr-x 1 root root 1075 Oct 10 19:50 vf2-overlay-i2c1.dtbo
-rwxr-xr-x 1 root root 866 Oct 10 19:50 vf2-overlay-uart3.dtbo

//run.sh 스크립트를 실행하여 overlay 디바이스 트리 적용
# ./run.sh
```

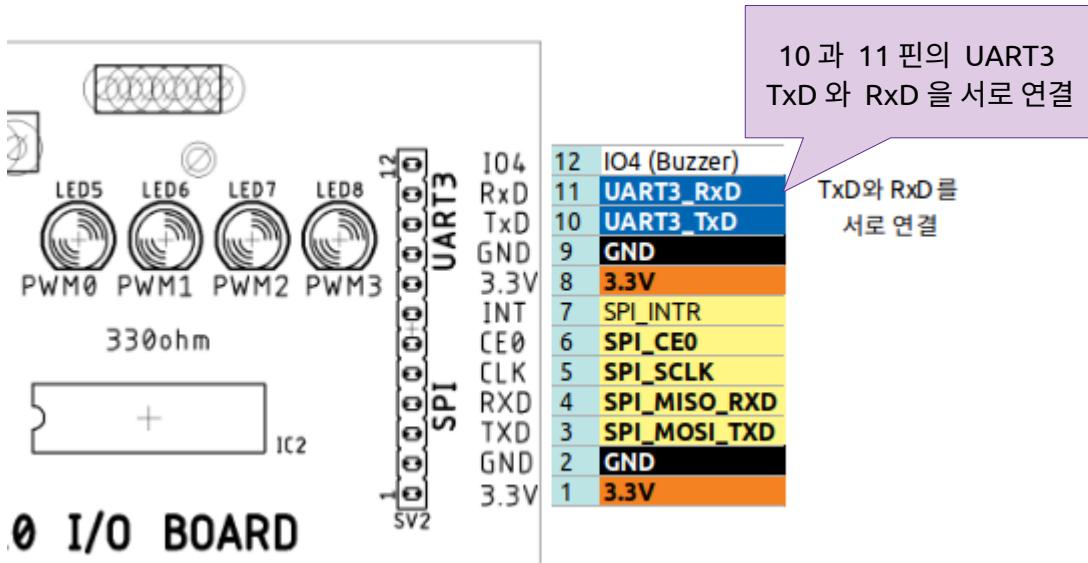
uart3 장치파일 확인

```
# ll /dev/ttys*
crw----- 1 root tty      4, 64 Oct  8 23:53 /dev/ttys0
crw-rw---- 1 root dialout 4, 65 Jun 10 2022 /dev/ttys1
crw-rw---- 1 root dialout 4, 66 Jun 10 2022 /dev/ttys2
crw-rw---- 1 root dialout 4, 67 Oct  8 23:52 /dev/ttys3
crw-rw---- 1 root dialout 4, 68 Jun 10 2022 /dev/ttys4
crw-rw---- 1 root dialout 4, 69 Jun 10 2022 /dev/ttys5
```

UART3 시리얼 테스트

UART3을 테스트하기 위해서 아래 그림과 같이 10과 11핀의 UART3 TxD와 RxD을 헤더캡을 사용하여 서로 연결합니다. (점퍼선으로 연결해도 됨) 이렇게 하면 UART3에서 전송한 데이터를 로컬에서 바로 수신 받을 수 있습니다. 로컬에서 시리얼 루프 테스트는 이런 방식으로 합니다.

입출력 장치 활용 방법



커널연구회에서 제공한 시리얼 테스트 프로그램을 다음과 같이 실행 합니다. TxD 와 RxD 를 연결 했으므로 데이터를 입력하여 전송(Send)하면 그대로 수신(Recv) 됩니다.

시리얼 테스트 프로그램 실행(root 권한에서 실행)

```
//커널연구회에서 제공한 실습 소스 경로로 이동
# cd riscv-jh7110-io/app/uart/

//첫번째 파라미터는 시리얼 포트 번호.
# ./send 3

>Send: hello uart3
<Recv: hello uart3

>Send: test so good.
<Recv: test so good.
```

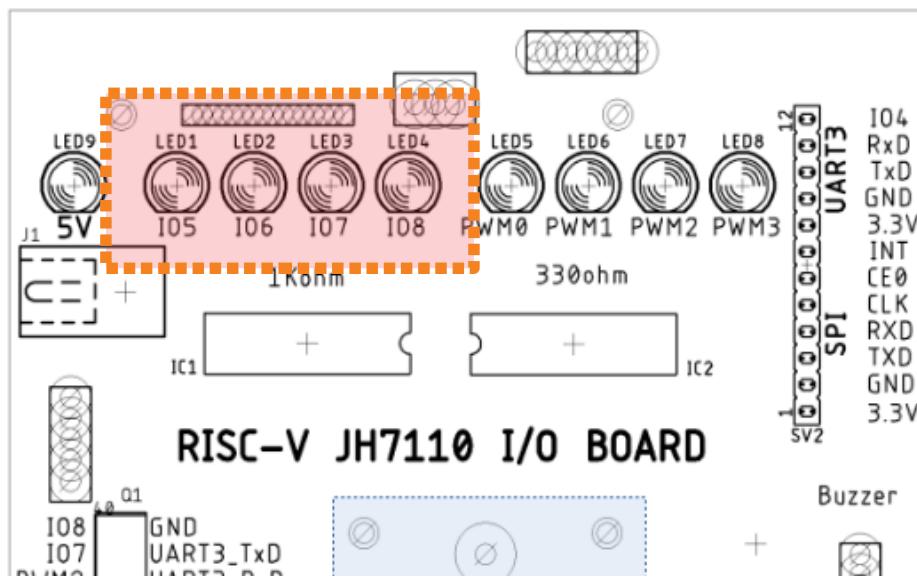
GPIO 기능 사용 방법

GPIO 는 StarFive JH7110 커널에서 기본적으로 제공하기 때문에 이것을 그대로 사용합니다. 다음과 같이 GPIO 51, 50, 61, 44 번을 LED1, 2, 3, 4 에 연결 했습니다.

실습용 GPIO 번호 테이블 (LED 연결)

LED 출력 기호	GPIO 제어 파일	GPIO 번호	헤더 40 핀 번호
IO5 (LED1)	/sys/class/gpio/gpio51	51	18
IO6 (LED2)	/sys/class/gpio/gpio50	50	22
IO7 (LED3)	/sys/class/gpio/gpio61	61	38
IO8 (LED4)	/sys/class/gpio/gpio44	44	40

실습용 보드 핀맵 위치



GPIO LED 출력 테스트

GPIO 는 StarFive JH7110 커널에서 기본적으로 제공하는 /sys/class/gpio 파일에서 다음과 같이 테스트 할 수 있습니다.

GPIO 51 핀 제어 파일 생성

```
# cd /sys/class/gpio
# echo 51 > export
# cd gpio51
# ls -al

-rw-r--r-- 1 root root 4096 Jun 11 01:59 active_low
lrwxrwxrwx 1 root root      0 Jun 11 01:59 device -> ../../../../../../gpiochip0
-rw-r--r-- 1 root root 4096 Jun 11 01:59 direction
-rw-r--r-- 1 root root 4096 Jun 11 01:59 edge
drwxr-xr-x 2 root root      0 Jun 11 01:59 power
lrwxrwxrwx 1 root root      0 Jun 11 01:59 subsystem -
> ../../../../../../class/gpio
-rw-r--r-- 1 root root 4096 Jun 11 01:58 uevent
-rw-r--r-- 1 root root 4096 Jun 11 01:59 value
```

GPIO 51 핀에 (High/Low) 출력하기

```
# echo out > direction
# echo 1 > value
# cat value
1
# echo 0 > value
# cat value
0
```

GPIO 제어용 소스 활용

커널연구회에서 제공한 GPIO 테스트 프로그램을 다음과 같이 활용할 수 있습니다

GPIO 테스트 프로그램 실행(root 권한에서 실행)

```
//커널연구회에서 제공한 실습 소스 경로로 이동
# cd riscv-jh7110-io/app/gpio/

//첫번째 파라미터는 테스트 반복(loop) 회수, 입력하지 않으면 무한 loop로 실행
# ./gpio-test 5

gpio loop count:<5>, step:<1>
_gpio_export:: gpio:51, cnt:2
_gpio_export:: gpio:50, cnt:2
_gpio_export:: gpio:61, cnt:2
_gpio_export:: gpio:44, cnt:2
_gpio_unexport:: gpio:51, cnt:2
_gpio_unexport:: gpio:50, cnt:2
_gpio_unexport:: gpio:61, cnt:2
_gpio_unexport:: gpio:44, cnt:2
```

위와 같이 실행하면 LED1, 2, 3, 4 가 1 초 간격으로 번갈아 가면서 깜박입니다. 자세한 내용은

커널연구회에서 제공한 gpio 소스를 확인하여 분석할 수 있습니다.

인터럽트 스위치 사용 방법

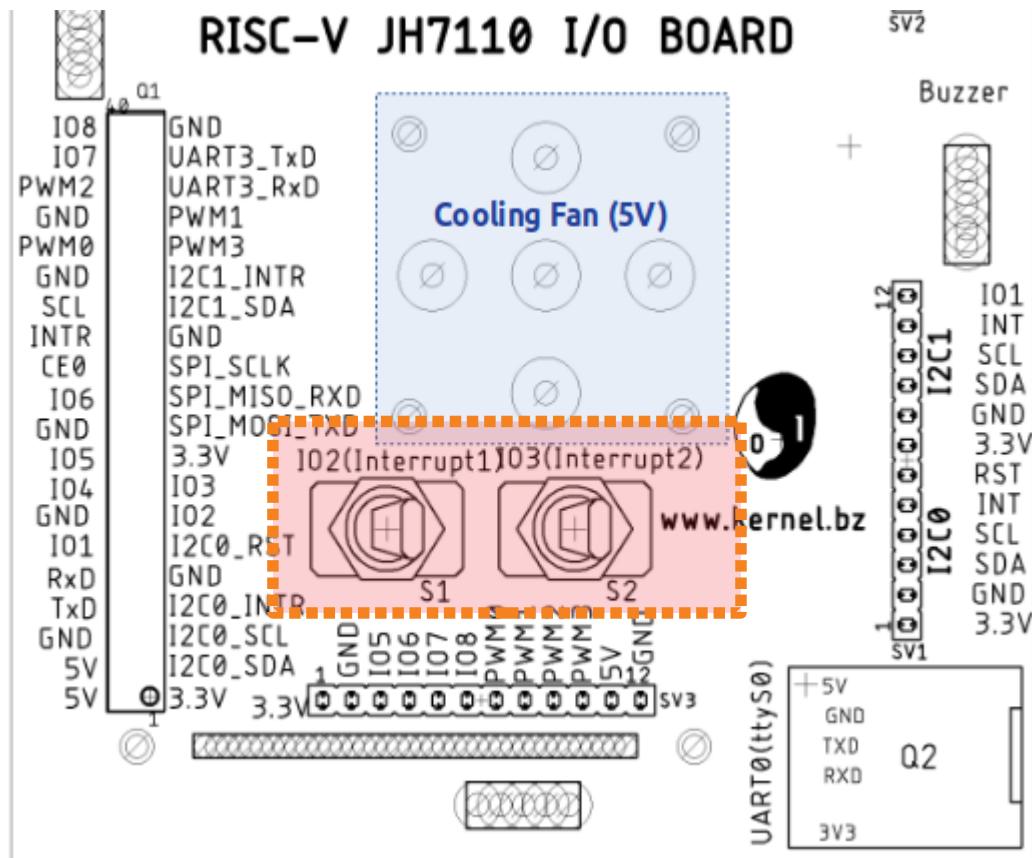
인터럽트는 스위치는 커널연구회에서 GPIO43 (IO2)와 GPIO47 (IO3)에 추가한 기능입니다. 이것을 실습하기 위해서는 커널연구회에서 제공한 디바이스 드라이버 모듈을 추가해야 합니다.

실습용 보드 핀맵 위치

인터럽트 헤더핀 위치(스위치)

device	function	GPIO	pin number	GPIO	function	device
LED4	IO8	44	40	39	GND	
LED3	IO7	61	38	37	UART3 (TxD)	
LED8	PWM2	36	36	35	UART3 (RxD)	ttyS3
	GND		34	33	PWM1	LED6
LED5	PWM0	46	32	31	PWM3	LED7
	GND		30	29	I2C1 (INTR)	I2C1
I2C1	I2C1 (SCL)	40	28	27	I2C1 (SDA)	
SPI	SPI (INTR)	56	26	25	GND	
	SPI (CEO)	49	24	23	SPI (SCLK)	
LED2	IO6	50	22	21	SPI (MISO) RXD	SPI
	GND		20	19	SPI (MOSI) TXD	
LED1	IO5	51	18	17	3.3V	
Buzzer	IO4	54	16	15	IO3	Switch2
	GND		14	13	IO2	Switch1
	IO1	38	12	11	I2C0 (REST)	
ttyS0	UART0 (RXD)	6	10	9	GND	
	UART0 (TXD)	5	8	7	I2C0 (INTR)	I2C0
	GND		6	5	I2C0 (SCL)	
	5V		4	3	I2C0 (SDA)	
	5V		2	1	3.3V	

실습용 보드 위치(인터럽트 스위치)



인터럽트 디바이스 드라이버 모듈

커널연구회에서 제공한 디바이스 드라이버 모듈을 다음과 같이 insmod 하고 실행 합니다.

인터럽트 드라이버 모듈 실행(root 권한에서 실행)

```
//커널연구회에서 제공한 실습 경로로 이동  
# cd riscv-jh7110-io/modules/  
  
//user-gpio-driver.ko을 커널에 삽입  
# insmod user-gpio-driver.ko  
  
[ 4494.711087] gpio_driver_init:: Major:244, Minor:0  
[ 4494.716029] gpio_driver_init:: gpio_irq_sw1 = 66  
[ 4494.720778] gpio_driver_init:: gpio_irq_sw2 = 67  
[ 4494.725447] gpio_driver_init:: Done. End.
```

위와 같이 user-gpio-driver.ko 드라이버 모듈을 커널에 삽입하고 Swich1 을 누르면 인터럽트를 처리하여 LED1, 2 가 점등되고, Swich2 을 누르면 인터럽트를 처리하여 LED3, 4 가 점등됩니다. 커널 디바이스 드라이버 모듈에서 인터럽트가 실행되는 과정은 본 매뉴얼에서 다루는 범위가 아니고, 커널연구회에서 제공하는 교육을 신청하여 관련 소스와 함께 도움을 받을 수 있습니다.

I2C 사용 방법

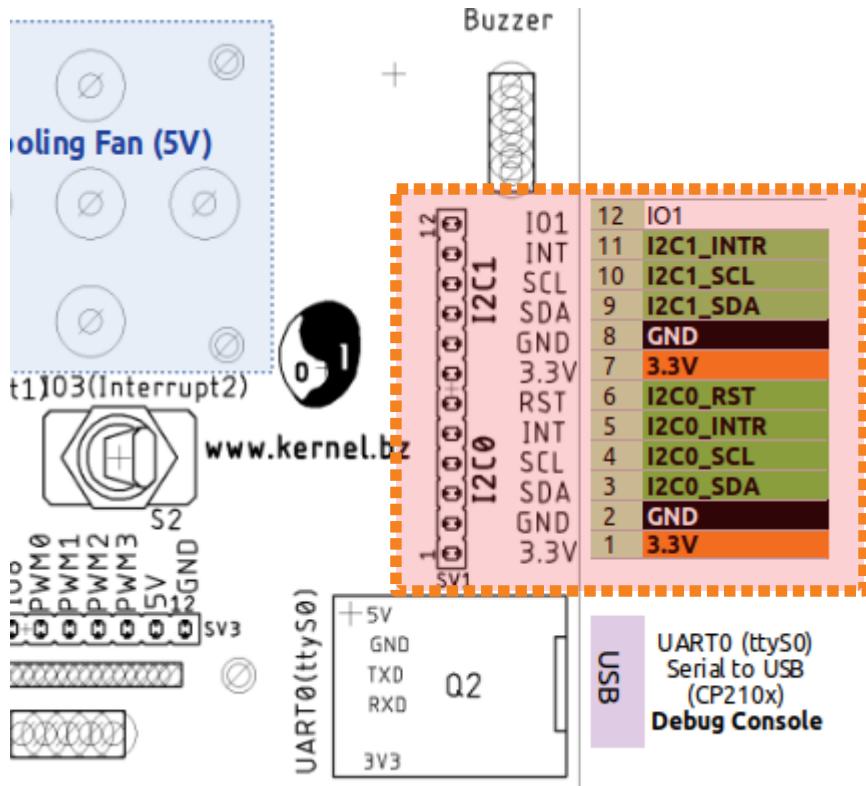
I2C0 번 포트는 StarFive JH7110 커널에서 기본적으로 제공하는 것이고 I2C1 번 포트는 커널연구회에서 디바이스 트리를 추가한 것입니다.

실습용 보드 핀맵 위치

I2C 헤더핀 위치

device	function	GPIO	pin number	GPIO	function	device
LED4	IO8	44	40	39	GND	
LED3	IO7	61	38	37	UART3 (TxD)	ttyS3
LED8	PWM2	36	36	35	UART3 (RxD)	
	GND		34	33	PWM1	LED6
LED5	PWM0	46	32	31	PWM3	LED7
	GND		30	29	I2C1 (INTR)	
I2C1	I2C1 (SCL)	40	28	27	I2C1 (SDA)	I2C1
SPI	SPI (INTR)	56	26	25	GND	
	SPI (CEO)	49	24	23	SPI (SCLK)	
LED2	IO6	50	22	21	SPI (MISO) RxD	SPI
	GND		20	19	SPI (MOSI) TxD	
LED1	IO5	51	18	17	3.3V	
Buzzer	IO4	54	16	15	IO3	Switch2
	GND		14	13	IO2	Switch1
	IO1	38	12	11	I2C0 (REST)	
ttyS0	UART0 (RxD)	6	10	9	GND	I2C0
	UART0 (TxD)	5	8	7	I2C0 (INTR)	
	GND		6	5	I2C0 (SCL)	
	5V		4	3	I2C0 (SDA)	
	5V		2	1	3.3V	

실습용 보드 위치(I2C 포트)



i2c 테스트 도구

I2C는 i2c-tools 패키지 프로그램을 “`apt install i2c-tools`”로 설치하여 다음과 같이 기본적인 내용은 확인할 수 있습니다.

i2cdetect (root 권한으로 실행)

```
$ sudo i2cdetect -l
```

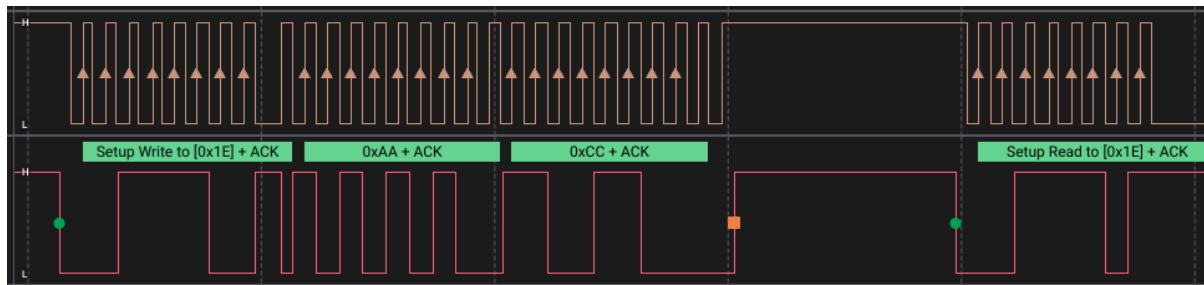
입출력 장치 활용 방법

i2c-0	i2c	Synopsys DesignWare I2C adapter	I2C adapter
i2c-2	i2c	Synopsys DesignWare I2C adapter	I2C adapter
i2c-5	i2c	Synopsys DesignWare I2C adapter	I2C adapter
i2c-6	i2c	Synopsys DesignWare I2C adapter	I2C adapter
i2c-7	i2c	Inno HDMI	I2C adapter

I2C 데이터 전송(Master --> Slave 쓰기) (root 권한으로 실행)

```
//I2C0 의 0x1E 주소의 오프셋 0 에 0xAA, 0xCC 데이터 보내기
```

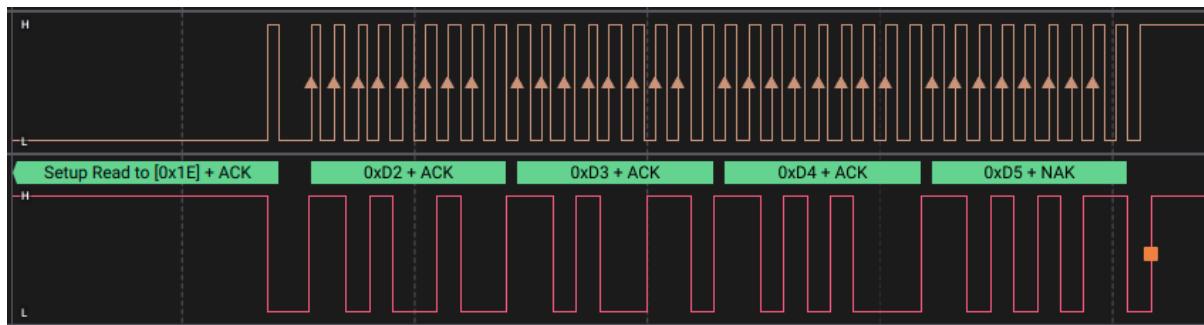
```
$ sudo i2cset -y 0 0x1E 0x0 0xAA 0xCC
```



I2C 데이터 수신(Master <- Slave 읽기) (root 권한으로 실행)

```
//I2C0 의 0x1E 주소의 오프셋 0에서 데이터 받기(4 바이트 수신)
```

```
$ sudo i2cget -y 0 0x1E 0x0
```



i2c0 포트 확인

i2c0 번 포트는 다음과 같이 확인할 수 있습니다.

i2c0 포트 확인 (i2cdetect)

```
# i2cdetect -l

i2c-0 i2c Synopsys DesignWare I2C adapter I2C adapter
i2c-2 i2c Synopsys DesignWare I2C adapter I2C adapter
i2c-5 i2c Synopsys DesignWare I2C adapter I2C adapter
i2c-6 i2c Synopsys DesignWare I2C adapter I2C adapter
i2c-7 i2c Inno HDMI
```

i2c0 장치파일 확인

```
# ll /dev/i2c-*
crw-rw---- 1 root i2c 89, 0 Jun 10 2022 /dev/i2c-0
crw-rw---- 1 root i2c 89, 2 Jun 10 2022 /dev/i2c-2
crw-rw---- 1 root i2c 89, 5 Jun 10 2022 /dev/i2c-5
crw-rw---- 1 root i2c 89, 6 Jun 10 2022 /dev/i2c-6
crw-rw---- 1 root i2c 89, 7 Jun 10 2022 /dev/i2c-7
```

i2c1 포트 추가

i2c1 번 포트는 StarFive JH7110 커널에서 기본적으로 제공하지 않으므로, 다음과 같이 커널연구회에서 제공하는 디바이스 트리 overlay 을 적용하여 사용 가능합니다.

overlay 디바이스트리 적용 (root 권한에서 실행)

```
//커널연구회에서 제공한 실습 소스 경로로 이동
# cd riscv-jh7110-io/dtb/overlay/

# ll

-rwxr-xr-x 1 user user 275 Oct 10 19:51 run.sh
-rwxr-xr-x 1 root root 1075 Oct 10 19:50 vf2-overlay-i2c1.dtbo
-rwxr-xr-x 1 root root 866 Oct 10 19:50 vf2-overlay-uart3.dtbo

//run.sh 스크립트를 실행하여 overlay 디바이스 트리 적용
# ./run.sh
```

i2c1 포트 확인 (i2cdetect)

```
# i2cdetect -l

i2c-0 i2c      Synopsys DesignWare I2C adapter      I2C adapter
i2c-1 i2c      Synopsys DesignWare I2C adapter      I2C adapter
i2c-2 i2c      Synopsys DesignWare I2C adapter      I2C adapter
i2c-5 i2c      Synopsys DesignWare I2C adapter      I2C adapter
i2c-6 i2c      Synopsys DesignWare I2C adapter      I2C adapter
i2c-7 i2c      Inno HDMI
```

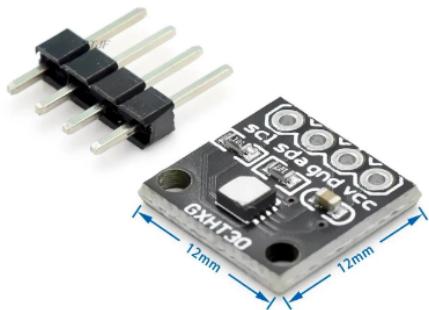
i2c1 장치파일 확인

```
# ll /dev/i2c-*
crw-rw---- 1 root i2c 89, 0 Jun 10 2022 /dev/i2c-0
crw-rw---- 1 root i2c 89, 1 Oct 8 23:32 /dev/i2c-1
crw-rw---- 1 root i2c 89, 2 Jun 10 2022 /dev/i2c-2
crw-rw---- 1 root i2c 89, 5 Jun 10 2022 /dev/i2c-5
crw-rw---- 1 root i2c 89, 6 Jun 10 2022 /dev/i2c-6
crw-rw---- 1 root i2c 89, 7 Jun 10 2022 /dev/i2c-7
```

I2C 온도 습도 센서(HT30) 사용 방법

데이터 시트는 인터넷에서 다운로드 가능:

<http://www1.futureelectronics.com/doc/Sensirion/SHT30-DIS-P2.5KS.pdf>



온도 습도 센서(HT30) 정보

Datasheet SHT3x-DIS

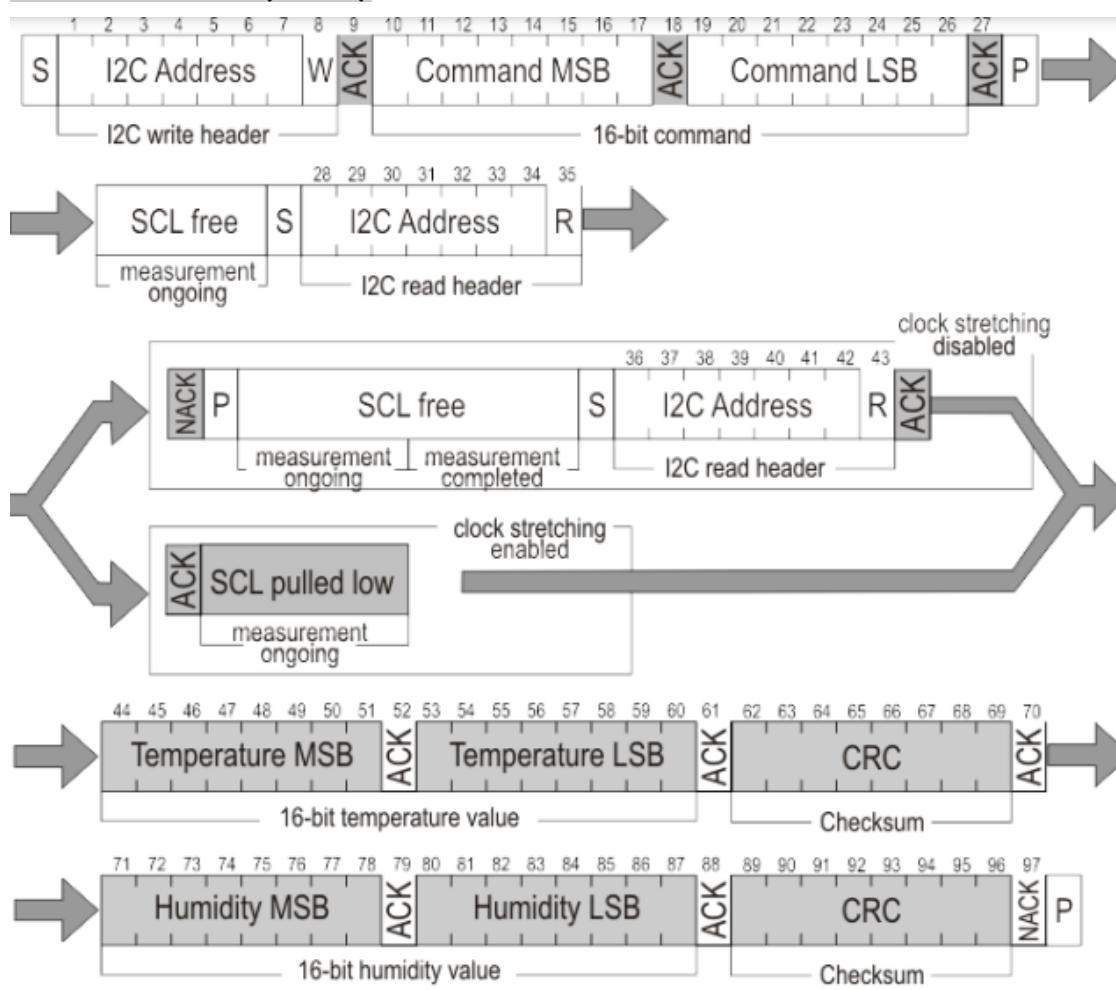
Humidity and Temperature Sensor

- Fully calibrated, linearized, and temperature compensated digital output
- Wide supply voltage range, from 2.15 V to 5.5 V
- I2C Interface with communication speeds up to 1 MHz and two user selectable addresses
- Typical accuracy of $\pm 1.5\% \text{RH}$ and $\pm 0.1^\circ \text{C}$ for SHT35
- Very fast start-up and measurement time
- Tiny 8-Pin DFN package



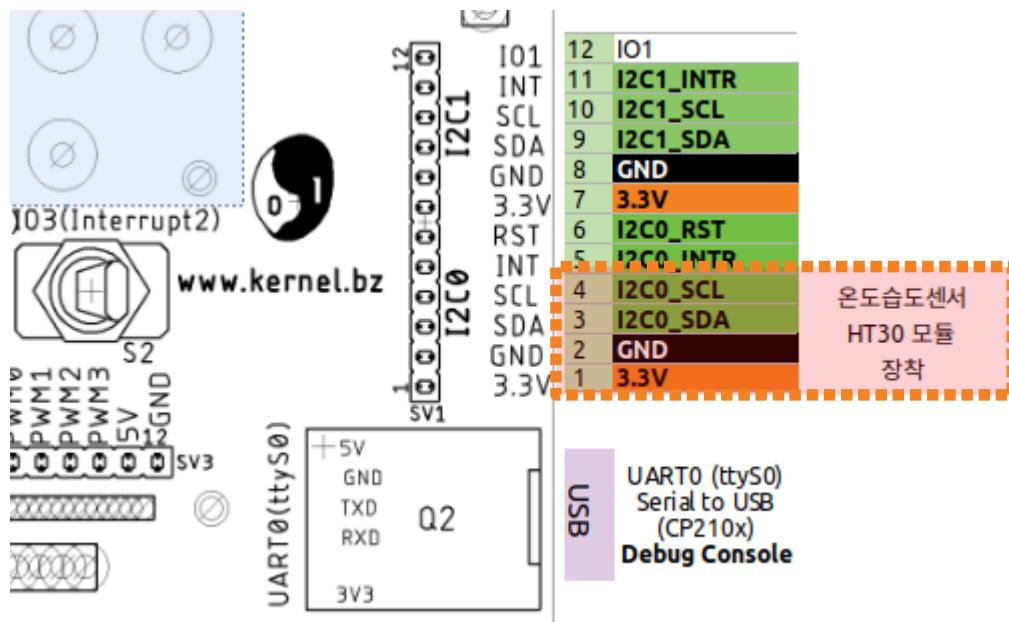
입출력 장치 활용 방법

- I2C 인터페이스 동작 (**Slave Address: 0x44**)
- 동작전압: 2.15V ~ 5.5V
- 온도값 범위(2 바이트): 1 ~ 65535
- 습도값 범위(2 바이트): 1 ~ 65535
- 데이터 응답 대기시간: 9ms

Slave Address(0x44)

온도 습도 센서(HT30) 데이터 읽기

온도습도 센서(HT30) 모듈(별도구매)을 다음과 같이 1 번(3.3V), 2 번(GND), 3 번(SDA), 4 번(SCL) 핀에 연결 합니다.



커널연구회에서 제공하는 I2C 테스트 프로그램 소스를 다음과 같이 실행 합니다.

i2c 테스트 프로그램 실행:

- 첫번째 파라미터: i2c 포트 번호
- 두번째 파라미터: i2c 장치 slave 주소 (조도센서 HT30 은 0x44, 10 진수 68)
- 세번째 파라미터: i2c 데이터 읽기 반복(loop) 회수

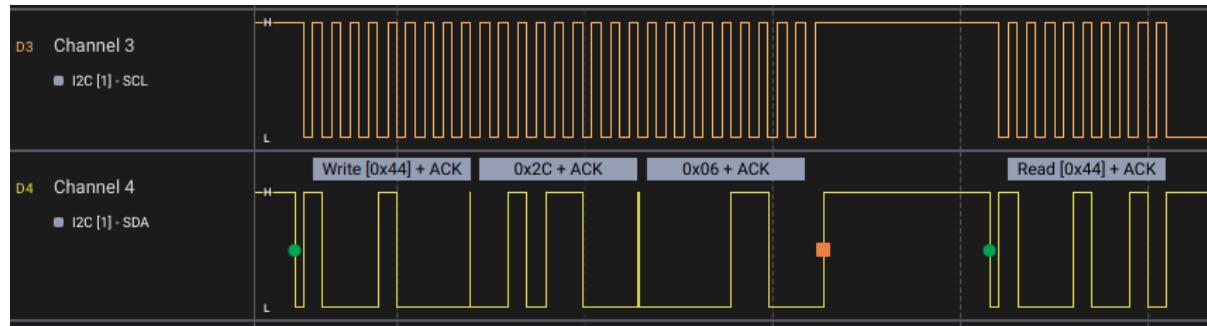
입출력 장치 활용 방법

i2c에서 온도 습도 센서(HT30) 읽기 (root 권한에서 실행)

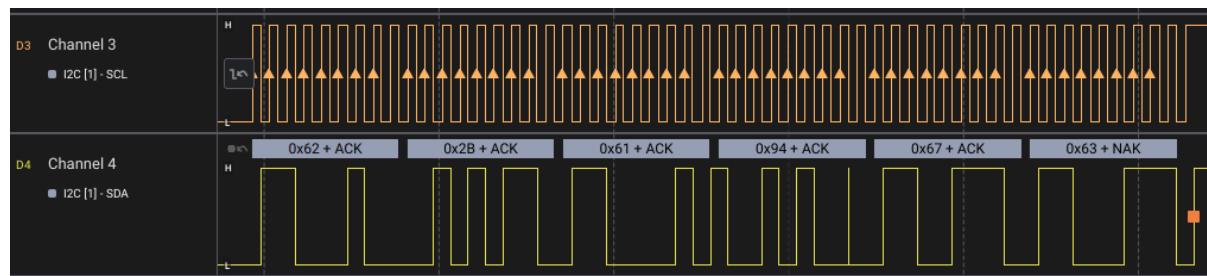
```
# ./i2c 0 68 10
```

```
>Reading Data Count <6>: [62] [2B] [61] [94] [67] [63] : Temperature: 25.13, Humidity: 37.99
>Reading Data Count <6>: [62] [3E] [D7] [94] [27] [5E] : Temperature: 25.15, Humidity: 37.93
>Reading Data Count <6>: [62] [33] [9B] [94] [CF] [26] : Temperature: 25.14, Humidity: 38.10
>Reading Data Count <6>: [62] [33] [9B] [94] [5C] [4C] : Temperature: 25.14, Humidity: 37.98
>Reading Data Count <6>: [62] [43] [63] [94] [56] [97] : Temperature: 25.16, Humidity: 37.97
>Reading Data Count <6>: [62] [3B] [22] [94] [10] [0C] : Temperature: 25.15, Humidity: 37.90
>Reading Data Count <6>: [62] [33] [9B] [94] [8A] [EE] : Temperature: 25.14, Humidity: 38.03
>Reading Data Count <6>: [62] [43] [63] [94] [10] [0C] : Temperature: 25.16, Humidity: 37.90
>Reading Data Count <6>: [62] [33] [9B] [94] [2D] [85] : Temperature: 25.14, Humidity: 37.93
>Reading Data Count <6>: [62] [38] [71] [94] [5C] [4C] : Temperature: 25.14, Humidity: 37.98
```

i2c 디지털 로직 확인(0x44 주소에 명령어 쓰기)



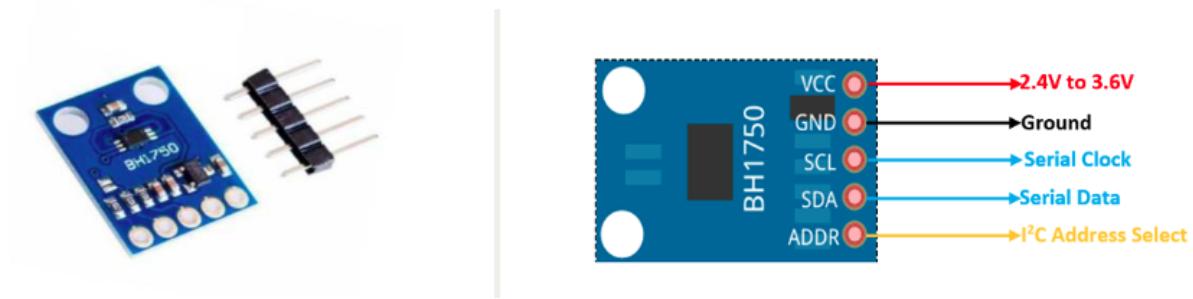
i2c 디지털 로직 확인(0x44 주소에서 온도 습도 데이터 읽기)



I2C 조도센서(BH1750) 사용 방법

데이터 시트는 인터넷에서 다운로드 가능:

https://components101.com/sites/default/files/component_datasheet/BH1750.pdf

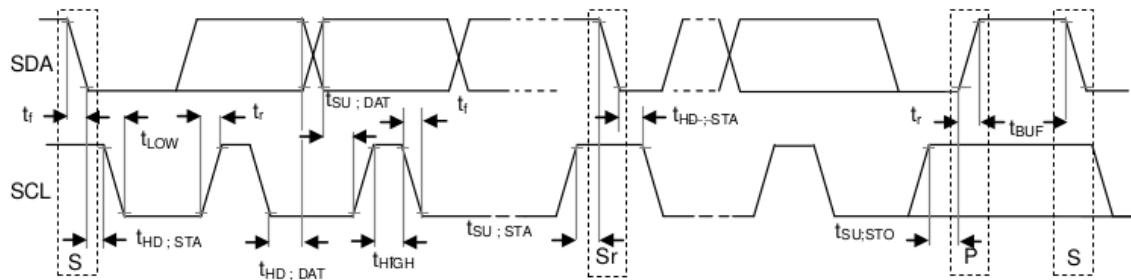


- I2C 인터페이스 동작(Slave Address: 0x23)
- 동작전압: 2.4V ~ 3.6V
- 조도값 범위(2 바이트): 1 ~ 65535

조도센서(BH1750) 정보

Slave Address(0x23)

입출력 장치 활용 방법



2) Write Format

BH1721FVC is not able to accept plural command without stop condition. Please insert SP every 1 Opecode.

ST	Slave Address "0100011"	R/W 0	Ack	Opecode	Ack	SP
----	----------------------------	----------	-----	---------	-----	----

3) Read Format

ST	Slave Address "0100011"	R/W 1	Ack	$2^{15} \ 2^{14} \ 2^{13} \ 2^{12} \ 2^{11} \ 2^{10} \ 2^9 \ 2^8$	High Byte [15:8]	Ack	SP
Low Byte [7:0] $2^7 \ 2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0$							

from Master to Slave from Slave to Master

조도센서(BH1750) 데이터 읽기

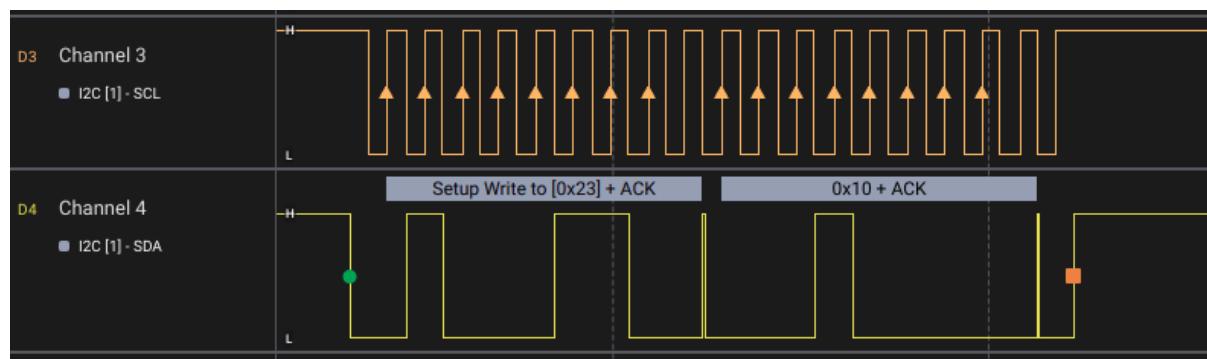
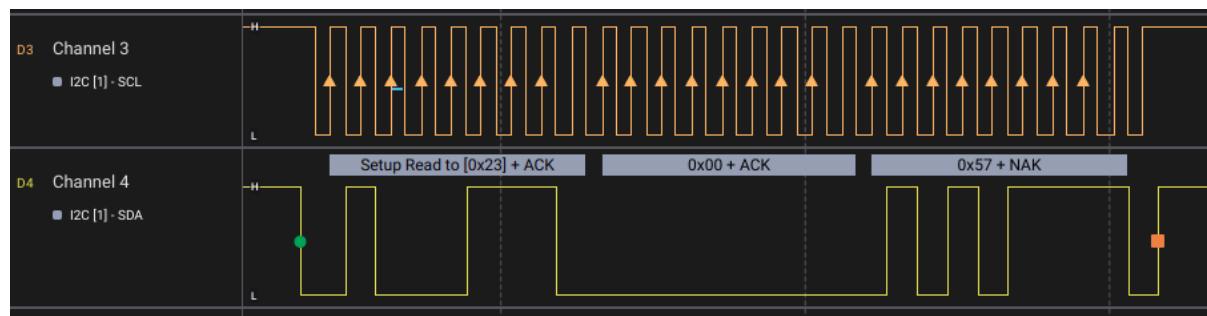
i2c 테스트 프로그램 실행:

- 첫번째 파라미터: i2c 포트 번호
- 두번째 파라미터: i2c 장치 slave 주소 (조도센서 BH1750 은 0x23, 10 진수 35)
- 세번째 파라미터: i2c 데이터 읽기 반복(loop) 회수

i2c에서 조도센서(BH1750) 읽기 (root 권한에서 실행)

```
# ./i2c 0 35 5

>Reading Data Count <2>: [00] [57] : Value: 87
>Reading Data Count <2>: [00] [5C] : Value: 92
>Reading Data Count <2>: [00] [5B] : Value: 91
>Reading Data Count <2>: [00] [56] : Value: 86
>Reading Data Count <2>: [00] [62] : Value: 98
```

i2c 디지털 로직 확인(0x23 주소에 명령어 쓰기)**i2c 디지털 로직 확인(0x23 주소에서 조도 데이터 읽기)**

PWM 활용 방법

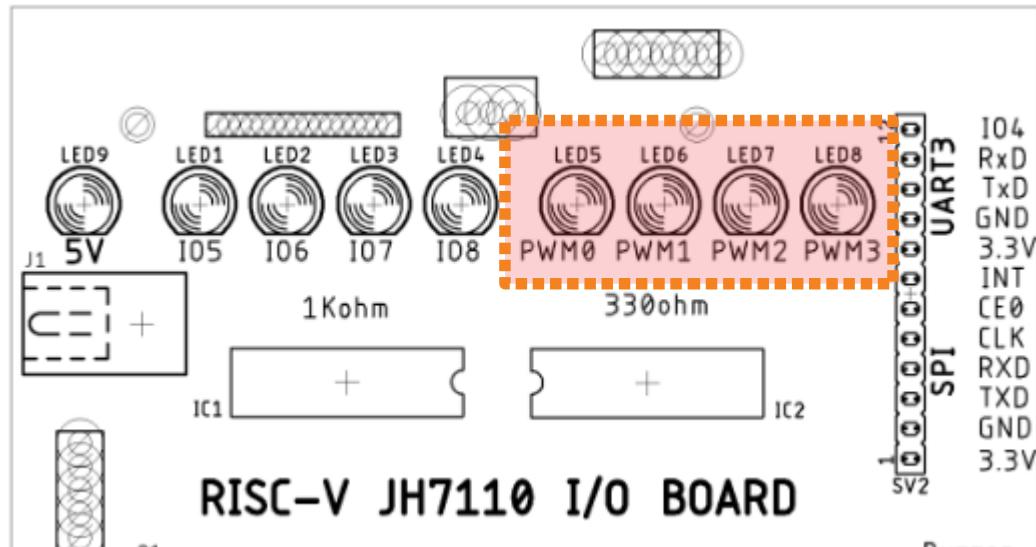
PWM0, PWM1은 StarFive JH7110 커널에서 기본적으로 제공하고, PWM2와 PWM3은 커널연구회에서 디바이스 트리를 추가 하였습니다.

PWM 출력 기호	PWM 제어 파일	GPIO 핀번호	헤더 40 핀 번호
PWM0 (LED5)	/sys/class/pwm/pwmchip0/pwm0	GPIO46	32
PWM1 (LED6)	/sys/class/pwm/pwmchip0/pwm1	GPIO59	33
PWM2 (LED7)	/sys/class/pwm/pwmchip0/pwm2	GPIO36	36
PWM3 (LED8)	/sys/class/pwm/pwmchip0/pwm3	GPIO39	31

실습용 보드 핀맵 위치

device	function	GPIO	pin number	GPIO	function	device
LED4	IO8	44	40	39	GND	
LED3	IO7	61	38	37	UART3 (TxD)	
LED8	PWM2	36	36	35	UART3 (RXD)	ttyS3
	GND		34	33	PWM1	LED6
LED5	PWM0	46	32	31	PWM3	LED7
	GND		30	29	I2C1 (INTR)	
I2C1	I2C1 (SCL)	40	28	27	I2C1 (SDA)	I2C1
SPI	SPI (INTR)	56	26	25	GND	
	SPI (CEO)	49	24	23	SPI (SCLK)	
LED2	IO6	50	22	21	SPI (MISO) RxD	SPI
	GND		20	19	SPI (MOSI) TxD	
LED1	IO5	51	18	17	3.3V	
Buzzer	IO4	54	16	15	IO3	Switch2
	GND		14	13	IO2	Switch1
	IO1	38	12	11	I2C0 (REST)	
ttyS0	UART0 (RxT)	6	10	9	GND	
	UART0 (TxT)	5	8	7	I2C0 (INTR)	I2C0
	GND		6	5	I2C0 (SCL)	
	5V		4	3	I2C0 (SDA)	
	5V		2	1	3.3V	

실습용 보드 위치(PWM 출력 LED)



PWM 출력 테스트

PWM은 StarFive JH7110 커널에서 기본적으로 제공하는 /sys/class/pwm 파일에서 다음과 같이 테스트 할 수 있습니다.

GPIO46 (PWM0) 제어 파일 생성

```
# cd /sys/class/pwm/pwmchip0
# echo 0 > export
# cd pwm0
# ls -al

-r--r--r-- 1 root root 4096 Sep 11 02:40 capture
-rw-r--r-- 1 root root 4096 Sep 11 02:40 duty_cycle
-rw-r--r-- 1 root root 4096 Sep 11 02:40 enable
-rw-r--r-- 1 root root 4096 Sep 11 02:40 period
```

입출력 장치 활용 방법

```
-rw-r--r-- 1 root root 4096 Sep 11 02:40 polarity
drwxr-xr-x 2 root root     0 Sep 11 02:40 power
-rw-r--r-- 1 root root 4096 Sep 11 02:40 ueven
```

PWM0 제어 데이터 출력

```
// pwm 주기(주파수)를 1ms로 설정
# echo 1000000 > period

// Duty Cycle을 50%로 설정
# echo 500000 > duty_cycle

// 동작 시작
# echo 1 > enable
```

PWM 제어용 소스 활용

커널연구회에서 제공한 PWM 테스트 프로그램을 다음과 같이 활용할 수 있습니다

PWM 테스트 프로그램 실행(root 권한에서 실행)

```
// 커널연구회에서 제공한 실습 소스 경로로 이동
# cd riscv-jh7110-io/app/pwm/

// 첫번째 파라미터는 테스트 반복(loop) 회수, 반복회수는 50 이상 충분히 크게 입력
// 입력하지 않으면 무한 loop로 실행
# ./pwm-test 50

_pwm_export:: pwm:0, cnt:1
_pwm_export:: pwm:1, cnt:1
_pwm_export:: pwm:2, cnt:1
_pwm_export:: pwm:3, cnt:1
pwm loop count:<50>, step:<1>
```

입출력 장치 활용 방법

```
_pwm_unexport:: pwm:0, cnt:1
_pwm_unexport:: pwm:1, cnt:1
_pwm_unexport:: pwm:2, cnt:1
_pwm_unexport:: pwm:3, cnt:1
```

위와 같이 실행하면 LED5, 6, 7, 8 가 1 초 간격으로 번갈아 가면서 밝기 변화면서 깜박입니다. 자세한 내용은 커널연구회에서 제공한 pwm 소스를 확인하여 분석할 수 있습니다.

모터 제어 PWM

아래는 모터를 제어하기 위한 PWM 신호를 분석한 것입니다. (모터 제어시 참조)

Duty 비율: 10% (0.1ms) / PWM 주기 1ms



Duty 비율: 60% (0.6ms) / PWM 주기 1ms

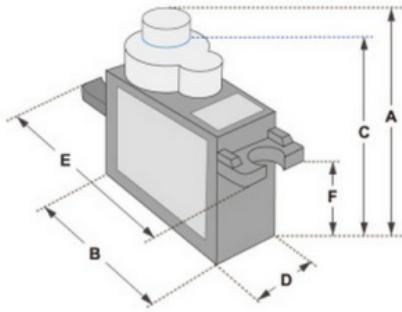


입출력 장치 활용 방법

Duty 비율: 90% (0.9ms) / PWM 주기| 1ms



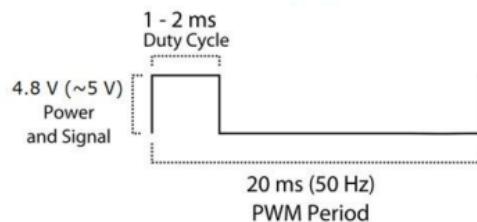
서보 모터 PWM



Dimensions & Specifications	
A (mm)	: 32
B (mm)	: 23
C (mm)	: 28.5
D (mm)	: 12
E (mm)	: 32
F (mm)	: 19.5
Speed (sec)	: 0.1
Torque (kg-cm)	: 2.5
Weight (g)	: 14.7
Voltage	: 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.

PWM=Orange (☱☱)
Vcc = Red (+)
Ground=Brown (-)



- left: 1ms/20ms (0.05)
- middle: 1.5ms/20ms (0.075)
- right: 2ms/20ms (0.1)

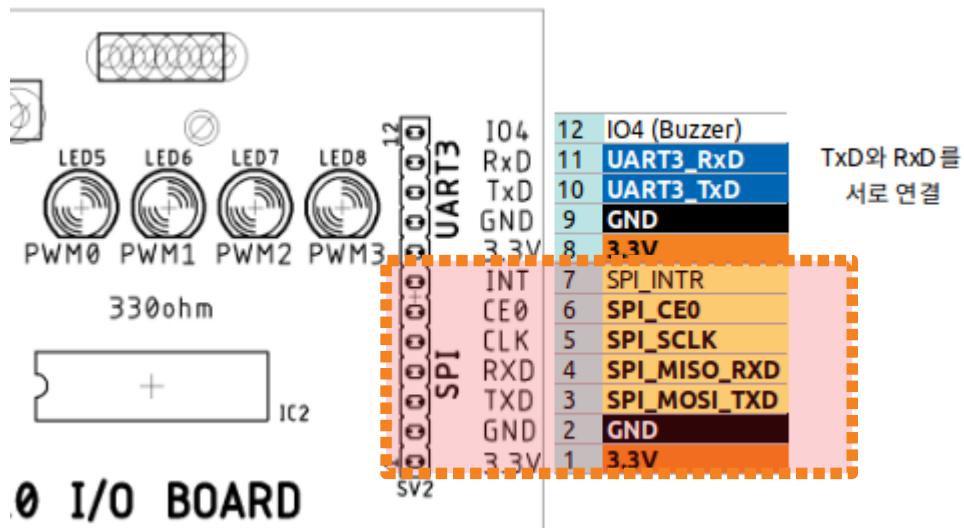
SPI 활용 방법

SPI는 StarFive JH7110 커널에서는 disable 되어 있습니다. 커널연구회에서 디바이스 트리를 수정하여 "/dev/spidev1.0" 장치 노드를 동작 가능하도록 했습니다. 따라서 SPI를 사용하기 위해서는 커널연구회에서 배포하는 커널을 포팅 해야 합니다. (좀더 자세한 내용은 커널연구회에서 진행하는 교육을 통하여 학습 가능)

실습용 보드 핀맵 위치

device	function	GPIO	pin number	GPIO	function	device
LED4	IO8	44	40	39	GND	
LED3	IO7	61	38	37	UART3 (TxD)	
LED8	PWM2	36	36	35	UART3 (RxD)	ttyS3
	GND		34	33	PWM1	LED6
LED5	PWM0	46	32	31	PWM3	LED7
	GND		30	29	I2C1 (INTR)	
	I2C1 (SDA)	40	20	27	I2C1 (SCL)	I2C1
SPI	SPI (INTR)	56	26	25	GND	
	SPI (CEO)	49	24	23	SPI (SCLK)	
LED2	IO6	50	22	21	SPI (MISO) RxD	SPI
	GND		20	19	SPI (MOSI) TxD	
LED1	IO5	51	18	17	3.3V	
Buzzer	IO4	54	16	15	IO3	Switch2
	GND		14	13	IO2	Switch1
	IO1	38	12	11	I2C0 (REST)	
ttyS0	UART0 (RxD)	6	10	9	GND	
	UART0 (TxD)	5	8	7	I2C0 (INTR)	I2C0
	GND		6	5	I2C0 (SCL)	
	5V		4	3	I2C0 (SDA)	
	5V		2	1	3.3V	

실습용 보드 위치



I/O BOARD

SPI 제어용 소스 활용

커널연구회에서 제공한 SPI 테스트 프로그램을 다음과 같이 활용할 수 있습니다

SPI 장치파일 확인

```
$ ll /dev/spi*
crw----- 1 root root 153, 0 Jun 10 2022 /dev/spidev1.0
```

SPI 테스트 프로그램 실행(root 권한에서 실행)

```
//커널연구회에서 제공한 실습 소스 경로로 이동
# cd riscv-jh7110-io/app/spi/

//spi 실행 (첫번째 파라미터는 테스트 반복회수)
# ./spi 5

spi_config_get:: mode:[0], bits:<8>, speed:<500000>Hz, <500>KHz

spi_recv_thread:<2338926928>, fd:<3>
spi_recv_thread:: fd:<3>
>TX | 11 33 77 FF 11 33 77 FF  | .3w..3w. |
>RX | 01 01 01 01 01 01 01  | .....
spi_recv_thread:: read length:<8>
<<RX | 01 01 01 01 01 01 01 01  | .....

>TX | 11 33 77 FF 11 33 77 FF  | .3w..3w. |
>RX | 01 01 01 01 01 01 01  | .....
spi_recv_thread:: read length:<8>
<<RX | 01 03 07 0F 01 03 07 0F  | .....

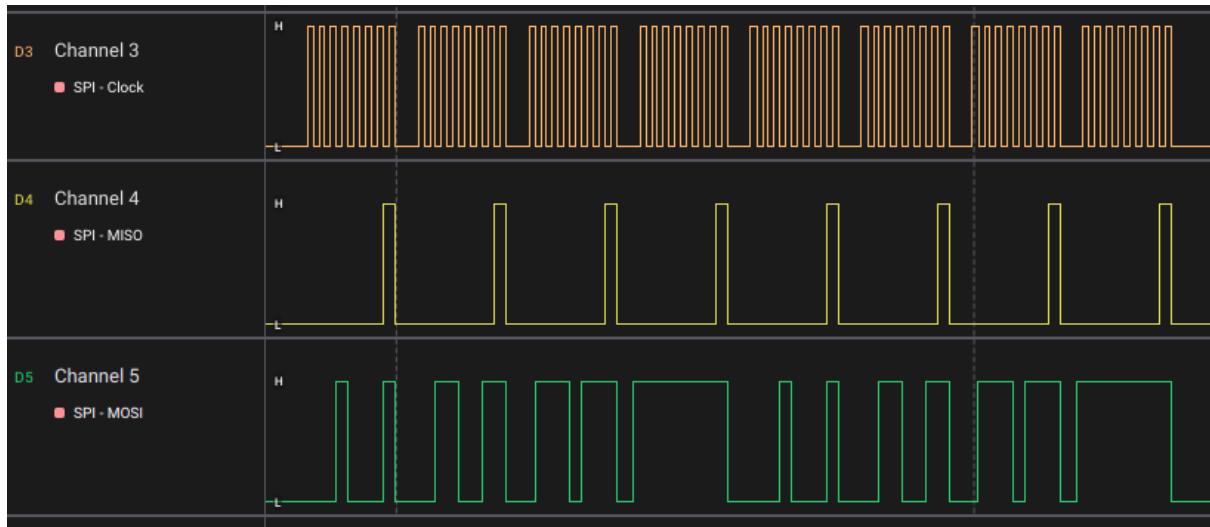
>TX | 11 33 77 FF 11 33 77 FF  | .3w..3w. |
>RX | 01 01 01 01 01 01 01  | .....
spi_recv_thread:: read length:<8>
<<RX | 01 03 07 0F 01 03 07 0F  | .....
```

./spi 을 실행할때 첫번째 파라미터에 전달한 회수 만큼 테스트를 반복 합니다.

MOSI(TxD) 데이터를 전송하면 MISO(RxD) 데이터를 수신 합니다.

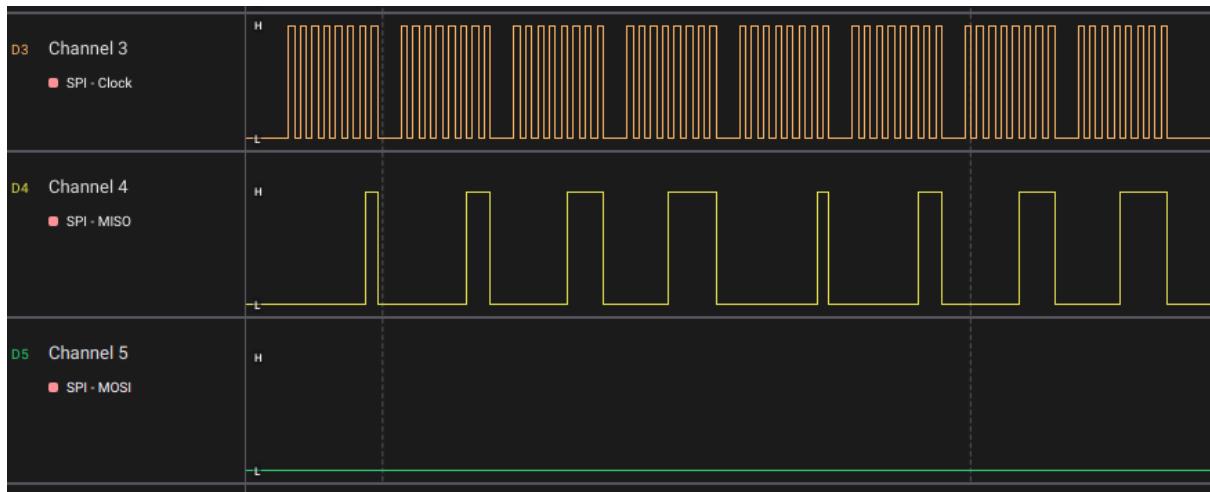
입출력 장치 활용 방법

SPI 디지털 로직 확인(전송, MOSI, TxD)



MOSI(TxD) 전송 데이터: 0x11, 0x22, 0x33, 0xFF, 0x11, 0x22, 0x33, 0xFF

SPI 디지털 로직 확인(수신, MISO, RxD)

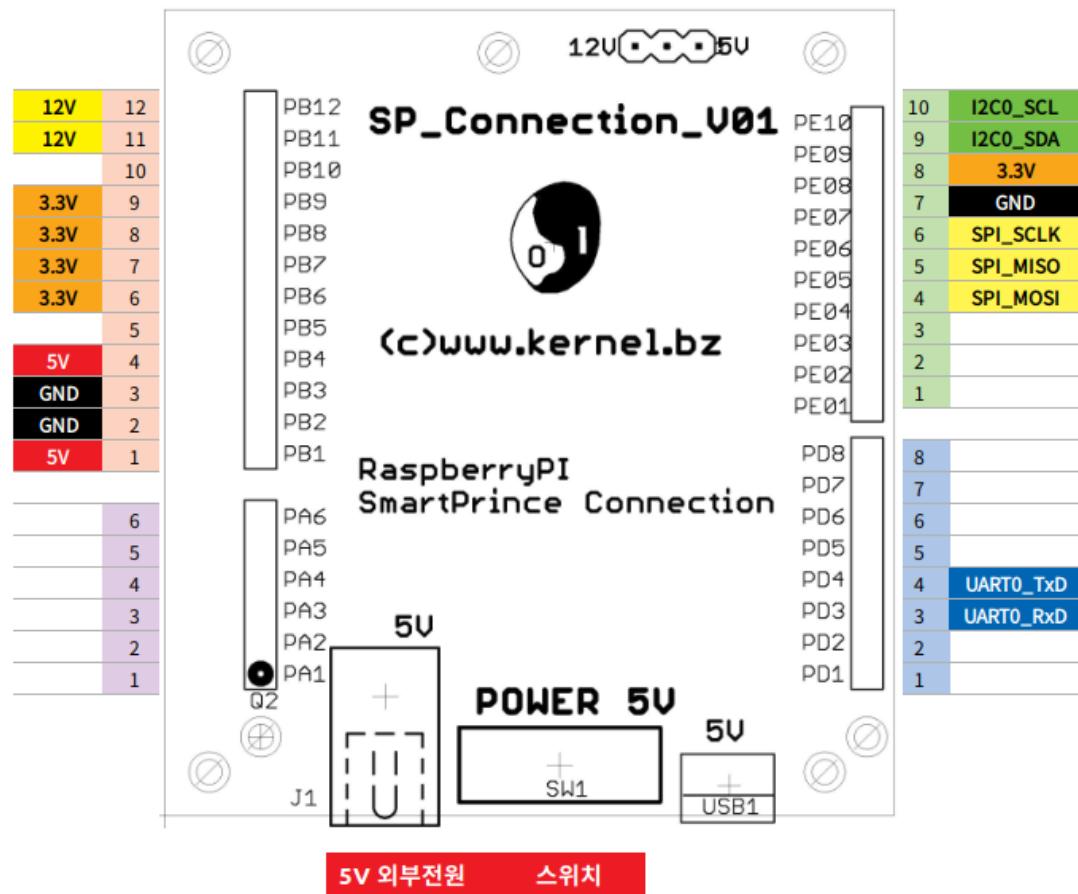


MISO(RxD) 수신 데이터: 0x01, 0x02, 0x03, 0x0F, 0x01, 0x02, 0x03, 0x0F

입출력 확장 가능

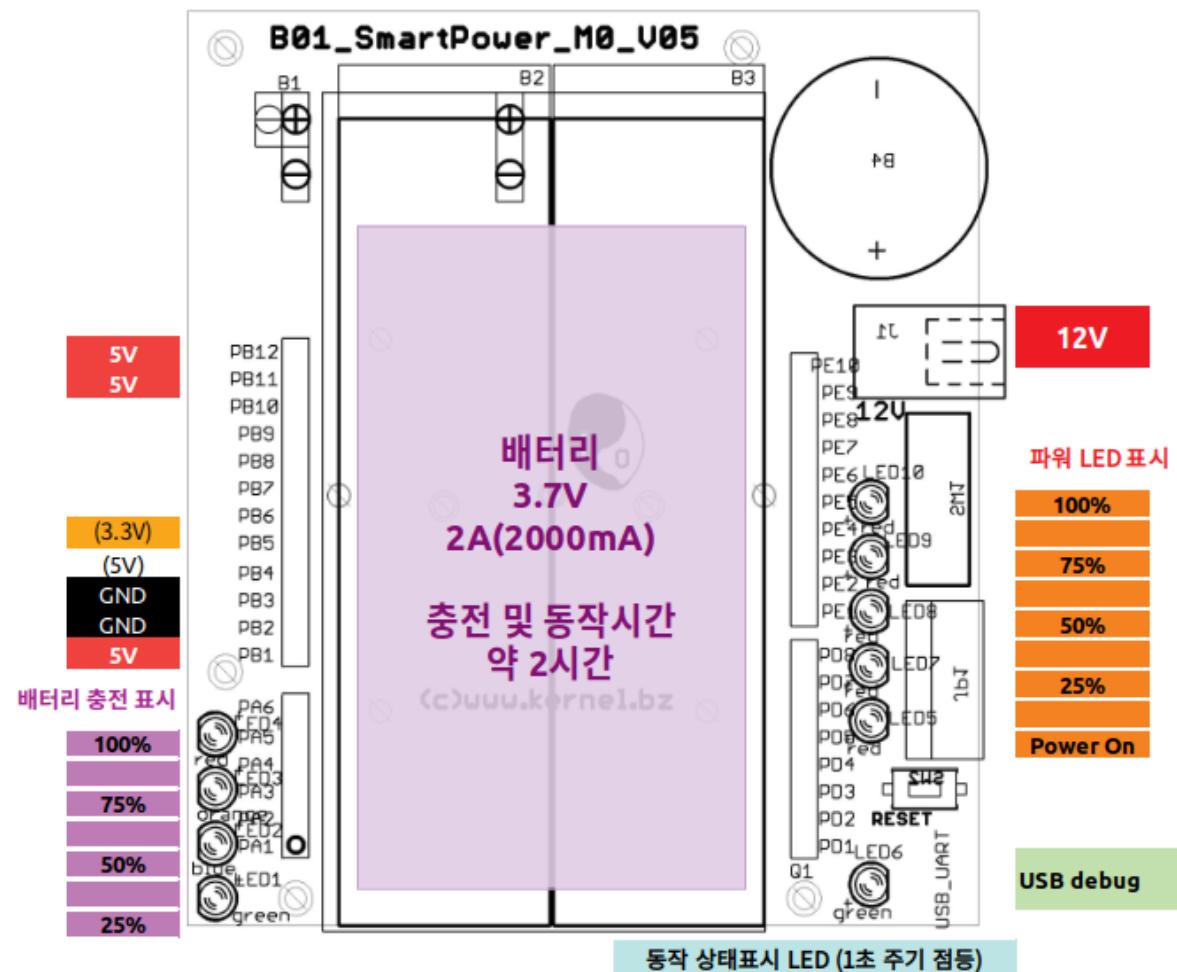
입출력 확장 기능은 RISC-V 입출력 보드의 40 핀맵에 세로 방향으로 적층하여 확장할 수 있도록 구성했습니다. 아래 내용들은 추가적인 보드 구매가 필요하며 커널연구회에서 진행하는 교육을 통하여 관련 소스와 함께 기능들을 자세히 학습할 수 있습니다.

입출력 적층 연결핀 구성도



배터리 Power 보드 연결 구성도

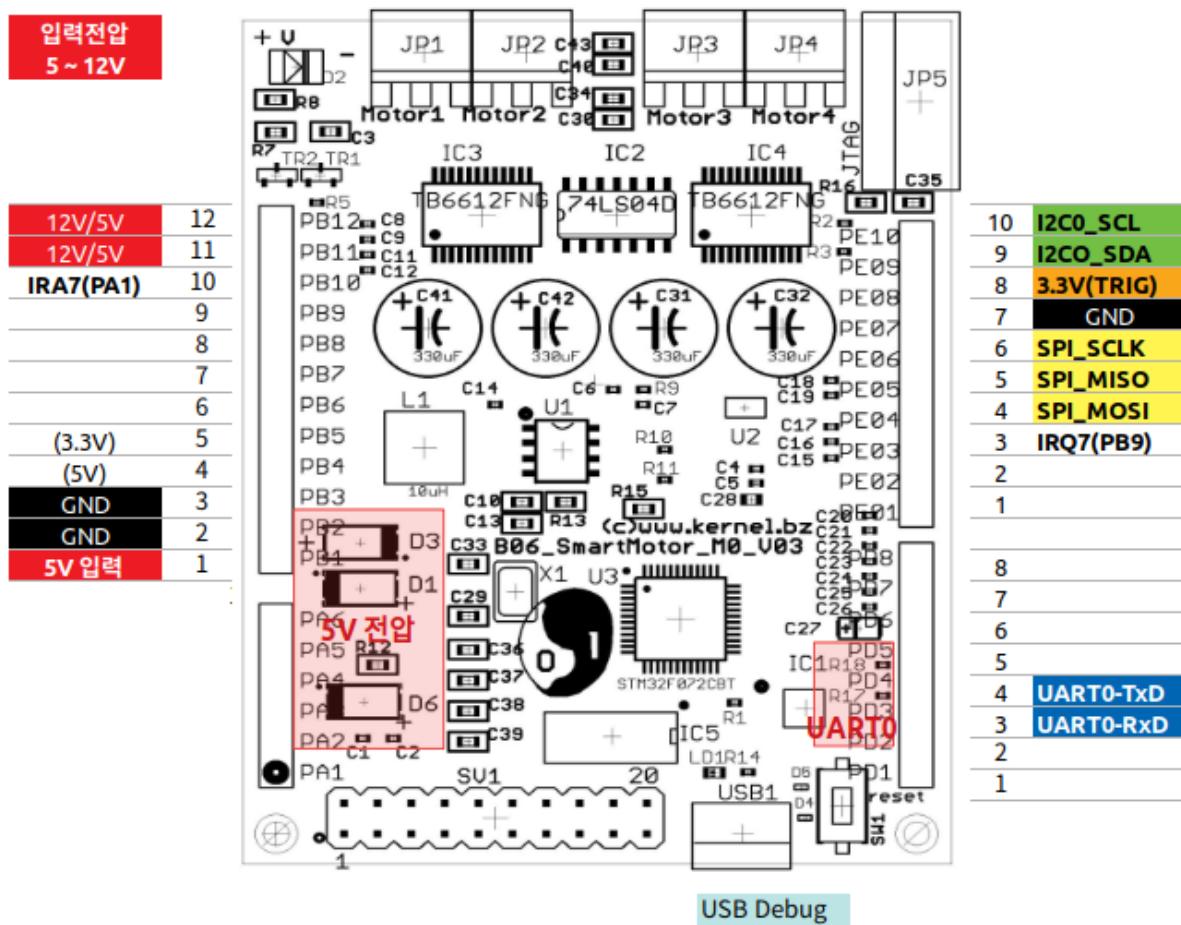
- Cortex-M0(STM32F0) MCU 내장
- 12V 외부 어댑터 전원공급(전류는 2A 이상)
- 배터리(3.7V 리튬폴리머) 충전기능
- 배터리 용량 2000mA
- 배터리 충전 상태 및 전류사용량 모니터링
- 보드 표면 온도 모니터링
- 배터리 충전 및 동작시간: 약 2 ~ 3 시간
- 리눅스 커널 모터제어 드라이버 모듈 제공



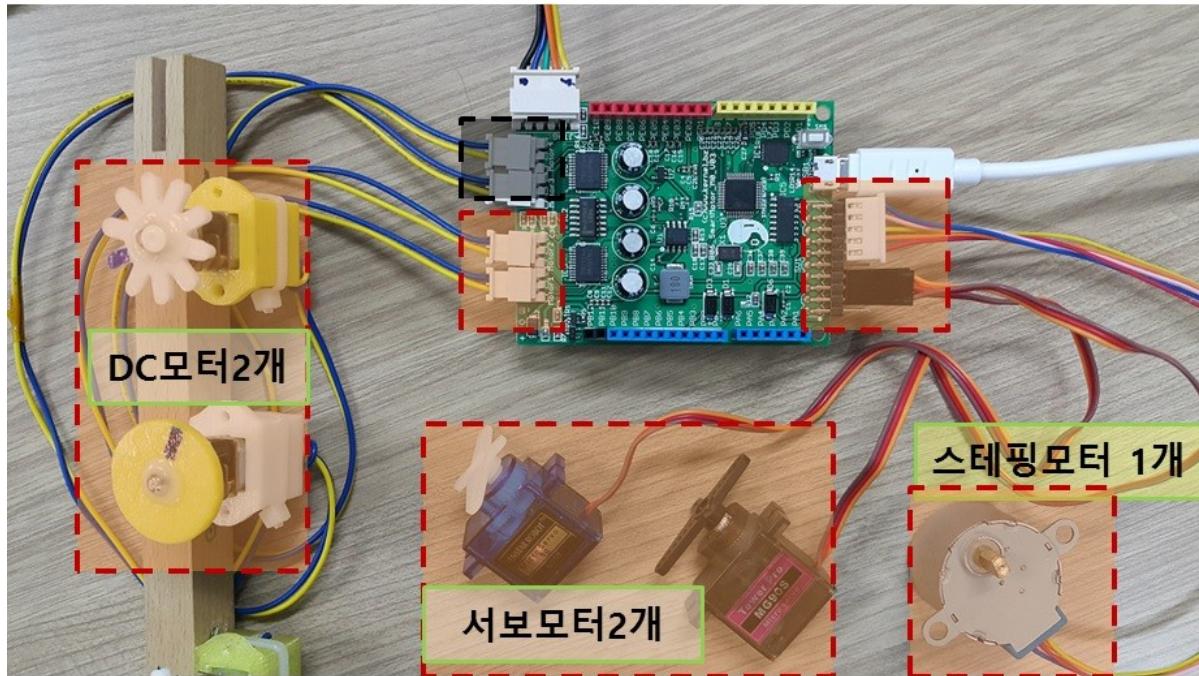
Motor 제어 보드 연결 구성도

- 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

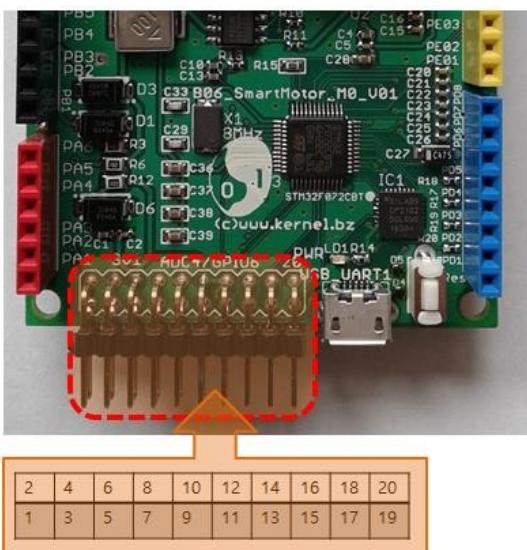
Motor 보드 핀맵



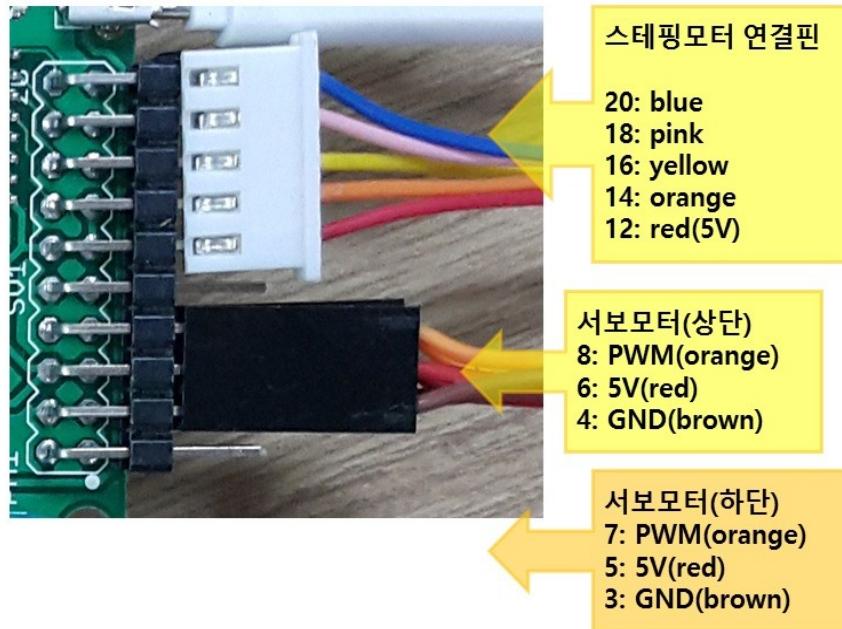
모터 연결 구성도



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, 스텝핑 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
```

참고로 index=0 을 전달하면 DC 모터와 서보모터, 스텝모터를 번갈아 가며 루프 테스트 합니다.

이때 running 은 테스트 회수가 됩니다. running=0 을 전달하면 무한 루프 테스트로 동작 합니다.

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]

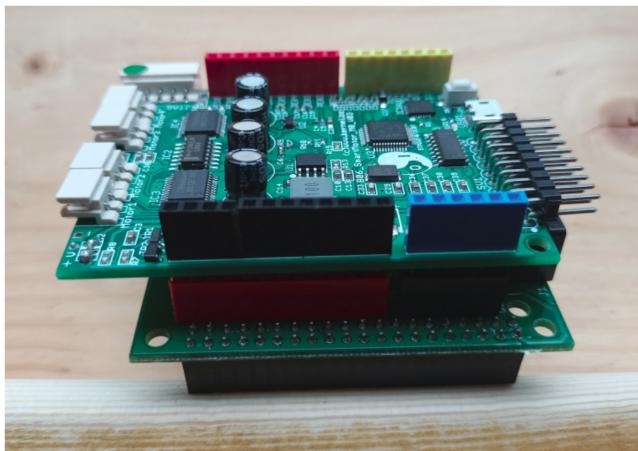
스테핑 모터 제어 명령(root 권한에서 실행)

```
//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
```

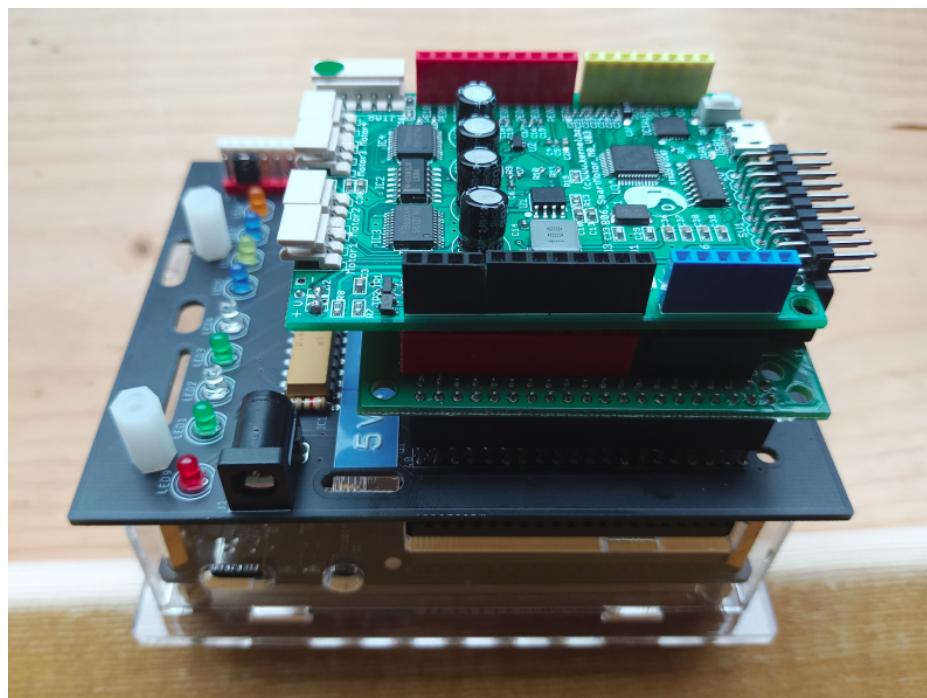
좀더 자세한 내용들은 커널연구회 홈페이지를 통하여 확인해 주시기 바랍니다.

확장 보드 조합 구성

모터 제어 보드



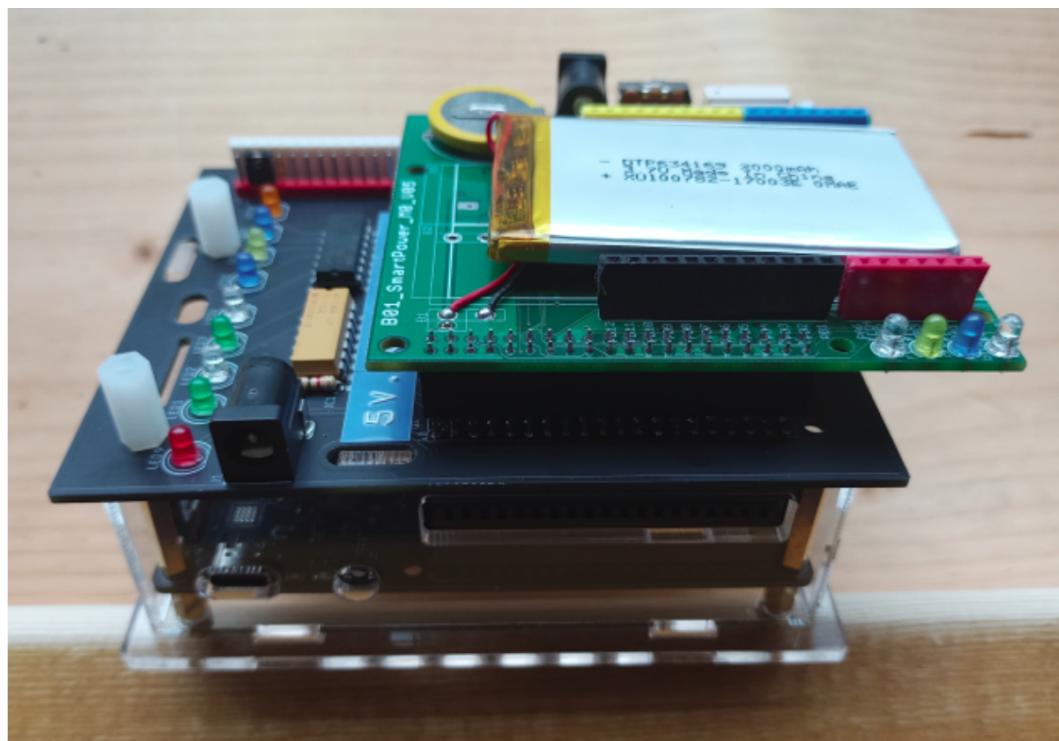
모터 제어 + RISCV 입출력 보드



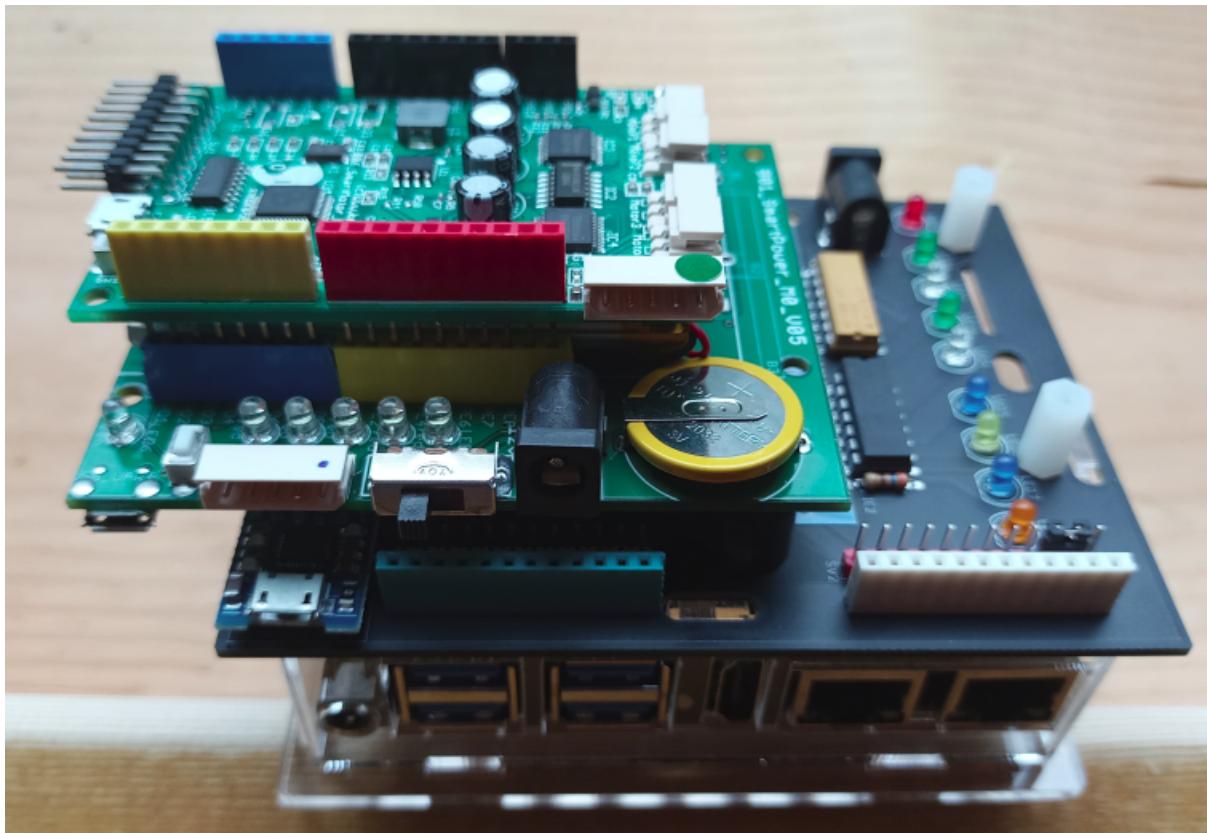
배터리 파워 보드



배터리 파워 + RISCV 입출력 보드



배터리 파워 + 모터 제어 + RISCV 입출력 보드



좀더 자세한 내용들은 커널연구회 홈페이지를 통하여 확인해 주시기 바랍니다.

감사합니다.