12

2024년 11월 15일 금요일 오전 8:52

브레이크아웃 보드를 이용한 프로토타이핑

맞춤형 보드 브링업 (custom board bring-up)이란?

새로운 하드웨어 보드를 처음부터 끝까지 설정하여, 제대로 동작하도록 만드는 전체 과정

(임베디드 리눅스 엔지니어가 몇 번이고 반복적으로 수행해야 하는 작업)

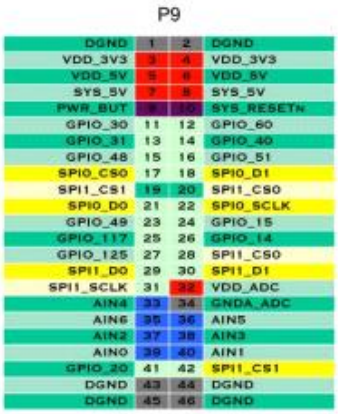
장치 트리의 소스에 회로도 매핑하기

<회로도와 데이터 시트 읽기>

비글본 블랙에는 I/O용 2 x 46핀 확장 헤더 존재 (GPIO 외에도 UART, I2C, SPI 통신 포트 포함)

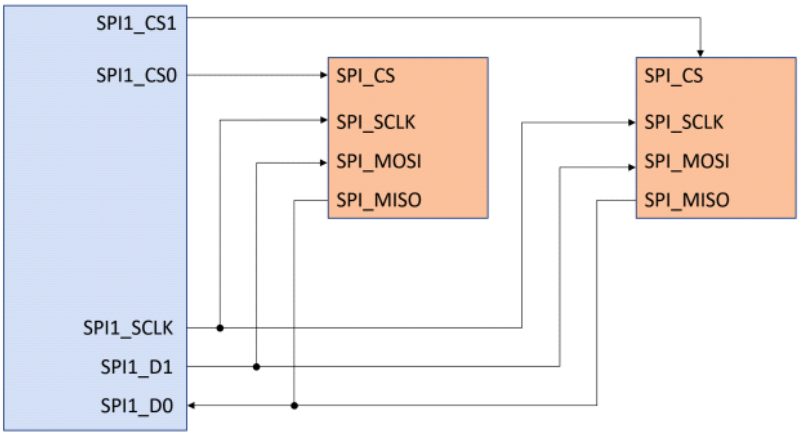
특징	UART (Universal Asynchronous Receiver-Transmitter)	SPI (Serial Peripheral Interface)	I2C (Inter-Intergrated Circuit)
통신 방식	비동기식 (Asynchronous)	동기식 (Synchronous)	동기식 (Synchronous)
주요 용도	포인트 투 포인트(Point-to-Point) 통신 주로 컴퓨터와 외부 장치 간 통신	마스터-슬레이브(Master-Slave) 구조	여러 장치와의 멀티 마스터-슬레이브 통신
통신 방식	Half-Duplex	Full-Duplex	Half-Duplex
필요한 선 수	2선 (TX, RX)	4선 (MOSI, MISO, SCK, CS)	2선(SDA, SCL)
장점	- 간단한 하드웨어 설계 - 긴 거리 전송 가능 - 저속 통신에 적합	- 빠른 데이터 전송 - 고속 통신에 적합 - 여러 장치 연결 가능	- 여러 장치 연결 가능 - 적은 선 수로 여러 장치 연결
단점	-상대적으로 낮은 속도 -한번에 2개의 장치만 통신 가능	-많은 선 필요 -여러 장치 연결 시 핀 수 증가	-상대적으로 낮은 속도 -복잡한 주소 관리 필요

(P9 확장 헤더 SPI 포트)

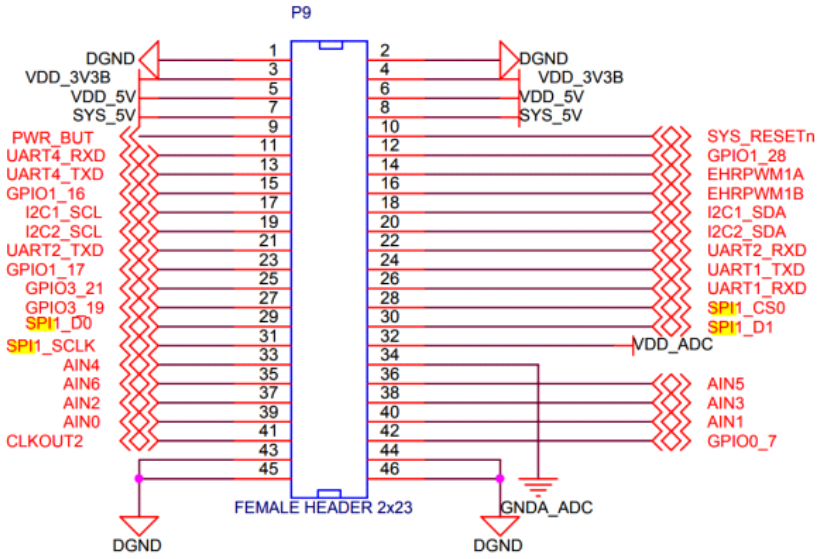


- **CS(Chip Select)** 신호는 여러 개의 주변 장치가 연결된 버스에서 특정 장치를 선택할 때 사용
- CS 신호라인을 로우로 설정하면 버스에 전송할 주변 기기 선택됨, '액티브 로우 (active low)'라고 함

(2개의 주변 기기에 연결된 비글본 블랙의 SPI1 버스 블록 다이어그램)



(비글본 블랙 회로도 - P9 확장 헤더의 4개(28~31)에 SPI1 레이블에 지정돼 있음을 알 수 있음)



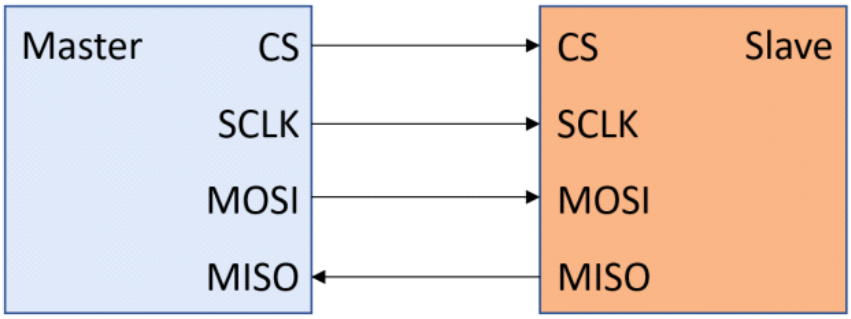
- SPI1 핀(19, 20, 42)과 SPI0 핀(17, 18, 21, 22)의 기본 용도가 I2C와 UART로 변경됨
  => 이 변경은 장치 트리 소스 파일의 핀 맵핑 구성에 의해 이루어짐
  누락된 SPI 신호 라인을 확장 헤더의 해당 핀으로 연결하려면, 적절한 핀 맵핑 구성 필요
  핀 맵핑은 프로토타이핑 단계에서 런타임에 변경할 수 있지만, 최종 하드웨어에서는 컴파일 타임에 설정해야 함

SPI의 4개 주요 신호 라인

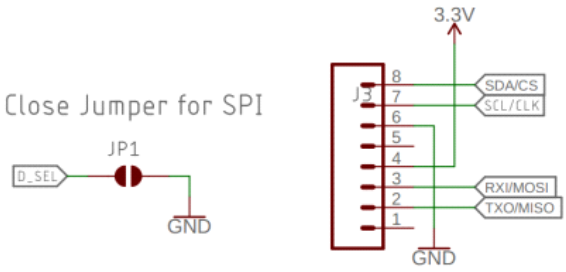
- **SCLK**: 클럭 신호 (마스터가 생성)
- **D0**: MISO (마스터 → 슬레이브)
- **D1**: MOSI (슬레이브 → 마스터)
- **CS0**: Chip Select (슬레이브 선택)

=> SPI는 전이중 인터페이스로 고속 데이터 전송을 지원

(4개의 SPI 신호 모두의 방향을 보여주는 블록 다이어그램)



(GPS 브레이크 아웃의 D_SEL 점퍼와 SPI 커넥터)



- **GPS 모듈(ZOE-M8Q)을 비글본 블랙에 연결**

  - ZOE-M8Q의 **SPI** 핀은 기본적으로 **비활성화** 상태
    => D_SEL 핀을 **GND에 연결**하고, **JP1 점퍼를 닫으면** SPI가 활성화

  - CS, CLK, MOSI, MISO 핀을 3.3V와 GND와 함께 배치하고, 2.2kΩ 풀업 저항 사용
  - 비글본 블랙의 P9 확장 헤더에서 3.3V와 GND를 연결하고, GPS 모듈의 SPI 라인을 해당 핀에 연결
  - 리눅스에서 SPI 버스 활성화

<비글본 블랙에 데비안 설치하기>

- BegleBorad.org 는 다양한 개발 보드용 데비안 이미지를 제공

(비글본용 Debian Busster IoT 마이크로 SD 카드 이미지를 다운로드하는 명령어)

```
$ wget https://debian.BeagleBoard.org/images/bone-debian-10.3-iot-armhf-
2020-04-06-4gb.img.xz
```

- **마이크로SD 카드에 기록하고 부팅하기**

  1. Etcher를 사용해 BeagleBoard.org에서 다운로드한 bone-debian-10.3-iot-armhf-2020-04-06-4gb.img.xz 파일을 마이크로SD 카드에 기록
  2. 마이크로SD 카드를 비글본 블랙에 삽입하고, 5V 전원 공급 장치로 전원 켬
  3. 이더넷 케이블을 비글본 블랙에 연결하고, 이를 라우터의 빈 포트에 연결
     (온보드 이더넷 표시등이 깜박이면 비글본 블랙이 온라인 상태)

- 인터넷 액세스를 통해 데비안 내에서 깃 저장소로 부터 패키지를 설치하고 코드를 가져올 수 있음

(리눅스 호스트에서 비글본 블랙으로 ssh 하는 명령어)

```
$ ssh debian@beaglebone.local
```

debian 사용자의 암호 프롬프트에서 tmppwd를 입력

<spidev 활성화 하기>

- 리눅스는 SPI 장치에 대해 read()와 write()를 지원하는 사용자 공간 API인 spidev 제공
- 이 API는 비글본 블랙용 데비안 Buster 이미지에 포함되어 있으며, spidev 커널 모듈을 통해 확인

(spidev 커널 모듈 검색)

```
debian@beaglebone:~$ lsmod | grep spi
spidev                 20480  0
```

(사용 가능한 SPI 주변 기기 주소)

```
$ ls /dev/spidev*
/dev/spidev0.0  /dev/spidev0.1  /dev/spidev1.0  /dev/spidev1.1
```

/dev/spidev0.0과 /dev/spidev0.1 노드는 SPI0 버스에 존재
/dev/spidev1.0과 /dev/spidev1.1 노드는 SPI1 버스에 존재

U-Boot는 비글본 블랙의 장치 트리 위에 오버레이를 로드
U-Boot의 uEnv.txt 구성 파일을 편집해 로드할 장치 트리 오버레이를 선택할 수 있음

```
$ cat /boot/uEnv.txt
#Docs: http://elinux.org/Beagleboard:U-boot_partitioning_
layout_2.0

uname_r=4.19.94-ti-r42
.
.
.
###U-Boot Overlays###
###Documentation: http://elinux.org/
Beagleboard:BeagleBoneBlack_Debian#U-Boot_Overlays
###Master Enable
enable_uboot_overlays=1
###
.
.
.
###Disable auto loading of virtual capes (emmc/video/wireless/
adc)
#disable_uboot_overlay_emmc=1
#disable_uboot_overlay_video=1
#disable_uboot_overlay_audio=1
#disable_uboot_overlay_wireless=1
#disable_uboot_overlay_adc=1
.
.
.
###Cape Universal Enable
enable_uboot_cape_universal=1
```

- enable_uboot_overlays와 enable_uboot_cape_universal 환경 변수가 모두 1로 설정돼 있는지 확인

- 이 구성 파일은 U-Boot 환경에 적용되므로 /boot/uEnv.txt 에 저장한 모든 변경 사항을 적용하려면 재부팅

(U-Boot가 로드한 장치 트리 오버레이를 나열 명령어)

```
$ cd /opt/scripts/tools
$ sudo ./version.sh | grep UBOOT
UBOOT: Booted Device-Tree:[am335x-boneblack-uboot-univ.dts]
UBOOT: Loaded Overlay:[AM335X-PRU-RPROC-4-19-TI-00A0]
UBOOT: Loaded Overlay:[BB-ADC-00A0]
UBOOT: Loaded Overlay:[BB-BONE-eMMC1-01-00A0]
UBOOT: Loaded Overlay:[BB-NHDMI-TDA998x-00A0]
```

(사용가능한 모든 핀 그룹을 보는 명령어)

```
$ cat /sys/kernel/debug/pinctrl/*pinmux*/pingroups
```

(SPI 핀 그룹만 보려면 다음 코드 사용)

```
$ cat /sys/kernel/debug/pinctrl/*pinmux*/pingroups | grep spi
group: pinmux_P9_19_spi_cs_pin
group: pinmux_P9_20_spi_cs_pin

 group: pinmux_P9_17_spi_cs_pin
 group: pinmux_P9_18_spi_pin
 group: pinmux_P9_21_spi_pin
 group: pinmux_P9_22_spi_sclk_pin
 group: pinmux_P9_30_spi_pin
 group: pinmux_P9_42_spi_cs_pin
 group: pinmux_P9_42_spi_sclk_pin
```

- 일반적으로 버스의 모든 SPI 핀은 동일한 핀 그룹에서 함께 다중화

(am335x-boneblack-uboot-univ.dts파일은 다음 incluede문들을 담고 있음)

```
#include "am33xx.dtsi"
#include "am335x-bone-common.dtsi"
#include "am335x-bone-common-univ.dtsi"
```

- 이 .dts 파일은 포함된 3개의 .dtsi 파일과 함께 장치 트리 소스를 정의
- dtc 도구는 소스파일들을 am335x-boneblack-ubootuniv.dtb파일로 컴파일
- U-Boot는 케이프 범용 장치 트리 위에 장치 트리 오버레이를 로드 (장치 트리 오버레이의 파일 확장자는 .dtbo)

(Pinmux_P9_17_spi_cs_pin에 대한 핀 그룹 정의)

```
P9_17_spi_cs_pin: pinmux_P9_17_spi_cs_pin { pinctrl-single,pins = <
    AM33XX_IOPAD(0x095c, PIN_OUTPUT_PULLUP | INPUT_EN | MUX_MODE0) >; };
```

- Pinmux_P9_17_spi_cs_pin 그룹은 P9 확장 헤더 핀 17을 SPI0 버스의 CS 핀 역할을 하도록 설정

(P9_17_pinmux 정의, pinmux_P9_17_spi_cs_pin 이 참조됨)

```
/* P9_17 (ZCZ ball A16) */
P9_17_pinmux {
    compatible = "bone-pinmux-helper";
    status = "okay";
    pinctrl-names = "default", "gpio", "gpio_pu", "gpio_pd",
"gpio_input", "spi_cs", "i2c", "pwm", "pru_uart";
    pinctrl-0 = <&P9_17_default_pin>;
    pinctrl-1 = <&P9_17_gpio_pin>;
    pinctrl-2 = <&P9_17_gpio_pu_pin>;
    pinctrl-3 = <&P9_17_gpio_pd_pin>;
    pinctrl-4 = <&P9_17_gpio_input_pin>;
    pinctrl-5 = <&P9_17_spi_cs_pin>;
    pinctrl-6 = <&P9_17_i2c_pin>;
    pinctrl-7 = <&P9_17_pwm_pin>;
    pinctrl-8 = <&P9_17_pru_uart_pin>;
};
```

- Spi_cs는 기본 구성이 아니므로 SPI0 버스는 초기에 비활성화돼 있음

(/dev/spidev0.0을 활성화하려면 다음 config_pin 명령 실행)

```
$ config-pin p9.17 spi_cs
Current mode for P9_17 is:    spi_cs
$ config-pin p9.18 spi
Current mode for P9_18 is:    spi
$ config-pin p9.21 spi
Current mode for P9_21 is:    spi
$ config-pin p9.22 spi_sclk
Current mode for P9_22 is:    spi_sclk
```

- 권한 오류가 발생하면 앞에 sudo를 붙여 config-pin 명령 다시 실행
- 데비안 사용자의 비밀번호로 temppwd 를 입력

(데비안에는 깃이 설치돼 있으므로, 아카이브 복제)

```
ysj@build-master:~$ git clone https://github.com/PacktPublishing/Mastering-Embedded-Linux-Programming-Third-Edition.git MELP
Cloning into 'MELP'...
remote: Enumerating objects: 582, done.
remote: Counting objects: 100% (69/69), done.
remote: Compressing objects: 100% (44/44), done.
remote: Total 582 (delta 31), reused 51 (delta 25), pack-reused 513 (from 1)
Receiving objects: 100% (582/582), 266.58 KiB | 1.37 MiB/s, done.
Resolving deltas: 100% (219/219), done.
ysj@build-master:~$
```

(비글본 블랙 부팅할 때 /dev/spidev0.0 활성화 명령어)

```
$ MELP/Chapter12/config-spi0.sh
```

(spidev_test 프로그램 컴파일을 하려면 다음 명령 사용)

```
$ cd MELP/Chapter12/spidev-test
$ gcc spidev_test.c -o spidev_test
```

(spidev_test 프로그램 실행)

```
$ ./spidev_test -v
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF F0 0D | ......@....?.........
.........?.
RX | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 | ....................
...........
```

- -v 플래그는 --verbose의 약자이며 TX 버퍼의 내용을 표시
- SPI의 전이중 특성상 버스 마스터가 전송하는 동안 데이터를 수신
- 이 경우, RX 버퍼는 데이터가 수신되지 않았으므로 모두 0

(점퍼 와이어를 이용해 핀 (SPI0_D1을 비글본 블랙의 P9 확장 헤더에 있는 핀21(SPI0_D0)에 연결 - 루프백)

P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETn
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_40
GPIO_48	15	16	GPIO_51
SPI0_CS0	17	18	SPI0_D1
SPI1_CS1	19	20	SPI1_CS0
SPI0_D0	21	22	SPI0_SCLK
GPIO_49	23	24	GPIO_15
GPIO_117	25	26	GPIO_14
GPIO_125	27	28	SPI1_CS0
SPI1_D0	29	30	SPI1_D1
SPI1_SCLK	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	SPI1_CS1
DGND	43	44	DGND
DGND	45	46	DGND

- P9 확장 헤더의 맵은 USB 포트가 하단에 있을 때 헤더가 비글본 블랙의 왼쪽에 오도록 방향 설정
- SPI0_D1에서 SPI0_D0까지의 점퍼 와이어는 MOSI 를 MISO에 공급해 루프백 연결 형성

(루프백 연결 있는 상태에서 spidev_test 프로그램을 다시 실행)

```
$ ./spidev_test -v
spi mode: 0x0
bits per word: 8

max speed: 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF F0 0D |
......@.......................
RX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF F0 0D |
......@.......................
```

- RX 버퍼의 내용은 이제 TX 버퍼의 내용과 일치 할 것이며, /dev/spidev0.0 인터페이스가 완전히 작동함