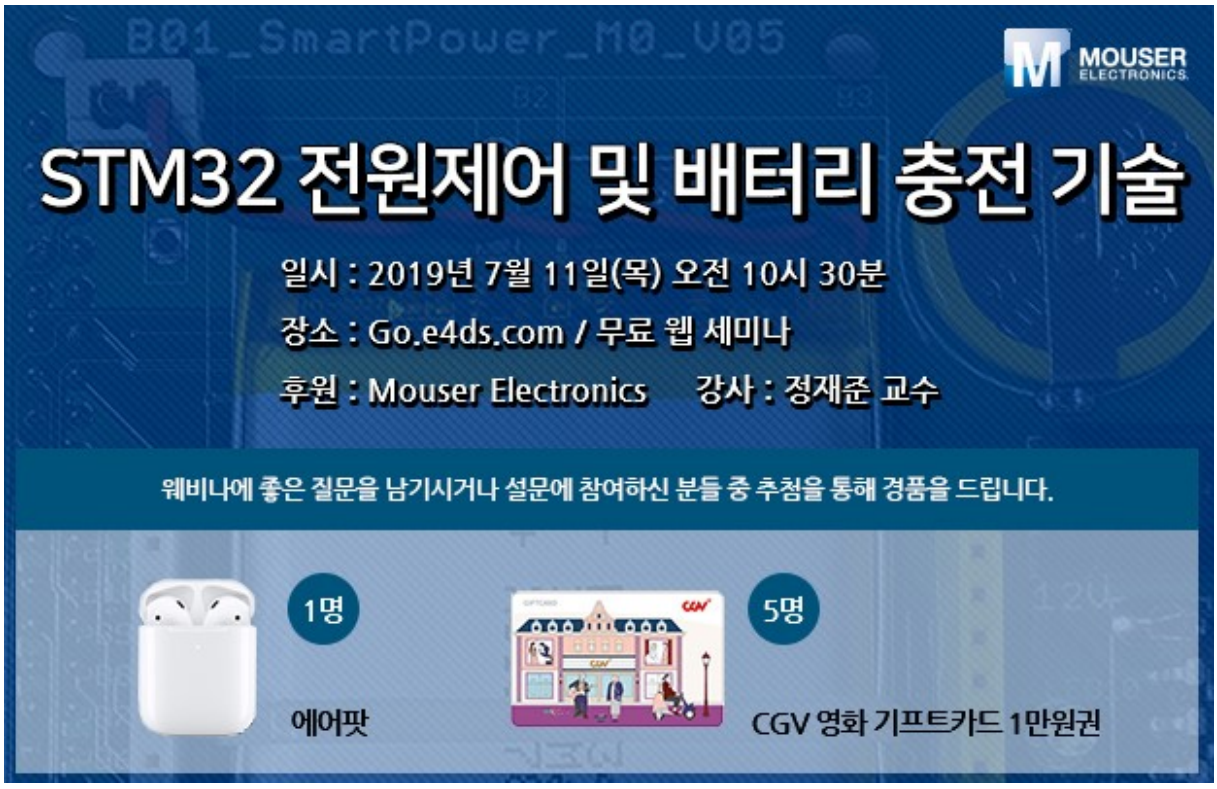
STM32 전원제어 및 배터리 충전 기술



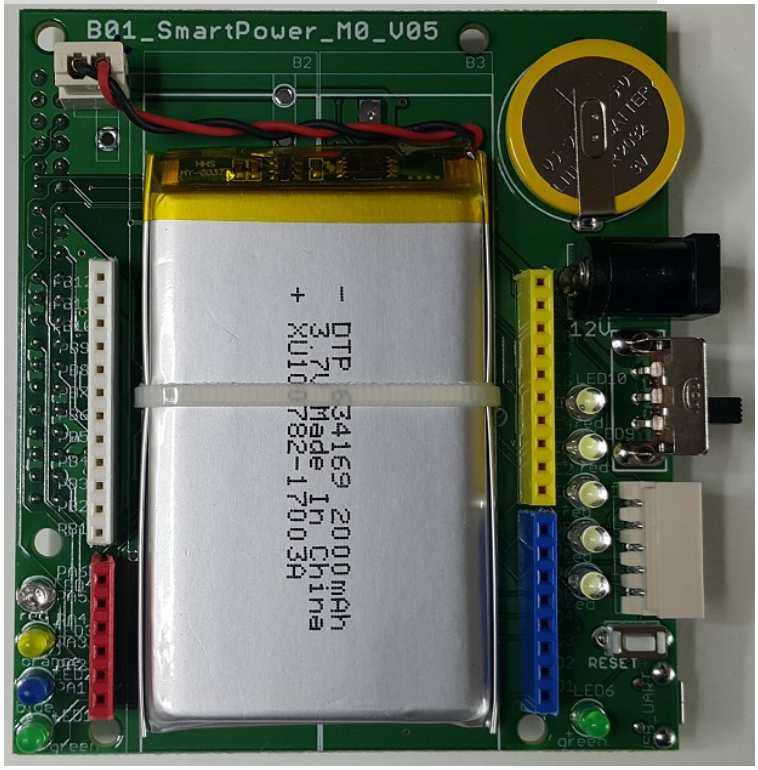
SmartPower(전원배터리) 모듈

커널연구회에서 보드를 설계하여 제작한 SmartPower 모듈의 외형은 아래 사진과 같다. 이 모듈의 제품 모델명은 B01_SmartPower_V05 이다. 이 보드는 라즈베리파이 40 핀 I/O 핀에 장착할 수 있고, 아두이노 헤더핀과도 호환된다. 외부에 12V 전원 어댑터 연결잭을 통하여 전원을 공급하고, 보드 상단에 장착되어 있는 배터리를 충전할 수 있다. 12V 외부 전원과 배터리를 모두 사용할 수 있고, 여기에 적층 형태로 장착되는 라즈베리파이, 아두이노, 무선통신모듈, 센서, 모터, 카메라, 사운드 모듈등에 전원을 공급하고 On/Off 스위칭한다. Cortex-M0(STM32F0) MCU 가 탑재되어 있어서 배터리 충전상태, 전류 사용량, 전원 On/Off 스위칭 제어, 보드 표면 온도, RTC 날짜 및 시간등을 시리얼통신 명령어로 모니터링하고 관리할 수 있다. 이 보드에 있는 마이크로 USB 를 PC 에 연결하고 PC 에서 시리얼통신 터미널을 실행하여 AT 명령어들을 사용하여 보드의 데이터를 조회하고 변경 및 관리한다. 좀더 자세한 내용들은 아래부터 자세히 설명된다.

2.1 하드웨어 제원 및 기능

먼저, B01_SmartPower_V05 의 하드웨어 제원 및 기능에 대해서 알아보도록 하자. 아래 사진은 이 모듈의 외형을 보여주는 것이다.

SmartPower 모듈(B01_SmartPower_V05) 외형

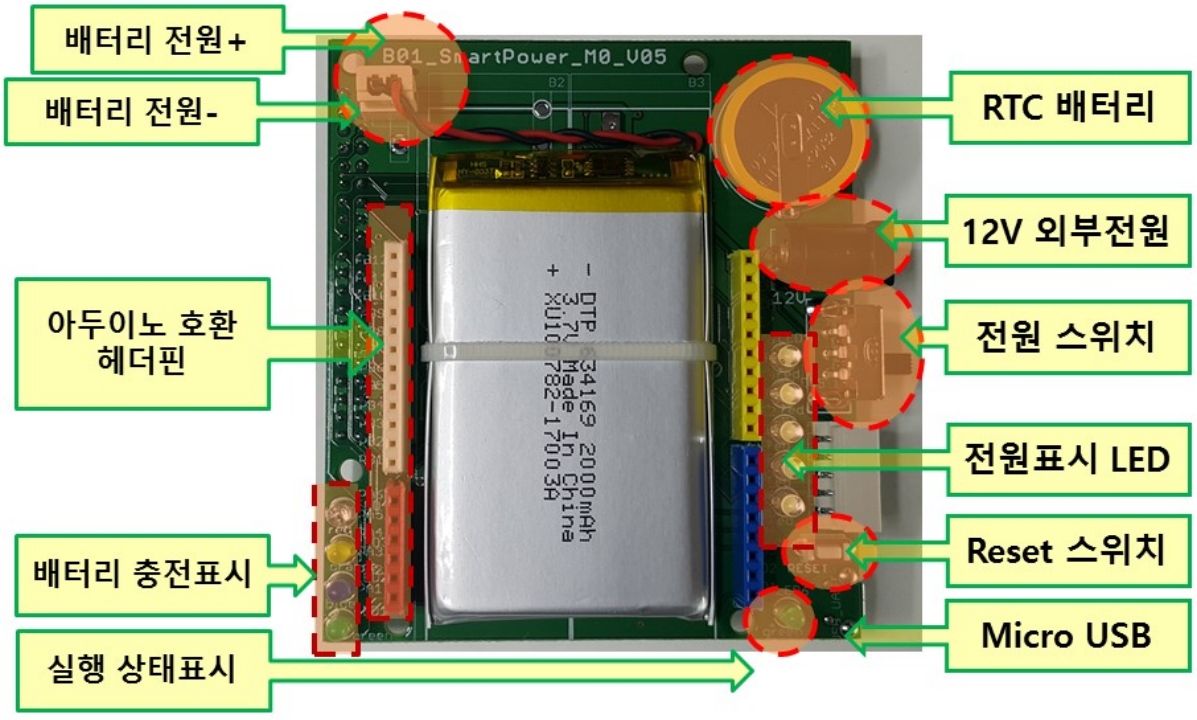


SmartPower 모듈(B01_SmartPower_V05) 제원 및 기능요약

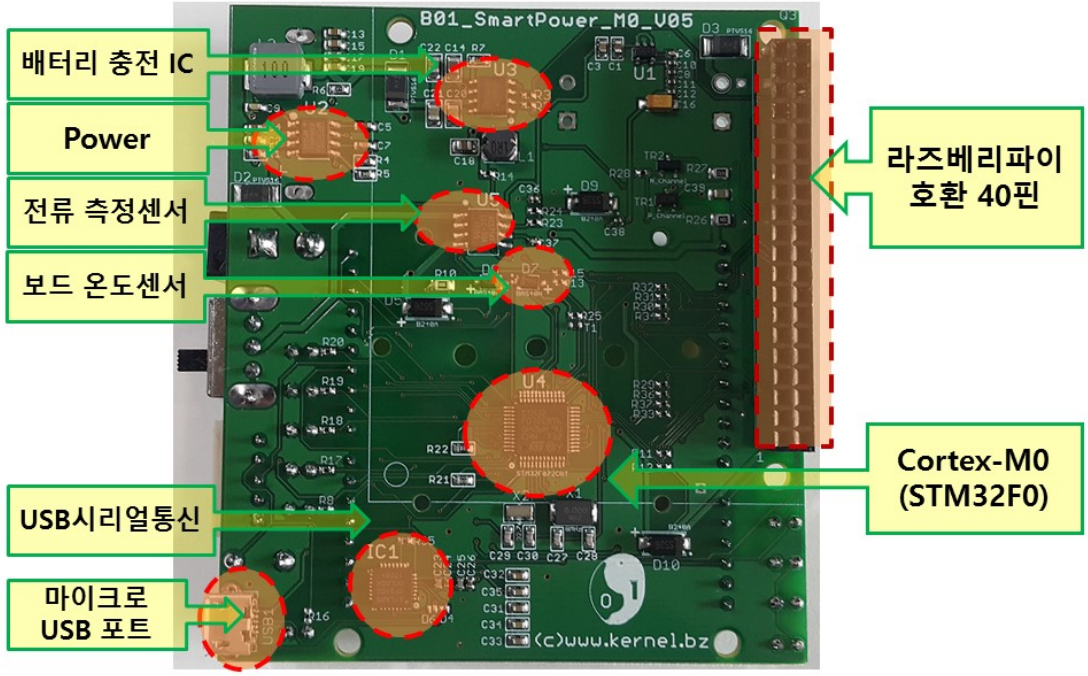
- 보드 크기는 가로 75mm, 세로 85mm (라즈베리파이 및 아두이노 헤더핀 호환)
- Cortex-M0(STM32F0) MCU 내장
- 12V 외부 어댑터 전원공급(전류는 2A 이상 권장)
- 배터리(3.7V 리튬폴리머) 충전기능
- 배터리 용량 2000mA
- 배터리 충전 상태 및 전류사용량 모니터링 가능
- 보드 표면 온도 모니터링
- 라즈베리파이, 아두이노, 무선모듈, 센서, 모터, 카메라, 사운드 모듈에 전원공급 및 각각의 모듈별 전원을 On/Off 스위칭(효율적 전원관리)
- 마이크로 USB 포트에 PC 의 시리얼통신 터미널을 연결하여 AT 명령어로 동작 제어

아래 부터는 이 SmartPower 모듈(B01_SmartPower_V05)에 있는 각각의 부품별 기능을 설명한다.

B01_SmartPower_V05 상단부 설명

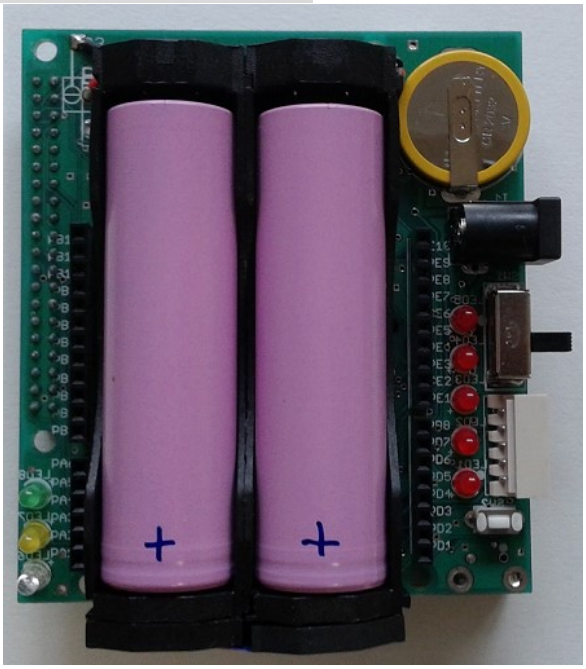


B01_SmartPower_V05 하단부 설명



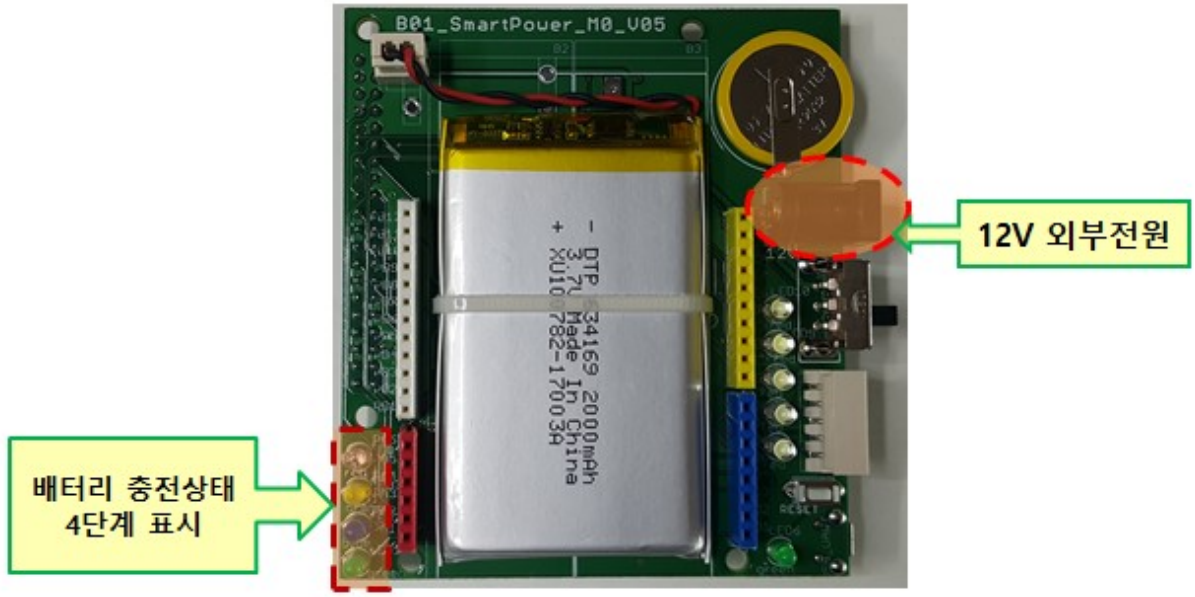
아래 사진은 이 POWER 모듈(B01_SmartPower_V05)에 대용량 배터리 5A(2500mA 2 개 병렬)를 연결한 모습이다.

B01_SmartPower_V05에 대용량 배터리 장착



배터리 충전은 아래 그림과 같이 12V 외부 어댑터 전원을 인가하면 전원 스위치에 상관없이 바로 충전을 시작한다. 충전 상태는 왼쪽 하단부에 있는 LED 를 통해서 4 단계로 표시된다. 제일 하단부부터 첫번째 LED 가 점등되면 20% 충전, 두번째 LED 가 점등되면 40% 충전, 세번째 LED 가 점등되면 60% 충전, 네번째 LED 가 점등되면 80% 충전, 모든 LED 가 다 켜지면 100% 충전된 상태를 나타낸다.

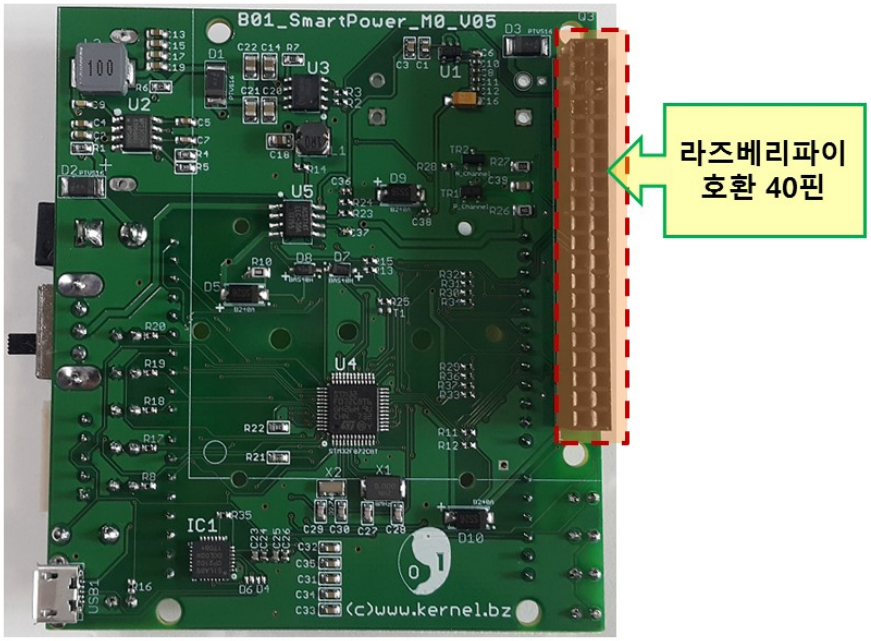
배터리 충전상태 4단계 표시 기능

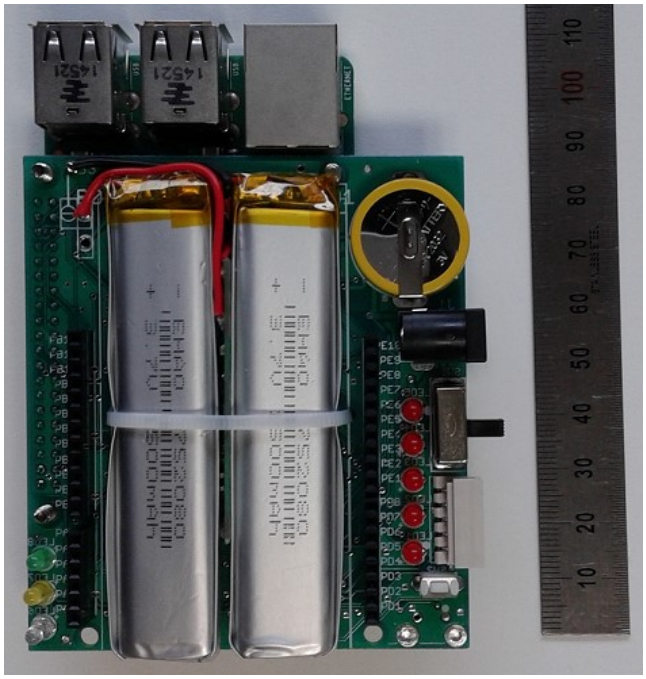


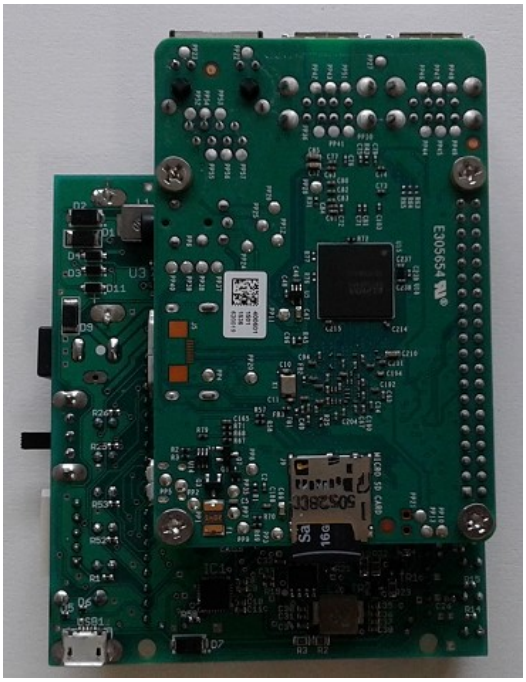
배터리(2000mA 기준) 충전 시간은 약 2 시간 소요되고, 동작시간은 평균적으로 5 시간정도 동작한다.

아래 사진들은 SmartPower 보드의 40 핀에 라즈베리파이를 연결한 모습이다.

B01_SmartPower_V05에 라즈베리파이 연결



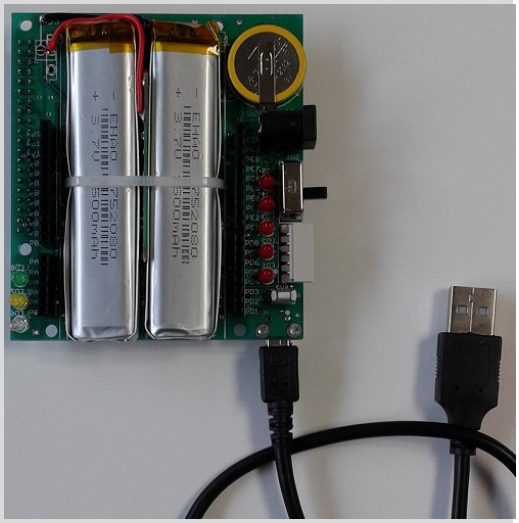




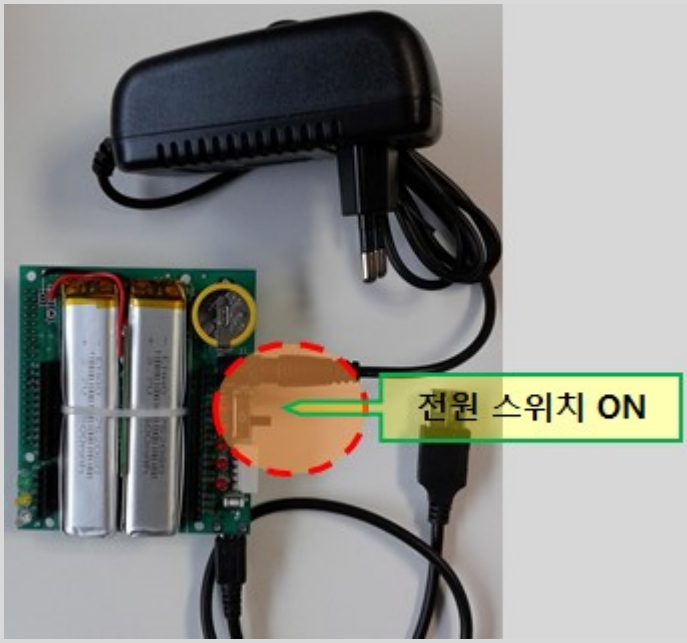
2.2 윈도우 PC에 연결방법

아래 사진처럼, SmartPower 모듈(B01_SmartPower_V05)에 있는 마이크로 USB 포트를 윈도우가 설치되어 있는 PC 에 연결한다. USB 연결선을 PC 에 장착만 하면, USB 로부터 5V 전원이 공급되므로 SmartPower 모듈 하나는 충분히 동작된다. 하지만, 모듈을 여러 개 연결한 경우에는 USB 단독으로 공급되는 전류(500mA)가 약하므로 외부전원이나 배터리 전원 스위치를 On 해 주어야 한다.

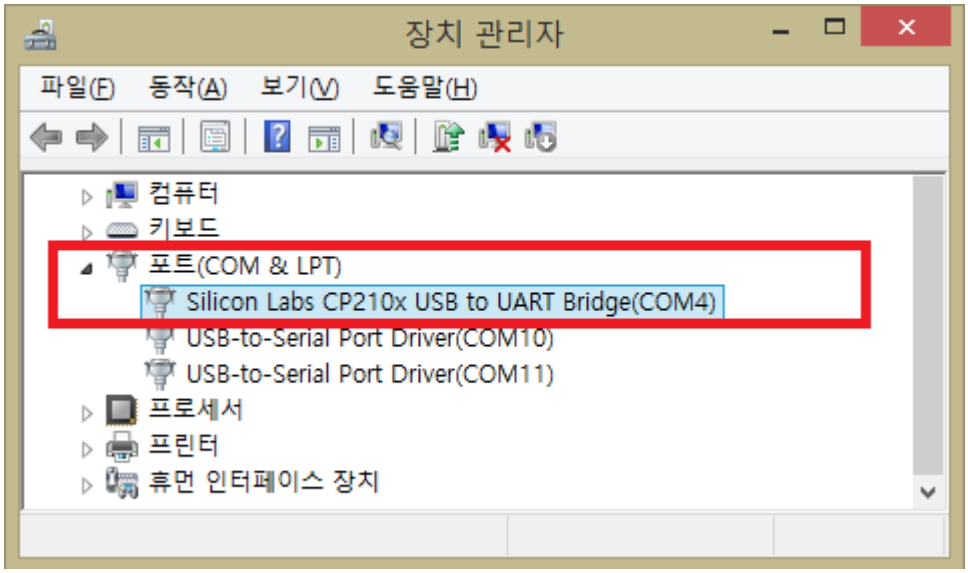
모듈 한 개는 USB 전원가능



여러 모듈인 경우 외부전원 스위치 On



윈도우의 장치 관리자를 확인해 보면 아래와 같이 _SmartPower 의 시리얼 장치가 자동 인식된다.

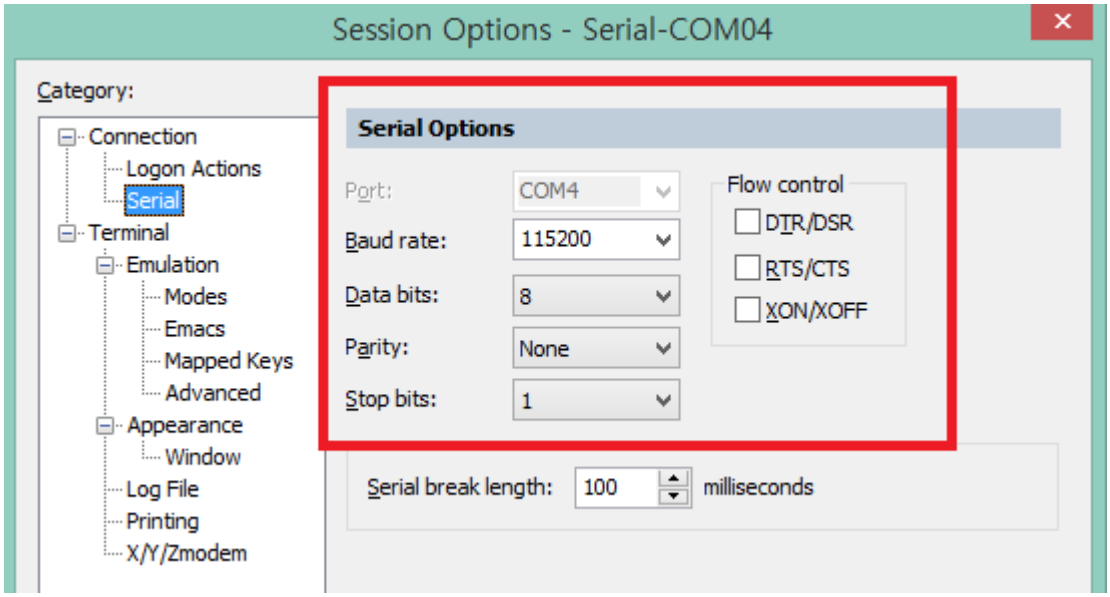


커널연구회는 B01_SmartPower_V05 에 USB 시리얼(UART) 장치로 Silicon Labs 사의 CP2102 IC 를 사용했다. 이 장치는 PC 에 기본적으로 드라이버가 설치되어 있으므로 별도로 드라이버를 설치하지 않아도 자동으로 인식된다.

PC 의 장치관리자에서 USB 시리얼(UART) 통신 장치의 포트 번호를 확인한다. 위의 그림에서는 COM4 포트이다. (포트 번호는 PC 마다 달라질 수 있다.)

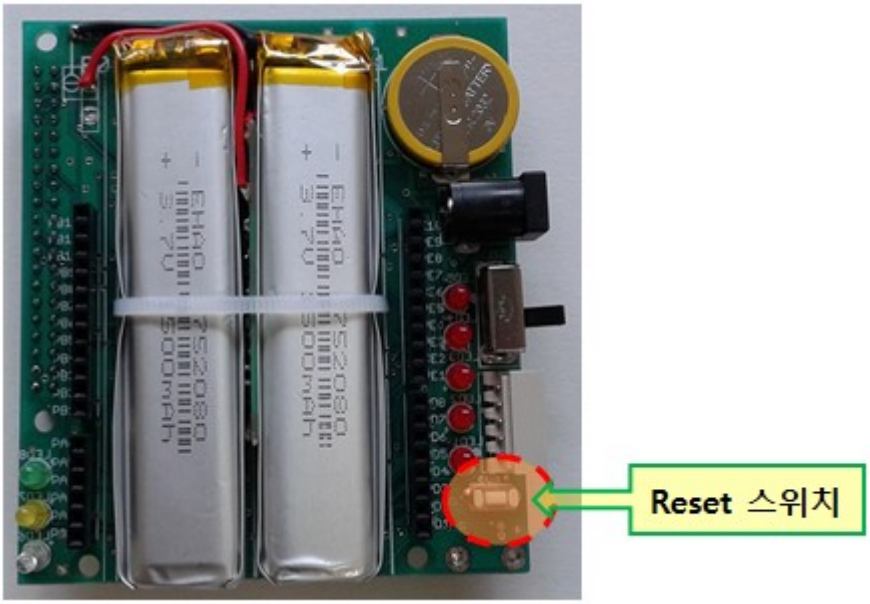
PC 에서 시리얼 터미널 프로그램을 실행한다. 윈도우 시리얼 터미널이나 Putty, SecureCRT 등 사용하기 편한 시리얼 터미널 프로그램을 실행하면 된다. 실행한 시리얼 터미널 프로그램의 시리얼 통신 설정(옵션) 메뉴에서 시리얼 옵션들을 다음과 같이 설정한다.

시리얼 통신 옵션 설정



시리얼 Baud rate 는 115200, Data bits 는 8, Parity 는 None, Stop bits 는 1 로 설정하고, Flow control 는 모두 체크하지 않는다.

위와 같이 시리얼통신 옵션들을 설정한후, B01_SmartPower_V05 보드의 리셋 버턴을 한번 눌러주면 시리얼 터미널 화면에 아래와 같은 메시지들이 출력되면서 동작을 시작한다.




```
r_uart_init(USART1): [OK]
user_uart_init(USART3): [OK]
WARNING: user_rtc_init: 83: RTC: External reset occured.
user_rtc_init(): [OK]
user_timer6_init(): [OK]
user_timer7_init(): [OK]
user_i2c_init(I2C1): [OK]
usr_adc_dma_init(): [OK]
(c)www.kernel.bz B01_SmartPower_V05 Started
```

시리얼 터미널에 키보드로부터 AT? 라고 명령을 입력하면 다음과 같은 AT 명령어 사용법(도움말)이 출력된다. 이 명령어들의 자세한 사용법은 다음절부터 자세히 설명한다.

AT 명령어 도움말

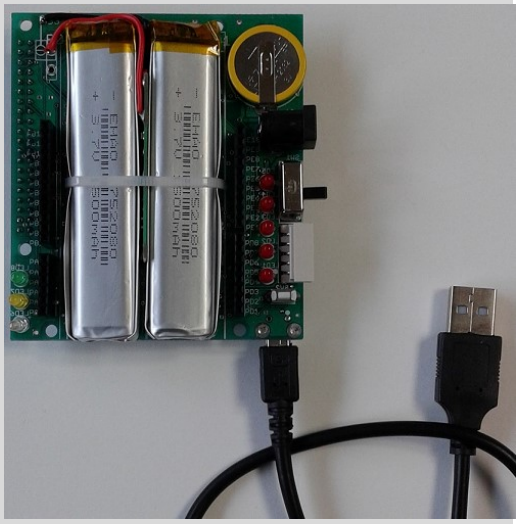
```
-------- Module Version ----------------------------------------
B01_SmartPower_V05 (c)www.kernel.bz

-------- Date Time Command -------------------------------------
AT+TIME?        Time Output
AT+DATE?        Date Output
AT+DTIM?        Date Time Output
AT+TIME=hhmmss  Time Input(hh:24Hour, mm:Minute, ss:Second)
AT+DATE=YYMMDDWW Date Input(YY:Year, MM:Month, DD:Day, WW:Week)
        (01:Mon, 02:Tue, 03:Wed, 04:Thu, 05:Fri, 06:Sat, 07:Sun)
AT+ALRM=hhmmss  Alarm Time Input
```

```
-------- Data Check Command ------------------------------------
AT+BATT?        Battery Data Output
AT+CURR?        Current Data Output
AT+THEM?        Themistor Data Output

-------- Power Control Command ---------------------------------
AT+TRIG=n,0     OFF(n:0=Raspi, 1=WiFi, 2=Sensor, 3=Sound, 4=Motor)
AT+TRIG=n,1     ON(n:0=Raspi, 1=WiFi, 2=Sensor, 3=Sound, 4=Motor)

[OK]
```

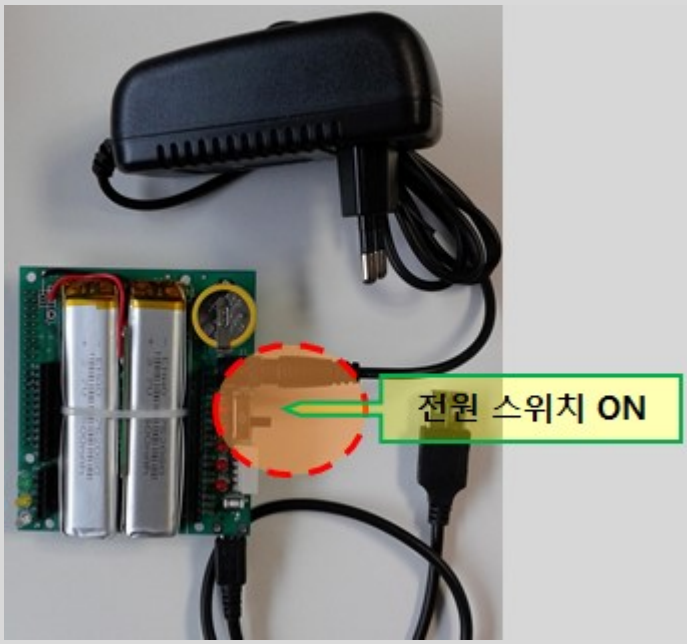
2.3 리눅스 PC에 연결방법

SmartPower 모듈(B01_SmartPower_V05)에 있는 마이크로 USB 포트를 리눅스가 설치되어 있는 PC 에 연결한다. USB 연결선을 PC 에 장착만 하면, USB 로부터 5V 전원이 공급되므로 SmartPower 모듈 하나는 충분히 동작된다. 하지만, 모듈을 여러 개 연결한 경우에는 USB 단독으로 공급되는 전류(500mA)가 약하므로 외부전원이나 배터리 전원 스위치를 On 해 주어야 한다.

모듈 한 개는 USB 전원가능



여러 모듈인 경우 외부전원 스위치 On



위와 같이 연결했다면, 리눅스 터미널에서 다음과 같은 리눅스 명령어를 입력하여 B01_SmartPower_V05 모듈의 USB 시리얼통신 장치가 연결되었는지 확인한다. (별도로 설정하지 않아도 자동으로 연결된다)

```
$ dmesg | grep tty

[    0.000000] console [tty0] enabled
[  247.957798] usb 2-2: cp210x converter now attached to ttyUSB0
```

아래 명령어로 SmartPower 모듈의 USB 시리얼 장치가 /dev/ttyUSB0 노드에 연결되어 있음을 확인할 수 있다.

```
$ ll /dev/ttyUSB*

crw-rw---- 1 root dialout 188, 0 Mar 31 18:44 /dev/ttyUSB0
```

이제, 리눅스 명령창에서 시리얼 터미널 프로그램을 실행한다. 리눅스에서는 minicom 을 시리얼 터미널 프로그램으로 많이 사용한다. 이 프로그램이 설치되어 있지 않다면, 다음과 같이 설치한다.

```
$ sudo apt-get install minicom
```

이제 minicom 을 다음과 같이 실행한다. (루트 권한으로 실행)

```
$ sudo minicom
```

실행 메시지가 다음과 같이 화면에 출력된다.

```
Welcome to minicom 2.6.1

OPTIONS: I18n
Port /dev/ttyUSB0

Press CTRL-A Z for help on special keys
```

시리얼통신 포트가 위와 같이 /dev/ttyUSB0 로 설정되어 있지 않고, 통신 속도(baud rate)도 115200 으로 설정되어 있지 않다면, 키보드에서 Ctrl-A 를 누른후 O 키를 눌러서 다음과 같이 옵션 설정 메뉴로 진입한다.

```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
+--------------------------+
```

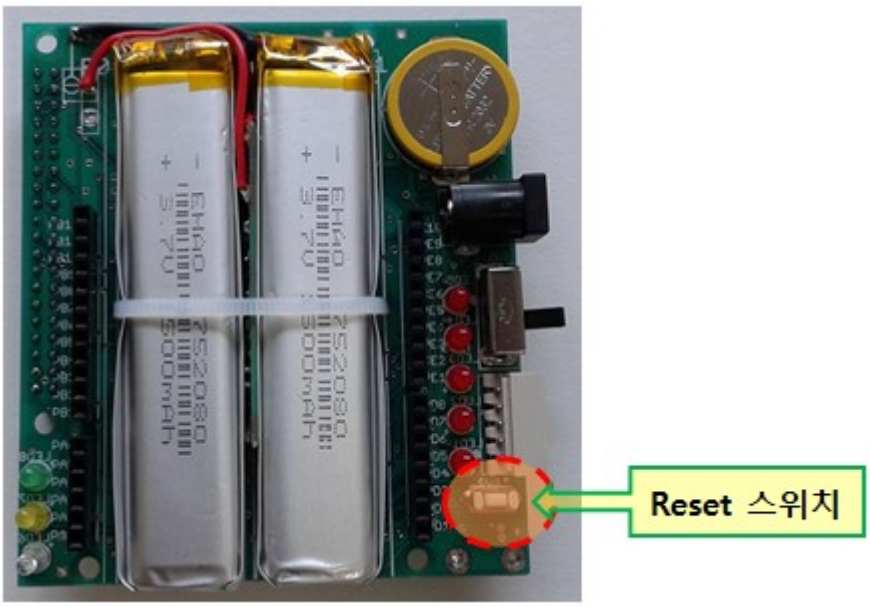
화살표를 아래로 눌러서 세번째 메뉴인 Serial port setup 을 선택한후 엔터하면 아래와 같이 설정 옵션 메뉴가 나타난다.

```
+-----------------------------------------------------------------------+
| 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?                                              |
+-----------------------------------------------------------------------+
```

화면에 나타난 옵션들을 위와 같이 맞추어 준다. 그런다음 다시 상위 메뉴로 돌아가서 Save Setup as dfl 메뉴를 엔터하여 설정내용을 저장하고 Exit 하여 빠져 나온다.

```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
+--------------------------+
```

위와 같이 시리얼통신 옵션들을 설정한후, B01_SmartPower_V05 보드의 리셋 버턴을 한번 눌러주면 minicom 시리얼 터미널 화면에 아래와 같은 메시지들이 출력되면서 동작을 시작한다.




```
r_uart_init(USART1): [OK]
user_uart_init(USART3): [OK]
WARNING: user_rtc_init: 83: RTC: External reset occured.
user_rtc_init(): [OK]
user_timer6_init(): [OK]
user_timer7_init(): [OK]
user_i2c_init(I2C1): [OK]
usr_adc_dma_init(): [OK]
(c)www.kernel.bz B01_SmartPower_V05 Started
```

시리얼 터미널에 키보드로부터 AT? 라고 명령을 입력하면 다음과 같은 AT 명령어 도움말이 출력된다. 이 명령어들의 자세한 사용법은 다음절부터 자세히 설명한다.

AT 명령어 도움말

```
-------- Module Version ----------------------------------------
B01_SmartPower_V05 (c)www.kernel.bz
```

```
-------- Date Time Command -------------------------------------
AT+TIME?        Time Output
AT+DATE?        Date Output
AT+DTIM?        Date Time Output
AT+TIME=hhmmss  Time Input(hh:24Hour, mm:Minute, ss:Second)
AT+DATE=YYMMDDWW Date Input(YY:Year, MM:Month, DD:Day, WW:Week)
        (01:Mon, 02:Tue, 03:Wed, 04:Thu, 05:Fri, 06:Sat, 07:Sun)
AT+ALRM=hhmmss  Alarm Time Input

-------- Data Check Command -------------------------------------
AT+BATT?        Battery Data Output
AT+CURR?        Current Data Output
AT+THEM?        Themistor Data Output

-------- Power Control Command ----------------------------------
AT+TRIG=n,0     OFF(n:0=Raspi, 1=WiFi, 2=Sensor, 3=Sound, 4=Motor)
AT+TRIG=n,1     ON(n:0=Raspi, 1=WiFi, 2=Sensor, 3=Sound, 4=Motor)

[OK]
```

2.4 AT 명령어 사용법

SmartPower 모듈과 PC간에 USB 시리얼 라인이 연결되고 옵션들이 정상적으로 설정되었다면 AT 명령어들을 사용하여 SmartPower 모듈에 프로그램되어 있는 펌웨어 기능들을 동작시킬 수 있다. (혹시 동작하지 않는다면 앞의 연결과정을 다시한번 점검해 주기 바란다)

참고로, AT 명령어들은 소문자/대문자 구분없이 입력하면 되고, SmartPower 모듈 내부에서는 모두 대문자로 변환하여 동작된다.

2.4.1 도움말 확인하기

시리얼 터미널 창에서 키보드로부터 AT? 을 입력하고 엔터하면 다음과 같은 도움말이 화면에 출력된다. B01_SmartPower_V05 는 모듈의 모델명과 버전이다. 명령어들은 크게 3 가지로 나누어져 있다. 첫째는 날짜와 시간을 조회하고 설정하는 명령들이고, 둘째는 모듈 내부의 데이터들을 확인하는 명령어들이고, 셋째는 전원을 On/Off 제어하는 명령어들이다. 이것들에 대해서 아래부터 자세히 사용법을 설명한다.

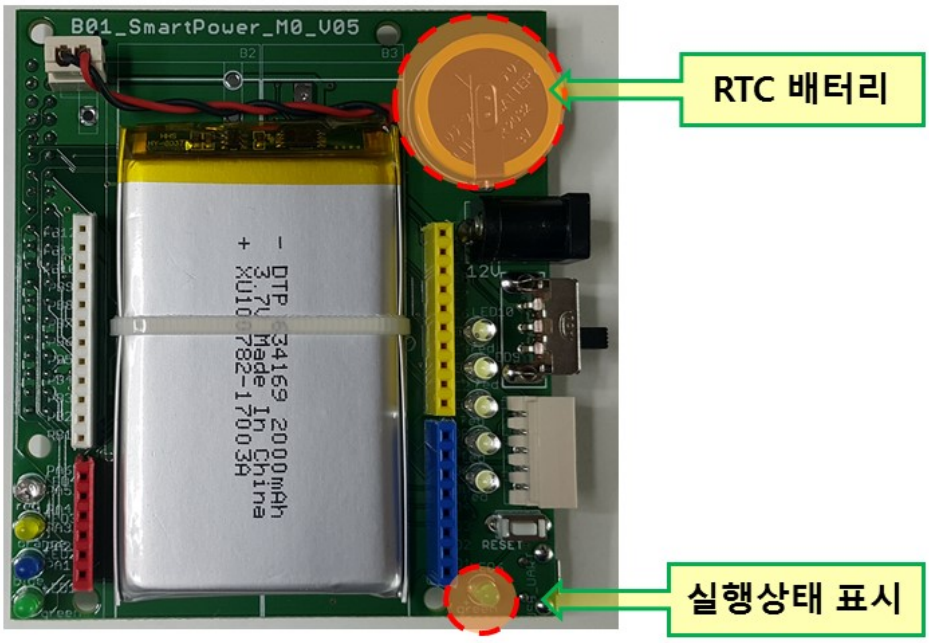
AT 명령어 도움말

```
-------- Module Version ----------------------------------------
B01_SmartPower_V05 (c)www.kernel.bz

-------- Date Time Command -------------------------------------
AT+TIME?        Time Output
AT+DATE?        Date Output
AT+DTIM?        Date Time Output
AT+TIME=hhmmss  Time Input(hh:24Hour, mm:Minute, ss:Second)
AT+DATE=YYMMDDWW Date Input(YY:Year, MM:Month, DD:Day, WW:Week)
        (01:Mon, 02:Tue, 03:Wed, 04:Thu, 05:Fri, 06:Sat, 07:Sun)
AT+ALRM=hhmmss  Alarm Time Input

-------- Data Check Command ------------------------------------
AT+BATT?        Battery Data Output
AT+CURR?        Current Data Output
AT+THEM?        Themistor Data Output

-------- Power Control Command ---------------------------------
```

```
AT+TRIG=n,0      OFF(n:0=Raspi, 1=WiFi, 2=Sensor, 3=Sound, 4=Motor)
AT+TRIG=n,1      ON(n:0=Raspi, 1=WiFi, 2=Sensor, 3=Sound, 4=Motor)

[OK]
```

2.4.2 날짜 시간 조회 및 설정

날짜와 시간은 SmartPower 모듈에 있는 Cortex-M0(STM32F0)의 RTC 에서 유지 관리된다. 전원이 Off 되어도 다음과 같이 코인셀 배터리가 3V 를 항상 공급해 주므로, 한번 설정된 날짜와 시간은 항상 현재 일시로 유지된다.



그리고 SmartPower 모듈 내부에서 타이머가 항상 작동하고 있으며 실행 상태를 표시하기 위해서 녹색 LED 를 1 초마다 점등한다. (위의 사진에서 오른쪽 하단부의 녹색 LED) 날짜와 시간을 관리하는 AT 명령어는 다음과 같이 6 가지가 SmartPower 모듈에 프로그램되어 있다. 이들의 사용법을 익혀서 프로그램에 응용할 수 있다.

날짜와 시간관리 AT 명령어들

```
AT+TIME?
AT+DATE?
AT+DTIM?
AT+TIME=
AT+DATE=
```

```
AT+ALRM=
```

AT 명령어 끝에 ?를 붙이면 데이터를 조회(출력)하는 것이고, =를 붙이면 데이터를 설정(입력)하는 것이다.

AT+TIME? 명령어를 입력하면 다음과 같이 현재 시간이 출력된다.

AT+TIME? 명령어

```
AT+TIME?
20:04:50
[OK]
```

AT+DATE? 명령어를 입력하면 다음과 같이 현재 날짜가 출력된다.

AT+DATE? 명령어

```
AT+DATE?
2017-03-31
[OK]
```

AT+DTIM? 명령어를 입력하면 다음과 같이 현재 날짜, 시간, 요일을 출력한다.

AT+DTIM? 명령어

```
AT+DTIM?
2017-03-31 20:07:55 (FRIDAY)
[OK]
```

AT+TIME=hhmmss 명령어를 입력하면 다음과 같이 시간을 설정한다. hh 에는 두자리로 24 시간 단위의 시간을 입력하고 mm 에는 두자리로 분을 입력하고 ss 에는 두자리로 초를 입력한다.
아래의 예제는 20 시 11 분 01 초로 현재 시간을 설정하는 것이다.

AT+TIME=hhmmss 명령어

```
AT+TIME=201101
[OK]
```

AT+DATE=YYMMDDWW 명령어를 입력하면 다음과 같이 날짜를 설정한다. YY 에는 두자리로 년도를 입력하고 MM 에는 두자리로 월을 입력하고 DD 에는 두자리로 날짜를 입력하고 WW 는 두자리로 요일을 입력한다. 요일은 01(월요일), 02(화요일), 03(수요일), 04(목요일), 05(금요일), 06(토요일), 07(일요일)로 입력해야 한다. 아래의 예제는 17 년 03 월 31 일 05(금요일)로 현재 날짜를 설정하는 것이다.

AT+DATE=YYMMDDWW 명령어

```
AT+DATE=17033105
[OK]
```

AT+ALRM=hhmmss 명령어를 입력하면 알람시간을 설정하는 것이다. hh 에는 두자리로 24 시간 단위의 시간을 입력하고 mm 에는 두자리로 분을 입력하고 ss 에는 두자리로 초를 입력한다. 아래의 예제는 20 시 30 분 01 초로 알람 시간을 설정하는 것이다. 이 시간이 되면 내부적으로 알람 인터럽트가 발생하므로 이시간에 해야하는 일들을 수행할 수 있다.

AT+ALRM=hhmmss 명령어

```
AT+ALRM=203001
[OK]
```

알람 인터럽트는 모듈간에 설정된 신호를 바탕으로 인터럽트 서비스 루틴을 작성해야 하므로, 나중에 전체 모듈들을 조립하여 어플리케이션을 작성할 때 좀더 자세히 설명할 예정이다.

2.4.3 내부 데이터 체크

내부 데이터 체크 AT 명령어들은 SmartPower 모듈내의 배터리 잔량과 전류 사용량, 보드 표면 온도등을 체크하여 그 값을 출력한다. 다음과 같이 3 가지 종류가 있다.

내부 데이터 체크 AT 명령어들

```
AT+BATT?
AT+CURR?
AT+THEM?
```

AT+BATT? 명령어는 아래와 같이 배터리 잔량을 출력한다.

AT+BATT? 명령어

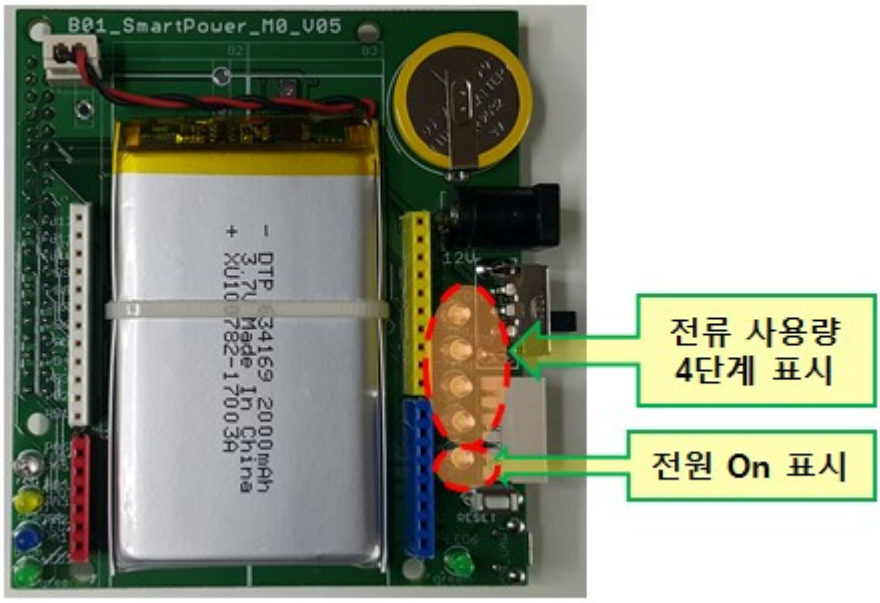
```
AT+BATT?
2960
[OK]
```

AT+CURR? 명령어는 아래와 같이 전류 사용량을 출력하므로 어플리케이션에서 파워 소비량을 점검할 수 있다.

AT+CURR? 명령어

```
AT+CURR?
2527
[OK]
```

SmartPower 모듈에는 아래 사진과 같이 5개의 전원표시 LED가 있다. 이 LED들은 전류사용량에 따라서 다음과 같이 표시된다.



하단부에 있는 LED부터, 첫번째 LED는 전원 스위치를 On하여 전원이 인가되면 무조건 On된다. 두번째 LED는 전류 사용량이 600mA가 넘어가면 On된다. 세번째 LED는 전류 사용량이 800mA 넘어가면 On된다. 네번째 LED는 전류 사용량이 1000mA이 넘어가면 On된다. 제일 상단에 있는 다섯번째 LED는 전류 사용량이 1400mA 넘어가면 On된다.

참고로, 아래 테이블은 SmartPrince의 각 모듈별 전류 사용량을 측정하여 정리한 것이다.

모듈별 전류 사용량

모듈명	전류 소모량
Power Module	50mA
Sensor Module	47mA
Voice Module	90mA
Camera Module	150mA
Motor Module	88mA
DC Motor Active(1개 평균)	70mA
소계	**425mA**
RPI3(라즈베리파이) Booting	400mA
RPI3(라즈베리파이) Normal	250mA
CM3(컴퓨팅모듈) Booting	900mA
CM3(컴퓨팅모듈) Normal	300mA
HDMI Active	200mA
KBD/Mouse	100mA
Normal(평균)	**900mA**
Max(최대)	**2025mA**

AT+THEM? 명령어는 아래와 같이 보드의 표면 온도값을 출력하므로 어플리케이션에서 보드의 표면 온도를 모니터링할 수 있다.

AT+THEM? 명령어

```
AT+THEM?
1811
[OK]
```

2.4.4 전원 On/Off 제어

전원 On/Off 제어용 AT 명령어는 아래와 같이 2 가지 종류가 있다. AT+TRIG=n,0 은 전원을 Off 하는 명령이고 AT+TRIG=n,1 은 전원을 On 하는 명령이다. 전원 제어는 SmartPower 모듈에 적층으로 장착되어 있는 다른 모듈들의 전원을 On/Off 한다. 이렇게 하면 배터리 소모량을 효율적으로 줄일 수 있다.

전원 제어 AT 명령어들

```
AT+TRIG=n,0
AT+TRIG=n,1
```

AT+TRIG=n 에 전달하는 다섯개의 숫자값은 아래 테이블과 같이 전원 제어(On/Off)할 외부 모듈들을 선택하는 역할을 한다.

n 값	모듈명
0	라즈베리파이 3 에 대한 전원 제어
1	무선 WiFi 모듈(SmartWiFi)에 대한 전원 제어
2	센서(SmartSensor)와 카메라(SmartCamera) 모듈에 대한 전원 제어
3	사운드 모듈(SmartSound)에 대한 전원 제어
4	모터 모듈(SmartMotor)에 대한 전원 제어
5	라즈베리파이 컴퓨팅모듈(CM3)에 대한 전원 제어

예를들면, 아래의 명령은 라즈베리파이 모듈(SmartPi CM3)의 전원을 On 하는 AT 명령어이다.

```
AT+TRIG=0,1
[OK]
```

아래의 명령은 라즈베리파이 모듈(SmartPi CM3)의 전원을 Off 하는 AT 명령어이다.

```
AT+TRIG=0,0
[OK]
```

아래의 명령은 무선 WiFi 모듈(SmartWiFi)의 전원을 Off 하는 AT 명령어이다.

```
AT+TRIG=1,0
[OK]
```

아래의 명령은 무선 WiFi 모듈(SmartWiFi)의 전원을 On 하는 AT 명령어이다.

```
AT+TRIG=1,1
[OK]
```

아래의 명령은 모터 모듈(SmartMotor)의 전원을 Off 하는 AT 명령어이다.

```
AT+TRIG=4,0
[OK]
```

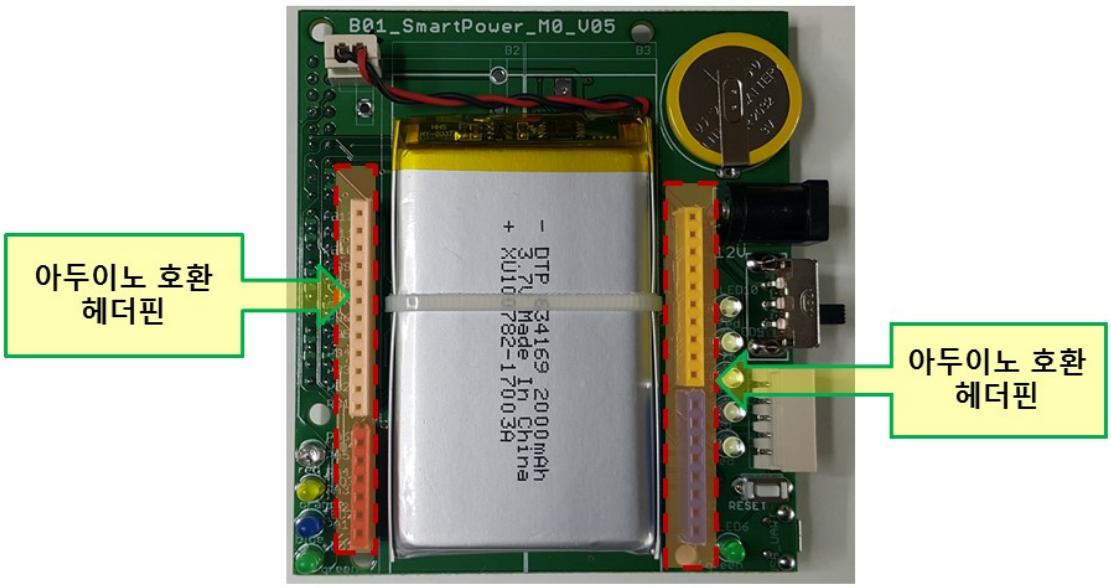
아래의 명령은 모터 모듈(SmartMotor)의 전원을 On 하는 AT 명령어이다.

```
AT+TRIG=4,1
[OK]
```

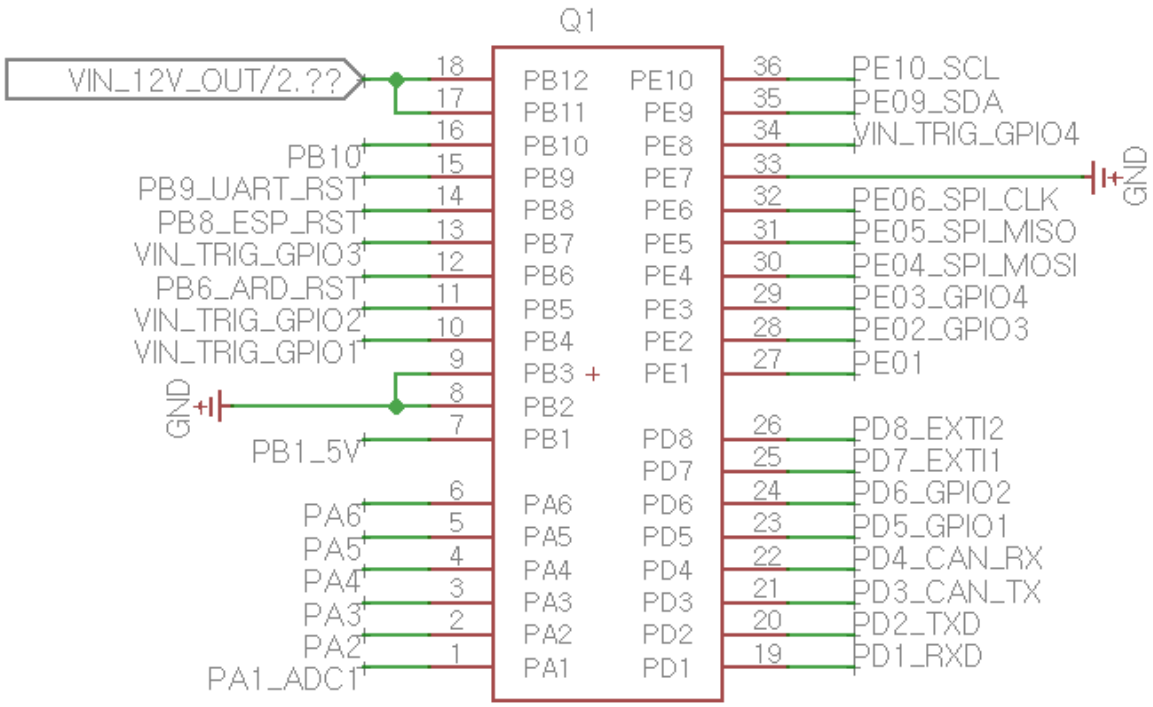
위와 같이 모듈별로 전원을 On/Off 할 수 있으므로 동작하고 있는 모듈만 전원을 On 시키고, 쉬고 있는 모듈은 전원을 Off 하여 배터리 소모를 효율적으로 줄일 수 있다.

2.5 확장핀 연결 및 사용예제

SmartPower 보드의 확장핀은 아두이노 헤더핀과 호환되며 핀맵은 아래와 같다.

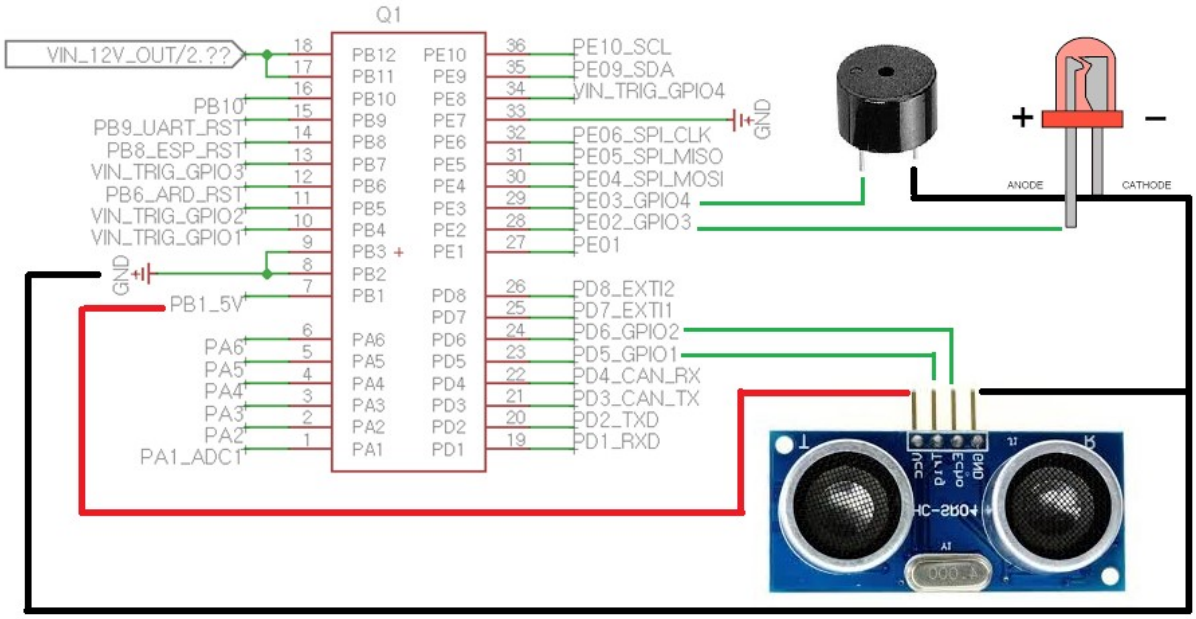


핀맵 회로도



위의 회로도에 초음파 센서와 부저를 연결한 예제는 다음과 같다. 초음파 센서에서 감지되는 거리가 가까워지면 부저를 빠르게 소리내고 거리가 멀어지면 느리게 소리내는 코딩 예제에 해당한다.

부저와 초음파 센서 연결



소스 코드 수정 (user_gpio.h)

```
//User LED
#define GPIO_USER_LED1              GPIOA
#define GPIO_USER_LED1_PIN          GPIO_PIN_12     //PA12, PE2, User LED1
#define GPIO_USER_LED2              GPIOA
#define GPIO_USER_LED2_PIN          GPIO_PIN_15     //PA15, PE3, User LED2

//User GPIO
#define GPIO_USER_GPIO1_TRIG        GPIOA
#define GPIO_USER_GPIO1_PIN         GPIO_PIN_8      //PA8, PD5, GPIO1 Trig
#define GPIO_USER_GPIO2_ECHO        GPIOA
#define GPIO_USER_GPIO2_PIN         GPIO_PIN_11     //PA11, PD6, GPIO2 Echo
```

소스 코드 수정 (user_gpio.c)

```
void user_gpio_init(void)
{
    //User LED Init
    user_gpioA_init(GPIO_USER_LED1_PIN, GPIO_MODE_OUTPUT_PP);       //PA12: PE2: User LED1
    HAL_GPIO_WritePin(GPIO_USER_LED1, GPIO_USER_LED1_PIN,  GPIO_PIN_RESET);
    user_gpioA_init(GPIO_USER_LED2_PIN, GPIO_MODE_OUTPUT_PP);       //PA15: PE3: User LED2
    HAL_GPIO_WritePin(GPIO_USER_LED2, GPIO_USER_LED2_PIN,  GPIO_PIN_RESET);

    //User GPIO Init
    user_gpioA_init(GPIO_USER_GPIO1_PIN, GPIO_MODE_OUTPUT_PP);       //PA8: PD5: User GPIO1(Trig)
    HAL_GPIO_WritePin(GPIO_USER_GPIO1_TRIG, GPIO_USER_GPIO1_PIN,  GPIO_PIN_RESET);
    user_gpioA_init(GPIO_USER_GPIO2_PIN, GPIO_MODE_OUTPUT_PP);       //PA11: PD6: User GPIO2(Echo)
    HAL_GPIO_WritePin(GPIO_USER_GPIO2_ECHO, GPIO_USER_GPIO2_PIN,  GPIO_PIN_RESET);
}
```

소스 코드 수정 (user_timer.c)

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
        //User LED
        HAL_GPIO_TogglePin(GPIO_USER_LED1, GPIO_USER_LED1_PIN); //PA12: PE2: User LED1
        HAL_GPIO_TogglePin(GPIO_USER_LED2, GPIO_USER_LED2_PIN); //PA15: PE3: User LED2

        //User GPIO
        HAL_GPIO_TogglePin(GPIO_USER_GPIO1_TRIG, GPIO_USER_GPIO1_PIN); //PA12: PE2: User LED1
        HAL_GPIO_TogglePin(GPIO_USER_GPIO2_ECHO, GPIO_USER_GPIO2_PIN); //PA15: PE3: User LED2
}
```