NetEye6410 Ver. 2.2

NetEye6410 Ver. 2.2

사용자 매뉴얼 Ver. 1.0



SM Information and Communication

Homepage : http:www.smic21.com
Support : +82-2-6292-2100 (FAX +82-2-6292-2108)
E-Mail : khg@smic.co.kr / khg@smic21.com
Address : 서울시 금천구 가산동 SK 트윈테크타워 A-401 (우편번호 153-023)

NOTICE

SAMS Test bed는 스마트 자동 제어 관리 시스템 (SAMS) 시제품 개발 사업에 대한 산출물로써 SAMS Controller의 기능이 시스템 내에서 원활히 동작하는지 테스트 하는 시스템 입니다.

사용자 임의로 테스트 베드를 파손·분리·재구성 하였을 시에 발생할 수 있는 오동작에 주의 하시기 바랍니다.

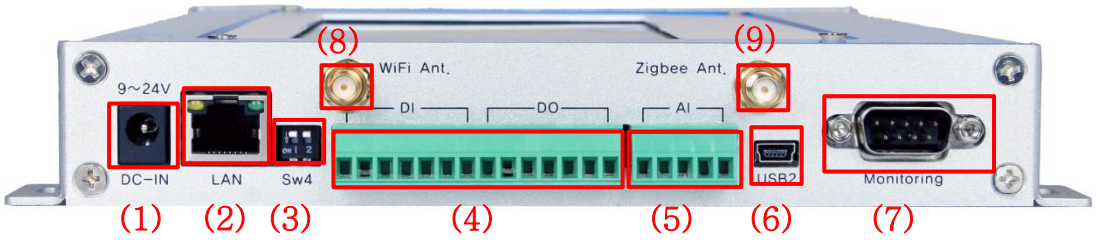
WARNING

- 시스템에 전원이 인가된 후에는 각별히 주의 하십시오
- 운용 중에 절대로 손을 가까이 대지 마십시오
- 다른 전원 장비를 가까이 두고 운용하지 마십시오
- 물이나 습기가 많은 장소에서는 운용하지 마십시오
- 뜨거운 열은 가급적 피해 운용 하십시오
- 날카로운 칼, 가위 등은 장비를 훼손할 우려가 있습니다

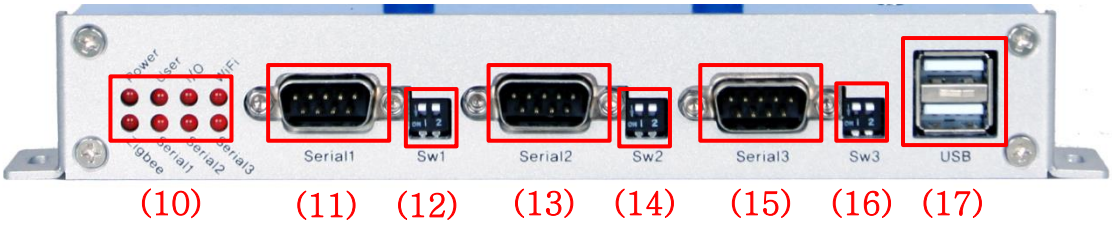
1. NetEye6410 v2.2 개요

1-1. 제품구성

(1) DC-IN 전원 커넥터 (9~24V)
(2) RJ45 10/100M Ethernet 커넥터
(3) Wi-Fi 설정 초기화 스위치
(4) DI/DO 터미널블럭
(5) AI 터미널블럭
(6) Zigbee 포트
(7) Monitoring 포트
(8) Wi-Fi 안테나
(9) Zigbee 안테나



(10) 상태 LED
(11) Serial 1 포트
(12) Serial 1 포트 RS232/422/485 설정 스위치
(13) Serial 2 포트
(14) Serial 2 포트 RS232/422/485 설정 스위치
(15) Serial 3 포트
(16) Serial 3 포트 RS232/422/485 설정 스위치
(17) USB 연결 호스트



(18) SD Card 슬롯



1-2. LED 상태

- LED ①은 전원 상태를 나타내는 LED 입니다.
- LED ②는 사용자 프로그램에서 제어 할 수 있는 LED 입니다.
- LED ③은 DI/DO 상태를 나타내는 LED입니다.
  (1.5초마다 점멸함, 순간적인 점멸은 DI/DO와 CPU간의 통신을 뜻함)
- LED ④는 Wi-Fi Module의 상태를 나타내는 LED 입니다.
  (전원 입력 시 0.5초마다 점멸, 부팅 완료 시 1초마다 점멸, 연결 완료 시 1.5초마다 점멸)
- LED ⑤는 Zigbee 상태를 나타내는 LED 입니다.
- LED ⑥은 Serial1의 상태를 나타내는 LED 입니다.
- LED ⑦은 Serial2의 상태를 나타내는 LED 입니다.
- LED ⑧은 Serial3의 상태를 나타내는 LED 입니다.



1-3. 제품 외관



1-4. NetEye6410 v2.2 기본 구성품

- NetEye6410 본체 * 1 pcs
- 5dB Wireless LAN (Wi-Fi) Antenna * 1 pcs
  (Zigbee Antenna 는 옵션 선택 시 별도 제공)
- 9V DC Adaptor * 1 pcs
- Manual CD-ROM * 1 pcs



<NetEye6410 본체>



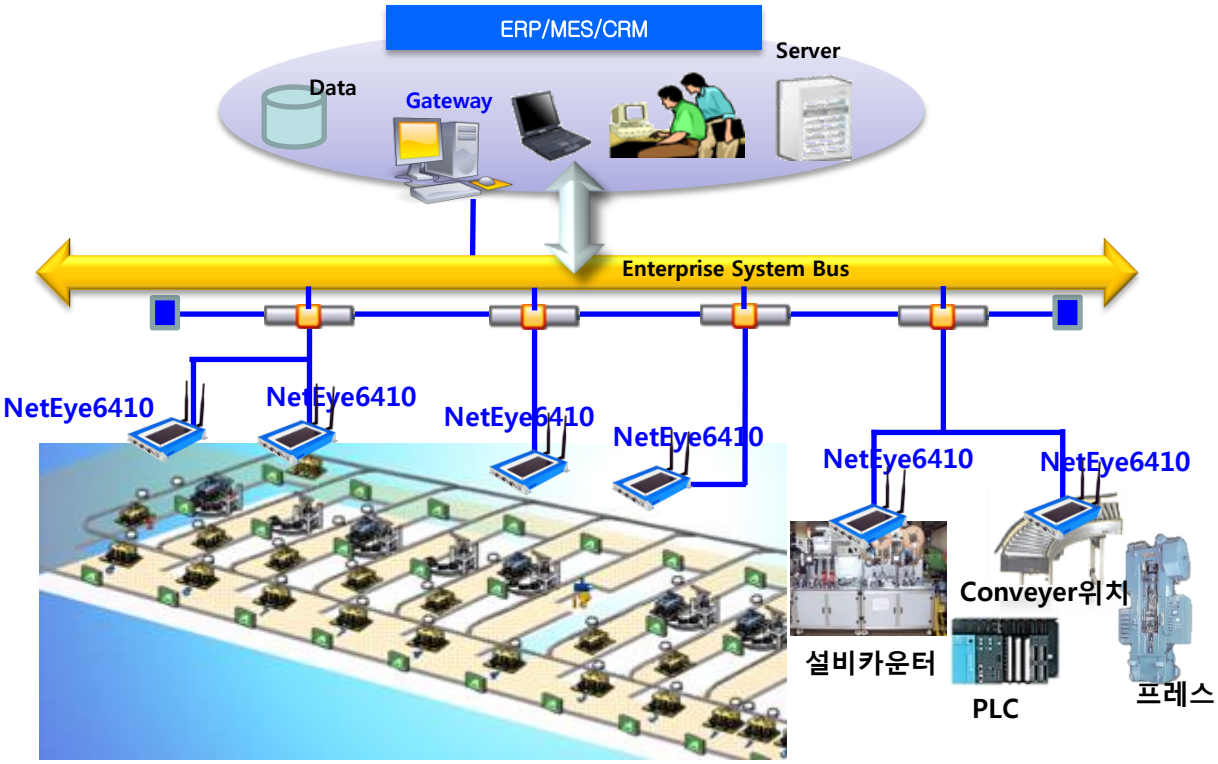
<9VDC Adaptor>



<Manual CD-ROM>

1-5. Application 연결 구성도

NetEye6410은 Programmable 제품으로 ERP/MES/CRM 등 자동화 생산 공장 시스템 환경에 최적의 사양을 지원해주는 다기능 제품입니다. Conveyer에서 생산된 제품들의 생산 수량을 파악하는 별도의 설비카운터 없이 센서 신호를 Digital Input Channel에 연결시켜주면 신호를 감지할 때 마다 카운터를 저장합니다. 또 PLC로 연결되어진 설비들의 각종 전기적 신호(Analog Signal)들과 배전반에 설치된 LCD모니터에 표기되는 각종 생산정보 Data들을 Serial Port로 연결하여 Network을 통해 Server나 Client로 전달하는 역할을 합니다. NetEye6410은 현재 Master-K, GLOFA 등 PLC에 설치환경에 최적화되어 있으며, HANYOUNG, YOKOGAWA 등 온도컨트롤러와도 최적화되어 자동화 공장 생산 환경에서 작동 중에 있습니다.



Factory Shop Floor Machine Digital Counter Contact Points

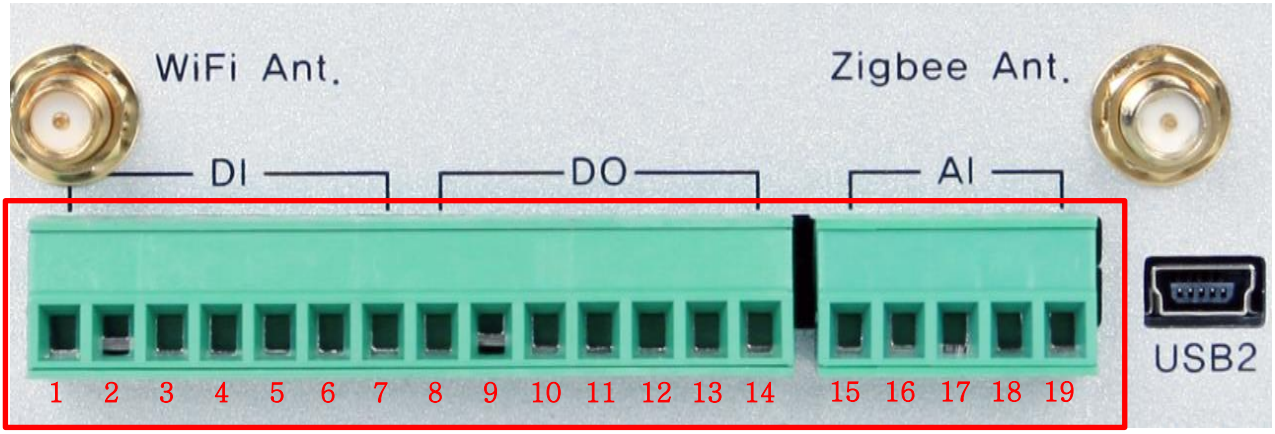
1-6. NetEye6410 v2.2 제품 사양

Items	Description
Resolution	12-bit
Input channel number	16
Input range	0~5V, 0~10V, 4~20mA
protection	Lightning, static
Accuracy	+-10%
Sample rate	200sample/second(16channels) 1000 sample/second(1 channel)
Output BUS	Ethernet/Wireless Lan (Wi-Fi)/RS232/RS422/RS485/Zigbee(옵션)
Power Input	+9V ~ 45V (AC/DC)
Power consumption	39mA@24VDC
Ambient temperature	-20 ~ 85℃ (-13 ~ 185°F)
Color	Blue/Silver
Size	115*90*43 mm

1-7. DI/DO/AI 구성도

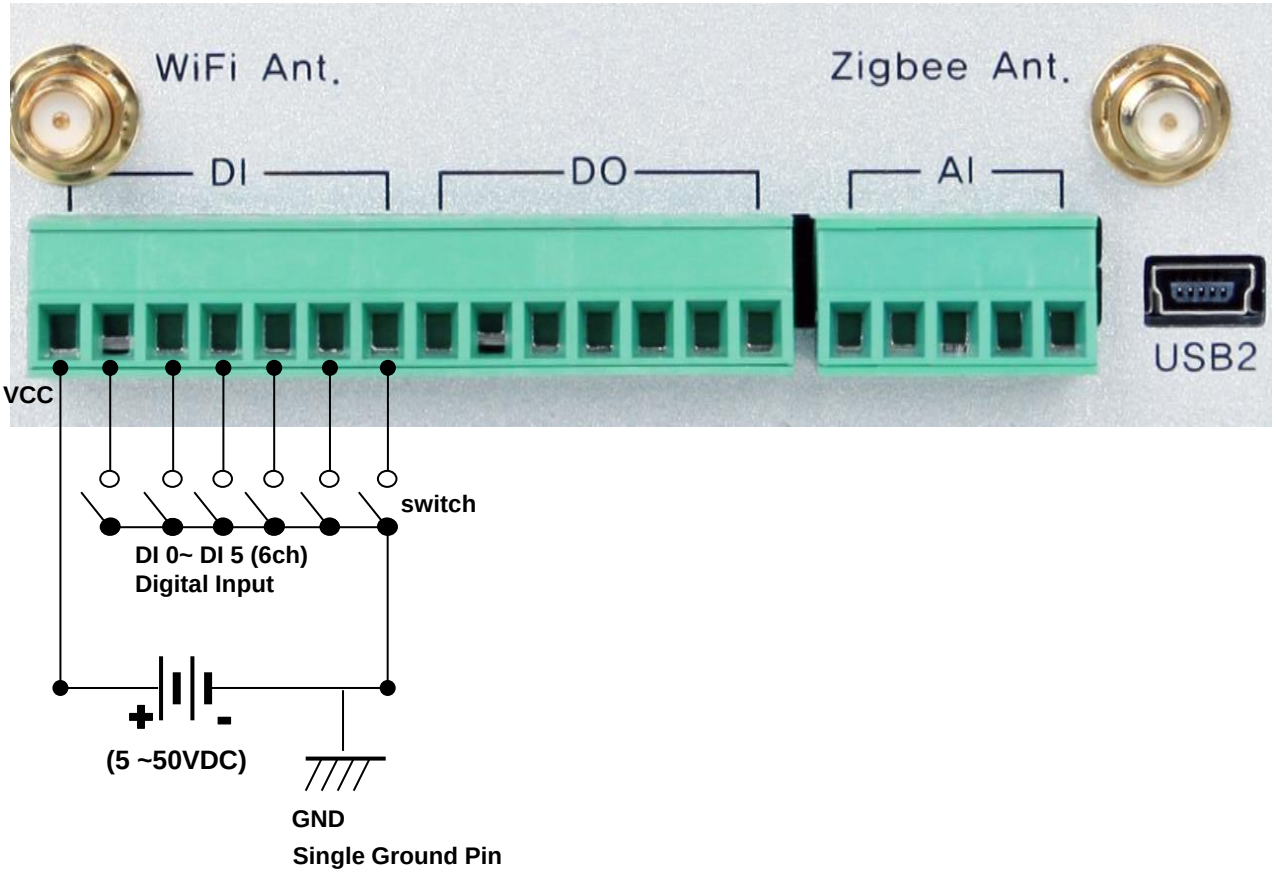
1-7-1. NetEye6410 v.2.2 Terminal Block Pin Array

(1) DI/DO VCC (+5V~+50V)
(2) DI 0 channel
(3) DI 1
(4) DI 2
(5) DI 3
(6) DI 4
(7) DI 5
(8) DO 0 channel
(9) DO 1
(10) DO 2
(11) DO 3
(12) DO 4
(13) DO 5
(14) DO GND
(15) AI 0 channel
(16) AI 1
(17) AI 2
(18) AI 3
(19) AI GND

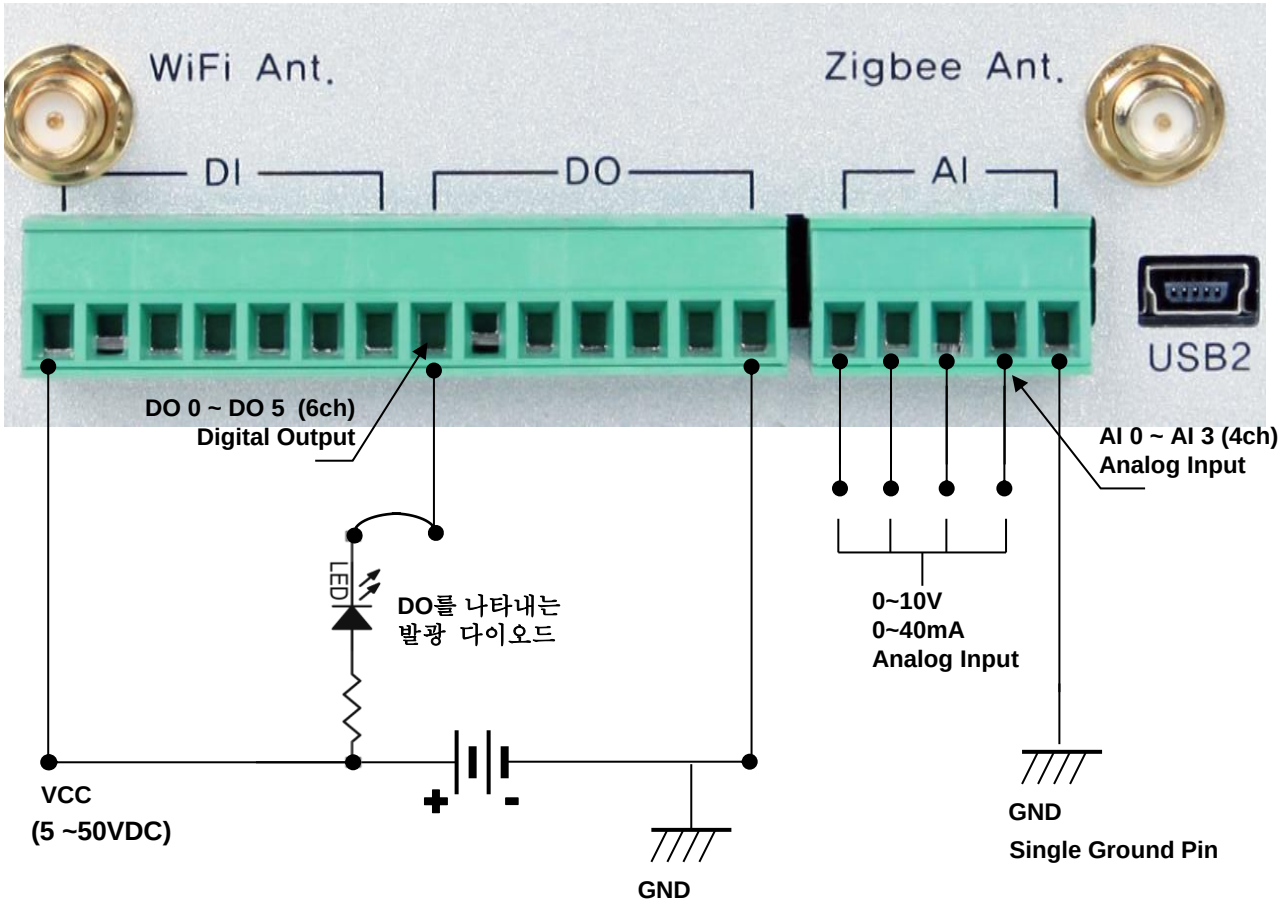


1-7-2. NetEye6410 DI/DO/AI 연결 구성

(1) Digital Input 결선



(2) Digital Output / Analog Input 결선



2. NetEye6410 v2.2 Getting Start

2-1. NetEye6410 Booting



① 전원 부분 9~24V 어댑터를 꽂으면 전원이 들어옵니다.

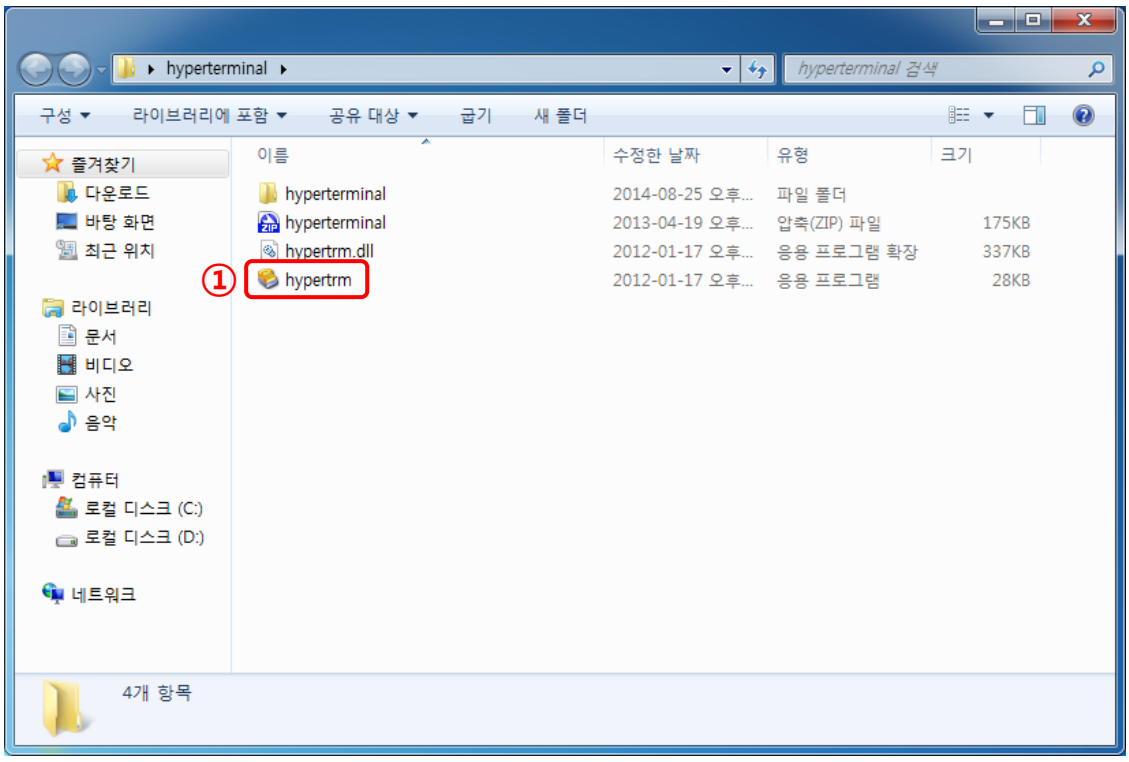


② 크로스 케이블과 USB Serial Converter 케이블을 연결 한 후 크로스 케이블을 NetEye6410 Monitoring 포트에 연결합니다.

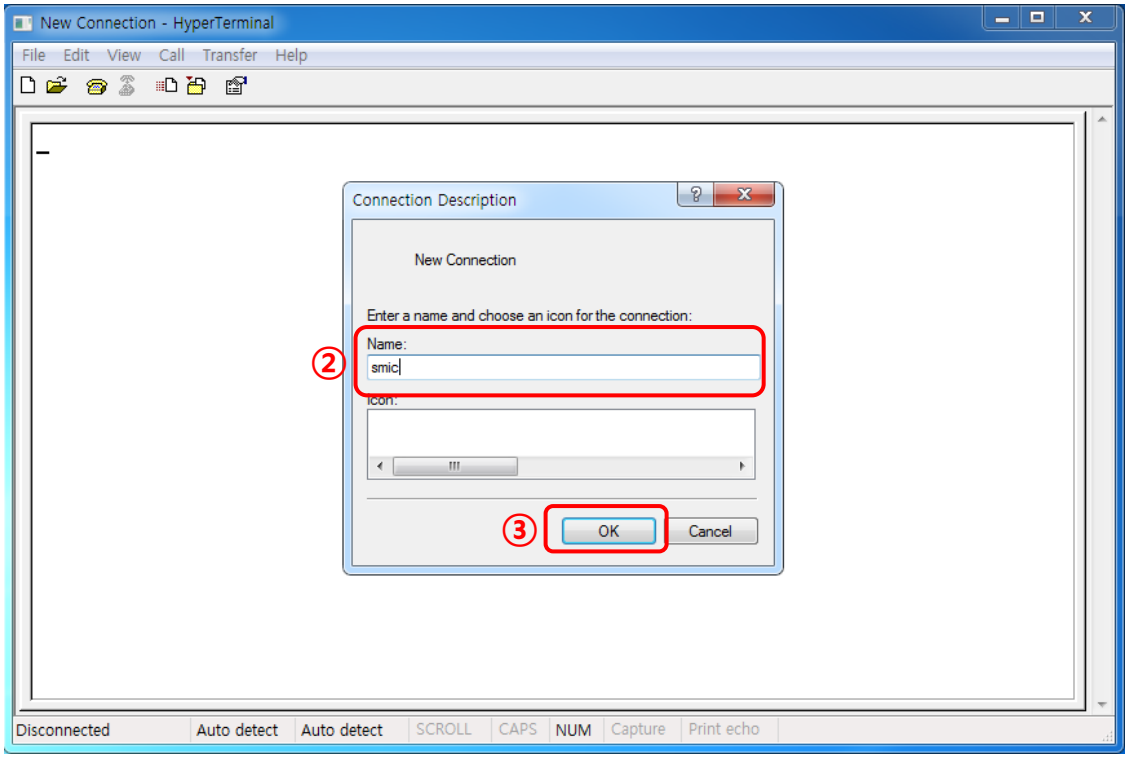
전원을 인가하면 NetEye6410 Booting이 됩니다.

2-2. NetEye6410과 PC와 연결하기

2-2-1. hyperterminal (하이퍼터미널) 사용방법

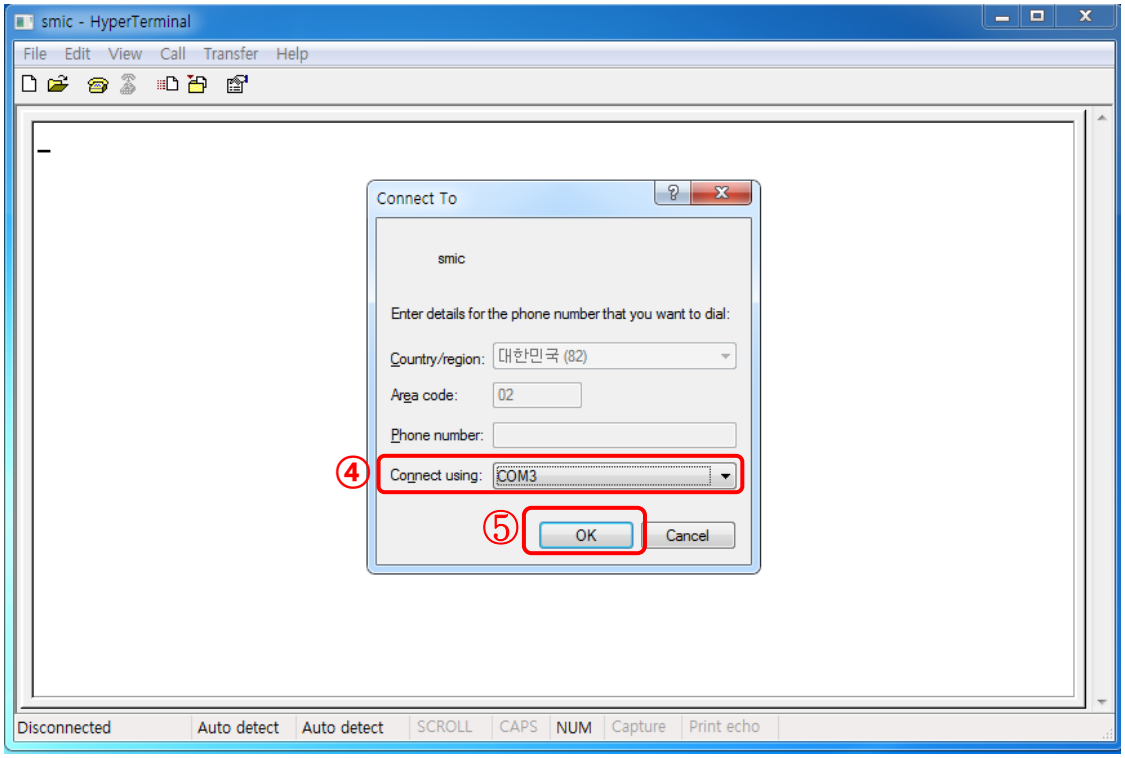


① 하이퍼터미널 프로그램을 더블 클릭하여 실행 시킵니다.



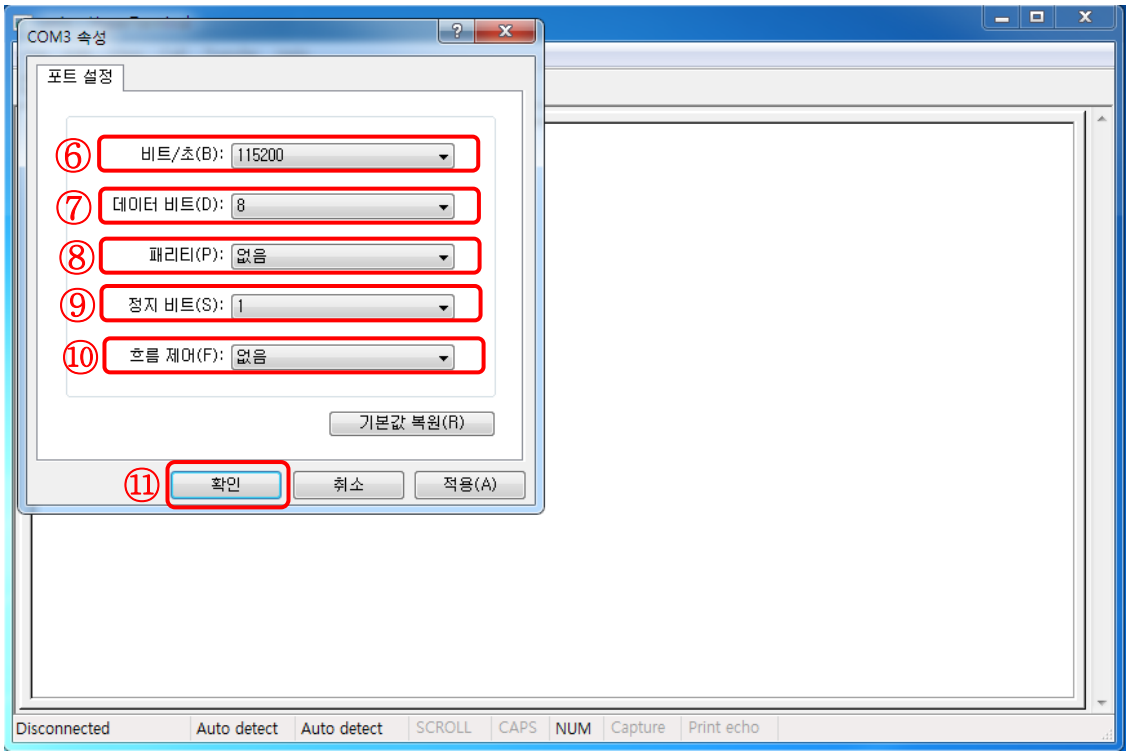
② Connection Description 창이 나타나면 파일이름을 입력합니다.

③ OK 버튼을 클릭합니다.



④ 사용자에 맞는 포트번호를 확인하여 설정합니다.

⑤ OK 버튼을 클릭합니다.



⑥ 비트/초(B) –> baud rate값이 115200 고정이므로 115200 으로 설정 합니다.

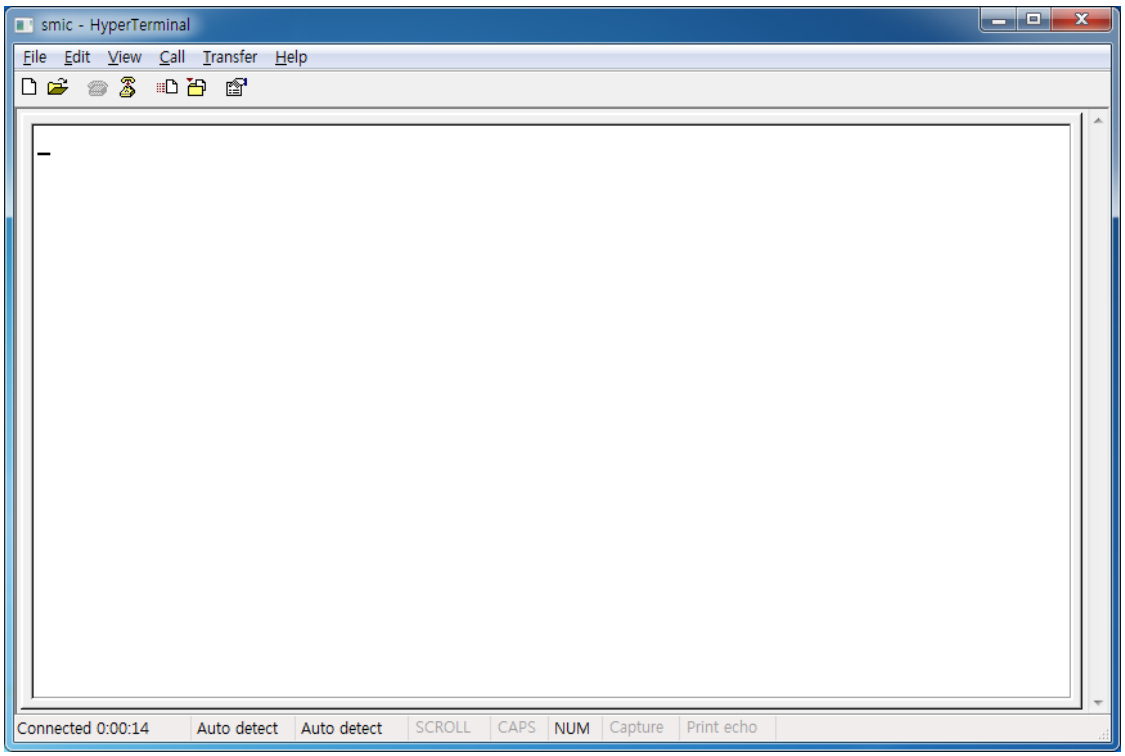
⑦ 데이터 비트(D) –> 기본 설정인 8 로 설정 합니다.

⑧ 패리티(P) –> 없음 으로 설정 합니다

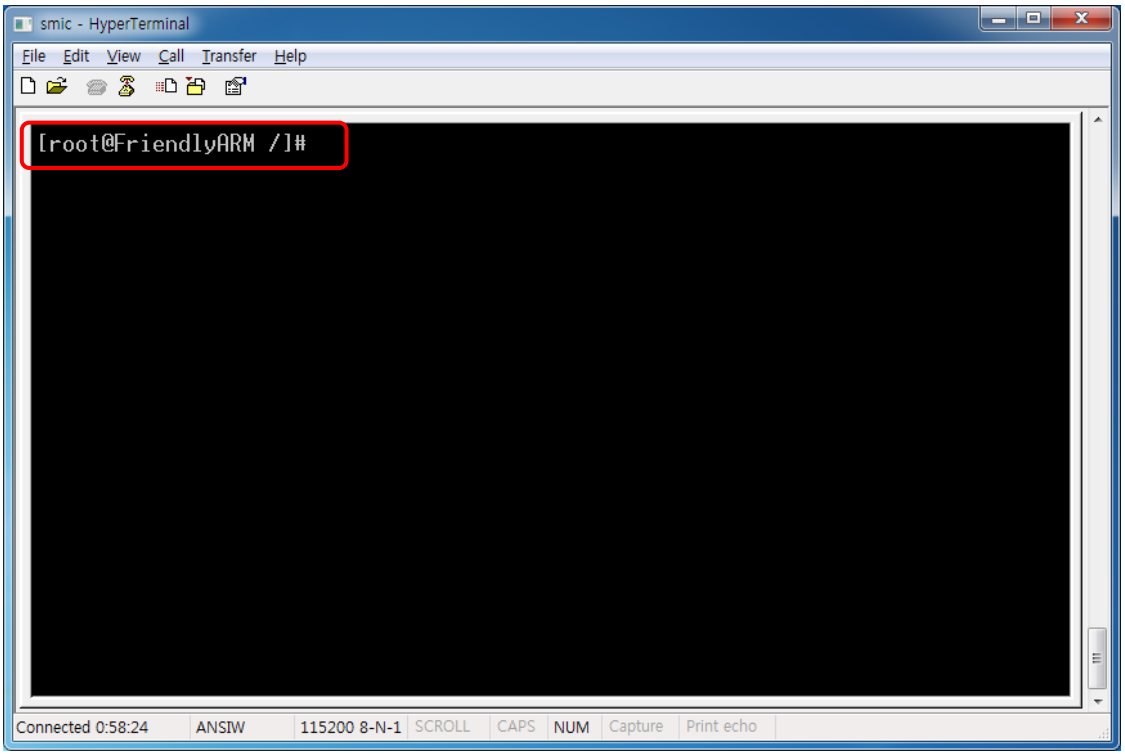
⑨ 정지 비트(S) –> 1 로 설정 합니다.

⑩ 흐름 제어(F) –> 없음 으로 설정 합니다

⑪ 확인 버튼을 클릭 합니다.



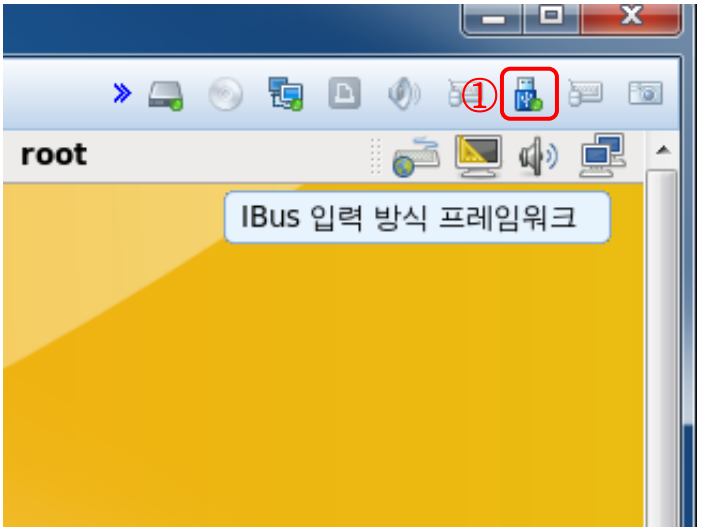
포트설정을 완료하면 위 화면과 같이 시리얼로 접속이 됩니다. 그 다음 Enter키를 한번 누릅니다.



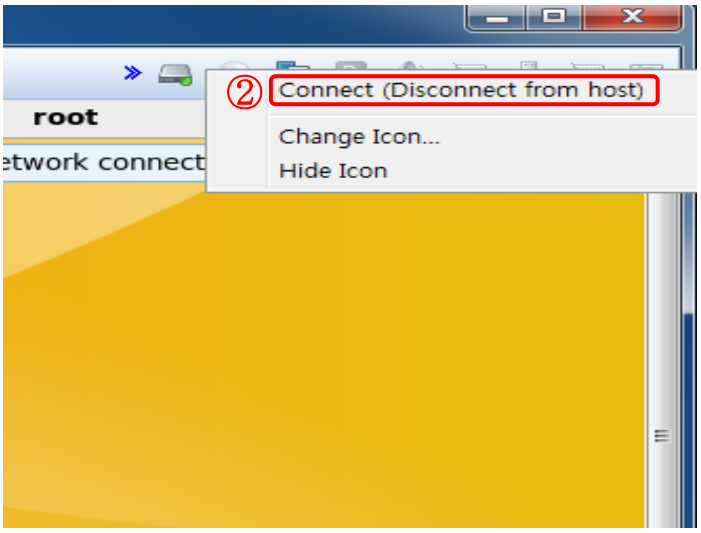
Enter를 누르고 위 화면과 같이 [root@FriendlyARM /]# 이 보이면 하이퍼터미널로 NetEye6410에 성공적으로 접속됩니다.

2-2-2. minicom(미니컴) 사용방법

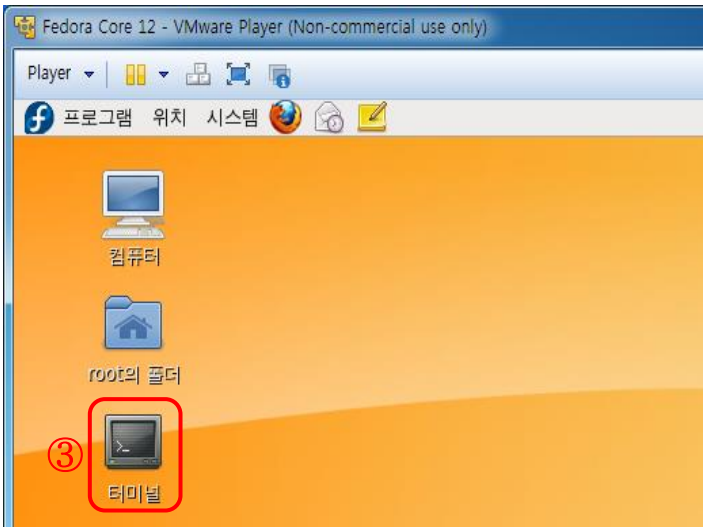
VMware를 실행을 시키면 그 다음으로 NetEye6410 전원을 켜고 Monitor Cable 연결을 합니다.



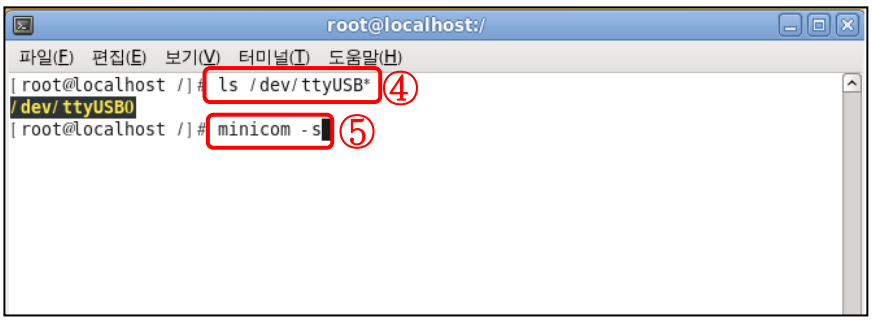
① 마우스 오른쪽 버튼을 클릭합니다.



② Connect 클릭합니다.

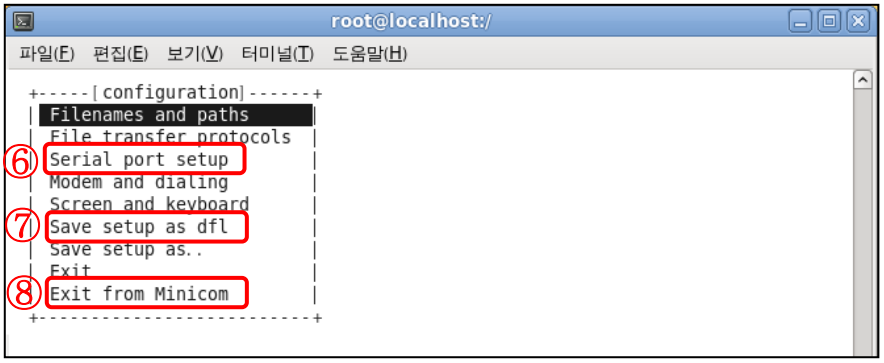


③ 터미널 프로그램을 실행합니다.

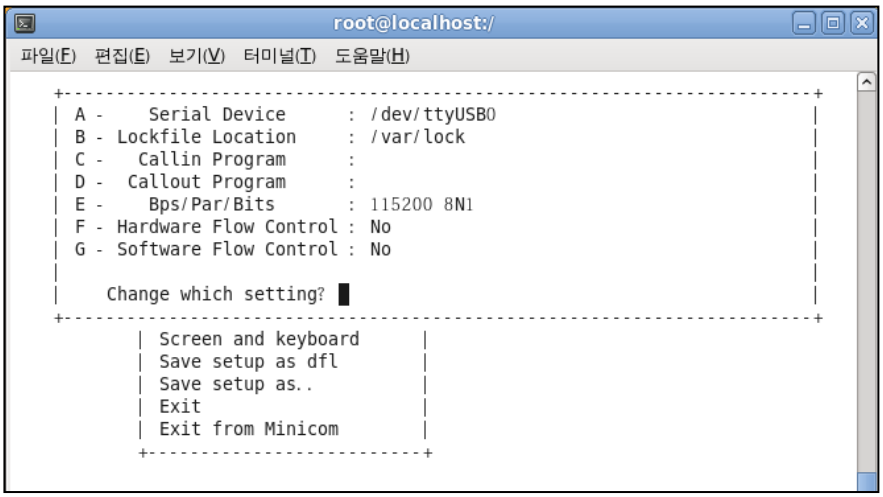


④ ls /dev/ttyUSB* 를 입력해서 사용중인 USB 넘버를 확인합니다.

⑤ minicom –s 를 입력해 serial 통신 환경을 설정합니다.

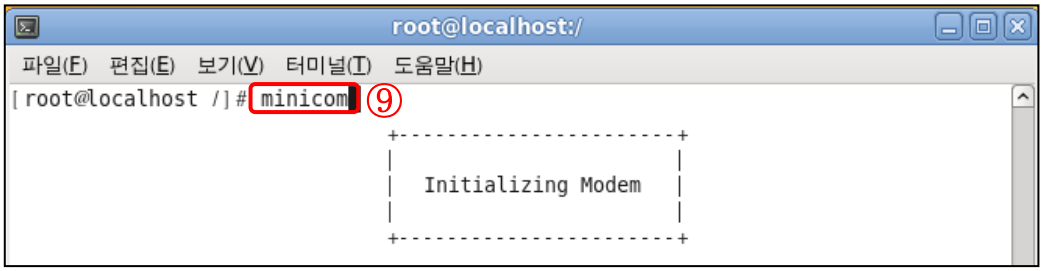


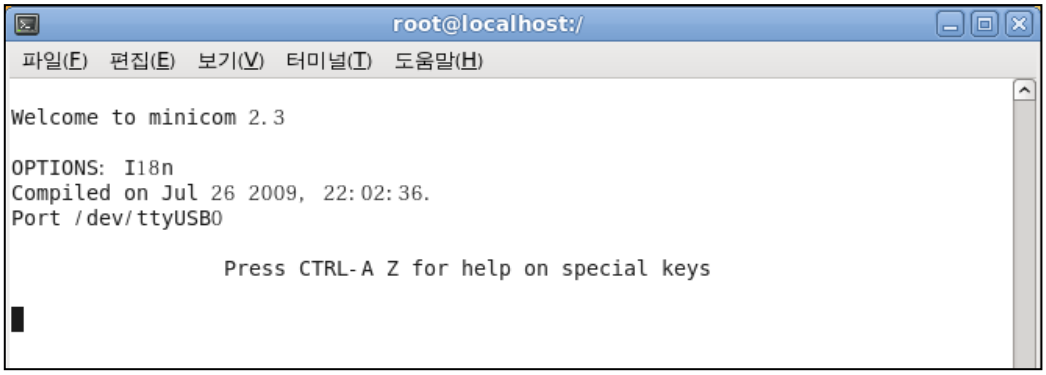
⑥ serial port setup 메뉴로 들어가 NetEye6410에 맞는 환경으로 변경합니다.
아래 그림과 같이 설정합니다.



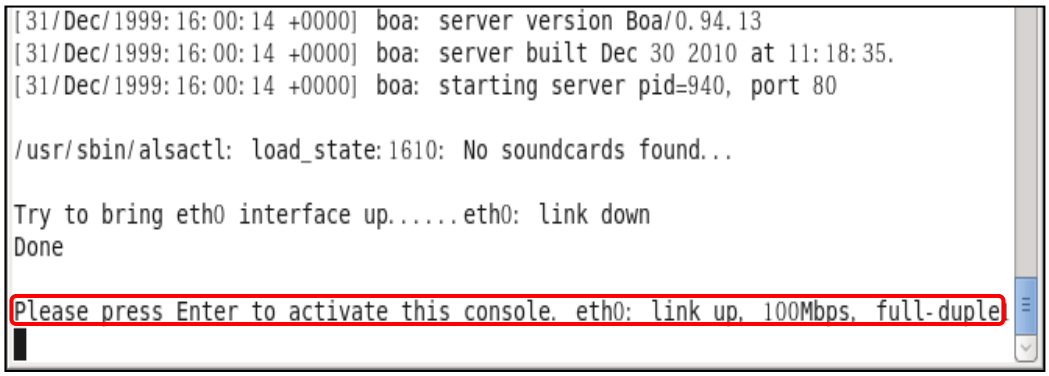
⑦ 설정이 완료되면 Save setup as dfl 메뉴로 들어가 설정 값을 저장합니다.

⑧ Exit from Minicom 을 눌러 설정합니다.

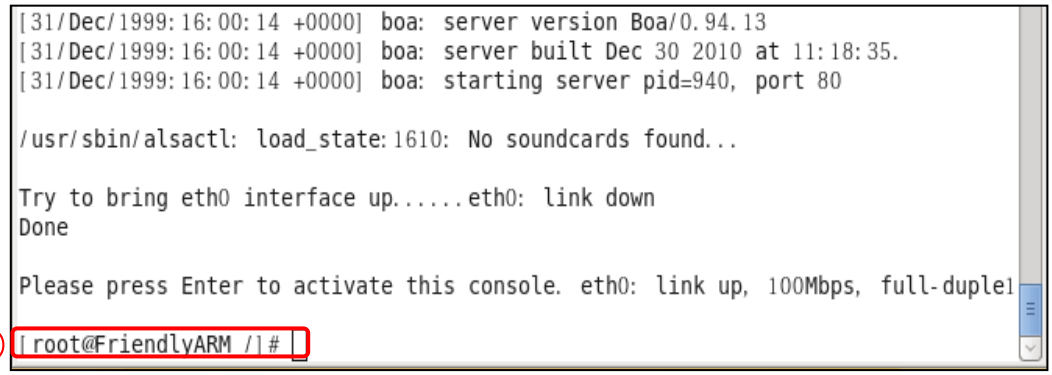




⑨ 모든 설정을 완료하였으므로 NetEye6410 Board 에 minicom으로 접속 하기 위해 minicom을 입력 합니다.

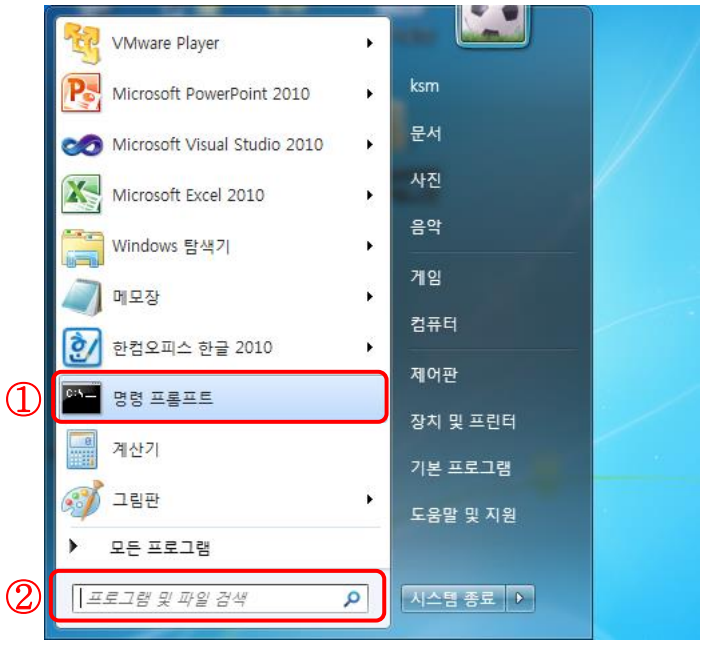


⑩ Enter를 누릅니다.

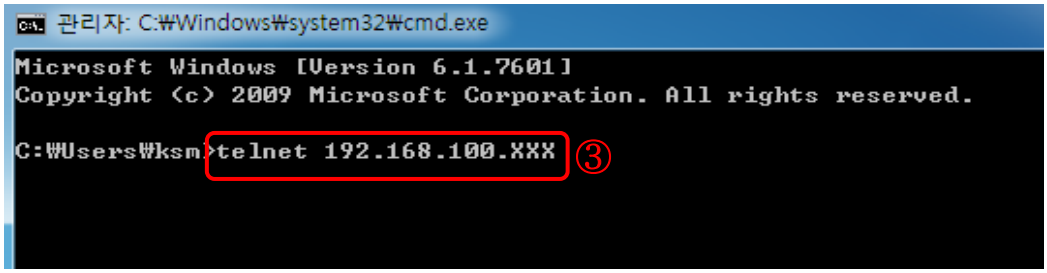


⑪ Enter를 누르고 위 화면과 같이 [root@FriendlyARM /]# 이 보이면 minicom으로 NetEye6410에 성공적으로 접속됩니다

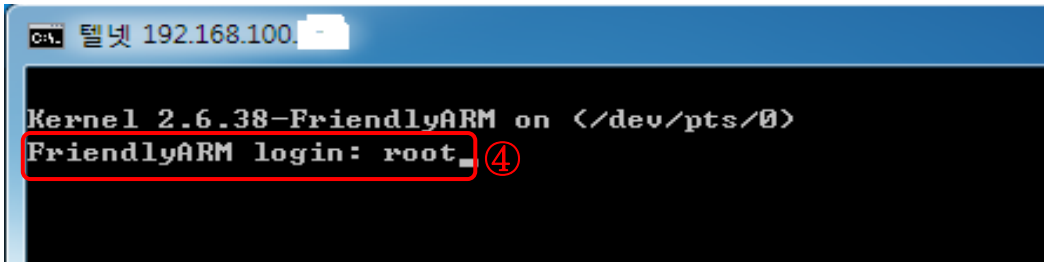
2-2-3. telnet(텔넷) 사용방법



시작 프로그램에서 ① 명령 프롬프트나 ② 프로그램 및 파일 검색에서 cmd를 입력 합니다.

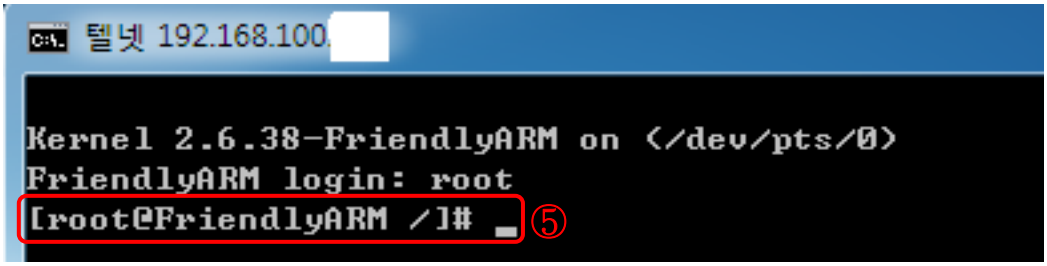


③ telnet 192.168.100.XXX NetEye6410에 맞는 IP주소를 입력합니다.
(IP주소설정은 "2-5NetEye6410 LAN 사용하기"를 참고하시기 바랍니다.)



telnet(텔넷)으로 NetEye6410에 접속합니다.

④ 리눅스에서 사용하는 사용자 이름(root)을 입력합니다.



⑤ [root@FriendlyARM /]# 이 보이면 telnet(텔넷)으로 NetEye6410에 성공적으로 접속됩니다.

2-3. NetEye6410의 Wi-Fi 모듈 사용하기

2-3-1. 설정 모드

1) AP Mode (Access Point Mode)

AP Mode는 초기 설정 하기 위한 무선 모드입니다. 초기에 AP Mode로 되어 있고 SSID는 SMIC21_AP, 보안은 설정이 되어있지 않으며 IP address는 10.10.100.254로 설정되어 있습니다. 우선 NetEye6410에 전원을 인가한 후 PC에서 SMIC21_AP 에 해당하는 무선 네트워크를 찾아 연결한 후 web browser를 통하여 10.10.100.254로 접속하여 해당 Parameter들을 변경 합니다.

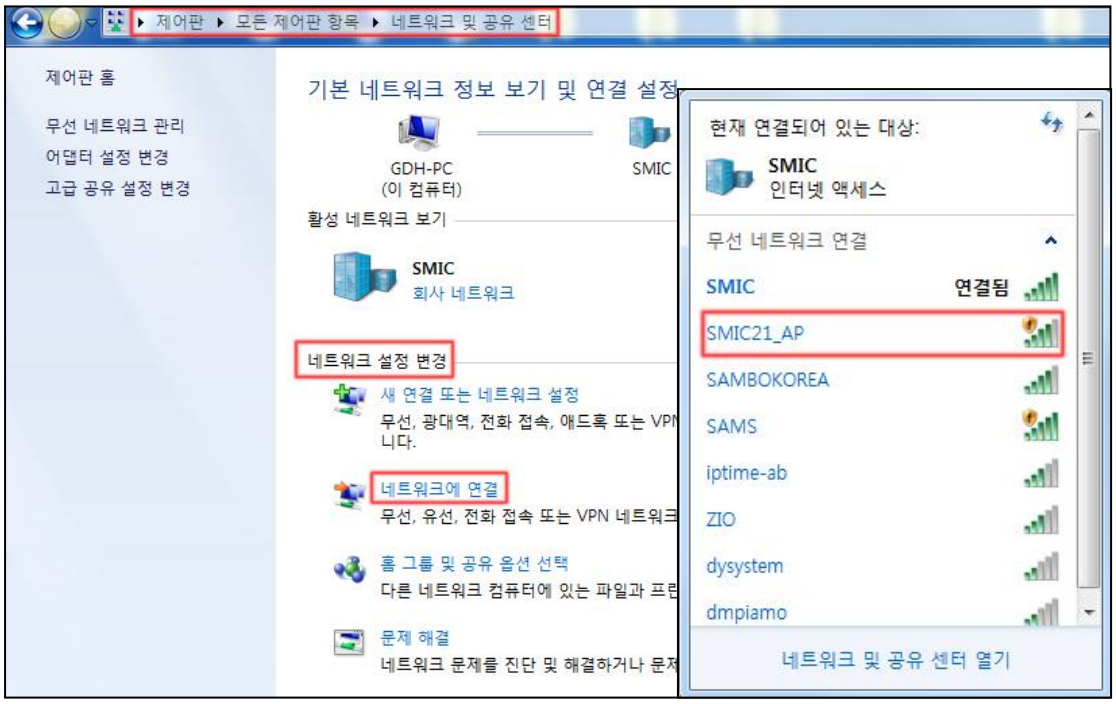
2) STA Mode (Station Mode)

STA는 AP mode의 클라이언트 모드입니다. AP Mode로 접속하여 연결하고자 하는 AP의 SSID, Security Mode, IP address등을 설정합니다. AP Mode로 접속하여 STA Interface Setting, Application으로 Parameters들을 변경 한 후 최종적으로 STA Mode를 선택해야 합니다.

2-3-2. STA모드 설정 과정 안내

Step 1. NetEye6410에 전원을 연결합니다.

Step 2. 제어판 -> 모든 제어판 항목 -> 네트워크 및 공유 센터 -> 네트워크 설정 변경 -> 네트워크에 연결 - > SMIC21_AP 접속을 합니다.



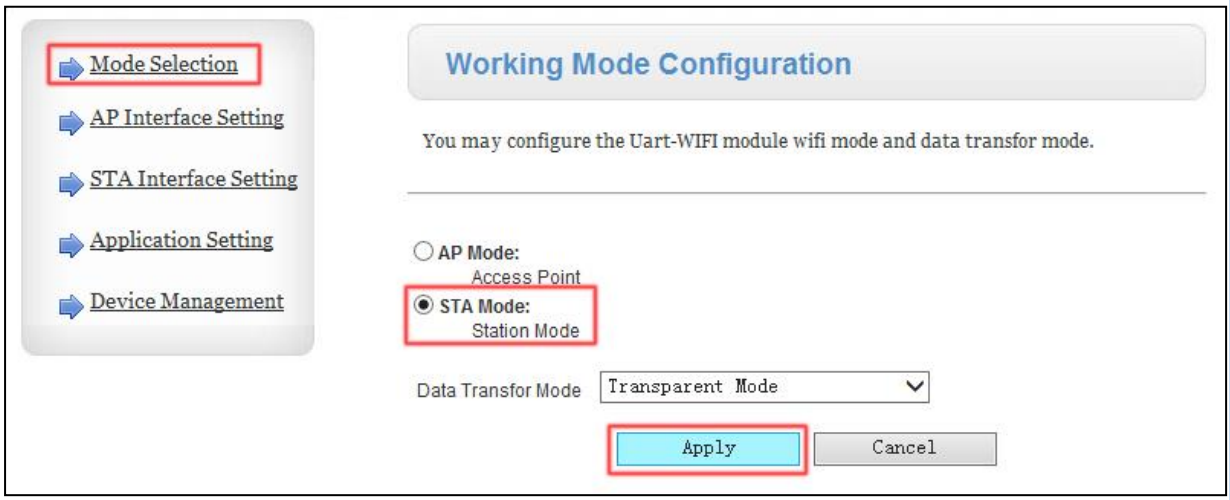
Step 3. Internet Explorer를 열고 주소 창에 http://10.10.100.254에 접속을 합니다.

사용자 ID -> admin 암호 -> admin

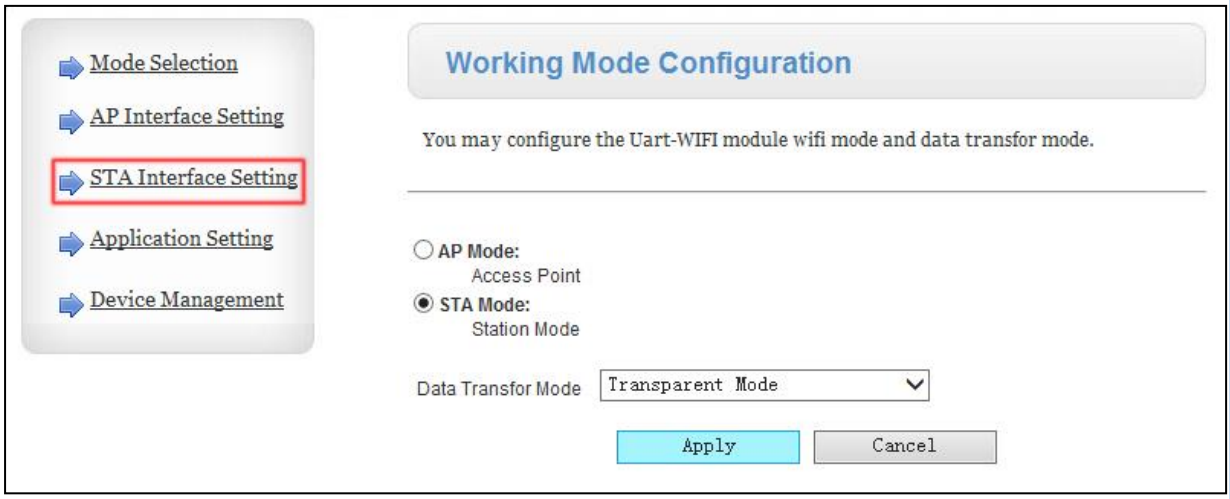


* AP Interface Setting 은 수정하지 않습니다.

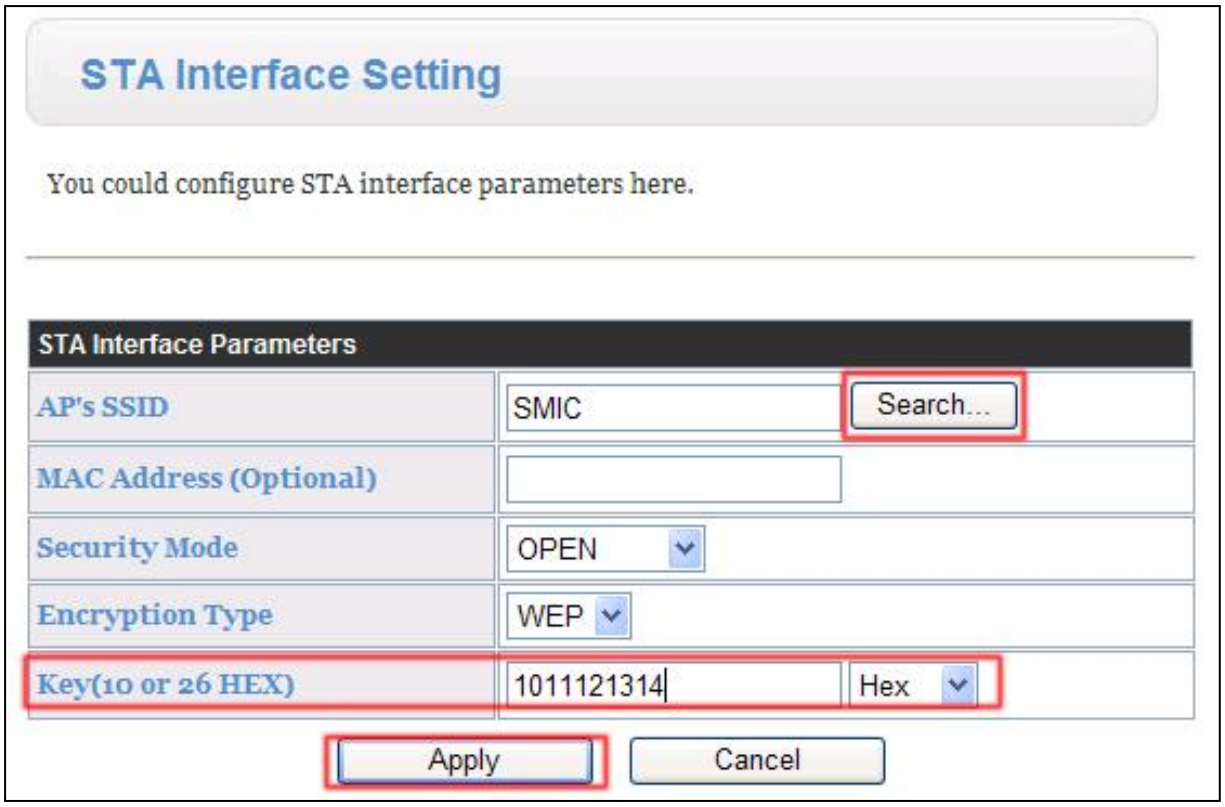
Step 4. 모드 선택 메뉴에서 STA 모드로 선택하고 Apply 버튼 클릭합니다.



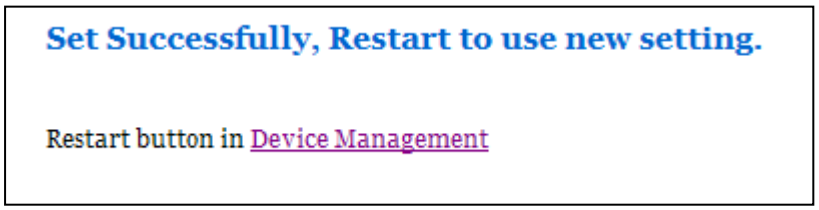
Step 5. 모드 설정이 끝나면 STA Interface Setting 메뉴를 클릭합니다.



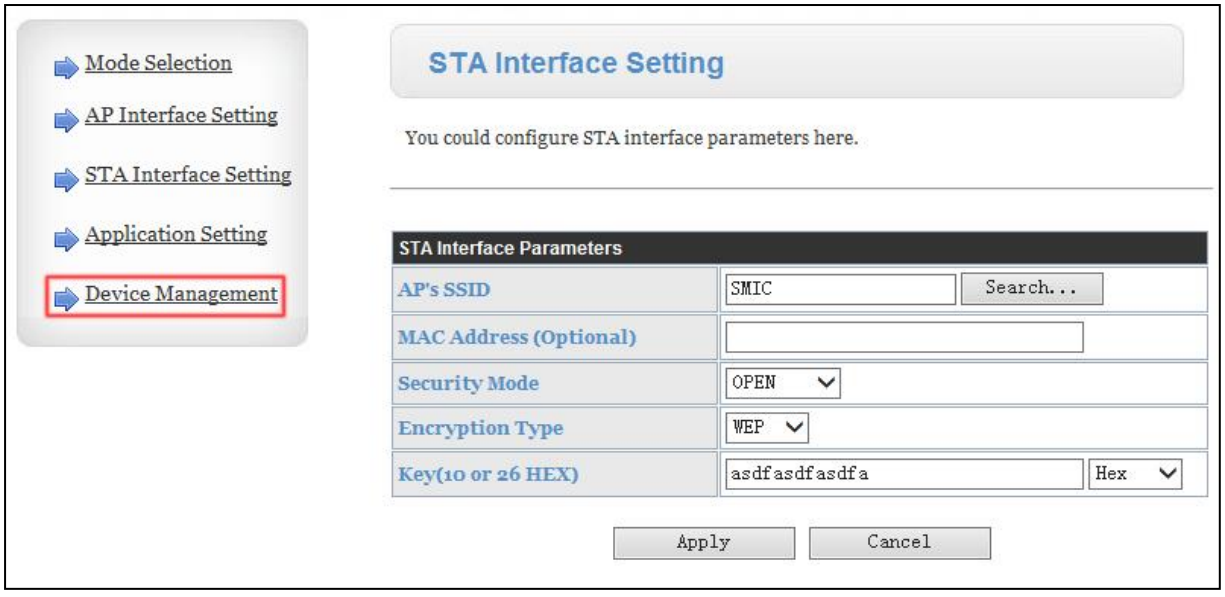
Step 6. STA Interface Setting 메뉴에서 Search 버튼을 클릭합니다. 연결하시려는 무선 AP 목록이 나옵니다. 목록 중에 %가 높을수록 가장 가까이 있는 무선AP입니다. 연결하려는 SSID가 맞는지 확인 하시고 선택을 합니다. SSID선택을 하셨다면 아래쪽에 보안설정을 하셔야 되는데 무선AP의 보안설정대로 입력을 하시고 Apply 버튼 클릭합니다.



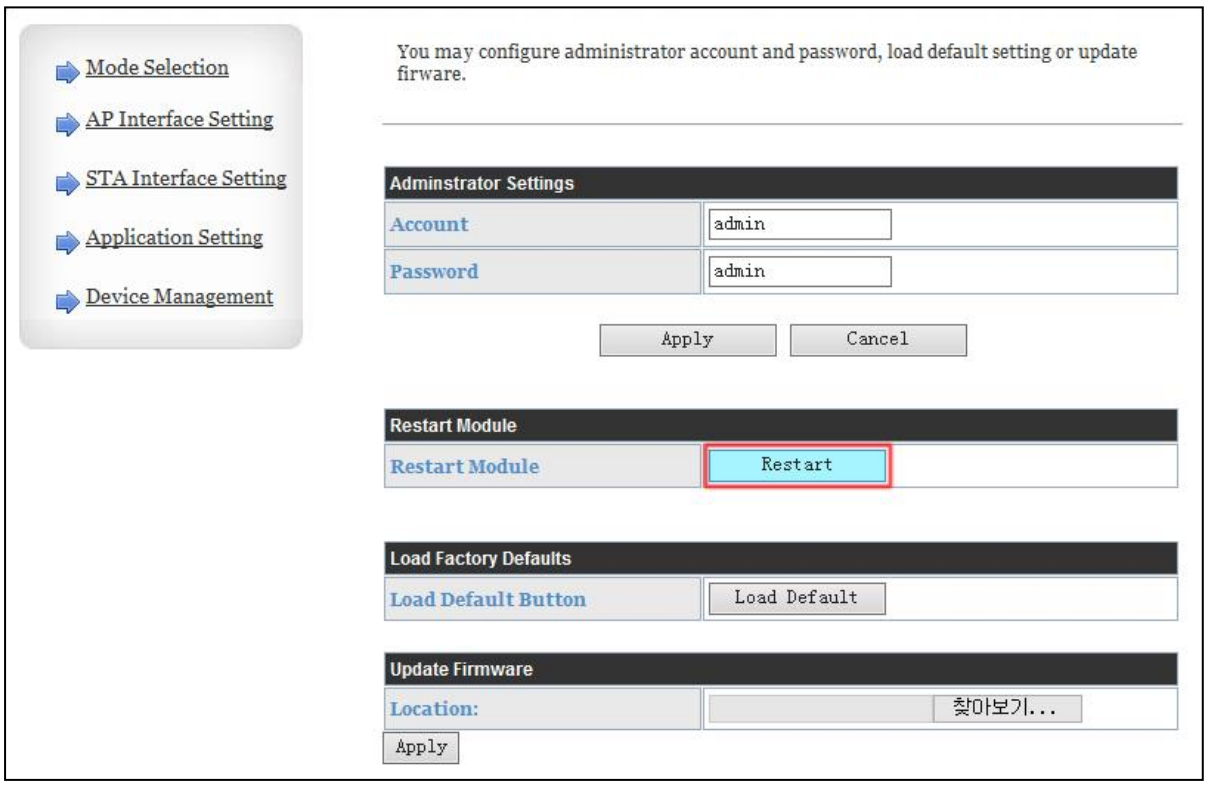
Step 7. 설정이 완료되면 성공 메시지 창이 나타납니다.



Step 8. 메시지 창이 나타나면 Device Management 메뉴 버튼을 클릭합니다.



Step 9. Device Management 메뉴에서 Restart 버튼을 클릭합니다.



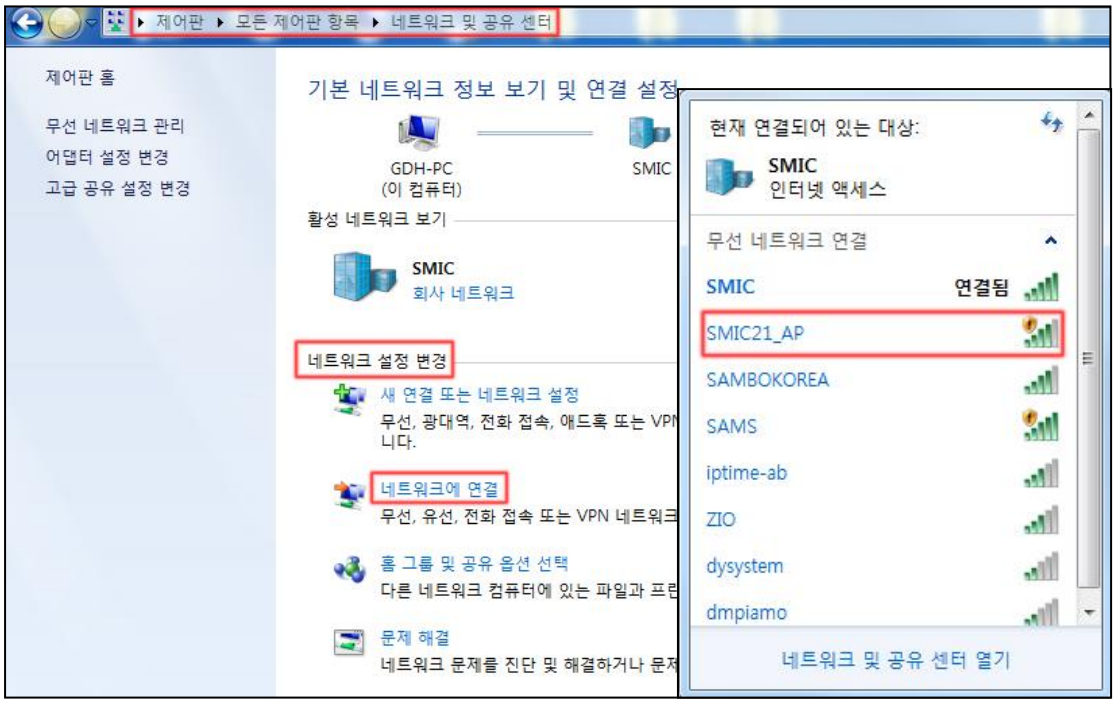
Step 10. 재부팅 메시지를 확인할 수 있습니다. 30초 정도 기다리시면 설정 이 완료 됩니다.



2-3-3. AP모드 설정 과정 안내

Step 1. NetEye6410에 전원을 연결합니다.

Step 2. 제어판 -> 모든 제어판 항목 -> 네트워크 및 공유 센터 -> 네트워크 설정 변경 -> 네트워크에 연결 - > SMIC21_AP 접속을 합니다.

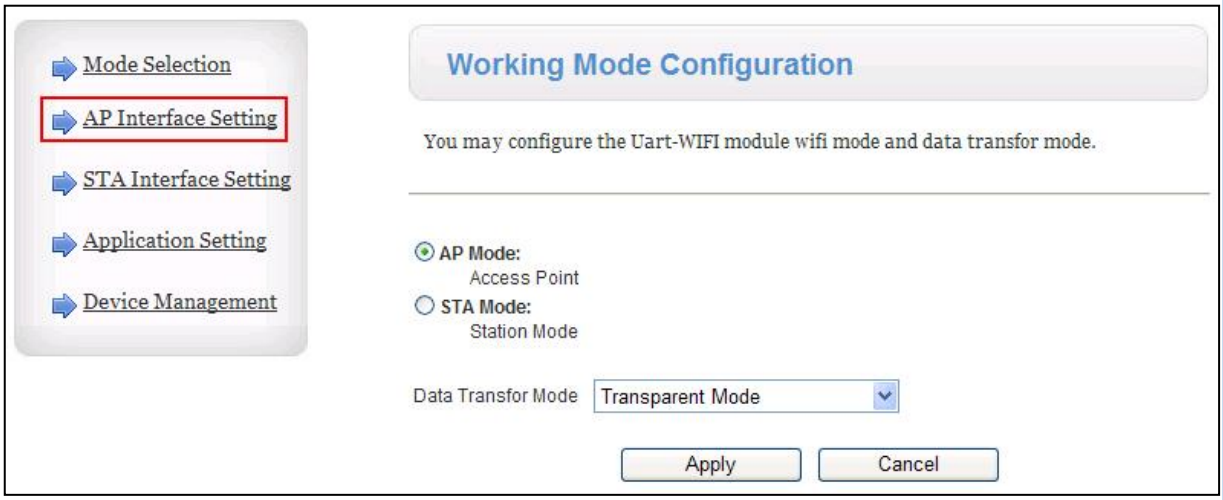


Step 3. Internet Explorer를 열고 주소 창에 http://10.10.100.254에 접속을 합니다.

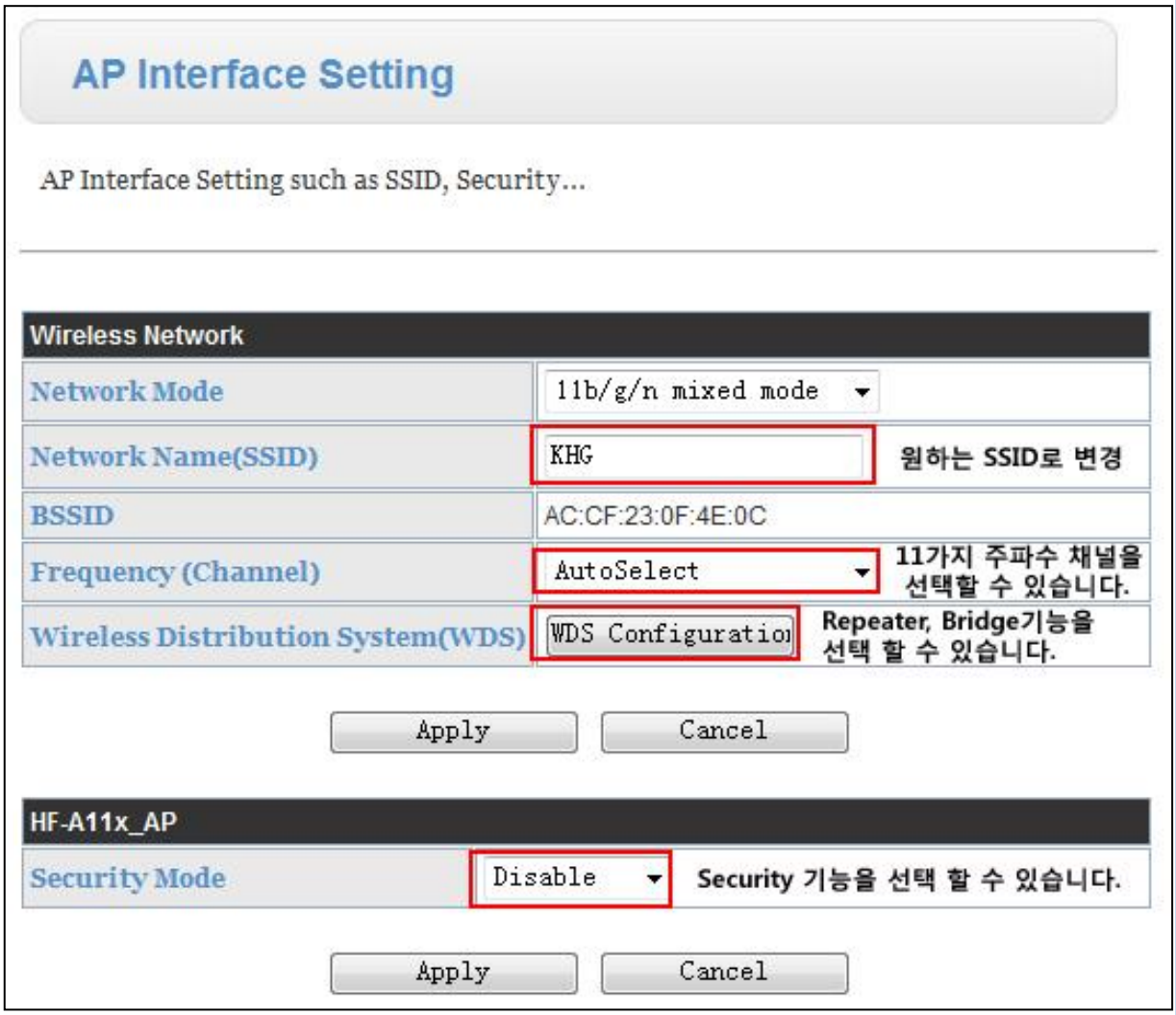
사용자 ID -> admin 암호 -> admin



Step 4. AP Interface Setting 버튼 클릭합니다.



Step 5. AP Interface Setting 에서 설정 값을 변경합니다.

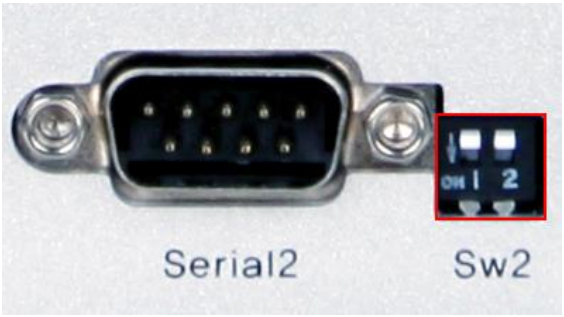


Step 6. 설정이 끝났습니다. 독립적인 AP로 사용 할 수 있습니다.

2-4. NetEye6410의 Serial Port 사용하기

2-4-1. 딥 스위치(DIP switch)로 RS232 / 485 / 422 Serial Port 확인 방법.

Step 1. 딥 스위치(DIP switch) 1번 , 2번 2개가 올라가 있으면 RS232 입니다.



Step 2. 딥 스위치(DIP switch) 1번을 내리면 RS485 입니다.



Step 3. 딥 스위치(DIP switch) 2번을 내리면 RS422 입니다.



Step 4. 딥 스위치(DIP switch) 1번, 2번 2개가 내려져 있으면 사용 안함 입니다.



2-4-2. 9Pin(핀) Configuration

Pin No.	Signal	Description		
		RS-232	RS-422*	RS-485*
1	RX+		RX+	
2	RXD	RX		
3	TXD	TX		
4	TX+		TX+	TX+
5	GND	GND	GND	GND
6	NC			
7	RX-		RX-	
8	TX-		TX-	TX-
9	NC			

2-5. NetEye6410의 LAN 사용하기

2-5-1. LAN 케이블 배열 순서

Step 1.다이렉트(Direct) 케이블



컴퓨터와 허브간을 연결할 때 사용합니다. 공유기, 허브 등에 연결하실 때 사용하는 케이블입니다.
(띠주,주,띠녹,파,띠파,녹,띠갈,갈)

Step 2.크로스(Cross) 케이블



허브없이 컴퓨터끼리 1:1로 직접 연결할 때 사용합니다.
(띠녹,녹,띠주,파,띠파,주,띠갈,갈)

1 -> Tx+

2 -> Tx-

3 -> Rx+

6 -> Rx-

Tx : Transmit Data 데이터 송신

Rx : Receive Data, 데이터 수신

2-5-2. IP address 바꾸는 방법

Step 1. 우선 IP를 확인 합니다. IP확인 방법은 ifconfig를 입력합니다.

```
[root@FriendlyARM /]# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:90:00:A0:90:90
          inet addr:192.168.1.230  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:222 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:20781 (20.2 KiB)  TX bytes:0 (0.0 B)
          Interrupt:108 Base address:0x2000
```

Step 2. IP를 확인 후 변경할 IP를 입력합니다. 입력방법은 ifconfig eth0를 입력한 후 변경할 IP주소를 입력을 합니다. Ex) ifconfig eth0 192.168.100.230

```
[root@FriendlyARM /]# ifconfig eth0 192.168.100.230
```

Step 3. 변경이 되었는지 확인 방법은 ifconfig 입력을 합니다. 변경된 IP를 확인 하실 수 있습니다.

```
[root@FriendlyARM /]# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:90:00:A0:90:90
          inet addr:192.168.100.230  Bcast:192.168.100.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3642 errors:0 dropped:3 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:337117 (329.2 KiB)  TX bytes:0 (0.0 B)
          Interrupt:108 Base address:0x2000
```

2-5-3. Mac address 바꾸는 방법

Step 1. 먼저 변경할 Mac address를 확인합니다.

```
eth0      Link encap:Ethernet  HWaddr 08:90:00:A0:90:90
```

Step 2. 주소까지 확인 했으면 순서 대로 명령어를 입력 하십시오. ①ifconfig eth0 down ②ifconfig eth0 hw ether 변경할 Mac address 입력하십시오. ③ifconfig eth0 up (down -> Ethernet사용 안 함 up - > Ethernet사용 함)

```
[root@FriendlyARM /]#①ifconfig eth0 down
[root@FriendlyARM /]#②ifconfig eth0 hw ether 08:90:00:A0:90:95
[root@FriendlyARM /]#③ifconfig eth0 up
```

Step 3. ifconfig 입력 후 변경 된 Mac address를 확인 하십시오.

```
eth0      Link encap:Ethernet  HWaddr 08:90:00:A0:90:95
          inet addr:192.168.1.230  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:108 Base address:0x2000
```

2-6. USB 사용하기

2-6-1. USB 연결



2-Port USB 2.0 Host

2-6-2. 시스템 접근

내부 저장소가 없는 경우

내부 저장소를 별도로 구매하지 않은 경우, USB 이동식 저장 장치를 NetEye6410에 연결하면 자동으로 저장 장치를 인식하여 /udisk 디렉토리에 마운트 됩니다.

내부 저장소가 있는 경우

제품 구매 시 내부 확장 저장소(SSD) 를 같이 구매하신 경우, USB를 이동식 저장 장치를 연결하여 사용해야 할 경우 명령어를 통해 수동으로 마운트 하십시오.

```
[root@FriendlyARM /]# du -sh /udisk
758.0M  /udisk
[root@FriendlyARM /]# ls /mnt
[root@FriendlyARM /]# du -sh /mnt
0       /mnt
[root@FriendlyARM /]# mount -t vfat /dev/sdb /mnt
[root@FriendlyARM /]# du -sh /mnt
2.5G    /mnt
[root@FriendlyARM /]# umount /mnt
[root@FriendlyARM /]# ls /mnt
[root@FriendlyARM /]# du -sh /mnt
0       /mnt
[root@FriendlyARM /]#
```

- 내부 확장 저장소가 존재 하므로 /udisk 디렉토리를 확인하면 데이터가 들어 있습니다.
- /mnt 디렉토리가 비어있습니다.
- mount 명령어를 이용해 /mnt 디렉토리에 USB 저장 장치를 마운트 합니다.
- du 명령어를 이용해 정상적으로 마운트 되었는지 확인합니다.
- 사용이 끝나면 umount 명령어를 이용해 USB 저장 장치를 언마운트 합니다.

USB 이동식 저장 장치의 파일 시스템은 FAT32를 권장 합니다.

2-7. SD Card 사용하기

2-7-1. SD Card 연결



2-7-2. 시스템 접근

SD Card를 NetEye6410에 연결 하면 자동으로 인식하여 /sdcard 디렉토리를 생성하여 마운트 합니다.

SD Card 파일 시스템은 FAT32를 권장 합니다.

3. NetEye6410 기본 사항

3-1. NetEye6410의 직렬 통신 사용하기

3-1-1. NetEye6410 직렬 통신 소개

NetEye6410은 직렬 포트(DP9) 3개를 이용해 통신이 가능합니다. 직렬 통신을 위한 장치를 물리적으로 연결한 후에, 통신에 필요한 환경 변수를 설정 하십시오. 직렬 통신을 위한 환경 변수는 Baud Rate, Data Bits, Parity, Stop Bits, Flow Control 이 있습니다.

Buad Rate	직렬 통신의 전송 속도를 나타냅니다.
Data Bits	한 바이트에서 사용할 비트수를 설정합니다.
Parity	직렬 통신시 송·수신된 데이터의 에러 유무를 나타내는 비트입니다.
Stop Bits	데이터의 마지막을 나타내는 비트입니다.
Flow Control	흐름제어 사용여부를 설정합니다.

3-1-2. 직렬 통신을 C 표준 라이브러리 함수

NetEye6410에서 직렬 통신은 프로그램으로 구현 됩니다. NetEye6410에 포팅(Porting) 된 Embedded Linux에서는 장치를 파일로 인식하기 때문에 직렬 통신을 사용하기 위해서 Open(), Read(), Write(), Close() 함수를 사용합니다. Open함수를 사용하여 장치를 연결한 후 터미널 장치의 속성을 설정하기 위한 tcsetattr()함수를 사용합니다.

1) open( ) 함수

장치를 사용하기 위해 파일(장치)을 열기합니다.

Header	fcntl.h
형태	int open (const char *FILENAME, int FLAGS[,mode_t MODE])
인수	FILENAME : 경로가 포함된 대상 파일 이름입니다. FLAGS : 파일에 대한 열기 옵션입니다. [,mode_t MODE] : O_CREAT사용시 생성될 때 지정되는 파일 접근 권한입니다.
반환	성공시 파일 디스크립터의 양의 정수 값, 실패시 -1을 반환합니다.

```
1   #include <fcntl.h>
2
3   int fd;
4
5   fd = open("/dev/ttySAC1", O_WRONLY | O_CREAT | O_EXCL, 0644);
6
7   if (-1 ==  fd )
8   {
9      printf( "Open 실패");
10     exit( 1) ;
11  }
```

- 5행 : /dev/ttySAC1은 직렬 포트 1번 입니다.

2) read ( ) 함수

해당 직렬 통신 장치로부터 데이터를 수신합니다.

Header	unistd.h
형태	ssize_t read (int fd, void *buffer, size_t nbytes)
인수	fd : 직렬 통신 장치의 파일 디스크립터입니다. buffer : 데이터를 수신하여 저장할 버퍼를 설정합니다. nbytes : 버퍼의 크기입니다.
반환	성공시 읽어들인 바이트 수를, 실패시 -1을 반환합니다.

```
1  #include <unistd.h>
2
3  #define BUFF_SIZE 1024
4
5  char buff[BUFF_SIZE];
6  int fd;
7
8  read( fd, buff, BUFF_SIZE);
```

3) write ( ) 함수

해당 직렬 통신 장치로 데이터를 송신합니다.

Header	unistd.h
형태	ssize_t write (int fd, const void *buf, size_t n)
인수	fd : 직렬 통신 장치의 파일 디스크립터입니다. buffer : 송신할 데이터가 담긴 버퍼입니다. nbytes : 버퍼의 크기입니다.
반환	성공시 쓰기를 한 바이트 수를, 실패시 -1을 반환합니다.

```
1  #include <unistd.h>
2
3  char *temp = "NetEye6410";
4  int fd;
5  write( fd, temp, strlen( temp));
```

4) close ( ) 함수

해당 직렬 장치의 연결을 종료합니다.

Header	unistd.h
형태	int close(int fd)
인수	fd : 연결을 종료할 직렬 통신 장치의 파일 디스크립터입니다.
반환	정상으로 실행되었다면 0을, 실패하면 -1을 반환합니다

```
1  #include <unistd.h>
2
3  int fd;
4  close( fd);
```

4) tcsetattr ( ) 함수

파일 디스크립터와 연관된 터미널 디바이스의 속성을 설정한다.

Header	termios.h
형태	int tcsetattr(int fd, int when, const struct termios *termios_p)
인수	fd : 속성을 변경할 장치의 파일 디스크립터입니다. when : 설정을 언제 할지 나타냅니다. termios_p : 터미널 디바이스의 환경 변수 구조체입니다.
반환	정상으로 실행되었다면 0을, 실패하면 -1을 반환합니다

```
1  #include <termios.h>
2
3  struct termios newtio;
4  -open()...-
5
6  memset(&newtio, 0, sizeof(newtio));
7  newtio.c_iflag = IGNPAR;
8  newtio.c_oflag = 0;
9  newtio.c_cflag = CS8|CLOCAL|CREAD;
10 newtio.c_cflag |= B57600;
11 newtio.c_lflag = 0;
12 newtio.c_cc[VTIME] = 20;
13 newtio.c_cc[VMIN] = 0;
14
15 tcflush(fd, TCIFLUSH);
16
17 tcsetattr(fd, TCSANOW, &newtio);
```

3-1-3. NetEye6410 직렬 통신 장치 이름

NetEye6410에 연결된 직렬 통신 장치의 이름 입니다.



/dev/ttySAC0	NetEye6410의 모니터링 포트
/dev/ttySAC1	Serial 1 에 연결된 장치
/dev/ttySAC2	Serial 2 에 연결된 장치
/dev/ttySAC3	Serial 3 에 연결된 장치

3-1-4. termios 구조체

시리얼 장치 파일은 터미널 장치로 분류됩니다. 터미널 인터페이스는 termios 형의 구조체에서 값을 설정하고 함수 호출을 사용하여 제어됩니다. termios는 POSIX에 의해 지정된 표준 인터페이스입니다. 구조체와 함수는 모두 헤더 파일 termios.h에 정의되어 있습니다.

```
1  #define NCCS 19
2  struct termios {
3          tcflag_t c_iflag; /* input mode flags */
4          tcflag_t c_oflag; /* output mode flags */
5          tcflag_t c_cflag; /* control mode flags */
6          tcflag_t c_lflag; /* local mode flags */
7          cc_t c_cc[NCCS]; /* control characters */
8  };
```

c_iflag	수신된 데이터를 내부 메모리에 가져오기 전 단계에서의 입력 처리를 설정합니다.
c_oflag	출력 처리 방법을 설정합니다.
c_cflag	직렬 통신을 위한 환경 변수를 설정합니다.
c_lflag	echo를 사용할 것인지 설정합니다.
c_cc 배열	EOF, STOP 등의 제어 동작들을 어떤 문자로 설정 할 것인가를 결정합니다.

3-1-5. 샘플 프로그램 작성

직렬 통신 장치를 연결한 뒤에, 데이터를 송·수신 하는 프로그램을 작성합니다.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <termios.h>
#include <math.h>

void Error_Msg(void);
int Char_to_Dec(char *str);
int CloseSerial(int fd);
int OpenSerial(char *device_name, int BAUDRATE);

int main(int argc, char **argv) {
        int fd=0;
        int cnt_;
        char *df;
        int Baudrate=0;
        char device_name[13]="/dev/ttySAC1";
        df=device_name;
        int c;
        int DBG_=0;
        int HEX_=0;
        int i=0;
        int BufSize=0;

        while((c=getopt(argc, argv, "dP:hb:s:"))!= -1) {
                switch(c) {
                        case 'd':
                                DBG_=1;
                                printf("-d₩n");
                                break;
                        case 's':
                                BufSize=Char_to_Dec(optarg);
                                printf("-s%d₩n", BufSize);
                                i|=4;
                                break;
                        case 'h':
                                HEX_=1;
                                printf("-h₩n");
                                break;
                        case 'P':
                                if(DBG_) printf("-P%s₩n", optarg);
                                if(optarg[0]=='1') df[11]='1';
                                else if(optarg[0]=='2')df[11]='2';
                                else if(optarg[0]=='3')df[11]='0';
                                else if(optarg[0]=='4')df[11]='1';
                                else if(optarg[0]=='5')df[11]='2';
                                else if(optarg[0]=='6')df[11]='3';
                                if(optarg[0]>='3') {
                                        df[8]='U';
                                        df[9]='S';
                                        df[10]='B';
                                }
                                if(optarg[0]<'1' || optarg[0]>'6')Error_Msg();
                                printf("-P%s₩n", df);
                                i|=1;
                                break;
```

```
59                      case 'b':
60                              Baudrate = Char_to_Dec(optarg);
61                              i|=2;
62                              printf("-b%d₩n", Baudrate);
63                              break;
64                      case '?':
65                              printf("argment error : %s₩n", optarg);
66                              Error_Msg();
67                              break;
68              }
69              if(DBG_)printf("getopt %c₩n", c);
70      }
71      if(i!=7)Error_Msg();
72      char Recv_data[BufSize];
73      fd=OpenSerial(df, Baudrate);
74      printf("Serial open₩n");
75      printf("device name: %s₩n", df);
76      printf("Baud rate %d₩n", Baudrate);
77      while(1) {
78              cnt_=0;
79              cnt_=read(fd, Recv_data, BufSize);
80              printf("₩nRecv: ");
81              printf("%s₩n", Recv_data);
82              if(HEX_) {
83                      for(i=0;i<cnt_;i++) {
84                              printf("%02x ", Recv_data[i]);
85                      }
86                      printf("₩nCnt:%d HCnt:%d₩n₩n", cnt_, i);
87              }
88              write(fd, Recv_data, strlen(Recv_data));
89              printf("Send: ");
90              printf("%s₩n", Recv_data);
91              if(HEX_) {
92                      for(i=0;i<strlen(Recv_data);i++) {
93                              printf("%02x ", Recv_data[i]);
94                      }
95                      printf("₩n");
96              }
97              memset(Recv_data, 0, cnt_);
98      }
99      CloseSerial(fd);
100     return 0;
101 }
102
103 void Error_Msg(void)
104 {
105     fprintf(stderr, "₩n₩nUsage                  : ./Serial [Option] ₩n");
106     fprintf(stderr, "₩n₩nOption");
107     fprintf(stderr, "₩n-p       : Serial port number (1 ~ 6)");
108     fprintf(stderr, "₩n         1 : Serial 1");
109     fprintf(stderr, "₩n         2 : Serial 2");
110     fprintf(stderr, "₩n         3 : USB 0");
111     fprintf(stderr, "₩n         4 : USB 1");
112     fprintf(stderr, "₩n         5 : USB 2");
113     fprintf(stderr, "₩n         6 : USB 3");
114     fprintf(stderr, "₩n₩n-b     : Baudrate");
115     fprintf(stderr, "₩n₩n-s     : Size");
116     fprintf(stderr, "₩n₩n-h     : Display Hex");
117     fprintf(stderr, "₩n₩n");
118     exit(1);
119 }
```

```
120  int Char_to_Dec(char *str){
121          int lenth=strlen(str);
122          int result=0;
123          int i=0;
124          for(i=0;i<lenth;i++){
125                  str[i]-='0';
126                  result+=str[i]*pow_(10, lenth-i-1);
127          }
128          return result;
129  }
130  int pow_(int val, int num){
131          int i;
132          int result=1;
133          if(num==0)return 1;
134          for(i=0;i<num;i++) {
135                  result*=val;
136          }
137          return result;
138  }
139  int CloseSerial(int fd){
140          close(fd);
141          return fd;
142  }
143
144  int OpenSerial(char *device_name, int BAUDRATE){
145          int fd;
146          struct termios newtio;
147          fd = open(device_name, O_RDWR | O_NOCTTY);
148          if(fd < 0){
149                  printf("Serial Port Open Fail.₩n");
150                  return -1;
151          }
152          memset(&newtio, 0, sizeof(newtio));
153          newtio.c_iflag = IGNPAR;
154          newtio.c_oflag = 0;
155          newtio.c_cflag = CS8|CLOCAL|CREAD;
156          switch(BAUDRATE) {
157                  case 460800 : newtio.c_cflag |= B460800;
158                          break;
159                  case 230400 : newtio.c_cflag |= B230400;
160                          break;
161                  case 115200 : newtio.c_cflag |= B115200;
162                          break;
163                  case 57600 : newtio.c_cflag |= B57600;
164                          break;
165                  case 38400 : newtio.c_cflag |= B38400;
166                          break;
167                  case 19200 : newtio.c_cflag |= B19200;
168                          break;
169                  case 9600: newtio.c_cflag |= B9600;
170                          break;
171                  default : newtio.c_cflag |= B57600;
172                          break;
173          }
174          newtio.c_lflag = 0;
175          newtio.c_cc[VTIME] = 20;
176          newtio.c_cc[VMIN] = 0;
177          tcflush(fd, TCIFLUSH);
178          tcsetattr(fd, TCSANOW, &newtio);
180          return fd;
181  }
```

- 19행: open()을 사용하기 위해 장치의 경로를 지정해줍니다.
- 27~70행 : 프로그램 실행시 입력 받은 매개변수를 처리합니다.
- 73행 : OpenSerial()를 이용하여 직렬 통신 장치를 연결합니다.
- OpenSerial( )로 분기되면, 직렬 통신에 필요한 환경 변수들을 설정합니다.
- 78~98행 : 직렬 통신 장치로부터 데이터를 반복해서 루프-백 합니다.

3-1-6. 샘플 프로그램 실행하기

샘플 프로그램으로 NetEye6410의 직렬 통신을 시험해 볼 수 있습니다.

프로그램을 실행 하기 전에 직렬 통신 장치가 연결되었는지 확인 하십시오.

```
1  #./Serial
2  Usage : ./Serial [Option]
3  Option
4  -P : Serial port number (1~6)
5  1 : Serial 1
6  2 : Serial 2
7  3 : USB 0
8  4 : USB 1
9  5 : USB 2
10 6 : USB 3
11
12 -b : Baudrate
13
14 -s : Size
15
16 -h : Display Hex
```

- 4~10행 : 사용하려는 직렬 포트를 지정하십시오.
- 12행 : 통신속도를 입력하십시오.
- 14행 : 수신된 데이터를 프로그램 스택에 가져오는 버퍼의 사이즈를 지정하십시오.
- 16행 : 콘솔에 표시되는 바이트를 Hex 값으로 설정합니다.

```
1  #./Serial –P1 –h –s10 -b57600
2
3  (....)
4
5  Recv: (수신 데이터 )
6
7  Cnt: 0 Hcnt:0
8
9  Send: (Echo)
```

샘플 프로그램은 콘솔 명령어를 이용해 실행합니다.

- [파일 명] [직렬 장치 이름] [데이터 표시 방식] [버퍼의 크기] [직렬 통신 속도]

3-2 NetEye6410의 소켓 사용하기

3-2-1 NetEye6410 TCP/IP 소켓 소개

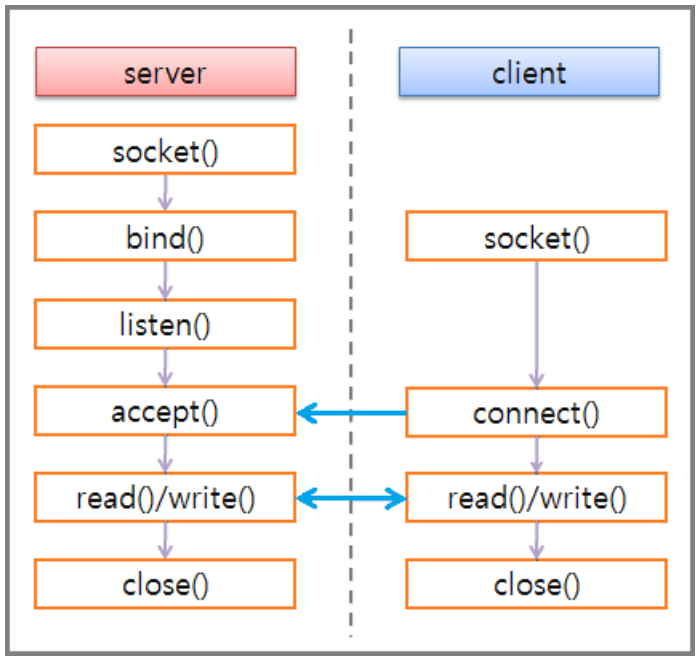
NetEye6410은 LAN 또는 Wi-Fi 같은 물리적으로 연결된 네트워크 상에서 "소켓"이라는 소프트웨어 장치를 통해 네트워크 통신을 사용 할 수 있습니다. 소켓 통신을 하기 위해서는 LAN과 Wi-Fi가 네트워크에 연결되어 있어야 합니다. 관련된 자료는 **2.NetEye6410 Getting Start** 항목을 참고 하십시오.

소켓 프로그램을 시작하기 전에 팁

- RJ-45 포트에 LED가 점멸 되고 있는지 확인 하십시오.
- 핑 (ping) 테스트를 통해 네트워크에 연결 되었는지 확인 하십시오.
- 안테나가 분리되어 있으면 무선 네트워크의 품질이 저하 됩니다.

3-2-2. 소켓 표준 C 라이브러리 함수

NetEye6410에서 동작하는 소켓 프로그램 함수는 표준 C 라이브러리 함수와 동일합니다.



1) socket( ) 함수

소켓을 생성하여 반환합니다.

Header	sys/types.h sys/socket.h
형태	int socket(int domain, int type, int protocol);
인수	domain : 소켓이 사용할 프로코톨 체계 (Protocol Family) 정보 전달합니다. type : 소켓의 데이터 전송방식에 대한 정보 전달합니다. protocol : 두 컴퓨터간 통신에 사용되는 프로토콜 정보 전달합니다.
반환	성공시 소켓 디스크립터의 양의 정수 값, 실패시 -1을 반환합니다.

```
1   #include <sys/types.h>
2   #include <sys/socket.h>
3
4   int server_socket;
5   server_socket = socket( PF_INET, SOCK_STREAM, 0);
6
7   if (-1 == server_socket)
8   {
9           printf( "server socket 생성 실패");
10          exit( 1) ;
11  }
```

2) bind ( ) 함수

IP 주소와 포트 번호를 지정합니다.

Header	sys/types.h
	sys/socket.h
형태	int bind(int sockfd, struct sockaddr *myaddr, socklen_t addrlen);
인수	sockfd : 포트를 지정할 소켓 디스크립터입니다.
	myaddr : 주소 정보가 담긴 구조체입니다.
	addrlen : myaddr 구조체의 크기입니다.
반환	성공시 0, 실패시 -1을 반환합니다.

```
1   #include <sys/types.h>
2   #include <sys/socket.h>
3
4   struct sockaddr_in server_addr;
5
6   -socket()...-
7
8   memset( &server_addr, 0, sizeof( server_addr));
9   server_addr.sin_family = AF_INET;
10  server_addr.sin_port = htons( 4000);
11  server_addr.sin_addr.s_addr= htonl( INADDR_ANY);
12
13  if( -1 == bind( server_sock, (struct sockaddr*)&server_addr, sizeof( server_addr)))
14  {
15          printf( "bind() 실행 에러₩n"); exit( 1);
16  }
```

3) listen ( ) 함수

클라이언트의 접속 요청을 기다리도록 설정합니다.

Header	sys/socket.h
형태	int listen(int sockfd, int backlog);
인수	sockfd : 포트를 지정할 소켓 디스크립터입니다.
	backlog : 대기 메시지 큐의 개수입니다.
반환	성공시 0, 실패시 -1을 반환합니다

```
#include <sys/types.h>
#include <sys/socket.h>

-socket()...-

-bind()...-

if( -1 == listen(server_socket, 5))
{
        printf( "listen() 실행 실패₩n");
        exit( 1);
}
```

4) accept ( ) 함수

클라이언트의 접속 요청을 받아들이고 클라이언트와 통신하는 전용 소켓을 생성합니다.

Header	sys/types.h
	sys/socket.h
형태	int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
인수	sockfd : 포트를 지정할 소켓 디스크립터입니다.
	addr : 클라이언트 주소 정보를 가지고 있는 구조체입니다.
	addrlen : addr 구조체의 크기입니다.
반환	성공시 새로운 소켓 디스크립터, 실패시 -1을 반환합니다.

```
#include <sys/types.h>
#include <sys/socket.h>

int client_sock;
int client_addr_size;

struct sockaddr_in client_addr;

-socket()...-

-bind()...-

-listten()...-

client_addr_size = sizeof( client_addr);
client_sock=accept(server_sock,(struct sockaddr*)&client_addr, &client_addr_size);

if ( -1 == client_sock)
{
        printf( "클라이언트 연결 수락 실패₩n");
        exit( 1);
}
```

5) connect ( ) 함수

생성한 소켓을 통해 서버로 접속을 요청합니다.

Header	sys/types.h sys/socket.h
형태	int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);
인수	sockfd : 포트를 지정할 소켓 디스크립터입니다. serv_addr : 서버 주소 정보를 가지고 있는 구조체입니다. addrlen : serv_addr 구조체의 크기입니다.
반환	성공시 새로운 소켓의 디스크립터, 실패시 -1을 반환합니다.

```
1  #include <sys/types.h>
2  #include <sys/socket.h>
3
4  int client_socket;
5  int client_addr_size;
6
7  struct sockaddr_in server_addr;
8
9  -socket()...-
10
11 memset( &server_addr, 0, sizeof( server_addr));
12 server_addr.sin_family = AF_INET;
13 server_addr.sin_port = htons( 4000);
14 server_addr.sin_addr.s_addr= inet_addr( "127.0.0.1");
15
16 if( -1==connect(client_socket,(struct sockaddr*)&server_addr,sizeof( server_addr)))
17 {
18         printf( "접속 실패₩n");
19         exit( 1);
20 }
```

6) read ( ) 함수

해당 소켓으로 데이터를 수신합니다.

Header	unistd.h
형태	ssize_t read (int sockfd, void *buffer, size_t nbytes)
인수	sockfd : 데이터를 수신할 소켓 디스크립터입니다. buffer : 데이터를 수신하여 저장할 버퍼를 설정합니다. nbytes : 버퍼의 크기입니다.
반환	성공시 읽어들인 바이트 수를, 실패시 -1을 반환합니다.

```
1  #include <unistd.h>
2
3  #define BUFF_SIZE 1024
4
5  char buff[BUFF_SIZE];
6  int fd;
7
8  read( fd, buff, BUFF_SIZE);
```

7) write ( ) 함수

해당 소켓으로 데이터를 송신합니다.

Header	unistd.h
형태	ssize_t write (int sockfd, const void *buf, size_t n)
인수	sockfd : 데이터를 수신할 소켓 디스크립터입니다. buffer : 송신할 데이터가 담긴 버퍼입니다. nbytes : buffer의 크기입니다.
반환	성공시 쓰기를 한 바이트 수를, 실패시 -1을 반환합니다.

```
1  #include <unistd.h>
2
3  char *temp = "smic21.com";
4  int fd;
5  write( fd, temp, strlen( temp));
```

8) close ( ) 함수

해당 소켓을 종료합니다.

Header	unistd.h
형태	int close(int sockfd)
인수	sockfd : 연결을 종료할 소켓 디스크립터입니다.
반환	성공시 0을, 실패시 -1을 반환합니다.

```
1  #include <unistd.h>
2
3  int fd;
4  close( fd);
```

3-2-4. 샘플 프로그램

다음 예제는 클라이언트 실행 시 메시지를 송신 후 다시 수신하는 에코 서버&클라이언트 예제입니다.

Server.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>

int sock_init(char* tcp_PORT, int* sock_server, int* sock_client);

int main(int argc, char** argv)
{
        int sock_server, sock_client;
        int i=0;
        char buff[1024] = {0x00};
        int ret;
        if(argc != 2) {
                printf("usage: %s port₩n", argv[0]);
                return 0;
        }
        sock_init(argv[1], &sock_server, &sock_client);
        while(1)
        {
                ret = read(sock_client, buff, 1024);
                write(sock_client, buff, 1024);
                if(ret > 0) {
                        printf("recv[%u, %d byte(s)] %s₩n", i--, ret, buff);
                        memset(buff, 0, ret);
                }
        }
        close(sock_client);
        close(sock_server);
        return 0;
}

int  sock_init(char* tcp_PORT, int* sock_server, int* sock_client){
        struct sockaddr_in addr_server, addr_client;
        int size_addr_client;

        (*sock_server) = socket(PF_INET, SOCK_STREAM, 0);
        if((*sock_server) == -1) {
                perror("socket()₩n");
                return 0;
        }
        memset(&addr_server, 0, sizeof(addr_server));
        addr_server.sin_family=AF_INET;
        addr_server.sin_addr.s_addr=htonl(INADDR_ANY);
        addr_server.sin_port = htons(atoi(tcp_PORT));

        if(bind((*sock_server),(struct sockaddr*)&addr_server, sizeof(addr_server)) == -1) {
                perror("bind()");
                close((*sock_server));
                return 0;
        }
```

```
55          if(listen((*sock_server), 1) == -1) {
56                  perror("listen()");
57                  close((*sock_server));
58                  return 0;
59          }
60          size_addr_client = sizeof(addr_client);
61          (*sock_client)=accept((*sock_server),(struct sockaddr*)&addr_client,&size_addr_client);
62          if((*sock_client) == -1) {
63                  perror("accept()");
64                  close((*sock_server));
65                  return 0;
66          }
67          usleep(100*1000);
68          return 0;
69  }
```

- 21행 : sock_init()를 사용하여 socket(),bind(),listen(), accept()를 순서대로 실행합니다.
- 22~30행 : 클라이언트가 송신한 데이터를 반복해서 루프-백 합니다.
- 40행 : socket()을 사용하여 소켓을 생성합니다.
- 50행 : 소켓의 IP주소와 포트 번호를 부여합니다.
- 55행 : 클라이언트 접속 대기열을 생성합니다.
- 61행 : 클라이언트의 접속을 허용 합니다.

Client.c

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <arpa/inet.h>
6  #include <sys/types.h>
7  #include <sys/socket.h>
8
9  int main(int argc, char** argv)
10 {
11         if(argc != 4) {
12                 printf("usage: %s server_ip server_port message₩n", argv[0]);
13                 return 0;
14         }
15         int sock;
16         struct sockaddr_in addr_sock;
17         char sock_buff[255];
18         int sock_read_size=0;
19
20         sock = socket(PF_INET, SOCK_STREAM, 0);
21         if(sock == -1) {
22                 perror("sock()₩n");
23                 return 0;
24         }
25         memset(&addr_sock, 0, sizeof(addr_sock));
26         addr_sock.sin_family=AF_INET;
27         addr_sock.sin_addr.s_addr=inet_addr(argv[1]);
28         addr_sock.sin_port = htons(atoi(argv[2]));
29         if(connect(sock, (struct sockaddr*)&addr_sock, sizeof(addr_sock)) == -1) {
30                 perror("connect()₩n");
31                 return 0;
32         }
33         write(sock, argv[3], strlen(argv[3]));
34         memset(sock_buff, 0, 255*sizeof(char));
35         sock_read_size = read(sock, sock_buff, 255);
36         if(sock_read_size>0){
37                 printf("%s₩n", sock_buff);
38         }
39         close(sock);
40         return 0;
41 }
```

- 20 : 소켓을 생성합니다.
- 29 : 서버로 접속을 요청합니다.
- 33 : 서버로 데이터를 송신합니다.
- 50 : 서버로부터 데이터를 수신합니다.

3-2-5. Example Test

NetEye6410 구매 시 제공되는 샘플 프로그램을 이용해 소켓 서버와 소켓 클라이언트에 대한 동작들을 직접 확인해 보십시오.

```
1  #./server 5566
2  recv[0, 8byte(s)] 01020304
```

```
1  #./client 192.168.100.173 5566 01020304
2  01020304
```

샘플 프로그램은 콘솔 명령어를 이용해 실행합니다.

- 서버 프로그램 : [파일 명] [포트 번호]
- 클라이언트 프로그램 : [파일 명] [서버의 IP주소] [서버의 포트 번호] [보낼 메시지]

3-3. NetEye6410의 Digital I/O 사용하기

3-3-1. DI/DO 세부 사항

- DI 동작 전압(허용전류) : 5VDC(50mA)
- DO 출력 전압 : infinity (Max. 80VDC)
- DO 출력 전류 : Max. 50mA
- Isolated type

3-3-2. Di 사용 시 부하의 출력 방식을 확인 하십시오

NetEye6410은 주문 사양에 따라 DI 의 동작 방식이 결정됩니다. 사용하기 전에, 연결하는 부하의 출력 방식이 Dry Contact 인지 Wet Contact 인지 확인하십시오. 주문 사양을 선택하지 않은 경우 Wet Contact로 동작 합니다.

부하 선정 팁

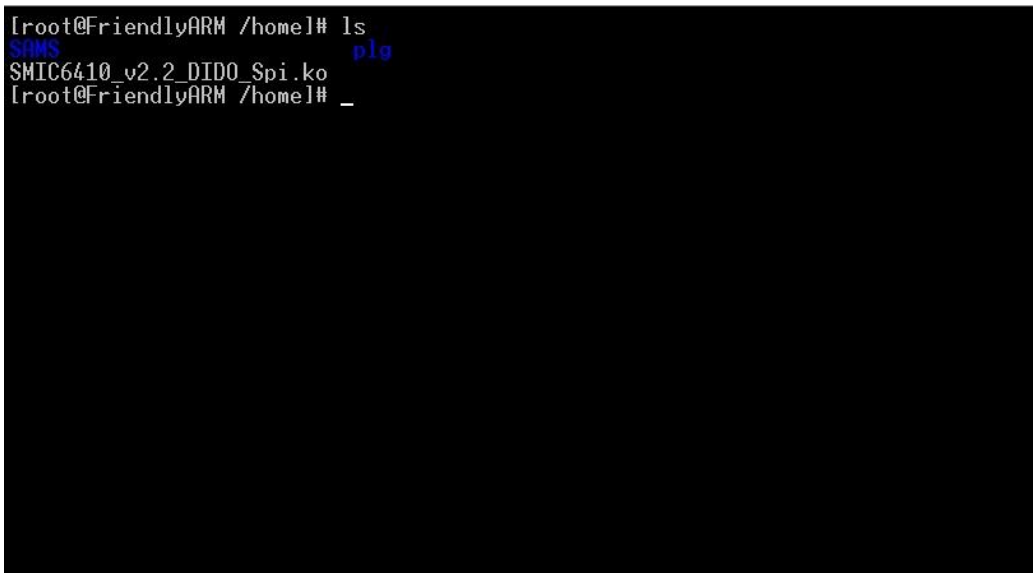
- Dry Contact 출력은 부하 출력단에서 전압이 실리지 않습니다.
- Dry Contact 출력의 대부분 릴레이 출력 입니다.
- Wet Contact 출력은 부하에 DC 또는 AC 전압이 실려 있는 출력 입니다.
- AC출력의 부하는 NetEye6410과 연결하는 것을 허용하지 않습니다.

3-3-3. Do 사용 시 부하의 구동 전류를 확인 하십시오

50mA 이상의 전류가 흐르는 부하를 연결하지 마십시오. 부하에 흐르는 전류가 클 경우, 릴레이를 이용하여 외부 전원을 사용해 제어 하십시오.

3-3-4. 디바이스 드라이버의 사용

C언어를 이용해 프로그램을 작성하여 DI/DO 기능을 사용 할 수 있습니다. DI/DO를 사용하기 위해서는 디바이스 드라이버 모듈 파일이 있어야 합니다. 만약 NetEye6410 내부 저장소에 이 모듈이 존재하지 않을 경우, 여기를 통해 원하는 자료를 획득 하십시오.



3-3-4. 디바이스 드라이버의 사용

디바이스 드라이버가 커널에 적재되어 있는지 확인 합니다.

```
[root@FriendlyARM /home]# lsmod
SMIC6410_v2.2_DIDO_Spi 1428 0 - Live 0xbf000000
[root@FriendlyARM /home]#
```

lsmod 명령어를 사용해 커널에 적재된 모듈 파일을 확인 할 수 있습니다. 커널에 해당 디바이스 드라이버 모듈이 존재하지 않을 경우 insmod 명령어를 사용하여 새로 커널에 적재 합니다.

```
[root@FriendlyARM /home]# lsmod
[root@FriendlyARM /home]# insmod SMIC6410_v2.2_DIDO_Spi.ko
[root@FriendlyARM /home]# lsmod
SMIC6410_v2.2_DIDO_Spi 1428 0 - Live 0xbf006000
[root@FriendlyARM /home]# _
```

자세한 리눅스 명령어는 **4-1.임베디드 리눅스 명령어** 를 참고하십시오.

3-3-5. DI/DO 데이터 획득

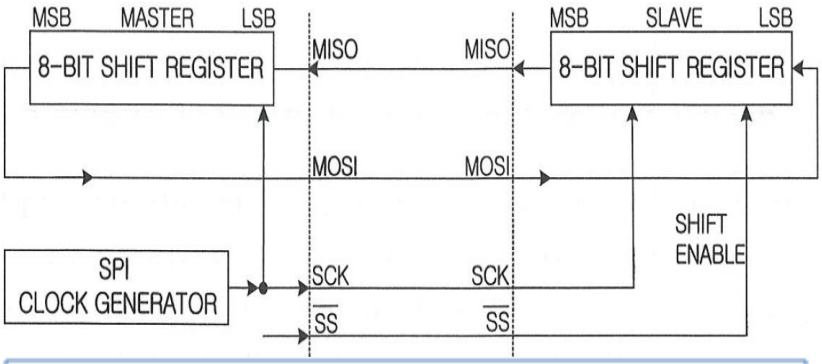
DI/DO는 SPI 통신을 이용해 부하에서 발생한 데이터를 메모리로 가져올 수 있습니다.

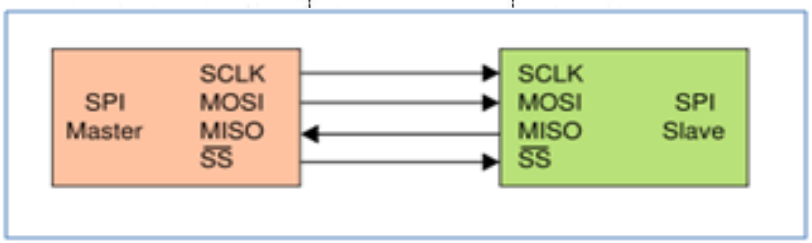
SPI 통신의 특징

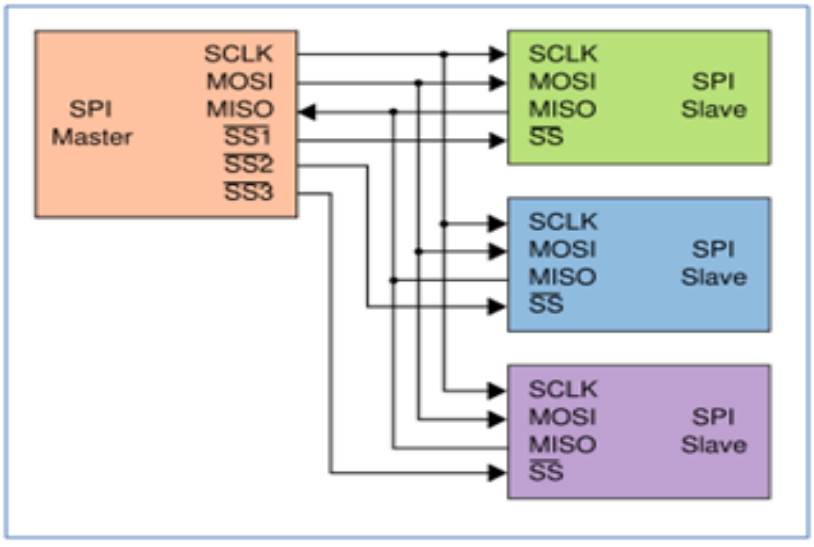
- Motorola에서 MCU와 주변 장치간의 Serial 통신을 위해 고안
- 전이중(Full-duplex), 4선 동기식
- 하나의 Master 다수 Slave의 선택적 1:1 통신

사용되는 신호

- SS (Slave Select) : Slave 선택 신호, active low
- SCLK (Serial Clock) : Clock 신호 (Master)
- MISO (Master Input, Slave Output) : Master 입력, Slave 출력
- MOSI (Master Output, Slave Input) : Master 출력, Slave 입력

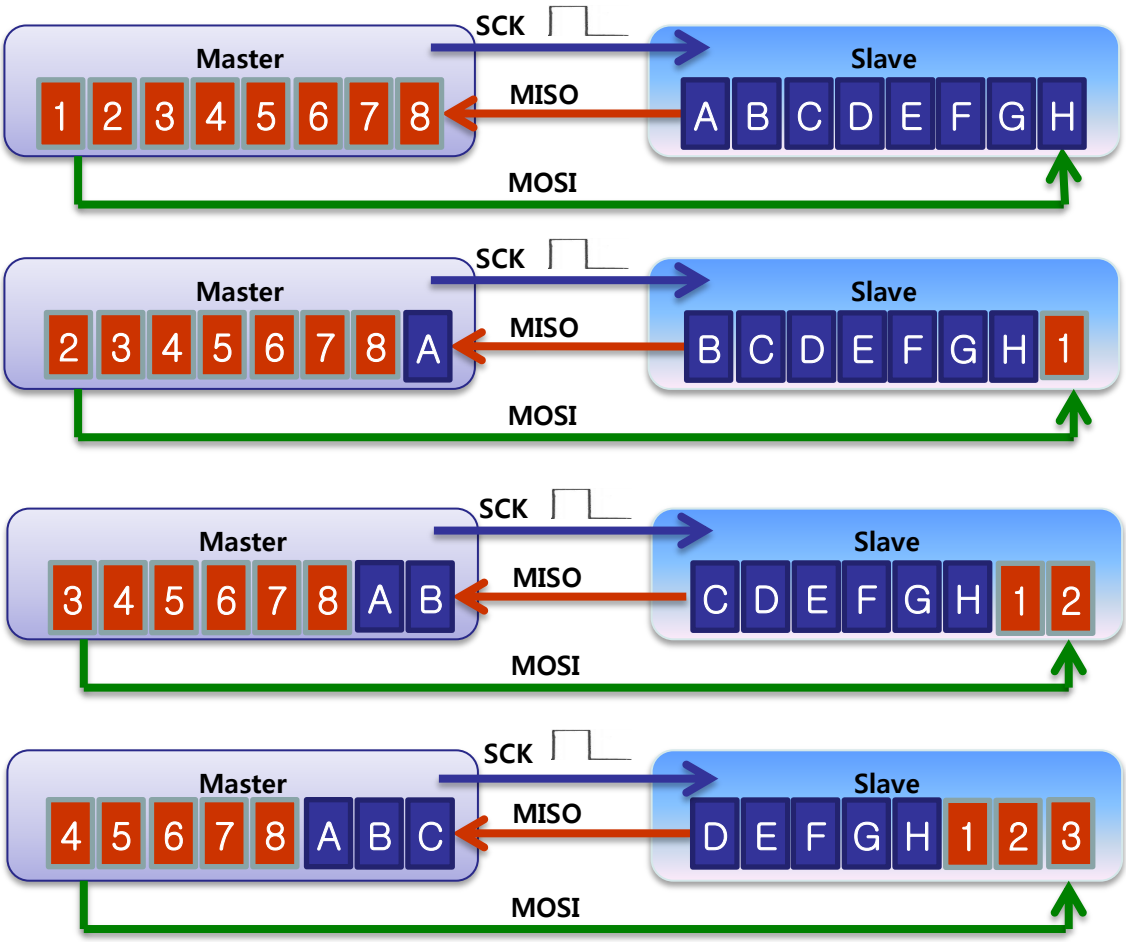






SPI 통신 동작 구조

- Master에서 통신할 Slave의 SS Pin을 Low로 하여 Slave Select
- Master에서 발생시킨 Clock에 의해 1Clock 당 1Bit 씩 Shift
- Master와 Slave가 동시에 data 교환
- 통신 완료 후 SS Pin을 High로 하여 Slave Select 해제
- Data 레지스터에 값을 넣지 않으면 쓰레기 값 전송
- MSB, LSB 우선 설정 가능



MSB 우선 전송 예

3-3-6. 프로그램 작성하기

NetEye6410에서 DI/DO 기능을 사용하는 프로그램을 작성합니다. 크로스 개발 환경이 구축된 상태에서 진행 하십시오. 크로스 개발 환경과 관련된 내용은 **4-4. 개발 환경 구축**을 참고 하십시오.

통신 프로토콜

NetEye6410에서 SPI통신을 하기 위한 통신 프레임 구조는 총 6 바이트로 구성됩니다.

- Function Code : DI/DO를 읽을지 쓸지 결정 합니다. 읽기 (0x03), 쓰기 (0x06)
- Register Address : 데이터가 담겨있는 레지스터 주소 입니다. Di (0x64), Do (0x65)
- DATA : DI 또는 DO 데이터 필드 입니다.
- NULL : 사용하지 않는 필드 입니다.

Field Num	Field Name
1	Funtion Code
2	Register Address
3	NULL
4	NULL
5	NULL
6	NULL

Master

Field Num	Field Name
1	NULL
2	Function Code
3	Register Address
4	NULL
5	Reserved
6	DATA

Slave

읽기 동작

Field Num	Field Name
1	Funtion Code
2	Register Address
3	Reserved
4	DATA
5	NULL
6	NULL

Master

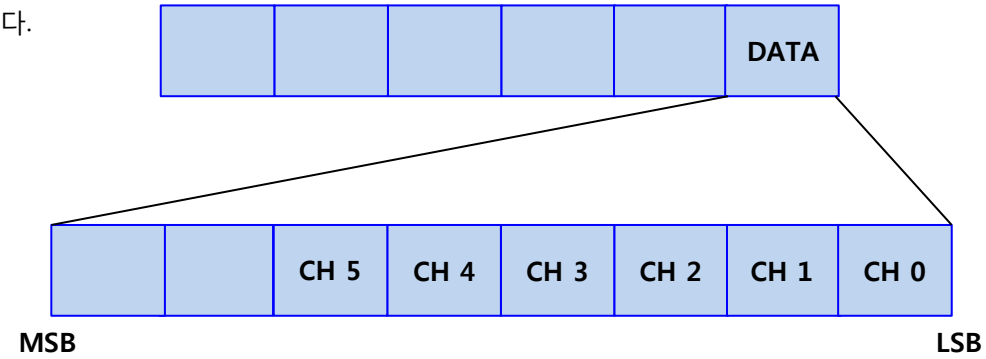
Field Num	Field Name
1	NULL
2	Function Code
3	Register Address
4	NULL
5	Reserved
6	DATA

Slave

쓰기 동작

DI/DO 의 데이터

NetEye6410의 DI 또는 DO 데이터는 1바이트 입니다. DI와 DO의 각 채널은 1비트로 표현되며, DI/DO의 각 채널 별로 DATA 필드의 LSB (Least Significant Bit) 부터 순서대로 매핑 (Mapping) 됩니다.



DI/DO 상태 읽기

NetEye6410의 DI/DO 상태를 읽을 수 있는 프로그램을 작성합니다.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <fcntl.h>
5
6  #define DEVICE    "/dev/SMIC6410_v2.2_DIDO_Spi"
7
8  typedef struct
9  {
10    unsigned char ch0 : 1;
11    unsigned char ch1 : 1;
12    unsigned char ch2 : 1;
13    unsigned char ch3 : 1;
14    unsigned char ch4 : 1;
15    unsigned  char ch5 : 1;
16    unsigned char res0 : 1;
17    unsigned char res1 : 1;
18 } DIO_STATUS;
19
20 int main(void)
21 {
22    int fd;
23    unsigned char buf[6] = {0};
24    DIO_STATUS dio;
25
26    fd = open(DEVICE, O_RDWR | O_NOCTTY);
27
28    buf[0] = 0x03;
29    buf[1] = 0x64;    // Di : 0x64, Do : 0x65
30    read(fd, buf, 1);
31
32    memcpy(&di, &buf[5], sizeof(DIO_STATUS));
33
34    printf("ch0 : %d₩n", dio.ch0);
35    printf(" ch1 : %d₩n", dio.ch1);
36    printf(" ch2 : %d₩n", dio.ch2);
37    printf(" ch3 : %d₩n", dio.ch3);
38    printf(" ch4 : %d₩n", dio.ch4);
39    printf(" ch5 : %d₩n", dio.ch5);
40
41    return 0;
42 }
```

- 6행 : 디바이스 드라이버 이름
- 26행 : 디바이스 드라이버를 open
- 28~30행 : 프로토콜에 맞게 읽기 (또는 쓰기) 동작 수행
- 32행 : DI 데이터를 메모리에 복사

DO 쓰기

NetEye6410을 이용하여 부하에 High (또는 Low) 신호를 보내려는 경우, DO 쓰기 동작을 하는 프로그램을 작성합니다.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <fcntl.h>
5
6  #define DEVICE    "/dev/SMIC6410_v2.2_DIDO_Spi"
7
8  typedef struct
9  {
10    unsigned char ch0 : 1;
11    unsigned char ch1 : 1;
12    unsigned char ch2 : 1;
13    unsigned char ch3 : 1;
14    unsigned char ch4 : 1;
15    unsigned  char ch5 : 1;
16    unsigned char res0 : 1;
17    unsigned char res1 : 1;
18 } DIO_STATUS;
19
20 int main(void)
21 {
22    int fd;
23    unsigned char buf[6] = {0};
24    DIO_STATUS dio;
25
26    fd = open(DEVICE, O_RDWR | O_NOCTTY);
27
28    buf[0] = 0x06;
29    buf[1] = 0x65;
30    buf[3] = 0x01;    // CH0 LOW , others HIGH
31    read(fd, buf, 1);
32
33    if(buf[5] == 0x01)
34    {
35       memcpy(&di, &buf[5], sizeof(DIO_STATUS));
36
37       printf("ch0 : %d₩n", dio.ch0);
38       printf(" ch0 : %d₩n", dio.ch0);
39       printf(" ch0 : %d₩n", dio.ch0);
40       printf(" ch0 : %d₩n", dio.ch0);
41       printf(" ch0 : %d₩n", dio.ch0);
42       printf(" ch0 : %d₩n", dio.ch0);
43    }
44    return 0;
45 }
```

- 28행 : 쓰기 동작 수행
- 29행 : DO 레지스터의 주소 (0x65)
- 30행 : Ch0은 LOW, 나머지 채널은 HIGH
- 33행 : 데이터가 정확히 파싱 (parsing) 되었는지 검사

3-5. NetEye6410의 Analog Input 사용하기

3-4-1. AI 의 세부 사항

- 전압 범위 : 0-10VDC
- 전류 범위 : 0-40mA
- 분해능 (Resolution) : 10-bit
- Isolated type

3-4-2. 디바이스 드라이버의 사용

NetEye6410에서 AI 기능을 사용하기 위해서는 Digital I/O 에서 사용하는 드라이버와 똑같은 모듈이 커널에 적재되어 있어야 합니다. 디바이스 드라이버에 대한 자세한 설명은 **3-3.Digital I/O 사용하기** 를 참고하십시오.

3-4-3. Ai 데이터 획득

NetEye6410에서 AI 기능은 DI/DO와 같은 방식으로 데이터를 메모리로 가져 옵니다. 통신에 대한 자세한 설명은 **3-3-5. Di/Do 데이터 획득** 을 참고하십시오.

3-4-4. 프로그램 작성하기

NetEye6410에서 AI 기능을 사용하는 프로그램을 작성합니다. 크로스 개발 환경이 구축된 상태에서 진행 하십시오. 크로스 개발 환경과 관련된 내용은 **4-4. 개발 환경 구축**을 참고 하십시오.

통신 프로토콜

NetEye6410에서 SPI통신을 하기 위한 통신 프레임 구조는 총 6 바이트로 구성됩니다.

- Function Code : AI 를 읽을 수 있도록 0x03을 사용합니다.
- Register Address : 데이터가 담겨있는 레지스터 주소 입니다.
  - CH0 주소 : 0x96
  - CH1 주소 : 0x97
  - CH2 주소 : 0x98
  - CH3 주소 : 0x99
- DATA : 2 bytes의 데이터 필드 입니다.
- NULL : 사용하지 않는 필드 입니다.

Field Num	Field Name
1	Funtion Code
2	Register Address
3	NULL
4	NULL
5	NULL
6	NULL

Master

Field Num	Field Name
1	NULL
2	Function Code
3	Register Address
4	NULL
5	DATA
6	DATA

Slave

AI 데이터 수집 프로그램

NetEye6410을 이용하여 AI 데이터를 수집하는 프로그램을 작성합니다.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <fcntl.h>
5
6  #define DEVICE    "/dev/SMIC6410_v2.2_DIDO_Spi"
7
8  typedef struct
9  {
10    unsigned short ch0;
11    unsigned short ch1;
12    unsigned short ch2;
13    unsigned short ch3;
14 } AI_DATA_PARSE
15
16 int main(void)
17 {
18    int fd;
19    unsigned char buf[6] = {0};
20    AI_DATA_PARSE ai;
21
22    fd = open(DEVICE, O_RDWR | O_NOCTTY);
23
24    buf[0] = 0x03;
25    buf[1] = 0x96;    // Ai CH0
26    read(fd, buf, 1);
27
28    ai.ch0 = (unsigned short)((buf[4] << 8) | buf[5]);
29
30    printf("%d₩n", ai.ch0);
31
32    return 0;
33 }
```

- 25행 : AI Ch0 레지스터 주소 입니다.
- 28행 : AI 데이터는 2바이트로 표현됩니다.

Ai 표현 방식

NetEye6410을 이용하여 AI 데이터를 수집하는 경우, 연결되는 부하에 따라 전압,전류 또는 Raw Data를 선택적으로 표현 할 수 있습니다. SPI 통신을 이용하여 프로토콜에 맞춰 쓰기동작을 수행 합니다.

Write Value	Remark
0	Raw Data
1	0-5V
2	0-10V
3	0-20mA
4	0-40mA

AI 데이터 표현 변경 프로그램

0-20mA의 데이터를 보내는 부하를 AI Ch0에 연결한경우, 158번지에 3번 값으로 쓰기 동작을 수행하고, 150번지에서 데이터를 가져 오는 프로그램을 작성합니다.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <fcntl.h>
5
6  #define DEVICE    "/dev/SMIC6410_v2.2_DIDO_Spi"
7
8  typedef struct
9  {
10    unsigned short ch0;
11    unsigned short ch1;
12    unsigned short ch2;
13    unsigned short ch3;
14 } AI_DATA_PARSE
15
16 int main(void)
17 {
18    int fd;
19    unsigned char buf[6] = {0};
20    AI_DATA_PARSE ai;
21
22    fd = open(DEVICE, O_RDWR | O_NOCTTY);
23
24    buf[0] = 0x06;
25    buf[1] = 0x9E;
26    buf[3] = 0x03;    // 0-20mA
27    read(fd, buf, 1);
28
29    memset(buf, 0, 6);
30
31    buf[0] = 0x03;
32    buf[1] = 0x96;
33    read(fd, buf, 1);
34
35    ai.ch0 = (unsigned short)((buf[4] << 8) | buf[5]);
36
37    printf("%d₩n", ai.ch0);
38
39    return 0;
40 }
```

- 24행 : 쓰기 동작을 수행 합니다.
- 25행 : AI Ch0의 상태를 바꾸는 주소 입니다.
- 26행 : AI 표현 방식의 표를 참고 하십시오.

3-5. NetEye6410의 DI/DO/AI 레지스터 맵

3-5-1. 샘플 프로그램 사용

샘플 프로그램은 **여기**를 통해 다운로드 받으실 수 있습니다. 샘플 프로그램을 이용하여 NetEye6410의 DI/DO/AI기능을 모두 사용할 수 있습니다.

샘플 프로그램의 읽기 동작

NetEye6410의 콘솔 화면에서 샘플 프로그램을 실행한 뒤에 r을 입력하면 읽기 동작이 수행됩니다. 사용자는 DI/DO/AI 전체 데이터를 모두 확인 할 수 있습니다.



샘플 프로그램의 쓰기 동작

쓰기 동작을 수행하기 위해서는 w를 입력합니다. 다음 단계로 넘어가 Address 에 해당 주소 값을 입력한 뒤에 엔터를 입력하면 다음 단계로 넘어갑니다. 입력하고자 하는 값을 입력하고 엔터를 입력합니다.



3-5-2. NetEye6410 DI/DO/AI Register Map

Address	Data Size	Bit Seq.	Value Range		Description	Property
			Min	Max		
100	1	0			Reserved	
		1			Reserved	
		2	0	1	DI Ch5	R
		3	0	1	DI Ch4	R
		4	0	1	DI Ch3	R
		5	0	1	DI Ch2	R
		6	0	1	DI Ch1	R
		7	0	1	DI Ch0	R
101	1	0			Reserved	
		1			Reserved	
		2	0	1	DO Ch5	R/W
		3	0	1	DO Ch4	R/W
		4	0	1	DO Ch3	R/W
		5	0	1	DO Ch2	R/W
		6	0	1	DO Ch1	R/W
		7	0	1	DO Ch0	R/W
102	2		0	4294967295	DI Ch0 Counter	R/W
104	2		0	4294967295	DI Ch1 Counter	R/W
106	2		0	4294967295	DI Ch2 Counter	R/W
108	2		0	4294967295	DI Ch3 Counter	R/W
110	2		0	4294967295	DI Ch4 Counter	R/W
112	2		0	4294967295	DI Ch5 Counter	R/W
114-120	reserved					
121	1		0	1	DI status selection	R/W
122	1		0	1	DI status display	R/W
123-149	reserved					
150	1		0	65535	AI Ch0	R
151	1		0	65535	AI Ch1	R
152	1		0	65535	AI Ch2	R
153	1		0	65535	AI Ch3	R
154-157	reserved					
158	1		0	4	AI Ch0 Status	R/W
159	1		0	4	AI Ch1 Status	R/W
160	1		0	4	AI Ch2 Status	R/W
161	1		0	4	AI Ch3 Status	R/W

- (ex) DI Ch0 Counter, value of counter = (102) * 65536 + (103)
- DI status selection : 0 = falling edge, 1 = rising edge, default is falling edge
- DI status display : 0 = ON/OFF, 1 = OFF/ON, default is 0 = ON/OFF

3-6. NetEye6410의 Zigbee 사용하기 (옵션)

NetEye6410은 주문 사양에 따라 Zigbee 통신 기능을 사용 할 수 있습니다. Zigbee를 사용해야 하는 경우 제품 주문 시 반드시 Zigbee 옵션을 선택 하십시오.

3-6-1. Zigbee 통신

- Low Rate WPANs
- Low-power consumption
- 250 Kbps maximum speed
- Unlicensed frequency bands
- 2.4 GHz, 915 MHz, and 868 MHz
- 802.15.4 standard

ZigBee® Alliance
Wireless Control That Simply Works

3-6-2. Zigbee Coordinator

Zigbee 통신은 NetEye6410 내부에 별도로 장착된 Zigbee Coordinator 가 수행 합니다. Zigbee 개발 환경은 제품 구매 시 Zigbee 사양을 선택하시면 받을 수 있습니다.

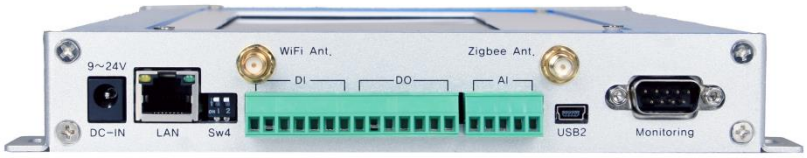
TinyOS는 UC Berkeley에서 개발된 센서 네트워크를 위한 무료 운영체제 입니다. TinyOS는 다양한 하드웨어, MAC 프로토콜, 네트워크 프로톨, 센서 인터페이스를 소스레벨로 완전 공개하고 있습니다. TinyOS 플랫폼에서 프로그램을 작성할 때는 NesC 언어를 사용 합니다.

Zigbee Coordinator 사양

Zigbee Coordinator는 MSP430F1611을 CPU로 사용하며, MSP430F1611은 TI(Texas Instrument)사의 초저전력 16비트 RISC 마이크로 건트롤러 입니다. 그리고 Zigbee 통신을 위한 RF 트랜시버로 TI사의 CC2420칩을 사용 합니다. CC2420 chip은 2.4 GHz 대역의 IEEE 802.15.4 표준을 지원 합니다.

Zigbee Coordinator를 사용하기

Zigbee Coordinator에 펌웨어를 다운로드 하기 위해서는 Mini-USB (B-type) 케이블을 연결 합니다.



ZIGBEE

TinyOS와 NesC에 대한 기술 지원을 받으십시오.

㈜신명정보통신 기술연구소 에서는 귀하의 문제를 적극적으로 해결해 드립니다.

3-6-3. Zigbee Coordinator 과의 통신

NetEye6410에서 MCU와 Zigbee Coordinator는 내부 USB로 연결되어 있습니다. 시리얼 통신 프로그램을 작성하여 Zigbee를 통해 획득한 데이터를 내부 메모리에 파싱 합니다. Zigbee 통신을 위한 프로그램 작성 시, 디바이스 이름은 /dev/ttyUSB0 을 사용하십시오.

- Baud rate : 57600
- Data Bit : 8
- Parity : None
- Stop Bit : 1
- Flow Control : None

프로그램 작성

Zigbee 통신을 통해 들어온 메시지를 내부 메모리에 가져오는 프로그램을 작성 합니다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <termios.h>
#include  <sys/types.h>

#define DEVICE   "/dev/ttyUSB0"
#define BAUDRATE   B57600

int main(void)
{
   int fd;
   struct termios newtio;
   unsigned char RcvBuf[10] = {0};
   int i, RcvSize = 0;

   fd = open(DEVICE, O_RDWR | O_NOCTTY);

   memset(&newtio, 0, sizeof(newtio));
   newtio.c_iflag = IGNPAR;
   newtio.c_oflag = 0;
   newtio.c_cflag = CS8 | CLOCAL | CREAD;

   newtio.c_cflag  |= BAUDRATE;
   newtio.c_lflag = 0;
   newtio.c_cc[VTIME] = 1;
   newtio.c_cc[VMIN] = 1;

   tcflush(fd, TCIFLUSH);
   tcsetattr(fd, TCSANOW, &newtio);

   while(1)
   {
      RcvSize = read(fd, RcvBuf, 4);
      if(RcvSize == 4)
      {
         for(i=0; i<RcvSize; i++)
         {
            printf("%.2x ", RcvBuf[i]);
         }
         putchar('₩n');
      }
```

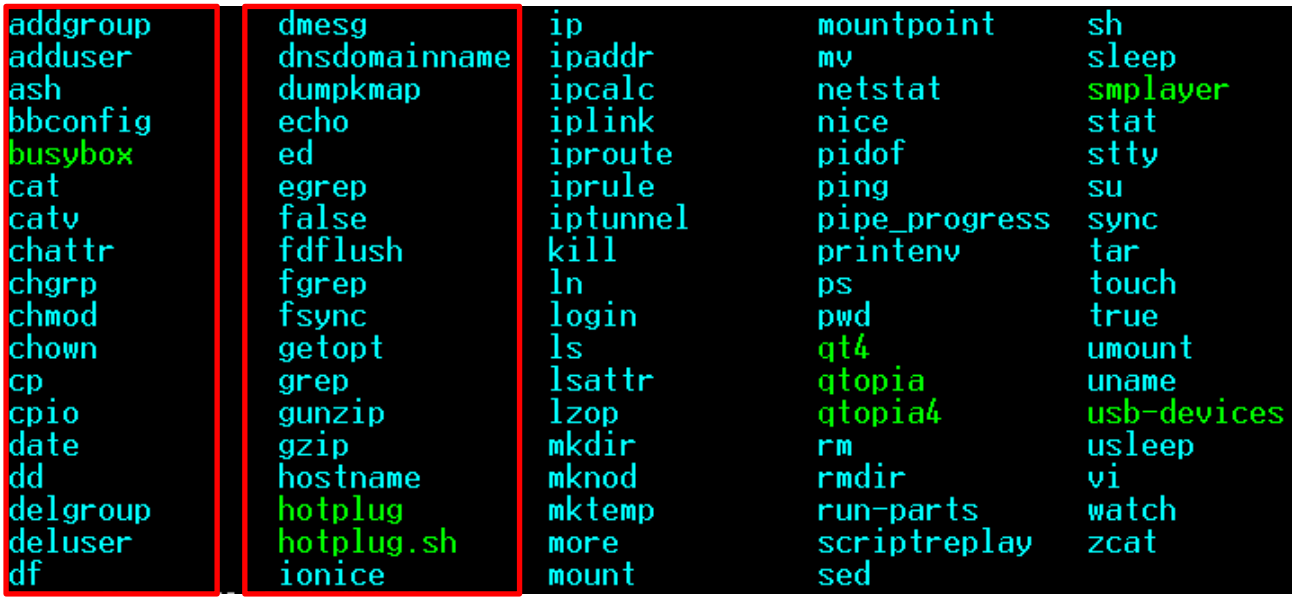
```
44      }
45      return 0;
46  }
```

- 8행 : Zigbee Coordinator에 대한 드라이버 이름 입니다.
- 9행 : Zigbee Coordinator의 시리얼 통신 속도 입니다.
- 18행 : 파일 디스크립터를 오픈 합니다.
- 20~26행 : 시리얼 통신을 위한 환경 변수를 입력 합니다.
- 27행 : Blocking Timeout 을 100ms로 설정 합니다.
- 28행 : 최소 입력 문자를 1개로 설정 합니다.
- 33~44행 : Zigbee 통신을 통해 받은 데이터를 확인 합니다.

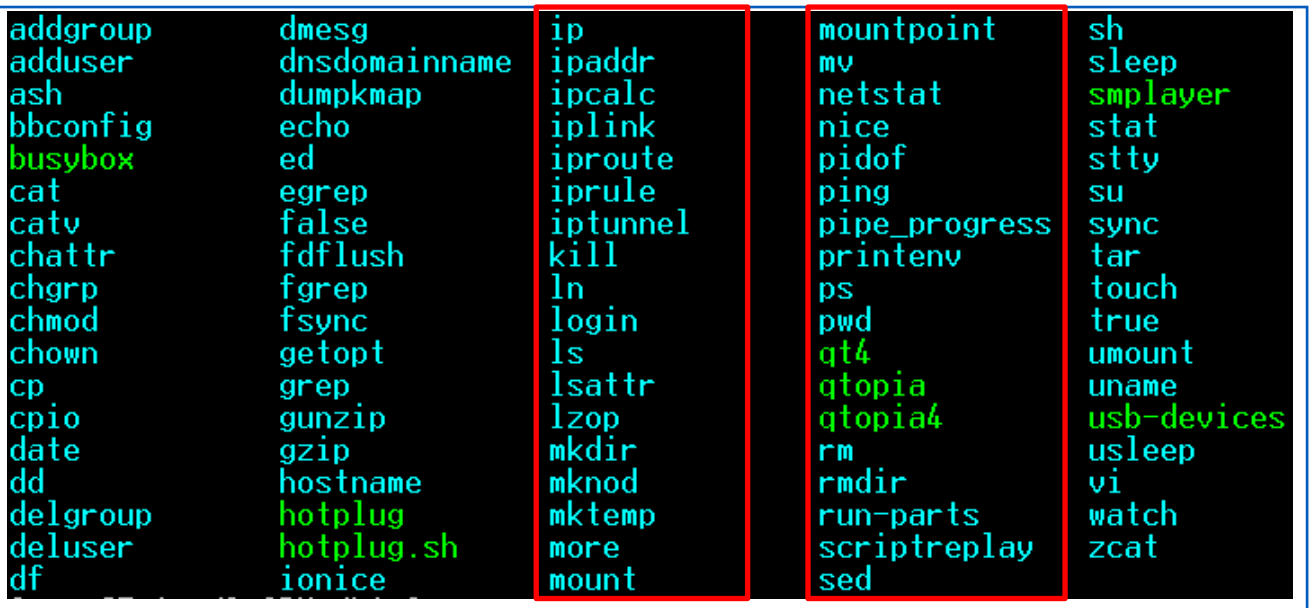
3-7. NetEye6410의 LCD 사용하기

4-1 임베디드 리눅스 명령어

콘솔 명령어에 대한 설명은 help옵션을 통해 확인 할 수 있습니다.



- addgroup : 그룹 추가 시 사용하는 명령어
- adduser : 새로운 사용자 계정 생성 명령어
- cat : 파일 생성(파일이 존재하지 않으면 새로 생성, 그렇지 않으면 파일에 이미 들어 있던 내용 위에 덮어 쓰여집니다.)
- chgrp : 파일 그룹 바꾸기
- chmod : 파일 허가권 바꾸기
- chown : 파일의 소유권 바꾸기
- cp : 파일 복사
- cpio : 파일 복사
- date : 시스템의 시간과 날짜를 보여주거나 수정
- dd : 파일의 특정 부분을 복사
- delgroup : 그룹 삭제
- deluser : 사용자 계정 삭제
- df : 사용자 자신의 HDD의 상세 정보를 보여줍니다.
- dmesg : 시스템 로그를 확인할 수 있는 명령어
- echo : 임의의 한줄 입력을 출력해서 보여줍니다.
- ed : 대화형 편집기
- fsync : 파일에 쓴 데이터와 실제 하드웨어의 동기화하 명령어
- getopt : 프로그램 실행 명령행 인자를 파싱하는 명령어
- grep : 입력 대상에서 특정 문자열을 포함한 라인을 추출하는 명령어
- gunzip : gz 파일의 압축 해제 명령어
- gzip : gzip 형태의 파일을 압축 명령어
- hostname : 호스트 이름을 출력하는 명령어



- kill : 특정 프로세서를 종료시키는 명령어
- ln : 지정한 타깃파일의 링크를 만드는 명령어
- login : 사용자 인증과정 (ID를 입력하는 명령어)
- ls : 파일과 디렉토리 구조를 보여줍니다.
- lsattr : 파일에 설정되어 있는 속성을 확인하는 명령어
- lzop : gzip과 유사한 파일 압축파일을 만드는 명령어
- mkdir : 디렉토리 생성 명령어
- mknod : 특수파일 생성(FIFO, 캐릭터 특수 파일, 블록 특수 파일) 명령어
- mktemp : 임시 파일이나 디렉토리 생성
- more : 한페이지 이상되는 출력 내용을 한 화면의 페이지 단위로 보여주며 검색 기능과 원하는 페이지로 이동 기능을 제공합니다.
- mount : 시스템에 플로피 디스크나 시디롬을 mount, 즉 시스템이 인식하게 해주는 명령어
- mountpoint : 디렉토리 마운트 포인트 유무확인 하는 명령어
- mv : 파일 이동 혹은 파일 이름 변경
- netstat : 외부와의 연결상태를 파악하거나 라우팅테이블을 확인 할 수 있는 명령어
- nice : 커널 스케줄러 내에 해당 프로세스의 우선순위를 지정하는 명령어
- pidof : 실행중인 프로그램의 프로세스 ID를 찾는 명령어
- ping : 외부 네트워크와 연결이 정상적으로 이루어졌는지 확인하는 명령어
- printenv 환경의 전부 또는 일부를 출력하는 명령어
- ps : 프로세스의 사용자, CPU/Memory점유율 등 각종 프로세스 정보를 옵션에 맞추어 출력합니다.
- pwd : 현재 작업중인 디렉토리의 절대 경로를 보여주는 명령어
- rm : 파일 삭제 명령어
- rmdir : 폴더 삭제 명령어 (안에 파일이 존재하면 명령어가 실행되지 않습니다.)
- run-parts : 대상 디렉토리에 들어 있는 모든 파일명을 아스키 순서대로 실행시켜 줍니다.
- sed : ed의 기본적 기능을 따왔으며, ed는 대화형 편집기이며, sed는 스트리밍 편집기입니다.



- sh : 쉘을 불러내는 명령어
- sleep : 지정한 시간동안 멈출 수 있는 명령어 (초 단위)
- stat : 파일이나 시스템 상태를 출력해주는 명령어
- stty : 터미널 라인 설정을 변화/출력하는 명령어
- su : 사용 중에 다른 ID로 재접속하는 명령어
- sync : 시스템내의 파일버퍼를 실제 파일시스템에 적용시켜 디스크에 저장하도록 하는 명령어
- tar : 압축을 하거나 풀때 사용하는 명령어
- touch : 파일의 날짜와 시간 정보를 변경하는 명령어
- umount : 시스템에서 mount명령으로 인식된 장치들, 플로피디스크, 시디롬, 하드디스크를 인식하지 못하게 해주는 명령어
- uname : 시스템 정보 확인 명령어
- usleep : 지정한 시간동안 멈출 수 있는 명령어 (micro 초 단위)
- vi : 편집기 실행 명령어
- watch : 결과 값을 원하는 시간 간격으로 볼 수 있는 명령어
- zcat : 압축 파일을 표준 출력으로 확장하는 명령어

4-2 임베디드 리눅스 파일시스템 살펴보기



4-2-1 /bin

필수적인 실행 명령어들이 모여 있습니다. 대부분이 모든 사용자가 실행 가능한 것들이며, 우리가 쓰는 거의 모든 기본 명령어들이 담겨 있습니다. 예를 들어 cat, chmod, chown, cp, date, echo, kill, ln, ls, mkdir, Mv, ps, pwd, rm, sh, su, vi 등이 있습니다.

4-2-2 /home

사용자들의 모든 홈 디렉토리가 담겨 있습니다. 시스템을 다시 설치한다면 이디렉토리를 백업해 두는 것을 권장합니다.

4-2-3 /mnt

마운트 포인트 디렉토리로써 꼭 여기에만 마운트해야 하는 것은 아닙니다. 하지만 만들어진 곳에 만들어주는 것을 권장합니다.

4-2-4 /root

Root(수퍼 유저)의 홈 디렉토리 입니다. /home에 위치하지 않고 바로 /root를 이용합니다.
이유는 root(수퍼 유저)외에 다른 유저가 함부로 접근하지 못 하도록 합니다.

4-2-5 /sys

리눅스 시스템이 필요로 하는 파일들의 디렉토리입니다. 손대지 않는 것을 권장합니다.

4-2-6 /var

var 디렉토리 아래에는 시스템 작동 중 변경되는 파일들이 담겨 있습니다. 로그 파일이나 스풀 파일들이 있는데, 즉, 다른 시스템과 공유가 되지 않음을 의미합니다.

4-2-7 /dev

각종 디바이스 파일들이 위치해 있는데 크게 **블록 디바이스**와 **캐릭터 디바이스**로 나눌 수 있습니다. 블록 디바이스란 HDD와 같은 주변 장치를 말하는데, 데이터가 블록 단위로 읽고 쓰여지며 랜덤하게 액세스할 수 있습니다. 반면 캐릭터 디바이스는 입출력이 한 바이트 단위로 이루어지며 데이터가 순차적으로 읽고 쓰여집니다. 디바이스를 새로 만들 때에는 **mknod** 명령을 이용하면 되며, 물론 /bin에 위치해 있습니다.

4-2-8 /lib

공유 라이브러리와 커널 모듈들이 담겨있습니다. 이전에 /bin과 /sbin 디렉토리에 있는 실행 명령어들이 실행될 때 필요한 공유 라이브러리들이 존재합니다. 파일을 추가만 하시고 삭제는 하지 않는 것을 권장합니다. 시스템이 멈출 수 있습니다.

4-2-9 /sbin

수퍼 유저가 사용하는 명령어들이 거의 다 들어 있습니다. 그러므로 물론 일반 유저는 사용할 수 없습니다. 예를 들면 halt, reboot, fdisk, mkfs등이 있습니다. 잘못 조작시 전체 시스템에 이상을 초래할 수도 있습니다.

4-2-10 /tmp

임시 디렉토리입니다. 임시로 파일들의 간이 저장 창고 역할을 합니다.

4-2 임베디드 리눅스 파일시스템 살펴보기



4-2-11 /etc

/etc 디렉토리에는 시스템 설정 파일들이 모여 있습니다. 시스템을 백업하고자 할 때는 꼭 이 디렉토리를 가장 먼저 백업하는 것을 권장합니다.

전체 사용자의 암호를 갖고 있는 /etc/passwd, X Window, Fonts 설정파일이 있습니다.

- **/etc/init.d/rcs.d(실행 프로그램 디렉토리)**

MS윈도우의 config.sys에 해당한다고 보시면 됩니다. 하지만 파일이 하나가 아니고 한 디렉토리가에 찹니다. 리눅스 시작시 시동해야 할 데몬이나 스크립트들이 모여있습니다.

- **/etc/fstab(마운트 정보 파일)**

파일 시스템의 마운팅 여부를 설정합니다. 하드나 파티션을 추가한 경우 이 곳의 설정을 변경합니다. Fstab은 시스템에 아주 중요한 파일입니다. 파일이 잘못된 정보를 갖거나 존재하지 않을 경우엔 부팅이 안될 수 있습니다.

- **/etc/passwd(사용자 계정 정보 파일)**

Passwd 파일은 리눅스 사용자들의 계정 정보와 패스워드, 사용 쉘 및 홈 디렉토리에 대한 정보를 가지고 있습니다.

- **/etc/group(그룹 정보 파일)**

사용자 그룹에 대해 정의되어 있는 파일로, 시스템의 모든 계정은 한 개 이상의 그룹에 포함되어 있으며, 그룹에 대한 변경은 직접 파일을 수정하거나 명령어를 이용하여서 변경할 수 있습니다.

4-2-12 /proc

일종의 가상 파일 시스템으로 메모리 정보만을 담고 있습니다. 우리가 사용하는 프로그램중에서 ps 프로그램이 여기에서 프로세스 정보 등을 참조합니다.

4-2-3 /usr

/usr 디렉토리는 시스템에 필수적인 요소는 아니지만 사용자에게는 아주 중요한 디렉토리입니다.

거의 모든 응용프로그램이 이곳에 위치하며 일부 설정 파일도 들어 있습니다.

라이브러리도 존재하고 컴파일 환경도 위치하고 있습니다. 무엇보다도 X Window가 여기에 설치 되어 있습니다.

- **/usr/bin**

우리가 사용하는 대부분의 응용프로그램이 이곳에 위치합니다. 대부분의 사용자 실행 명령어들이 담겨 있습니다. gcc나 perl등의 개발도구도 여기 담겨 있습니다.

- **/usr/X11R6**

X 윈도우에 관계되는 모든 프로그램, 설정 파일, 그리고 지원 파일들이 들어 있습니다.

- **/usr/include**

C 컴파일러에 대한 헤더 파일을 포함하고 있습니다..

- **/usr/info**

여러가지 GNU 프로그램(gcc, make, autoconf..)의 info 파일들이 담겨 있습니다. 이 디렉토리를 이용하는 방법은 단순히 "info 프로그램 ex) info gcc –"라고 치면 됩니다.

- **/usr/lib**

/lib와 같은 기능을 하지만 주로 응용프로그램에 관련된 라이브러리가 위치합니다.

- **/usr/local**

새로운 프로그램들이 설치되는 곳으로 프로그램 관리를 용이하게 할 수 있도록 되어 있습니다. Make install로 프로그램을 설치시 기본 디렉토리가 대부분 /usr/local 로 되어 있는 이유입니다.

- **/usr/man**

Info와 비슷한 역할을 합니다. 매뉴얼 페이지가 담겨 있는 것으로 역시 'man 프로그램'을 쳐서 이용할 수 있습니다. 모르는 명령어들은 man으로 확인하는 것을 권장합니다.

- **/usr/doc**

각종 리눅스에 관한 문서들이 담겨 있습니다.

- **/usr/src**

컴파일 되지 않은 다양한 프로그램 소스들이 들어 있습니다. 비디오 드라이버 모듈을 만들기도 하고 RPM을 만들어 내기도 합니다. 리눅스 커널 소스도 이곳에 위치하고 있습니다.

4-4 개발 환경 구축

4-4-1. 크로스 개발 환경

리눅스 커널에서 Qtopia/QT4, bootloader 및 기타 응용 프로그램을 실행 가능한 파일로 컴파일 하려면 크로스 개발 환경이 필요합니다. NetEye6410 전용 크로스 컴파일러는 arm-linux-gcc-4.5.1입니다. 이 컴파일러는 ARMv6에 명령어 세트(Instruction Set)를 지원합니다.

크로스 개발 환경 구축

- 1단계 : 제공되는 압축 파일 "arm-linux-gcc-4.5.1-v6-vfp-20101103.tgz"을 특정 디렉토리에 다운로드 또는 복사 하십시오. 예를 들어 "tmp₩" 디렉토리에 복사하고 다음 명령을 실행 합니다.

#cd ₩tmp

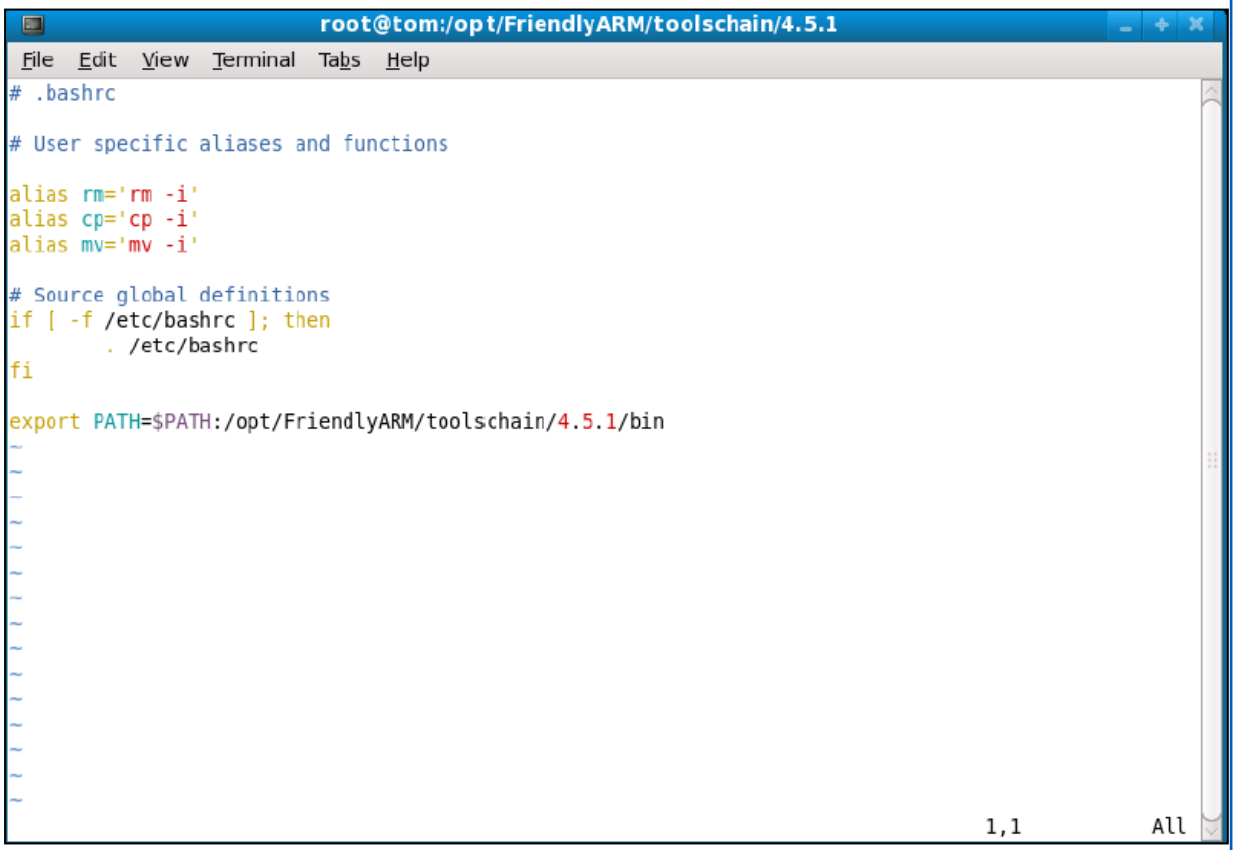
#tar xvzf arm-linux-gcc-4.5.1-v6-vfp-20101113.tgz –c /

위 명령은 "arm-linux-gcc"를 **"/opt/FriendlyARM/toolschain/4.5.1**"에 설치 합니다.

- 2단계 : 시스템 변수에 컴파일러의 경로를 추가 합니다. 다음 명령을 실행 합니다.

#gedit/root/.bashrc

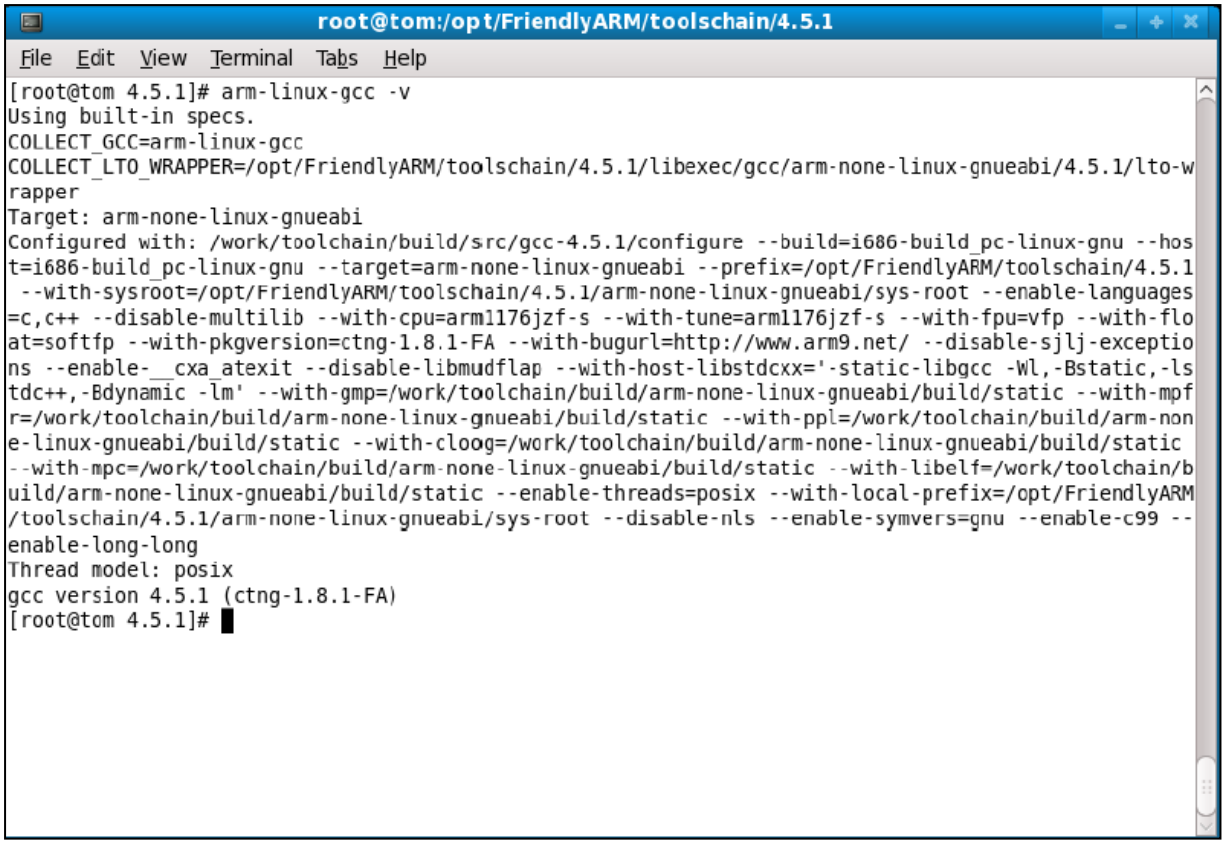
해당 파일에 "export PATH=$PATH:/opt/FriendlyARM/toolschain/4.5.1/bin"을 마지막 줄에 작성 하십시오. 작성이 완료되면 해당 파일을 저장하고 종료 하십시오.



크로스 컴파일러를 설치 한 뒤에, 시스템을 재부팅 하십시오.

arm-linux-gcc –v

명령어를 이용해 크로스 컴파일러가 성공적으로 설치되었는지 확인하십시오.



SM Information & Communication

Homepage : http://www.smic21.com
Sales Support : +82-2-6292-2100 / khg@smic21.com
Address : A-401, SK TwinTechTower, 345-9 Gasan-Dong, Geumcheon-Gu, Seoul, Korea (153-023)