

제08강

시리얼 통신 제어

시리얼 통신

시리얼 모니터링

루프백 시리얼 통신

Win. PC와의 시리얼 통신

가상 머신과의 시리얼 통신

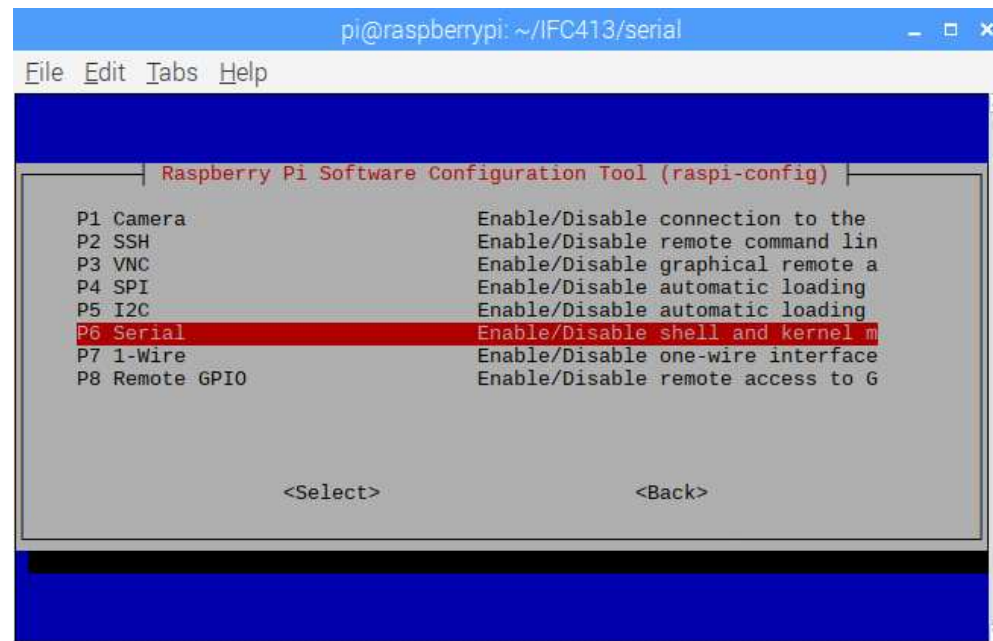
아두이노 보드와의 시리얼 통신

~~KUT51보드와의 시리얼 통신~~

시리얼통신

* Serial 활성화후 재부팅

\$ sudo raspi-config



\$ sudo reboot

: 시리얼통신을 위한 장치 파일 /dev/ttyS0 생성

시리얼통신(계속)

* 몇가지 확인사항

\$ ls /dev/tty* -al

```
crw--w---- 1 root tty      4,  8 Jul 11 10:48 /dev/tty8
crw--w---- 1 root tty      4,  9 Jul 11 10:48 /dev/tty9
crw-rw---- 1 root dialout 204, 64 Jul 11 10:48 /dev/ttyAMA0
crw--w---- 1 root tty      4, 64 Jul 11 10:48 /dev/ttyS0
crw----- 1 root root      5,  3 Jul 11 10:48 /dev/ttyprintk
pi@raspberrypi:~ $
```

: 디바이스 파일 **/dev/ttyS0** 이 추가생성된 것을 확인

: 이 장치 파일의 내장된 접근 권한 자는 슈퍼유저임을 유의

\$ cat /boot/config.txt

```
# Enable audio (loads snd_bcm2835)
dtparam=audio=on
start_x=1
gpu_mem=128
dtoverlay=w1-gpio
enable_uart=1
```

: enable_uart=1의 라인이 추가된 것을 확인

시리얼통신(계속)

* 몇가지 확인사항(계속)

\$ cat /boot/cmdline.txt

```
pi@raspberrypi:~/IFC413/08_UART $ cat /boot/cmdline.txt
dwc_otg.lpm_enable=0 console=serial0,115200 console=tty1 root=PARTUUID=5605427a-02 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait
pi@raspberrypi:~/IFC413/08_UART $
```

: 내장된 **baudrate**는 115200 임을 확인가능

\$ dmesg | grep tty

```
pi@raspberrypi:~/IFC413/08_UART $ dmesg | grep tty
[ 0.000000] Kernel command line: coherent_pool=1M 8250.nr_uarts=1 bcm2708_fb.fbwidth=1280 bcm2708_fb.fbheight=1024 bcm2708_fb.fbswap=1 vc_mem.mem_base=0x3ec00000 vc_mem.mem_size=0x40000000 dwc_otg.lpm_enable=0 console=ttyS0,115200 console=tty1 root=PARTUUID=5605427a-02 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait
[ 0.000908] console [tty1] enabled
[ 1.029007] 3f201000.serial: ttyAMA0 at MMIO 0x3f201000 (irq = 81, base_baud = 0) is a PL011 rev2
[ 1.037691] console [ttyS0] disabled
[ 1.044730] 3f215040.serial: ttyS0 at MMIO 0x0 (irq = 53, base_baud = 3125000) is a 16550
[ 2.074783] console [ttyS0] enabled
pi@raspberrypi:~/IFC413/08_UART $
```

: **ttyS0** 활성화 확인 가능

시리얼통신(계속)

* 시리얼 모니터링

: USB-TTL 시리얼 케이블 활용

: 회로연결(전원공급기있는 경우, 3핀만 연결)

// 흰색(RxD)핀을 rpi의 TxD와 연결,

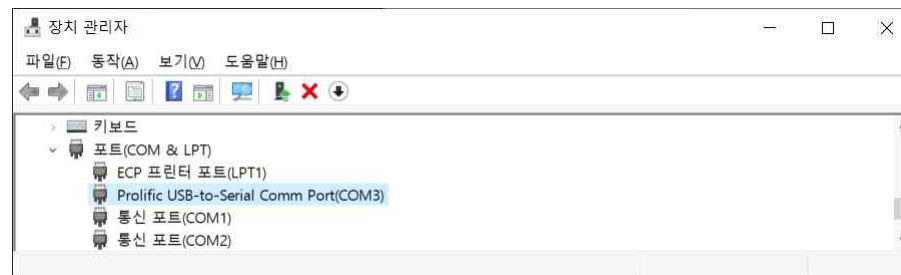
// 녹색(TxD)핀을 rpi의 RxD와 연결,

// 검정(Gnd)는 Gnd와 연결, 빨강(Vcc)는 5V(보드에 전원공급)

: 드라이버 다운로드 및 설치

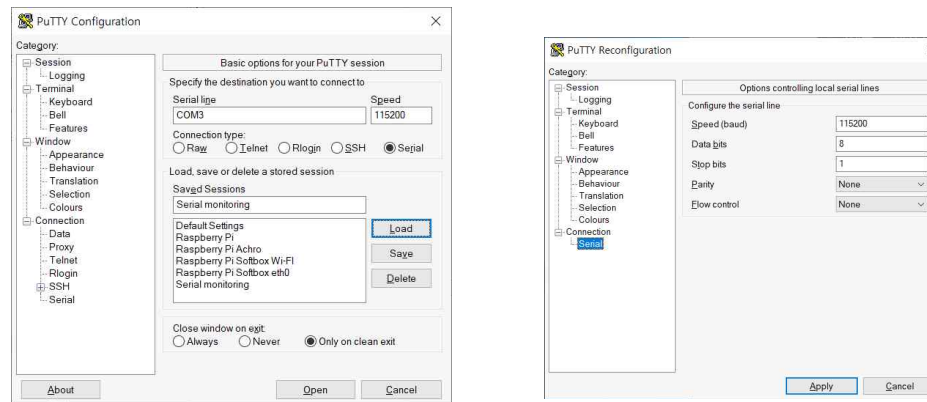
<https://learn.adafruit.com/adafruit-raspberry-pi-lesson-5-using-a-console-cable/software-installation-windows>

: 시리얼포트 확인 (Win. 장치관리자)



: PuTTY로 접속

// Flow control 항목을 필히 none으로 설정후, Open



: 라즈베리파이보드 전원 재투입, 혹은 임의키 눌러봄

// SSH를 통해 새 터미널을 여는 것이 아님

```
COM3 - PuTTY
[ 0.264286] bcm2835-aux-uart 3f215040.serial: could not get clk: -517

Raspbian GNU/Linux 9 raspberrypi ttyS0
raspberrypi login: pi
Password:
Last login: Tue May 28 01:17:09 UTC 2019 on tty1
Linux raspberrypi 4.9.41-v7+ #1023 SMP Tue Aug 8 16:00:15 BST 2017 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~$ ls
Desktop  Downloads  Pictures  python_games  Videos
Documents  Music      Public    Templates
pi@raspberrypi:~$
```

시리얼통신(계속)

*** wiringSerial.h 라이브러리(/usr/include)**

: 시리얼 통신을 위한 함수 제공

–시리얼통신을 위한 디바이스파일과 보레이트 설정하여 열기, 닫기, 버리기

```
extern int serialOpen(const char *device, const int baud) ;  
extern void serialClose(const int fd) ;  
extern void serialFlush(const int fd) ;
```

–한 문자, 문자열, 서식지정된 문자열 전송하는 함수

```
extern void serialPutchar(const int fd, const unsigned char c) ;  
extern void serialPuts(const int fd, const char *s) ;  
extern void serialPrintf(const int fd, const char *message, ...) ;
```

–수신할 데이터가 가용한지와 한 문자를 수신하는 함수

```
extern int serialDataAvail(const int fd) ;  
extern int serialGetchar(const int fd) ;
```

루프백 시리얼통신

[실습1] 루프백 시리얼통신

- : 자신의 시스템에서 보낸 데이터를 자신이 수신하는 방식
- : ASCII 코드(1Byte) 데이터들 송신 및 수신하는 프로그램
- : TxD 핀(BCM_GPIO #14)과 RxD 핀(BCM_GPIO #15)을 직접 연결



```
$ nano uart_01.c
//=====
// uart_01.c
//      loopback
//      RxD(BCM_GPIO #14), TxD(BCM_GPIO #15)
//=====
```



```
#include <stdio.h>
#include <string.h>
#include <wiringPi.h>           // delay()
#include <wiringSerial.h>

#define BAUD    115200

int main(void) {
    int fd;
    unsigned char asc, rec;

    if((fd = serialOpen("/dev/ttyS0", BAUD)) < 0) {
        printf("Device file open error!! use sudo ...Wn");
        return 1;
    }

    printf("[UART testing..... loopback]Wn");

    asc = 65;
    while(1) {
        printf("Transmitting ... %d ", asc);
        serialPutchar(fd, asc);

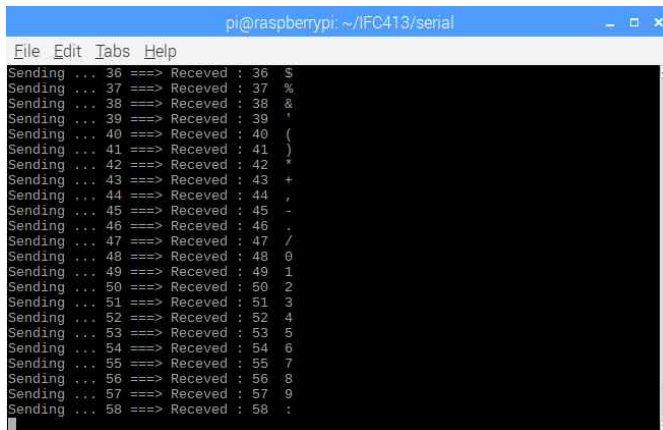
        delay(1);

        if(serialDataAvail(fd)) {
            rec = serialGetchar(fd);
            printf("==> Received : %d  %cWn", rec, rec);
        }
    }
}
```

```
        serialFlush(fd);  
    }  
  
    delay(300);  
    asc+ +;  
}  
  
return 0;  
}
```

\$ make uart_01

\$ sudo ./uart_01 // 실행시 필히

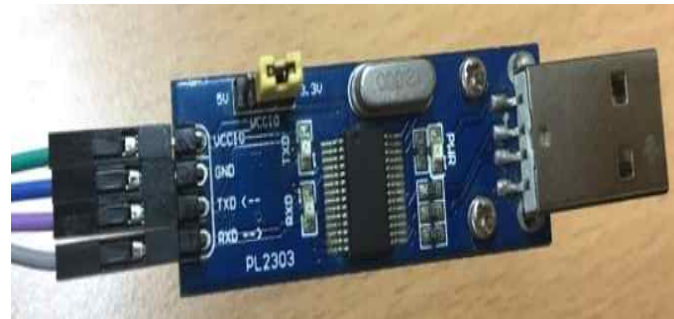


```
pi@raspberrypi: ~/IFC413/serial  
File Edit Tabs Help  
Sending ... 36 ==> Received : 36 $  
Sending ... 37 ==> Received : 37 %  
Sending ... 38 ==> Received : 38 &  
Sending ... 39 ==> Received : 39 '  
Sending ... 40 ==> Received : 40 {  
Sending ... 41 ==> Received : 41 }  
Sending ... 42 ==> Received : 42 ~  
Sending ... 43 ==> Received : 43 +  
Sending ... 44 ==> Received : 44 -  
Sending ... 45 ==> Received : 45 .  
Sending ... 46 ==> Received : 46 /  
Sending ... 47 ==> Received : 47 0  
Sending ... 48 ==> Received : 48 1  
Sending ... 49 ==> Received : 49 2  
Sending ... 50 ==> Received : 50 3  
Sending ... 51 ==> Received : 51 4  
Sending ... 52 ==> Received : 52 5  
Sending ... 53 ==> Received : 53 6  
Sending ... 54 ==> Received : 54 7  
Sending ... 55 ==> Received : 55 8  
Sending ... 56 ==> Received : 56 9  
Sending ... 57 ==> Received : 57 :  
Sending ... 58 ==> Received : 58 :
```

Win PC와의 시리얼통신

* USB 시리얼 포트경우

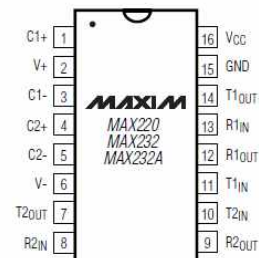
: USB to Serial케이블, **혹은 PL2303 모듈 사용**



Win PC와의 시리얼통신(계속)

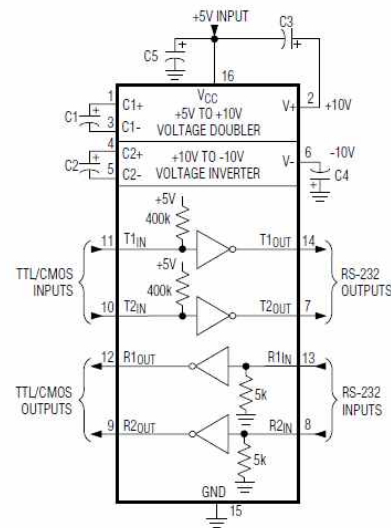
* MAX232CPE

: 신호레벨 변환 IC



DIP/SO

CAPACITANCE (μF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1



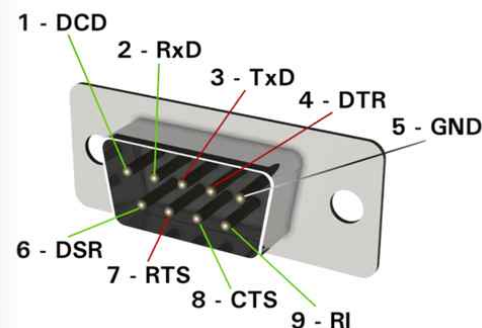
Win PC와의 시리얼통신(계속)

* RS232C 커넥터(D-Sub 9)

: 사용할 핀은 #2(RxD) , #3(TxD), #5(GND)이며,

: #5(GND) 핀은 사용하지 않아도 됨

number	name	description	level	Dir.	Etc.
1	DCD	Data Carrier Detect	RS232	Input	Mandatory
2	RXD	Receive Data	RS232	Input	Mandatory
3	TXD	Transmit Data	RS232	Output	Mandatory
4	DTR	Data Terminal Ready	RS232	Output	Optional
5	GND	Ground	Ground	-	Mandatory
6	DSR	Data Set Ready	RS232	Input	Optional
7	RTS	Request To Send	RS232	Output	Optional
8	CTS	Clear To Send	RS232	Input	Optional
9	RI	Ring Indicator	RS232	Input	Optional



Win PC와의 시리얼통신(계속)

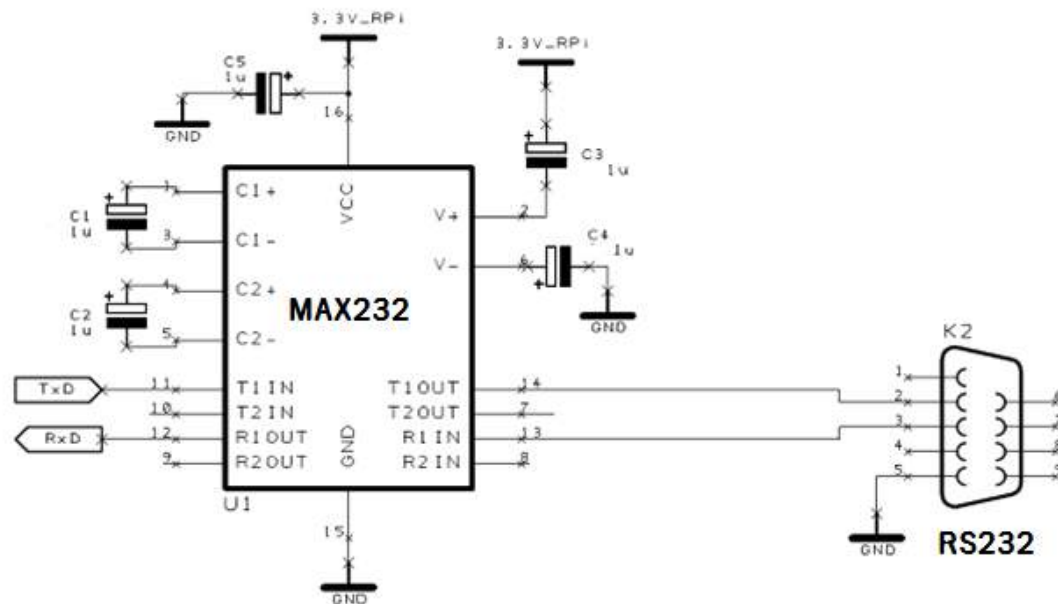
* 회로구성

: MAX232 IC와 RS232 컨넥터간의 TxD, RxD 단자는 서로 교차로 연결

: 아래 그림과 같이 구성하면 교차 연결된 것임

(T1OUT - RxD, R1IN - TxD)

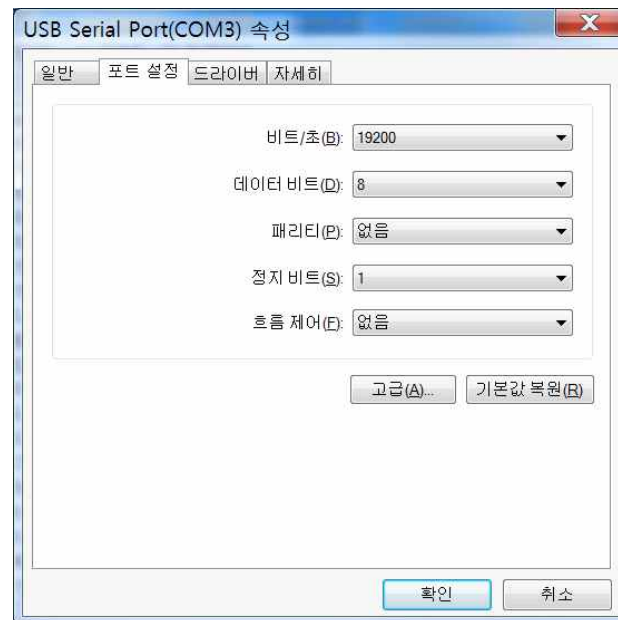
: 1uP 캐패시터 5개, 기판고정용 9핀 암 RS232 컨넥터 1개 소요



Win PC와의 시리얼통신(계속)

* USB Serial Port 설정(Win PC)

- : USB 시리얼케이블 연결후, 장치관리자에서 시리얼포트 확인
- : 추가 시리얼포트 없는 경우 관련 드라이버 설치할 것
- : 통신 속성 정보 설정(19200, 등등)



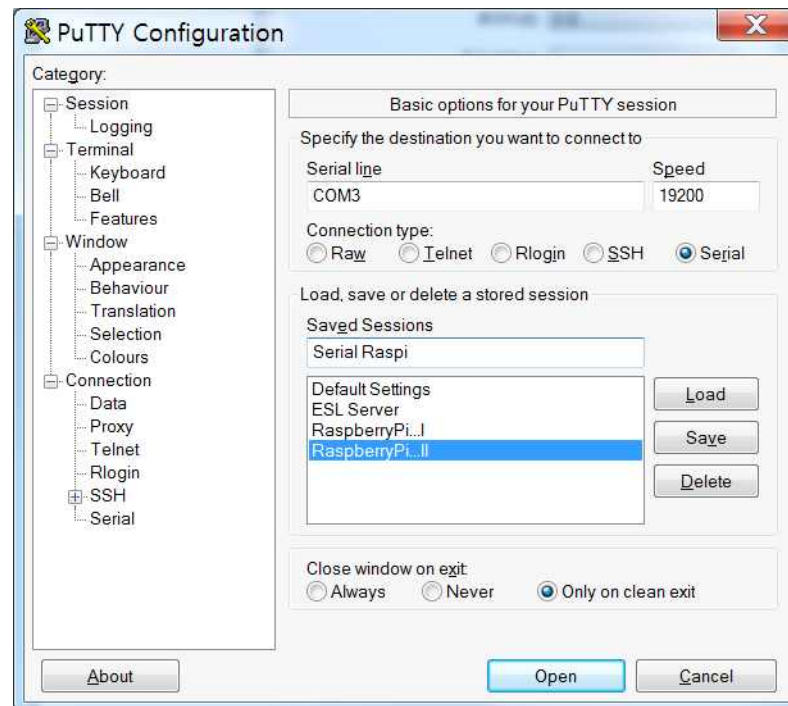
Win PC와의 시리얼통신(계속)

* PuTTY 통신프로그램

: 연결 유형에서 Serial을 체크

: 통신포트 및 보 레이트를 설정후 Open 클릭

: 설정정보 저장하여 추후 운영시 활용가능



Win PC와의 시리얼통신(계속)

[실습2] Win PC와의 시리얼통신

: PC에서 전송한 데이터를 수신후 되전송하는 프로그램

```
$ nano uart_02.c

//=====
// uart_02.c
//      with Windows PC
//      retransmit a received char from Win PC
//=====
#include <stdio.h>
#include <string.h>

#include <wiringSerial.h>

#define BAUD      19200          // 115200

int main(void) {
    int fd;
    int rec;
```

```
if((fd = serialOpen("/dev/ttyS0", BAUD)) < 0) {
    printf("Device file open error!! use sudo ...Wn");
    return 1;
}

printf("[UART test with Win PC...]Wn");
serialPuts(fd, "[UART test with Win PC...]Wn");

while(1) {
    if(serialDataAvail(fd)) {
        rec = serialGetchar(fd);
        printf("Received from Win PC : %d %c Wn",
               rec, (char)rec);

        // re-transmit..
        serialPutchar(fd, rec);
        serialFlush(fd);
    }
}

return 0;
}
```

\$ make uart_02

Win PC와의 시리얼통신(계속)

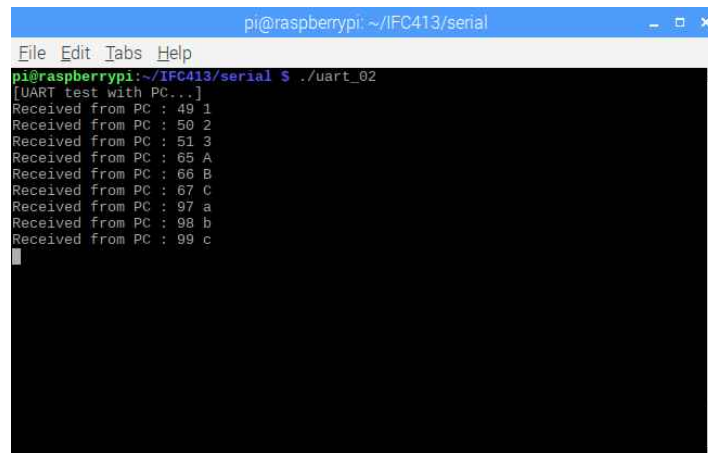
* 실습절차

단계1) Windows PC에서 PuTTY 시리얼통신 프로그램을 실행

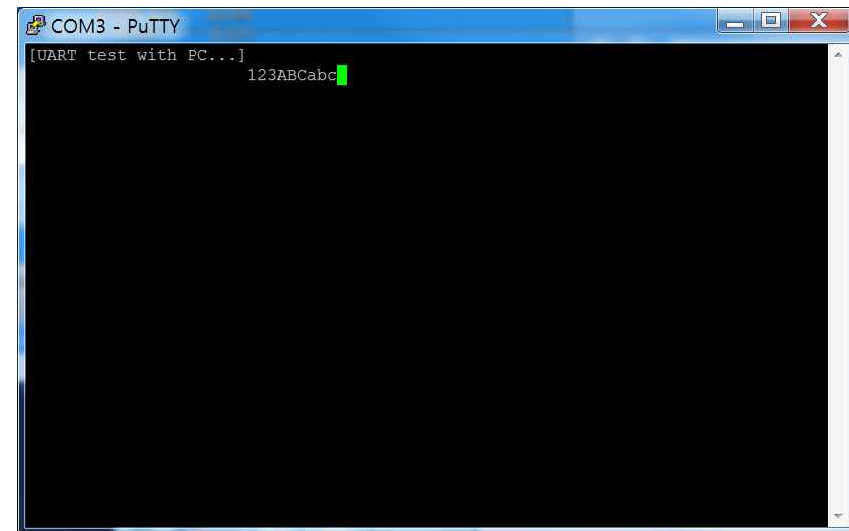
단계2) 라즈베리파이에서 다음과 같이 실행

```
$ sudo ./uart_02
```

단계3) Windows PC에서 PuTTY 화면에서 문자 입력



```
pi@raspberrypi: ~/IFC413/serial
File Edit Tabs Help
pi@raspberrypi:~/IFC413/serial $ ./uart_02
[UART test with PC...]
Received from PC : 49 1
Received from PC : 50 2
Received from PC : 51 3
Received from PC : 65 A
Received from PC : 66 B
Received from PC : 67 C
Received from PC : 97 a
Received from PC : 98 b
Received from PC : 99 c
```



```
COM3 - PuTTY
[UART test with PC...]
123ABcabc
```

Win PC와의 시리얼통신(계속)

[실습3] 시리얼통신에 의한 LED 제어

: Win PC에서 전송되는 문자 중 1, 혹은 0에 따라
라즈베리파이 보드의 LED를 ON/OFF하는 프로그램

```
$ nano uart_03.c
```

```
//=====
// uart_03.c
//      with Windows PC
//      control LED on Raspberry Pi board
//=====
#include <stdio.h>
#include <string.h>
#include <wiringPi.h>

#include <wiringSerial.h>

#define P_LED    1                // BCM_GPIO #18
#define BAUD     19200            // 115200

int main(void) {
```

```
int fd;
int rec;

if((fd = serialOpen("/dev/ttyS0", BAUD)) < 0) {
    printf("Device file open error!! use sudo ...Wn");
    return 1;
}

if(wiringPiSetup() == -1)
    return 1;

pinMode(P_LED, OUTPUT);

printf("[UART test with Win PC + LED control]Wn");
serialPuts(fd, "[UART test with Win PC + LED control]Wn");

while(1) {
    if(serialDataAvail(fd)) {
        rec = serialGetchar(fd);
        printf("Received from Win PC : %d %c ",
               rec, (char)rec);

        // re-transmit..
        serialPutch(char, rec);
        serialFlush(fd);

        //// LED control....
        if(rec == '0') {
```

```
        digitalWrite(P_LED, LOW);
        printf("==> Led OFF....\n");
    }
    else if(rec == '1') {
        digitalWrite(P_LED, HIGH);
        printf("==> Led ON....\n");
    }
    else
        printf("==> No control data....\n");

    }

}

return 0;
}
```

\$ make uart_03

```
pi@raspberrypi:~/IFC413/serial $ ./uart_03
[UART test with PC + LED control]
Received from PC : 97 a ==> No control data....
Received from PC : 115 s ==> No control data....
Received from PC : 100 d ==> No control data....
Received from PC : 49 1 ==> Led ON....
Received from PC : 97 a ==> No control data....
Received from PC : 100 d ==> No control data....
Received from PC : 48 0 ==> Led OFF....
Received from PC : 65 A ==> No control data....
Received from PC : 68 D ==> No control data....
```

```
COM3 - PuTTY
[UART test with PC + LED control]
asdiad0Ad
```

가상머신과의 시리얼통신

* 요구사항

: VM에 시리얼포트 추가

: VM의 통신프로그램 mincom 설치 및 환경설정

가상머신과의 시리얼통신(계속)

* 가상머신 환경설정

: Win의 장치관리자에서 시리얼포트 설정확인

- 기본 COM1(마더보드의 통신포트)
- USB-Serial Adaptor 경우 기타의 COMx

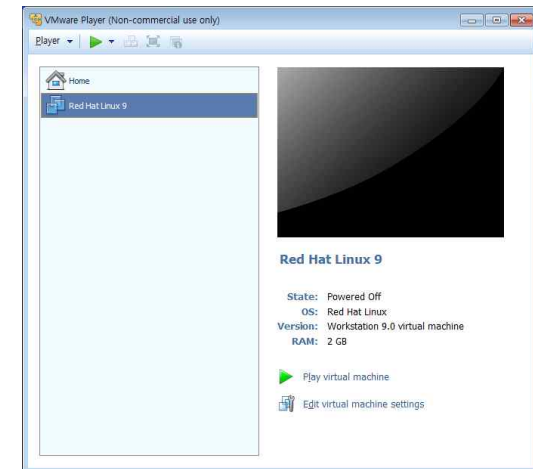
: **시리얼포트** 추가

"Player-Manager-VM settings"클릭

"Hardware탭"에서

"Add" 클릭하여

Serial Port 추가, COM1 설정



: **CD/DVD항** **바른드라이브**(Windows와 동일) 지정!

가상머신과의 시리얼통신(계속)

* minicom

- : 텍스트방식의 터미널에뮬레이션 통신 프로그램
(Windows의 hyper terminal과 유사)
- : 모뎀 제어, 파일 업로드/다운로드 등 통신에 필요한
주요 기능 포함
- : 임베디드시스템 개발 과정에서
타깃보드와 개발호스트간 시리얼 통신을 위해 사용
- : 타깃보드에 명령 전달 및 출력 메시지 확인용

* minicom 설치

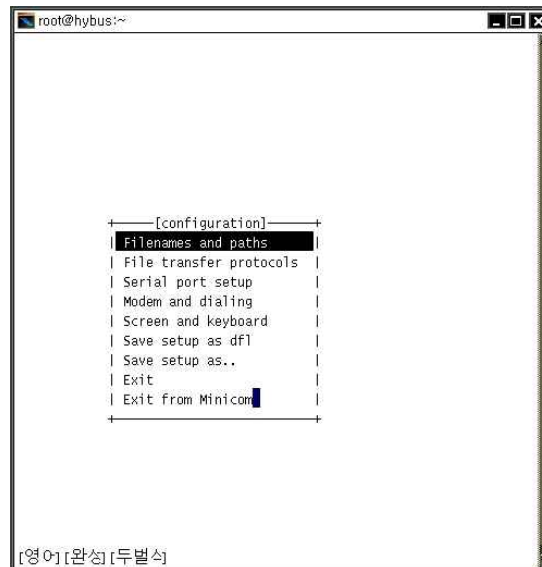
```
root@ubuntu:~# apt install minicom
```

가상머신과의 시리얼통신(계속)

* minicom 환경 설정

: 통신 포트, 통신 속도 등 설정이 필요

minicom -s



가상머신과의 시리얼통신(계속)

* minicom 직렬 포트 환경 설정

: 'Serial Port Setup'항목 선택

A-Serial Device	: <u>/dev/ttyS1</u>
B-Lockfile Location	: /var/lock
E-Bps/Par/Bits	: <u>115,200bps 8N1</u>
F-Hardware Flow Control	: <u>No</u>

- Serial Device ... 시리얼포트1(/dev/ttyS0), **시리얼포트2(/dev/ttyS1)**
- Bps/Par/Bits ... 통신 속도 및 패리티 비트 등 설정
- Hardware Flow Control ... **No로**

: 직렬 포트 환경 저장

- 'Save setup as df1'항목 선택

: 종료

- 'Exit from minicom'항목 선택

가상머신과의 시리얼통신(계속)

- * 다음과 같이하여 동작 확인!!(2019.09.05.)
 - : USB-TTL Serial cable 연결(Win관리자에서 COM3)
 - : vm에서 시리얼포트2(COM3)선택 상태
 - : /dev/ttyS1 설정

```
+-----+
| A -   Serial Device       : /dev/ttyS1
| 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?
+-----+
| Screen and keyboard |
```

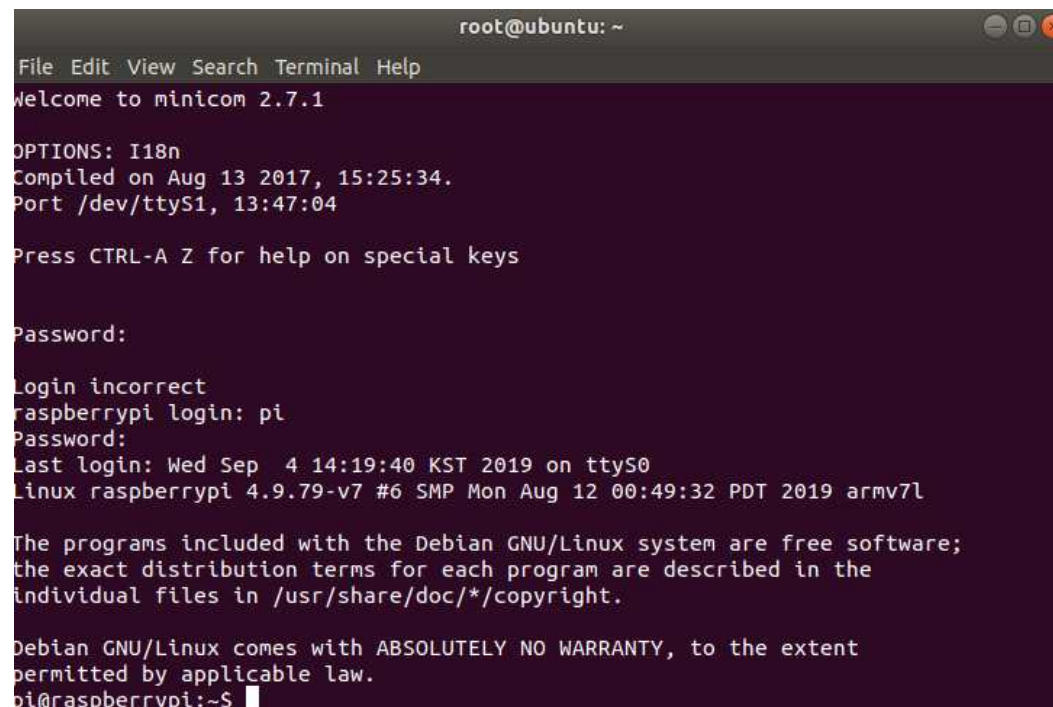
참고) Windows와 VM간 USB시리얼 전환시

- : VM 플레이어 우상단 USB 아이콘 및 시리얼포트 아이콘 활용

가상머신과의 시리얼통신(계속)

* 통신확인

minicom



```
root@ubuntu: ~
File Edit View Search Terminal Help
Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyS1, 13:47:04

Press CTRL-A Z for help on special keys

Password:
Login incorrect
raspberrypi login: pi
Password:
Last login: Wed Sep 4 14:19:40 KST 2019 on ttyS0
Linux raspberrypi 4.9.79-v7 #6 SMP Mon Aug 12 00:49:32 PDT 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
pi@raspberrypi:~$
```

: 종료시 Ctrl-A-q 누름

가상머신과의 시리얼통신(계속)

[실습4] 시리얼통신에 의한 LED 제어

: 가상머신에서 전송되는 문자 중 1, 혹은 0에 따라
라즈베리파이 보드의 LED를 ON/OFF하는 프로그램

: [실습3] 참조, 소스 그대로 사용 테스트

* 실습 절차

: 가상머신에서 통신프로그램 실행(minicom)

minicom

: 라즈베리파이에서

\$ sudo ./uart_03

Arduino보드와의 시리얼통신

* Arduino Mega ADK 보드

- : USB-Serial 컨버터가 내장되어 있음
- : 이 컨버터를 경유하는 시리얼통신으로 목적 코드 업로드함
- : 4개 시리얼 포트 제공
- : 목적코드 업로드시 사용 포트 ... 시리얼 포트 0 (serial)

* Arduino 보드측 회로

- : 아두이노 보드의 pin #3에 LED 회로 연결



Arduino보드와의 시리얼통신(계속)

* Arduino 보드용 소스

- : 스케치에서 다음의 소스를 작성하여 컴파일
- : 수신한 문자가 0이면 LED OFF, 1이면 ON 기능
- : 통신속도는 19200으로.

```
//=====
// uart_04_arduino.ino
//      for Arduino board
//      LED controlled by remote RaspberryPi
//=====
const int ledPin = 3;          // LED에 연결된 핀 번호

char data;                    // 수신 문자를 저장하기 위한 변수

void setup() {
    pinMode(ledPin, OUTPUT);    // LED를 출력으로 설정
    Serial.begin(19200);        // 시리얼 개방, 통신속도 설정
}

void loop() {
```



```
switch(data) {
    case '0' :
        digitalWrite(ledPin, LOW);    // LED OFF
        break;
    case '1' :
        digitalWrite(ledPin, HIGH);   // LED ON
        break;

    default :
        break;
}

// 수신버퍼에 데이터가 있을 때 마다 호출되는 콜백 함수
void serialEvent() {
    data = Serial.read();
    Serial.println(data);
}
```

Arduino보드와의 시리얼통신(계속)

* 두 보드간 연결 절차

1) 컴파일 후 목적코드를 업로드(플래쉬 메모리에 기록)

- : 업로딩하기 위해서는 다음의 2가지 작업 선행 필요,
- : 메뉴의 도구 항목중 보드와 시리얼포트 항목에서
아두이노 보드의 모델 설정과 ISP를 위한 시리얼 포트 설정후,
- : 업로드 아이콘을 클릭

2) PC에서 아두이노 보드 및 USB 케이블 제거

3) 두 보드간 RxD, TxD 단자를 교차하여 연결

- : 라즈베리파이 보드와 아두이노 보드간 시리얼 통신을 위해

4) 아두이노보드에 전원 인가

- : 아두이노보드에 연결된 USB 단자를 라즈베리파이보드의 USB 단자에
연결하여 아두이노 보드에 전원 인가
(아두이노보드에서 아두이노보드용 프로그램 실행중)

Arduino보드와의 시리얼통신(계속)

[실습5] 아두이노보드의 LED 제어

: 시리얼통신에 의해 아두이노보드 상의 LED를 제어

: 라즈베리파이용 소스

```
$ nano uart_04.c
```

```
//=====
// uart_04.c
//      Arduino LED Control
//      for Raspberry Pi board
//      pair file : uart_04_arduino.ino
//=====
#include <stdio.h>
#include <string.h>

#include <wiringSerial.h>

#define BAUD 19200

int main(void) {
    int fd;
```

```
char con;

if((fd = serialOpen("/dev/ttyS0", BAUD)) < 0) {
    printf("Device file open error!! use sudo ...Wn");
    return 1;
}

printf("[UART test with Arduino board]Wn");
serialPuts(fd, "[UART test with Arduino board]Wn");

while(1) {
    printf("WnInput a char: ");
    con = getchar();
    getchar();           // trash enter key
    //fflush(stdin);

    //serialPutchar(serialGetchar(fd));
    // transmit
    serialPutchar(fd, con);
    serialFlush(fd);
}

return 0;
}
```

`$ make uart_04`

\$ sudo ./uart_04

: 아두이노 보드의 LED를 제어하기 위해
0, 혹은 1의 문자를 입력

```
pi@raspberrypi:~/IFC413/serial $ ./uart_04
[UART test with Arduino board]

Input a char: a

Input a char: 1

Input a char: 0

Input a char: █
```

응용 과제

[응용 1] 시리얼 통신

: 각자 응용과제를 정의하여 구현
:

[응용 2] KUT51 보드와의 시리얼 통신

: 마이크로프로세서및실습 교과의 실습키트 활용
: 마프교재 끝장 C51 프로그래밍의 마지막 실습과제로
데이터 송수신을 테스트할 것