Chapter4
시리얼 통신

정보보안암호수학과 20182209 김수빈

먼지센서 | SDS011 ←→ 노트북

SDS011

미세먼지 센서

Hterm

먼지센서로 들어오는 데이터를 한눈에 확인가능

목표

데이터가 잘 들어오는지 확인하여 나중에 시리얼 통신 코딩했을 때
코드 이외의 부분에서 신경쓰지 않도록 미리 확인

먼지센서 | SDS011 ←→ 노트북

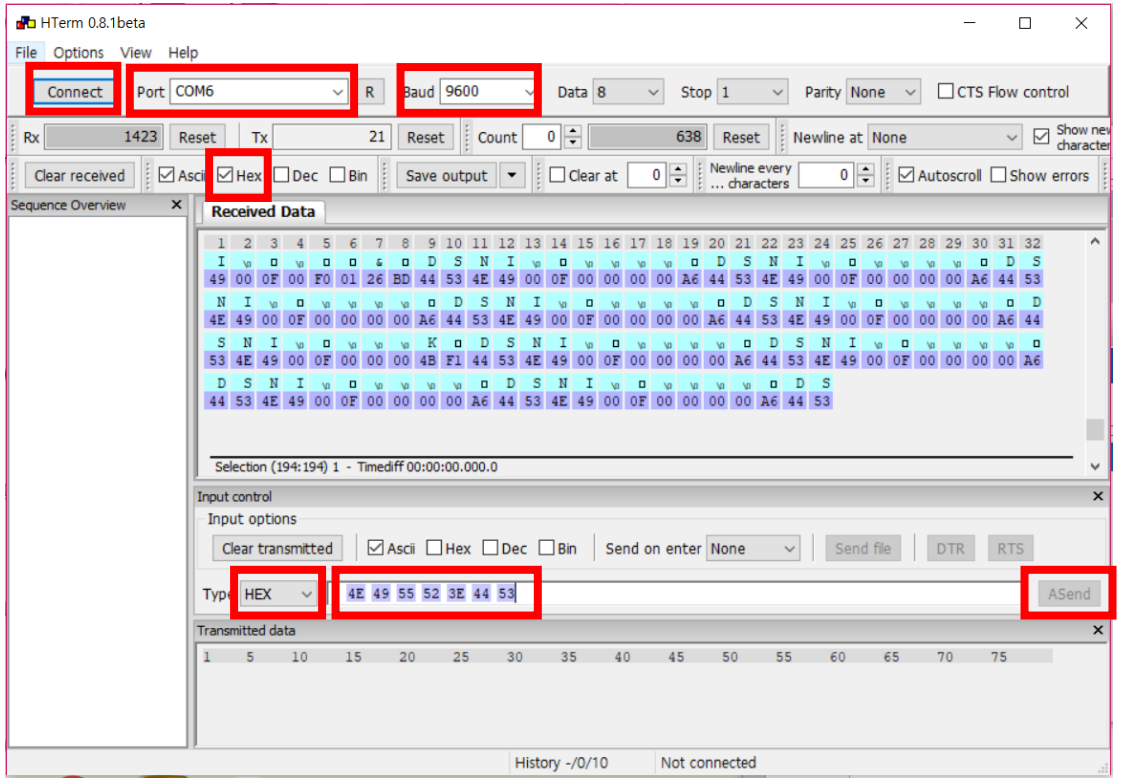
활동

1. 노트북과 SDS011을 연결한다.

Number	Symbol	Pin Description
①	GND	Ground, Connect with System ground
②	P2	Low pulse Signal output(P2) of large particle, active low PWM
	TXD	Number data output of particle (UART mode)
③	Vcc	Input Supply voltage (+5 V)
④	P1	Low pulse Signal output(P1) of small particle, active low PWM
⑤	NC	No connection
	RXD	Command receive from system (UART mode)

먼지센서 | SDS011 ←→ 노트북

2. Hterm 실행
3. Post : COM6 / Baud : 9600 → Hex 체크 → connect
4. Type : HEX → 4E 49 55 52 3E 44 53 → Asend → Start
5. Type : HEX → 4E 49 55 43 2F 44 53 → Asend → Start



→ 데이터가 잘 들어오는지 확인

• SET → Dust Sensor

STX (2Bytes)		Command (2Bytes)		Checksum	ETX (2Bytes)	
1st	2nd	3rd	4th	5th	6th	7th
0x4E	0x49	0x55	0x52	(1st+2nd+3rd+4th) & 0xFF	0x44	0x53

→ 4E 49 55 52 3E 44 53

• SET → Dust Sensor

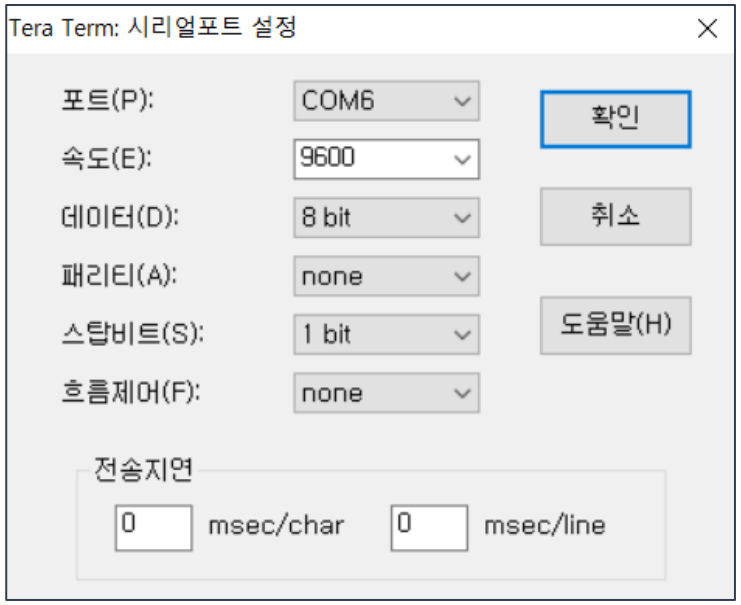
STX (2Bytes)		Command (2Bytes)		Checksum	ETX (2Bytes)	
1st	2nd	3rd	4th	5th	6th	7th
0x4E	0x49	0x55	0x43	(1st+2nd+3rd+4th) & 0xFF	0x44	0x53

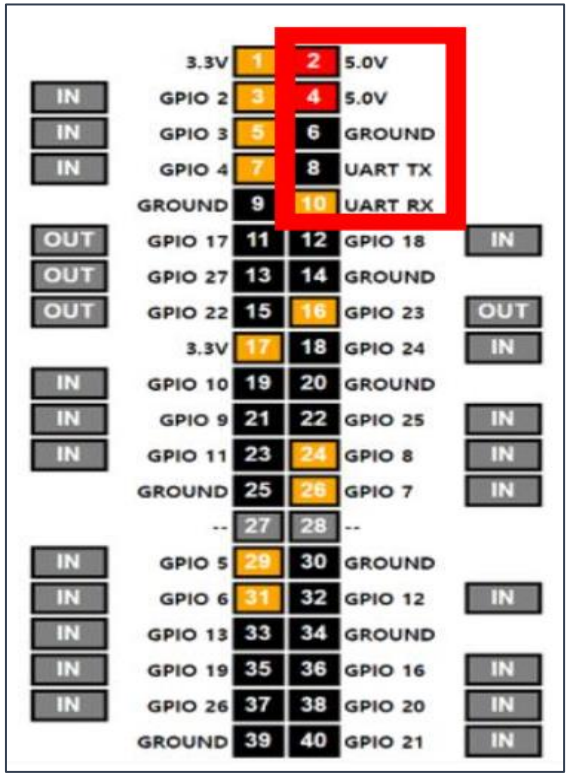
→ 4E 49 55 43 2F 44 53

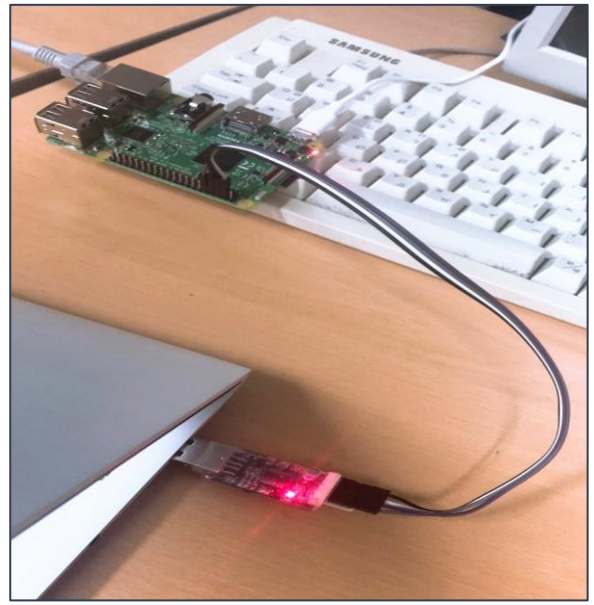
시리얼 통신 환경 구축 | 라즈베리파이 ←→ 노트북

시리얼 통신 환경 구축

1. 노트북과 라즈베리파이를 연결한다.
2. Teraterm
3. 설정 → 시리얼포트 설정
4. 포트 : COM6 / 속도 : 9600 설정

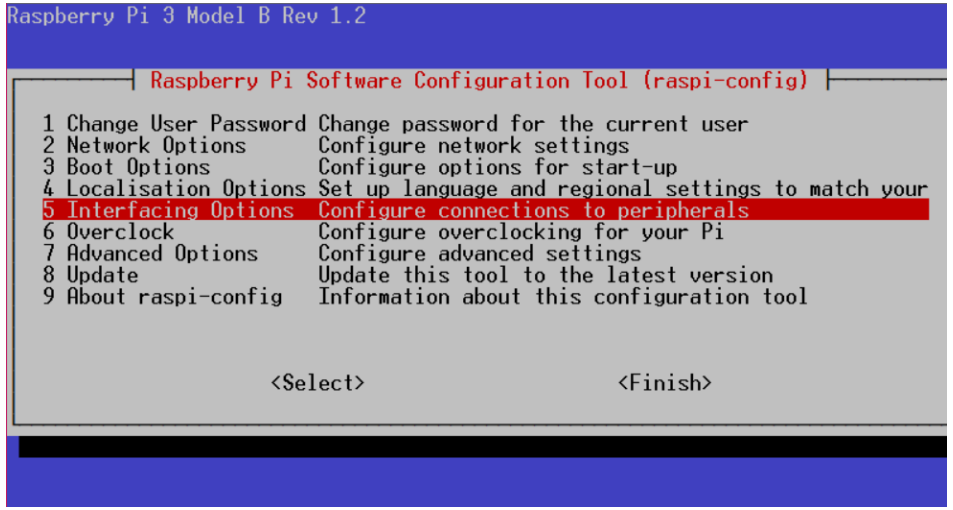


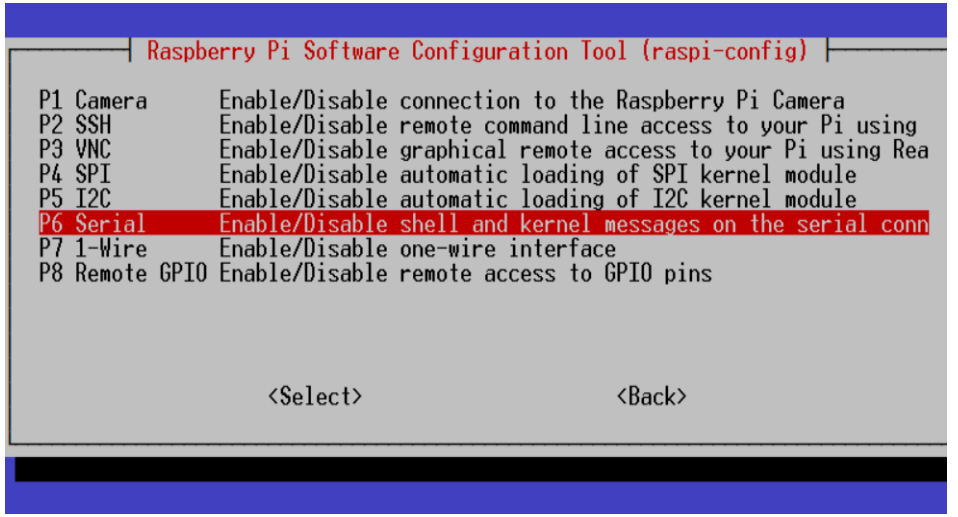




시리얼 통신 환경 구축 | 라즈베리파이 ←→ 노트북

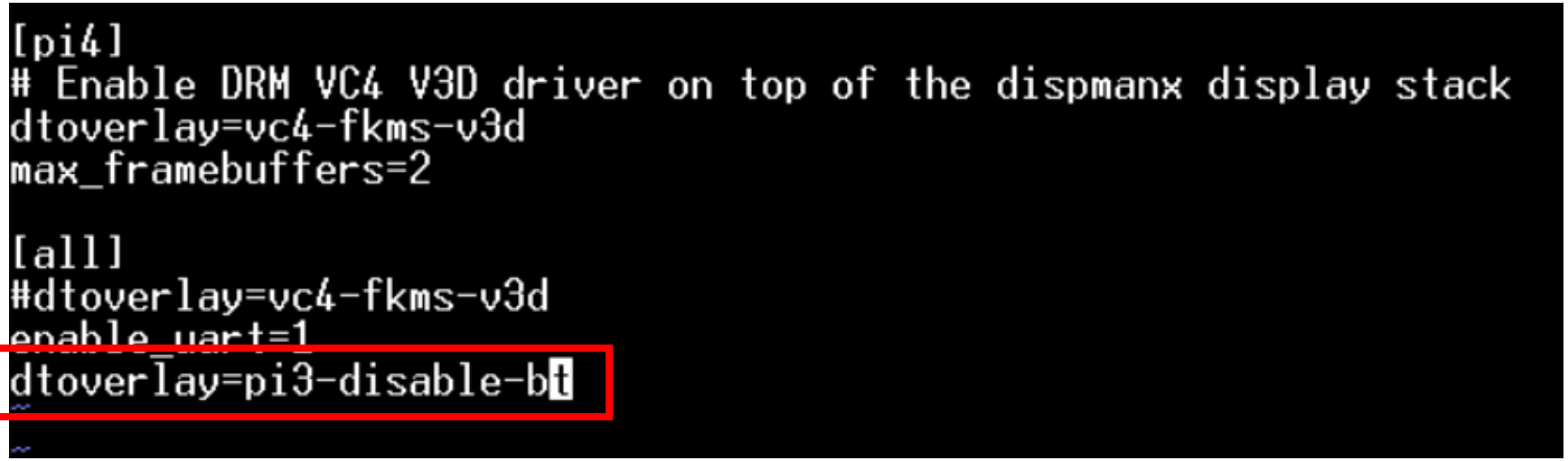
5. “sudo raspi-config” 입력
6. 5번 interfacing Options 선택
7. 6번 Serial 선택
8. No → Yes → Enter





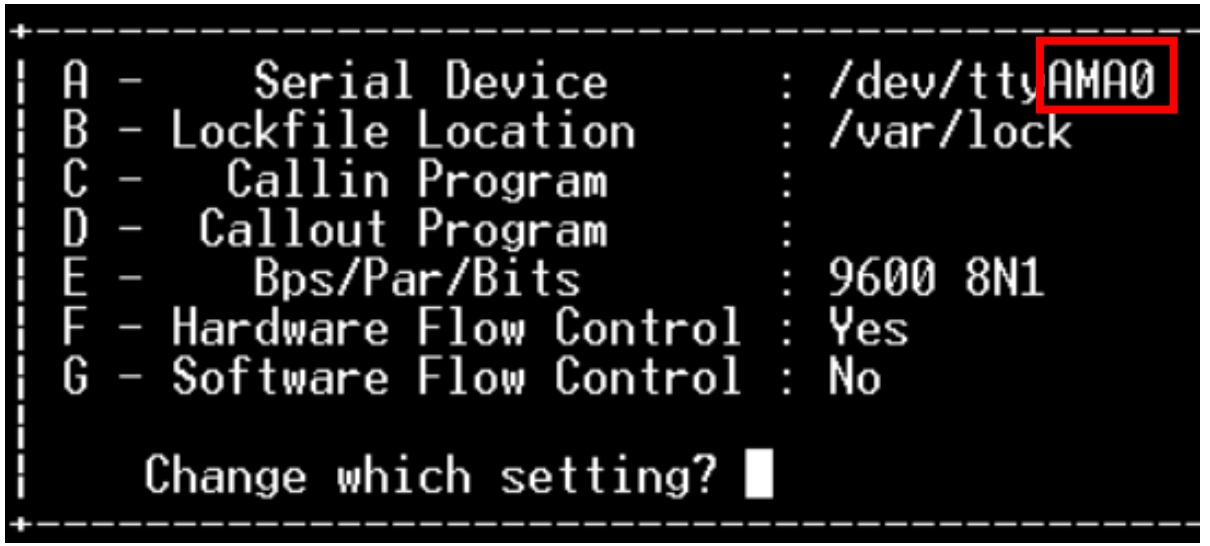
시리얼 통신 환경 구축 | 라즈베리파이 ←→ 노트북

9. “sudo vi /boot/config.txt” 입력
10. 맨 마지막에 “dtoverlay=pi3-disable-bt” 입력



시리얼 통신 환경 구축 | 라즈베리파이 ←→ 노트북

11. “sudo minicom –s” 입력
12. Serial port setup → A → “AMA0” 입력 → Enter → Save setup as df1



시리얼 통신 | SDS011 ←→ 라즈베리파이 ←→ 노트북

시리얼 통신

라즈베리파이와 SDS011와 노트북을 연결한다.

→ 시리얼 통신 잘 되는지 확인

소켓 시리얼 프로그래밍 | serial.c

serial.c 코드

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <fcntl.h>
#include <sys/poll.h>
#include <termios.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>

#define DEVPORT "/dev/ttyAMA0"

int set_uart(char* device_name, int baudrate);

int main()
{
    int serial_fd = 0;
    int i = 0, len = 0;
    unsigned char buf[128] = { 0x00, };

    serial_fd = set_uart(DEVPORT, B115200);
```

```c
    while (1)
    {
        /* method 1
        len = read(serial_fd, buf, 10);
        printf("read len : %d\n", len);
        for(i = 0; i < len; i++)
        printf("%02x ", buf[i]);
        printf("\n");
        */

        /* method 2
        len = read(serial_fd, &buf[i], 1);
        if(len == 1)
        {
        printf("%02x ", buf[i]);
        if(buf[i] == 0xab)
        printf("\n");
        i++;
        }
        */
    }

    return 0;
}
```

소켓 시리얼 프로그래밍 | serial.c

serial.c 코드

```c
int set_uart(char* device_name, int baudrate)
{
    struct termios newtio;
    int serial_fd;

    memset(&newtio, 0, sizeof(newtio));
    serial_fd = open((char *)device_name, O_RDWR | O_NOCTTY);

    printf("serial_fd : %d₩n", serial_fd);

    if (serial_fd < 0)
    {
        printf("serial fd open fail !!!₩n");
        return -1;
    }

    newtio.c_cflag = baudrate;
    newtio.c_cflag |= CS8;
    newtio.c_cflag |= CLOCAL;
    newtio.c_cflag |= CREAD;
    newtio.c_iflag = IGNPAR;
    newtio.c_oflag = 0;
    newtio.c_lflag = 0;
    newtio.c_cc[VTIME] = 1;
    newtio.c_cc[VMIN] = 0;

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

    return serial_fd;
}
```

소켓 프로그래밍 + LEA + 시리얼 통신

serial_server.c

serial_server.c = server_echo.c + LEA복호화.c

serial_client.c

serial_client.c = serial.c + client_echo.c + LEA암호화.c

1. uart setting function
2. socket setting function
3. dust sensor setting
4. serial data (dust data) read function
5. send data (server)

목표

1. 서버와 클라이언트를 연결
2. 클라이언트에서 serial로 먼지데이터 암호화해서 서버로 전달
3. 서버에서 암호화된 먼지데이터 복호화

소켓 프로그래밍 + LEA + 시리얼 통신

serial_server.c ←→ serial_client.c

192.168.0.10 - pi@raspberrypi: ~/Desktop/serial VT

메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)

```
pi@raspberrypi:~ $ cd Desktop
pi@raspberrypi:~/Desktop $ ls
Desktop  lea  serial
pi@raspberrypi:~/Desktop $ cd serial
pi@raspberrypi:~/Desktop/serial $ ls
lea.c  serial_client  serial_client.c  serial_server  serial_server.
pi@raspberrypi:~/Desktop/serial $ ./serial_server
New Client : 192.168.0.10

Encrypt data : c6 26 b3 5b 2e b1 64 7a a1 a5 bd f3 a5 1d 0e f2
Decrypt data : 4e 49 01 a4 01 59 01 ef 86 44 53 00 00 00 00 00

Encrypt data : a2 35 03 54 c2 f5 d4 d2 05 5a ee 48 7d ce ae 60
Decrypt data : 4e 49 01 aa 00 cf 03 1e 32 44 53 00 00 00 00 00

Encrypt data : 1f 53 23 4c 62 89 af 6a 56 40 d6 44 90 b2 d6 b7
Decrypt data : 4e 49 01 bc 01 29 03 81 02 44 53 00 00 00 00 00

Encrypt data : 2c 9e 8b 6d 3f 69 ae 8d bc d2 59 10 f2 e3 ce 62
Decrypt data : 4e 49 01 bc 00 2a 02 07 87 44 53 00 00 00 00 00

Encrypt data : c5 01 7c b4 bd ce 76 71 aa bc aa 87 20 12 30 d1
Decrypt data : 4e 49 01 c2 02 8e 01 c8 b3 44 53 00 00 00 00 00

Encrypt data : 1b f0 82 62 98 c6 2f ee a7 cd 8e 86 7f 9d e2 41
Decrypt data : 4e 49 01 cb 01 5c 00 0c cc 44 53 00 00 00 00 00
```

192.168.0.10 - pi@raspberrypi: ~/Desktop/serial VT

메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)

```
pi@raspberrypi:~ $ cd Desktop
pi@raspberrypi:~/Desktop $ ls
Desktop  lea  serial
pi@raspberrypi:~/Desktop $ cd serial
pi@raspberrypi:~/Desktop/serial $ ls
lea.c  serial_client  serial_client.c  serial_server  serial_server.c
pi@raspberrypi:~/Desktop/serial $ ./serial_client
serial_fd : 3
dust sensor setting done.

read len : 11
DUST DATA : 4e 49 01 a4 01 59 01 ef 86 44 53 00 00 00 00 00
Encrypt data : c6 26 b3 5b 2e b1 64 7a a1 a5 bd f3 a5 1d 0e f2

read len : 11
DUST DATA : 4e 49 01 aa 00 cf 03 1e 32 44 53 00 00 00 00 00
Encrypt data : a2 35 03 54 c2 f5 d4 d2 05 5a ee 48 7d ce ae 60

read len : 11
DUST DATA : 4e 49 01 bc 01 29 03 81 02 44 53 00 00 00 00 00
Encrypt data : 1f 53 23 4c 62 89 af 6a 56 40 d6 44 90 b2 d6 b7

read len : 11
DUST DATA : 4e 49 01 bc 00 2a 02 07 87 44 53 00 00 00 00 00
Encrypt data : 2c 9e 8b 6d 3f 69 ae 8d bc d2 59 10 f2 e3 ce 62

read len : 11
DUST DATA : 4e 49 01 c2 02 8e 01 c8 b3 44 53 00 00 00 00 00
Encrypt data : c5 01 7c b4 bd ce 76 71 aa bc aa 87 20 12 30 d1

read len : 11
DUST DATA : 4e 49 01 cb 01 5c 00 0c cc 44 53 00 00 00 00 00
Encrypt data : 1b f0 82 62 98 c6 2f ee a7 cd 8e 86 7f 9d e2 41
```

소켓 프로그래밍 + LEA + 시리얼 통신 | serial_server.c

serial_server.c

```c
#include <lea.h>
#include <unistd.h>
#include <time.h>
#include <fcntl.h>
#include <sys/poll.h>
#include <termios.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>

#define MAX_BUF_SIZE    1024
#define DEVPORT "/dev/ttyAMA0"

int set_socket();

int main()
{
    struct sockaddr_in client_addr;
    struct sockaddr_in server_addr;
    int connect_sock = 0;
    int comm_sock = 0;
    int i = 0;
    BYTE recvBuf[MAX_BUF_SIZE] = { 0, };
    BYTE sendBuf[MAX_BUF_SIZE] = { 0, };
    BYTE K[16] =
    { 0x0f, 0x1e, 0x2d, 0x3c, 0x4b, 0x5a, 0x69, 0x78, 0x87, 0x96, 0xa5, 0xb4, 0xc3, 0xd2, 0xe1, 0xf0 };
    BYTE P[16] = { 0 };
    WORD RoundKey[144] = { 0 };

    KeySchedule_128(K, RoundKey);

    comm_sock = set_socket();
```

```c
    while (1)
    {
        memset(recvBuf, 0x00, MAX_BUF_SIZE);

        if (read(comm_sock, recvBuf, MAX_BUF_SIZE) <= 0)
        {
            printf("read error : ₩n");
            close(comm_sock);
        }

        printf("Encrypt data : ");
        for (i = 0; i < 16; i++)
        {
            printf("%02x ", recvBuf[i]);
        }
        printf("₩n");

        memset(P, 0x00, 16);
        Decrypt(24, RoundKey, P, recvBuf);

        printf("Decrypt data : ");
        for (i = 0; i < 16; i++)
        {
            printf("%02x ", P[i]);
        }
        printf("₩n₩n");

    }

    close(comm_sock);
    close(connect_sock);

    return 0;

}
```

소켓 프로그래밍 + LEA + 시리얼 통신 | serial_server.c

serial_server.c

```
int set_socket()
{
    struct sockaddr_in client_addr;
    struct sockaddr_in server_addr;

    int connect_sock = 0;
    int comm_sock = 0;
    int client_addr_len = 0;
    int ret = 0;

    client_addr_len = sizeof(client_addr);

    connect_sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (connect_sock == -1)
    {
        printf("SOCKET CREATE ERROR!!!₩n");
        return 1;
    }

    memset(&server_addr, 0x00, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    server_addr.sin_port = htons(9000);
```

```
    ret = bind(connect_sock, (struct sockaddr *)&server_addr, sizeof(server_addr));

    listen(connect_sock, 5);

    memset(&client_addr, 0x00, sizeof(client_addr));
    comm_sock = accept(connect_sock, (struct sockaddr *)&client_addr, &client_addr_len);
    if (comm_sock == -1)
    {
        printf("SOCKET CREATE ERROR!₩n");
        return 1;
    }
    printf("New Client : %s₩n₩n", inet_ntoa(client_addr.sin_addr));

    return comm_sock;
}
```

소켓 프로그래밍 + LEA + 시리얼 통신 | serial_client.c

serial_client.c

```c
#include <lea.h>
#include <unistd.h>
#include <time.h>
#include <fcntl.h>
#include <sys/poll.h>
#include <termios.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>

#define MAX_BUF_SIZE    1024
#define DEVPORT "/dev/ttyAMA0"

int set_uart(char* device_name, int baudrate);
void dust_sensor(unsigned char *command_buf, int serial_fd);
int data_read(int serial_fd, unsigned char* sd_buf);
int set_socket();

int main()
{
    struct sockaddr_in server_addr;
    int serial_fd = 0;
    int i = 0, len = 0;
    int comm_sock = 0;
    int server_addr_len = 0;
    BYTE buf[128] = { 0x00, };
    BYTE recvBuf[MAX_BUF_SIZE] = { 0, };
    BYTE sendBuf[MAX_BUF_SIZE] = { 0, };
    BYTE K[16] =
    { 0x0f, 0x1e, 0x2d, 0x3c, 0x4b, 0x5a, 0x69, 0x78, 0x87, 0x96, 0xa5, 0xb4, 0xc3, 0xd2, 0xe1, 0xf0 };
    WORD RoundKey[144] = { 0 };
```

소켓 프로그래밍 + LEA + 시리얼 통신 | serial_client.c

serial_client.c

```
    KeySchedule_128(K, RoundKey);

    //1. uart setting function
    serial_fd = set_uart(DEVPORT, B9600);

    //2. socket setting function
    comm_sock = set_socket();

    //3. dust sensor setting
    dust_sensor(buf, serial_fd);

    while (1)
    {
        //4. serial data (dust data) read funciton
        len = data_read(serial_fd, buf);
        printf("read len : %d₩n", len); //읽은 바이트 수 출력

        /*먼지데이터 출력*/
        printf("DUST DATA : ");
        for (i = 0; i < 16; i++)
        {
            printf("%02x ", buf[i]); //버퍼값 출력
        }
        printf("₩n");
```

```
        //4.5 encryption
        memset(sendBuf, 0X00, 16);
        Encrypt(24, RoundKey, buf, sendBuf);
        printf("Encrypt data : ");
        for (i = 0; i < 16; i++)
        {
            printf("%02x ", sendBuf[i]);
        }
        printf("₩n₩n");


        //5. send data (server)
        if (write(comm_sock, sendBuf, 16) <= 0)
        {
            printf("write error₩n");
            return 1;
        }

    }
    close(comm_sock);

    return 0;
}
```

소켓 프로그래밍 + LEA + 시리얼 통신 | serial_client.c

serial_client.c

```c
int set_uart(char* device_name, int baudrate)
{
    struct termios newtio;
    int serial_fd;

    memset(&newtio, 0, sizeof(newtio));
    serial_fd = open((char *)device_name, O_RDWR | O_NOCTTY);

    printf("serial_fd : %d₩n", serial_fd);

    if (serial_fd < 0)
    {
        printf("serial fd open fail !!!₩n");
        return -1;
    }

    newtio.c_cflag = baudrate;
    newtio.c_cflag |= CS8;
    newtio.c_cflag |= CLOCAL;
    newtio.c_cflag |= CREAD;
    newtio.c_iflag = IGNPAR;
    newtio.c_oflag = 0;
    newtio.c_lflag = 0;
    newtio.c_cc[VTIME] = 3;
    newtio.c_cc[VMIN] = 1;

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

    return serial_fd;
}
```

```c
void dust_sensor(unsigned char *command_buf, int serial_fd)
{
    command_buf[0] = 0x4E;
    command_buf[1] = 0x49;
    command_buf[2] = 0x55;
    command_buf[3] = 0x52;
    command_buf[4] = (command_buf[0] + command_buf[1] + command_buf[2] + command_buf[3]) & 0xff;
    command_buf[5] = 0x44;
    command_buf[6] = 0x53;

    write(serial_fd, command_buf, 7);

    sleep(1);
    command_buf[0] = 0x4E;
    command_buf[1] = 0x49;
    command_buf[2] = 0x55;
    command_buf[3] = 0x43;
    command_buf[4] = (command_buf[0] + command_buf[1] + command_buf[2] + command_buf[3]) & 0xff;
    command_buf[5] = 0x44;
    command_buf[6] = 0x53;

    write(serial_fd, command_buf, 7);

    printf("dust sensor setting done.₩n₩n");
}
```

소켓 프로그래밍 + LEA + 시리얼 통신 | serial_client.c

serial_client.c

```c
int data_read(int serial_fd, unsigned char* sd_buf)
{
    unsigned char buf[128] = { 0x00, };
    int i = 0, len = 0;

    while (1)
    {
        len = read(serial_fd, &buf[i], 1);

        if (buf[0] == 0x4E && buf[1] == 0x49)
        {
            if (buf[9] == 0x44 && buf[10] == 0x53)
            {
                memcpy(sd_buf, buf, 11);
                return i+1;
            }
        }
        i++;
    }
    return 0;
}
```

```c
int set_socket()
{
    struct sockaddr_in server_addr;
    int comm_sock = 0;
    int server_addr_len = 0;

    comm_sock = socket(PF_INET, SOCK_STREAM, 0);
    if (comm_sock == -1)
    {
        printf("error :₩n");
        return 1;
    }

    memset(&server_addr, 0x00, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = inet_addr("192.168.0.10");
    server_addr.sin_port = htons(9000);
    server_addr_len = sizeof(server_addr);

    if (connect(comm_sock, (struct sockaddr *)&server_addr, server_addr_len) == -1)
    {
        printf("connect error :₩n");
        return 1;
    }

    return comm_sock;
}
```