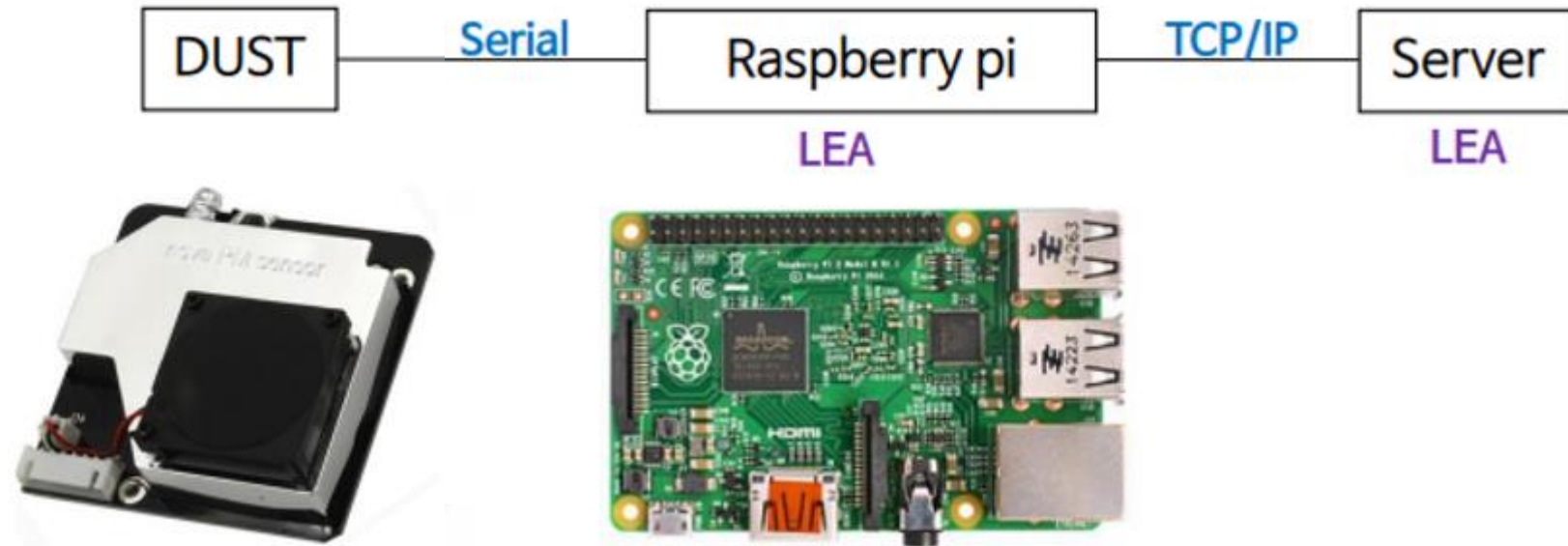


2018.12. ~ 2019.02.

스터디 총 정리
IoT 환경에서의 미세먼지 데이터
통신 보안 설계

정보보안암호수학과 20182209 김수빈
Github : https://github.com/K-subin/Socket_LEA_Serial

최종 목표는?



목차

1. 사용한 H/W
2. 공부한 통신
3. 통신을 하기 위해 추가적으로 필요한 S/W
4. LEA 암호 알고리즘
5. 라즈베리파이 개발환경 구축 과정
6. 소켓 프로그래밍
7. serial 통신하기

사용한 H/W



1. 라즈베리 파이
2. SDS011

라즈베리 파이

라즈베리 파이란?

- 초소형 컴퓨터로 RAM, CPU가 장착되어 있고, USB포트와 HDMI입력, 랜선포트 등이 있다
- 리눅스 기반의 라즈비안 OS를 SD카드에 저장하여 운영체제를 구동할 수 있다



실행 방법

OS를 SD카드에 설치한 후, SD카드를 라즈베리 파이에 삽입하고 teraterm을 통해 실행한다

SDS011

SDS011이란?

- 레이저 산란의 원리를 사용하여 공기 중 $0.3 \sim 10\mu\text{m}$ 사이의 입자 농도를 얻을 수 있게 하는 기기이다
- PM2.5(미세먼지), PM10(부유먼지)를 측정 가능하다
- 4.7~5.3V의 전력이 필요하다
- UART 통신을 하기 위한 조건
< Baudrate : 9600 / Databit : 8 / Paritybit : NO / Stopbit : 1 / DataPacketfrequency : 1Hz >



공부한 통신



1. TCP/IP 통신
2. Serial 통신 & UART 통신



CONTENT



CONTENT

TCP/IP

TCP란?

- 연결형 서비스를 지원하는 전송계층 프로토콜
- 호스트간 신뢰성 있는 데이터 전달(ACK)과 흐름제어 및 혼잡제어를 제공하는 프로토콜

IP란?

- 컴퓨터 네트워크에서 장치들이 서로를 인식하고 통신을 하기 위해서 사용하는 4바이트로 이루어진 특수한 번호(주소)

TCP/IP

- IP는 패킷 전달 여부를 보증하지 않고, 패킷을 보낸 순서와 받는 순서가 다를 수 있기 때문에 신뢰성 있는 데이터 전달을 위해서는 TCP와 함께 동작한다
- TCP는 올바른 통신을 하도록 하고 IP는 TCP 패킷을 전송하는 역할을 한다

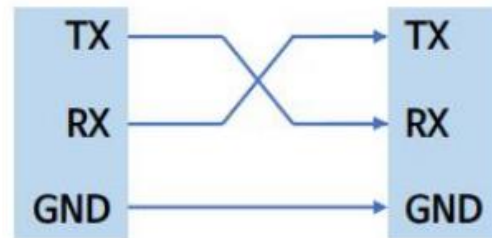
Serial 통신 & UART 통신

Serial 통신이란?

- 직렬 방식으로 데이터를 주고 받는 모든 통신으로 UART통신을 예로 들 수 있음

UART 통신이란?

- 비동기식 시리얼 통신 (→ 비동기식 : 송수신 양측이 시간 간격(Timing)을 맞출 수 있는 동기신호(Clock)를 가지고 있지 않는 통신 방식)
- 하드웨어와의 연결
 1. RX : 데이터 수신
 2. TX : 데이터 송신
 3. RX \leftrightarrow TX 교차연결
 4. GND 연결
 5. baud rate 설정(1초에 몇 개의 신호가 전송되는가)



▼ UART 통신을 위한 USB 포트



통신을 하기 위해 추가적으로 필요한 S/W

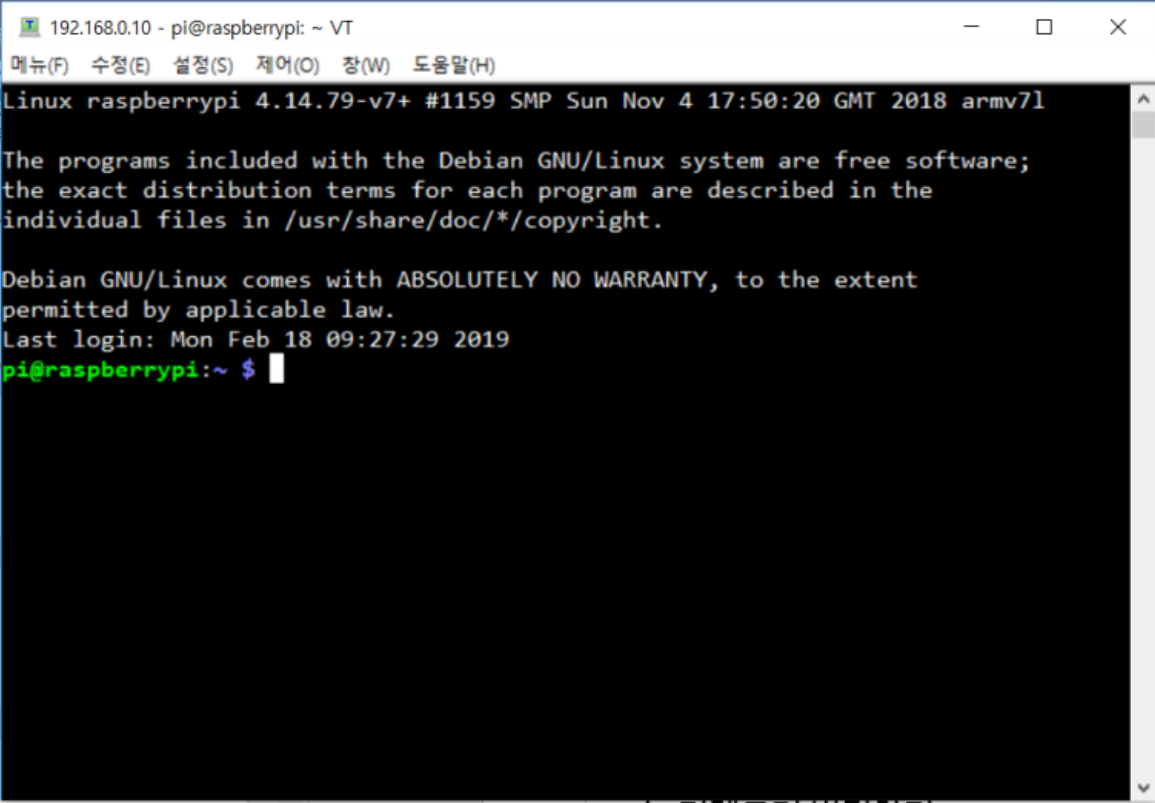


1. Teraterm
2. Filezilla
3. Hterm

Teraterm

Teraterm이란?

- 시리얼(Serial)통신과 텔넷(Telnet), SSH 통신의 모니터링을 위한 터미널 프로그램



```
192.168.0.10 - pi@raspberrypi: ~ VT
메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)
Linux raspberrypi 4.14.79-v7+ #1159 SMP Sun Nov 4 17:50:20 GMT 2018 armv7l

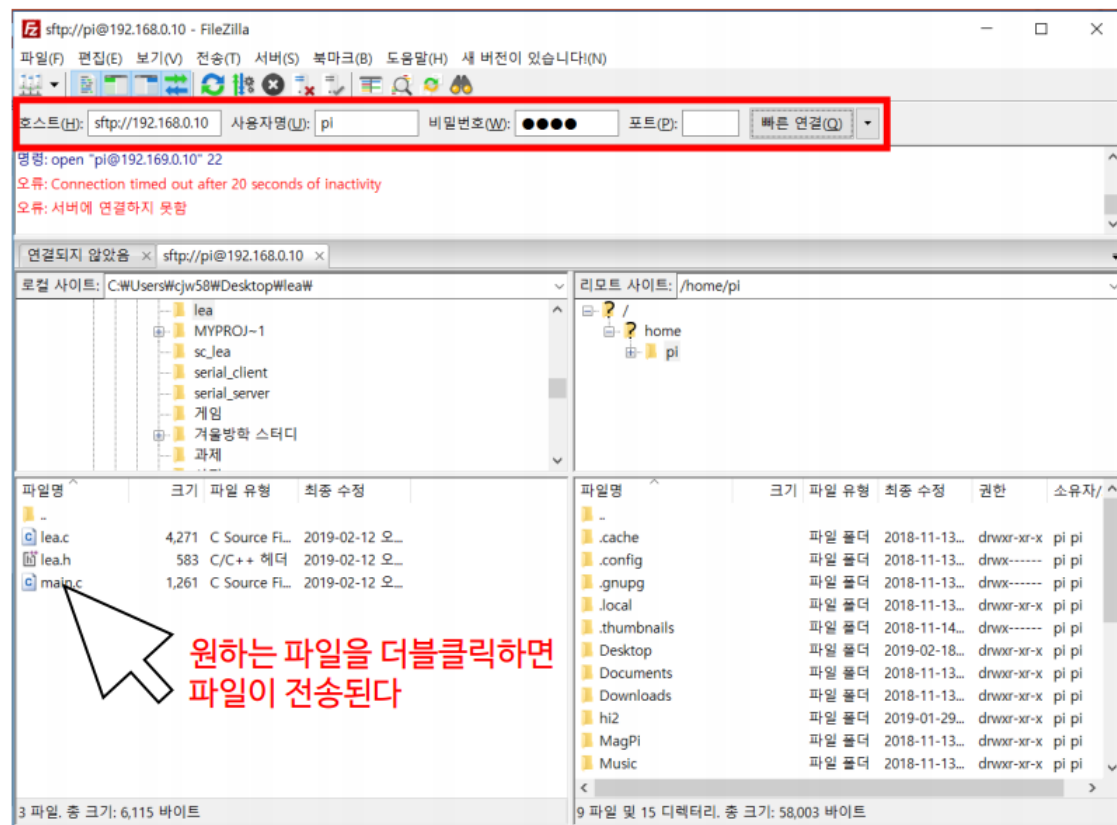
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Feb 18 09:27:29 2019
pi@raspberrypi:~ $
```

Filezila

Filezila란?

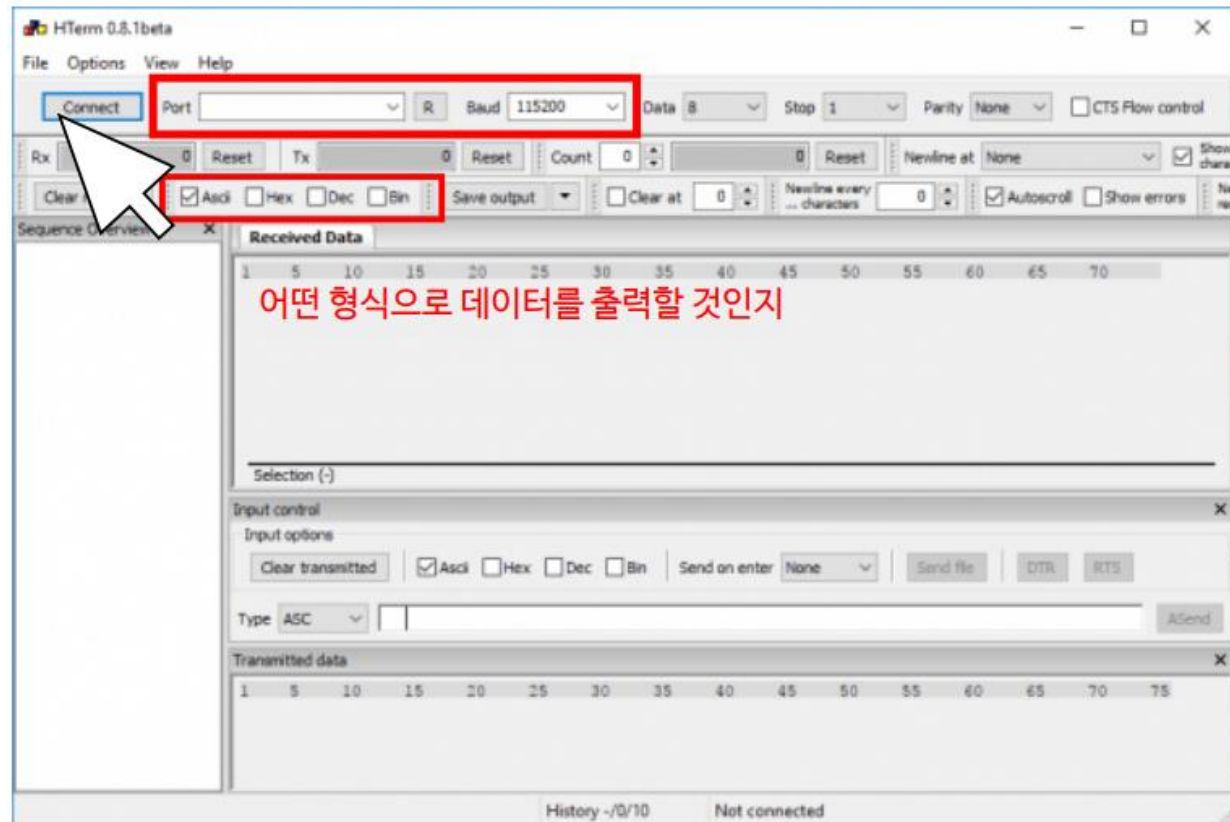
- 인터넷을 통해 한 컴퓨터에서 다른 컴퓨터로 파일을 전송할 수 있게 해주는 프로그램인 FTP프로토콜



Hterm

Hterm이란?

- 통신포트로부터 들어오는 데이터를 읽는 프로그램



LEA 암호 알고리즘



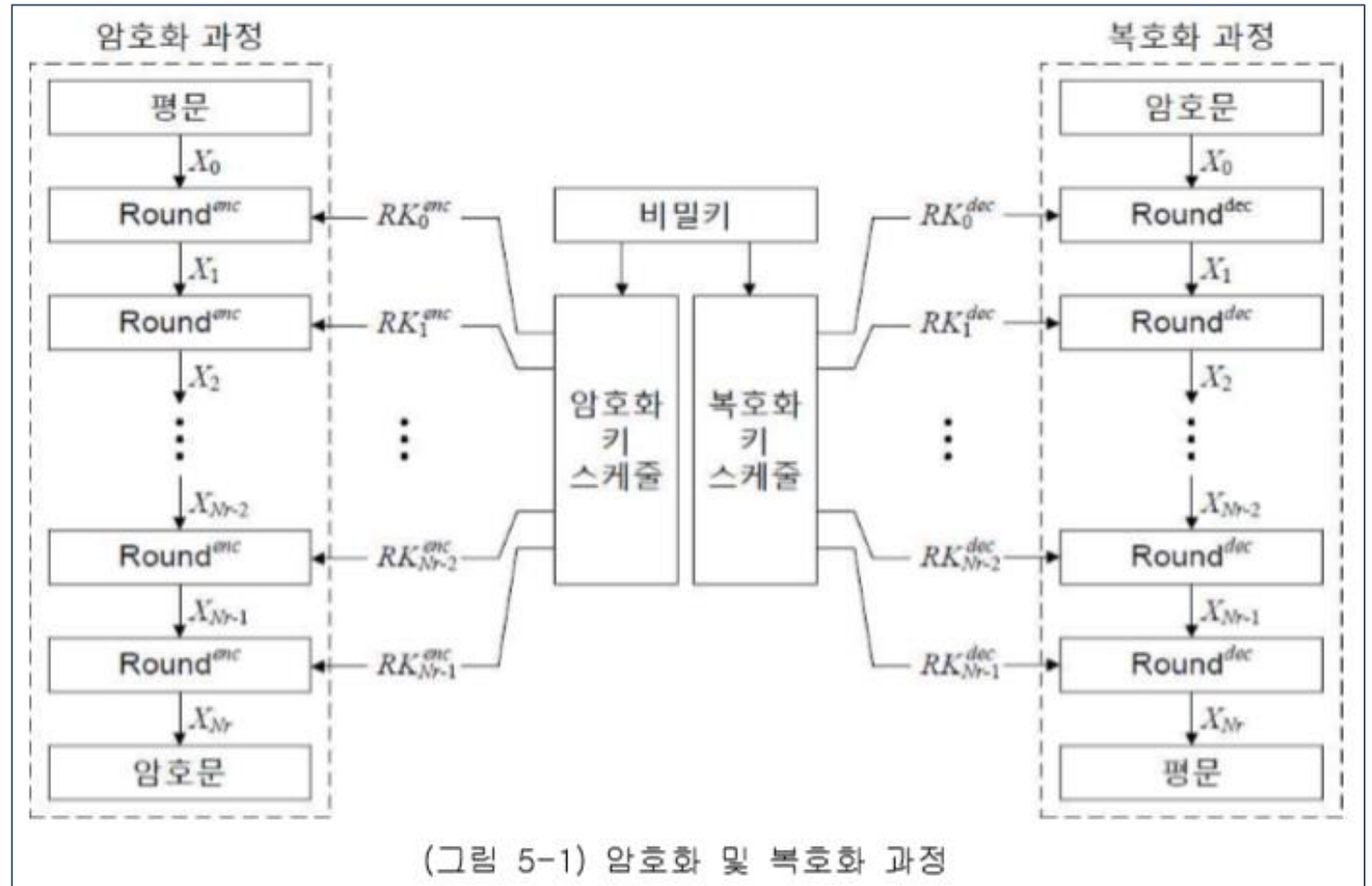
1. LEA 전체 알고리즘
2. LEA 키 스케줄링 LEA.c
3. LEA 암호화 LEA.c
4. LEA 복호화 LEA.c
5. LEA main.c

LEA 전체 알고리즘

LEA 전체 알고리즘

<표 5-1> LEA 규격

구분	Nb	Nk	Nr
LEA-128	16	16	24
LEA-192	16	24	28
LEA-256	16	32	32



LEA 키 스케줄링 LEA.c

LEA 키 스케줄링 알고리즘

$\delta[0] = \text{c3efe9db},$	$\delta[1] = \text{44626b02},$
$\delta[2] = \text{79e27c8a},$	$\delta[3] = \text{78df30ec},$
$\delta[4] = \text{715ea49e},$	$\delta[5] = \text{c785da0a},$
$\delta[6] = \text{e04ef22a},$	$\delta[7] = \text{e5c40957}.$

알고리즘 4 LEA-192 암호화 키 스케줄 함수: $(RK_0^{\text{enc}}, \dots, RK_{27}^{\text{enc}}) \leftarrow \text{KeySchedule}_{192}^{\text{enc}}(K)$

입력: 192비트 비밀키 K

출력: 28개의 192비트 암호화 라운드키 RK_i^{enc} ($0 \leq i \leq 27$)

```
1: T ← K
2: for i = 0 to 27 do
3:   T[0] ← ROL1(T[0] ⊕ ROLi(δ[i mod 6]))
4:   T[1] ← ROL3(T[1] ⊕ ROLi+1(δ[i mod 6]))
5:   T[2] ← ROL6(T[2] ⊕ ROLi+2(δ[i mod 6]))
6:   T[3] ← ROL11(T[3] ⊕ ROLi+3(δ[i mod 6]))
7:   T[4] ← ROL13(T[4] ⊕ ROLi+4(δ[i mod 6]))
8:   T[5] ← ROL17(T[5] ⊕ ROLi+5(δ[i mod 6]))
9:   RKienc ← (T[0], T[1], T[2], T[3], T[4], T[5])
10: end for
```

알고리즘 3 LEA-128 암호화 키 스케줄 함수: $(RK_0^{\text{enc}}, \dots, RK_{23}^{\text{enc}}) \leftarrow \text{KeySchedule}_{128}^{\text{enc}}(K)$

입력: 128비트 비밀키 K

출력: 24개의 192비트 암호화 라운드키 RK_i^{enc} ($0 \leq i \leq 23$)

```
1: T ← K
2: for i = 0 to 23 do
3:   T[0] ← ROL1(T[0] ⊕ ROLi(δ[i mod 4]))
4:   T[1] ← ROL3(T[1] ⊕ ROLi+1(δ[i mod 4]))
5:   T[2] ← ROL6(T[2] ⊕ ROLi+2(δ[i mod 4]))
6:   T[3] ← ROL11(T[3] ⊕ ROLi+3(δ[i mod 4]))
7:   RKienc ← (T[0], T[1], T[2], T[1], T[3], T[1])
8: end for
```

알고리즘 5 LEA-256 암호화 키 스케줄 함수: $(RK_0^{\text{enc}}, \dots, RK_{31}^{\text{enc}}) \leftarrow \text{KeySchedule}_{256}^{\text{enc}}(K)$

입력: 256비트 비밀키 K

출력: 32개의 192비트 암호화 라운드키 RK_i^{enc} ($0 \leq i \leq 31$)

```
1: T ← K
2: for i = 0 to 31 do
3:   T[6i mod 8] ← ROL1(T[6i mod 8] ⊕ ROLi(δ[i mod 8]))
4:   T[(6i+1) mod 8] ← ROL3(T[(6i+1) mod 8] ⊕ ROLi+1(δ[i mod 8]))
5:   T[(6i+2) mod 8] ← ROL6(T[(6i+2) mod 8] ⊕ ROLi+2(δ[i mod 8]))
6:   T[(6i+3) mod 8] ← ROL11(T[(6i+3) mod 8] ⊕ ROLi+3(δ[i mod 8]))
7:   T[(6i+4) mod 8] ← ROL13(T[(6i+4) mod 8] ⊕ ROLi+4(δ[i mod 8]))
8:   T[(6i+5) mod 8] ← ROL17(T[(6i+5) mod 8] ⊕ ROLi+5(δ[i mod 8]))
9:   RKienc ← (T[6i mod 8], T[(6i+1) mod 8], T[(6i+2) mod 8], T[(6i+3) mod 8],
              T[(6i+4) mod 8], T[(6i+5) mod 8])
10: end for
```


LEA 키 스케줄링 LEA.c

<LEA.c>

```
#include "lea.h"

WORD delta[8] = {
    0xc3efe9db,
    0x44626b02,
    0x79e27c8a,
    0x78df30ec,
    0x715ea49e,
    0xc785da0a,
    0xe04ef22a,
    0xe5c40957
};

//32비트 비트열 x의 i비트 좌측 순환이동
int ROL(int i, WORD value)
{
    return (value << i) | (value >> (32 - i));
}

//32비트 비트열 x의 i비트 우측 순환이동
int ROR(int i, WORD value)
{
    return (value >> i) | (value << (32 - i));
}
```

```
void KeySchedule_128(BYTE *K, WORD *RK)
{
    WORD T[4];

    memcpy(T, K, 16); //8*16 = 128

    int i;
    for (i = 0; i < 24; i++)
    {
        T[0] = ROL(1, T[0] + ROL(i, delta[i % 4]));
        T[1] = ROL(3, T[1] + ROL(i + 1, delta[i % 4]));
        T[2] = ROL(6, T[2] + ROL(i + 2, delta[i % 4]));
        T[3] = ROL(11, T[3] + ROL(i + 3, delta[i % 4]));
        RK[i * 6 + 0] = T[0];
        RK[i * 6 + 1] = T[1];
        RK[i * 6 + 2] = T[2];
        RK[i * 6 + 3] = T[3];
        RK[i * 6 + 4] = T[0];
        RK[i * 6 + 5] = T[1];
    }
}
```

LEA 키 스케줄링 LEA.c

<LEA.c>

```
void KeySchedule_192(BYTE *K, WORD *RK)
{
    WORD T[6];

    memcpy(T, K, 24); //8*24 = 192

    int i;
    for (i = 0; i < 28; i++)
    {
        T[0] = ROL(1, T[0] + ROL(i, delta[i % 6]));
        T[1] = ROL(3, T[1] + ROL(i + 1, delta[i % 6]));
        T[2] = ROL(6, T[2] + ROL(i + 2, delta[i % 6]));
        T[3] = ROL(11, T[3] + ROL(i + 3, delta[i % 6]));
        T[4] = ROL(13, T[4] + ROL(i + 4, delta[i % 6]));
        T[5] = ROL(17, T[5] + ROL(i + 5, delta[i % 6]));
        RK[i * 6 + 0] = T[0];
        RK[i * 6 + 1] = T[1];
        RK[i * 6 + 2] = T[2];
        RK[i * 6 + 3] = T[3];
        RK[i * 6 + 4] = T[4];
        RK[i * 6 + 5] = T[5];
    }
}
```

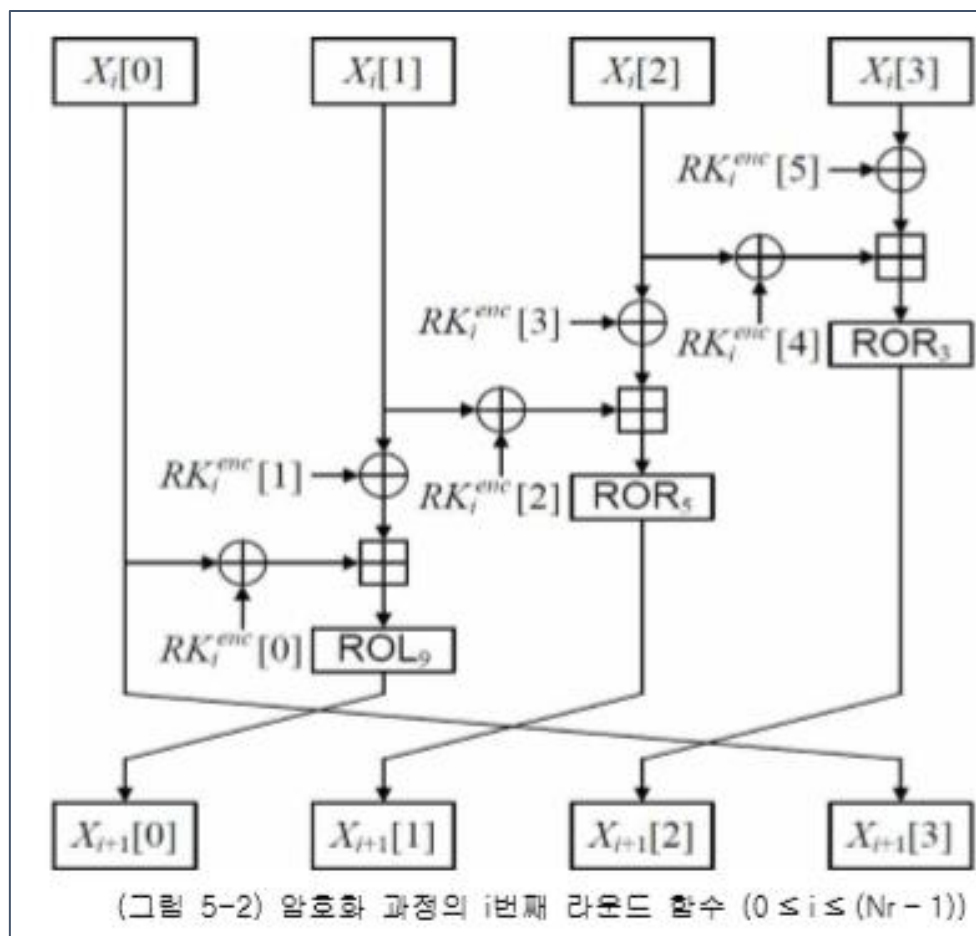
```
void KeySchedule_256(BYTE *K, WORD *RK)
{
    WORD T[8];

    memcpy(T, K, 32); //8*32 = 256

    int i;
    for (i = 0; i < 32; i++)
    {
        T[(6 * i + 0) % 8] = ROL(1, T[(6 * i + 0) % 8] + ROL(i, delta[i % 8]));
        T[(6 * i + 1) % 8] = ROL(3, T[(6 * i + 1) % 8] + ROL(i + 1, delta[i % 8]));
        T[(6 * i + 2) % 8] = ROL(6, T[(6 * i + 2) % 8] + ROL(i + 2, delta[i % 8]));
        T[(6 * i + 3) % 8] = ROL(11, T[(6 * i + 3) % 8] + ROL(i + 3, delta[i % 8]));
        T[(6 * i + 4) % 8] = ROL(13, T[(6 * i + 4) % 8] + ROL(i + 4, delta[i % 8]));
        T[(6 * i + 5) % 8] = ROL(17, T[(6 * i + 5) % 8] + ROL(i + 5, delta[i % 8]));
        RK[i * 6 + 0] = T[(i * 6 + 0) % 8];
        RK[i * 6 + 1] = T[(i * 6 + 1) % 8];
        RK[i * 6 + 2] = T[(i * 6 + 2) % 8];
        RK[i * 6 + 3] = T[(i * 6 + 3) % 8];
        RK[i * 6 + 4] = T[(i * 6 + 4) % 8];
        RK[i * 6 + 5] = T[(i * 6 + 5) % 8];
    }
}
```

LEA 암호화 LEA.c

LEA 암호화 알고리즘



알고리즘 1 암호화 함수: $C \leftarrow \text{Encrypt}(P, RK_0^{\text{enc}}, RK_1^{\text{enc}}, \dots, RK_{Nr-1}^{\text{enc}})$

입력: 128비트 평문 P , Nr 개의 192비트 라운드키 $RK_0^{\text{enc}}, RK_1^{\text{enc}}, \dots, RK_{Nr-1}^{\text{enc}}$

출력: 128비트 암호문 C

- 1: $X_0 \leftarrow P$
- 2: for $i = 0$ to $(Nr - 1)$ do
- 3: $X_{i+1} \leftarrow \text{Round}^{\text{enc}}(X_i, RK_i^{\text{enc}})$
- 4: end for
- 5: $C \leftarrow X_{Nr}$

알고리즘 2 암호화 과정의 i 번째 라운드 함수: $X_{i+1} \leftarrow \text{Round}^{\text{enc}}(X_i, RK_i^{\text{enc}})$

입력: 128비트 내부상태 변수 X_i , 192비트 라운드키 RK_i^{enc}

출력: 128비트 내부상태 변수 X_{i+1}

- 1: $X_{i+1}[0] \leftarrow \text{ROL}_9((X_i[0] \oplus RK_i^{\text{enc}}[0]) \boxplus (X_i[1] \oplus RK_i^{\text{enc}}[1]))$
- 2: $X_{i+1}[1] \leftarrow \text{ROR}_5((X_i[1] \oplus RK_i^{\text{enc}}[2]) \boxplus (X_i[2] \oplus RK_i^{\text{enc}}[3]))$
- 3: $X_{i+1}[2] \leftarrow \text{ROR}_3((X_i[2] \oplus RK_i^{\text{enc}}[4]) \boxplus (X_i[3] \oplus RK_i^{\text{enc}}[5]))$
- 4: $X_{i+1}[3] \leftarrow X_i[0]$

LEA 암호화 LEA.c

<LEA.c>

```
void Encrypt(int Nr, WORD *RK, BYTE *P, BYTE *C)
{
    WORD X_Round[4];
    WORD X_NextRound[4];

    memcpy(X_Round, P, 16);

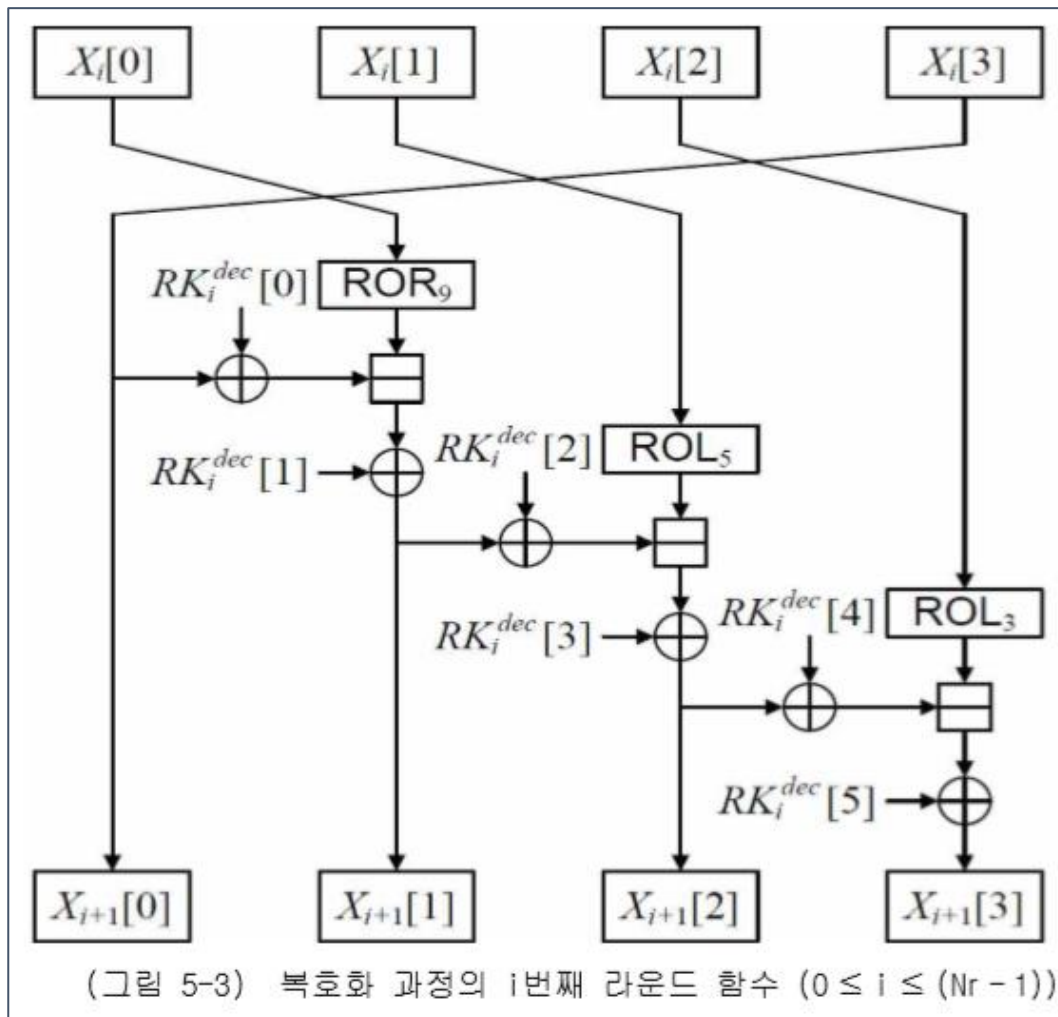
    int i;
    for (i = 0; i < Nr; i++)
    {
        X_NextRound[0] = ROL(9, (X_Round[0] ^ RK[i * 6 + 0]) + (X_Round[1] ^ RK[i * 6 + 1]));
        X_NextRound[1] = ROR(5, (X_Round[1] ^ RK[i * 6 + 2]) + (X_Round[2] ^ RK[i * 6 + 3]));
        X_NextRound[2] = ROR(3, (X_Round[2] ^ RK[i * 6 + 4]) + (X_Round[3] ^ RK[i * 6 + 5]));
        X_NextRound[3] = X_Round[0];

        memcpy(X_Round, X_NextRound, 16);
    }

    memcpy(C, X_NextRound, 16);
}
```

LEA 복호화 LEA.c

LEA 복호화 알고리즘



알고리즘 6 복호화 함수: $P \leftarrow \text{Decrypt}(C, RK_0^{\text{dec}}, RK_1^{\text{dec}}, \dots, RK_{Nr-1}^{\text{dec}})$

입력: 128비트 암호문 C , Nr 개의 192비트 라운드키 $RK_0^{\text{dec}}, RK_1^{\text{dec}}, \dots, RK_{Nr-1}^{\text{dec}}$

출력: 128비트 평문 P

- 1: $X_0 \leftarrow C$
- 2: **for** $i = 0$ to $(Nr - 1)$ **do**
- 3: $X_{i+1} \leftarrow \text{Round}^{\text{dec}}(X_i, RK_i^{\text{dec}})$
- 4: **end for**
- 5: $P \leftarrow X_{Nr}$

알고리즘 7 복호화 과정의 i 번째 라운드 함수: $X_{i+1} \leftarrow \text{Round}^{\text{dec}}(X_i, RK_i^{\text{dec}})$

입력: 128비트 내부상태 변수 X_i , 192비트 라운드키 RK_i^{dec}

출력: 128비트 내부상태 변수 X_{i+1}

- 1: $X_{i+1}[0] \leftarrow X_i[3]$
- 2: $X_{i+1}[1] \leftarrow (\text{ROR}_9(X_i[0]) \oplus (X_{i+1}[0] \oplus RK_i^{\text{dec}}[0])) \oplus RK_i^{\text{dec}}[1]$
- 3: $X_{i+1}[2] \leftarrow (\text{ROL}_5(X_i[1]) \oplus (X_{i+1}[1] \oplus RK_i^{\text{dec}}[2])) \oplus RK_i^{\text{dec}}[3]$
- 4: $X_{i+1}[3] \leftarrow (\text{ROL}_3(X_i[2]) \oplus (X_{i+1}[2] \oplus RK_i^{\text{dec}}[4])) \oplus RK_i^{\text{dec}}[5]$

$$RK_i^{\text{dec}} = RK_{Nr-i-1}^{\text{enc}} \quad (0 \leq i \leq (Nr - 1))$$

LEA 복호화 LEA.c

<LEA.c>

```
void Decrypt(int Nr, WORD *RK, BYTE *D, BYTE *C)
{
    WORD X_Round[4];
    WORD X_NextRound[4];

    memcpy(X_Round, C, 16);

    int i;
    for (i = 0; i < Nr; i++)
    {
        X_NextRound[0] = X_Round[3];
        X_NextRound[1] = (ROR(9, X_Round[0]) - (X_NextRound[0] ^ RK[((Nr - i - 1) * 6) + 0])) ^ RK[((Nr - i - 1) * 6) + 1];
        X_NextRound[2] = (ROL(5, X_Round[1]) - (X_NextRound[1] ^ RK[((Nr - i - 1) * 6) + 2])) ^ RK[((Nr - i - 1) * 6) + 3];
        X_NextRound[3] = (ROL(3, X_Round[2]) - (X_NextRound[2] ^ RK[((Nr - i - 1) * 6) + 4])) ^ RK[((Nr - i - 1) * 6) + 5];

        memcpy(X_Round, X_NextRound, 16);
    }

    memcpy(D, X_Round, 16);
}
```

LEA main.c

<main.c>

```
#include "lea.h"

int main()
{
    int i;
    int Nk;
    int Nr;

    //K = 16, 24, 32 byte
    printf("LEA 키의 바이트 길이를 입력하세요.(16, 24, 32)\\n");
    scanf("%d", &Nk);

    BYTE K[16] =
    {
        0x0f, 0x1e, 0x2d, 0x3c, 0x4b, 0x5a, 0x69, 0x78, 0x87, 0x96, 0xa5, 0xb4, 0xc3, 0xd2, 0xe1, 0xf0
    };

    //RoundKey = 144, 168, 192 bit
    WORD RoundKey[144] = { 0, };
    BYTE P[16] = { 0 };
    BYTE C[16] = { 0 };
    BYTE D[16] = { 0 };
```

LEA main.c

<main.c>

```
if (Nk == 16)
{
    Nr = 24;
    KeySchedule_128(K, RoundKey);
}
if (Nk == 24)
{
    Nr = 28;
    KeySchedule_192(K, RoundKey);
}
if (Nk == 32)
{
    Nr = 32;
    KeySchedule_256(K, RoundKey);
}

printf("Nk = %d\n", Nk);
printf("Nr = %d\n\n", Nr);
printf("Key : ");
for (i = 0; i < Nk; i++)
{
    printf("0x%02x ", K[i]);
}
printf("\n\n");
```

```
WORD tmp = 0;
printf("Write plaintext : ");
for (i = 0; i < 16; i++)
{
    scanf("%x", &tmp);
    P[i] = tmp & 0xff;
}

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

Encrypt(Nr, RoundKey, P, C);
printf("\n");
```

```
printf("Ciphertext :");
for (i = 0; i < 16; i++)
{
    printf("0x%02x ", C[i]);
}
printf("\n\n");

Decrypt(Nr, RoundKey, D, C);
printf("\n");

printf("Plaintext : ");
for (i = 0; i < 16; i++)
{
    printf("0x%02x ", D[i]);
}

return 0;
}
```


라즈베리파이 개발환경 구축 과정



1. 라즈비안 OS 다운
2. SSH 설정
3. Wi-Fi 설정
4. 라즈베리파이 연결하기
5. 비밀번호 변경하기
6. Wi-Fi 접속하기

라즈비안 OS 다운

라즈비안 OS란?

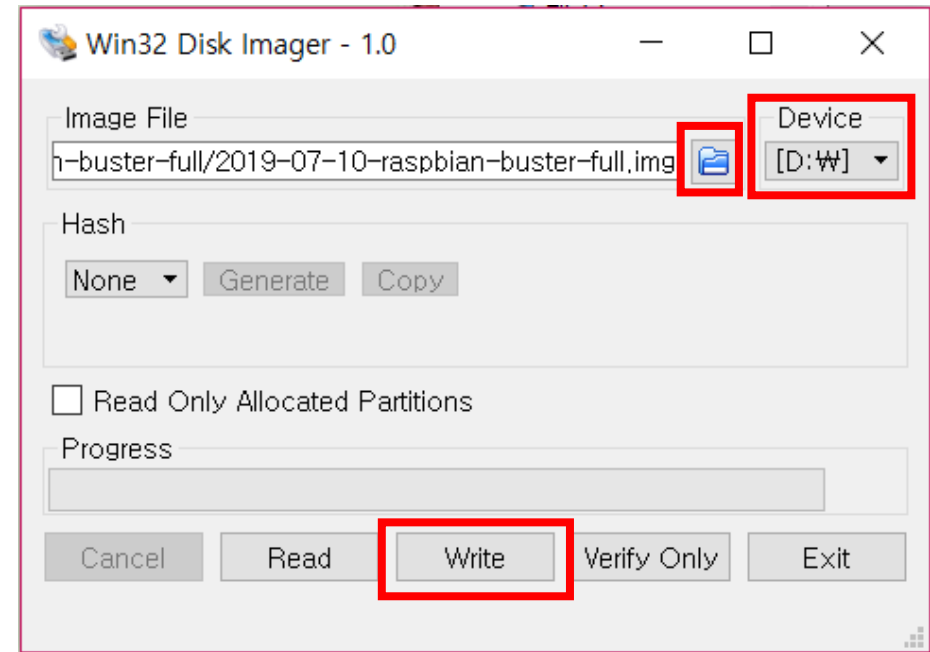
데미안 리눅스를 소형화하여 라즈베리 파이 전용으로 개발한 운영체제

라즈비안 이미지 다운로드

(<https://www.raspberrypi.org/>)

마이크로 SD카드에 라즈비안 이미지 쓰기

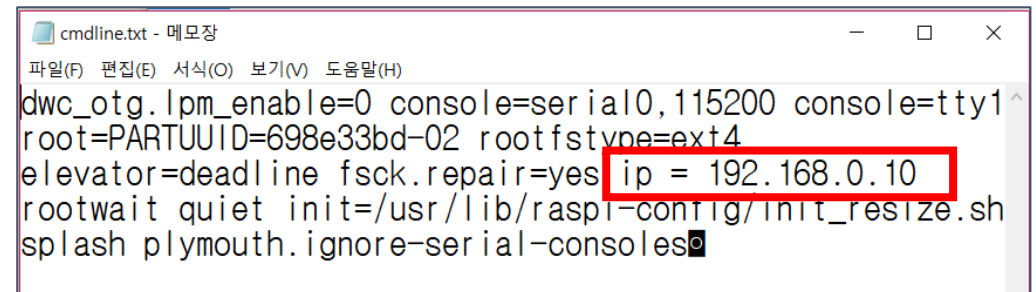
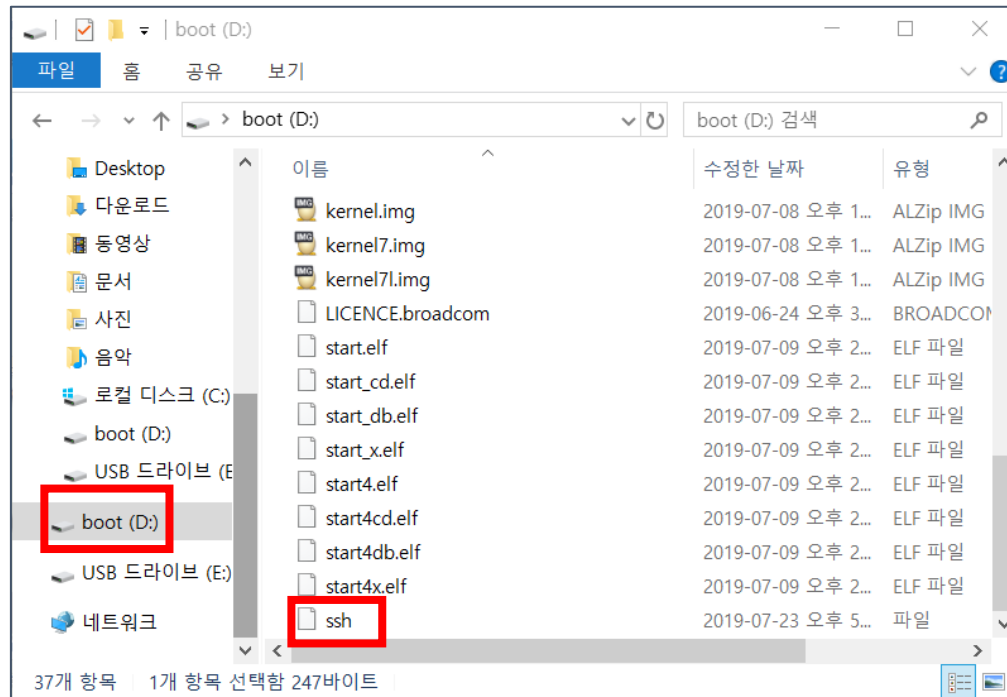
1. Win 32 Disk Imager 프로그램을 설치
2. 폴더 모양의 아이콘을 클릭해 .img 확장자로 되어있는 라즈비안 이미지 선택
3. 마이크로 SD카드가 삽입된 USB장치를 컴퓨터에 연결
-> "Device" 영역에 드라이브가 인식됨.
4. "Write"를 눌러 마이크로 SD카드에 라즈비안 이미지 쓰기



SSH 설정

SSH 설정

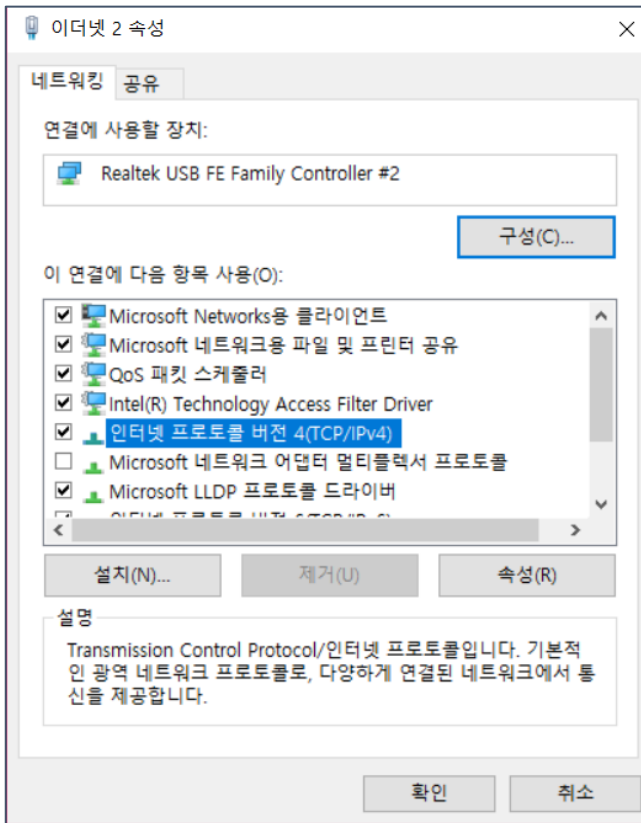
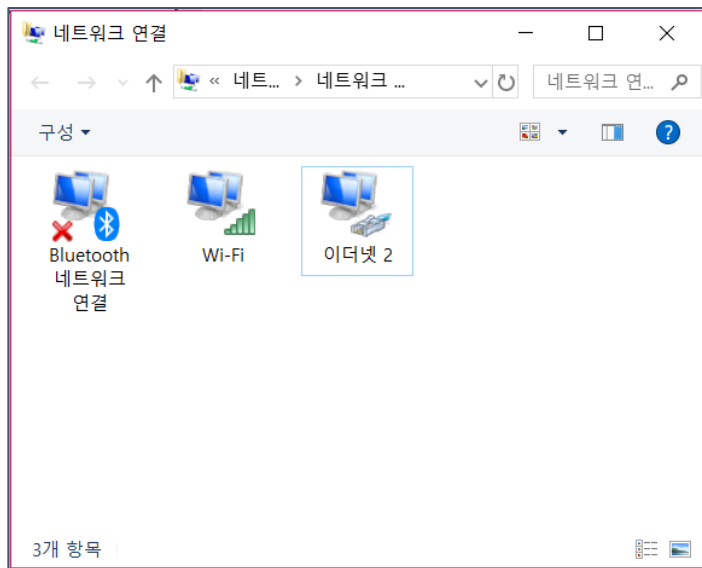
1. 마이크로 SD카드를 컴퓨터에 다시 연결하면 Boot 폴더가 나타남
2. Boot 폴더 안에 "ssh" 파일명의 확장자 없는 파일을 삽입
3. cmdline.txt 파일을 연 뒤, yes 뒤에 "ip = 192.168.0.10"을 넣고 Ctrl+S로 저장



Wi-Fi 설정

Wi-Fi 설정

1. 네트워크 설정 변경 → 어댑터 옵션 변경 → 이더넷 속성
2. 인터넷 프로토콜 버전4(TCP/IPv4) 더블클릭



3. ip 주소에 "192.168.0.100" 입력
4. 서브넷 마스크에 "255.255.255.0" 입력

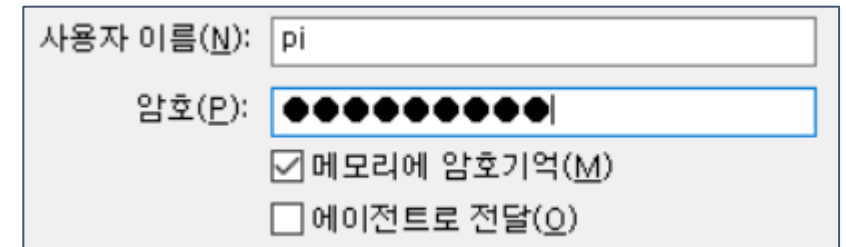
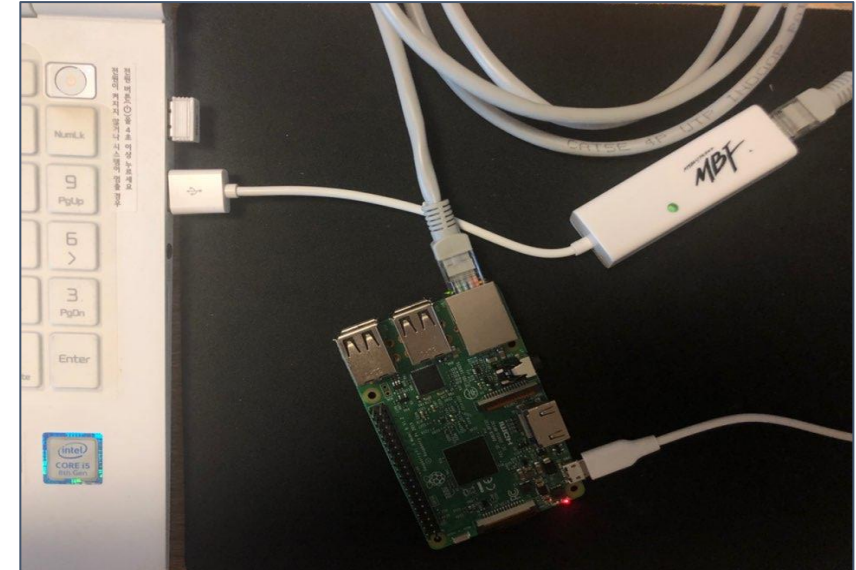
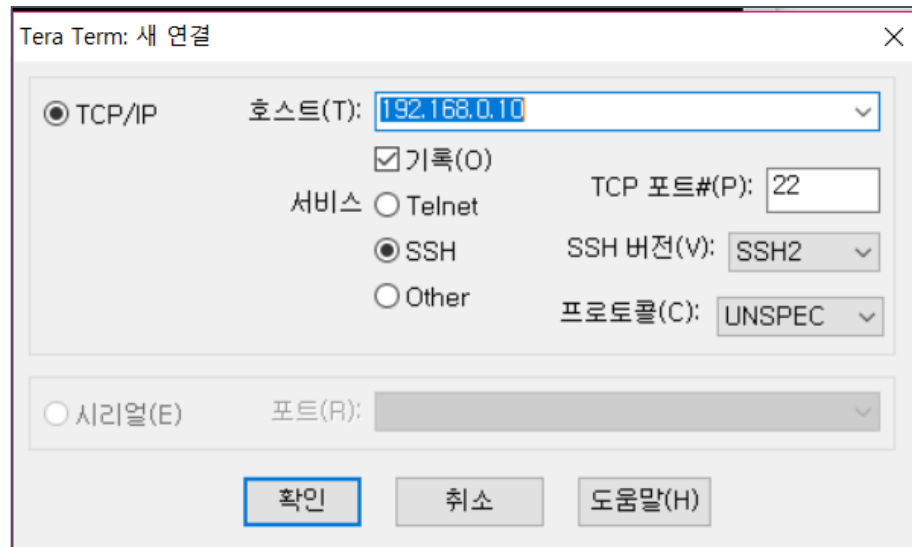
● 다음 IP 주소 사용(S):

IP 주소(I):	192 . 168 . 0 . 100
서브넷 마스크(U):	255 . 255 . 255 . 0
기본 게이트웨이(D):	.

라즈베리파이 연결하기

라즈베리파이 연결하기

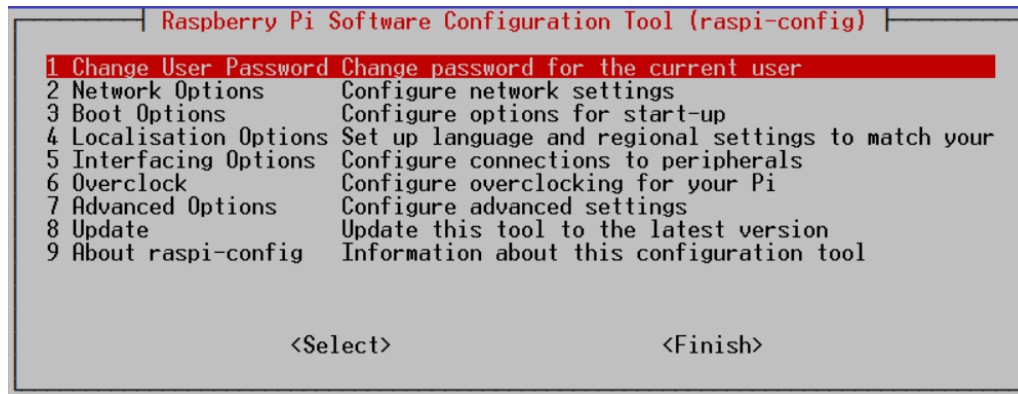
1. 설정을 마친 SD카드를 라즈베리파이와 연결
2. 라즈베리파이와 노트북을 랜선으로 연결
3. Tera Term 실행
4. 호스트에 "192.168.0.10" 입력
5. 사용자 이름에 "pi" 입력, 암호에 "raspberrry" 입력



비밀번호 변경하기

비밀번호 변경하기

1. 라즈베리 환경 설정하는 명령어 "sudo raspi-config" 입력
2. "Change User Password" 선택
3. 새로운 패스워드를 2번 입력



```
pi@raspberrypi:~$ sudo raspi-config
New password:
Retype new password: █
```

Wi-Fi 접속하기

Wi-Fi 접속하기

1. wpa_passphrase "SSID" wi-fi비밀번호 입력
2. sudo vim.tiny /etc/wpa_supplicant/wpa_supplicant.conf 입력
3. 1번 실행결과 복사(Alt + c) 후 붙여넣기(Alt + v) → :wq 저장하고 나가기 → 재부팅
4. Ifconfig 명령어를 통해 wifi가 제대로 설정 됐는지 확인

```
pi@raspberrypi:~$ sudo iwlist wlan0 scan | grep SK_WiFiGIGAF6FA
ESSID:"SK_WiFiGIGAF6FA"
pi@raspberrypi:~$ wpa_passphrase "SK_WiFiGIGAF6FA" 1603046749
network={
    ssid="SK_WiFiGIGAF6FA"
    #psk="1603046749"
    psk=fd1f9dee028f5bcb0b12877353dd43e8cf0ab72d46cbd323e8ce2861d8aee2c9
}
pi@raspberrypi:~$ sudo vim.tiny /etc/wpa_supplicant/wpa_supplicant.conf
```

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="SK_WiFiGIGAF6FA"
    #psk="1603046749"
    psk=fd1f9dee028f5bcb0b12877353dd43e8cf0ab72d46cbd323e8ce2861d8aee2c9
}
```

소켓 프로그래밍



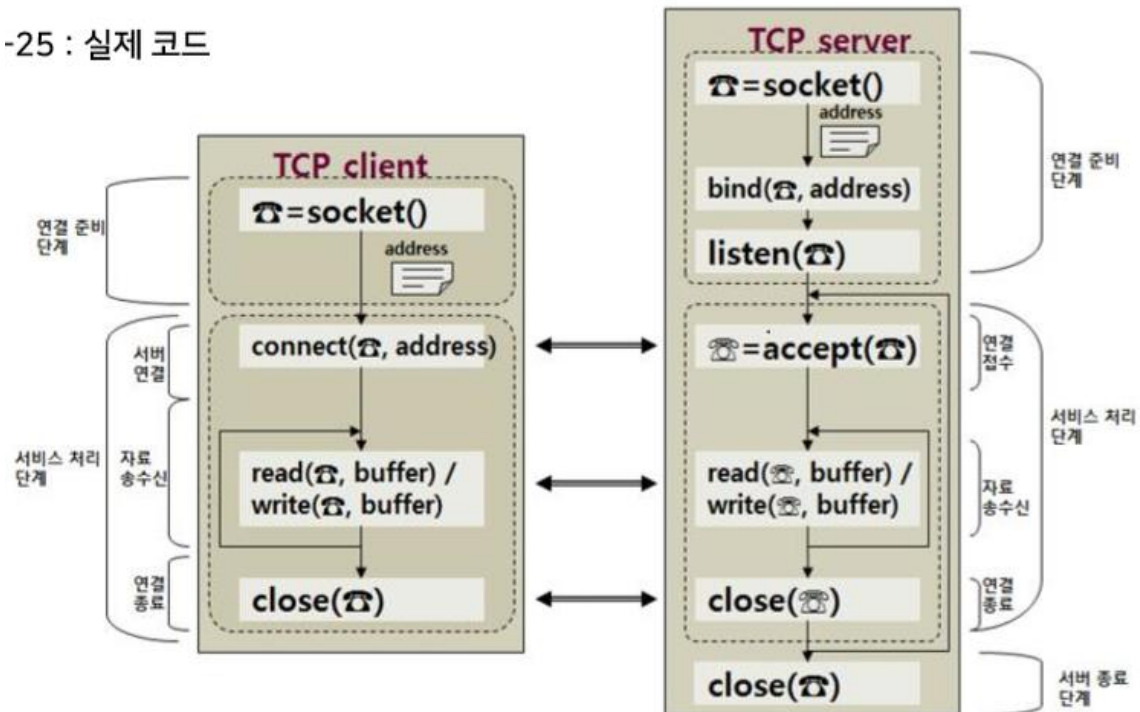
1. 소켓 프로그래밍
2. 소켓 프로그래밍 (server \leftrightarrow client)
3. 소켓 + LEA

소켓 프로그래밍

소켓 프로그래밍이란?

소켓 계층에서 제공하는 함수들을 이용한 프로그래밍

-25 : 실제 코드



소켓 프로그래밍 server ↔ client

<server_echo.c>

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

#define MAX_BUF_SIZE 1024

int main()
{
    struct sockaddr_in client_addr;
    struct sockaddr_in server_addr;
    int connect_sock = 0;
    int comm_sock = 0;
    int client_addr_len = 0;
    int n = 0;
    int ret = 0;
    unsigned char recvBuf[MAX_BUF_SIZE] = { 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);

while (1)
{
    memset(&client_addr, 0x00, sizeof(client_addr));

    comm_sock = accept(connect_sock, (struct sockaddr *)&client_addr, &client_addr_len);
    printf("New Client : %s\n", inet_ntoa(client_addr.sin_addr));

    memset(recvBuf, 0x00, MAX_BUF_SIZE);

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

    printf("receive message : %s\n", recvBuf);
    if (write(comm_sock, recvBuf, MAX_BUF_SIZE) <= 0)
    {
        printf("write error : \n");
        close(comm_sock);
    }

    close(comm_sock);
}
```

소켓 프로그래밍 server ↔ client

<client_echo.c>

```
#include <stdio.h> //STanDard Input Output
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <sys/socket.h> // "socket 함수 사용" , "inet_addr 함수 사용"
#include <netinet/in.h> // "inet_addr 함수 사용",
#include <sys/stat.h>
#include <arpa/inet.h> // "inet_addr 함수 사용"
#include <sys/types.h> // "socket 함수 사용"

#define MAX_BUF_SIZE 1024

int main()
{
    struct sockaddr_in server_addr;
    int comm_sock = 0;
    int server_addr_len = 0;
    unsigned char recvBuf[MAX_BUF_SIZE] = { 0, };
    unsigned char sendBuf[MAX_BUF_SIZE] = { 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;
}

memset(sendBuf, 0x00, MAX_BUF_SIZE);

printf("input message : ");
scanf("%[^\n]s", sendBuf);

if (write(comm_sock, sendBuf, MAX_BUF_SIZE) <= 0)
{
    printf("write error\n");
    return 1;
}

memset(recvBuf, 0x00, MAX_BUF_SIZE);

if (read(comm_sock, recvBuf, MAX_BUF_SIZE) <= 0)
{
    printf("read error\n");
    return 1;
}

printf("read : %s\n", recvBuf);

close(comm_sock);

return 0;
}
```

소켓 프로그래밍 + LEA server ↔ client

목표는?

1. 서버와 클라이언트를 연결
2. 클라이언트에서 평문 입력한 후 암호화하여 서버로 전달
3. 서버에서 암호문을 받아 복호화해 평문 읽음

server.c



client.c

```
192.168.0.10 - pi@raspberrypi: ~/Desktop/lea VT
Linux raspberrypi 4.19.57-v7+ #1244 SMP Thu Jul 4 18:45:25 BST 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.
Last login: Tue Aug 27 17:27:28 2019 from 192.168.0.100
pi@raspberrypi:~ $ cd Desktop
pi@raspberrypi:~/Desktop $ ls
Desktop lea serial
pi@raspberrypi:~/Desktop $ cd lea
pi@raspberrypi:~/Desktop/lea $ ls
client.c lea.c server.c socket_client socket_server
pi@raspberrypi:~/Desktop/lea $ ./socket_server
New Client : 192.168.0.10

Ciphertext : 0x9f 0xc8 0x4e 0x35 0x28 0xc6 0xc6 0x18 0x55 0x32 0xc7 0xa7 0x04 0x6
4 0x8b 0xfd

Plaintext : 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1a 0x1b 0x1c 0x1
d 0x1e 0x1f
pi@raspberrypi:~/Desktop/lea $
```

```
192.168.0.10 - pi@raspberrypi: ~/Desktop/lea VT
Linux raspberrypi 4.19.57-v7+ #1244 SMP Thu Jul 4 18:45:25 BST 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.
Last login: Tue Aug 27 17:28:00 2019 from 192.168.0.100
pi@raspberrypi:~ $ cd Desktop
pi@raspberrypi:~/Desktop $ ls
Desktop lea serial
pi@raspberrypi:~/Desktop $ cd lea
pi@raspberrypi:~/Desktop/lea $ ls
client.c lea.c server.c socket_client socket_server
pi@raspberrypi:~/Desktop/lea $ ./socket_client
Write Plaintext : 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f

Plaintext : 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1a 0x1b 0x1c 0x1
d 0x1e 0x1f

pi@raspberrypi:~/Desktop/lea $
```

소켓 프로그래밍 + LEA server \leftrightarrow client

< server.c = server_echo.c + LEA 복호화 >

```
#include <lea.h>
#include <unistd.h>
...
#include <sys/socket.h>
#include <sys/stat.h>
#include <arpa/inet.h>
#include <sys/types.h>

#define MAX_BUF_SIZE 1024

int main()
{
    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;
    int i, Nk, Nr;
    BYTE recvBuf[MAX_BUF_SIZE] = { 0, };
    WORD RoundKey[144] = { 0, };
    BYTE K[16] =
    { 0x0f, 0x1e, 0x2d, 0x3c, 0x4b, 0x5a, 0x69, 0x78, 0x87, 0x96, 0xa5, 0xb4, 0xc3, 0xd2, 0xe1, 0xf0 };
    BYTE P[16] = { 0 };
    Nk = 16;
    Nr = 24;
```

소켓 프로그래밍 + LEA server ↔ client

< server.c = server_echo.c + LEA 복호화 >

```
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);

printf("New Client : %s\n", inet_ntoa(client_addr.sin_addr));

KeySchedule_128(K, RoundKey);

memset(recvBuf, 0x00, MAX_BUF_SIZE);
if (read(comm_sock, recvBuf, MAX_BUF_SIZE) <= 0)
{
    printf("read error : \n");
    close(comm_sock);
}
```

```
printf("Ciphertext : ");
for (i = 0; i < 16; i++)
{
    printf("0x%02x ", recvBuf[i]);
}
printf("\n\n");

Decrypt(Nr, RoundKey, P, recvBuf);

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

close(comm_sock);
close(connect_sock);

return 0;
}
```

소켓 프로그래밍 + LEA server \leftrightarrow client

< client.c = client_echo.c + LEA 암호화 >

```
#include <le.h>
#include <unistd.h>
...
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/stat.h>
#include <arpa/inet.h>
#include <sys/types.h>

#define MAX_BUF_SIZE 1024

int main()
{
    struct sockaddr_in server_addr;
    int comm_sock = 0;
    int server_addr_len = 0;
    int i, Nk, Nr;
    BYTE recvBuf[MAX_BUF_SIZE] = { 0, };
    BYTE sendBuf[MAX_BUF_SIZE] = { 0, };
    WORD RoundKey[144] = { 0, };
    BYTE K[16] =
    { 0x0f, 0x1e, 0x2d, 0x3c, 0x4b, 0x5a, 0x69, 0x78, 0x87, 0x96, 0xa5, 0xb4, 0xc3, 0xd2, 0xe1, 0xf0 };
    BYTE P[16] = { 0 };
    Nk = 16;
    Nr = 24;

    /*통신 소켓 만들기*/
    comm_sock = socket(PF_INET, SOCK_STREAM, 0);
    if (comm_sock == -1)
    {
        printf("error : %n");
        return 1;
    }
}
```

소켓 프로그래밍 + LEA server \leftrightarrow client

< client.c = client_echo.c + LEA 암호화 >

```
/*server_addr 구조체 선언*/
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);

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

KeySchedule_128(K, RoundKey);

/*평문 입력*/
WORD tmp = 0;
printf("Write Plaintext : ");
for (i = 0; i < 16; i++)
{
    scanf("%x", &tmp);
    P[i] = tmp & 0xff;
}
printf("\n");
```

```
/*평문 출력*/
printf("Plaintext : ");
for (i = 0; i < 16; i++)
{
    printf("0x%02x ", P[i]);
}
printf("\n\n");

/*암호화*/
memset(sendBuf, 0x00, MAX_BUF_SIZE);
Encrypt(Nr, RoundKey, P, sendBuf);
printf("\n");

/*write*/
if (write(comm_sock, sendBuf, MAX_BUF_SIZE) <= 0)
{
    printf("write error\n");
    return 1;
}

/*소켓 종료*/
close(comm_sock);

return 0;
}
```


serial 통신하기



1. 먼지센서 연결 (SDS011 \leftrightarrow Hterm)
2. 시리얼 통신 환경 구축 (라즈베리파이 \leftrightarrow Teraterm)
3. 소켓 시리얼 프로그래밍
4. 소켓 + serial 통신 + LEA (먼지데이터 & server \leftrightarrow client)

먼지센서 연결 SDS011 ↔ Hterm

목표는?

- 데이터가 잘 들어오는지 확인하여 나중에 시리얼 통신 코딩했을 때 코드 이외의 부분에서 신경 쓰지 않도록 미리 확인

SDS011 ↔ Hterm 연결

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 연결 SDS011 ↔ Hterm

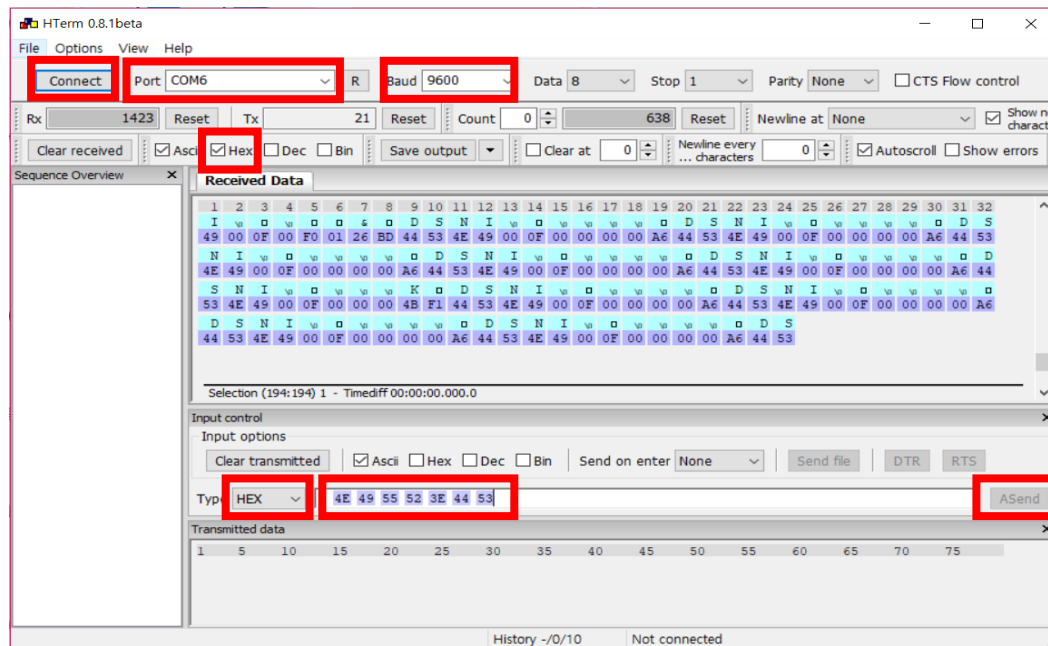
SDS011 ↔ Hterm 연결

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

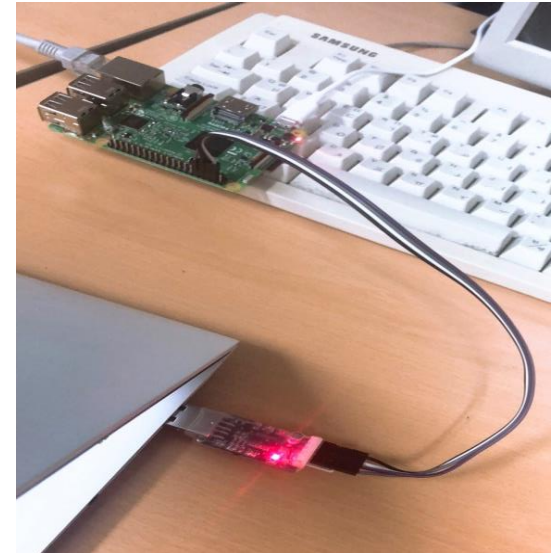
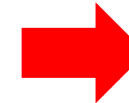
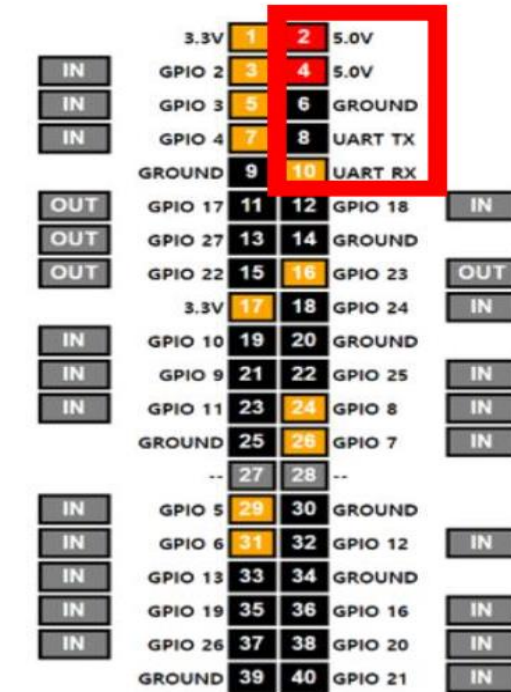
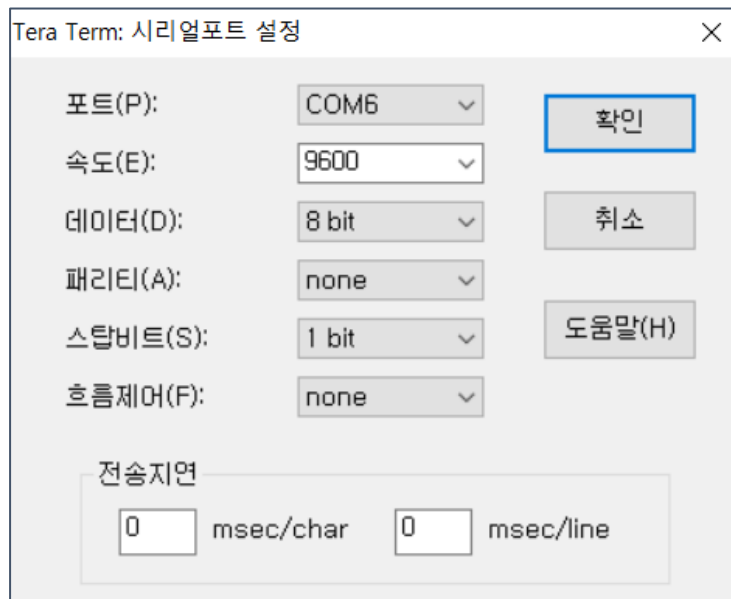
STX (2Bytes)		Command (2Bytes)		Checksum	ETX (2Bytes)	
1st	2nd	3rd	4th	5th	6th	7th
0x4E	0x49	0x55	0x43	(1st+2nd+3rd+4th) & 0xFF	0x44	0x53

→ 4E 49 55 43 2F 44 53

시리얼 통신 환경 구축 라즈베리파이 ↔ Teraterm

시리얼 통신 환경 구축

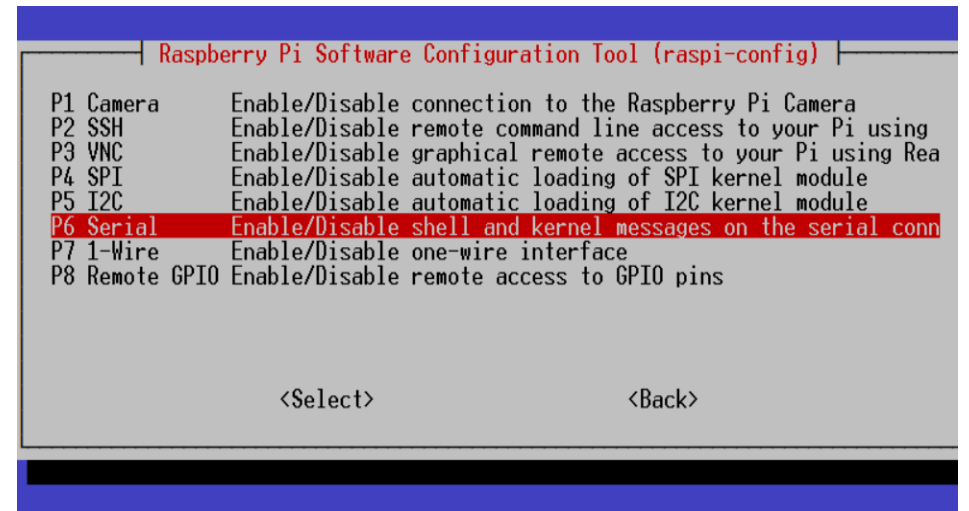
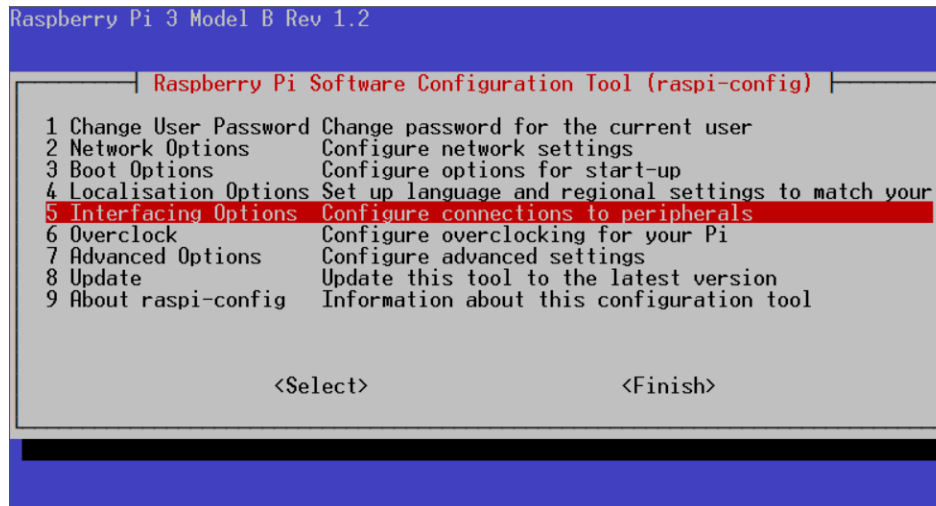
1. 노트북과 라즈베리파이를 연결한다.
2. Teraterm 실행
3. 설정 → 시리얼포트 설정
4. 포트 : COM6 / 속도 : 9600 설정



시리얼 통신 환경 구축 라즈베리파이 ↔ Teraterm

시리얼 통신 환경 구축

2. "sudo raspi-config" 입력
3. 5번 interfacing Options 선택
4. 6번 Serial 선택
5. No → Yes → Enter



시리얼 통신 환경 구축 라즈베리파이 ↔ Teraterm

시리얼 통신 환경 구축

6. "sudo vi /boot/config.txt" 입력
7. 맨 마지막에 "dtoverlay=pi3-disable-bt" 입력

```
[pi4]
# Enable DRM VC4 V3D driver on top of the dispmanx display stack
dtoverlay=vc4-fkms-v3d
max_framebuffers=2

[all]
#dtoverlay=vc4-fkms-v3d
enable_uart=1
dtoverlay=pi3-disable-bt
```

8. "sudo minicom -s" 입력
9. Serial port setup → A → "AMA0" 입력 → Enter → Save setup as df1

```
+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup           |
| Modem and dialing            |
| Screen and keyboard          |
| Save setup as df1            |
| Save setup as..              |
| Exit                          |
| Exit from Minicom            |
+-----+
```

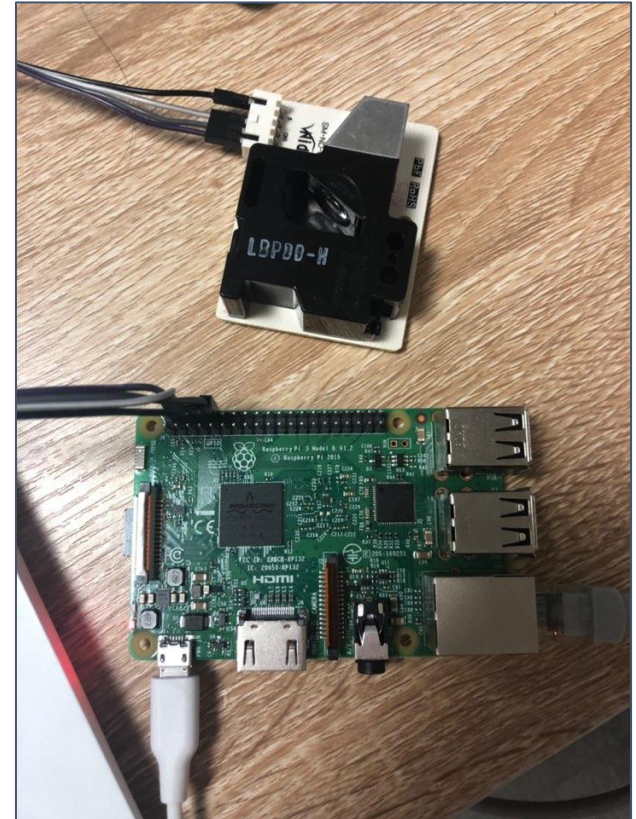
```
+-----+
| A - Serial Device           : /dev/ttyAMA0 |
| B - Lockfile Location       : /var/lock    |
| C - Callin Program          :              |
| D - Callout Program          :              |
| E - Bps/Par/Bits             : 9600 8N1     |
| F - Hardware Flow Control    : Yes          |
| G - Software Flow Control    : No           |
|                               |
| Change which setting? █      |
+-----+
```

시리얼 통신 환경 구축 SDS011 ↔ 라즈베리파이 ↔ Teraterm

시리얼 통신 환경 구축

라즈베리파이와 SDS011와 노트북을 연결한다.

→ 시리얼 통신 잘 되는지 확인



소켓 시리얼 프로그래밍

<serial.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);
```

```
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;
}
```

```
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 + serial 통신 + LEA server ↔ client

목표는?

1. server와 client를 연결
2. client에서 serial 통신으로 먼지데이터 암호화해서 server로 전달
3. server에서 암호화된 먼지데이터 복호화

serial_server.c



serial_client.c

```
192.168.0.10 - pi@raspberrypi: ~/Desktop/serial VT
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_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
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 + serial 통신 + LEA server ↔ client

< serial_server.c = server_echo.c + LEA 복호화 >

```
#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();
```

```
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 + serial 통신 + LEA server \leftrightarrow client

< serial_server.c = server_echo.c + LEA 복호화 >

```
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 + serial 통신 + LEA server \leftrightarrow client

< serial_client.c = serial.c + client_echo.c + LEA 암호화 >

```
#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 + serial 통신 + LEA server \leftrightarrow client

< serial_client.c = serial.c + client_echo.c + LEA 암호화 >

```
KeySchedule_128(K, RoundKey);

//1. uart setting function
serial_fd = set_uart(DEVPOR, 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 function
    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 + serial 통신 + LEA server \leftrightarrow client

< serial_client.c = serial.c + client_echo.c + LEA 암호화 >

```
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;
}
```

```
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 + serial 통신 + LEA server \leftrightarrow client

< serial_client.c = serial.c + client_echo.c + LEA 암호화 >

```
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;
}
```

```
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;
}
```