임베디드프로세서1 텀프로젝트

전자공학부 임베디드시스템 전공

2019146037 홍석영

1번

프로젝트 문제 및 핵심 코드 설명

- 1번은 키패드 및 부저를 연결하여 키패드 숫자 (1 ~ 9)까지는 음계의 4-도 ~ 5-레를 출력하고, 나머지는 5-미 를 부저를 통해 출력하는 것이 목표입니다.

- 부저는 PORTG.4를 출력으로 설정합니다.

- 키패드는 (PE4 ~ PE7)과 (PB0 ~ PB7)을 출력으로 설정하여 제어합니다.
  또한 PORTC 0~3을 이용해 키패드의 col값을 읽고 , 4~7를 이용해 키패드의 Row값을 읽습니다.

- KEYPADE 번호를 눌렀을 때 PORT상태에 따라 부저의 소리가 나게 설계 하였습니다.
  (EX.) 1번을 눌렀을 때 KEYPADE == 0x01로, if문을 사용하여 KEYPADE가 0x01일 때, '4-도' 소리가 나게 합니다.

1번 핵심코드

부저 관련 핵심 코드

```
#define    DO    1908 // 262Hz (3817us) 1908us
#define    RE    1700 // 294Hz (3401us) 1701us
#define    MI    1515 // 330Hz (3030us) 1515us
#define    FA    1432 // 349Hz (2865us) 1433us
#define    SOL   1275 // 370Hz (2703us) 1351us
#define    LA    1136 // 440Hz (2273us) 1136us
#define    SI    1012 // 494Hz (2024us) 1012us
#define    HIGH_DO    956 // 262Hz (1912us) 956us
#define    HIGH_RE    903 // 262Hz (1806us) 903us
#define    HIGH_MI    852 // 262Hz (1704us) 852us

void myDelay_us(unsigned int delay){
    int i;
    for(i=0; i<delay; i++){
        _delay_us(1);
    }
}

void SSound(int time) {
    int i, tim;
    tim = 25000 / time;     //0.05 초
    for(i=0; i<tim; i++){
        PORTG |= (1<<PG4); //buzzer on, PORTG의 4번 핀 off(out 1)
        myDelay_us(time);
        PORTG &= ~(1<<PG4); //buzzer off, PORTG의 4번 핀 on(on 1)
        myDelay_us(time);
    }
    PORTG |= (1<<PG4); // buzzer off, PORTG의 4번 핀 off(out 0)
}
```

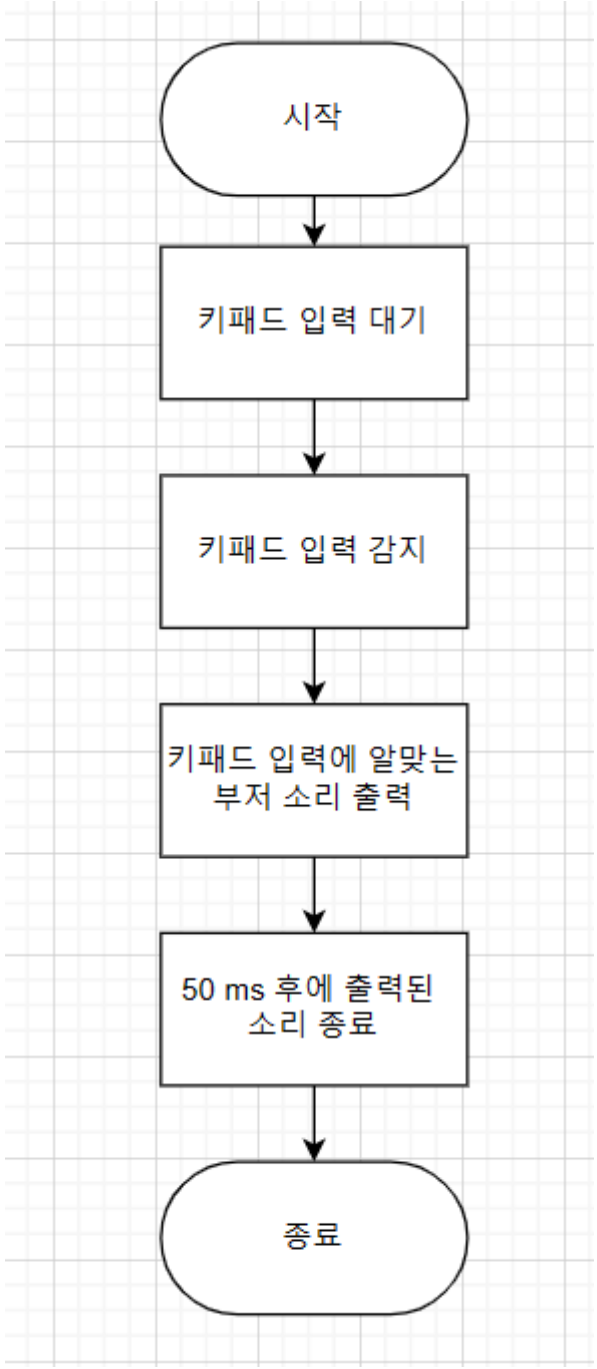
```
void gen_sound(int sw)
{
    if(         sw == 0x01){ SSound(DO); _delay_ms(500);} // 1
    else if(    sw == 0x02){ SSound(RE); _delay_ms(500);} // 2
    else if(    sw == 0x03){ SSound(MI); _delay_ms(500);} // 3
    else if(    sw == 0x04){ SSound(HIGH_MI); _delay_ms(500);} // M1
    else if(    sw == 0x05){ SSound(FA); _delay_ms(500);} // 4
    else if(    sw == 0x06){ SSound(SOL); _delay_ms(500);} // 5
    else if(    sw == 0x07){ SSound(LA); _delay_ms(500);} // 6
    else if(    sw == 0x08){ SSound(HIGH_MI); _delay_ms(500);} // M2
    else if(    sw == 0x09){ SSound(SI); _delay_ms(500);} // 7
    else if(    sw == 0x0a){ SSound(HIGH_DO); _delay_ms(500);} // 8
    else if(    sw == 0x0b){ SSound(HIGH_RE); _delay_ms(500);} // 9
    else if(    sw == 0x0c){ SSound(HIGH_MI); _delay_ms(500);} // M3
    else if(    sw == 0x0d){ SSound(HIGH_MI); _delay_ms(500);} // *
    else if(    sw == 0x0e){ SSound(HIGH_MI); _delay_ms(500);} // 0
    else if(    sw == 0x0f){ SSound(HIGH_MI); _delay_ms(500);} // #
    else if(    sw == 0x10){ SSound(HIGH_MI); _delay_ms(500);} // M4
}
// 여기까지 부저
void PORT_Init(void){
    DDRG  |= (1<<PG4);    // 부저와 연결되는 PORTG.4를 출력으로 설정
    PORTG |= (1<<PG4);    // 교육용보드의 BUZZ는 회로가 ACctive-Low로 되어있음으로
                          // HIGH 상태 출력하여 부저 꺼줌
```

1번 핵심코드

키패드 관련 핵심 코드

```
#define SetRow(x)   PORTC |= 0xF0;  PORTC &= ~(1<<((x-1)+PC4));
unsigned char Port_char[] = {
    0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xd8, 0x80, 0x90, 0x88, 0x83, 0xc4, 0xa1, 0x84, 0x8e
}; // 애노드 공통

unsigned int Port_fnd[] = {0x1f, 0x2f, 0x4f, 0x8f}; // FND0 ON, FND1 ON, FND2 ON, FND3 ON

void PORT_Init(void){
    DDRG  |= (1<<PG4);   // 부저와 연결되는 PORTG.4를 출력으로 설정
    PORTG |= (1<<PG4);   // 교육용보드의 BUZZ는 회로가 ACctive-Low로 되어있음으로
                         // HIGH 상태 출력하여 부저 꺼줌

    DDRE = 0xf0;        // FND 제어위한 출력 설정(PE4..7)
    DDRB = 0xff;        // FND 제어위한 출력 설정(PB0..7)
    DDRC = 0xF0;        // PORTC 4 ~ 7 은 Row 선택위한 출력으로 설정
                        // PORTC 0 ~ 3 은 col 값 읽기위한 입력으로 설정
}
```

```
keypad_code = 0xFF;  // 키패드 입력감지 위해 초기값 설정(이후에 0xff가 아니면 키패드 입력 있음)
for(sel_row=1; sel_row<=4; sel_row++){// row를 바꾸어 가며 키패드 읽기
    SetRow(sel_row);   // row 선택
    _delay_ms(10);      // 안정적인 col 읽기 위함
    col = (PINC & 0x0F);
    switch(col){
        case 0x01 : keypad_code = (sel_row - 1)*4 + 1; break; // 1번 col의 스위치가 눌린 경우
        case 0x02 : keypad_code = (sel_row - 1)*4 + 2; break; // 2번 col의 스위치가 눌린 경우
        case 0x04 : keypad_code = (sel_row - 1)*4 + 3; break; // 3번 col의 스위치가 눌린 경우
        case 0x08 : keypad_code = (sel_row - 1)*4 + 4; break; // 4번 col의 스위치가 눌린 경우
    }
}
```



1번
FLOW CHART

2번

프로젝트 문제 및 핵심 코드 설명

- 2번은 1번 상태에서 서보 모터 + FND + LCD 를 연결하는것이 목표입니다.
- 초기상태는 if문을 사용하여 '#' 버튼을 눌러서 비밀번호 입력창이 뜨기 전까지는 FND와 LCD에 아무것도 뜨지 않게 하였습니다.
- FND는 PORTE를 통해 FND Digit을 선택하고, PORTB를 통해 키패드 번호를 출력합니다.
  키패드의 문자(*,# 등...)는 FND에 표시할 때 아스키코드 10진수로 표현합니다.
  M1 , M2, M3, M4는 각각 'A', 'B', 'C', 'D' 로 표현하였습니다.
- LCD 는 POATA 와 PORTG 의 하위 4비트를 출력으로 사용합니다.
- 서보 모터는 PORTF.4를 출력으로 사용합니다.

2번 핵심 코드

<1> 초기 상태에서는 부저를 통하여 "4-솔"음을 1회/ 100ms 출력

```
void TWO_SSound(int time) {
    int i, tim;
    tim = 50000 / time;     //0.1 초
    for(i=0; i<tim; i++){
        PORTG |= (1<<PG4); //buzzer on, PORTG의 4번 핀 off(out 1)
        myDelay_us(time);
        PORTG &= ~(1<<PG4); //buzzer off, PORTG의 4번 핀 on(on 1)
        myDelay_us(time);
    }
    PORTG |= (1<<PG4); // buzzer off, PORTG의 4번 핀 off(out 0)
}

TWO_SSound(SOL);
```

2번 핵심 코드

<2> 초기 상태 RC 서보 모터

```
unsigned int T2_DUTY_TIME_us;    // PWM 파형 듀티 폭 시간
unsigned int T2_CYCLE_TIME_us;   // PWM 파형 주기 시간

unsigned int T2_DUTY_TIME_CNT_us; // PWM 파형 듀티 시간을 세기 위한 카운트 변수
unsigned int T2_CYCLE_TIME_CNT_us; // PWM 파형 주기 시간을 세기 위한 카운트 변수

// 100uS 주기 인터럽트
ISR(TIMER2_COMP_vect){
    T2_CYCLE_TIME_CNT_us += 100;
    T2_DUTY_TIME_CNT_us += 100;
    if(T2_DUTY_TIME_CNT_us <= T2_DUTY_TIME_us){
        PORTF |= (1<<PF3);
        }else{
        PORTF &= ~(1<<PF3);
    }

    if(T2_CYCLE_TIME_CNT_us == T2_CYCLE_TIME_us){
        T2_CYCLE_TIME_CNT_us = 0;
        T2_DUTY_TIME_CNT_us = 0;
    }
}
```

```
void Init_Timer2(void) {  //타이머/카운터 2 - 2초 설정 및 PC PWM 듀티비 설정
    TIMSK |= (1 << OCIE2);                // ⊙ 출력비교 인터럽트2 허가
    TCCR2 = (1 << WGM21) | (2 << CS20);   // ⊙ CTC 모드, 1024분주
    OCR2 = 184;                            // ⊙ 약 100us
    // OCR2 = (14745600Hz / 8 분주비) * 100us = 184.32
}

void Init_TimerINT(void) {
    Init_Timer2();
    SREG |= 0x80;
}
void SetServoDeg(unsigned int deg) {
    T2_DUTY_TIME_us = 500 + (deg*200/18); // 듀티 폭 시간 생성부분
}

void PORT_TIMER_Init(void){
    DDRF |= (1 << PF3);
}
```

2번 핵심 코드

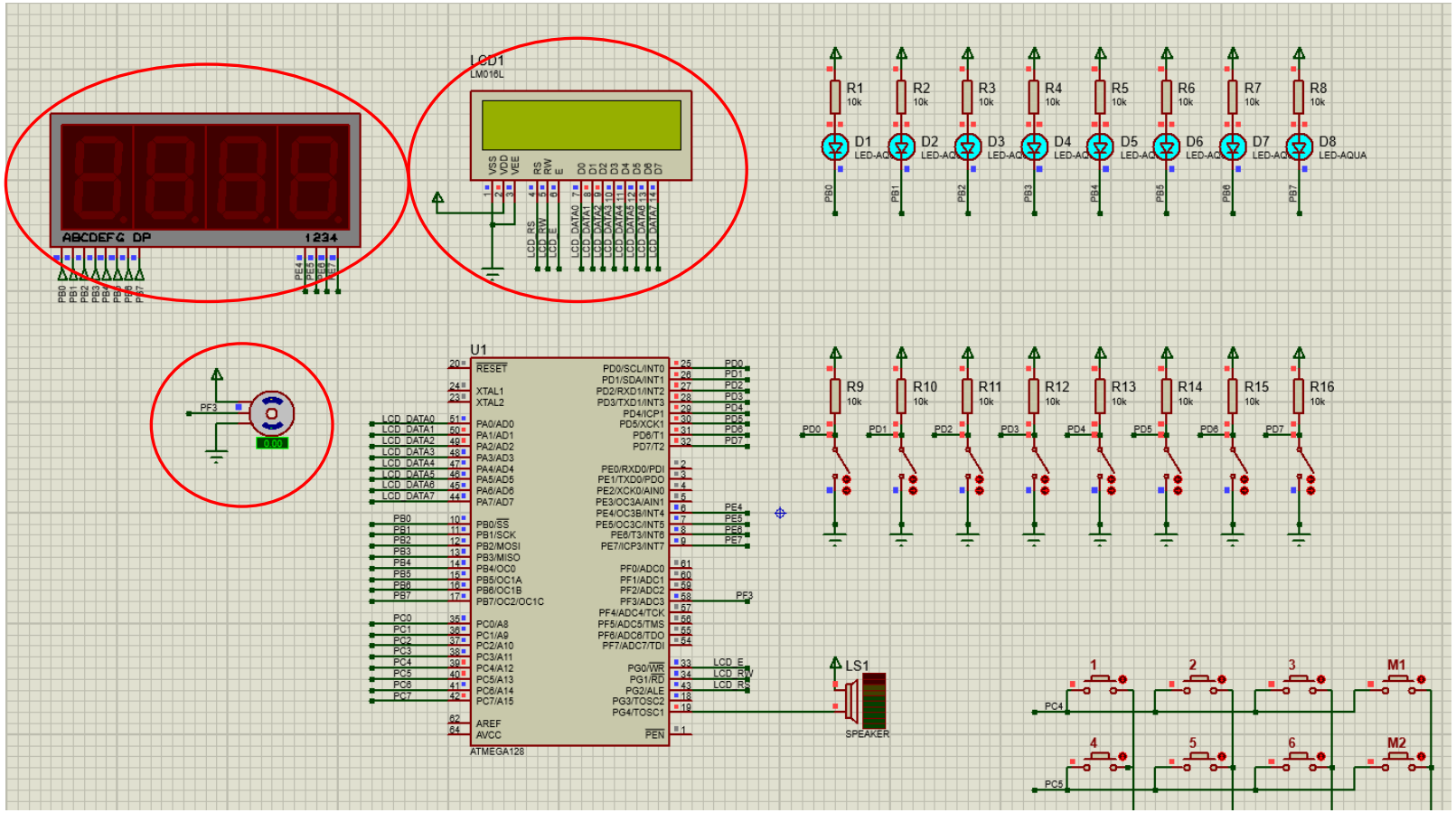
<2> 초기 상태 LCD

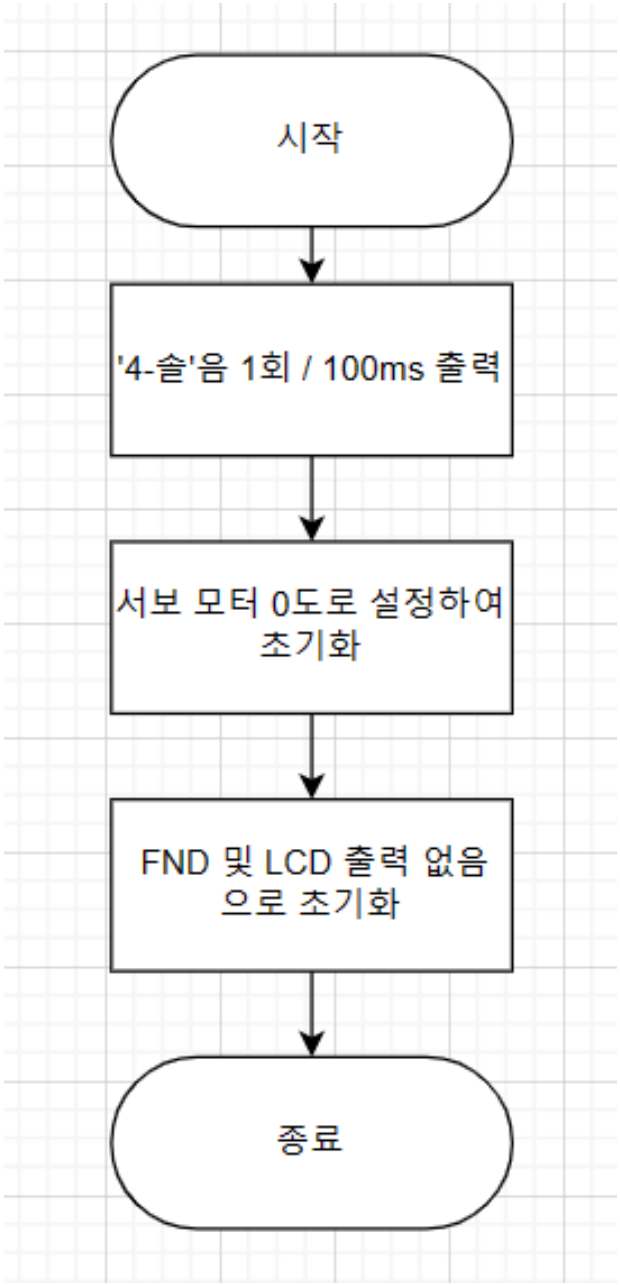
```
#define LCD_WDATA PORTA     // LCD 데이터 버스 정의 (데이터 쓰기)
#define LCD_WINST PORTA     // LCD 데이터 버스 정의 (명령어 쓰기)
#define LCD_CTRL  PORTG     // LCD 제어 신호 정의
#define LCD_EN    0         // Enable 신호
#define LCD_RW    1         // 읽기(1)/쓰기(0)
#define LCD_RS    2         // 데이터(1)/명령어(0)
#define RIGHT 1
#define LEFT 0

void PortInit (void)
{
    DDRA = 0xFF; // PORTA(LCD data pin)를 출력으로
    DDRG = 0x0F; // PORTG의 하위 4비트를 출력으로
}
```

```
void LCD_Comm(unsigned ch)
{
    LCD_CTRL &= ~(1 << LCD_RS);      // RS==0으로 명령(IR 레지스터 접근 설정)
    LCD_CTRL &= ~(1 << LCD_RW);      // RW==0으로 쓰기 설정
    LCD_CTRL |=  (1<< LCD_EN);       // LCD Enable
    _delay_us(50);                   // 시간지연
    LCD_WINST = ch;                  // 명령어쓰기
    _delay_us(50);                   // 시간지연
    LCD_CTRL &= ~(1 << LCD_EN);      // LCD Enable
}

void LCD_Data( unsigned char ch )
{
    LCD_CTRL |= (1 << LCD_RS);       // RS==1으로 데이터(DR 레지스터 접근 설정)
    LCD_CTRL &= ~(1 << LCD_RW);      // RW==0으로 쓰기 설정
    LCD_CTRL |= (1 << LCD_EN);       // LCD Enable
    _delay_us(50);                   // 시간지연
    LCD_WDATA = ch;                  // 명령어쓰기
    _delay_us(50);                   // 시간지연
    LCD_CTRL &= ~(1 << LCD_EN);      // LCD Enable
}
```

```
void LCD_Init(void)
{
    LCD_Comm(0x30);     // 초기화 Set
    _delay_us(4100);    // 4.1ms 지연
    LCD_Comm(0x30);     // 초기화 Set
    _delay_us(100);     // 1000us 지연
    LCD_Comm(0x30);     // 초기화 Set
    _delay_us(100);     // 1000us 지연

    LCD_Comm(0x38);     // 초기화 Set, 데이터 길이 8Bit, 표시라인 2행 사용을 위한 기능 설정
    _delay_us(1000);    // 명령을 처리 하는데 최소 40us 지연이 발생하기에 여유를 고려하여 설정

    LCD_Comm(0x0c); // 화면 On/Off(3) , Display ON/Off(2), Cursor On/Off(1), Blink On/Off(1)
    _delay_us(1000); // 1000us 지연

    LCD_Comm(0x01); // LCD Clear
    _delay_us(2000); // 2000us 지연
    LCD_Comm(0x06);  // Cursor Entry Mode Set, 표시 위치 +`씩 증가
    _delay_us(1000); // 1000us 지연
}
```

2번 초기상태





2번
FLOW CHART

3번

프로젝트 문제 및 핵심 코드 설명

- 3번은 비밀번호를 키패드로 입력하고 일치 여부를 확인하는 것이 목표입니다.
- 키패드를 누를때마다 테스트 패스워드(test_password)에 키패드 값을 저장하고 '*' 키패드를 누르면 사용자 패스워드(user_password)와 비교하여 일치 여부를 파악한 뒤 배열 값을 초기화 시킵니다.
- 일치 여부는 정확도 변수(accuracy)를 통해 테스트 패스워드와 사용자 패드워드의 값을 하나하나씩 비교해 맞을 때마다 accuracy 값을 1씩 증가시켜 accuracy값이 비밀번호 배열 값 길이(15) 값이 되면 일치하였다 판단하고 배열 길이보다 작을 시 틀렸다고 판단합니다.
- if 문을 사용해 일치했을 경우와 불일치 했을 경우의 코드를 작성하여 문제의 조건대로 실행되게 합니다.

3번 핵심 코드

<3> 부저 소리 '500ms'를 실행해주는 코드

```
void THREE_SSound(int time) {
    int i, tim;
    tim = 250000 / time;     //0.5 초
    for(i=0; i<tim; i++){
        PORTG |= (1<<PG4); //buzzer on, PORTG의 4번 핀 off(out 1)
        myDelay_us(time);
        PORTG &= ~(1<<PG4); //buzzer off, PORTG의 4번 핀 on(on 1)
        myDelay_us(time);
    }
    PORTG |= (1<<PG4); // buzzer off, PORTG의 4번 핀 off(out 0)
}
```

3번 핵심 코드

```
else{
    if(         sw == 0x01){LCD_pos(1,position); LCD_CHAR('1'); test_password[position-1] = 1; position += 1; select_count = 0;} // 1
    else if(    sw == 0x02){LCD_pos(1,position); LCD_CHAR('2'); test_password[position-1] = 2; position += 1; select_count = 0;} // 2
    else if(    sw == 0x03){LCD_pos(1,position); LCD_CHAR('3'); test_password[position-1] = 3; position += 1; select_count = 0;} // 3
    else if(    sw == 0x04){position = position;} // M1
    else if(    sw == 0x05){LCD_pos(1,position); LCD_CHAR('4'); test_password[position-1] = 4; position += 1; select_count = 0;} // 4
    else if(    sw == 0x06){LCD_pos(1,position); LCD_CHAR('5'); test_password[position-1] = 5; position += 1; select_count = 0;} // 5
    else if(    sw == 0x07){LCD_pos(1,position); LCD_CHAR('6'); test_password[position-1] = 6; position += 1; select_count = 0;} // 6
    else if(    sw == 0x08){position = position;} // M2
    else if(    sw == 0x09){LCD_pos(1,position); LCD_CHAR('7'); test_password[position-1] = 7; position += 1; select_count = 0;} // 7
    else if(    sw == 0x0a){LCD_pos(1,position); LCD_CHAR('8'); test_password[position-1] = 8; position += 1; select_count = 0;} // 8
    else if(    sw == 0x0b){LCD_pos(1,position); LCD_CHAR('9'); test_password[position-1] = 9; position += 1; select_count = 0;} // 9
    else if(    sw == 0x0c){position = position; select_count = 0;} // M3
    else if(    sw == 0x0d){
        select_count +=1;
        for(int i = 0; i <sizeof(user_password); i++){
            if(test_password[i] == user_password[i]) { accuracy += 1; }
            else { accuracy = accuracy;}}

            if (accuracy == sizeof(user_password)) {LCD_Clear(); error_count = 0;
                THREE_SSound(DO); _delay_ms(1000); THREE_SSound(MI); _delay_ms(1000); THREE_SSound(SOL); _delay_ms(1000);  THREE_SSound(HIGH_DO);
                SetServoDeg(90);
                LCD_pos(0,1); LCD_STR(str_dooropen);
            }
            else {LCD_Clear(); error_count += 1;
                THREE_SSound(LA); _delay_ms(1000); THREE_SSound(LA); _delay_ms(1000); THREE_SSound(LA);
                LCD_pos(0,1); LCD_STR(str_password);
                LCD_pos(1,2); LCD_STR(str_error);
                _delay_ms(3000); LCD_Clear(); fnd_value = 0; state = 0;
            }
            for (int i=0; i<15; i++){

            test_password[i] = 10; // 다음번에 번호를 입력할 때 겹치지 않도록 초기화 시켜주는 과정
            }
            position = 1; accuracy = 0;
        } // *

        else if(    sw == 0x0e){LCD_pos(1,position); LCD_CHAR('0'); test_password[position-1] = 0; position += 1; select_count = 0;} // 0
        else if(    sw == 0x0f){state = 1; position = 1; select_count = 0; LCD_Clear(); LCD_pos(0,1); _delay_ms(1); LCD_STR(str_password); LCD_pos(1,1); LCD_STR(number); select_count = 0;} // #
        else if(    sw == 0x10){position = position; select_count = 0;} // M4
```

<2> 키패드 누를 때마다 LCD에 번호 값 출력

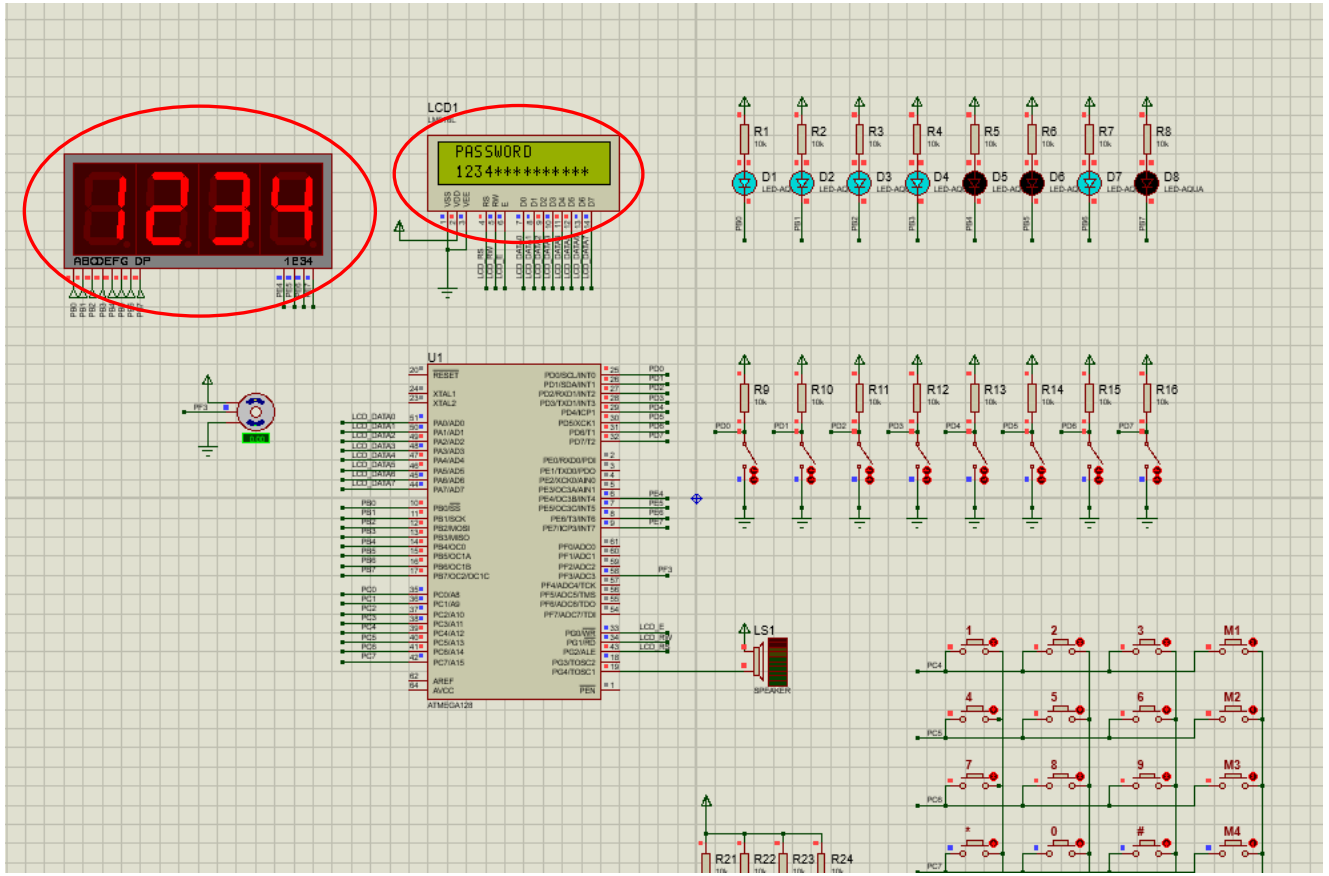
<3> '*' 눌러 비밀 번호 일치 확인 및
서브모터 '90'도로 변경 및
부저 소리 출력

<1> '#'버튼 눌렀을 때

3번 핵심 코드

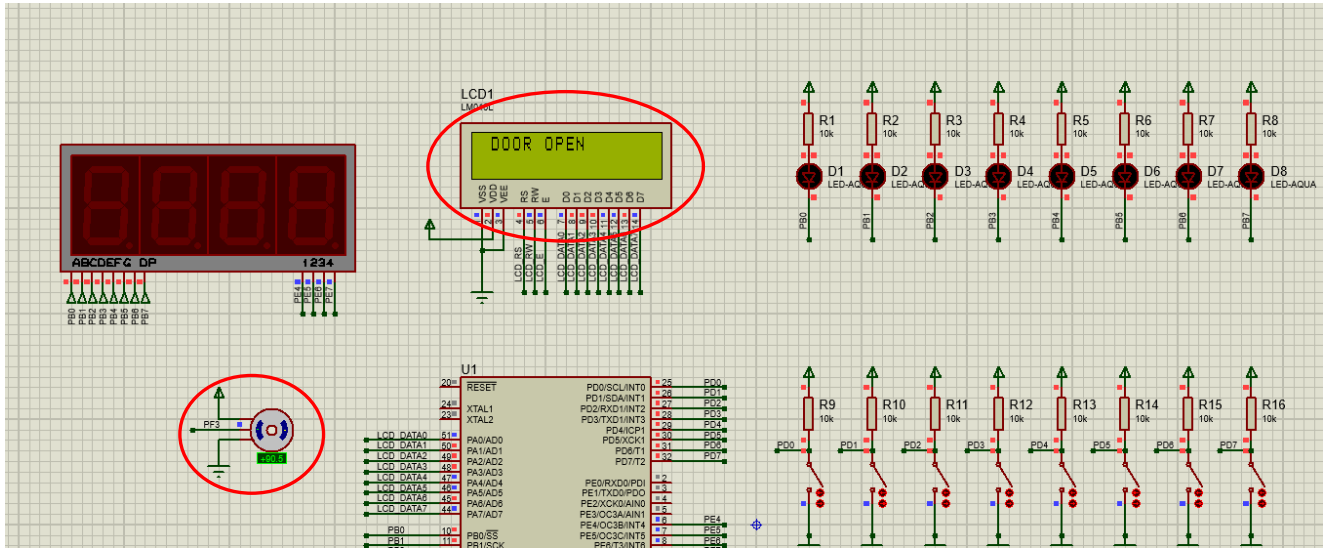
<2> 키패드 누를 때마다 FND에 번호 값 출력

```
else{
FND_Disp(fnd_value); // FND에 숫자 출력 [4자리]
}

 if((keypad_code != 0xFF) && (hkeypad_code != keypad_code)){
     fnd_value %= 1000;
     fnd_value *= 10;
     fnd_value += getNum(keypad_code);
 }
 hkeypad_code = keypad_code; // 키패드가 눌렸으면 FND 표시
```

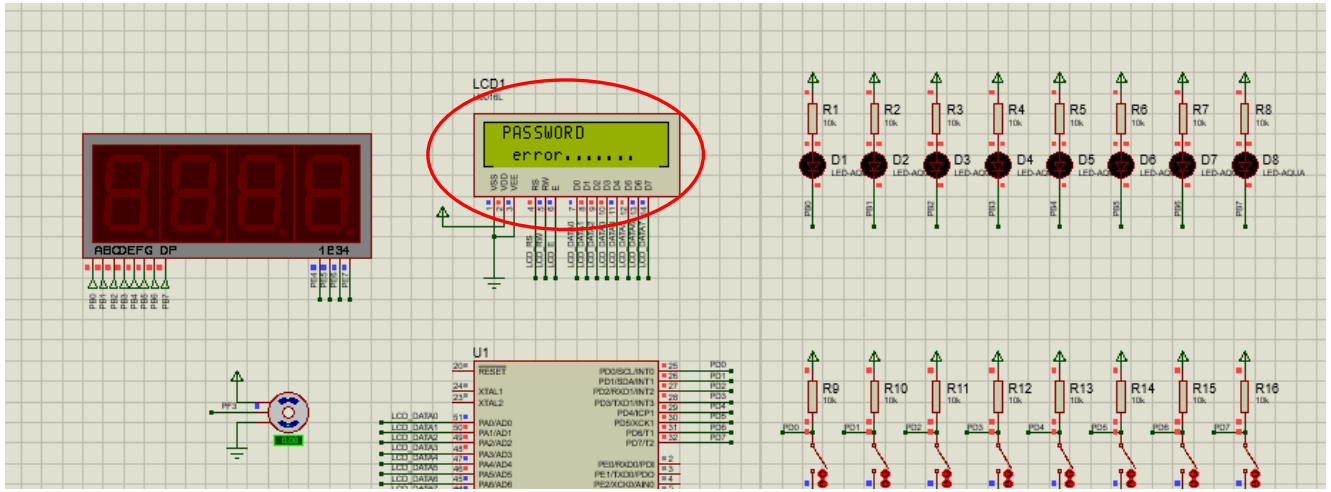


3번
LCD와 FND 화면 출력

<3> 비밀번호 일치 시



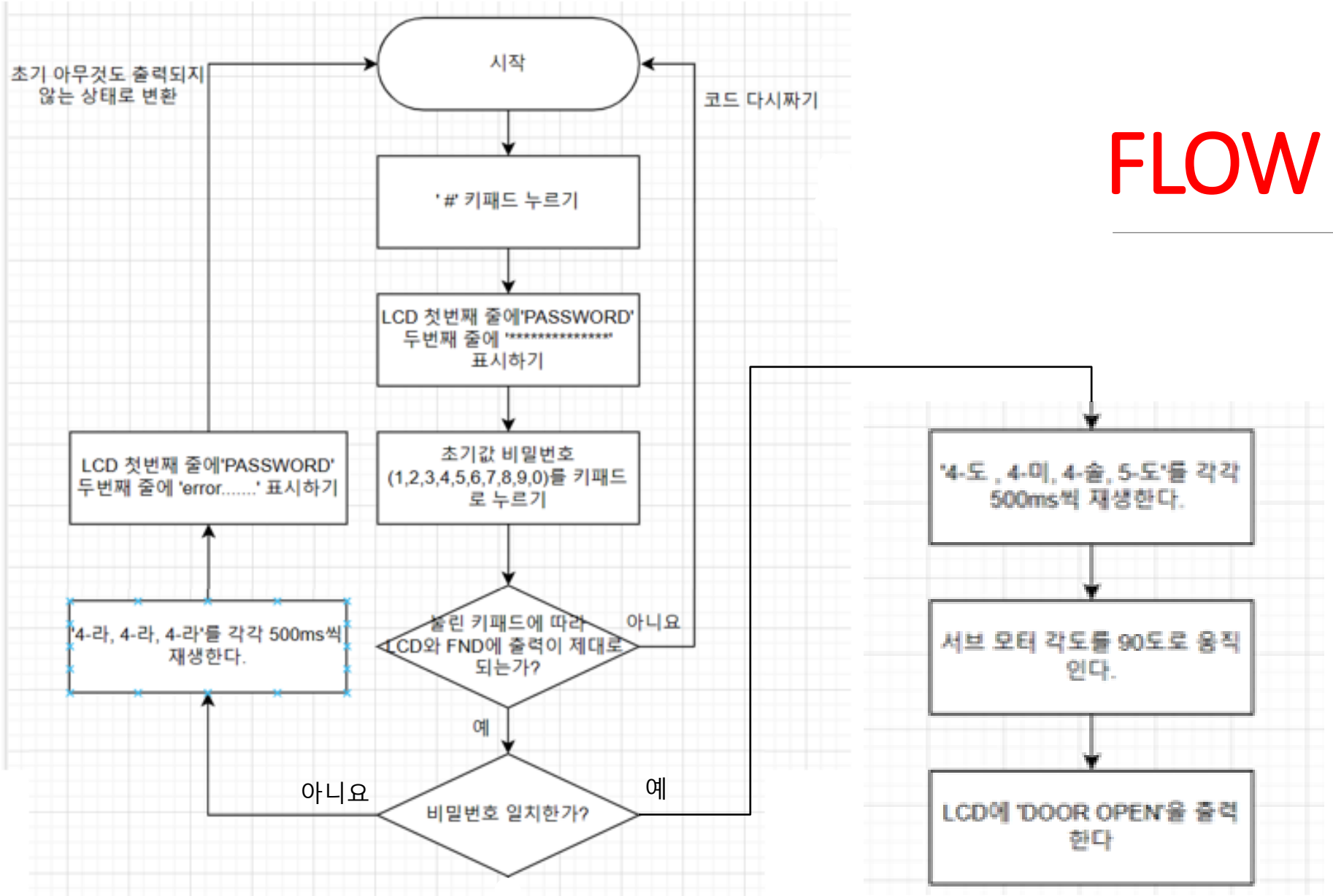
<3> 비밀번호 불일치 시



3번

LCD와 FND 화면 출력

3번
FLOW CHART



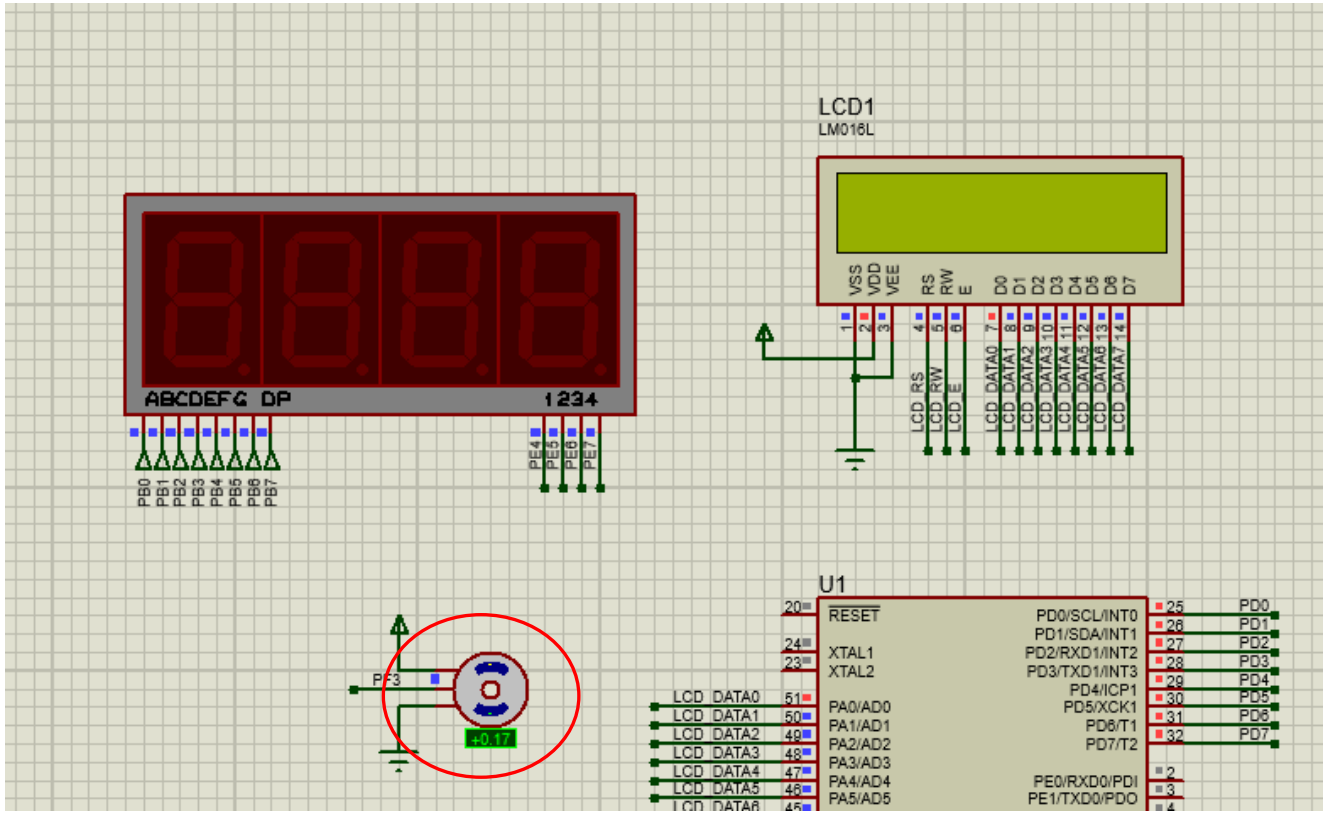
4번

프로젝트 문제 및 핵심 코드 설명

- 4번은 '*' 버튼을 3초 이상 눌렀을 때 그에 알맞은 부저소리와 서보 모터를 '0'도로 바꾸는게 목표입니다.
- '*' 버튼이 눌렸을 때 비밀번호가 일치할 경우와 일치하지 않을 경우가 있는데 두개의 케이스 모두 진행이 완료되는데 약 3초의 시간이 걸립니다. 따라서 눌린 경우 select_count 값을 1씩 증가시켜 연속으로 2번 진행되었을 경우 서보 모터를 움직입니다.
- 서보 모터는 타이머/카운터 에서 설정한 듀티비 및 분주를 통해 약 0도의 각도를 만드는 함수를 실행시켜서 서보 모터를 제어 합니다.
- 부저의 경우 '4-솔 4-파 4-미 4-레' 사이에 0.5초 지연 시간을 배치해서 음계의 출력 횟수를 파악할 수 있게 합니다.

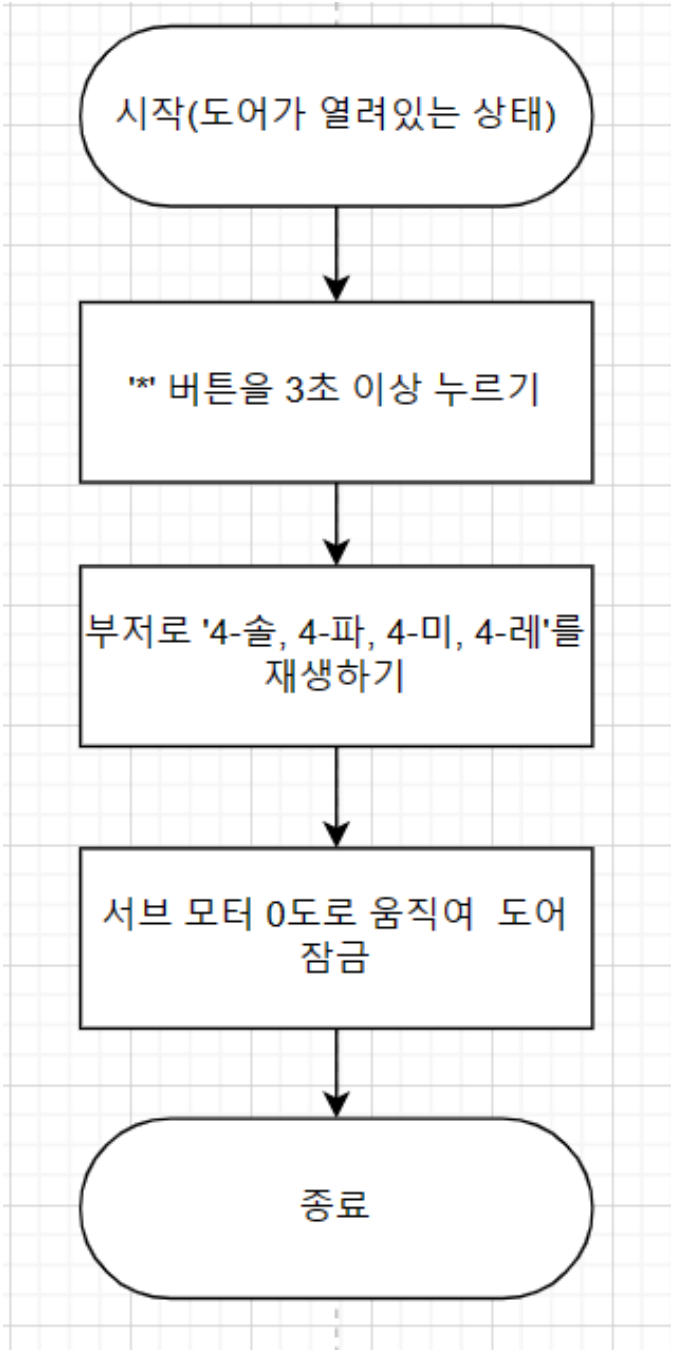
4번 핵심 코드

```
// 4번
if(select_count >=2 ) {
    SSound(SOL); _delay_ms(500); SSound(FA); _delay_ms(500); SSound(MI); _delay_ms(500); SSound(RE);
    SetServoDeg(0);
    select_count = 0;
}
else {
    select_count = select_count;
}
```



4번

서브 모터 '0'도로 설정하기



4번
FLOW CHART

5번

프로젝트 문제 및 핵심 코드 설명

- 5번은 외부 인터럽트(PD0)를 이용해 비밀번호를 변경하는 것이 목표입니다.
- 외부 인터럽트에 의해 PD0 가 눌렸을 때 Count 값이 1이 증가하고, if문을 사용해 Count 값이 1일때 비밀번호를 재 설정하는 코드를 작성하고 else 일때 작성된 비밀번호를 새로운 비밀번호로 저장하는 코드를 작성한 뒤 Count 값을 다시 0으로 초기화 시켜줍니다.
- Count 값이 1일 때, 새로운 비밀번호 배열(new_password)에 키패드로 입력받은 키들을 저장하고, 그 뒤 else 부분일 때 새로운 비밀번호(new_password)의 값을 사용자 패스워드(user_password)에 넣어 새롭게 비밀번호를 변경합니다.
- 부저의 경우 '4-도 4-레 4-미 ' 사이에 0.5초 지연 시간을 배치해서 음계의 출력 횟수를 파악할 수 있게 합니다.

5번 핵심 코드

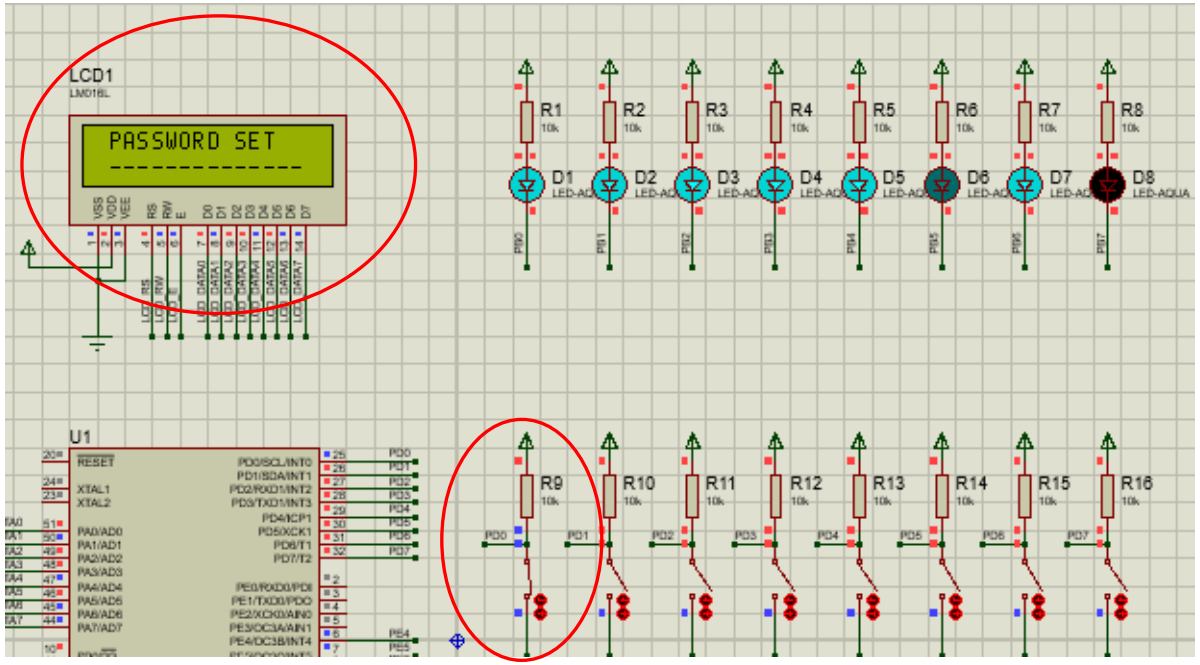
함수 코드

```
unsigned char Count;
ISR (INT0_vect){
    unsigned char str_password_set[] = "PASSWORD SET";
    unsigned char str_input_password_set[] = "--------------";
    unsigned char str_success[] = "-SUCCESS-";
    // 비밀번호 변경
    if (Count == 0){
        LCD_Clear();
        LCD_pos(0,1); LCD_STR(str_password_set);
        LCD_pos(1,1); LCD_STR(str_input_password_set);
        Count++;
    }
    // 비밀번호 변경 완료
    else{
        LCD_Clear();
        SSound(DO); _delay_ms(1000); SSound(RE); _delay_ms(1000); SSound(MI);
        LCD_pos(0,1); LCD_STR(str_password_set);
        LCD_pos(1,1); LCD_STR(str_success);
        Count = 0;
    }
}
void interrupt_init(void)
{
    EIMSK = 0x01;       // 외부인터럽트 종류 선택
    EICRA = 0x02;       // 인터럽트 제어방법 선택(INT0 하강에지트리거)
    DDRD = 0x00;         // TACT 스위치를 입력으로 설정 + 포트 D를 입력으로 선택
    sei();              // 인터럽트 동작시작(전체인터럽트허가)
}
// 여기까지 외부 인터럽트
```

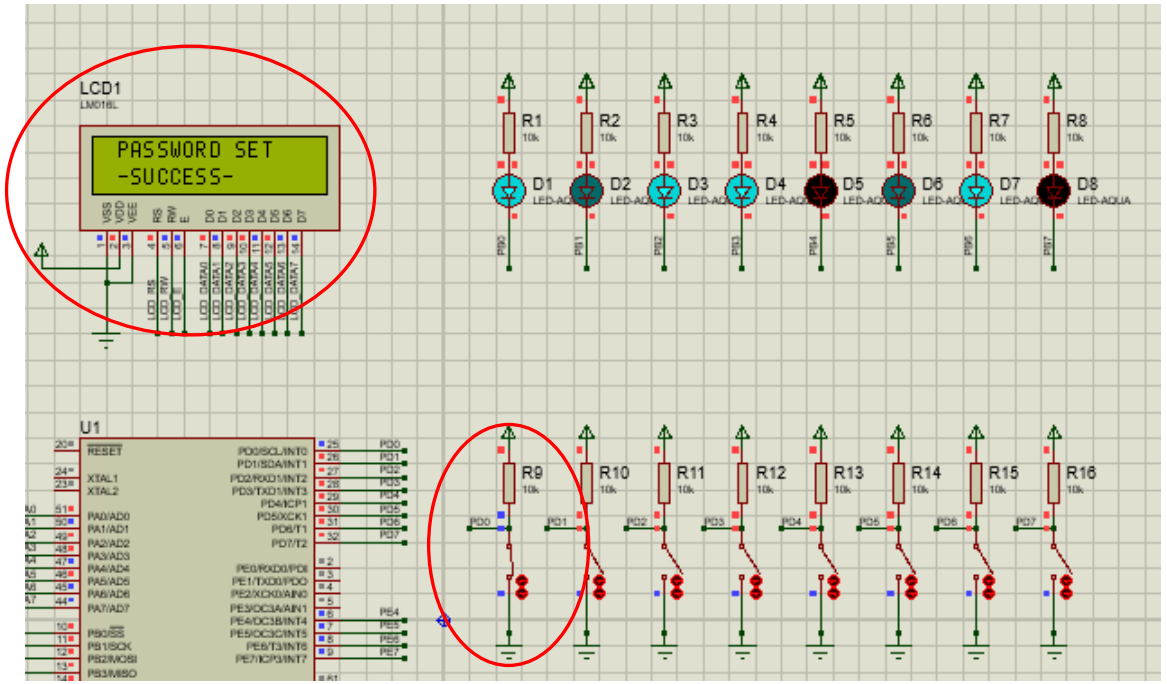
main문 안쪽 키패드 암호에 적용하는 코드

```
sw = keypad_code;
if(Count ==1){
    select_count = 0;
    if(         sw == 0x01){LCD_pos(1,position); LCD_CHAR('1'); new_password[position-1] = 1; position += 1;} // 1
    else if(    sw == 0x02){LCD_pos(1,position); LCD_CHAR('2'); new_password[position-1] = 2; position += 1;} // 2
    else if(    sw == 0x03){LCD_pos(1,position); LCD_CHAR('3'); new_password[position-1] = 3; position += 1;} // 3
    else if(    sw == 0x04){position = position;} // M1
    else if(    sw == 0x05){LCD_pos(1,position); LCD_CHAR('4'); new_password[position-1] = 4; position += 1;} // 4
    else if(    sw == 0x06){LCD_pos(1,position); LCD_CHAR('5'); new_password[position-1] = 5; position += 1;} // 5
    else if(    sw == 0x07){LCD_pos(1,position); LCD_CHAR('6'); new_password[position-1] = 6; position += 1;} // 6
    else if(    sw == 0x08){position = position;} // M2
    else if(    sw == 0x09){LCD_pos(1,position); LCD_CHAR('7'); new_password[position-1] = 7; position += 1;} // 7
    else if(    sw == 0x0a){LCD_pos(1,position); LCD_CHAR('8'); new_password[position-1] = 8; position += 1;} // 8
    else if(    sw == 0x0b){LCD_pos(1,position); LCD_CHAR('9'); new_password[position-1] = 9; position += 1;} // 9
    else if(    sw == 0x0c){position = position;} // M3
    else if(    sw == 0x0d){position = position;} // *

    else if(    sw == 0x0e){LCD_pos(1,position); LCD_CHAR('0'); new_password[position-1] = 0; position += 1;} // 0
    else if(    sw == 0x0f){position = position;} // #
    else if(    sw == 0x10){position = position;} //M4
   // 다음번 번호 입력할 때 이번 입력과 겹치지 않도록 배열을 초기화 하는 과정
   for(int i=0; i<(position-1); i++){
       user_password[i] = new_password[i];
       test_password[i] = 10;
   }
   for(int i=position-1; i<15; i++){
       user_password[i] = 10;
       new_password[i] = 10;
       test_password[i] = 10;
   }
}
```

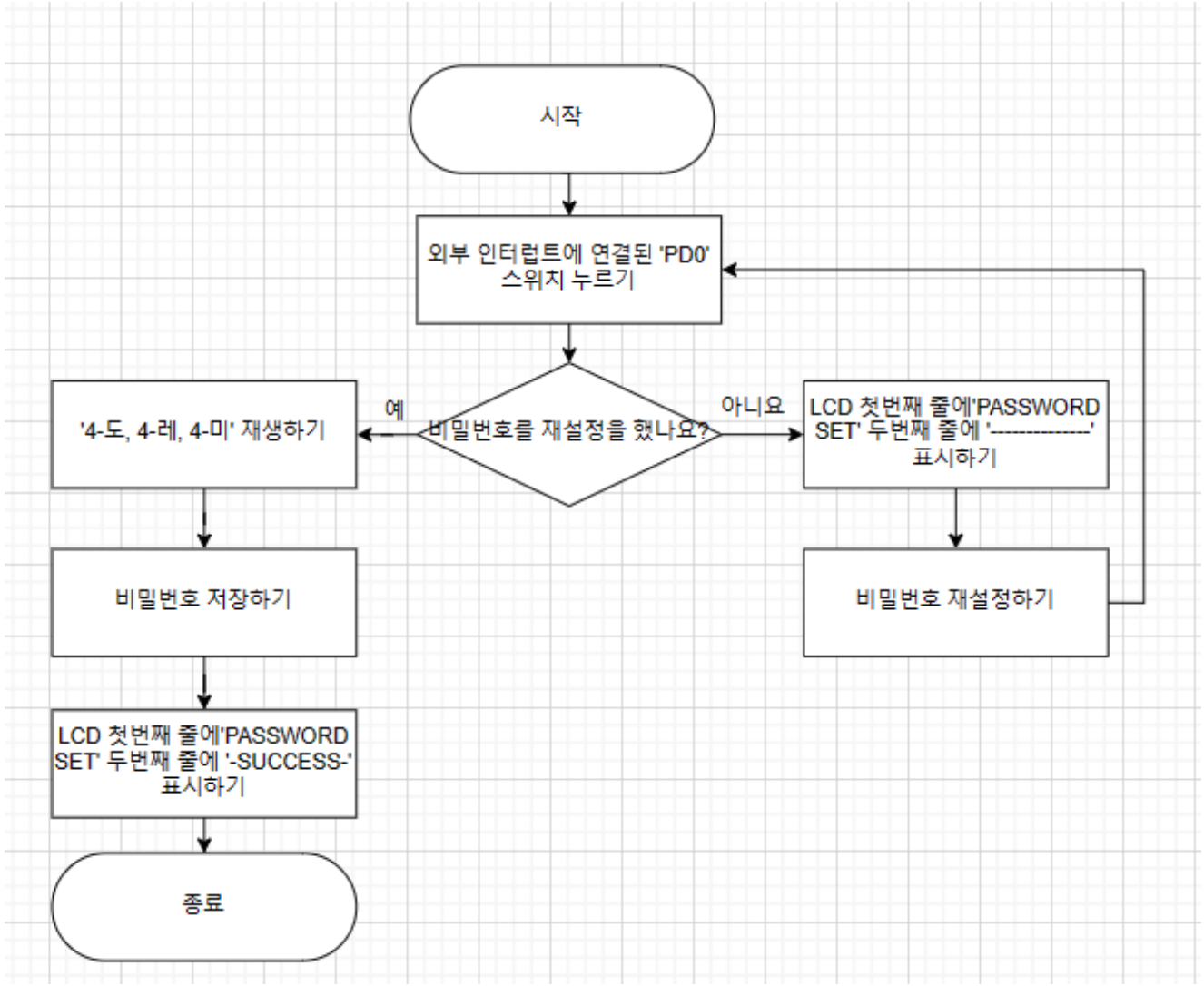
비밀번호 변경 화면



비밀번호 변경 완료 화면



5번



5번
FLOW CHART

6번

프로젝트 문제 및 핵심 코드 설명

- 6번은 보안 대책을 작성하는 코드로서, 비밀번호 3회 이상 틀릴 시 + 금고의 이동 또는 충격이 발생할 시 싸이렌 소리가 들리게 하고, 이를 마스터 키 번호를 눌러 종료하는 것이 목표입니다.
- 비밀 번호를 틀릴 때 마다 비밀번호 변수(error_count)를 1증가 시켜 변수 값이 3이상이 될 때 조건문을 활용하여 싸이렌 및 LCD에 경고문을 작성합니다.
- 디지털 출력 진동 센서 및 조건문을 활용하여 이동 또는 충격이 발생될 때 싸이렌 경고음과 LCD에 경고문을 출력합니다.
- 이 때, 싸이렌은 '레' 음부터 높은 '솔' 음에 달하는 300~750Hz 소리를 천천히 발생시켜 만듭니다.
- 마스터 키 모드진입은 '*'과 '#' 버튼을 동시에 눌렀을 때 작동되게 합니다.

6번 핵심 코드

<1> 비밀번호 3회 이상 틀렸을 때 코드

```
void Siren(void){
    for (int i =1700; i>637; i--){
        SSound(i);
        i -=10;
    }

    for (int i = 638; i<1701; i++){
        SSound(i);
        i+=20;
    }
}
```

```
        // 6번
        if(error_count >= 3) {
            // 비밀번호 3번 이상 틀릴 시
            Siren();
            LCD_pos(0,1); LCD_STR(str_warning);
            LCD_pos(1,2); LCD_STR(str_theft);
        }
    }
```

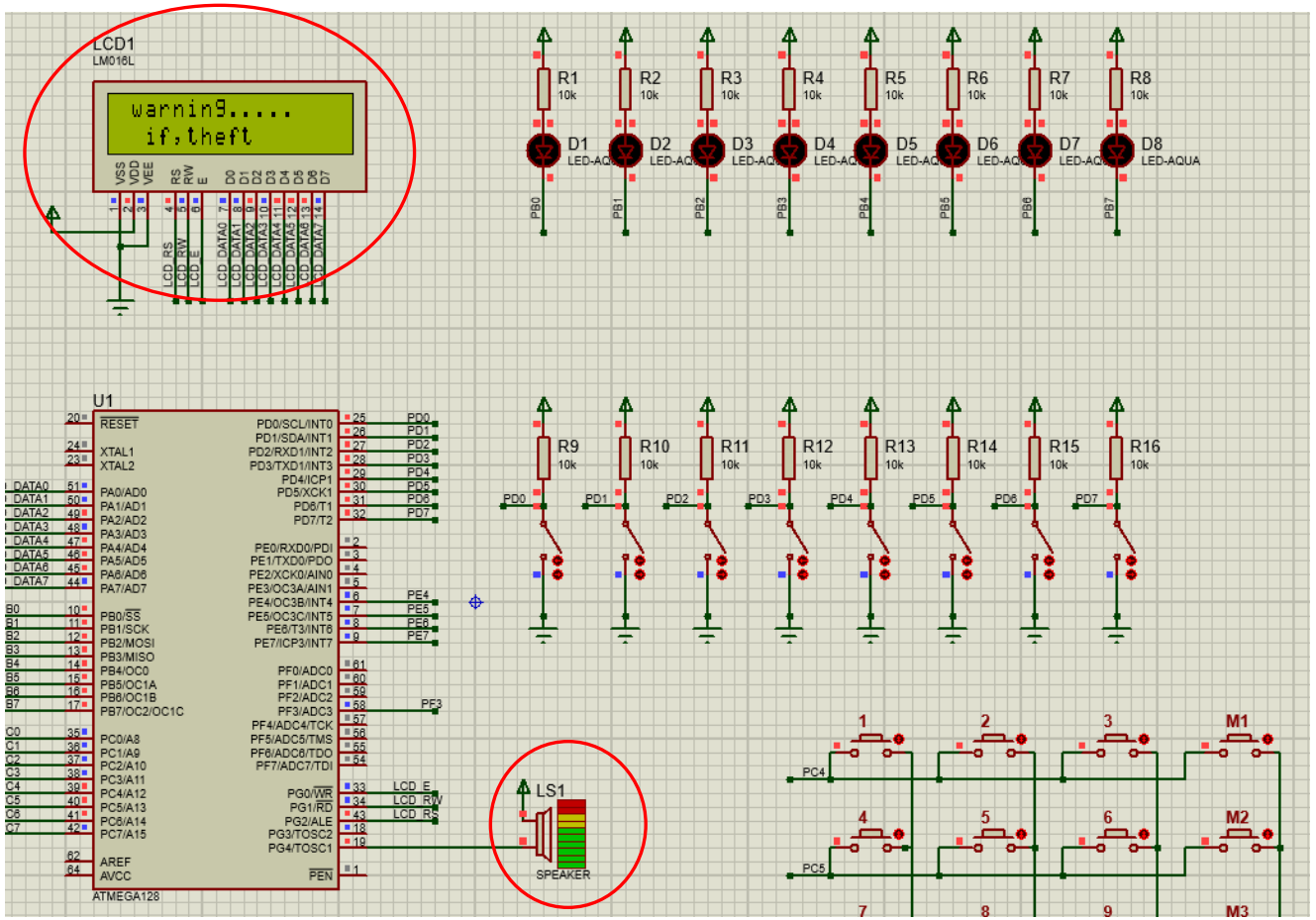
6번 핵심 코드

<2> 진동센서 감지 관련 코드

```
void Stealing(void) {
    unsigned char str_stealwarning[] = "warning.....";
    unsigned char str_stealing[] = "stealing!!!";
    Siren();
    LCD_pos(0,1); LCD_STR(str_stealwarning);
    LCD_pos(1,2); LCD_STR(str_stealing);
}
```

<3> 마스터 키 모드 관련 코드

```
void Master(void) {
    unsigned char str_mastermode[] = "MASTER MODE";
    unsigned char masterkey[] = "2019146037";
    unsigned char user_master[15];
    unsigned char star[] = "**********"
    LCD_pos(0,1); LCD_STR(str_mastermode);
    LCD_pos(1,1); LCD_STR(star);
}
```

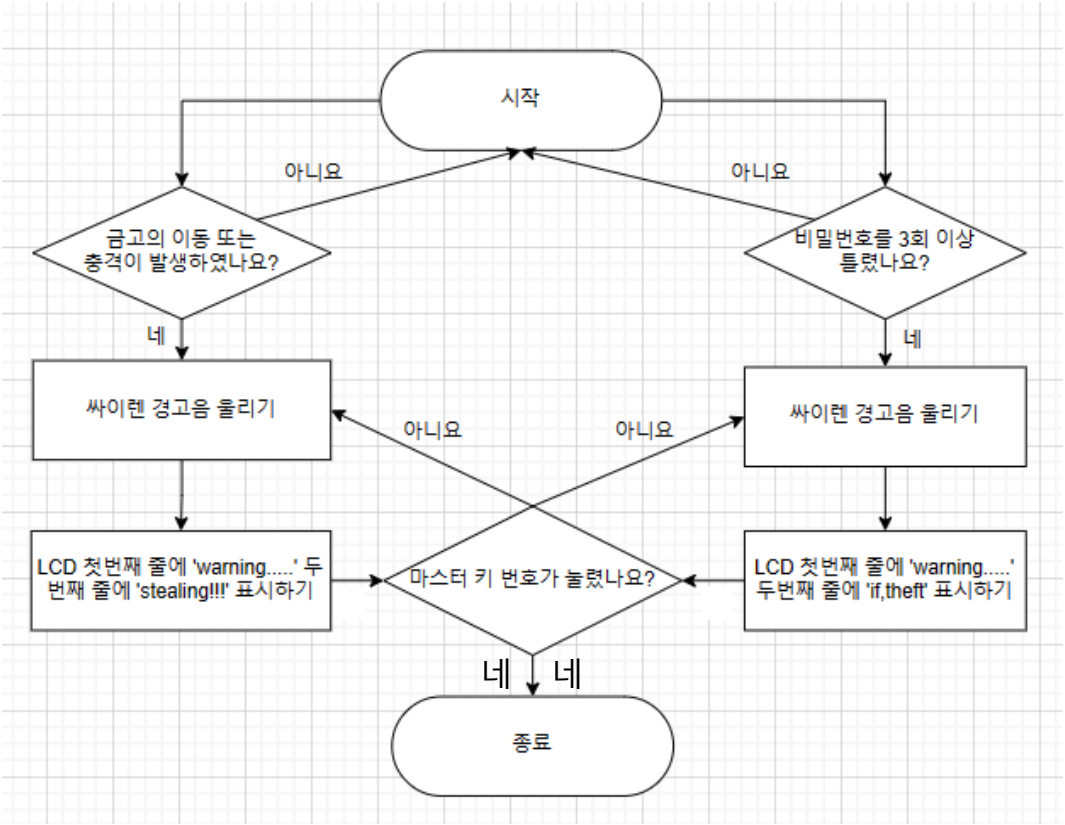


6-1번

비밀번호 3번 이상 틀릴시 경고하기

6번 FLOW CHART

외부 순차도



내부 순차도 (마스터키 진입 순차도)