1 목차

1 텀 프로젝트의 내용 정리

1. 본 프로젝트에서는 다음의 디지털 금고 기능을 구현하도록 한다.

① 초기 상태에서는 FND 및 LCD 에 아무것도 출력 하지 않는다.

② 키패드의 # 버튼을 누르면 LCD 에 아래와 같은 초기 상태를 표시한다.

	P	A	S	S	W	O	R	D						
	*	*	*	*	*	*	*	*	*	*	*	*	*	*

③ 키패드의 숫자 6~14 자리를 누르고 * 버튼을 눌러 비밀번호 일치하는지 확인한다.

(초깃값은 학생의 학번 이다.)

-> 키패드 눌림시 모든 키값은 FND 에 출력후 사용자가 확인하도록 한다.

-> 키 값은 눌린 순서대로 LCD 에 아래와 같이 출력한다.

-> 이때 키패드를 누를 때마다 부저로 '시'를 출력한다.

-> 킷값 모두 입력 후 * 누르면 ④번 순서로 진행한다.

	P	A	S	S	W	O	R	D						
	1	2	3	4	5	6	7	8	9	0	*	*	*	*

④ 비밀번호가 일치하는지 확인

1) 일치 시 부저로 "도 미 솔 도(한음 올림)"를 재생하고 서보 모터를 180 도로 움직여 도어 잠금 해제 후 아래와 같이 LCD 에 출력한다.

	D	O	O	R		O	P	E	N					

2) 불일치 시 부저로 "라 라 라"를 재생하고 LCD 에 다음과 같이 출력한다.

	P	A	S	S	W	O	R	D						
		e	r	r	o	r	.	.	.	.	.	.	.	

3) 부저를 출력한 이후 초기상태로 복귀한다.

⑤ 도어 잠금은 * 버튼 혹은 PORTD 의 1 번 스위치가 3 초 이상 눌리면 "솔 파 미 레"를 재생하고 서보 모터를 움직여 도어 잠금(0 도로 이동)

-> 잠금 후 초기상태로 복귀함.

⑥ 비밀번호 설정은 외부 PORTD 의 0 번 스위치(인터럽트 적용)를 눌러 설정 한다.

1) PORTD 의 0 번에 연결된 스위치를 누름(인터럽트 이용)

2) LCD 에 아래와 같이 출력하고 비밀번호를 누르는 대로 LCD 에 출력 한다.

	P	A	S	S	W	O	R	D		S	E	T		
	-	-	-	-	-	-	-	-	-	-	-	-	-	-

-> 비밀번호 기재시 LCD 에 눌린대로 출력 하도록 한다.(③번 참조)

3) 비밀번호를 모두 기입후 다시 PORTD 의 0 번에 연결된 스위치를 누르면 부저로 "도 레 미"를 재생하고 저장한다.

4) 저장이 완료 되면 LCD 에 다음과 같이 출력한다.

	P	A	S	S	W	O	R	D		S	E	T		
		-	S	U	C	C	E	S	S	-				

5) 3 초 이후 초기상태로 복귀한다.

2. 구현한 디지털 금고의 보안을 위하여 다음과 같은 보안대책을 강구한다.

① 비밀번호를 3 회 이상 틀릴 시 싸이렌 경고음을 부저로 울리고, LCD 에 다음과 같이 출력한다.

	w	a	r	n	i	n	g	.	.	.	.	.		
		i	f	,	t	h	e	f	t					

② 디지털 출력 진동 센서를 활용하여 금고의 이동 또는 충격이 발생을 검출 하고, 검출 되었을 경우 싸이렌 경고음을 부저로 울리고, LCD 에 다음과 같이 출력한다.

	w	a	r	n	i	n	g	.	.	.	.	.		
		s	t	e	a	l	i	n	g	!	!	!		

③ 상기 보안 동작은 마스터 키 번호를 누르기 전까지 무한 반복 하며, 마스터 키 번호를 눌러 키 값이 일치 하면 정상 모드로 복귀 한다.

-> 마스터 키 값을 누르기 위한 모드 진입은 *, # 버튼을 동시에 3 초 이상 누른다.

	M	A	S	T	E	R		M	O	D	E			
	*	*	*	*	*	*	*	*	*	*	*	*	*	*

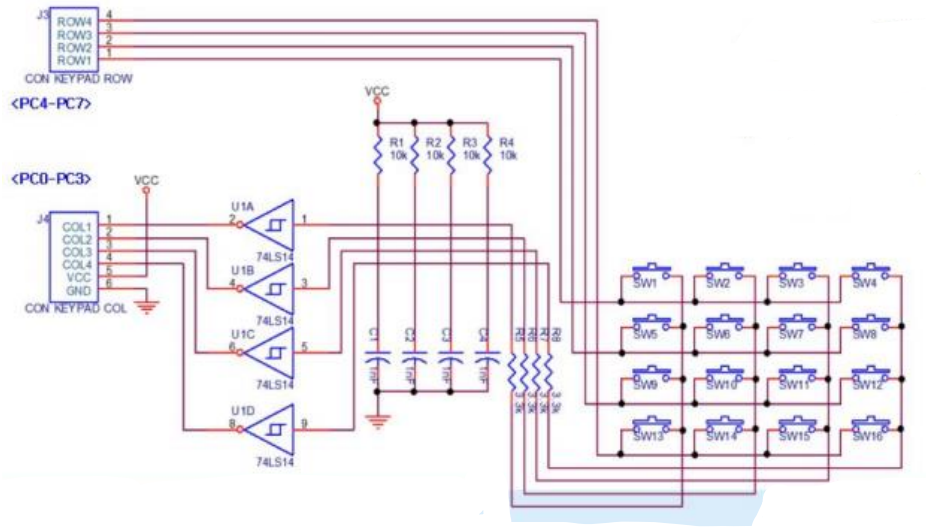
-> 키 값을 누르면 아래와 같이 * 자리에 실제 누른 키 값을 출력 한다.

	M	A	S	T	E	R		M	O	D	E			
	4	3	*	4	7	#	2	0	2	1				

2 핵심 내용 도출 및 구현 방안

2.1 KEYPAD 인터페이스 구현

키보드 인터페이스를 구현하기 위해 행 선택 출력과 열 선택에 의한 키 입력을 이용하여 주기적으로 키보드를 스캔하는 방식인 키 스캔 방식을 사용하여 키 데이터를 읽어들이도록 한다. 행(row)를 순차적으로 선택할 수 있도록 PORTC 의 상위 니블 PC4~PC7 비트를 1 비트씩 '0'으로 순차적인 출력을 하고, 열(col)에 해당하는 PORTC 의 하위 니블 PC0~PC3 비트를 한번에 모두 읽으면 키 매트릭스의 모든 키 정보를 읽을 수 있다.



키 입력 값 / 키 스캔 라인	PC0(누름) X = 1	PC1(누름) X = 2	PC2(누름) X = 3	PC3(누름) X = 4
PC7~PC4 = 1110, Y = 0	SW1	SW2	SW3	SW4
PC7~PC4 = 1101, Y = 1	SW5	SW6	SW7	SW8
PC7~PC4 = 1011, Y = 2	SW9	SW10	SW11	SW12
PC7~PC4 = 0111, Y = 3	SW13	SW14	SW15	SW16

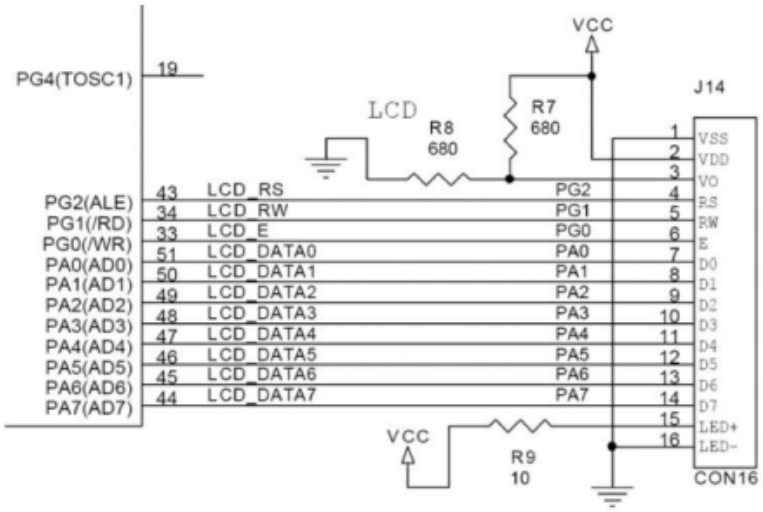
키패드의 정보를 읽는 과정에서 채터링이 발생하므로 디바운싱 프로그램을 통해 채터링 현상을 제거해야 한다. 소프트웨어적으로 바운싱 현상을 제거하기 위해 일단 스위치 정보가 읽히면 바운싱이 종료된 후에 스위치 정보를 읽으면 된다. 그 흐름도는 교재의 그림 5.22 를 참고한다.

2.2 LCD 구동

LCD 의 정해진 위치에 문자를 표시하기 위해 명령 레지스터에 DDRAM 주소 설정 명령을 이용하여 위치를 설정한다. 표시하고자 하는 문자를 문자 발생기 ROM 상에 있는 문자를 호출하여 데이터 레지스터에 기록한다.

문자(열)을 이용하여 LCD 에 표시하는 함수

```
➢ void LCD_CHAR(unsigned char c)     // 하나의 문자를 LCD 에 표시하는 함수
➢ void LCD_STR(unsigned char *str)    //문자열을 LCD 에 표시하는 함수
```

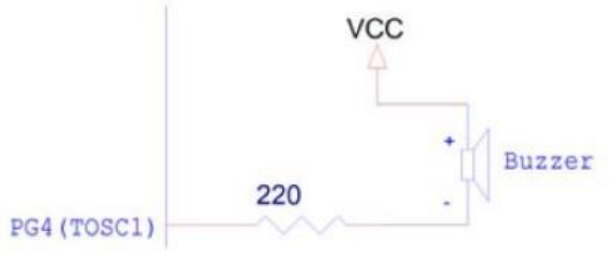


2.3 BUZZER 의 활용

부저란 원하는 주파수로 부저에 인가하는 전압을 ON/OFF 하여 소리를 발생하는 소자이다.

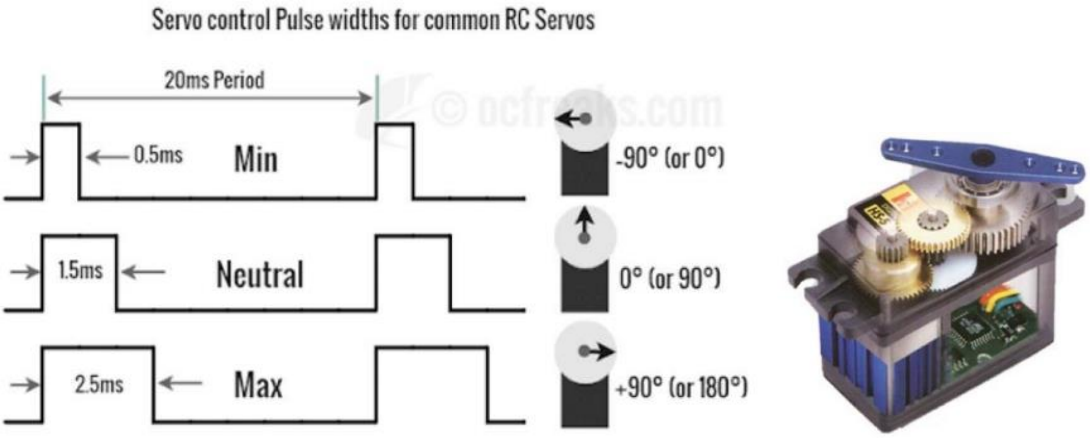
원하는 주파수의 주기와 듀티비를 갖는 구형파의 파형을 부저에 입력하면 음과 음의 크기가 변하게 된다.

```
#define DO 1908
#define RE 1700
#define MI 1515
#define FA 1432
#define SOL 1275
#define LA 1136
#define SI 1012
#define DDO 956
```



2.4 RC 서보 모터

물체의 위치, 방위, 자세, 회전 속도 등을 제어량으로 하고 목표치의 변화에 뒤따르도록 구성된 자동제어계를 서보 기구라 한다. RC servo motor 는 PWM 을 이용하여 각도를 제어할 수 있는 모터이다.



3 핵심 내용별 구현을 위한 프로그램 코드

3.1 KEYPAD

```
//-------------------------------------KEYPAD---------------------------------
unsigned char KeyScan(void) //4x4 키패드 스캔 함수, 출력값은 10 진수 1~16
{
    // 상위 니블(4 비트)을 스위칭(연속적으로 돌아가면서)하면서 출력
    unsigned int key_scan_line = 0xEF;

    // 키 스캔 라인 변경을 위한 반복문 인자, 키 매트릭스 열의 입력 값
    unsigned char key_scan_loop = 0, getPinData=0;

    // 키 스캔 반복문
    for(key_scan_loop=0; key_scan_loop<4; key_scan_loop++)
    {
        // 키 매트릭스의 스캔 라인 설정을 위한 PORTC 출력값
        PORTC = key_scan_line;
        delay_us(1);
        // 키 매트릭스의 열 값 취득
        // C 포트의 하위 니블(4 비트)RKQT, 74LS14 사용으로 인한 신호 반전
        getPinData = PINC & 0x0f;
        if(getPinData != 0x00)
        {
```

```
            switch(getPinData)
            {
                case 0x01: key_num = key_scan_loop*4 + 1; break;  //1110 의 반전
                                                                  입력이 맞는 경우
                case 0x02: key_num = key_scan_loop*4 + 2; break;
                case 0x04: key_num = key_scan_loop*4 + 3; break;
                case 0x08: key_num = key_scan_loop*4 + 4; break;
            }
            return key_num;
        }
        key_scan_line = (key_scan_line << 1);       // Init_data(0xEF) 를 시프트함
        delay_us(1);
    }
}


unsigned char Key_data_trans(unsigned char New_key_data)   // 4x3 키패드로 입력하기
위한 함수
{
    unsigned char key_num = 0;
    if(New_key_data%4 != 0)
    {
        key_num = (New_key_data/4)*3 + (New_key_data%4);
        switch(key_num)
        {
            case 10: key_num = FND_Star; break;
            case 11: key_num = FND_Null; break;
            case 12: key_num = FND_Sharp;
                     sharp_flag = 1;
                     break;
        }
    }
    else
        key_num = (New_key_data/4)+9;
    return key_num;
}
```

3.2 LCD

```
void LCD_CHAR(unsigned char c)      // 한 문자 출력
{
    LCD_Data(c);
    delay_ms(1);
}
```

```
void LCD_STR(unsigned char *str)    // 문자열 출력
{
    while(*str != 0)
    {
        LCD_CHAR(*str);
        str++;
    }
}
```

```
void LCD_pos(unsigned char row, unsigned char col)  // LCD 포지션 설정
{
    LCD_Comm(0x80 | ((row*0x40) + col) );
}
```

3.3 BUZZER

```
//-----------------------------------BUZZER---------------------------------
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 = 50000 / time;
    for(i=0; i<tim; i++)
    {
        PORTG |= (1<<PORTG4); //buzzer on, PORTG 의 4 번 핀 on (out1)
        myDelay_us(time);
        PORTG &= ~(1<<PORTG4); //buzzer off, PORTG 의 4 번 핀 off(out0)
        myDelay_us(time);
    }
    PORTG &= ~(1<<PORTG4);    //buzzer off, PORTG 의 4 번 핀 off (out 0)

}
```

3.4 SERVO MOTOR

```
//-----------------------------------MOTOR-----------------------------------
interrupt [TIM2_COMP] void timer2_cmp(void)
{
    T2_DUTY_TIME_cnt_us += 100;
    T2_CYCLE_TIME_cnt_us += 100;

    if(T2_DUTY_TIME_cnt_us <= T2_DUTY_TIME_us)
    {SERVO_PIN_HIGH();}
    else
    {SERVO_PIN_LOW();}

    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, 1024 분주
    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);
}
```

4 구현을 위한 전체 프로그램 코드

```
#include <mega128.h>
#include <delay.h>

#define LCD_WDATA PORTA     // LCD 데이터 포트 정의
#define LCD_WINST PORTA
#define LCD_CTRL  PORTG     // LCD 제어포트 정의
#define LCD_EN    0
#define LCD_RW    1
#define LCD_RS    2

#define FND_Null    0
#define FND_Star    14
#define FND_Sharp   15

// 음계 define
#define DO 1908
#define RE 1700
#define MI 1515
#define FA 1432
#define SOL 1275
#define LA 1136
#define SI 1012
#define DDO 956

// SERVO PIN 정의
#define SERVO_PIN_LOW()     PORTF &= ~(1<<PORTF3)
#define SERVO_PIN_HIGH()    PORTF |= (1<<PORTF3)

int i = 0;
int j = 0;
int k =0 ;
int security = 0;
int sharp_flag = 0;
int right = 0;
int wrong = 0;
unsigned char input_pw[16];
unsigned char pw[16] = "2019146018";
unsigned char Key_off_flag = 0;
unsigned char New_key_data = 0;
unsigned int 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, 0x0f};
unsigned char key_num = 0;
unsigned int buf_seg[4] = {FND_Null, FND_Null, FND_Null, FND_Null};

unsigned int T2_DUTY_TIME_us;
unsigned int T2_CYCLE_TIME_us;
unsigned int T2_DUTY_TIME_cnt_us;
unsigned int T2_CYCLE_TIME_cnt_us;
```

```
void PortInit (void)
{
    // LCD
    DDRA = 0xFF;
    DDRG = 0x0F;
    // FND
    DDRE = 0xf0;    // FND Sel
    DDRB = 0xff;    // FND Data Line
    // KEYPAD
    DDRC = 0xf0;    //상위 4 비트 Row
    PORTC = 0xff;   // 포트 초기화
    PORTE = Port_fnd[4];    //ALL FND OFF
    // BUTTON
    DDRD = 0x00;
    // MOTOR
    DDRF |= (1<<PORTF3);
    // BUZZER
    DDRG |= (1<<PORTG4);
    PORTG &= ~(1<<PORTG4);
}

//-----------------------------------LCD---------------------------------
void LCD_Data(unsigned char ch)     // LCD_DR 에 데이터 출력
{
    LCD_CTRL |= (1 << LCD_RS);      // RS=1, =0 으로 데이터 쓰기 사이클
    LCD_CTRL &= ~(1 << LCD_RW);
    LCD_CTRL |= (1 << LCD_EN);      // LCD Enavle
    delay_us(50);
    LCD_WDATA = ch;                 // 데이터 출력
    delay_us(50);
    LCD_CTRL &= ~(1 << LCD_EN);     // LCD Disable
}

void LCD_Comm(unsigned char ch)     // LCD IR 에 명령어 쓰기
{
    LCD_CTRL &= ~(1 << LCD_RS);     // RS==0 으로 명령어 쓰기 사이클
    LCD_CTRL &= ~(1 << LCD_RW);
    LCD_CTRL |= (1 << LCD_EN);      // LCD Enable
    delay_us(50);
    LCD_WINST = ch;                 // 명령어 쓰기
    delay_us(50);
    LCD_CTRL &= ~(1 << LCD_EN);     // LCD Disable
}

void LCD_CHAR(unsigned char c)      // 한 문자 출력
{
    LCD_Data(c);
    delay_ms(1);
}

void LCD_STR(unsigned char *str)    // 문자열 출력
{
```

```
    while(*str != 0)
    {
        LCD_CHAR(*str);
        str++;
    }
}

void LCD_pos(unsigned char row, unsigned char col)  // LCD 포지션 설정
{
    LCD_Comm(0x80 | ((row*0x40) + col) );
}

void LCD_Init(void)     // LCD 초기화
{
    LCD_Comm(0x30);     // 초기화 Set,
    delay_us(4100);     // 4.1ms 지연
    LCD_Comm(0x30);     // 초기화 Set,
    delay_us(100);      // 100us 지연
    LCD_Comm(0x30);     // 초기화 Set,
    delay_us(100);      // 100us 지연
    LCD_Comm(0x38);     // 초기화 Set, 데이터길이 8bit, 표시라인 2 행 사용
    delay_us(1000);     // 명령을 처리하는데 최소 40us 지연이 발생하기에 여유를 고려
    LCD_Comm(0x0c);     // Display ON, Cursor OFF, Blink OFF
    delay_us(1000);     // 40us 이상을 기다림
    LCD_Comm(0x01);     // LCD Clear
    delay_us(2000);     // 1.64ms 이상을 기다림
    LCD_Comm(0x06);     // Cursor Entry Mode Set, 표시위치 +1 씩 증가
    delay_us(1000);     // 40us 이상을 기다림
}

//-----------------------------------SEGMENT---------------------------------
void Print_Segment(unsigned int* seg_value)
{
    PORTE = Port_fnd[0];
    PORTB = Port_char[seg_value[0]];
    delay_ms(1);

    PORTE = Port_fnd[1];
    PORTB = Port_char[seg_value[1]];
    delay_ms(1);

    PORTE = Port_fnd[2];
    PORTB = Port_char[seg_value[2]];
    delay_ms(1);

    PORTE = Port_fnd[3];
    PORTB = Port_char[seg_value[3]];
    delay_ms(1);
}
```

```
//-----------------------------------KEYPAD-------------------------------
unsigned char KeyScan(void) //4x4 키패드 스캔 함수, 출력값은 10 진수 1~16
{
    // 상위 니블(4 비트)을 스위칭(연속적으로 돌아가면서)하면서 출력
    unsigned int key_scan_line = 0xEF;

    // 키 스캔 라인 변경을 위한 반복문 인자, 키 매트릭스 열의 입력 값
    unsigned char key_scan_loop = 0, getPinData=0;

    // 키 스캔 반복문
    for(key_scan_loop=0; key_scan_loop<4; key_scan_loop++)
    {
        // 키 매트릭스의 스캔 라인 설정을 위한 PORTC 출력값
        PORTC = key_scan_line;
        delay_us(1);
        // 키 매트릭스의 열 값 취득
        // C 포트의 하위 니블(4 비트)RKQT, 74LS14 사용으로 인한 신호 반전
        getPinData = PINC & 0x0f;
        if(getPinData != 0x00)
        {
            switch(getPinData)
            {
                case 0x01: key_num = key_scan_loop*4 + 1; break;  //1110 의 반전
                                                                    입력이 맞는 경우
                case 0x02: key_num = key_scan_loop*4 + 2; break;
                case 0x04: key_num = key_scan_loop*4 + 3; break;
                case 0x08: key_num = key_scan_loop*4 + 4; break;
            }
            return key_num;
        }
        key_scan_line = (key_scan_line << 1);       // Init_data(0xEF) 를 시프트함
        delay_us(1);
    }
}

unsigned char Key_data_trans(unsigned char New_key_data)   // 4x3 키패드로 입력하기
                                                              위한 함수
{
    unsigned char key_num = 0;
    if(New_key_data%4 != 0)
    {
        key_num = (New_key_data/4)*3 + (New_key_data%4);
        switch(key_num)
        {
            case 10: key_num = FND_Star; break;
            case 11: key_num = FND_Null; break;
            case 12: key_num = FND_Sharp;
                     sharp_flag = 1;
                     break;
        }
    }
    else
```

```
        key_num = (New_key_data/4)+9;
    return key_num;
}

//-----------------------------------BUZZER---------------------------------
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 = 50000 / time;
    for(i=0; i<tim; i++)
    {
        PORTG |= (1<<PORTG4); //buzzer on, PORTG 의 4 번 핀 on (out1)
        myDelay_us(time);
        PORTG &= ~(1<<PORTG4); //buzzer off, PORTG 의 4 번 핀 off(out0)
        myDelay_us(time);
    }
    PORTG &= ~(1<<PORTG4);    //buzzer off, PORTG 의 4 번 핀 off (out 0)

}

//-----------------------------------MOTOR----------------------------------
interrupt [TIM2_COMP] void timer2_cmp(void)
{
    T2_DUTY_TIME_cnt_us += 100;
    T2_CYCLE_TIME_cnt_us += 100;

    if(T2_DUTY_TIME_cnt_us <= T2_DUTY_TIME_us)
    {SERVO_PIN_HIGH();}
    else
    {SERVO_PIN_LOW();}

    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 - 2 초 설정 및 PC PWM 듀티비 설정
    TIMSK |= (1<<OCIE2);                // 출력비교 인터럽트 2 허가
    TCCR2 = (1<<WGM21) | (2<<CS20);     // CTC 모드, 1024 분주
```

```
    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);
}

//-----------------------------------INTERRUPT---------------------------------
interrupt [EXT_INT0] void ext_int0_isr(void)
{   // PORTD 0 번 스위치 눌렸을 때 발생하는 외부 인터럽트 (비밀번호 재설정)

    if(k%2 == 0)
    {
        LCD_pos(0,1);
        LCD_STR("PASSWORD SET   ");
        LCD_pos(1,1);
        LCD_STR("---------------");
        LCD_pos(1,1);

        while(1)
        {
            New_key_data = KeyScan();
            Print_Segment(buf_seg);

            if(New_key_data)
            {
                SSound(SI);
                delay_ms(100);
                key_num = Key_data_trans(New_key_data);
                if(Key_off_flag)
                {
                    if((key_num>=0) && (key_num<=9))
                    {
                        input_pw[i] = key_num + '0';
                        LCD_CHAR(input_pw[i]);
                        i++;
                        if(i>=14)
                        {
                            i = 0;
                            LCD_pos(1,1);
                        }
                    }
                    Key_off_flag = ~Key_off_flag;
                }
```

```
                else
                {
                    buf_seg[0] = key_num;
                }
            }
            else if (key_num == 14)
            {
                break;
            }
            else
            {
                Key_off_flag = 0xff;
            }
        }
    }

    if(k%2 == 1)        // D0 번을 두번째 눌렀을 때
    {
        SSound(DO);
        delay_ms(100);
        SSound(RE);
        delay_ms(100);
        SSound(MI);
        delay_ms(100);

        LCD_pos(0,1);
        LCD_STR("PASSWORD SET  ");
        LCD_pos(1,1);
        LCD_STR(" -SUCCESS-      ");
        LCD_pos(1,1);
    }

    for(i=0;i<16;i++)
    {
        pw[i] = input_pw[i];
    }

    k++;
}


void main(void)
{
    PortInit();                 // 포트 설정
    LCD_Init();                 // LCD 포트 설정
    Init_TimerINT();            // 타이머 초기화
    T2_DUTY_TIME_us = 1500;     // PWM 듀티 폭 설정
    T2_CYCLE_TIME_us = 20000;   // PWM 주기 설정

    //INT0 Falling Edge 감지 설정 (ISC01 : 1, ISC00 : 0)   //누를 때 동작
    EICRA &= ~(3<<ISC00);                   // 1111 1110
    EICRA |= (1<<ISC01) | (0<<ISC00);       // 0000 0010
```

```
EIMSK |= (1<<INT0);       //INT0 활성화
SREG |= (1<<SREG_I);     //전체 인터럽트 허가

while(1)
{
    New_key_data = KeyScan();
    Print_Segment(buf_seg);

    if(New_key_data)        // 새로운 값이 keypad 에 들어왔음
    {
        SSound(SI);
        delay_ms(100);
        key_num = Key_data_trans(New_key_data);

        if(Key_off_flag)
        {
            if(sharp_flag == 1 && (key_num>=0) && (key_num<=9))
            {
                // 0 ~ 9 번호를 눌렀을 때
                input_pw[i] = key_num + '0';
                LCD_CHAR(input_pw[i]);
                i++;

                if(i>=14)    // 비밀번호는 최대 14 자리
                {
                    i = 0;
                    LCD_pos(1,1);
                }
            }
            else if (key_num>9)
            {
                if(key_num == 15)       // # 눌렀을 때
                {
                    for(j=0;j<16;j++)
                    {
                        input_pw[j] = 0;    // 입력했던 input_pw 초기화
                    }

                    // 초기 화면 출력
                    LCD_pos(0,1);
                    LCD_STR("PASSWORD       ");
                    LCD_pos(1,1);
                    LCD_STR("**************");
                    LCD_pos(1,1);
                }

                else if(key_num == 14)  // * 눌렀을 때
                {
                    // 비밀번호 일치 확인
                    for(i=0;i<16;i++)
                    {
                        if(pw[i] == input_pw[i])
```

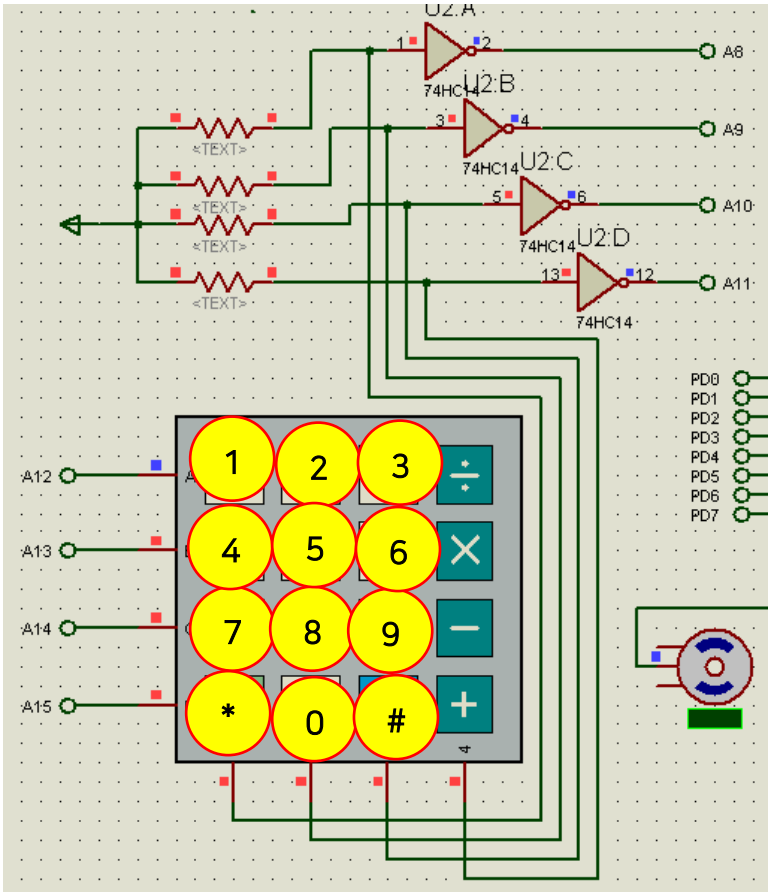
```
                    {
                        // 비밀번호 일치 시 right flag 1
                        right = 1;
                    }
                    else
                    {   // 비밀번호 불일치시
                        wrong = 1;
                        right = 0;
                        break;
                    }
                }
            }
        }
        Key_off_flag = ~Key_off_flag;
    }
    else
    {
        buf_seg[0] = key_num;
    }
}
else
{
    Key_off_flag = 0xff;
}


if(right == 1)
{
    // ④ - 1) 비밀번호 일치 시
    LCD_pos(0,1);
    LCD_STR("DOOR OPEN       ");
    LCD_pos(1,1);
    LCD_STR("                ");
    SSound(DO);
    delay_ms(100);
    SSound(MI);
    delay_ms(100);
    SSound(SOL);
    delay_ms(100);
    SSound(DDO);
    SetServoDeg(180);
    delay_ms(3000);
    // ④ - 3) 초기상태 복귀
    LCD_pos(0,1);
    LCD_STR("                ");
    LCD_pos(1,1);
    LCD_STR("                ");

    right = 0;
}

if(wrong == 1)
```

```
{
    // ④ - 2) 비밀번호 불일치 시
    LCD_pos(0,1);
    LCD_STR("PASSWORD        ");
    LCD_pos(1,1);
    LCD_STR(" error.....        ");
    SSound(LA);
    delay_ms(100);
    SSound(LA);
    delay_ms(100);
    SSound(LA);
    delay_ms(3000);
    // ④ - 3) 초기상태 복귀
    LCD_pos(0,1);
    LCD_STR("                ");
    LCD_pos(1,1);
    LCD_STR("                ");
    security++;  // 보안대책 2 - ① 을 위한 security 1 증가
    wrong = 0;

}

if(security>=3)
{
    // 보안대책 2 - ① ) 3회 이상 틀릴 시 경고음
    LCD_pos(0,1);
    LCD_STR("warning...        ");
    LCD_pos(1,1);
    LCD_STR(" if.theft         ");
    SSound(DDO);
    delay_ms(50);
    SSound(SOL);
    delay_ms(50);

    while(key_num==10)  // 2 - ③) 마스터키모드
    {
        // 키패드의 ÷ 버튼 누를 시 마스터키모드 진입
        delay_ms(100);
        LCD_pos(0,1);
        LCD_STR("MASTER MODE        ");
        LCD_pos(1,1);
        LCD_STR("***************");
    }
}

if(!(PIND & (1<<PORTD1)))
{
    // 1 - ⑤ ) PIND1 번 스위치로 도어 잠금
    delay_ms(3000);
    SSound(SOL);
    delay_ms(100);
    SSound(FA);
```

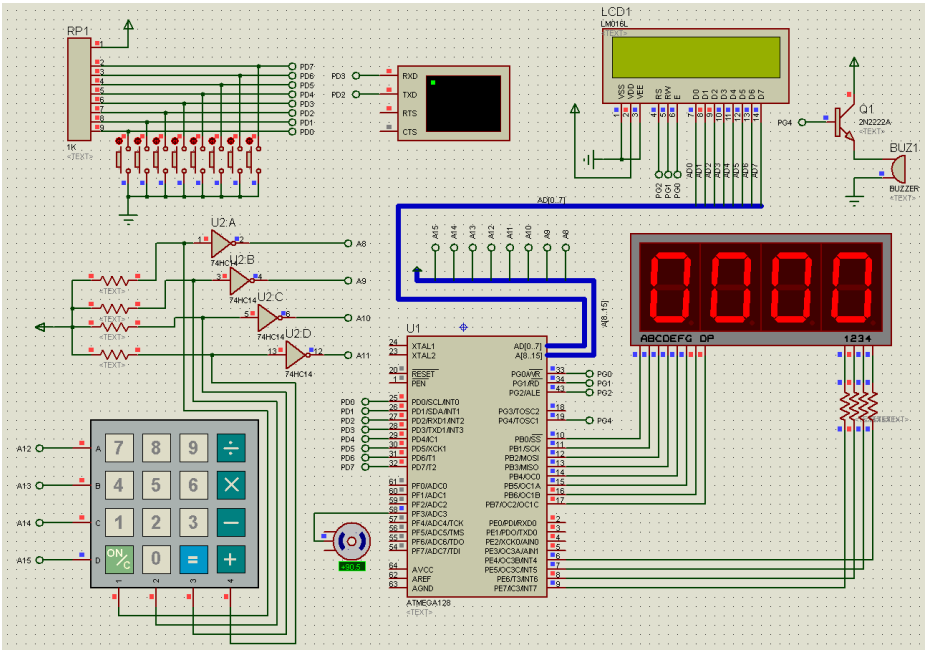
```
            delay_ms(100);
            SSound(MI);
            delay_ms(100);
            SSound(RE);
            SetServoDeg(0);
            delay_ms(3000);
            //초기상태 복귀
            LCD_pos(0,1);
            LCD_STR("               ");
            LCD_pos(1,1);
            LCD_STR("               ");
        }

    }
}
```

5 구현/동작 결과

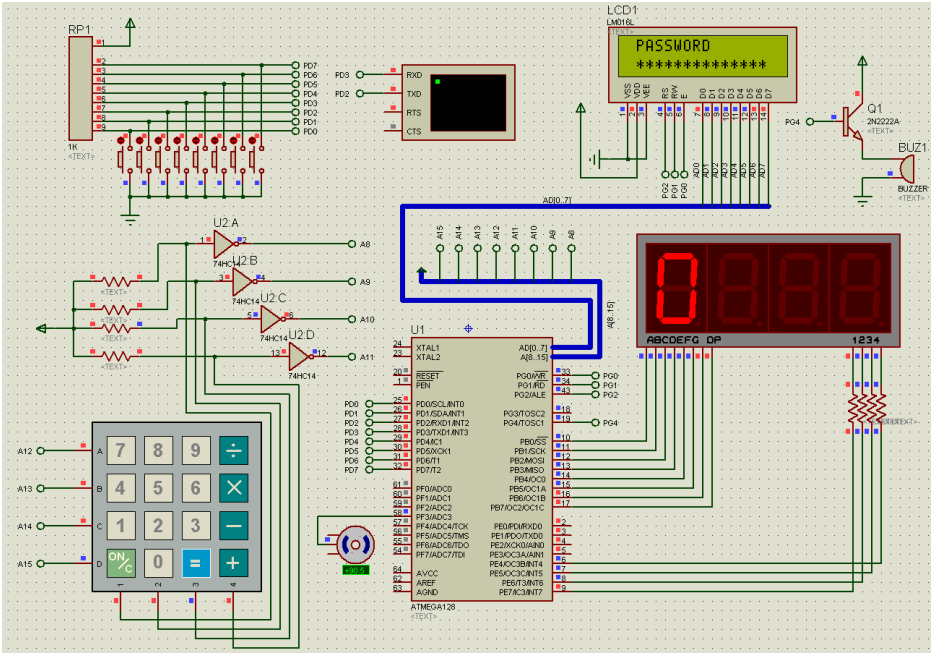
시뮬레이션 상에서의 결선과 코드를 고려하였을 때, 다음 그림과 같이 키패드를 읽도록 한다.



5.1 초기 상태 출력

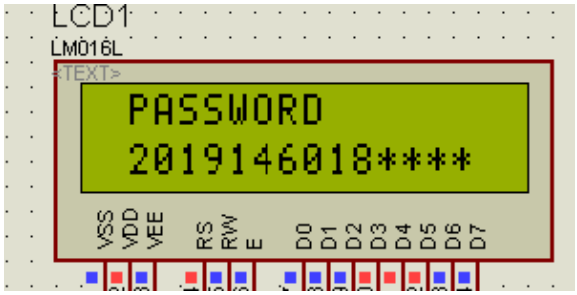


가장 처음 proteus 를 실행시켰을 때의 화면이다.

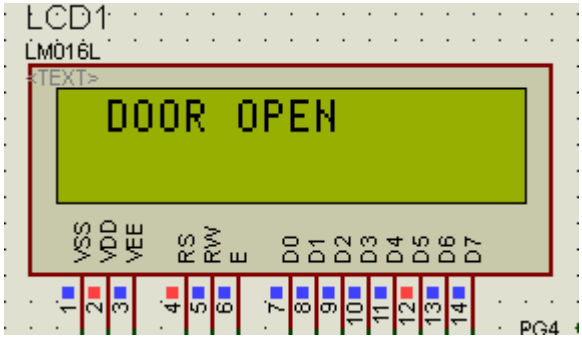


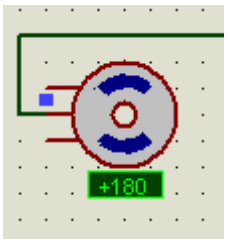
버튼을 누르면 위와 같이 출력된다.

5.2 비밀번호 일치 시



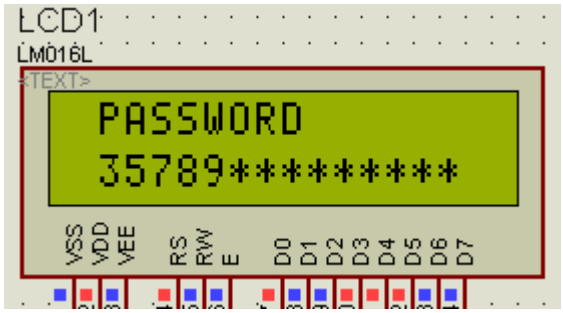
← 이처럼 비밀번호를 올바르게 입력하게 되면,

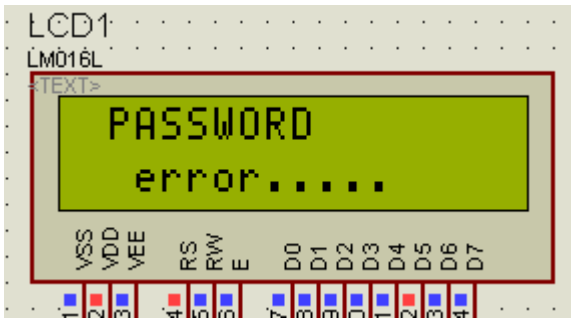




위와 같이 LCD 에 “DOOR OPEN”을 출력하고 서보모터가 180 도 돌아가서 잠금 해제 된다.

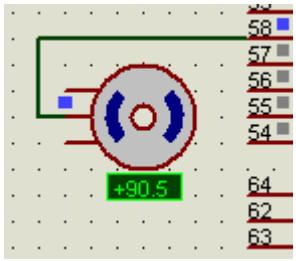
5.3 비밀번호 불일치 시



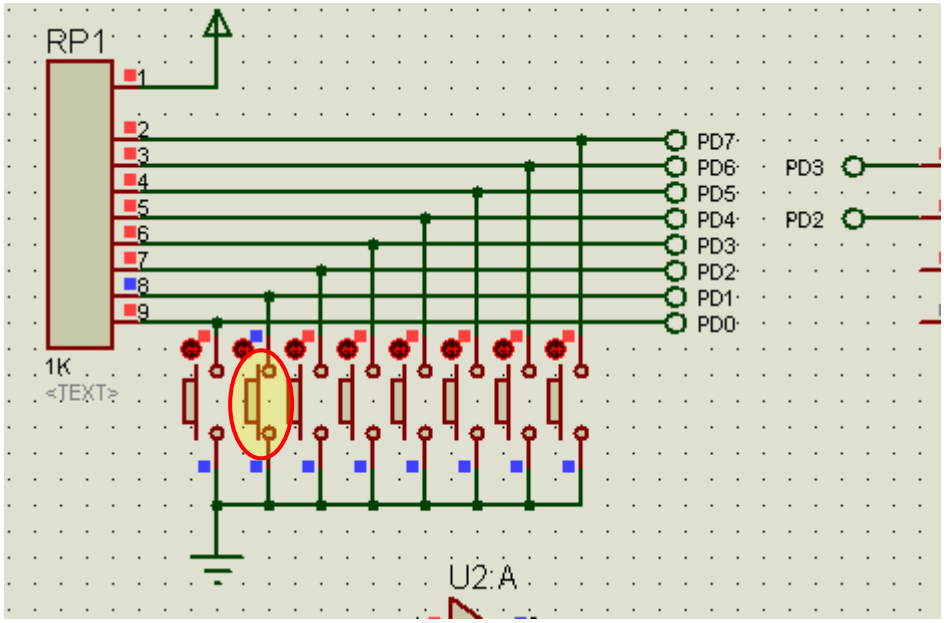


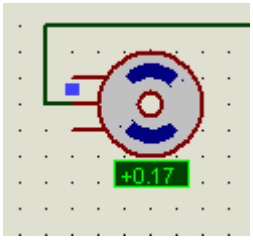
위의 사진처럼 비밀번호를 올바르게 입력하지 않았을 시에는 LCD 화면에 “PASSWORD”, “error....” 를 출력한다.

5.4 도어 잠금



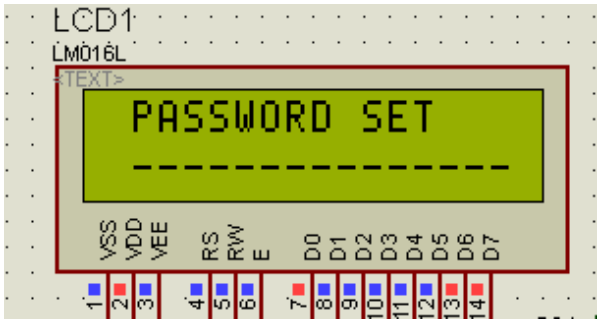
↩ 왼쪽 그림과 같이 서보모터가 90 도인 상태에서,

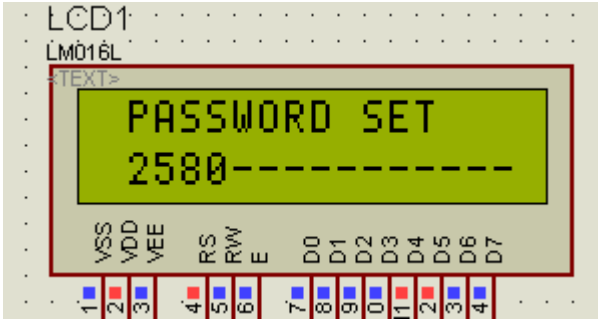




PD1 번 버튼을 3 초 이상 누르면 서보모터가 0 으로 돌아가서 도어 잠금 상태가 된다.

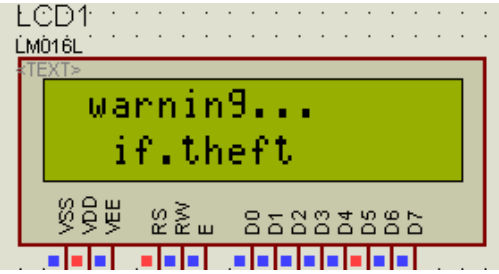
5.5 비밀번호 설정 (미완성)





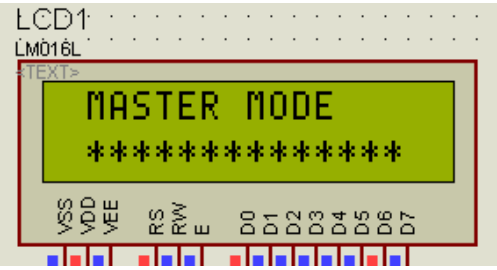
인터럽트를 사용하여 1 - ⑥ 의 조건대로 PD0 번 버튼을 눌렀을 때 비밀번호 재설정을 하고자 하였으나, 위 그림처럼 "PASSWORD SET", "----------"를 출력하는 것과 입력한 숫자를 순서대로 출력하는 부분까지만 완성하였다.

5.6 싸이렌 경고음



비밀번호 불일치가 3 회 이상 발생하면 위와 같은 화면이 출력되면서 싸이렌 경고음이 울린다.

5.7 마스터키 모드 (미완성)



위의 싸이렌 경고음이 울리고 난 후, 키패드의 ÷ 버튼을 누르면 마스터키 모드에 진입하도록 해 보았다. 하지만 진입 한 후 정상 모드로 복귀하는 부분을 완성하지 못하였다.

6 고찰

6.1 구현 내용별 참조 사항

- 키패드 인터페이스, BUZZER
  - ➢ 교재 219 ~ 234 p, 강의자료 I/O 포트의 활용 – KEYPAD & BUZZER 참고
- 타이머를 활용한 SERVO Motor 의 구동, LCD 제어,
  - ➢ 강의자료 참고

6.2 검토 및 개선점

수많은 시행착오를 겪은 후 위와 같은 결과물을 완성했다. 주어진 문제 모두를 해결하지는 못하였다. 모든 코드는 교재나 강의자료에 있는 코드를 응용한 것이지만, 인터럽트를 사용하는 부분에 있어서 많이 부족했던 것이라는 생각이 든다. 또한 사용자가 확인할 수 있도록 FND 출력을 하는 부분에서 어려움을 겪고 결국 완전한 결과를 만들어내지 못했는데, FND 관련 함수들을 잘 해야겠다고 생각했다.