```
/*
 * door_lock_system.c
 *
 * Created: 2020-02-29 오후 2:48:09
 * Author : chw91
 */

/*
TIM0 : RGB (62.5Hz) : CTC
TIM1 : SERVO (50Hz) : FAST PWM
TIM3 : BUZZER  : CTC
UART1 : BT (9600bps)
UART0 : Touch Sensor (38400bps)


>> 20.03.04 수정
BUZZER : TIM0 > TIM3 수정
RGB LED : TIM3 > TIM0 수정

GPIOA : LED (RGB? >> PWM을 통해 신호를 주면 확실하긴 함 )
>>일단 RGB는 A포트던 C포트던 연결해놓을 것임.
TIM0 비교매치 인터럽트 루틴에서 모두 처리해줄 계획. 만약 루틴에서 낭비되는 시간이 너무 길다고
판단되면 포기. >> 괜찮음.

>> 20.03.13 수정
TIMER0에 대한 비교매치 인터럽트는 setup에서 활성화 시키도록 함.
TIMER3에 대한 CTC 모드의 주파수 대역 찾는 것은 ICR으로 모두 찾아야 함. OCR값을 바꿔주면
*/

/**16MHz일 경우
#define F_CPU 16000000UL
*/
/**8MHz일 경우*/
#define F_CPU 8000000UL


#include <avr/io.h>
#include <string.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h>


//#define BAUD 9600


//-----------------DEFINE-----------------//

#define MYUBRR(BAUD) (F_CPU/16/(BAUD)-1)
#define SUCCESS 1
#define FAILURE 0

#define PRESS '1'
```

```
#define RELEASE '0'

#define _60Hz 16667
#define _120Hz 8334
#define _240Hz 4167
#define _480Hz 2084
#define _960Hz 1042
#define _1920Hz 521
#define _500Hz 2000
#define C 1675
#define D 1706
#define E 1511
#define F 1175
#define G 1250
#define A 1125
#define B 1021
#define Cs 588


/**setSoundClip 아래 효과음 리스트 추가 요망*/
#define BUZZ_MUTE 0x00
#define BUZZ_BEEP 0x01 //터치 시
#define BUZZ_LOWBAT 0x02 //슬립모드 깰 시 배터리가 부족할 때
#define BUZZ_FAIL 0x03 //비밀번호 실패했을 때
#define BUZZ_SUCCESS 0x04 //비밀번호 성공했을 때
#define BUZZ_SLEEPOUT 0x05 //슬립모드 빠지기 1초 전
#define BUZZ_CLOSEDOOR 0x06//문 닫히고 2초 뒤
#define BUZZ_SILENCE 0x07 //무음모드
#define BUZZ_BELLIN 0x08 // 벨소리모드 들어갈 때
#define BUZZ_SIREN 0x09 //5회이상 실패했을 때

/**setStateRGB 아래 현재 상태 리스트 추가 요망*/
#define RGB_OFF 0x00 //아예 끔
#define RGB_DEFAULT 0x01 //평상시
#define RGB_LOWBAT 0x02 //배터리 부족할 때
#define RGB_FAIL 0x03 //실패
#define RGB_SUCCESS 0x04 //성공
#define RGB_SLEEPOUT 0x05
#define RGB_CLOSE_DOOR 0x06
#define RGB_SIREN 0x07
#define RGB_TOUCHED 0x08
//
#define UPCOUNT 1
#define DOWNCOUNT -1

#define MUTE_ENABLE 1
#define MUTE_DISABLE -1

//Servo Degree
#define OPEN 90
#define CLOSE 0

#define DOOR_OPEN_STATE 1
```

```
#define DOOR_CLOSE_STATE 0

#define HIGH 1
#define LOW 0

//Fail Stack
#define STACK_RANGE 5
#define OVERSTACK ((STACK_RANGE)+1)
//----------------------------------------//

//---------------VALUE&Struct-------------//
typedef unsigned char u8;
typedef uint32_t u32;
int pressed_flag[13]={0,};
int pw_i=1;
int status_flag =0; // 0 is default ???dummy???

struct{
   volatile u8 rx0Flag; // uart received flag
   volatile u8 sensorData[6];
   volatile u8 tx0Buf[50];
   volatile u8 tx0Cnt,tx0CntMax;
   volatile u8 receive_cplt_sensor;
   u8 KEY[14]; //parsed data
   u8 DataBuff[4];
   u8 PW[50];
}TOUCH;

struct{
   volatile u8 rx1Flag; // uart received flag
   volatile u8 rx1Buf[50], tx1Buf[50];
   volatile u8 rx1Cnt, tx1Cnt,tx1CntMax;
   int IdBuff;
   int CommandBuff;
   int SendBtnLoc, PaswrdFlg, MuteBtnLoc;
   //volatile u8 rxTimeoutCnt;
}BT;


struct{
   volatile u32 t_1ms;
   volatile u32 buzz_1ms;
   volatile u32 rgb_16ms;
   volatile u32 tick_tim;
   volatile u32 sleep_tim;
   volatile u32 rgb_1ms;
   volatile u32 doubleClick_tick_1ms;
   volatile u32 motor_1ms;
   volatile int wait_1s; //*

   }TICK;
volatile unsigned char RxData0, RxData1;


volatile int led_R, led_G, led_B;
volatile int led_R_buf, led_G_buf, led_B_buf;
u8 RGB_status_flag=0;
volatile int dimming_count_flag=UPCOUNT;
volatile int sleep_flag=0;
u8 music_flag=0;
int motor_flag=0,MuteModeFlag=MUTE_ENABLE, mode_change_stack=0, clickTimerON_SW=0,
```

```
wait_1s_flag=0;
int door_open_flag=DOOR_CLOSE_STATE, reedSW_state=LOW, command_door_lock=0;

int fail_stack=0;
int IsWakeup=0;
int bright;

//--------------------------------
//------------Define Function-------------//
void setup();

//TIM0
void setRGB(char _R, char _G, char _B);
//void setBright(int br);
void setStateRGB(char state);
void RGB_Drive();
//TIM3
void setOCR3A(int num);
void setOCR3B(int num);
void setOCR3C(int num);
void setICR3(int num);
/*
BUZZER
 >> CTC Toggle mode
*/
void setSoundClip(char clip);
void buzz_play();
void buzz_MUTE();
void setSoundNote(int note);
void MuteMode_toggle();

//TIM1
void setOCR1A(int num);
void setOCR1B(int num); //dummy
void setICR1(int num);
int convertServoAngle(int angle);
void setServoAngle(int angle);
void motor_drive();

/*
SERVO MOTOR
Control Period : 20ms
angle : 1~2ms  >> OCR 2000~4000
*/



//UART 0 : Touch Sensor
//bps 가급적이면 높아야 함. bps : 38400
void UART0_Init(unsigned int ubrr);
void uart0_sensor_tx_string(char * data);
void uart0_sensor_tx_char(char data);

//UART 1 : B.T.
//bps 그렇게 높을 필요 없음. 센서 데이터보단 느리면 좋음.
void UART1_Init(unsigned int ubrr);
void uart1_BT_tx_string(char * data);
```

```
//--------------------------------------//
void IntToString(int num);

void gotoSleep();

void sleep_stack_clear();

void PW_buffer_clear();

void voltage_detect();

void ClickSensor(char number);

void dataParsing();

void SendPW();

void Receive_BT_DATA();

ISR(INT0_vect)
{
   //wake up시에만 해당 구문이 돌도록
   sleep_stack_clear();//매번 설정한 시간이 되기 전, 키입력 인터럽트가 발생한다면 해당 시간이 초기화되도록 한다.

   if(sleep_flag)
   {
      sleep_flag=0;
     PORTA |= 0x80;
      //dummy
     // uart1_BT_tx_string("R"); //자릿값 요청 필수로 해줘야 함.

   }

}
//UART Rx Interrupt
ISR(USART0_RX_vect) // {  <, X, Y, Z, >  }
{
   static u8 cnt=0;
   u8 _buff=UDR0; //와... 이걸 생각 못했네. 사용하고나면 레지스터 초기화되서 한 칸씩 밀린거임...


   switch(cnt){
      case 0: if(_buff=='<'){
         TOUCH.sensorData[cnt]=_buff;
         cnt++;
      } break;
      case 4: if(_buff=='>'){
         TOUCH.sensorData[cnt]=_buff;
         cnt=0;
         UCSR0B &= ~(0x80);
         TOUCH.receive_cplt_sensor=1;
      }break;
      default:
         TOUCH.sensorData[cnt]=_buff;
         cnt++;
      break;
```

```
   }

}
//--------------------------------
//문자열을 받고 나서 문자열이 끝나는 부분에 대한 비교 후 인터럽트를 꺼주는 방식.
//단점 : 중간에 null이 들어가면? 구분해낼 수 없음.
ISR(USART0_TX_vect)
{
   if(TOUCH.tx0Buf[TOUCH.tx0Cnt]=='\0') //문자열 마지막부분은 보내지 않는 것이 특징.
   {
      UCSR0B &= ~0x40; //Tx 송신완료 interrupt disabled.
      memset((char*)TOUCH.tx0Buf,0,sizeof(TOUCH.tx0Buf));
   }
   else UDR0 = TOUCH.tx0Buf[TOUCH.tx0Cnt++]; // 0은 이미 보냈고 1이 보내지고 2로 증가,
2보내지고 3으로 증가되고 다음 루틴에서 인터럽트 비활성화
}

//--------------------------------

//-------------------ISR------------------//
//블루투스 데이터 받을 때 사용
ISR(USART1_RX_vect) //
{
   RxData1=UDR1; //우선 임시 저장소로 사용. 어플에서 본격적으로 받을 땐 파싱해줘야 함.
}

ISR(USART1_TX_vect)
{
   if(BT.tx1Buf[BT.tx1Cnt]=='\0') //문자열 마지막부분은 보내지 않는 것이 특징.
   {
      UCSR1B &= ~0x40; //Tx 송신완료 interrupt disabled.
      memset((char*)BT.tx1Buf,0,sizeof(BT.tx1Buf));
   }
   else UDR1 = BT.tx1Buf[BT.tx1Cnt++]; // 0은 이미 보냈고 1이 보내지고 2로 증가, 2보내지고
3으로 증가되고 다음 루틴에서 인터럽트 비활성화
}


//깨알 팁. Tick Timer를 사용하고자 한다면, ticks가 255만큼 증가되는 시간은 0.016s가 걸리며,
16번 ISR에 들어오는 시간은 1ms이다.
ISR(TIMER0_COMP_vect) //대충 16khz 속도로 들어옴.
{
   static unsigned char ticks=0;//uint8_t
   static int twice=0;
   twice++;
   if(twice%=2) return;
   //dummy test
   //PORTC^=0x01;


   if(ticks!=255)
   {
      if(ticks == led_R) PORTA &= ~(0x01);
      if(ticks == led_G) PORTA &= ~(0x02);
```

```
      if(ticks == led_B) PORTA &= ~(0x04);

   }
   else
   {
      //파형 갱신 명령을 여기다가 넣자.
     led_B=led_B_buf;
     led_G=led_G_buf;
     led_R=led_R_buf;
      PORTA|=( ((led_B!=0?1:0)<<2) | ((led_G!=0?1:0)<<1) | ((led_R!=0?1:0)<<0) );
   }

   TICK.tick_tim++;
   //1ms
   if(!(TICK.tick_tim%16))
   {
      //PORTC ^=0x01;//dummy code
      TICK.t_1ms++;
      TICK.buzz_1ms++;
      TICK.sleep_tim++;
     TICK.rgb_1ms++;
     TICK.motor_1ms++;
    TICK.doubleClick_tick_1ms++;
    if(TICK.wait_1s<1500)TICK.wait_1s++;
   }

   if(!(TICK.tick_tim%256)) //대략 16ms 정도 지날 때마다 카운팅
      {
          TICK.rgb_16ms++;
       if((TICK.rgb_16ms%32)==0) //대략 0.512초마다 카운팅이 바뀜
         {
            //PORTC^=0x01;
            if(dimming_count_flag!=UPCOUNT)dimming_count_flag =UPCOUNT;
            else dimming_count_flag =DOWNCOUNT;
         }
      }
   ticks++; //255 이후 Clear.



}
//----------------------------------------//

int main(void)
{

    /* Replace with your application code */

   setup();

    //dummy
   //uart1_BT_tx_string("Project Start \n");
    //setRGB(50,50,50);
    setSoundClip(BUZZ_SLEEPOUT);//dummy
   setStateRGB(RGB_DEFAULT);
   setServoAngle(0);
   while (1)
    {
    //dummycode  PORTC=0x01;
```

```
  //////////
    //tick 타이머는 모두 인터럽트에서 싹다 처리해야될 것 같다. 처리하는게 메인에서도
관여하면 꼬인다 매우 중요!
    if(TICK.sleep_tim==29000) setSoundClip(BUZZ_SLEEPOUT); //슬립모드 빠질 준비
    else if(TICK.sleep_tim>=30000) //30초 뒤에 슬립모드 빠짐.
   {//성공 시 2초 뒤에 다시 슬립모드 빠지도록 구성.
        gotoSleep();
   }

  //5회 이상 실패한 경우.
  if(fail_stack==OVERSTACK){ //fail_stack=6인 상태
      if(TICK.t_1ms>=20){ // check per 20ms
             PORTC ^=0x01;
             TICK.t_1ms=0;
             if((RxData1>>4) == 0x06){//셔플신호가 들어왔을 때
                    fail_stack=0;
                    RxData1=0;
                    setStateRGB(RGB_DEFAULT);
                    setSoundClip(BUZZ_MUTE);
             }
      }

//         _delay_ms(10000);
//         setStateRGB(RGB_DEFAULT);
//         fail_stack=0;
      } //TICK.sleep_tim=15000;
  else{
             if(TICK.t_1ms>=10){ // check per 20ms
                    TICK.t_1ms=0;
                    uart0_sensor_tx_char('R'); //센서 데이터 갱신 명령 전송. 어차피
센서데이터 수신은 최대 대략 768Hz마다 들어옴.

                    //uartTx("R"); 혹은 char을 'R'로 보내줘야지

                    if(TOUCH.receive_cplt_sensor){ // 데이터 버퍼에 정상적으로
데이터가 들어왔을 때
                    TOUCH.receive_cplt_sensor=0;
                    dataParsing();
                    }//parsing end


               }//tick end
              if(IsWakeup==0){ //블루투스 연결됐을 때만  버퍼에 들어갈 수 있도록 수정
요망.
              for(int i=0; i<12;i++){
                            if(TOUCH.KEY[i]==PRESS){
                              if(pressed_flag[i]==0){
                              switch(i){

                                     case 0: if(BT.MuteBtnLoc==0x00)
MuteMode_toggle();
                                     else if(BT.SendBtnLoc==0x00)SendPW();
                                     else ClickSensor('0');
```

```
                                    break;
                                    case 1: if(BT.MuteBtnLoc==0x01)
MuteMode_toggle();
                                    else if(BT.SendBtnLoc==0x01)SendPW();
                                    else ClickSensor('1');
                                    break;
                                    case 2: if(BT.MuteBtnLoc==0x02)
MuteMode_toggle();
                                    else if(BT.SendBtnLoc==0x02)SendPW();
                                    else ClickSensor('2');
                                    break;
                                    case 3: if(BT.MuteBtnLoc==0x03)
MuteMode_toggle();
                                    else if(BT.SendBtnLoc==0x03)SendPW();
                                    else ClickSensor('3');
                                    break;
                                    case 4: if(BT.MuteBtnLoc==0x04)
MuteMode_toggle();
                                    else if(BT.SendBtnLoc==0x04)SendPW();
                                    else ClickSensor('4');
                                    break;
                                    case 5: if(BT.MuteBtnLoc==0x05)
MuteMode_toggle();
                                    else if(BT.SendBtnLoc==0x05)SendPW();
                                    else ClickSensor('5');
                                    break;
                                    case 6: if(BT.MuteBtnLoc==0x06)
MuteMode_toggle();
                                    else if(BT.SendBtnLoc==0x06)SendPW();
                                    else ClickSensor('6');
                                    break;
                                    case 7: if(BT.MuteBtnLoc==0x07)
MuteMode_toggle();
                                    else if(BT.SendBtnLoc==0x07)SendPW();
                                    else ClickSensor('7');
                                    break;
                                    case 8: if(BT.MuteBtnLoc==0x08)
MuteMode_toggle();
                                    else if(BT.SendBtnLoc==0x08)SendPW();
                                    else ClickSensor('8');
                                    break;
                                    case 9: if(BT.MuteBtnLoc==0x09)
MuteMode_toggle();
                                    else if(BT.SendBtnLoc==0x09)SendPW();
                                    else ClickSensor('9');
                                    break;
                                    case 10: if(BT.MuteBtnLoc==0x0a)
MuteMode_toggle();
                                    else if(BT.SendBtnLoc==0x0a)SendPW();
                                    else ClickSensor('A');
                                    break;
                                    case 11: if(BT.MuteBtnLoc==0x0b)
MuteMode_toggle();
                                    else if(BT.SendBtnLoc==0x0b) SendPW();
                                    else ClickSensor('B');
                                    break;

                                }

                                    pressed_flag[i]=1;
                                }
```

```
                                }
                            else pressed_flag[i]=0;
                        } // for(int i=0; i<12;i++) end
                       }//if(sleep_flag!=1) end

                    Receive_BT_DATA();
                  RGB_Drive();
                  motor_drive();
                  buzz_play();

                }

            }// while end
}

void setup()
{
   //memset((char*)TOUCH.PW,0,sizeof(TOUCH.PW));

   sei();
   DDRA |= ((1<<DDRA7)|(0<<DDRA6)|(0<<DDRA5)|(1<<DDRA2)|(1<<DDRA1)|(1<<DDRA0));
   DDRB |= (1<<DDRB5);
   DDRC |=((1<<DDRC1)|(1<<DDRC0));//dummy test port 0,1 pin
   DDRE |= ((1<<DDRE3)); //OC3A핀 출력모드.

   //---------사용하지 않는 핀에 대해 PULL UP 설정--------//
   PORTA|=(0b01011000); //0,1,2번핀은 RGB 핀 7번핀은 릴레이, 6번핀은 저전압 경고 풀업저항
사용.
   PORTB|=(0x11011111); //5번 핀은 서보모터 핀
   PORTC|=(0b11111100); // 0, 1번핀은 디버그핀
   PORTD|=(0b11110010); // 2, 3번핀은 UART1, 0번핀은 INT0
   PORTE|=(0b11110100); // 3번핀은 BUZZ, 0,1번핀은 UART0
   PORTF|=(0b11111111);
   //---------사용하지 않는 핀에 대해 PULL UP 설정--------//


   //------------Default 입출력 설정---------------//

   PORTA |= 0x80; //릴레이 ON

   //------------Default 입출력 설정---------------//
   //External Interrupt
   EIMSK|= (1<<INT0);
   EICRA |= ((1<<ISC01)|(0<<ISC00)); //default HIGH > event, LOW : Falling Edge

   //UART0
   UART0_Init(MYUBRR(38400));
   UART1_Init(MYUBRR(9600)); //9600 -> 38400
   //TIM0
   //16Mhz > 8 prescailing(CS:010) >>2MHz/count >> 0.0625ms compare match >> GPIO PWM
주기는 0.0625ms*256 => 62.5Hz
   //CTC mode , OC0 disconnected
   TCCR0 = ((1<<CS00) | (0<< CS01) | (0<<CS02) |
(1<<WGM01)|(0<<WGM00)|(0<<COM01)|(0<<COM00));
```

```
   /** 16MHz일때
   OCR0=125
   */
   /**8MHz일 때*/
   OCR0 = 250; //
   TIMSK |= (1<<OCIE0); //to using tick TIMER

   //TIM1
   //COM : 10 : default HIGH, compare LOW
   //WGM : 1110 : FAST PWM, TOP:ICR1
   TCCR1A = ((1<<COM1A1)|(0<<COM1A0)|(1<<COM1B1)|(0<<COM1B0)|(1<<WGM11)|(0<<WGM10));
   TCCR1B = ((0<<CS12)|(1<<CS11)|(0<<CS10)|(1<<WGM13)|(1<<WGM12) );
   TCNT1H = 0;
   TCNT1L = 0;
   /**16MHz일 때
   setICR1(39999);//0x9c3f
   */
   /**8MHz일 때*/
   setICR1(19999);
   ////////////////////////////////////
   //TIM3
   // prescailing : 8
   // if ICR==4000 : 62.5Hz > 실제 주파수 31.25Hz
   //
   //COM : 01 : Toggle (in CTC)
   //WGM : 1100 : CTC : TOP : ICR3
   TCCR3A =
((0<<COM3A1)|(1<<COM3A0)|(0<<COM3B1)|(0<<COM3B0)|(0<<COM3C1)|(0<<COM3C0)|(0<<WGM31)|(0
<<WGM30));
   TCCR3B = ((0<<CS32)|(1<<CS31)|(0<<CS30)|(1<<WGM33)|(1<<WGM32) );
   TCNT3H = 0;
   TCNT3L = 0;
   setICR3(16667); //60Hz default



   // 전원 ON 시 default 값
   BT.SendBtnLoc=0x00;
   BT.MuteBtnLoc=0x01;

}



void setOCR3A(int num)
{
   OCR3AH = (unsigned char)(num>>8);
   OCR3AL = (unsigned char)(num&0xff);

}

void setOCR3B(int num)
{
   OCR3BH = (unsigned char)(num>>8);
   OCR3BL = (unsigned char)(num&0xff);

}
```

```
void setOCR3C(int num)
{
   OCR3CH = (unsigned char)(num>>8);
   OCR3CL = (unsigned char)(num&0xff);

}
void setICR3(int num){
   //high write first
   /**8MHz의 경우*/
   num=(int)(num*0.5);
   ICR3H = (unsigned char)(num>>8);
   ICR3L = (unsigned char)(num&0xff);
}


void UART0_Init(unsigned int ubrr)
{

   DDRE &= ~(0x01);
   //set Baud Rate
   UBRR0H= (unsigned char )(ubrr>>8);
   UBRR0L= (unsigned char )(ubrr&0xff);

   // Tx, Rx Enabled
   UCSR0B = ((1<<TXEN0) | (1<<RXEN0));
   UCSR0B |= ((1<<RXCIE0)); //수신 완료 인터럽트 허용

   // Set Frame format: data 8 bit, 1 stop bit
   UCSR0C = (0<<USBS0) |(1<<UCSZ01)| (1<<UCSZ00 );
}
void UART1_Init(unsigned int ubrr)
{
   //set Baud Rate
   UBRR1H= (unsigned char )(ubrr>>8);
   UBRR1L= (unsigned char )(ubrr&0xff);

   // Tx, Rx Enabled
   UCSR1B = ((1<<TXEN1) | (1<<RXEN1));
   UCSR1B |= ((1<<RXCIE1)); //수신 완료 인터럽트 허용

   // Set Frame format: data 8 bit, 1 stop bit
   UCSR1C = (0<<USBS1) |(1<<UCSZ11)| (1<<UCSZ10 );
}

void uart0_sensor_tx_char(char data)
{
   while(!(UCSR0A & (1<<UDRE0)));
   UDR0=data;
}
void uart0_sensor_tx_string(char * data)
{
   int _len=strlen(data);
   strncpy((char*)TOUCH.tx0Buf,data,_len);

   while(!(UCSR0A & (1<<UDRE0)));
   //UDR0=data;
   UDR0=TOUCH.tx0Buf[0];

   TOUCH.tx0Cnt=1;    //ISR상에서 사용할 cnt 시작 값
```

```
   TOUCH.tx0CntMax=_len+1; //ISR상에서 사용할 cnt max 값 (문자열 뒤에 붙는 \0도 포함)
   //cm0.txCntMax=5;
   //UCSR0B |= 0x40; //송신완료 인터럽트 활성화
   UCSR0B |= (1<<TXCIE0);
   //UCSR0A |= (1<<UDRE0);
}

void uart1_BT_tx_string(char * data)
{
   int _len=strlen(data);
   strncpy((char*)BT.tx1Buf,data,_len);



   while(!(UCSR1A & (1<<UDRE1)));
   //UDR0=data;
   UDR1=BT.tx1Buf[0];

   BT.tx1Cnt=1;    //ISR상에서 사용할 cnt 시작 값
   BT.tx1CntMax=_len+1; //ISR상에서 사용할 cnt max 값 (문자열 뒤에 붙는 \0도 포함)

   UCSR1B |= (1<<TXCIE1);//송신완료 인터럽트 활성화
   //UCSR0A |= (1<<UDRE0);
}

void ClickSensor(char number){
   TOUCH.PW[pw_i]=(unsigned char)number;
   setSoundClip(BUZZ_BEEP);
   setStateRGB(RGB_TOUCHED)
   pw_i++;
   mode_change_stack=0;
}
void SendPW(){


    TOUCH.PW[0]='<';
    TOUCH.PW[pw_i]='>';
    uart1_BT_tx_string((char *)TOUCH.PW);
    memset((char*)TOUCH.PW,0,sizeof(TOUCH.PW));
    pw_i=1;
    mode_change_stack=0;

}

void dataParsing(){
           for(int i=0; i<3; i++)
           { //노가다 파싱...과정1
              switch(TOUCH.sensorData[i+1])
              {

                 case '0': TOUCH.DataBuff[i+1]=0x00;
                 break;
                 case '1': TOUCH.DataBuff[i+1]=0x01;
                 break;
                 case '2': TOUCH.DataBuff[i+1]=0x02;
                 break;
                 case '3':TOUCH.DataBuff[i+1]=0x03;
                 break;
```

```
                case '4':TOUCH.DataBuff[i+1]=0x04;
                break;
                case '5':TOUCH.DataBuff[i+1]=0x05;
                break;
                case '6':TOUCH.DataBuff[i+1]=0x06;
                break;
                case '7':TOUCH.DataBuff[i+1]=0x07;
                break;
                case '8':TOUCH.DataBuff[i+1]=0x08;
                break;
                case '9':TOUCH.DataBuff[i+1]=0x09;
                break;
                case 'A':TOUCH.DataBuff[i+1]=0x0a;
                break;
                case 'B':TOUCH.DataBuff[i+1]=0x0b;
                break;
                case 'C':TOUCH.DataBuff[i+1]=0x0c;
                break;
                case 'D':TOUCH.DataBuff[i+1]=0x0d;
                break;
                case 'E':TOUCH.DataBuff[i+1]=0x0e;
                break;
                case 'F':TOUCH.DataBuff[i+1]=0x0f;
                break;

            }
          }

          //노가다 파싱 과정....2
//            TOUCH.KEY[0]=((TOUCH.DataBuff[1]&0b1000)!=0) ?PRESS:RELEASE;
//            TOUCH.KEY[3]=((TOUCH.DataBuff[1]&0b0100)!=0) ?PRESS:RELEASE;
//            TOUCH.KEY[6]=((TOUCH.DataBuff[1]&0b0010)!=0) ?PRESS:RELEASE;
//            TOUCH.KEY[9]=((TOUCH.DataBuff[1]&0b0001)!=0) ?PRESS:RELEASE;
//
//            TOUCH.KEY[1]=((TOUCH.DataBuff[2]&0b0001)!=0) ?PRESS:RELEASE;
//            TOUCH.KEY[4]=((TOUCH.DataBuff[2]&0b0010)!=0) ?PRESS:RELEASE;
//            TOUCH.KEY[7]=((TOUCH.DataBuff[2]&0b1000)!=0) ?PRESS:RELEASE;
//            TOUCH.KEY[10]=((TOUCH.DataBuff[2]&0b100)!=0) ?PRESS:RELEASE;
//
//            TOUCH.KEY[2]=((TOUCH.DataBuff[3]&0b0001)!=0) ?PRESS:RELEASE;
//            TOUCH.KEY[5]=((TOUCH.DataBuff[3]&0b0010)!=0) ?PRESS:RELEASE;
//            TOUCH.KEY[8]=((TOUCH.DataBuff[3]&0b0100)!=0) ?PRESS:RELEASE;
//            TOUCH.KEY[11]=((TOUCH.DataBuff[3]&0b1000)!=0) ?PRESS:RELEASE;

//터치 키패드 부착 방향을 반대로 오인하여 코드로 수정해줌. 세로방향으로 뒤집었음.
          TOUCH.KEY[9]=((TOUCH.DataBuff[1]&0b1000)!=0) ?PRESS:RELEASE;
          TOUCH.KEY[6]=((TOUCH.DataBuff[1]&0b0100)!=0) ?PRESS:RELEASE;
          TOUCH.KEY[3]=((TOUCH.DataBuff[1]&0b0010)!=0) ?PRESS:RELEASE;
          TOUCH.KEY[0]=((TOUCH.DataBuff[1]&0b0001)!=0) ?PRESS:RELEASE;

          TOUCH.KEY[10]=((TOUCH.DataBuff[2]&0b0001)!=0) ?PRESS:RELEASE;
          TOUCH.KEY[7]=((TOUCH.DataBuff[2]&0b0010)!=0) ?PRESS:RELEASE;
          TOUCH.KEY[4]=((TOUCH.DataBuff[2]&0b1000)!=0) ?PRESS:RELEASE;
          TOUCH.KEY[1]=((TOUCH.DataBuff[2]&0b100)!=0) ?PRESS:RELEASE;

          TOUCH.KEY[11]=((TOUCH.DataBuff[3]&0b0001)!=0) ?PRESS:RELEASE;
          TOUCH.KEY[8]=((TOUCH.DataBuff[3]&0b0010)!=0) ?PRESS:RELEASE;
          TOUCH.KEY[5]=((TOUCH.DataBuff[3]&0b0100)!=0) ?PRESS:RELEASE;
          TOUCH.KEY[2]=((TOUCH.DataBuff[3]&0b1000)!=0) ?PRESS:RELEASE;
```

```
        //dummy code
           TOUCH.KEY[12]='\n';
           TOUCH.KEY[13]=0;
        ////////////////

    UCSR0B |=(1<<RXCIE0);
}

void Receive_BT_DATA(){

         BT.CommandBuff=RxData1;

         if(RxData1!=0)sleep_stack_clear();//뭔가 데이터가 들어왔을 때? 슬립모드 카운터
클리어

         RxData1=0;
         if((BT.CommandBuff>>4) == 0x06) //id 값이 자릿값 위치(0x6n)
         {
            //Location 갱신 0x6n
            BT.SendBtnLoc = BT.CommandBuff&0x0f; //0x0~0xb 까지 들어오는지 체크
            BT.CommandBuff=0;
         }
         else if((BT.CommandBuff>>4)==0x04){ // 비밀번호 성공 실패 여부 수신
            //PW Success or Fail
            BT.PaswrdFlg=BT.CommandBuff&0x0f; //0x1~0x2이 들어오는지 체크
            if(BT.PaswrdFlg==0x01) {
               BT.PaswrdFlg=0;
               fail_stack=0;
               setServoAngle(OPEN);//open the door
               setSoundClip(BUZZ_SUCCESS);
               setStateRGB(RGB_SUCCESS);

               motor_flag=1; //다음 if문실행
               //TICK.motor_1ms=0;
            }//success
            else if(BT.PaswrdFlg==0x02) {
               BT.PaswrdFlg=0;
               fail_stack++;

               if(fail_stack==5){
                 setServoAngle(CLOSE);
                 setSoundClip(BUZZ_SIREN);
                 setStateRGB(RGB_SIREN);
                 //TICK.sleep_tim=25000;
               }
                else {
                   setServoAngle(CLOSE);
                   setSoundClip(BUZZ_FAIL);
                   setStateRGB(RGB_FAIL);
                }
               motor_flag=0;
            } //fail
            BT.CommandBuff=0;
         }//비밀번호 성공실패 체크 루틴 end
         else if((BT.CommandBuff>>4)==0x07)//#위치 수신
```

```
          {
             BT.MuteBtnLoc=BT.CommandBuff&0x0f;
             //////////아래에 코드 정리해서 넣으면 된다.
             //Define은 0이 되지 않도록 한다.

             BT.CommandBuff=0;
          }

          //비밀번호 일치 여부 0x40, 0x41
}



//------------------------------Actuator function--------------------------//
void setRGB(char _R, char _G, char _B)
{
   led_R_buf=_R;
   led_G_buf=_G;
   led_B_buf=_B;
}

void setStateRGB(char state)
{
     switch(state)
     {
        case RGB_OFF: RGB_status_flag=RGB_OFF; break;
        case RGB_DEFAULT: RGB_status_flag=RGB_DEFAULT; break;
        case RGB_LOWBAT: RGB_status_flag=RGB_LOWBAT; break;
        case RGB_SUCCESS: RGB_status_flag=RGB_SUCCESS; break;
        case RGB_FAIL: RGB_status_flag=RGB_FAIL; break;
        case RGB_CLOSE_DOOR: RGB_status_flag=RGB_CLOSE_DOOR; break;
        case RGB_SLEEPOUT: RGB_status_flag=RGB_SLEEPOUT; break;
      case RGB_SIREN: RGB_status_flag=RGB_SIREN; break;
          case RGB_TOUCHED: RGB_status_flag=RGB_TOUCHED; break;
     }
     TICK.rgb_16ms=0;
         TICK.rgb_1ms=0;
         dimming_count_flag = UPCOUNT;
}


void RGB_Drive()
{
   switch(RGB_status_flag)
   {
      case RGB_OFF: setRGB(0,0,0); break;
      case RGB_DEFAULT:
         if(dimming_count_flag==UPCOUNT)setRGB(0,0,8*(TICK.rgb_16ms%32));
         else setRGB(0,0,252-8*(TICK.rgb_16ms%32));
      break;
          case RGB_TOUCHED:
          if(TICK.rgb_1ms<10) setRGB(0,200,50);
          else if(TICK.rgb_1ms==20) setRGB(0,0,0);
           break;
      case RGB_LOWBAT:
         if(TICK.rgb_1ms<150) setRGB(255,0,127);
         else if(TICK.rgb_1ms==155) setRGB(0,0,0);
         else if(TICK.rgb_1ms==160) setRGB(255,0,127);
         else if(TICK.rgb_1ms==270) setRGB(0,0,0);
```

```
         else if(TICK.rgb_1ms==275) setRGB(255,0,127);
         else if(TICK.rgb_1ms==385) setRGB(0,0,0);
         else if(TICK.rgb_1ms==390) setRGB(255,0,127);
         else if(TICK.rgb_1ms==500) setRGB(0,0,0);
         else if(TICK.rgb_1ms==505) setRGB(255,0,127);
         else if(TICK.rgb_1ms==615) setRGB(0,0,0);
         else if(TICK.rgb_1ms==620) setRGB(255,0,127);
         else if(TICK.rgb_1ms==730) setRGB(0,0,0);
      break;
      case RGB_SUCCESS:
         if(TICK.rgb_1ms<200) setRGB(0,216,255);
         else if(TICK.rgb_1ms==210) setRGB(0,0,0);
         else if(TICK.rgb_1ms==220) setRGB(0,84,255);
         else if(TICK.rgb_1ms==400) setRGB(0,0,0);
         else if(TICK.rgb_1ms==430) setRGB(1,0,255);
         else if(TICK.rgb_1ms==900) setRGB(0,0,0);
      break;
      case RGB_FAIL:
         if(TICK.rgb_1ms<95) setRGB(255,0,0);
         else if(TICK.rgb_1ms==100) setRGB(0,0,0);
         else if(TICK.rgb_1ms==180) setRGB(255,0,0);
         else if(TICK.rgb_1ms==280) setRGB(0,0,0);
         else if(TICK.rgb_1ms==360) setRGB(255,0,0);
         else if(TICK.rgb_1ms==460) setRGB(0,0,0);
      break;

      case RGB_CLOSE_DOOR:
         if(TICK.rgb_1ms<100) setRGB(171,242,0);
         else if(TICK.rgb_1ms==120) setRGB(0,0,0);
         else if(TICK.rgb_1ms==170) setRGB(159,201,60);
         else if(TICK.rgb_1ms==270) setRGB(0,0,0);
         else if(TICK.rgb_1ms==290) setRGB(47,157,39);
         else if(TICK.rgb_1ms==400) setRGB(0,0,0);
         else if(TICK.rgb_1ms==510) setRGB(1,255,0);
         else if(TICK.rgb_1ms==800) setRGB(0,0,0);
      break;

      case RGB_SLEEPOUT:
         if(TICK.rgb_1ms<100) setRGB(255,187,0);
         else if(TICK.rgb_1ms==120) setRGB(0,0,0);
         else if(TICK.rgb_1ms==170) setRGB(255,187,0);
         else if(TICK.rgb_1ms==270) setRGB(0,0,0);
         else if(TICK.rgb_1ms==290) setRGB(255,94,0);
         else if(TICK.rgb_1ms==490) setRGB(0,0,0);
      break;
           case RGB_SIREN:
           if(TICK.rgb_1ms<95) setRGB(255,0,0);
           break;
   }
}
void setBright(int br)
{
   bright=br;
}



void setSoundClip(char clip)
{
    // 부저 관련 tick.clear
   switch(clip)
```

```
   {
      case BUZZ_MUTE: music_flag=BUZZ_MUTE; break;
      case BUZZ_BEEP: music_flag=BUZZ_BEEP; break;
     case BUZZ_FAIL: music_flag=BUZZ_FAIL; break;
      case BUZZ_SUCCESS: music_flag=BUZZ_SUCCESS; break;
     case BUZZ_LOWBAT: music_flag=BUZZ_LOWBAT; break;
     case BUZZ_SLEEPOUT: music_flag=BUZZ_SLEEPOUT; break;
     case BUZZ_CLOSEDOOR: music_flag=BUZZ_CLOSEDOOR; break;
    case BUZZ_SILENCE: music_flag=BUZZ_SILENCE; break;
    case BUZZ_BELLIN: music_flag=BUZZ_BELLIN; break;
    case BUZZ_SIREN: music_flag=BUZZ_SIREN; break;
   }
   TICK.buzz_1ms=0;
}
//have to called this function always

void buzz_play()
{
   if(MuteModeFlag==MUTE_DISABLE) {
      //wait 1s
      if(wait_1s_flag==1) {
         wait_1s_flag=0;
         TICK.wait_1s=0;
      }
      if(TICK.wait_1s>500){
        buzz_MUTE();
        return;
      }
   }

   //재생이 끝났으면 music_flag는 확실하게 MUTE로 들어가야 함. 안그러면 꼬이는 것 같다.

   switch(music_flag)
   {
      case BUZZ_MUTE:  buzz_MUTE(); break; //setICR3(0);. buzz_MUTE() 안에
music_flag=MUTE 넣어주는 명령 들어있음.

      case BUZZ_BEEP:
         TCCR3A |= (1<<COM3A0); //재생 시 타이머카운터 3번 채널 A채널 고유 핀 토글모드로
출력 설정.
         if(TICK.buzz_1ms<10)setICR3(_960Hz);
         else if(TICK.buzz_1ms==20) buzz_MUTE(); //buzz_MUTE();
      break;
     case BUZZ_FAIL:
      //TCCR3A |= (1<<COM3A0); //재생 시 타이머카운터 3번 채널 A채널 고유 핀 토글모드로 출력
설정.
      if(TICK.buzz_1ms<10)setSoundNote(600);
      else if(TICK.buzz_1ms==100) setSoundNote(BUZZ_MUTE); //buzz_MUTE();
      else if(TICK.buzz_1ms==180) setSoundNote(600);
      else if(TICK.buzz_1ms==280) setSoundNote(BUZZ_MUTE);
      else if(TICK.buzz_1ms==360) setSoundNote(600);
      else if(TICK.buzz_1ms==460) {buzz_MUTE();}
     break;
     case BUZZ_SUCCESS:
      if(TICK.buzz_1ms<200)setSoundNote(C);
      else if(TICK.buzz_1ms==210) setSoundNote(BUZZ_MUTE);
      else if(TICK.buzz_1ms==220) setSoundNote(E);
      else if(TICK.buzz_1ms==400) setSoundNote(BUZZ_MUTE);
```

```
      else if(TICK.buzz_1ms==430) setSoundNote(A);
      else if(TICK.buzz_1ms==900) buzz_MUTE();
     break;
     case BUZZ_LOWBAT:
        if(TICK.buzz_1ms<10)setSoundNote(600);
        else if(TICK.buzz_1ms==150) setSoundNote(1200); //buzz_MUTE();
        else if(TICK.buzz_1ms==300) setSoundNote(600);
        else if(TICK.buzz_1ms==450) setSoundNote(1200);
        else if(TICK.buzz_1ms==600) setSoundNote(600);
        else if(TICK.buzz_1ms==750) setSoundNote(1200);
        else if(TICK.buzz_1ms==900) {buzz_MUTE();IsWakeup=0;}
     break;
      case BUZZ_SLEEPOUT:
      if(TICK.buzz_1ms<100)setSoundNote(A);
      else if(TICK.buzz_1ms==120) setSoundNote(BUZZ_MUTE);
      else if(TICK.buzz_1ms==170) setSoundNote(C);
      else if(TICK.buzz_1ms==270) setSoundNote(BUZZ_MUTE);
      else if(TICK.buzz_1ms==290) setSoundNote(D);
      else if(TICK.buzz_1ms==490) {buzz_MUTE();IsWakeup=0;}
     break;
     case BUZZ_CLOSEDOOR:
      if(TICK.buzz_1ms<100)setSoundNote(G);
      else if(TICK.buzz_1ms==120) setSoundNote(BUZZ_MUTE);
      else if(TICK.buzz_1ms==170) setSoundNote(G);
      else if(TICK.buzz_1ms==270) setSoundNote(BUZZ_MUTE);
      else if(TICK.buzz_1ms==290) setSoundNote(G);
      else if(TICK.buzz_1ms==400) setSoundNote(BUZZ_MUTE);

      else if(TICK.buzz_1ms==510) setSoundNote(A);
      else if(TICK.buzz_1ms==530) setSoundNote(BUZZ_MUTE);
      else if(TICK.buzz_1ms==560) setSoundNote(B);
      else if(TICK.buzz_1ms==640) setSoundNote(BUZZ_MUTE);
      else if(TICK.buzz_1ms==660) setSoundNote(Cs);
      else if(TICK.buzz_1ms==800) buzz_MUTE();
      break;
    case BUZZ_SILENCE:
      if(TICK.buzz_1ms<100)setSoundNote(Cs);
      //else if(TICK.buzz_1ms==190) setSoundNote(BUZZ_MUTE);
      else if(TICK.buzz_1ms==200) setSoundNote(D);
      else if(TICK.buzz_1ms==400) buzz_MUTE();
      break;
     case BUZZ_BELLIN:
      if(TICK.buzz_1ms<100)setSoundNote(D);
      //else if(TICK.buzz_1ms==190) setSoundNote(BUZZ_MUTE);
      else if(TICK.buzz_1ms==200) setSoundNote(Cs);
      else if(TICK.buzz_1ms==400) buzz_MUTE();
     break;
    case BUZZ_SIREN: //5번 fail_stack =5 >> 6
      if(TICK.buzz_1ms<20) setSoundNote(600);
      else if(TICK.buzz_1ms==2000) setSoundNote(BUZZ_MUTE);
      else if(TICK.buzz_1ms==2010) {buzz_MUTE();fail_stack=OVERSTACK;}
    break;


   }
}
void MuteMode_toggle() //500ms 안에 연속 두번 들어온다면 그때 모드 전환이 이뤄짐.
{

   //setSoundClip(BUZZ_BEEP);
```

```
   if(TICK.doubleClick_tick_1ms>500) {mode_change_stack=0;mode_change_stack++;
clickTimerON_SW=1;} //평상시, 혹은 0.5초 이후 터치 시도
   else mode_change_stack++;   //0.5초 내로 추가 터치

   if(clickTimerON_SW==1){clickTimerON_SW=0; TICK.doubleClick_tick_1ms=0;} //start
count tick

   if(mode_change_stack==2){

if(MuteModeFlag==MUTE_ENABLE){setSoundClip(BUZZ_SILENCE);MuteModeFlag=MUTE_DISABLE;}
      else
if(MuteModeFlag==MUTE_DISABLE){setSoundClip(BUZZ_BELLIN);MuteModeFlag=MUTE_ENABLE;}
      wait_1s_flag=1; //효과음 재생하고나서 무음모드, 벨소리모드 진입.
   }
}

void buzz_MUTE()
{
   TCCR3A &= ~(1<<COM3A0); // 타이머카운터3번 A채널 고유 핀 출력 X
   music_flag = BUZZ_MUTE;
}

void setSoundNote(int note)
{
   if(BUZZ_MUTE!=note){TCCR3A |= (1<<COM3A0);setICR3(note);}
   else {TCCR3A &= ~(1<<COM3A0);}

}

void setOCR1A(int num)
{
   OCR1AH = (unsigned char)(num>>8);
   OCR1AL = (unsigned char)(num&0xff);

}

void setOCR1B(int num)
{
   OCR1BH = (unsigned char)(num>>8);
   OCR1BL = (unsigned char)(num&0xff);

}

void setICR1(int num)
{
   //39999 in 16MHz
   //19999 in  8MHz
   //high write first
   ICR1H = (unsigned char)(num>>8);
   ICR1L = (unsigned char)(num&0xff);
}

int convertServoAngle(int angle)
{
   //0~180 degree
   /**16Mhz 크리스탈 사용했을 경우
   int converted = 2000+angle*5;
   */
```

```
   /** 8MHz 크리스탈 사용했을 경우*/
   int converted = 500+angle*11; //DUTY range : 2.5%~12.5% : 500~ 2500
   return converted;
}

void setServoAngle(int angle)
{
   TCCR1A |= (1<<COM1A1);
   setOCR1A(convertServoAngle(angle));
}

void motor_drive()//하다 말았음. 액티브하이인지 로우인지 기억이 안남.
{//door_open_flag
   int reedSW_buff=(PINA&0x20);
   if((reedSW_buff)==0x20) door_open_flag=DOOR_OPEN_STATE;
   else if((reedSW_buff)!=0x20)door_open_flag=DOOR_CLOSE_STATE;

   if(motor_flag)
   { //문닫혀있는 상태 : DOOR CLOSE STATE > 잠금 풀리고 > 문 열리면 DOOR OPEN STATE
      if(door_open_flag==DOOR_OPEN_STATE){ // ____----
         reedSW_state=HIGH;
      }
      //falling edge일 때 잠금모드 진입해야지
      else if(door_open_flag==DOOR_CLOSE_STATE) {
         if(reedSW_state==HIGH)// ----____ 일때만
         {
            reedSW_state=LOW;
            command_door_lock=1;
            TICK.motor_1ms=0;
            //문닫아명령 tick=0부터 이제 3초 세기 시작, 그리고 sleep 타이머도 세팅
         }
      }
   }
   //TICK.sleep_tim=5000; //성공하고나서 빠르게 슬립모드 넣고 싶을 때.
   if(command_door_lock==1)
   {
      if(TICK.motor_1ms==1500)setServoAngle(CLOSE); //close the door
      else if(TICK.motor_1ms==2000)setSoundClip(BUZZ_CLOSEDOOR);
      else if(TICK.motor_1ms==2500)
      {
         TCCR1A&=~(1<<COM1A1);
         command_door_lock=0;
         motor_flag=0;
      }
   }

}

//-----------------------------Actuator end-------------------------//

void IntToString(int num)
{
   /** Dummy Function
   long int data=123456;
   int data1,data2;
   data1=data*0.001;
   data2=data%1000;
   char buff[5]={0,};
```

```
   sprintf(buff,"%d%d",data1,data2); //고유 자릿값
   */
}

void gotoSleep(){
   //사용하는 핀들도 다 슬립모드에선 입력모드로 바꿔도 될까?
   //우선 사용하지 않는 핀에 대해 PULL UP으로 설정.

//   MCUCR |=((1<<SE)|(0<<SM2)|(1<<SM1)|(0<<SM0));
   sleep_flag=1;
   PORTA &= ~(0x80); //릴레이 OFF
   PORTA &= ~(0x07); //RGB OFF

   set_sleep_mode (SLEEP_MODE_PWR_DOWN); //only interrupt
   sleep_enable ();
   sei();
   sleep_cpu();

   //아래 부터 다시 깰 경우 flag 작성
   //_delay_ms(20);
   IsWakeup=1;
   //비밀번호 버퍼 모두 지움

   PW_buffer_clear();
   memset((char*)TOUCH.DataBuff,0,sizeof(TOUCH.DataBuff));
   memset((char*)TOUCH.sensorData,0,sizeof(TOUCH.sensorData));
   voltage_detect();
   TICK.rgb_16ms=0;
}
void sleep_stack_clear()
{
   TICK.sleep_tim=0;
}
void PW_buffer_clear()
{
    memset((char*)TOUCH.PW,0,sizeof(TOUCH.PW));
}

//센서 문제로 인해 정상 동작하지 않음.
void voltage_detect()
{//출력 몇볼트인지 확인하기. 6V전원 >> 신호출력이 3.3V 전원공급되는 MCU 에서도 정상입력되는지 체크.
     //입력모드인데 풀업으로 걸려있음. >> sensor : Default HIGH, active low
     int voltageDetector_buff=(PINA&0x40);
     if((voltageDetector_buff)==0x40) {setSoundClip(BUZZ_SLEEPOUT);} //4.5V 미만일 때 저전압 경고 등
     else if((voltageDetector_buff)!=0x40) setSoundClip(BUZZ_LOWBAT);
}
```