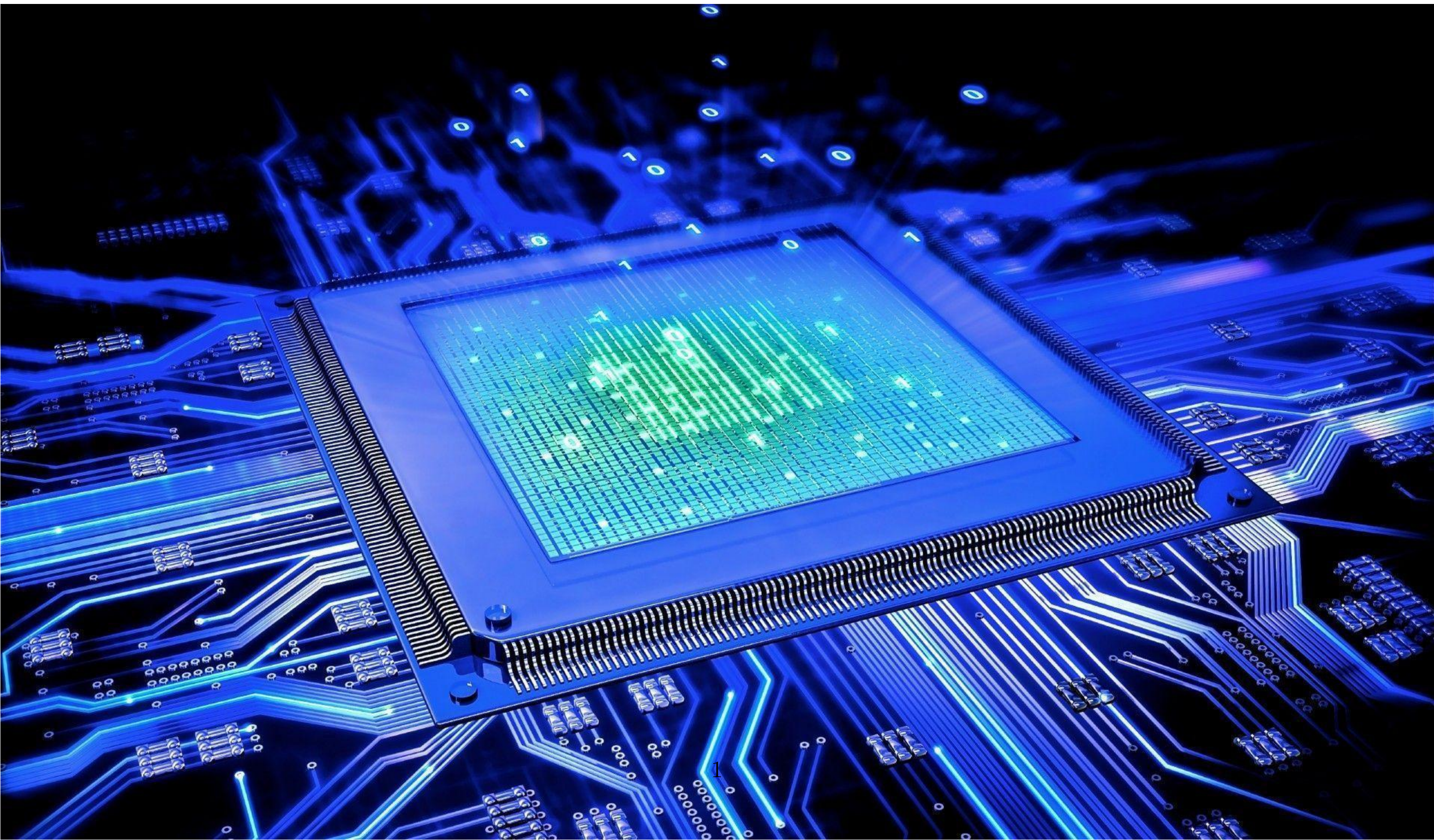
Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών &
Μηχανικών Υπολογιστών

Εργαστήριο Μικροϋπολογιστών

3^{η} Εργαστηριακή Αναφορά - EasyAVR6

Δήμος Δημήτρης (031 17 165)
Αμπατζή Ναυσικά (031 17 198)
7ο Εξάμηνο - Ροή Υ
Φθινόπωρο 2020



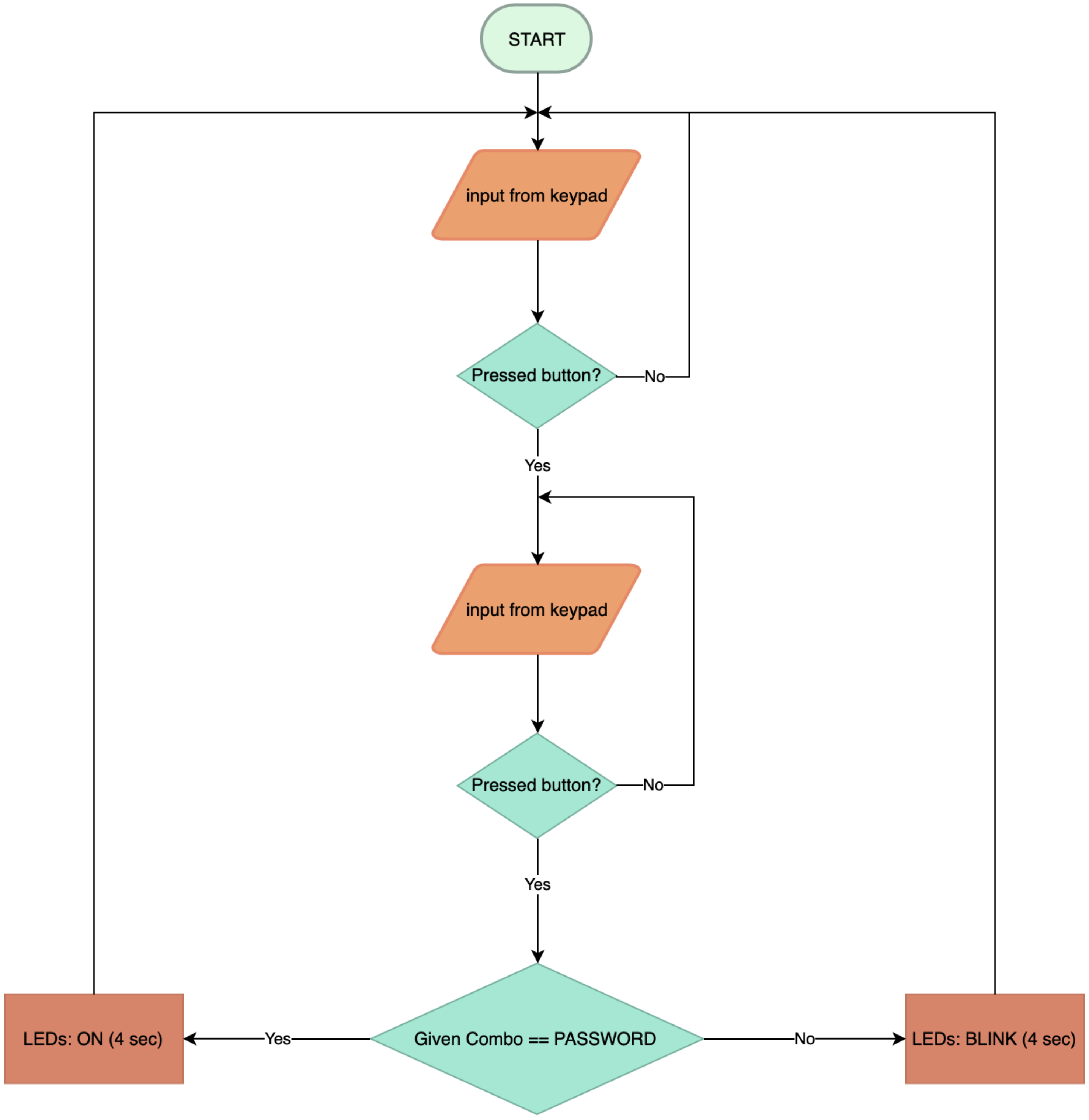
1 Άσκηση 3.1 - Ηλεκτρονική Κλειδαριά (C)

```
#define F_CPU 8000000UL
#define SPARK_DELAY 15
#define PASSWORD 13

#include <avr/io.h>
#include <util/delay.h>

unsigned char scan_row_sim(int);
unsigned int scan_keypad_sim(void);
unsigned int scan_keypad_rising_edge_sim(void);
unsigned char keypad_to_ascii_sim(unsigned int);

unsigned char temp;
unsigned int previous_state = 0;
unsigned char one_two;

int main(void) {

        unsigned char first, second;
        unsigned int given_combo;

        /* IO Settings */
        DDRB = 0xFF;
        DDRC = 0xF0;

        /* Main Loop */
    while(1) {

                do {    // wait, until a button is pressed
                        first = scan_keypad_rising_edge_sim();
                }while(!first);
                // transform into the right form
        first = keypad_to_ascii_sim(first) - 0x30;

                do {    // wait until a button is pressed
                        second = scan_keypad_rising_edge_sim();
                }while(!second);
                // transform into the right form
                second = keypad_to_ascii_sim(second) - 0x30;

                given_combo = first*10 + second;
                // given combo: correct -> LEDs: ON
                if(given_combo == PASSWORD) {
                        PORTB = 0xff;
                        _delay_ms(4000);
                        PORTB = 0x00;
                }
                // given combo: incorrect -> LEDs blink for 4 secs
                else {
                        for(int i = 0; i < 4; ++i) {
                                PORTB = 0xff;
                                _delay_ms(500);
                                PORTB = 0x00;
                                _delay_ms(500);
                        }
                }
                // no actual need - bug fix
                scan_keypad_rising_edge_sim();
        }
}


unsigned char scan_row_sim(int row) {
    volatile unsigned char pressed_row;

        temp = 0x08;
        PORTC = temp << row;

        _delay_us(500);
        asm("nop");
        asm("nop");
        pressed_row = PINC & 0x0f;

        return pressed_row;
}
```

```
unsigned int scan_keypad_sim(void) {
        unsigned int row1, row2, row3, row4;
    int pressed_button = 0x00;

        row1 = scan_row_sim(1);
        row2 = scan_row_sim(2);
        row3 = scan_row_sim(3);
        row4 = scan_row_sim(4);

        pressed_button = (row1 << 4) | (row2);
        if(pressed_button)
                one_two = 1;
        else {
                one_two = 0;
                pressed_button = (row3 <<4 ) | (row4);
        }
        PORTC = 0x00;
        return pressed_button;
}


unsigned int scan_keypad_rising_edge_sim(void) {

        unsigned int button1, button2;
        unsigned int current_state, final_state;

        button1 = scan_keypad_sim();
        _delay_ms(SPARK_DELAY);
        button2 = scan_keypad_sim();
        current_state = button1 & button2;
        final_state = current_state & (~ previous_state);
        previous_state = current_state;

        return final_state;
}
```

```
unsigned char keypad_to_ascii_sim(unsigned int final_state) {

        if(one_two) {
                switch (final_state) {
                        case 0x10:
                                return '1';
                        case 0x20:
                                return '2';
                        case 0x40:
                                return '3';
                        case 0x80:
                                return 'A';
                        case 0x01:
                                return '4';
                        case 0x02:
                                return '5';
                        case 0x04:
                                return '6';
                        case 0x08:
                                return 'B';
                }
        }
        else {
                switch(final_state){
                        case 0x10:
                                return '7';
                        case 0x20:
                                return '8';
                        case 0x40:
                                return '9';
                        case 0x80:
                                return 'C';
                        case 0x01:
                                return '*';
                        case 0x02:
                                return '0';
                        case 0x04:
                                return '#';
                        case 0x08:
```

```
                                        return 'D';
                    }
          }
}
```


START
input from keypad
Pressed button?
No
Yes
input from keypad
Pressed button?
No
Yes
Given Combo == PASSWORD
Yes
LEDs: ON (4 sec)
No
LEDs: BLINK (4 sec)

2 Άσκηση 3.2 - Ηλεκτρονική Κλειδαριά (assembly)

```
.include "m16def.inc"

.DSEG
        _tmp_: .byte 2

.CSEG
        .org 0x0
        rjmp MAIN

; -------------------------------
; -------- DEFINITIONS ---------
; -------------------------------
.equ one   = 0x10
.equ three = 0x40
.equ password = 0x0D
.def temp = r20
.def pressed  = r24
.def first  = r21
.def second = r22
.def counter = r16



; -------------------------------
; --------- ROUTINES -----------
; -------------------------------
leds_on: ser temp
                 out PORTB,temp
                 ret



leds_off: clr temp
                  out PORTB,temp
                  ret



scan_row_sim:
        out PORTC, r25
        push r24
        push r25
        ldi r24,low(500)
        ldi r25,high(500)
        rcall wait_usec
        pop r25
        pop r24
        nop
        nop
        in r24, PINC
        andi r24 ,0x0f
        ret


scan_keypad_sim:
        push r26
        push r27
        ldi r25 , 0x10
        rcall scan_row_sim
        swap r24
        mov r27, r24
        ldi r25 ,0x20
        rcall scan_row_sim
        add r27, r24
        ldi r25 , 0x40
        rcall scan_row_sim
        swap r24
        mov r26, r24
        ldi r25 ,0x80
        rcall scan_row_sim
        add r26, r24
        movw r24, r26
        clr r26
        out PORTC,r26
        pop r27
        pop r26
        ret
```

```
scan_keypad_rising_edge_sim:
        push r22
        push r23
        push r26
        push r27
        rcall scan_keypad_sim
        push r24
        push r25
        ldi r24 ,15
        ldi r25 ,0
        rcall wait_msec
        rcall scan_keypad_sim
        pop r23
        pop r22
        and r24 ,r22
        and r25 ,r23
        ldi r26 ,low(_tmp_)
        ldi r27 ,high(_tmp_)
        ld r23 ,X+
        ld r22 ,X
        st X ,r24
        st -X ,r25
        com r23
        com r22
        and r24 ,r22
        and r25 ,r23
        pop r27
        pop r26
        pop r23
        pop r22
        ret


keypad_to_ascii_sim:
        push r26
        push r27
        movw r26 ,r24
        ldi r24 ,'*'
        sbrc r26 ,0
```

```
rjmp return_ascii
ldi r24 ,'0'
sbrc r26 ,1
rjmp return_ascii
ldi r24 ,'#'
sbrc r26 ,2
rjmp return_ascii
ldi r24 ,'D'
sbrc r26 ,3
rjmp return_ascii
ldi r24 ,'7'
sbrc r26 ,4
rjmp return_ascii
ldi r24 ,'8'
sbrc r26 ,5
rjmp return_ascii
ldi r24 ,'9'
sbrc r26 ,6
rjmp return_ascii
ldi r24 ,'C'
sbrc r26 ,7
rjmp return_ascii
ldi r24 ,'4'
sbrc r27 ,0
rjmp return_ascii
ldi r24 ,'5'
sbrc r27 ,1
rjmp return_ascii
ldi r24 ,'6'
sbrc r27 ,2
rjmp return_ascii
ldi r24 ,'B'
sbrc r27 ,3
rjmp return_ascii
ldi r24 ,'1'
sbrc r27 ,4
rjmp return_ascii
ldi r24 ,'2'
sbrc r27 ,5
```

```
        rjmp return_ascii
        ldi r24 ,'3'
        sbrc r27 ,6
        rjmp return_ascii
        ldi r24 ,'A'
        sbrc r27 ,7
        rjmp return_ascii
        clr r24
        rjmp return_ascii
        return_ascii:
        pop r27
        pop r26
        ret


write_2_nibbles_sim:
        push r24
        push r25
        ldi r24 ,low(6000)
        ldi r25 ,high(6000)
        rcall wait_usec
        pop r25
        pop r24
        push r24
        in r25, PIND
        andi r25, 0x0f
        andi r24, 0xf0
        add r24, r25
        out PORTD, r24
        sbi PORTD, PD3
        cbi PORTD, PD3
        push r24
        push r25
        ldi r24 ,low(6000)
        ldi r25 ,high(6000)
        rcall wait_usec
        pop r25
        pop r24
        pop r24
        swap r24
        andi r24 ,0xf0
        add r24, r25
        out PORTD, r24
        sbi PORTD, PD3
        cbi PORTD, PD3
        ret


lcd_data_sim:
        push r24
        push r25
        sbi PORTD, PD2
        rcall write_2_nibbles_sim
        ldi r24 ,43
        ldi r25 ,0
        rcall wait_usec
        pop r25
        pop r24
        ret


lcd_command_sim:
        push r24
        push r25
        cbi PORTD, PD2
        rcall write_2_nibbles_sim
        ldi r24, 39
        ldi r25, 0
        rcall wait_usec
        pop r25
        pop r24
        ret


lcd_init_sim:
        push r24
        push r25
        ldi r24, 40
        ldi r25, 0
        rcall wait_msec
        ldi r24, 0x30
        out PORTD, r24
        sbi PORTD, PD3
        cbi PORTD, PD3
        ldi r24, 39
        ldi r25, 0
        rcall wait_usec
        push r24
        push r25
        ldi r24,low(1000)
        ldi r25,high(1000)
        rcall wait_usec
        pop r25
        pop r24
        ldi r24, 0x30
        out PORTD, r24
        sbi PORTD, PD3
        cbi PORTD, PD3
        ldi r24,39
        ldi r25,0
        rcall wait_usec
        push r24
        push r25
        ldi r24 ,low(1000)
        ldi r25 ,high(1000)
        rcall wait_usec
        pop r25
        pop r24
        ldi r24,0x20
        out PORTD, r24
        sbi PORTD, PD3
        cbi PORTD, PD3
        ldi r24,39
        ldi r25,0
        rcall wait_usec
        push r24
        push r25
```

```
        ldi r24 ,low(1000)
        ldi r25 ,high(1000)
        rcall wait_usec
        pop r25
        pop r24
        ldi r24,0x28
        rcall lcd_command_sim
        ldi r24,0x0c
        rcall lcd_command_sim
        ldi r24,0x01
        rcall lcd_command_sim
        ldi r24, low(1530)
        ldi r25, high(1530)
        rcall wait_usec
        ldi r24 ,0x06
        rcall lcd_command_sim
        pop r25
        pop r24
        ret


wait_msec:
        push r24
        push r25
        ldi r24,low(998)
        ldi r25,high(998)
        rcall wait_usec
        pop r25
        pop r24
        sbiw r24 , 1
        brne wait_msec
        ret


wait_usec:
        sbiw r24,1
        nop
        nop
        nop
        nop
        brne wait_usec
        ret


; ------------------------------------
; --------- MAIN PROGRAM -----------
; ------------------------------------
MAIN:

        ; stack initialization
        ldi temp,LOW(RAMEND)
        out SPL, temp
        ldi temp,HIGH(RAMEND)
        out SPH, temp


        ; I/O definition
        ldi temp, (1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
        out DDRC, temp
        ser temp
        out DDRB, temp
        out DDRD, temp

; ---------------------------------
; --------- MAIN LOOP -----------
; ---------------------------------
START:  clr r24
                rcall lcd_init_sim      ; clear LCD Display
                rcall leds_off          ; clear LEDs
                clr counter                     ; clear counter



                ; --------------------
                ; ------- INPUT -------
                ; --------------------
digit1: rcall scan_keypad_rising_edge_sim
                rcall keypad_to_ascii_sim
                cpi pressed, 0x00
```

```
            breq digit1

            subi pressed, 0x30
            mov first, pressed



digit2: rcall scan_keypad_rising_edge_sim
            rcall keypad_to_ascii_sim
            cpi pressed, 0x00
            breq digit2

            subi pressed, 0x30
            mov second, pressed


            ; check if input matches password
            ldi temp, 0x0a
            mul first, temp
            mov first, r0
            add first, second
            cpi first, password
            brne wrong_pass


            ; ---------------------------------
            ; ------ CORRECT GIVEN COMBO ------
            ; ---------------------------------
            rcall leds_on   ; set LEDs: ON

            ; display "WELCOME XX" on LCD Display
            ldi r24, 'W'
            rcall lcd_data_sim
            ldi r24, 'E'
            rcall lcd_data_sim
            ldi r24, 'L'
            rcall lcd_data_sim
            ldi r24, 'C'
            rcall lcd_data_sim
```

```
                 ldi r24, 'O'
                 rcall lcd_data_sim
                 ldi r24, 'M'
                 rcall lcd_data_sim
                 ldi r24, 'E'
                 rcall lcd_data_sim
                 ldi r24, ' '
                 rcall lcd_data_sim
                 ldi r24, '1'
                 rcall lcd_data_sim
                 ldi r24, '3'
                 rcall lcd_data_sim

                 ; delay for 4 sec
                 ldi r24,low(4000)
                 ldi r25,high(4000)
                 rcall wait_msec

                 ; no actual need - bug fix
                 rcall scan_keypad_rising_edge_sim
                 rjmp START


                 ; -------------------------------------
                 ; ------ INCORRECT GIVEN COMBO ------
                 ; -------------------------------------
wrong_pass:      rcall scan_keypad_rising_edge_sim

                 ; display "ALARM ON" on LCD Display
                 ldi r24, 'A'
                 rcall lcd_data_sim
                 ldi r24, 'L'
                 rcall lcd_data_sim
                 ldi r24, 'A'
                 rcall lcd_data_sim
                 ldi r24, 'R'
                 rcall lcd_data_sim
                 ldi r24, 'M'
                 rcall lcd_data_sim
```

```
                ldi r24, ' '
                rcall lcd_data_sim
                ldi r24, 'O'
                rcall lcd_data_sim
                ldi r24, 'N'
                rcall lcd_data_sim

loop_:  ; set LEDs: ON for 0.5 sec
        rcall leds_on
        push r24
        push r25
        ldi r24,low(500)
        ldi r25,high(500)
        rcall wait_msec
        pop r25
        pop r24

        ; set LEDs: OFF for 0.5 sec
        rcall leds_off
        push r24
        push r25
        ldi r24,low(500)
        ldi r25,high(500)
        rcall wait_msec
        pop r25
        pop r24
        inc counter
        cpi counter,0x04

        ; loop 4 times (2*0,5)*4 = 4 sec
        brne loop_
        ; then reinitialize counter
        clr counter

        ; no actual need - bug fix
        rcall scan_keypad_rising_edge_sim
        rjmp START
```