Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Μικροϋπολογιστών
Χειμερινό Εξάμηνο 2021-2022

4η Εργαστηριακή Άσκηση
Ηλεκτρονική κλειδαριά με αισθητήρα CO

Μέλος 1: Λαγός Αναστάσιος - 03113531
Μέλος 2: Αντώνιος Δημήτριος Αλικάρης - 03118062

1. Assembly

Σημειώσεις

Οι ρουτινές εξυπηρέτησης των διακοπών του ADC και του timer1 υλοποιήθηκαν έτσι ώστε να μεταβάλλουν και να αποθηκεύουν μόνο την κατάσταση των LEDs αερίου, χωρίς να δίνουν έξοδο στα LEDs. Αυτή η έξοδος γίνεται στην κύρια ρουτίνα.
Κατά τον συναγερμό (>70ppm) τα LEDs αερίου πρέπει να αναβοσβήνουν. Αυτό υλοποιήθηκε με την βοήθεια του T flag του SREG, του οποίου η τιμή γίνεται 0x00→ 0x01 ή 0x01→ 0x00 κάθε 100ms. Κατά την πληκτρολόγηση λάθους κωδικού, η χρονοκαθυστέρηση έχει χωριστεί ως εξής:

$$4 \times (0.1 \times 5 + 0.1 \times 5)$$

Έτσι κάθε 100ms που έχουμε καινούργια έξοδο από τον ADC, η τιμή αυτή ανανεώνεται στα LEDs.

Κώδικας

```
.DSEG
_tmp_:.byte 2

.DEF temp = r16         ;used for cpc instruction
.DEF clear_lcd = r17 ;indicates if the message "CLEAR" is on the LCD (clear_lcd == 0x01)
↪  or not (clear_lcd == 0x00)
.DEF gas_led = r18 ;stores the states of the gas_leds (PB0-PB6). We divided 1024(maximum
↪  number the ADC can read) by 8 = 128. That means 128 gas levels. With the first
↪  (<128) being all gas_leds off.
.DEF gas_detected = r19 ;indicates if the message "GAS DETECTED" is on the LCD
↪  (gas_detected == 0x01) or not (gas_detected == 0x00)

.CSEG
.include "m16def.inc"
;Define RESET,TIMER1 overflow and ADC routines
.org 0x00
rjmp main
.org 0x10
rjmp ISR_TIMER1_OVF
.org 0x1C
```

```
17  rjmp ISR_ADC
18
19  main:
20          ldi r24, low(RAMEND)        ;Initialize stack pointer
21          out SPL, r24
22          ldi r24, high(RAMEND)   ;RAMEND is defined in m16def.inc
23          out SPH, r24
24
25          clr temp
26          clr clear_lcd
27          clr gas_led
28          clr gas_detected
29          clr r28
30          clr r20
31          clr r21
32          ;T flag of SREG is used for flicking the gas leds
33          clt
34
35          ldi r24, (1 << PC7)|(1 << PC6)|(1 << PC5)|(1 << PC4)        ;Initialize 4 MSB of
            ↪  PORTC as outputs
36          out DDRC, r24
37
38          ser r24
39          out DDRB, r24                       ;Initialize LEDs
40          out DDRD, r24                       ;Initialize PORTD(LCD) as output
41
42          rcall ADC_init                       ;Initialize ADC
43          rcall TCNT1_init                ;Initialize timer1
44
45          clr r24
46          rcall lcd_init_sim                ;Clear LCD
47
48          sei                                                 ;Enable global interrupts
49
50          first_button:
51                  rcall scan_keypad_rising_edge_sim
52          mov r22, r24
53          or r22, r25
54                  cpi r22, 0
55                  ;The cpu spends most of the time in this code block therefore we output
                    ↪  any changes to the gas LEDs states performed by the ADC interrupt.
56                  out PORTB, gas_led
57
58                  breq first_button
59      first_ascii:
60                  rcall keypad_to_ascii_sim        ;Convert to ASCII for comparison
61                  mov r20, r24                ;first button
62
63          second_button:
64                  rcall scan_keypad_rising_edge_sim
65          mov r22, r24
66          or r22, r25
67          cpi r22, 0
```

```
68                ;The cpu might spend some time in this code block therefore we output
                  ↪  any changes to the gas LEDs states performed by the ADC interrupt.
69                out PORTB, gas_led
70
71                breq second_button
72        second_ascii:
73                rcall keypad_to_ascii_sim        ;Convert to ASCII for comparison
74                mov r21, r24                ;second button
75
76            check:        ;Check if the input code is correct. For our team, that's "71"
77                cpi r20, '7'
78                brne wrong
79            cpi r21, '1'
80            brne wrong
81                ;Past this point we have a correct password
82            correct:
83                cli                                        ;Stop global interrupts. We
                  ↪  don't want any of the gas detection features during this part
84
85            rcall lcd_init_sim        ;Clear LCD
86
87            ldi r24, 'W'
88            rcall lcd_data_sim
89            ldi r24, 'E'
90            rcall lcd_data_sim
91            ldi r24, 'L'
92            rcall lcd_data_sim
93            ldi r24, 'C'
94            rcall lcd_data_sim
95            ldi r24, 'O'
96            rcall lcd_data_sim
97            ldi r24, 'M'
98            rcall lcd_data_sim
99            ldi r24, 'E'
100           rcall lcd_data_sim
101
102           rcall scan_keypad_rising_edge_sim   ;for successful remote operation
103
104                   ldi r24, 0x80                ;Light up
105
106                   out PORTB, r24                ;the MSB of leds PB
107
108                   ldi r24, low(4000)
109                   ldi r25, high(4000)
110                   rcall wait_msec                ;Wait for 4 secs
111
112                   clr r24                                ;Turn off
113                   out PORTB, r24                ;the MSB of leds PB
114                   rcall lcd_init_sim        ;Clear "WELCOME"
115                   clr gas_detected        ;LCD is empty therefore this should be 0x00
116                   clr clear_lcd                ;LCD is empty therefore this should be 0x00
117                   sei                                        ;Enable global interrupts
                      ↪  again
118                   jmp first_button
```

```
119
120         wrong:
121         rcall scan_keypad_rising_edge_sim   ;for successful remote operation
122
123                 ldi r24,4                       ;Number of loops
124     wrong_loop:                                ;4 loops of 0.5 sec off 0.5 on
125                 push r24                        ;Save number of loops
126                 ;TURN ON
127                 ldi r24, 0x80
128                 or r24, gas_led                 ;Get state of gas leds
129                 out PORTB, r24                 ;Output to leds, both MSB and gas
130                 ldi r24, low(100)
131                 ldi r25, high(100)
132                 rcall wait_msec                 ;Wait 0.1 sec
133
134                 ldi r24, 0x80
135                 or r24, gas_led                 ;Get state of gas leds
136                 out PORTB, r24                 ;Output to leds, both MSB and gas
137                 ldi r24, low(100)
138                 ldi r25, high(100)
139                 rcall wait_msec                 ;Wait 0.1 sec
140
141                 ldi r24, 0x80
142                 or r24, gas_led                 ;Get state of gas leds
143                 out PORTB, r24                 ;Output to leds, both MSB and gas
144                 ldi r24, low(100)
145                 ldi r25, high(100)
146                 rcall wait_msec                 ;Wait 0.1 sec
147
148                 ldi r24, 0x80
149                 or r24, gas_led                 ;Get state of gas leds
150                 out PORTB, r24                 ;Output to leds, both MSB and gas
151                 ldi r24, low(100)
152                 ldi r25, high(100)
153                 rcall wait_msec                 ;Wait 0.1 sec
154
155                 ldi r24, 0x80
156                 or r24, gas_led                 ;Get state of gas leds
157                 out PORTB, r24                 ;Output to leds, both MSB and gas
158                 ldi r24, low(100)
159                 ldi r25, high(100)
160                 rcall wait_msec                 ;Wait 0.1 sec
161                 ;TURN OFF
162                 mov r24, gas_led
163                 out PORTB, r24
164                 ldi r24, low(100)
165                 ldi r25, high(100)
166                 rcall wait_msec                 ;Wait 0.1 sec
167
168                 mov r24, gas_led
169                 out PORTB, r24
170                 ldi r24, low(100)
171                 ldi r25, high(100)
172                 rcall wait_msec                 ;Wait 0.1 sec
```

```
173
174                 mov r24, gas_led
175                 out PORTB, r24
176                 ldi r24, low(100)
177                 ldi r25, high(100)
178                 rcall wait_msec                 ;Wait 0.1 sec
179
180                 mov r24, gas_led
181                 out PORTB, r24
182                 ldi r24, low(100)
183                 ldi r25, high(100)
184                 rcall wait_msec                 ;Wait 0.1 sec
185
186                 mov r24, gas_led
187                 out PORTB, r24
188                 ldi r24, low(100)
189                 ldi r25, high(100)
190                 rcall wait_msec                 ;Wait 0.1 sec
191
192                 mov r24, gas_led
193                 out PORTB, r24
194
195                 pop r24
196                 subi r24, 1                         ;Subtract one loop
197                 brne wrong_loop
198
199
200         jmp first_button
201
202         wait_msec:
203                 push r24
204                 push r25
205                 ldi r24, low(1000)
206                 ldi r25, high(1000)
207                 rcall wait_usec
208                 pop r25
209                 pop r24
210                 sbiw r24, 1
211                 brne wait_msec
212
213                 ret
214
215         wait_usec:
216                 sbiw r24, 1 ;2 cycles
217                 nop
218                 nop
219                 nop
220                 nop
221                 brne wait_usec ;1 cycle the majority of the time
222
223                 ret
224         ;ADC initialization routine
225         ADC_init:
226                 ldi r24, (1 << REFS0)           ;Vcc = 5V
```

```
227             out ADMUX, r24
228
229             ldi r24, (1 << ADEN) | (1 << ADIE) | (1 << ADPS2) | (1 << ADPS1) | (1 <<
                ↪  ADPS0)        ;Enable ADC, Enable ADC interrupts, prescaler = 128
230             out ADCSRA, r24
231             ret
232         ;TIMER1 initialization routine
233         TCNT1_init:
234             ldi r24, (1 << TOIE1)        ;Enable interrupts for timer1
235             out TIMSK, r24
236
237             ldi r24, (1 << CS12) | (0 << CS11) | (1 << CS10)        ;Set prescaler
                ↪  at 1024.
238             out TCCR1B, r24
239
240             ;Clock at 8MHz
241             ;8MHz / 1024(prescale) = 7812.5Hz
242             ;Interrupt every 100ms | interrupt every 0.1 * 7812.5 = 781.25 cycles
243             ;Overflow happens at 65536 cycles. Therefore start counting from 65536 -
                ↪  781.25 = 64755 = 0xFCF3
244             ldi r24, 0xFC
245             out TCNT1H, r24
246             ldi r24, 0xF3
247             out TCNT1L, r24
248
249             ret
250         ;TIMER1 overflow ISR
251         ISR_TIMER1_OVF:
252             push r24
253             ;Here we implement the gas LEDs flickering if the alarm is on. The T
                ↪  flag of SREG is used for determining whether the gas LEDs should be
                ↪  on/off(T flag = 0x00/0x01) for the next 100ms.
254             cpi r28, 0x01         ;r28 is indicative of whether the alarm (over
                ↪  70ppm) is on or off (0x01, 0x00)
255             brne continue_1         ;If the alarm is off, set T and continue with ISR
256             ;If the alarm is on then
257             brtc continue_1         ;If T was clear(gas_leds off) then they should be
                ↪  ON for the next 100ms
258             clt         ;If T was set (gas_leds on) then they should be OFF for the
                ↪  next 100ms.
259             jmp continue_2
260
261         continue_1:
262             set
263         continue_2:
264             in r24, SREG        ;Of course with have to save the state of SREG after
                ↪  we modify T.
265             push r24
266             ldi r24, (1 << CS12) | (0 << CS11) | (1 << CS10)
267             out TCCR1B, r24
268
269             ldi r24, 0xFC                       ;Initialize counter at 64755
270             out TCNT1H, r24
271             ldi r24, 0xF3
```

```
272             out TCNT1L, r24
273
274             in r24, ADCSRA
275             ori r24, (1 << ADSC)        ;Start AD Conversion
276             out ADCSRA, r24
277
278             pop r24                                     ;restore SREG and r24
279             out SREG, r24
280             pop r24
281
282             reti
283
284     ISR_ADC:
285             push r25
286             push r24
287             in r24, SREG
288             push r24
289
290             in r24, ADCL                        ;r24,r25 store the output of the ADC
291             in r25, ADCH
292
293             cpi r24, 0x80                        ; adc < 128
294             cpc r25, r1
295             brcs long_jump_under_128;Too far for relative jump
296
297             cpi r24, 0xCD
298             cpc r25, r1                                 ; adc < 205 (70ppm)
299             brcs long_jump_under_205
300             ;This code is over 70ppm
301             ldi r28, 0x01                       ;Alarm is on
302
303             brts continue_adc               ;If T is set then just calculate the
                ↪ new gas_led state.
304             clr gas_led                                 ;Else, a cleared T indicates
                ↪ that the gas_leds should be off
305             jmp ISR_ADC_EXIT                ;So exit the ISR
306     long_jump_under_128:
307             jmp under_128
308     long_jump_under_205:
309             jmp under_205
310     continue_adc:
311             cpi gas_detected, 0x01        ;If "GAS DETECTED" message is on LCD, no
                ↪ need to show it again
312             breq dont_show_LCD
313             ;If not on LCD then show it
314             rcall lcd_init_sim               ; Clear LCD
315
316             ldi r24, 'G'
317             rcall lcd_data_sim
318             ldi r24, 'A'
319             rcall lcd_data_sim
320             ldi r24, 'S'
321             rcall lcd_data_sim
322             ldi r24, ' '
```

```
323             rcall lcd_data_sim
324             ldi r24, 'D'
325             rcall lcd_data_sim
326             ldi r24, 'E'
327             rcall lcd_data_sim
328             ldi r24, 'T'
329             rcall lcd_data_sim
330             ldi r24, 'E'
331             rcall lcd_data_sim
332             ldi r24, 'C'
333             rcall lcd_data_sim
334             ldi r24, 'T'
335             rcall lcd_data_sim
336             ldi r24, 'E'
337             rcall lcd_data_sim
338             ldi r24, 'D'
339             rcall lcd_data_sim
340
341             ldi gas_detected, 0x01        ;"GAS DETECTED" on LCD, dont come back
                ↪ here unless its off!
342             clr clear_lcd                 ;"CLEAR" is not on LCD.
343         dont_show_LCD:
344
345             cp r24, r1                        ; adc < 256
346             ldi temp, 0x01
347             cpc r25, temp
348             brcs under_256
349
350             cpi r24, 0x80                     ; adc < 384
351             ldi temp, 0x01
352             cpc r25, temp
353             brcs under_384
354
355             cp r24, r1                        ; adc < 512
356             ldi temp, 0x02
357             cpc r25, temp
358             brcs under_512
359
360             cpi r24, 0x01                     ; adc < 640
361             ldi temp, 0x02
362             cpc r25, temp
363             brcs under_640
364
365             cp r24, r1                        ; adc < 768
366             ldi temp, 0x03
367             cpc r25, temp
368             brcs under_768
369
370             cpi r24, 0x01                     ; adc < 896
371             ldi temp, 0x03
372             cpc r25, temp
373             brcs under_896
374
375             ldi gas_led, 0x7F
```

```
376             jmp ISR_ADC_EXIT
377     under_128:
378             clr r28
379             cpi clear_lcd, 0x01                ;If "CLEAR" message is on LCD, no
                ↪  need to show it again
380             breq dont_show_clear_128
381
382             rcall lcd_init_sim                 ; Clear LCD
383
384             ldi r24, 'C'
385             rcall lcd_data_sim
386             ldi r24, 'L'
387             rcall lcd_data_sim
388             ldi r24, 'E'
389             rcall lcd_data_sim
390             ldi r24, 'A'
391             rcall lcd_data_sim
392             ldi r24, 'R'
393             rcall lcd_data_sim
394
395             ldi clear_lcd, 0x01                ;"CLEAR" on LCD, dont come back here
                ↪  unless its off!
396
397     dont_show_clear_128:
398             clr gas_detected                 ;"GAS DETECTED" is not on LCD.
399             clr gas_led
400             jmp ISR_ADC_EXIT
401
402     under_205:
403             clr r28
404             cpi clear_lcd, 0x01
405             breq dont_show_clear_205
406
407             rcall lcd_init_sim                 ; Clear LCD
408
409             ldi r24, 'C'
410             rcall lcd_data_sim
411             ldi r24, 'L'
412             rcall lcd_data_sim
413             ldi r24, 'E'
414             rcall lcd_data_sim
415             ldi r24, 'A'
416             rcall lcd_data_sim
417             ldi r24, 'R'
418             rcall lcd_data_sim
419
420             ldi clear_lcd, 0x01
421
422     dont_show_clear_205:
423             clr gas_detected
424             ldi gas_led, 0x01
425             jmp ISR_ADC_EXIT
426
427     under_256:
```

```
428             ldi gas_led, 0x01
429             jmp ISR_ADC_EXIT
430
431         under_384:
432             ldi gas_led, 0x03
433             jmp ISR_ADC_EXIT
434
435         under_512:
436             ldi gas_led, 0x07
437             jmp ISR_ADC_EXIT
438
439         under_640:
440             ldi gas_led, 0x0F
441             jmp ISR_ADC_EXIT
442
443         under_768:
444             ldi gas_led, 0x1F
445             jmp ISR_ADC_EXIT
446
447         under_896:
448             ldi gas_led, 0x3F
449
450         ISR_ADC_EXIT:
451             pop r24
452             out SREG, r24
453             pop r24
454             pop r25
455
456             reti
457
458
459         scan_row_sim:
460         out PORTC, r25 ; ? ?????????? ?????? ??????? ??? ?????? '1'
461         push r24 ; ????? ?????? ??? ?????????? ??? ?? ?????
462         push r25 ; ?????????? ??? ??????????? ?????????????
463         ldi r24,low(500) ; ?????????
464         ldi r25,high(500)
465         rcall wait_usec
466         pop r25
467         pop r24 ; ????? ????? ??????
468         nop
469         nop ; ?????????? ??? ?? ???????? ?? ????? ? ?????? ??????????
470         in r24, PINC ; ?????????? ?? ?????? (??????) ??? ????????? ??? ????? ?????????
471         andi r24 ,0x0f ; ???????????? ?? 4 LSB ???? ?? '1' ???????? ??? ????? ?????????
472         ret ; ?? ?????????
473
474         scan_keypad_sim:
475         push r26 ; ?????????? ???? ?????????? r27:r26 ????? ????
476         push r27 ; ????????? ???? ???? ???????
477         ldi r25 , 0x10 ; ?????? ??? ????? ?????? ??? ???????????? (PC4: 1 2 3 A)
478         rcall scan_row_sim
479         swap r24 ; ?????????? ?? ??????????
480         mov r27, r24 ; ??? 4 msb ??? r27
481         ldi r25 ,0x20 ; ?????? ?? ??????? ?????? ??? ???????????? (PC5: 4 5 6 B)
```

```
482         rcall scan_row_sim
483         add r27, r24 ; ?????????? ?? ?????????? ??? 4 lsb ??? r27
484         ldi r25 , 0x40 ; ?????? ??? ????? ?????? ??? ???????????? (PC6: 7 8 9 C)
485         rcall scan_row_sim
486         swap r24 ; ?????????? ?? ??????????
487         mov r26, r24 ; ??? 4 msb ??? r26
488         ldi r25 ,0x80 ; ?????? ??? ??????? ?????? ??? ???????????? (PC7: * 0 # D)
489         rcall scan_row_sim
490         add r26, r24 ; ?????????? ?? ?????????? ??? 4 lsb ??? r26
491         movw r24, r26 ; ???????? ?? ?????????? ????? ?????????? r25:r24
492         clr r26 ; ?????????? ??? ??? ???????????? ????????
493         out PORTC,r26 ; ?????????? ??? ??? ???????????? ????????
494         pop r27 ; ????????? ???? ?????????? r27:r26
495         pop r26
496         ret
497
498         scan_keypad_rising_edge_sim:
499         push r22 ; ?????????? ???? ?????????? r23:r22 ??? ????
500         push r23 ; r26:r27 ????? ???? ????????? ???? ???? ???????
501         push r26
502         push r27
503         rcall scan_keypad_sim ; ?????? ?? ??????????? ??? ?????????? ?????????
504         push r24 ; ??? ?????????? ?? ??????????
505         push r25
506         ldi r24 ,15 ; ?????????? 15 ms (??????? ????? 10-20 msec ??? ?????????? ???
            ↪  ???
507         ldi r25 ,0 ; ??????????? ??? ???????????? { ???????????? ???????????)
508         rcall wait_msec
509         rcall scan_keypad_sim ; ?????? ?? ??????????? ???? ??? ????????
510         pop r23 ; ??? ??????? ?????????? ??????????
511         pop r22
512         and r24 ,r22
513         and r25 ,r23
514         ldi r26 ,low(_tmp_) ; ??????? ??? ????????? ??? ????????? ????
515         ldi r27 ,high(_tmp_) ; ?????????? ????? ??? ???????? ????? r27:r26
516         ld r23 ,X+
517         ld r22 ,X
518         st X ,r24 ; ?????????? ??? RAM ?? ??? ?????????
519         st -X ,r25 ; ??? ?????????
520         com r23
521         com r22 ; ???? ???? ????????? ??? ????? ≪?????≫ ???????
522         and r24 ,r22
523         and r25 ,r23
524         pop r27 ; ????????? ???? ?????????? r27:r26
525         pop r26 ; ??? r23:r22
526         pop r23
527         pop r22
528         ret
529
530         keypad_to_ascii_sim:
531         push r26 ; ?????????? ???? ?????????? r27:r26 ????? ????
532         push r27 ; ????????? ???? ??? ???????
533         movw r26 ,r24 ; ?????? '1' ???? ?????? ??? ?????????? r26 ????????
534         ; ?? ???????? ??????? ??? ????????
```

```
535         ldi r24 ,'*'
536         ; r26
537         ;C 9 8 7 D # 0 *
538         sbrc r26 ,0
539         rjmp return_ascii
540         ldi r24 ,'0'
541         sbrc r26 ,1
542         rjmp return_ascii
543         ldi r24 ,'#'
544         sbrc r26 ,2
545         rjmp return_ascii
546         ldi r24 ,'D'
547         sbrc r26 ,3 ; ?? ??? ????? '1'?????????? ??? ret, ?????? (?? ????? '1')
548         rjmp return_ascii ; ?????????? ?? ??? ?????????? r24 ??? ASCII ???? ??? D.
549         ldi r24 ,'7'
550         sbrc r26 ,4
551         rjmp return_ascii
552         ldi r24 ,'8'
553         sbrc r26 ,5
554         rjmp return_ascii
555         ldi r24 ,'9'
556         sbrc r26 ,6
557         rjmp return_ascii ;
558         ldi r24 ,'C'
559         sbrc r26 ,7
560         rjmp return_ascii
561         ldi r24 ,'4' ; ?????? '1' ???? ?????? ??? ?????????? r27 ????????
562         sbrc r27 ,0 ; ?? ???????? ??????? ??? ????????
563         rjmp return_ascii
564         ldi r24 ,'5'
565         ;r27
566         ;? 3 2 1 B 6 5 4
567         sbrc r27 ,1
568         rjmp return_ascii
569         ldi r24 ,'6'
570         sbrc r27 ,2
571         rjmp return_ascii
572         ldi r24 ,'B'
573         sbrc r27 ,3
574         rjmp return_ascii
575         ldi r24 ,'1'
576         sbrc r27 ,4
577         rjmp return_ascii ;
578         ldi r24 ,'2'
579         sbrc r27 ,5
580         rjmp return_ascii
581         ldi r24 ,'3'
582         sbrc r27 ,6
583         rjmp return_ascii
584         ldi r24 ,'A'
585         sbrc r27 ,7
586         rjmp return_ascii
587         clr r24
588         rjmp return_ascii
```

```
589         return_ascii:
590         pop r27 ; ????????? ???? ?????????? r27:r26
591         pop r26
592         ret
593
594         write_2_nibbles_sim:
595         push r24 ; ????? ?????? ??? ?????????? ??? ?? ?????
596         push r25 ; ?????????? ??? ?????????? ????????????
597         ldi r24 ,low(6000) ; ?????????
598         ldi r25 ,high(6000)
599         rcall wait_usec
600         pop r25
601         pop r24 ; ????? ????? ??????
602         push r24 ; ??????? ?? 4 MSB
603         in r25, PIND ; ?????????? ?? 4 LSB ??? ?? ???????????
604         andi r25, 0x0f ; ??? ?? ??? ????????? ??? ????? ?????????? ?????????
605         andi r24, 0xf0 ; ???????????? ?? 4 MSB ???
606         add r24, r25 ; ??????????? ?? ?? ???????????? 4 LSB
607         out PORTD, r24 ; ??? ???????? ???? ?????
608         sbi PORTD, PD3 ; ???????????? ?????? Enable ???? ????????? PD3
609         cbi PORTD, PD3 ; PD3=1 ??? ???? PD3=0
610         push r24 ; ????? ?????? ??? ?????????? ??? ?? ?????
611         push r25 ; ?????????? ??? ???????????? ????????????
612         ldi r24 ,low(6000) ; ?????????
613         ldi r25 ,high(6000)
614         rcall wait_usec
615         pop r25
616         pop r24 ; ????? ????? ??????
617         pop r24 ; ??????? ?? 4 LSB. ????????? ?? byte.
618         swap r24 ; ???????????? ?? 4 MSB ?? ?? 4 LSB
619         andi r24 ,0xf0 ; ??? ?? ??? ????? ???? ????????????
620         add r24, r25
621         out PORTD, r24
622         sbi PORTD, PD3 ; ???? ?????? Enable
623         cbi PORTD, PD3
624         ret
625
626         lcd_data_sim:
627         push r24
628         push r25
629         sbi PORTD,PD2
630         rcall write_2_nibbles_sim
631         ldi r24,43
632         ldi r25,0
633         rcall wait_usec
634         pop r25
635         pop r24
636         ret
637
638         lcd_command_sim:
639         push r24 ; ?????????? ???? ?????????? r25:r24 ????? ????
640         push r25 ; ????????? ???? ??? ???????
641         cbi PORTD, PD2 ; ??????? ??? ?????????? ??????? (PD2=0)
642         rcall write_2_nibbles_sim ; ???????? ??? ??????? ??? ??????? 39?sec
```

```
643         ldi r24, 39 ; ??? ??? ?????????? ??? ????????? ??? ??? ??? ??????? ??? lcd.
644         ldi r25, 0 ; ???.: ???????? ??? ???????, ?? clear display ??? return home,
645         rcall wait_usec ; ??? ???????? ????????? ?????????? ??????? ????????.
646         pop r25 ; ????????? ???? ?????????? r25:r24
647         pop r24
648         ret
649
650         lcd_init_sim:
651         push r24 ; ?????????? ???? ?????????? r25:r24 ????? ????
652         push r25 ; ????????? ???? ??? ???????
653
654         ldi r24, 40 ; ???? ? ???????? ??? lcd ???????????? ??
655         ldi r25, 0 ; ????? ??????? ??? ???? ??? ???????????.
656         rcall wait_msec ; ??????? 40 msec ????? ???? ?? ??????????.
657         ldi r24, 0x30 ; ?????? ????????? ?? 8 bit mode
658         out PORTD, r24 ; ?????? ??? ???????? ?? ??????? ???????
659         sbi PORTD, PD3 ; ??? ?? ?????????? ??????? ??? ???????
660         cbi PORTD, PD3 ; ??? ??????, ? ?????? ??????????? ??? ?????
661         ldi r24, 39
662         ldi r25, 0 ; ??? ? ???????? ??? ?????? ????????? ?? 8-bit mode
663         rcall wait_usec ; ??? ?? ?????? ??????, ???? ?? ? ???????? ???? ??????????
664         ; ??????? 4 bit ?? ??????? ?? ?????????? 8 bit
665         push r24 ; ????? ?????? ??? ?????????? ??? ?? ?????
666         push r25 ; ?????????? ??? ??????????? ?????????????
667         ldi r24,low(1000) ; ?????????
668         ldi r25,high(1000)
669         rcall wait_usec
670         pop r25
671         pop r24 ; ????? ????? ??????
672         ldi r24, 0x30
673         out PORTD, r24
674         sbi PORTD, PD3
675         cbi PORTD, PD3
676         ldi r24,39
677         ldi r25,0
678         rcall wait_usec
679         push r24 ; ????? ?????? ??? ?????????? ??? ?? ?????
680         push r25 ; ?????????? ??? ??????????? ?????????????
681         ldi r24 ,low(1000) ; ?????????
682         ldi r25 ,high(1000)
683         rcall wait_usec
684         pop r25
685         pop r24 ; ????? ????? ??????
686         ldi r24,0x20 ; ?????? ?? 4-bit mode
687         out PORTD, r24
688         sbi PORTD, PD3
689         cbi PORTD, PD3
690         ldi r24,39
691         ldi r25,0
692         rcall wait_usec
693         push r24 ; ????? ?????? ??? ?????????? ??? ?? ?????
694         push r25 ; ?????????? ??? ??????????? ?????????????
695         ldi r24 ,low(1000) ; ?????????
696         ldi r25 ,high(1000)
```

```
697         rcall wait_usec
698         pop r25
699         pop r24 ; ????? ????? ??????
700         ldi r24,0x28 ; ??????? ?????????? ???????? 5x8 ????????
701         rcall lcd_command_sim ; ??? ???????? ??? ??????? ???? ?????
702         ldi r24,0x0c ; ??????????? ??? ??????, ???????? ??? ???????
703         rcall lcd_command_sim
704         ldi r24,0x01 ; ?????????? ??? ??????
705         rcall lcd_command_sim
706         ldi r24, low(1530)
707         ldi r25, high(1530)
708         rcall wait_usec
709         ldi r24 ,0x06 ; ??????????? ????????? ??????? ???? 1 ??? ??????????
710         rcall lcd_command_sim ; ??? ????? ??????????? ???? ??????? ?????????? ???
711         ; ????????????? ??? ????????? ????????? ??? ??????
712         pop r25 ; ????????? ???? ?????????? r25:r24
713         pop r24
714                 ret
```

2.C

Σημειώσεις

Στην C η λογική του προγράμματος είναι διαφορετική από αυτή του προγράμματος assembly. Εδώ η έξοδος στα LEDs δίνεται στην ρουτίνα εξυπηρέτησης του ADC.

Κώδικας

```
1  #define F_CPU 8000000UL
2  #include <avr/io.h>
3  #include <util/delay.h>
4  #include <avr/interrupt.h>
5
6
7  #define NOP() {__asm__ __volatile__("nop");}
8
9
10 int temp = 0x0000;
11 int t_flag = 0; // όταν είναι 0 βγαζει στην εξοδο 0, όταν είναι 1 βγάζει το επίπεδο του
   ↪ αερίου
12 int password = 0;//αν είναι 1 (σωστός κωδικός) κάνει το t_flag 1, για να μην
   ↪ αναβοσβήνουν τα 4 δευτερόλεπτα που η ειδική ομάδα είναι στην αίθουσα
13 int leds = 0x00, MSB_led = 0x00; //η κατάσταση του επιπέδου του αερίου και του MLSB της
   ↪ portb
14
15 ISR(TIMER1_OVF_vect) {
16         TCCR1B = (1<<CS12)|(0<<CS11)|(1<<CS10);
17         TCNT1 = 64755; //100ms
18         ADCSRA |= (1<<ADSC);
19
20         if(t_flag == 0)  t_flag = 1;        //ανα 100ms αλλάζουμε αυτήν την τιμή για να
           ↪ αναβοσβήσουμε τα led όταν το επίπεδο του αερίου ξεπεράσει τα όρια
21         else t_flag = 0;
```

```
        if(password == 1) t_flag = 1; //σε περίπτωση σωστού κωδικού σταματάμε την
        ↪  εναλλαγή
}


//Απο τον τύπο Vgas = C*M + Vgas0 βρήκαμε οτι για C = 70ppm, Vgas = 1.
//Επιπλέον η έξοδος του ADC είναι ADC = Vin * (1024/5) = 204.8*Vin, όπου Vin = Vgas.
//Άρα, έχουμε συναγερμό για ADC>205
//Χωρίζουμε τα επίπεδα σε 7, άρα 1024/7 = 147 τιμές το επίπεδο

ISR(ADC_vect){
        if(ADC < 147) PORTB = 0x01 | (MSB_led << 7);
        else if(ADC >= 147 && ADC < 205) PORTB = 0x03 | (MSB_led << 7);
        else if(t_flag == 1){
                if(ADC >= 205 && ADC < 294) leds = 0x03;
                else if(ADC >= 294 && ADC < 441) leds = 0x07;
                else if(ADC >= 441 && ADC < 588) leds = 0x0F;
                else if(ADC >= 588 && ADC < 735) leds = 0x1F;
                else if(ADC >= 735 && ADC < 882) leds = 0x3F;
                else if(ADC >= 882) leds = 0x7F;
                PORTB = leds | (MSB_led << 7);
        }
        else {
                PORTB = MSB_led << 7;
        }
}


void adc_init(){
        ADMUX = (1<<REFS0);
        ADCSRA = (1<<ADEN)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
}

int scan_row_sim(int r25) {
        PORTC = r25;
        int r24;
        _delay_us(500);
        NOP();
        NOP();
        r24 = PINC;
        r24 = r24 & 0x0F;
        return r24;

}

//a is LSB, b is MSB
void scan_keypad_sim(int *a, int *b) {

        int r25 = 0x10, r24, r27, r26;
        r24 = scan_row_sim(r25);
        r27 = ( (r24 & 0x0F) << 4 | (r24 & 0xF0) >> 4); //swap r24

        r25 = 0x20;
        r24 = scan_row_sim(r25);
```

```
        r27 += r24;

        r25 = 0x40;
        r24 = scan_row_sim(r25);
        r26 = ( (r24 & 0x0F) << 4 | (r24 & 0xF0) >> 4); //swap r24

        r25 = 0x80;
        r24 = scan_row_sim(r25);
        r26 += r24;
        *a = r26;
        *b = r27;

        PORTC = 0x00;
        return;
}

void scan_keypad_rising_edge_sim (int time, int *a, int *b) {
        int r22, r23, r24, r25;
         int r26;
        scan_keypad_sim(&r24, &r25);
        _delay_ms(15);
        scan_keypad_sim(&r22, &r23);
        r24 = r24 & r22;
        r25 = r25 & r23;
        r26 = temp;
        r25 = r25 << 8;
        temp = r25 | r24;
        r26 = ~r26;
        *a = r24 & (r26 & 0x00FF);
        *b = (r25 >> 8) & ((r26 & 0xFF00) >> 8);


        return;
        }

int keypad_to_ascii_sim (int r24, int r25) {
        int r26 = '0';
        switch(r24) {
                case 0x01:
                        r26 = '*';
                        break;
                case 0x02:
                        r26 = '0';
                        break;
                case 0x04:
                        r26 = '#';
                        break;
                case 0x08:
                        r26 = 'D';
                        break;
                case 0x10:
                        r26 = '7';
                        break;
                case 0x20:
```

```
129                         r26 = '8';
130                         break;
131                 case 0x40:
132                         r26 = '9';
133                         break;
134                 case 0x80:
135                         r26 = 'C';
136                         break;
137                 }
138         switch(r25) {
139                 case 0x01:
140                 r26 = '4';
141                 break;
142                 case 0x02:
143                 r26 = '5';
144                 break;
145                 case 0x04:
146                 r26 = '6';
147                 break;
148                 case 0x08:
149                 r26 = 'B';
150                 break;
151                 case 0x10:
152                 r26 = '1';
153                 break;
154                 case 0x20:
155                 r26 = '2';
156                 break;
157                 case 0x40:
158                 r26 = '3';
159                 break;
160                 case 0x80:
161                 r26 = 'A';
162                 break;
163         }
164
165         return r26;
166
167 }
168
169 int main(void)
170 {
171     DDRB = 0xFF;                //B as output
172         DDRC = 0xF0;                //PC0-3 input and PC4-PC7 output (for the 4x4
          ↪  keypad)
173         adc_init();
174         TIMSK = (1<<TOIE1);
175         TCCR1B = (1<<CS12)|(0<<CS11)|(1<<CS10);
176         TCNT1 = 64755;
177         sei();
178         /* Replace with your application code */
179     while (1)
180     {
181                  int  r24, r25, c1, c2;
```

```
182
183                 do {
184                         scan_keypad_rising_edge_sim(15, &r24, &r25);
185                         c1 = keypad_to_ascii_sim(r24, r25);
186                 }
187                 while(r24 == 0 && r25 == 0);        //while there is no pressed key,
                    ↪  continue to scan if sth is pressed
188                 do
189                 {
190                         scan_keypad_rising_edge_sim(15, &r24, &r25);
191                         c2 = keypad_to_ascii_sim(r24, r25);
192                 }
193                 while(r24 == 0 && r25 == 0);        //continue scanning for the 2nd
                    ↪  digit
194                 if (c1 == '7' && c2 == '1') {        //if the digit were 7 and then 1
195                         password = 1; //correct password
196                         MSB_led = 1;
197                         PORTB = 0x80 | leds;                                    //switch on
                            ↪  the PB7 and the state of the leds for 4 seconds
198                         for(int i = 0; i < 190; i++) {        //for 4 seconds
199                                 scan_keypad_rising_edge_sim(15, &r24,
                                    ↪  &r25);        //while scanning also (required in
                                    ↪  the remote access program)
200                                 _delay_ms(1);
201                         }
202                         PORTB = 0x00 | leds;                                    //after the
                            ↪  4 seconds switch off the PB7
203                         MSB_led = 0;
204                         password = 0;
205
206                 }
207                 else {                                                          //if the
                    ↪  digits were not 7 and 1
208                         for (int i = 0; i < 4; i++) {        //switch on and off the
                            ↪  leds 4 times
209                                 MSB_led = 1;
210                                 for(int i = 0; i < 35; i++) {        //for 0.5 sec each
211                                         scan_keypad_rising_edge_sim(15, &r24,
                                            ↪  &r25);        //while scanning (remote
                                            ↪  access program)
212                                         _delay_ms(1);
213                                 }
214                                 MSB_led = 0;
215                                 for(int i = 0; i < 35; i++) {
216                                         scan_keypad_rising_edge_sim(15, &r24, &r25);
217                                         _delay_ms(1);
218                                 }
219
220                         }
221                 }
222
223     }
224 }
```