



5η Εργαστηριακή Άσκηση Έλεγχος έντασης LEDs με χρήση PWM κυματομορφής

Μέλος 1: Λαγός Αναστάσιος - 03113531

Μέλος 2: Αντώνιος Δημήτριος Αλικάρης - 03118062

Γενική ιδέα

Για να αυξομειώσουμε την ένταση των LEDs, χρησιμοποιούμε τον χρονιστή/μετρητή TIMER/COUNTER0 για να παράξουμε μία PWM κυματομορφή συχνότητας 4kHz της οποίας το duty cycle μεταβάλλεται από το πληκτρολόγιο (με το πλήκτρο "1" αυξάνεται, ενώ με το πλήκτρο "2" μειώνεται). Λόγω απομακρυσμένης πρόσβασης στην πλακέτα, για εξακρίβωση επιτυχής λειτουργίας, συνδέουμε την κυματομορφή ως είσοδο στον μετατροπέα ADC και εμφανίζουμε την τάση στην οθόνη LCD.

Ο χρονιστής/μετρητής TIMER/COUNTER1 χρησιμοποιείται για την λειτουργία του ADC κάθε 100ms.

Το avr project περιλαμβάνει και 2 αρχεία .S που περιέχουν τις δύο απαραίτητες ρουτίνες για την λειτουργία της LCD (LCD_init και LCD_show) γραμμένες σε assembly.

Κύριος Κώδικας σε c

```
1  #undef F_CPU
2  #define F_CPU 8000000UL
3
4  #ifndef __DELAY_BACKWARD_COMPATIBLE__
5  #define __DELAY_BACKWARD_COMPATIBLE__
6  #endif
7
8  #include <avr/io.h>
9  #include <util/delay.h>
10 #include <avr/interrupt.h>
11 #include <stdlib.h>
12 #include <string.h>
13
14 #define NOP(){__asm__ __volatile__ ("nop");} //assembly nop
15
16 /*Keypad Functions*/
17 unsigned int scan_row_sim(unsigned int row);
18 void scan_keypad_sim();
19 unsigned int scan_keypad_rising_edge_sim(unsigned int flick_time);
20 unsigned char keypad_to_ascii_sim();
21
22 unsigned int swap(unsigned int val);
23
24 /*Delay functions*/
```

```

25 void wait_usec(unsigned int delay);
26 void wait_msec(unsigned int delay);
27
28 /*LCD Assembly*/
29 //External LCD routines in assembly. Source code shown below
30 extern void LCD_init();
31 extern void LCD_show(unsigned char cha);
32
33 /*Globals*/
34 unsigned int buttons[2], ram[2];
35 volatile unsigned int adc_output = 0;
36
37 /*function that delays for delay time*/
38 void wait_usec(unsigned int delay){
39     unsigned int i;
40     for(i = 0; i < (delay/10); i++) {    //10 usec delay for delay/10 times
41         _delay_us(10);
42     }
43     if (delay % 10) {    //delay for the remainder accordingly
44         _delay_us(delay % 10);
45     }
46 }
47 /*same function as wait_usec but for milliseconds*/
48 void wait_msec(unsigned int delay) {
49     unsigned int i ;
50     for(i = 0; i < (delay / 10); i++){
51         _delay_ms(10);
52     }
53     if(delay % 10) {
54         _delay_ms(delay % 10);
55     }
56 }
57 /*Function that does __asm__("swap")*/
58 unsigned int swap(unsigned int val) {
59     return ((val & 0x0F) << 4 | (val & 0xF0) >> 4);
60 }
61
62 /*Function that returns which columns are pressed for a given row*/
63 unsigned int scan_row_sim(unsigned int row) {
64     PORTC = row; //Search in line row.
65     wait_usec(500); //Delay required for a successful remote operation
66
67     NOP();
68     NOP(); //Delay to allow for a change of state
69
70     return PINC & 0x0F; //Return 4 LSB
71 }
72 /*Function that scans the whole keypad*/
73 void scan_keypad_sim() {
74     buttons[1] = swap(scan_row_sim(0x10)); // A 3 2 1
75     buttons[1] += scan_row_sim(0x20); // A 3 2 1 B 6 5 4
76
77     buttons[0] = swap(scan_row_sim(0x40)); // C 9 8 7
78     buttons[0] += (scan_row_sim(0x80)); // C 9 8 7 D # 0 *

```

```

79     PORTC = 0x00; // added only for the remote operation
80     return;
81 }
82
83 /*Function that checks which buttons where pressed since its last call*/
84 unsigned int scan_keypad_rising_edge_sim(unsigned int flick_time) {
85     scan_keypad_sim(); // do the first scan
86     unsigned int temp[2]; // store first scan
87     temp[0] = buttons[0];
88     temp[1] = buttons[1];
89     wait_msec(flick_time); //wait for flick time
90
91     scan_keypad_sim(); //scan second time
92     buttons[0] &= temp[0]; //remove flick values
93     buttons[1] &= temp[1];
94
95     temp[0] = ram[0]; //get the last state from previous call to rising_edge from
96     → "RAM"
97     temp[1] = ram[1];
98     ram[0] = buttons[0]; //update the new previous state
99     ram[1] = buttons[1];
100    buttons[0] &= ~temp[0]; //Keep values that change from 1 to 0
101    buttons[1] &= ~temp[1];
102
103    return (buttons[0] || buttons[1]);
104 }
105 /*Function that returns the ASCII code of button pressed*/
106 unsigned char keypad_to_ascii_sim() {
107     unsigned int select;
108     for(select = 0x01; select <= 0x80; select <= 1) {
109         switch(buttons[0] & select) {
110             case 0x01:
111                 return '*';
112             case 0x02:
113                 return '0';
114             case 0x04:
115                 return '#';
116             case 0x08:
117                 return 'D';
118             case 0x10:
119                 return '7';
120             case 0x20:
121                 return '8';
122             case 0x40:
123                 return '9';
124             case 0x80:
125                 return 'C';
126         }
127     }
128     for(select = 0x01; select <= 0x80; select <= 1) {
129         switch(buttons[1] & select) {
130             case 0x01:
131                 return '4';
132             case 0x02:

```

```

132         return '5';
133     case 0x04:
134         return '6';
135     case 0x08:
136         return 'B';
137     case 0x10:
138         return '1';
139     case 0x20:
140         return '2';
141     case 0x40:
142         return '3';
143     case 0x80:
144         return 'A';
145     }
146 }
147 return 0;
148 }
149
150 void PWM_init() {
151     //set TMRO in fast PWM mode with non-inverted output, prescale=8
152     TCCR0 = (1<<WGM00) | (1<<WGM01) | (1<<COM01) | (1<<CS01);
153     DDRB |= (1<<PB3);
154 }
155
156 void ADC_init(){
157     ADMUX = (1<<REFS0);
158     ADCSRA = (1<<ADEN)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
159 }
160
161 ISR(TIMER1_OVF_vect){
162     TCNT1 = 64755; //100ms
163     ADCSRA |= (1<<ADSC); //adc start conversion enable
164 }
165
166 ISR(ADC_vect){
167     adc_output = ADC;
168 }
169
170 //function which takes the ADC and shows in the lcd the converted voltage value
171 void convert_to_voltage_and_show(char adc_decimal_buffer[]) {
172     float analog_input = (float) 5 * adc_output / 1024;
173
174     char integer_part = (int) analog_input + '0';
175     unsigned int decimal_part = (int) (analog_input * 100) % 100;
176     itoa(decimal_part, adc_decimal_buffer, 10);
177
178     LCD_show(integer_part);
179     LCD_show('.');
180     LCD_show(adc_decimal_buffer[0]);
181     LCD_show(adc_decimal_buffer[1]);
182
183     return;
184 }
185

```

```

186
187 int main(void)
188 {
189     DDRD = 0xFF;           //Initialize LCD as output
190     DDRC = 0xF0;           //Initialize PORTC and LEDs. Internal resistor pull up must
                             → be deactivated
191     unsigned int duty = 0;
192     char adc_decimal[2];
193     unsigned int old_output = -1;
194     ADC_init();
195     LCD_init();
196     PWM_init();
197     TIMSK = (1<<TOIE1);
198     TCCR1B = (1<<CS12)|(0<<CS11)|(1<<CS10);
199     TCNT1 = 64755; //100ms
200     sei();
201     while(1) {
202         unsigned char button_pressed;
203         ram[0] = 0, ram[1] = 0;
204         while(1) {
205             if(scan_keypad_rising_edge_sim(15)) {
206                 button_pressed = keypad_to_ascii_sim();
207                 break;
208             }
209             if (old_output != adc_output) { //if the adc hasn't changed,
                             → dont show and compute the output again
210                 old_output = adc_output;
211                 LCD_init();
212                 LCD_show('V');
213                 LCD_show('o');
214                 LCD_show('1');
215                 LCD_show('\n');
216                 convert_to_voltage_and_show(adc_decimal);
217             }
218         }
219         if ((button_pressed == '1') && duty < 255) {
220             duty++; //increasing the upper pulse
221             OCR0 = duty;
222         }
223         else if((button_pressed == '2') && duty > 0){
224             duty--; //decreasing the upper pulse
225             OCR0 = duty;
226         }
227     }
228 }

```

Πρόσθετες συναρτήσεις σε assembly

LCD_init.S

```

1 #define _SFR_ASM_COMPAT 1
2 #define __SFR_OFFSET 0
3
4 #include <avr/io.h>

```

```

5
6 .global LCD_init
7
8 LCD_init:
9     rcall lcd_init_sim
10    ret
11
12    wait_msec:
13        push r24
14        push r25
15        ldi r24, lo8(1000)
16        ldi r25, hi8(1000)
17        rcall wait_usec
18        pop r25
19        pop r24
20        sbiw r24, 1
21        brne wait_msec
22
23        ret
24
25    wait_usec:
26        sbiw r24, 1 //2 cycles
27        nop
28        nop
29        nop
30        nop
31        brne wait_usec //1 cycle the majority of the time
32
33        ret
34
35    write_2_nibbles_sim:
36        push r24
37        push r25
38        ldi r24, lo8(6000)
39        ldi r25, hi8(6000)
40        rcall wait_usec
41        pop r25
42        pop r24
43        push r24
44        in r25, PIND
45        andi r25, 0x0f
46        andi r24, 0xf0
47        add r24, r25
48        out PORTD, r24
49        sbi PORTD, PD3
50        cbi PORTD, PD3
51        push r24
52        push r25
53        ldi r24, lo8(6000)
54        ldi r25, hi8(6000)
55        rcall wait_usec
56        pop r25
57        pop r24
58        pop r24

```

```

59     swap r24
60     andi r24 ,0xf0
61     add r24, r25
62     out PORTD, r24
63     sbi PORTD, PD3
64     cbi PORTD, PD3
65     ret
66
67     lcd_data_sim:
68     push r24
69     push r25
70     sbi PORTD,PD2
71     rcall write_2_nibbles_sim
72     ldi r24,43
73     ldi r25,0
74     rcall wait_usec
75     pop r25
76     pop r24
77     ret
78
79     lcd_command_sim:
80     push r24
81     push r25
82     cbi PORTD, PD2
83     rcall write_2_nibbles_sim
84     ldi r24, 39
85     ldi r25, 0
86     rcall wait_usec
87     pop r25
88     pop r24
89     ret
90
91     lcd_init_sim:
92     push r24
93     push r25
94     ldi r24, 40
95     ldi r25, 0
96     rcall wait_msec
97     ldi r24, 0x30
98     out PORTD, r24
99     sbi PORTD, PD3
100    cbi PORTD, PD3
101    ldi r24, 39
102    ldi r25, 0
103    rcall wait_usec
104    push r24
105    push r25
106    ldi r24,lo8(1000)
107    ldi r25,hi8(1000)
108    rcall wait_usec
109    pop r25
110    pop r24
111    ldi r24, 0x30
112    out PORTD, r24

```

```

113     sbi PORTD, PD3
114     cbi PORTD, PD3
115     ldi r24,39
116     ldi r25,0
117     rcall wait_usec
118     push r24
119     push r25
120     ldi r24 ,lo8(1000)
121     ldi r25 ,hi8(1000)
122     rcall wait_usec
123     pop r25
124     pop r24
125     ldi r24,0x20
126     out PORTD, r24
127     sbi PORTD, PD3
128     cbi PORTD, PD3
129     ldi r24,39
130     ldi r25,0
131     rcall wait_usec
132     push r24
133     push r25
134     ldi r24 ,lo8(1000)
135     ldi r25 ,hi8(1000)
136     rcall wait_usec
137     pop r25
138     pop r24
139     ldi r24,0x28
140     rcall lcd_command_sim
141     ldi r24,0x0c
142     rcall lcd_command_sim
143     ldi r24,0x01
144     rcall lcd_command_sim
145     ldi r24, lo8(1530)
146     ldi r25, hi8(1530)
147     rcall wait_usec
148     ldi r24 ,0x06
149     rcall lcd_command_sim
150     pop r25
151     pop r24
152     ret

```

LCD_show.S

```

1  #define _SFR_ASM_COMPAT 1
2  #define __SFR_OFFSET 0
3
4  #include <avr/io.h>
5
6  .global LCD_show
7
8  LCD_show:
9      rcall lcd_data_sim
10     ret
11

```



```

12      wait_msec:
13          push r24
14          push r25
15          ldi r24, lo8(1000)
16          ldi r25, hi8(1000)
17          rcall wait_usec
18          pop r25
19          pop r24
20          sbiw r24, 1
21          brne wait_msec
22
23          ret
24
25      wait_usec:
26          sbiw r24, 1 //2 cycles
27          nop
28          nop
29          nop
30          nop
31          brne wait_usec //1 cycle the majority of the time
32
33          ret
34
35      write_2_nibbles_sim:
36      push r24
37      push r25
38      ldi r24 ,lo8(6000)
39      ldi r25 ,hi8(6000)
40      rcall wait_usec
41      pop r25
42      pop r24
43      push r24
44      in r25, PIND
45      andi r25, 0x0f
46      andi r24, 0xf0
47      add r24, r25
48      out PORTD, r24
49      sbi PORTD, PD3
50      cbi PORTD, PD3
51      push r24
52      push r25
53      ldi r24 ,lo8(6000)
54      ldi r25 ,hi8(6000)
55      rcall wait_usec
56      pop r25
57      pop r24
58      pop r24
59      swap r24
60      andi r24 ,0xf0
61      add r24, r25
62      out PORTD, r24
63      sbi PORTD, PD3
64      cbi PORTD, PD3
65      ret

```

```

66
67     lcd_data_sim:
68     push r24
69     push r25
70     sbi PORTD,PD2
71     rcall write_2_nibbles_sim
72     ldi r24,43
73     ldi r25,0
74     rcall wait_usec
75     pop r25
76     pop r24
77     ret
78
79     lcd_command_sim:
80     push r24
81     push r25
82     cbi PORTD, PD2
83     rcall write_2_nibbles_sim
84     ldi r24, 39
85     ldi r25, 0
86     rcall wait_usec
87     pop r25
88     pop r24
89     ret
90
91     lcd_init_sim:
92     push r24
93     push r25
94     ldi r24, 40
95     ldi r25, 0
96     rcall wait_msec
97     ldi r24, 0x30
98     out PORTD, r24
99     sbi PORTD, PD3
100    cbi PORTD, PD3
101    ldi r24, 39
102    ldi r25, 0
103    rcall wait_usec
104    push r24
105    push r25
106    ldi r24,lo8(1000)
107    ldi r25,hi8(1000)
108    rcall wait_usec
109    pop r25
110    pop r24
111    ldi r24, 0x30
112    out PORTD, r24
113    sbi PORTD, PD3
114    cbi PORTD, PD3
115    ldi r24,39
116    ldi r25,0
117    rcall wait_usec
118    push r24
119    push r25

```

```

120     ldi r24 ,lo8(1000)
121     ldi r25 ,hi8(1000)
122     rcall wait_usec
123     pop r25
124     pop r24
125     ldi r24,0x20
126     out PORTD, r24
127     sbi PORTD, PD3
128     cbi PORTD, PD3
129     ldi r24,39
130     ldi r25,0
131     rcall wait_usec
132     push r24
133     push r25
134     ldi r24 ,lo8(1000)
135     ldi r25 ,hi8(1000)
136     rcall wait_usec
137     pop r25
138     pop r24
139     ldi r24,0x28
140     rcall lcd_command_sim
141     ldi r24,0x0c
142     rcall lcd_command_sim
143     ldi r24,0x01
144     rcall lcd_command_sim
145     ldi r24, lo8(1530)
146     ldi r25, hi8(1530)
147     rcall wait_usec
148     ldi r24 ,0x06
149     rcall lcd_command_sim
150     pop r25
151     pop r24
152     ret

```