# 4η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"

Δρυς-Πεντζάκης Οδυσσεύς el19192
Μαρκαντωνάτος Γεράσιμος el19149

## 4.1

Θέλουμε να απεικονίσουμε το Vin οθόνη lcd. Αρχικοποιούμε την οθόνη lcd, πολλαπλασιάζουμε με 5 το ADC και παίρνοντας τα 3 msb του πολλαπλασιασμένου ADC παίρνουμε το ακέραιο μέρος του Vin, κάνουμε ori με το 00110000 ώστε να απεικονίσουμε σωστά τον αριθμό στην οθόνη και απεικονίζουμε τον αριθμό Για τα δύο δεκαδικά πολλαπλασιάζουμε το ADC κάθε φορά με το 10 παίρνουμε τα 3 msb και επαναλαμβάνουμε την διαδικασία.

assembly:

```
.include "m328PBdef.inc"
.equ PD3 = 3
.equ PD2 = 2
.def temp = r16
.org 0x00
 jmp reset

.org 0x2A
 jmp conversion
reset:
    ldi temp, high(RAMEND)
    out SPH,temp
    ldi temp, low(RAMEND)
    out SPL,temp

    sei



    ser r24
    out DDRD, r24
    rcall lcd_init
    ldi r24, low(2)
    ldi r25, high(2)
```

```
    rcall wait_msec

    ldi temp, 0xFF
    out DDRB, temp

    ldi temp, 0x00
    out DDRC, temp

    ldi temp, 0b01000010
    sts ADMUX, temp

    ldi temp, 0b10101111
    sts ADCSRA, temp

    ldi temp, 0b00000000
    sts ADCSRB, temp

    ldi r20,0x00
    out PORTB, r20

    ldi r24, HIGH(6070)
    sts TCNT1H, r24
    ldi r24, LOW(6070)
    sts TCNT1L, r24



start:
    lds temp, ADCSRA
    ori temp, (1<<ADSC)
    sts ADCSRA, temp

count:
    ldi r24, low(1000)
    ldi r25, high(1000)
    rcall wait_msec
    inc r20
    out PORTB, r20
    jmp count


conversion:
rcall lcd_init
ldi r24, low(2)
ldi r25, high(2)
rcall wait_msec
lds r17,ADCL
lds r18,ADCH
```

```
mov r22,r18
mov r21,r17
lsl r17
rol r18
lsl r17
rol r18
add r17,r21
adc r18,r22
mov r22,r18
andi r18,0b00011100
sub r22,r18
lsr r18
lsr r18
ori r18,0b00110000
mov r24,r18
rcall lcd_data


ldi r24, '.'
rcall lcd_data


mov r21,r17
mov r18,r22
lsl r17
rol r18
lsl r17
rol r18
lsl r17
rol r18
add r17,r21
adc r18,r22
add r17,r21
adc r18,r22
mov r22,r18
andi r18,0b00111100
sub r22,r18
lsr r18
lsr r18
ori r18,0b00110000
mov r24, r18
rcall lcd_data




mov r18,r22
```

```
mov r21,r17
lsl r17
rol r18
lsl r17
rol r18
lsl r17
rol r18
add r17,r21
adc r18,r22
add r17,r21
adc r18,r22
andi r18,0b00111100
lsr r18
lsr r18
ori r18,0b00110000
mov r24, r18
rcall lcd_data
ldi r24, low(500)
ldi r25, high(500)
rcall wait_msec
reti



lcd_init:
ldi r24 ,40
ldi r25 ,0
rcall wait_msec
ldi r24 ,0x30
out PORTD ,r24
 sbi PORTD ,PD3
cbi PORTD ,PD3
ldi r24 ,100
ldi r25 ,0
rcall wait_usec
 ldi r24 ,0x30
out PORTD ,r24
 sbi PORTD ,PD3
cbi PORTD ,PD3
ldi r24 ,100
ldi r25 ,0
rcall wait_usec
ldi r24 ,0x20
out PORTD ,r24
 sbi PORTD ,PD3
cbi PORTD ,PD3
ldi r24 ,100
ldi r25 ,0
```

```
rcall wait_usec
 ldi r24 ,0x28
 rcall lcd_command
ldi r24 ,0x0c
 rcall lcd_command
 ldi r24 ,0x01
rcall lcd_command
ldi r24 ,low(5000)
ldi r25 ,high(5000)
rcall wait_usec
 ldi r24 ,0x06
rcall lcd_command

ret

lcd_command:
cbi PORTD ,PD2 rcall write_2_nibbles
 ldi r24 ,100
ldi r25 ,0
rcall wait_usec
ret

lcd_data:
 sbi PORTD ,PD2
rcall write_2_nibbles
ldi r24 ,100
ldi r25 ,0
rcall wait_usec
ret

write_2_nibbles:
push r24
in r25 ,PIND
andi r25 ,0x0f
andi r24 ,0xf0
add r24 ,r25
out PORTD ,r24
 sbi PORTD ,PD3
cbi PORTD ,PD3
pop r24
 swap r24
andi r24 ,0xf0
add r24 ,r25
out PORTD ,r24
 sbi PORTD ,PD3
cbi PORTD ,PD3
ret
```

```
wait_msec:              ; 1 msec delay per call
push r24                ; 2 cycles
push r25                ; 2 cycles
ldi r24,low(125)        ; 1 cycle
ldi r25,high(125)       ; 1 cycle
rcall wait_usec         ; 3 cycles
pop r25                 ; 2 cycles
pop r24                 ; 2 cycles
sbiw r24,1              ; 2 cycles
brne wait_msec          ; 1 or 2 cycles
ret                     ; 4 cycles

wait_usec:              ; Called 125 times
sbiw r24,1              ; 2 cycles (2 usec)
nop                     ; 1 cycle (1 usec)
nop                     ; 1 cycle (1 usec)
nop                     ; 1 cycle (1 usec)
nop                     ; 1 cycle (1 usec)
brne wait_usec          ; 1 or 2 cycles (1 or 2 usec)
ret                     ; 4 cycles (4 usec)
```

# 4.1c

Για το πρόγραμμα σε C ο υπολογισμός του Vin γίνεται πιο εύκολα, υπολογίζοντας το Vin από τον δοσμένο τύπο και χρησιμοποιώντας το uint8_t για την απομόνωση του ακέραιου μέρους.

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <stdio.h>
#include <util/delay.h>
#include <stdlib.h>

#define LCD_Dir  DDRB                   /* Define LCD data port direction */
#define LCD_Port PORTB                  /* Define LCD data port */
#define RS PB0                             /* Define Register Select pin */
#define EN PB1


void Write_2_Nibbles(uint8_t in) {
   uint8_t temp = in;
   uint8_t p = PIND;
   p &= 0x0F;
   in &= 0xF0;
   in |= p;
   PORTD = in;
```

```
    PORTD |= 0x08;
    PORTD &= 0xF7;

    in = temp;
    in &= 0x0F;
    in = in << 4;
    in |= p;
    PORTD = in;
    PORTD |= 0x08;
    PORTD &= 0xF7;

    return;
}

void LCD_data(uint8_t c) {
    PORTD |= 0x04;
    Write_2_Nibbles(c);
    _delay_us(100);
    return;
}

void LCD_command(uint8_t c) {
    PORTD &= 0xFB;
    Write_2_Nibbles(c);
    _delay_us(100);
    return;
}

void LCD_init(void) {
    _delay_ms(40);

    PORTD = 0x30;
    PORTD |= 0x08;
    PORTD &= 0xF7;
    _delay_us(100);

    PORTD = 0x30;
    PORTD |= 0x08;
    PORTD &= 0xF7;
    _delay_us(100);

    PORTD = 0x20;
    PORTD |= 0x08;
    PORTD &= 0xF7;
    _delay_us(100);

    LCD_command(0x28);
    LCD_command(0x0C);
```

```
    LCD_command(0x01);
    _delay_us(5000);

    LCD_command(0x06);
}

static volatile float adc;
static volatile uint8_t adc1;
static volatile uint8_t adc2;
static volatile uint8_t adc3;


int main()
{
    DDRD = 0xFF;
    DDRC = 0x00;
    ADMUX = (1 << REFS0) | (1 << MUX1);
    ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
    ADCSRB = 0x00;
    DIDR0 = ~(1 << ADC2D);
    float decimal;
    while (1) {
        ADCSRA |= (1 << ADSC);
        while ((ADCSRA & (1 << ADSC)) == (1 << ADSC));

        adc = ADC;
        adc = (adc * 5) / 1024;

        adc1 = (uint8_t)adc;
        decimal = adc - adc1;
        adc2 = (uint8_t)(decimal * 10);
        adc3 = (uint8_t)(((decimal * 10) - adc2) * 10);


        adc1 |= 0x30;
        adc2 |= 0x30;
        adc3 |= 0x30;

        LCD_init();
        _delay_ms(2);

        LCD_data(adc1);
        LCD_data('.');
        LCD_data(adc2);
        LCD_data(adc3);

        _delay_ms(100);
    }
```

}

# 4.3

Θέλουμε το πρόγραμμα να παράγει μία PWM κυματομορφή στον ακροδέκτη PB1 με συχνότητα 5KHz όταν είναι πατημένο κάποιο από τα πλήκτρα PB2 – PB5, ενώ όταν δεν είναι πατημένο κάποιο από αυτά τα πλήκτρα τότε δεν παράγεται κυματομορφή. Φορτώνουμε την τιμή 399 στον καταχωρητή ICR1 ώστε να έχουμε συχνότητα 5KHz και αναλόγως το κουμπί που έχουμε πατήσει αλλάζουμε το duty cycle βάζοντας στην τιμή του καταχωρητή OCR1A το κατάλληλο πολλαπλάσιο του ICR1.

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

static volatile float adc;
static volatile uint8_t adc1;
static volatile uint8_t adc2;
static volatile uint8_t adc3;

void Write_2_Nibbles(uint8_t in) {
    uint8_t temp = in;
    uint8_t p = PIND;
    p &= 0x0F;
    in &= 0xF0;
    in |= p;
    PORTD = in;
    PORTD |= 0x08;
    PORTD &= 0xF7;

    in = temp;
    in &= 0x0F;
    in = in << 4;
    in |= p;
    PORTD = in;
    PORTD |= 0x08;
    PORTD &= 0xF7;

    return;
}

void LCD_data(uint8_t c) {
    PORTD |= 0x04;
    Write_2_Nibbles(c);
    _delay_us(100);
    return;
}
```

```
void LCD_command(uint8_t c) {
    PORTD &= 0xFB;
    Write_2_Nibbles(c);
    _delay_us(100);
    return;
}

void LCD_init(void) {
    _delay_ms(40);

    PORTD = 0x30;
    PORTD |= 0x08;
    PORTD &= 0xF7;
    _delay_us(100);

    PORTD = 0x30;
    PORTD |= 0x08;
    PORTD &= 0xF7;
    _delay_us(100);

    PORTD = 0x20;
    PORTD |= 0x08;
    PORTD &= 0xF7;
    _delay_us(100);

    LCD_command(0x28);
    LCD_command(0x0C);
    LCD_command(0x01);
    _delay_us(5000);

    LCD_command(0x06);
}



int main(void) {
    float decimal;
    TCCR1A = (0<<WGM10) | (1<<WGM11) |(1<<COM1A1) ;
    TCCR1B = (1<<WGM12) | (1<<CS11)|(1<<WGM13);
    DDRB=0b00111111;
    OCR1AL = 0x80;
    DDRD = 0xFF;
    DDRC = 0x00;
    ADMUX = (1 << REFS0) | (1 << MUX1);
    ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
    ADCSRB = 0x00;
    DIDR0 = ~(1 << ADC2D);
```

```
while(1) {
   ICR1=0x0000;
   if((PINB&0b00000100)==0){
      while((PINB&0b00000100)==0);
         ICR1=399;
         OCR1A =ICR1*0.2;
         ADCSRA |= (1 << ADSC);
         while ((ADCSRA & (1 << ADSC)) == (1 << ADSC));

         adc = ADC;
         adc = (adc * 5) / 1024;

         adc1 = (uint8_t)adc;
         decimal = adc - adc1;
         adc2 = (uint8_t)(decimal * 10);
         adc3 = (uint8_t)(((decimal * 10) - adc2) * 10);


         adc1 |= 0x30;
         adc2 |= 0x30;
         adc3 |= 0x30;

         LCD_init();
         _delay_ms(2);

         LCD_data('2');
         LCD_data('0');
         LCD_data('%');
         LCD_command(0b11000000);
         LCD_data(adc1);
         LCD_data('.');
         LCD_data(adc2);
         LCD_data(adc3);




   _delay_ms(100);

      }
   else if ((PINB&0b00001000)==0){
      while((PINB&0b00001000)==0);
         ICR1=399;
         OCR1A =ICR1*0.4;
         ADCSRA |= (1 << ADSC);
         while ((ADCSRA & (1 << ADSC)) == (1 << ADSC));

         adc = ADC;
```

```
        adc = (adc * 5) / 1024;

        adc1 = (uint8_t)adc;
        decimal = adc - adc1;
        adc2 = (uint8_t)(decimal * 10);
        adc3 = (uint8_t)(((decimal * 10) - adc2) * 10);


        adc1 |= 0x30;
        adc2 |= 0x30;
        adc3 |= 0x30;

        LCD_init();
        _delay_ms(2);

        LCD_data('4');
        LCD_data('0');
        LCD_data('%');
        LCD_command(0b11000000);
        LCD_data(adc1);
        LCD_data('.');
        LCD_data(adc2);
        LCD_data(adc3);
    }
else if ((PINB&0b00010000)==0){
    while((PINB&0b0010000)==0);
        ICR1=399;
        OCR1A =ICR1*0.6;
        ADCSRA |= (1 << ADSC);
        while ((ADCSRA & (1 << ADSC)) == (1 << ADSC));

        adc = ADC;
        adc = (adc * 5) / 1024;

        adc1 = (uint8_t)adc;
        decimal = adc - adc1;
        adc2 = (uint8_t)(decimal * 10);
        adc3 = (uint8_t)(((decimal * 10) - adc2) * 10);


        adc1 |= 0x30;
        adc2 |= 0x30;
        adc3 |= 0x30;

        LCD_init();
        _delay_ms(2);

        LCD_data('6');
```

```
            LCD_data('0');
            LCD_data('%');
            LCD_command(0b11000000);
            LCD_data(adc1);
            LCD_data('.');
            LCD_data(adc2);
            LCD_data(adc3);
        }
    else if ((PINB&0b00100000)==0){
        while((PINB&0b00100000)==0);
            ICR1=399;
            OCR1A =ICR1*0.8;
            ADCSRA |= (1 << ADSC);
            while ((ADCSRA & (1 << ADSC)) == (1 << ADSC));

            adc = ADC;
            adc = (adc * 5) / 1024;

            adc1 = (uint8_t)adc;
            decimal = adc - adc1;
            adc2 = (uint8_t)(decimal * 10);
            adc3 = (uint8_t)(((decimal * 10) - adc2) * 10);


            adc1 |= 0x30;
            adc2 |= 0x30;
            adc3 |= 0x30;

            LCD_init();
            _delay_ms(2);

            LCD_data('8');
            LCD_data('0');
            LCD_data('%');
            LCD_command(0b11000000);
            LCD_data(adc1);
            LCD_data('.');
            LCD_data(adc2);
            LCD_data(adc3);
        }
    }

}
```