Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών
Εργαστήριο Μικροϋπολογιστών

Έκθεση 4ης Εργαστηριακής Άσκησης

Στοιχεία:
Ομάδα: **39**
Μέλη: **Ευάγγελος Μυργιώτης, Αριστοτέλης Γρίβας**

Ζήτημα 4.1 :

Γνωρίζουμε ότι το ADC παίρνει τιμές από 0-1023 όπου το 1023 δηλώνει (1023/1024)*5 = 4,995 Volt, άρα το 1 Volt αντιστοιχεί στην τιμή 204 στο ADC. Αντίστοιχα το 20 στα 0,1 Volt και το 2 στα 0.01 Volt.

Άρα αρκεί να βρούμε:

- πόσες X φορές χωράει το 204 στο περιεχόμενο του ADC,
- πόσες Y φορές χωράει το 20 στο περιεχόμενο του ADC και
- πόσες Z φορές χωράει το 2 στο περιεχόμενο του ADC

Το τελικό αποτέλεσμα είναι το X,YZ όπου πειραματικά δείξαμε ότι είναι πάντα εντός 0.01 σε ακρίβεια.

Βέβαια όμως το περιεχόμενο του ADC αποθηκεύεται σε δυο καταχωρητές, οπότε δεν μπορούμε κατευθείαν να συγκρίνουμε με το 204. Για αυτό τον λόγο κρατάμε τα 8 κορυφαία bit και τα συγκρίνουμε με το 51 (¼ του 204) καθώς είναι σαν να διαιρέσαμε με το 4. Ύστερα έχουν μείνει αρκετά bit για να χωρέσουν σε ένα καταχωρητή και μετά τα κατάλληλα shift μπορούμε να κάνουμε τις συγκρίσεις με το 204, το 20 και το 2.

Κώδικας σε Assembly :

```
.include "m328PBdef.inc" ;ATmega328P microcontroller definitions

.def temp = r16
.def ADC_L = r21
.def ADC_H = r22
;.equ PD3 = 3
;.equ PD2 = 2

.org 0x00
rjmp main
.org 0x2A ;ADC Conversion Complete Interrupt
rjmp ADC_ROUTINE
reti
```

```
main:
ldi r24, low(RAMEND)
out SPL, r24
ldi r24, high(RAMEND)
out SPH, r24 ; stack pointer


ser r24
out DDRD, r24 ;  PORTD,PORTC output
out DDRB, r24 ;
clr r24
out DDRC, r24


; REFSn[1:0]=01 => select Vref=5V, MUXn[4:0]=0000 => select ADC0(pin PC0),
 ; ADLAR=1 => Left adjust the ADC result
 ldi temp, 0b01100000 ;
 sts ADMUX, temp
 ; ADEN=1 => ADC Enable, ADCS=0 => No Conversion,
 ; ADIE=0 => disable adc interrupt, ADPS[2:0]=111 => fADC=16MHz/128=125KHz
 ldi temp, 0b10001111
 sts ADCSRA, temp
sei



main1:

rcall lcd_init ; initialise LCD
ldi r24, low(2)
ldi r25, high(2) ;  2 msec delay
rcall wait_msec




lds temp, ADCSRA ;
;ori temp,( 1 << ADIE);
ori temp, (1<<ADSC) ; Set ADSC flag of ADCSRA

sts ADCSRA, temp



subi r20, -1
out  PORTB, r20
cpi r20,63
brlo continue
clr r20
continue:
ldi r24 ,low(10000)
ldi r25 ,high(10000)
rcall wait_msec
```

```
rjmp main1




lcd_init:
ldi r24 ,40
ldi r25 ,0
rcall wait_msec ;  40 msec
ldi r24 ,0x30 ;  8 bit mode
out PORTD ,r24 ;
sbi PORTD ,3 ;
    nop
    nop
cbi PORTD ,3 ;
    nop
    nop
ldi r24 ,1
ldi r25 ,0 ;  8-bit mode
rcall wait_msec

 ldi r24 ,0x30
out PORTD ,r24
 sbi PORTD ,3
 nop
 nop
cbi PORTD ,3
 nop
 nop
ldi r24 ,1
ldi r25 ,0
rcall wait_msec
ldi r24 ,0x20 ;  4-bit mode
out PORTD ,r24
 sbi PORTD ,3
 nop
 nop
cbi PORTD ,3
 nop
 nop
ldi r24 ,1
ldi r25 ,0
rcall wait_msec
 ldi r24 ,0x28
 rcall lcd_command
ldi r24 ,0x0c
 rcall lcd_command
 ldi r24 ,0x01
```

```
rcall lcd_command
ldi r24 ,low(10)
ldi r25 ,high(10)
rcall wait_msec
 ldi r24 ,0x06
rcall lcd_command

ret
```

```
write_2_nibbles:
push r24 ; 4 MSB
in r25 ,PIND ;  4 LSB
andi r25 ,0x0f
andi r24 ,0xf0
add r24 ,r25
out PORTD ,r24
 sbi PORTD ,3
cbi PORTD ,3 ; PD3=1,PD3=0
pop r24 .
 swap r24 ; 4 MSB swaped with  4 LSB
andi r24 ,0xf0
add r24 ,r25
out PORTD ,r24
 sbi PORTD ,3
 nop
 nop
cbi PORTD ,3
 nop
 nop
ret
```

```
lcd_data:
sbi PORTD ,2 ; (PD2=1)
rcall write_2_nibbles ;
ldi r24 ,10 ;
ldi r25 ,0 ;  lcd
```

```
rcall wait_msec
ret




lcd_command:
cbi PORTD ,2 ; (PD2=1)
rcall write_2_nibbles ;
ldi r24 ,10 ;
ldi r25 ,0 ;
rcall wait_msec ;
ret




ADC_ROUTINE:

push r24
push r25


lds ADC_L,ADCL ; Read ADC result(Left adjusted)
lds ADC_H,ADCH ;
mov r24, ADC_H
mov r25, ADC_L

clr r17
clr r18
clr r19


first_bcd:
    cpi r24, 51
    brsh first_single


shifts:
    rol r25
    rol r24
    rol r25
    rol r24



bcd:
    cpi r24, 204
    brsh single
bcd2:
```

```
    cpi r24, 20
    brsh first
bcd3:
    cpi r24, 2
    brsh second
    rjmp done

second:
    subi r19, -1
    subi r24, 2
    rjmp bcd3


first:
    subi r18, -1
    subi r24, 20
    rjmp bcd2

single:
    subi r17, -1
    subi r24, 204
    rjmp bcd

first_single:
    subi r17, -1
    subi r24, 51
    cpi r24, 51
    brlo shifts
    rjmp first_bcd


done:
    ori r17, 0b00110000
    mov r24, r17
    rcall lcd_data
    ldi r24, ','
    rcall lcd_data
    ori r18, 0b00110000
    mov r24, r18
    rcall lcd_data
    ori r19, 0b00110000
    mov r24, r19
    rcall lcd_data


    ldi r24, low(2000)
    ldi r25, high(2000) ; delay2 sec
    rcall wait_msec



    pop r25
    pop r24
reti
```

```
wait_msec:                  ;delay routine used in previous exercises too

ldi r23, 249

loop_inn:

dec r23
nop
brne loop_inn
sbiw r24, 1
brne wait_msec
ret
```

Κώδικας σε C :

```
#define F_CPU 16000000UL
#include "avr/io.h"
#include "util/delay.h"


unsigned char duty;
unsigned char i,a,p,z,q,d;
unsigned char x,y;
unsigned char temp,a1,a2,a3;
```

```
void write_2_nibbles(unsigned char q){
    int temp = q;   // r24 = temp = q
    int e = PIND;   //in r25, PIND
    e = e & 0x0f;    // andi r25, 0x0f
    temp = temp & 0xf0; // andi r24, 0xf0
    temp = temp | e;    // add r24, r25
    PORTD= temp;  //out PORTD ,r24
    PORTD |= 0x08;  //sbi PORTD ,3
    _delay_ms(1);
    PORTD &= 0b11110111; //cbi PORTD ,3
    _delay_ms(1);

    temp = q;   //pop r24
    temp = temp<<4;  //swap r24
    temp = temp & 0xf0;  //andi r24, 0xf0
```

```
    temp = temp | e;   //add r24,r25
    PORTD= temp;   //portd, r24
    PORTD |= 0x08;
    _delay_ms(1);
    PORTD &= 0b11110111;
    _delay_ms(2);

return;
}


void lcd_data(unsigned char p){
    PORTD |= 0x04;
    write_2_nibbles(p);
    _delay_ms(10);
    return;
}

void lcd_command(unsigned char z){
    PORTD &= 0b11111011;
    write_2_nibbles(z);
    _delay_ms(10);
    return;
}


void lcd_init(){
        _delay_ms(40); //wait_msec

        PORTD = 0x30;     //PORTD = r24 = 0x30
        PORTD |= 0x08;   // sbi PORTD, 3
        _delay_ms(1);
        PORTD &= 0b11110111; //cbi PORTD, 3
        _delay_ms(2);

        PORTD = 0x30;   //PORTD = r24 = 0x30
        PORTD |= 0x08;
        _delay_ms(1);
        PORTD &= 0b11110111;
        _delay_ms(2);
        PORTD = 0x20;
        PORTD |= 0x08;
        _delay_ms(1);
        PORTD &= 0b11110111;
        _delay_ms(2);
        lcd_command(0x28);
        lcd_command(0x0c);
        lcd_command(0x01);

        _delay_ms(10);

        lcd_command(0x06);

        _delay_ms(10);
```

```
        return;
}

void metatroph(){
    z = ADCL;
    y = ADCH;
    z = z & 0b11000000;
    a1 = a2 = a3 = 0;

    while(1){
        if(y >= 51){
            a1 += 1;
            y -= 51;
        }
        else break;
    }

    z = z >> 6;
    y = y << 2;
    y = y | z;

    while(1){
        if(y >= 204){
            a1 += 1;
            y -= 204;
        }
        else break;
    }

    while(1){
        if(y >= 20){
            a2 += 1;
            y -= 20;
        }
        else break;
    }

    while(1){
        if(y >= 2){
            a3 += 1;
            y -= 2;
        }
        else break;
    }

    a1 = a1 | 0b00110000;
    a2 = a2 | 0b00110000;
    a3 = a3 | 0b00110000;
    lcd_data(a1);
    lcd_data(',');
    lcd_data(a2);
    lcd_data(a3);
    _delay_ms(100);
```

```
    return;
}

int main(){
 DDRD |=  0b11111111;
DDRB |=  0b11111111;
DDRC |=  0b00000000;

ADMUX = 0b01100000;

ADCSRA = 0b10000111;




while(1){
    lcd_init();
    _delay_ms(2);



    ADCSRA |= (1 << ADSC);
    // _delay_ms(10);
    while(ADCSRA & 0b01000000){

    }
     metatroph();

     _delay_ms(10);


    }
}
```

Ζήτημα 4.2 :

Μετά από υπολογισμούς γνωρίζουμε ότι το CO έχει ξεπεράσει τα 70ppm όταν μετρήσουμε 1 volt από το ADC. Από την προηγούμενη άσκηση έχουμε το X οπύ αρκεί να το συγκρίνουμε με το 1 για να δούμε αν πρέπει να εμφανίσουμε το σήμα κίνδυνου. Κάνουμε επίσης τις κατάλληλες μετατροπές για να ορίσουμε τα επίπεδα (τα επίπεδα είναι μη γραμμικά) και ανάβουμε το αντίστοιχο λαμπάκι. Όταν είναι πάνω από 70ppm το λαμπάκι αναβοσβήνει. Για να το καταφέρουμε αυτό έχουμε ένα register που αλλάζει μεταξύ του 1 και του 0 όταν είμαστε σε επικίνδυνα επίπεδα. Όταν είναι 1 το λαμπάκι είναι ανοιχτό ενώ όταν είναι μηδέν σβήνει.

Κώδικας σε Assembly :

```
.include "m328PBdef.inc" ;ATmega328P microcontroller definitions

.def temp = r16
.def ADC_L = r21
.def ADC_H = r22


.org 0x00
rjmp main
.org 0x2A ;ADC Conversion Complete Interrupt
rjmp ADC_ROUTINE




main:
ldi r24, low(RAMEND)
out SPL, r24
ldi r24, high(RAMEND)
out SPH, r24 ;


ser r24
out DDRD, r24 ;
out DDRB, r24 ;
clr r24
out DDRC, r24


; REFSn[1:0]=01 => select Vref=5V, MUXn[4:0]=0011 => select ADC3(pin PC3),
 ; ADLAR=1 => Left adjust the ADC result
 ldi temp, 0b01100011 ;
 sts ADMUX, temp
 ; ADEN=1 => ADC Enable, ADCS=0 => No Conversion,
 ; ADIE=0 => disable adc interrupt, ADPS[2:0]=111 => fADC=16MHz/128=125KHz
 ldi temp, 0b10001111
 sts ADCSRA, temp
```

```
sei


main1:
rcall lcd_init ;
ldi r24, low(2)
ldi r25, high(2) ;  2 msec
rcall wait_msec


lds temp, ADCSRA ;
ori temp, (1<<ADSC) ; Set ADSC flag of ADCSRA
sts ADCSRA, temp

;wait_adc:
; lds temp, ADCSRA ;
; sbrc temp,ADSC ; Wait until ADSC flag of ADCSRA becomes 0
; rjmp wait_adc

ldi r24 ,low(100)
ldi r25 ,high(100)
rcall wait_msec

;cpi r19, 0
;breq  do_nothing

rjmp main1







lcd_init:
ldi r24 ,40 ;
ldi r25 ,0 ;
rcall wait_msec ;
ldi r24 ,0x30 ;
out PORTD ,r24 ;
 sbi PORTD ,3 ;
cbi PORTD ,3 ;
ldi r24 ,1
ldi r25 ,0
rcall wait_msec

 ldi r24 ,0x30
out PORTD ,r24
 sbi PORTD ,3
cbi PORTD ,3
ldi r24 ,1
ldi r25 ,0
rcall wait_msec
```

```
ldi r24 ,0x20 ;  4-bit mode
out PORTD ,r24
 sbi PORTD ,3
cbi PORTD ,3
ldi r24 ,1
ldi r25 ,0
rcall wait_msec
 ldi r24 ,0x28
 rcall lcd_command
ldi r24 ,0x0c
 rcall lcd_command
 ldi r24 ,0x01
rcall lcd_command
ldi r24 ,low(10)
ldi r25 ,high(10)
rcall wait_msec
 ldi r24 ,0x06
rcall lcd_command

ret









write_2_nibbles:
push r24 ;  4 MSB
in r25 ,PIND ; 4 LSB
andi r25 ,0x0f
andi r24 ,0xf0 ; 4 MSB
add r24 ,r25 ;  4 LSB
out PORTD ,r24 ;
 sbi PORTD ,3 ; Enable PD3
cbi PORTD ,3 ; PD3=1  PD3=0
pop r24 ;
 swap r24 ; 4 MSB 4 LSB
andi r24 ,0xf0 ;
add r24 ,r25
out PORTD ,r24
 sbi PORTD ,3 ;  Enable
 nop
 nop
cbi PORTD ,3
 nop
 nop
```

```
ret



lcd_data:
sbi PORTD ,2 ;  (PD2=1)
rcall write_2_nibbles ;  byte
ldi r24 ,10
ldi r25 ,0 ; lcd
rcall wait_msec
ret



lcd_command:
cbi PORTD ,2 ;  (PD2=1)
rcall write_2_nibbles ;
 ldi r24 ,10 ;  lcd.
ldi r25 ,0 ;  clear display  return home,
rcall wait_msec ;
ret



ADC_ROUTINE:
push r24
push r25


clr r19
lds ADC_L,ADCL ; Read ADC result(Left adjusted)
lds ADC_H,ADCH ;
;lsr ADC_H
mov r24, ADC_H
mov r25, ADC_L


;R24 HAS Cx
ldi r29, 0b00100000
cpi r24, 220
brsh display
ldi r29, 0b00010000
cpi r24, 160
brsh display
ldi r29, 0b00001000
cpi r24, 80
brsh display
```

```
ldi r29, 0b00000100
cpi r24, 40
brsh display
ldi r29, 0b00000010
cpi r24, 20
brsh display
ldi r29, 0b00000001



display:
    out PORTB, r29
    cpi r24, 51
    brsh GAS_DETECTED
    rol r25
    rol r24
    rol r25
    rol r24
    cpi r24, 204
    brsh GAS_DETECTED


CLEAR:
    ldi r19, 0
   ; cpi r30, 0x00
   ; breq done
    ldi r31, 0x00
    ldi r24, 'C'
    rcall lcd_data
    ldi r24, 'L'
    rcall lcd_data
    ldi r24, 'E'
    rcall lcd_data
    ldi r24, 'A'
    rcall lcd_data
    ldi r24, 'R'
    rcall lcd_data

    ldi r24, low(2000)
    ldi r25, high(2000) ; delay 2 sec
    rcall wait_msec
    rjmp done



 GAS_DETECTED:
    cpi r28, 1      ;r28 = 1 open, r28 = 0 close
    breq blink
    ldi r28, 1
    rjmp do_nothing
    blink:
    clr r29
    out PORTB, r29
```

```
    ldi r28, 0
    do_nothing:

    ldi r19, 1
    ;cpi r31, 0x01
    ;breq done
    ldi r31, 0x01
    ldi r24, 'G'
    rcall lcd_data
    ldi r24, 'A'
    rcall lcd_data
    ldi r24, 'S'
    rcall lcd_data
    ldi r24, ' '
    rcall lcd_data
    ldi r24, 'D'
    rcall lcd_data
    ldi r24, 'E'
    rcall lcd_data
    ldi r24, 'T'
    rcall lcd_data
    ldi r24, 'E'
    rcall lcd_data
    ldi r24, 'C'
    rcall lcd_data
    ldi r24, 'T'
    rcall lcd_data
    ldi r24, 'E'
    rcall lcd_data
    ldi r24, 'D'
    rcall lcd_data

    ldi r24, low(2000)
    ldi r25, high(2000) ; delay 2 sec
    rcall wait_msec




done:

    pop r25
    pop r24


    reti

wait_msec:                  ;delay routine used in previous exercises too

ldi r23, 249

loop_inn:
```

```
dec r23
nop
brne loop_inn
sbiw r24, 1
brne wait_msec
ret
```

Κώδικας σε C :

```
#define F_CPU 16000000UL
#include "avr/io.h"
#include "util/delay.h"


unsigned char duty;
unsigned char i,a,p,z,q,d;
unsigned char x,y, gas, lights, camera;
unsigned char temp,a1,a2,a3;

void write_2_nibbles(unsigned char q){
    int temp = q;   // r24 = temp = q
    int e = PIND;   //in r25, PIND
    e = e & 0x0f;    // andi r25, 0x0f
    temp = temp & 0xf0; // andi r24, 0xf0
    temp = temp | e;    // add r24, r25
    PORTD= temp;  //out PORTD ,r24
    PORTD |= 0x08;  //sbi PORTD ,3
    _delay_ms(1);
    PORTD &= 0b11110111; //cbi PORTD ,3
    _delay_ms(1);

    temp = q;   //pop r24
    temp = temp<<4;  //swap r24
    temp = temp & 0xf0;  //andi r24, 0xf0
    temp = temp | e;   //add r24,r25
    PORTD= temp;   //portd, r24
    PORTD |= 0x08;
    _delay_ms(1);
    PORTD &= 0b11110111;
    _delay_ms(2);

return;
}
```

```
void lcd_data(unsigned char p){
   PORTD |= 0x04;
   write_2_nibbles(p);
   _delay_ms(10);
   return;
}

void lcd_command(unsigned char z){
   PORTD &= 0b11111011;
   write_2_nibbles(z);
   _delay_ms(10);
   return;
}


void lcd_init(){
       _delay_ms(40); //wait_msec

       PORTD = 0x30;     //PORTD = r24 = 0x30
       PORTD |= 0x08;   // sbi PORTD, 3
       _delay_ms(1);
       PORTD &= 0b11110111; //cbi PORTD, 3
       _delay_ms(2);

       PORTD = 0x30;   //PORTD = r24 = 0x30
       PORTD |= 0x08;
       _delay_ms(1);
       PORTD &= 0b11110111;
       _delay_ms(2);
       PORTD = 0x20;
       PORTD |= 0x08;
       _delay_ms(1);
       PORTD &= 0b11110111;
       _delay_ms(2);
       lcd_command(0x28);
       lcd_command(0x0c);
       lcd_command(0x01);

       _delay_ms(10);

       lcd_command(0x06);

       _delay_ms(10);

       return;
}

int gas_detected(){
   lights = 1;

  _delay_ms(10);
```

```
        lcd_data('G');
       //  _delay_ms(2);
        lcd_data('A');
       //  _delay_ms(2);
        lcd_data('S');
       // _delay_ms(2);
        lcd_data(' ');
       // _delay_ms(2);
        lcd_data('D');
       //  _delay_ms(2);
        lcd_data('E');
        //_delay_ms(2);
        lcd_data('T');
        //_delay_ms(2);
        lcd_data('E');
       // _delay_ms(2);
        lcd_data('C');
       // _delay_ms(2);
        lcd_data('T');
        //_delay_ms(2);
        lcd_data('E');
       // _delay_ms(2);
        lcd_data('D');
        //_delay_ms(2);

        _delay_ms(20);
        return 0;


}

int clear(){
    lights = 0;
        _delay_ms(10);
        lcd_data('C');
       //  _delay_ms(2);
        lcd_data('L');
       //  _delay_ms(2);
        lcd_data('E');
       //  _delay_ms(2);
        lcd_data('A');
       // _delay_ms(2);
        lcd_data('R');

        _delay_ms(20);
        return 0;


}


int metatroph(){
   //lights = 0;
    z = ADCL;
```

```
    y = ADCH;

    z = z & 0b11000000;
   // a1 = a2 = a3 = 0;

    if(y >= 220){
        PORTB = 0b00100000;
    }
    else if(y >= 160){
        PORTB = 0b00010000;
    }
    else if(y >= 80){
        PORTB = 0b00001000;
    }
    else if(y >= 40){
        PORTB = 0b00000100;
    }
    else if(y >= 20){
        PORTB = 0b00000010;
    }
    else{
        PORTB = 0b00000001;
    }


    if(y >= 51){

        gas_detected();
    }
    else{
        z = z >> 6;
        y = y << 2;
        y = y | z;
        if(y >= 204){

            gas_detected();
        }
        else{

            clear();
        }
    }


    return 0;
}

int main(){

DDRD |=  0b11111111;
DDRB |=  0b11111111;
DDRC |=  0b00000000;

ADMUX = 0b01100011;
```

```
ADCSRA = 0b10000111;




while(1){
   lcd_init();



    ADCSRA |= (1 << ADSC);
   // _delay_ms(10);
   while(ADCSRA & 0b01000000){

   }
    metatroph();

    _delay_ms(10);

   if(lights == 1){
      if(camera == 1){
         camera = 0;
          PORTB = 0b00000000;
      }
      else{
         camera = 1;
      }
   }
    _delay_ms(100);

   }
}
```

Ζήτημα 4.3 :

Στους παρακάτω κώδικες ελέγχουμε εντός της main πιο κουμπί έχουμε πατήσει και κάνουμε συνέχει loop μέχρι να αφήσουμε το κουμπί.

Ανάλογα πιο κουμπί έχουμε πατημένο θέτουμε την τιμή του OCR1A στο κατάλληλο ποσοστό του ICR1 (=50 για να έχουμε την επιθυμητή συχνότητα με το prescaler μας). Μέσα στο loop για το κάθε κουμπί καλούμε την συνάρτηση metatroph εντός της οποίας

κάνουμε μετατροπή της ADC με τον ίδιο τρόπο με την 1 και αφού τυπώσουμε το ποσοστό που θέλουμε αλλάζουμε την γραμμή με την εντολή:

> lcd_command(0b11000000);

και τυπώνουμε το X,YZ.

Κώδικας C :

```
#define F_CPU 16000000UL
#include "avr/io.h"
#include "util/delay.h"


unsigned char duty;
unsigned char i,a,p,z,q,d,b;
unsigned char x,y;
unsigned char temp,a1,a2,a3;

void write_2_nibbles(unsigned char q){
    int temp = q;   // r24 = temp = q
    int e = PIND;   //in r25, PIND
    e = e & 0x0f;    // andi r25, 0x0f
    temp = temp & 0xf0; // andi r24, 0xf0
    temp = temp + e;    // add r24, r25
    PORTD= temp;  //out PORTD ,r24
    PORTD |= 0x08;  //sbi PORTD ,3
    _delay_ms(1);
    PORTD &= 0b11110111; //cbi PORTD ,3
    _delay_ms(1);

    temp = q;   //pop r24
    temp = temp<<4;  //swap r24
    temp = temp & 0xf0;  //andi r24, 0xf0
    temp = temp + e;   //add r24,r25
    PORTD= temp;   //portd, r24
    PORTD |= 0x08;
    _delay_ms(1);
    PORTD &= 0b11110111;
    _delay_ms(2);

return;
}


void lcd_data(unsigned char p){
    PORTD |= 0x04;
    write_2_nibbles(p);
    _delay_ms(10);
```

```
    return;
}

void lcd_command(unsigned char z){
    PORTD &= 0b11111011;
    write_2_nibbles(z);
    _delay_ms(10);
    return;
}


void lcd_init(){

    _delay_ms(40); //wait_msec

        PORTD = 0x30;     //PORTD = r24 = 0x30
        PORTD |= 0x08;   // sbi PORTD, 3
        _delay_ms(1);
        PORTD &= 0b11110111; //cbi PORTD, 3
        _delay_ms(2);

         PORTD = 0x30;     //PORTD = r24 = 0x30
        PORTD |= 0x08;   // sbi PORTD, 3
        _delay_ms(1);
        PORTD &= 0b11110111; //cbi PORTD, 3
        _delay_ms(2);

        PORTD = 0x30;   //PORTD = r24 = 0x30
        PORTD |= 0x08;
        _delay_ms(1);
        PORTD &= 0b11110111;
        _delay_ms(2);
        PORTD = 0x20;
        PORTD |= 0x08;
        _delay_ms(1);
        PORTD &= 0b11110111;
        _delay_ms(2);
        lcd_command(0x28);
        lcd_command(0x0c);
        lcd_command(0x01);

        _delay_ms(10);

        lcd_command(0x06);

        _delay_ms(10);

        return;
}


void metatroph(unsigned char d){
    z = ADCL;
    y = ADCH;
```

```
z = z & 0b11000000;
a1 = a2 = a3 = 0;
lcd_init();

while(1){
    if(y >= 51){
        a1 += 1;
        y -= 51;
    }
    else break;
}

z = z >> 6;
y = y << 2;
y = y | z;

while(1){
    if(y >= 200){
        a1 += 1;
        y -= 200;
    }
    else break;
}

while(1){
    if(y >= 20){
        a2 += 1;
        y -= 20;
    }
    else break;
}

while(1){
    if(y >= 2){
        a3 += 1;
        y -= 2;
    }
    else break;
}

//b = a1*100 +a2*10+ a3*1;




d = d | 0b00110000;
lcd_data(d);
lcd_data('0');
lcd_data('%');

lcd_command(0b11000000);

a1 = a1 | 0b00110000;
a2 = a2 | 0b00110000;
```

```
    a3 = a3 | 0b00110000;
    lcd_data(a1);
    lcd_data(',');
    lcd_data(a2);
    lcd_data(a3);
    _delay_ms(500);



    return;
}

int main(){



TCCR1A = (1 << WGM11) | (0 << WGM10)| (1 << COM1A1);
TCCR1B = (1 << WGM12) | (1 << WGM13) | (1 << CS11) |(1<< CS10);


ADMUX =0b01100001;


ADCSRA =0b10000111;


DDRD |=  0b11111111;
DDRB |=  0b00000010;
DDRC |=  0b00000000;


duty = 128;
OCR1AL = duty;
PORTD = 0;
lcd_init();
lcd_init();
lcd_init();

while(1){
    lcd_init();
    _delay_ms(2);

    x = ~PINB;
    ICR1 = 0;
   // TCCR1A = (0 << WGM11) | (0 << WGM10)| (0 << COM1A1);

    if( x & 0x04){
        TCCR1A = (1 << WGM11) | (0 << WGM10)| (1 << COM1A1);
        TCNT1=0;            /////

       while( x & 0x04 ){       //////
```

```
   OCR1AL = 10;
   ICR1 = 50;
   a = 0b00110010;

   ADCSRA |= (1 << ADSC);
// _delay_ms(10);
while(ADCSRA & 0b01000000){

}
 metatroph(a);

 _delay_ms(10);


 x = ~PINB;
   }              ////

   }


else if( x & 0x08 ){
    OCR1AL = 20;
    ICR1 = 50;
    a = 0b00110100;

    ADCSRA |= (1 << ADSC);
// _delay_ms(10);
while(ADCSRA & 0b01000000){

}
 metatroph(a);

 _delay_ms(10);

}

else if( x & 0x10 ){
   OCR1AL = 30;
   ICR1 = 50;
   a = 0b00110110;

   ADCSRA |= (1 << ADSC);
// _delay_ms(10);
while(ADCSRA & 0b01000000){

}
 metatroph(a);

 _delay_ms(10);

}

else if( x & 0x20 ){
      TCCR1A = (1 << WGM11) | (0 << WGM10)| (1 << COM1A1);
```

```
      TCNT1=0;             /////

      while( x & 0x20 ){        //////

      OCR1AL = 40;
      ICR1 = 50;
      a = 0b00111000;

      ADCSRA |= (1 << ADSC);
    // _delay_ms(10);
    while(ADCSRA & 0b01000000){

    }
     metatroph(a);

     _delay_ms(10);
        x = ~PINB;
      }              ////

    }

    else{
      //  TCCR1A = (0 << WGM11) | (0 << WGM10)| (0 << COM1A1);
    OCR1AL = 0;
    ICR1 = 0;
    a = 0b00110000;

    ADCSRA |= (1 << ADSC);
    // _delay_ms(10);
    while(ADCSRA & 0b01000000){

    }
     metatroph(a);

     _delay_ms(10);


    }



    }
}
```