3η Εργαστηριακή Αναφορά

Παπαδόπουλος Χαράλαμπος 03120199

Στρίφτης Γεώργιος 03121200

Άσκηση 1

Σκοπός της άσκησης είναι να εξοικειωθούμε με τους χρονιστές που παρέχονται από τον ATmega328PB και η παραγωγή PWM κυματομορφής στον ακροδέκτη PB1 με προσαρμοζόμενο Duty Cycle. Παρακάτω παρατίθεται ο κώδικας:

```
.def counter = r16                              ; for delay
.equ freq = 16
.def DC_VALUE = r19


.org 0x00
   rjmp setup

DC_TABLE:
   .db 8, 28, 48, 68, 88, 108, 128, 148, 168, 188, 208, 228, 248, 248

setup:
   clr DC_VALUE
   sei
   ldi r16, (1<<TOIE1)
   sts TIMSK1, r16

   ldi r16, (1<<CS10) | (1<<WGM12)       ; fpwm = 62500 = 16.000.000/(N * 265) -> N = 1
   sts TCCR1B, r16

   ldi r16, (1<<WGM10) | (1<<COM1A1)
   sts TCCR1A, r16


   ldi r30, low(DC_TABLE*2)           ; Load Z register (ZL) with low byte of array address
   ldi r31, high(DC_TABLE*2)          ; Load Z register (ZH) with high byte of array address
   adiw ZL, 6
   lpm DC_VALUE, Z
   sts OCR1AL, DC_VALUE                     ; Set OCR1A value


   ser r18
   out DDRB, r18                            ; Set PORTB as output


   clr r16
```

```
    out DDRD, r16                   ; PORTD input
    ser r16                         ; Enable pull-up resistors
    out portd, r16


main:
    in r17, PIND
    sbrs r17, 3                     ; check if PD3 is pressed
    rjmp increment                  ; if yes, go to control lights
    sbrs r17, 4
    rjmp decrement
    rjmp main

increment:
    ldi r24, low(100)
    ldi r25, high(100)
    rcall wait_x_ms
    lpm DC_VALUE, Z                 ; Load next duty cycle value
    cpi DC_VALUE, 248
    breq main
    adiw ZL, 1
    sts OCR1AL, DC_VALUE            ; Set OCR1A value
    rjmp main

decrement:
    ldi r24, low(100)
    ldi r25, high(100)
    rcall wait_x_ms
    lpm DC_VALUE, Z
    cpi DC_VALUE, 8
    breq main
    sbiw ZL, 1
    sts OCR1AL, DC_VALUE
    rjmp main
```

Παραπάνω φαίνεται ότι κάναμε χρήση ενός πίνακα για να αποθηκεύσουμε τις duty cycle τιμές από την ελάχιστη στη μέγιστη με 8% αύξηση στην κάθε τιμή. Έπειτα αρχικοποιούμε τα αντίστοιχα flags στους καταχωρητές TIMSK1, TCCR1B και TCCR1A.

Αρχικά βάζουμε το 50% της μέγιστης τιμής DC στον OCR1A και μετά ελέγχουμε συνέχεια εάν πατιέται το PD3 και το PD4. Στην πρώτη περίπτωση αυξάνουμε το DC κατά 8% ενώ στη δεύτερη το μειώνουμε.

Άσκηση 2

Σκοπός της άσκησης είναι να επεκτείνουμε το παραπάνω πρόγραμμα στο οποίο η έξοδος του φίλτρου PB1_PWM θα διαβάζεται από τον ADC και εμείς κάνοντας 16 διαδοχικές μετρήσεις(ανά 100 ms) θα βγάλουμε έναν μέσο όρο και ανάλογα με την μέση τιμή θα ανάβουν τα αντίστοιχα λαμπάκια στο PORTD. Παρακάτω φαίνεται ο κώδικας σε C:

```
int DC_VALUE[13] = {8, 28, 48, 68, 88, 108, 128, 148, 168, 188, 208, 228, 248};
uint16_t SAMPLES[16] = {0};

void _setup_pwm() {
    TCCR1A = (1<<WGM10) | (1<<COM1A1);
    TCCR1B = (1<<WGM12) | (1<<CS10);
}

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

uint16_t read_adc() {
    ADMUX |= (1 << MUX0);
    ADCSRA |= (1 << ADSC);
    while (ADCSRA & (1 << ADSC));
    return ADC;
}

void update_leds(uint16_t adc_value) {
    PORTD = 0x00;
    if (adc_value <= 200) {
        PORTD |= (1 << PD0);
    } else if (adc_value <= 400) {
        PORTD |= (1 << PD1);
    } else if (adc_value <= 600) {
        PORTD |= (1 << PD2);
    } else if (adc_value <= 800) {
        PORTD |= (1 << PD3);
    } else {
        PORTD |= (1 << PD4);
    }
}

int main() {
    _setup_pwm();
    _setup_adc();
    int index = 6;
    uint16_t sum = 0;
    int current_sample = 0;
    uint16_t adc_value;

    DDRB |= 0b11111111;
```

```
    DDRD |= 0b00111111;

    OCR1A = DC_VALUE[index];

    uint16_t elapsed_time = 0; // Timer variable for elapsed time

    while (1) {
        if (!(PIND & (1 << PIND6))) {
            if (index < 12) {
                index++;
                OCR1A = DC_VALUE[index];
                _delay_ms(100);
            }
        }
        if (!(PIND & (1 << PIND7))) {
            if (index > 0) {
                index--;
                OCR1A = DC_VALUE[index];
                _delay_ms(100);
            }
        }

        adc_value = read_adc();

        // Update samples and sum
        sum -= SAMPLES[current_sample];
        SAMPLES[current_sample] = adc_value;
        sum += adc_value;

        current_sample++;
        if (current_sample == 16) {
            current_sample = 0; // Reset sample index
        }

        // Increment elapsed time
        elapsed_time += 100; // Since we have a delay of 100 ms
        if (elapsed_time >= 1600) { // Check if 1.6 seconds have passed
            uint16_t average_adc_value = sum >> 4; // Calculate average
            update_leds(average_adc_value);
            elapsed_time = 0; // Reset elapsed time
        }

        _delay_ms(100);
    }
}
```

Αρχικοποιούμε ανάλογα τους καταχωρητές ADMUX, ADSCRA για τη σωστή λειτουργία του ADC και χρησιμοποιούμε τα πλήκτρα PD6 και PD7 για να αυξήσουμε και να μειώσουμε το DC αντίστοιχα. Έπειτα κάθε 100 ms ελέγχουμε εάν πατιέται κάποιο από τα παραπάνω πλήκτρα και στη συνέχεια παίρνουμε δείγμα από τον ADC και ανάλογα με το μέσο όρο μετά από 1.6 sec ανάβουμε τα αντίστοιχα λαμπάκια στο PD4.

Άσκηση 3

Σκοπός της άσκησης είναι η περαιτέρω επέκτασης του κώδικα της άσκησης 1 προσθέτοντας μία επιπλέον λειτουργία. Πιο συγκεκριμένα, θα έχει 2 modes. Στο πρώτο θα χρησιμοποιούμε τα PD1, PD2 για αύξηση και μείωση του DC και στο δεύτερο θα χρησιμοποιούμε το ποτενσιόμετρο για τη ρύθμιση του DC. Τα modes επιλέγονται από τα PD6(mode 1) και PD6(mode 2). Παρατίθεται παρακάτω ο κώδικας σε C:

```
void _setup_pwm(){
    TCCR1A = (1<<WGM10) | (1<<COM1A1);
    TCCR1B = (1<<WGM12) | (1<<CS10);
}

void _setup_adc(){
    ADMUX = 0b01000000;                                  // MUX = 0 -> POT1
    ADCSRA = 0b10000111;
}

uint16_t read_adc() {
    ADCSRA |= (1 << ADSC);
    while (ADCSRA & (1 << ADSC));                                  //wait for the flag to clear
    return ADC;
}

int main() {
    _setup_pwm();
    _setup_adc();
    index = 6;


    DDRB |= 0b11111111;

    DDRD |= 0b00111001;

    OCR1A = DC_VALUE[index];  // Set initial duty cycle

    while(1) {
        if (!(PIND & (1 << PIND6))) {
            mode = 1;
            _delay_ms(100);
        }
        if (!(PIND & (1 << PIND7))) {
            mode = 2;
            _delay_ms(100);
        }


        if (mode == 1) {
            if (!(PIND & (1 << PIND1))) {
                if (index < 12) {
                    index++;
```

```
                OCR1A = DC_VALUE[index];
            }
            _delay_ms(100);
        }
        if (!(PIND & (1 << PIND2))) {
            if (index > 0) {
                index--;
                OCR1A = DC_VALUE[index];
            }
            _delay_ms(100);
        }
    }

    if (mode == 2) {
        uint16_t adc_value = read_adc();
        index = (adc_value * 12) / 1023;  /*since we have 10bits(1024) we need to scale
for index * /
        OCR1A = DC_VALUE[index];
    }

    _delay_ms(100);
  }
}
```

Διαιρούμε την τιμή του ADC με το 1023 και το πολλαπλασιάζουμε με το 12 για να αυξήσουμε ή μειώσουμε ανάλογα το DC.